ADVANCED ELECTRONIC SYSTEMS

QUALITY PRODUCTS FROM A PROFESSIONAL COMPANY

ALS-SDA-8085-MEL
USER'S MANUAL

PIC EVALUATION BOARD PIC-03

89C51ED2 PROJECT BOARD

89C51ED2 EVALUATION BOARD

FPGA KIT

ADAPTIVE DELTA MODULATION KIT

ARM-2148 PLUS

8086 MEL

8085ME

INTERFACES

FLASH 89 C51ED2 BOARD

DIGITAL IO CARD

CONTENTS

CHAPTER 1 GENERAL INFORMATION

CHAPTER 2 INSTALLATION

CHAPTER 3 HARDWARE DESCRIPTION

CHAPTER 4 OPERATING INSTRUCTIONS

CHAPTER 1

1.1 INTRODUCTION

This manual provides all the necessary information for the installation, operation and routine maintenance of the SDA-85-MEL.

1.2 SPECIFICATIONS

CPU	8085 operating at 3.072 MHz
MEMORY	**EPROM:** Two JEDEC compatible 28-pin sockets provide up to 16/32K bytes using 2x 27128/27256. **RAM:** One JEDEC compatible 28-pin sockets provide up to 8/32K bytes of CMOS static RAM using 6264/62256; provision to battery back the entire system RAM. **NOTE:** The system is supplied with, 16K bytes of monitor EPROM and 8K bytes of RAM
PARALLEL I/O	48 I/O lines using PPI (2 X 8255). PPI (8255) signals are terminated in 2x26-pin berg stick headers-**P2 & P3** and the pin out is compatible with our range of educational interface modules.
SERIAL I/O	One RS-232C compatible interface, using USART-8251A with programmable baud rates through the 8253 programmable timer. Serial I/O lines are terminated in a standard 'DB9' female connector-**P1**
TIMER	Three 16-bit counter/timer using 8253 programmable timer. Counter 1 is used for serial I/O baud rate generation. Timer lines and interrupt lines are terminated in 26 pin berg strip header-**P5**
INTERRUPT	RST7.5 is used to provide keyboard VECT-INTR function. TRAP is used for single instruction step function RST5.5 is used for keyboard operation. RST6.5 and INTR are available to user. RST 7.5 and TRAP are available to the user if VECT-INTR & SINGLE STEP functions are disabled. **NOTE:** ADDITIONAL FEATURE ADDED IN 85MEL One priority interrupt controller using 8259A provides interrupts vectors for 8 external sources. IR0 & IR1 are jumper selectable.
KEYBOARD	A computer keyboard has to be interfaced
DISPLAY	16 characters and 2-line LCD display is mounted on the 85MEL
BUS SIGNALS	All address, data and control signals are terminated in a 50-pin berg stick header-**P6** for user expansion
MONITOR SOFTWARE	16K bytes of monitor (27128A) that allows programs entry, verification, debugging and execution from the system keyboard including line assembler/ disassembler in serial mode

KEYBOARD COMMANDS

- # RESET
- # VECT-INTR function
- # Substitute memory
- # Examine registers
- # Single instruction step
- # NEXT/ PREV memory contents
- # INSERT bytes into user program
- # Delete bytes from user program
- # Memory fill
- # EPROM read/ program
- # Assemble/Disassemble

SERIAL COMMANDS

All the above commands except Assemble/ Disassemble and Receive/ transmit Intel-hex record.

Installation procedure for SDA-85-MEL Microprocessor trainer kit

- Open the kits cover of the trainer kit; visually check for all components are properly mounted on the board.
- If not, press all the components once. Check for the default jumper setting correctly according to the component layout and default jumper connections. (Refer the sheet in the manual).
- Connect power cable to the power supply with correct polarity. If the power cable is already connected to the power supply then go to the next step.
- Connect the power connector to the kit (9 pin connector-**P4**).
- Connect computer keyboard to the connector marked as **USB connector**.
- Switch on the power supply.
- A message "**ALS SDA85-STA**" is displayed on the LCD display of the kit.
- Now the system is ready to take in any valid commands from the keyboard.
- Refer to chapter 6 in the manual for some test programs.
- For serial communication, refer chapter 5.

2.4 RS-232C CABLE CONNECTIONS

The RS232C cable connections required for establishing communication between SDA-85-MEL and computer system is given below.

85MEL		COMPUTER	
PIN NO	SIGNAL	PIN NO	SIGNAL
3	TXD	2	RXD
2	RXD	3	TXD
4	DTR	6	DSR
6	DSR	4	DTR
5	GND	5	GND

NOTE:

- The RS-232C interconnection cable requires a male DB 9 pin "D" type connector-P1 at the SDA-85-MEL end.
- If the computer does support the handshake protocol, the RTS and DTR O/P lines should be connected to CTS and DSR I/P lines respectively at both the ends.

CHAPTER 2

INSTALLATION

SDA-85-MEL is a compact, versatile, low cost 8085 microprocessor trainer. On unpacking the system, if no visible damages are noticed apply power through the 9-pin **'D'** type male connector-P4 provided for the purpose. The pin details and cable colour coding are given below.

2.1 POWER CONNECTOR P4 PIN DETAILS

PIN NO	DESCRIPTION	COLOUR CODE
4,5	GND	BLACK
9	+5V	BLUE/ORANGE/WHITE

2.2 KEYBOARD OPERATION

For keyboard operation, pins 4 & 5 are connected to GND and pin 9 is connected to +5V of the power supply. As soon as +5V and GND are connected and the power supply is switched on, the sign-on message **"ALS SDA85-STA"** should appear on the display. All the monitor commands explained in chapter 4 will then become active.

While making connections to the bus expansion connectors and I/O connector's ensure that the cables have their first pins, (normally red color) properly aligned with respect to the pin 1 marking on the PCB.

2.3 SERIAL OPERATION

To connect the SDA 85-MEL to a host computer system through its serial port, the procedure is as follows:

- Make the power supply connections as given in and switch on the power. On **POWER ON**, the SDA-85-MEL will be in keyboard mode of operation.
- Refer the section 4.5 for the initializations of baud rate at the SDA-85-MEL. Connect the RS-232C cable from the computer to the serial port connector DB 9 pin straight, female) –P1 on the SDA-85-MEL kit.
- Set computers serial interface for correct baud rate, data length & parity.
- Press the **RESET** key and then **'E'** key followed by the **"0"** key to transfer control to the computer. The prompt character **">"** will be displayed on the computer monitor screen indicating that it is ready for command character input. You may execute any of the commands that are described in detail in chapter 4.3.1.

Incase if there is no response from the SDA-85-MEL then,

- Check the RS-232C cable connections at both ends.
- If a message does appear but is garbled, check the data length, parity and baud rate setting and try again.
- If a computer system is the controlling device, check that the driver program is running (see chapter 5), the RS-232C cable is connected to the correct COM port, and that port is working.

CHAPTER 3

3.1 GENERAL DESCRIPTION

The SDA-85-MEL is compact, versatile, low-cost microprocessor trainer consisting of the 8085 CPU, bus buffers, provision for 64K memory (ROM and RAM), timer, parallel I/O, serial I/O, computer keyboard, 16X2 line display, optional EPROM programmer interface and provision for battery back-up for RAM. A powerful monitor in 16K bytes of EPROM allows program entry, verification and execution and debugging; line assembly and disassembly; transfer of files to and from a PC; EPROM programmer interface routines etc.

3.2 JUMPER PIN DETAILS

The jumper details are explained below:

SL NO	JUMPER NO	DESCRIPTION
1	JP1(1-2) (2-3)	Provides VCC to 26 pin connector (P2) GND
2	JP2(1-2) (2-3)	Connects to the IR1 interrupt i/p pin 19 of 8259A externally from 26-pin connector (P5 pin 10) Connects to the IR1 interrupt i/p pin 19 of 8259A from 8255 port line 2 PC3 (P3/4)
3	JP3(1-2) (2-3)	Connects to the IR0 interrupt i/p pin 18 of 8259A externally from 26-pin connector (P5/8) Connects to the IR0 interrupt i/p pin 18 of 8259A from 8255 port line PC3 (P2/4)
4	JP4	Connects port line (2 PC3 P3/4) to the connector P5 pin 14.
5	JP7	Connects to RST6.5 interrupts i/p of CPU through inverter from 8251.
6	JP9 (1-2) (3-4) (5-6)	Connects to 8253 clock 0 input Connects to 8252 clock 1 input Connects to 8253 output signal (out 0)
7	JP10 (1-2) (2-3)	 Connects GND to INTR line pin 10 of CPU Connects to INTR interrupt i/p to CPU from 8259A (pin 17)
8	JP11	Connects to INTR interrupt i/p line through inverter
9	JP12(SHORT) (OPEN)	Selects 16K EPROM memory (0-3FFFH) Selects 32K EPROM memory (0-7FFFh)
10	JP13(SHORT) (OPEN)	Connects to the TRAP interrupt i/p pin 6 of CPU through inverter for single step function Provides external clock signal (BCLK) from the bus connector Pin 32 through inverter to the CPU
11	JP14(SHORT) (OPEN)	Selects address lines A14(pin 27) for 27256 Selects 27128
12	JP15(1-2) (2-3)	Selects address line A13 (pin16) for 62256 RAM Selects 6264 RAM
13	JP16(SHORT) (OPEN)	Provides battery i/p to the circuit Disables battery backup
14	SW2(1-2) (2-3)	Connects to RST7.5 interrupt i/p of CPU externally from bus expansion connector Provides hardware interrupt using vect int key on the keyboard

With reference to the component layout the default jumper settings are:

SL.NO	JUMPERS	JUMPER SETTING
1	JP2 (1-2)	CLOSED
2	JP3 (1-2)	CLOSED
3	JP4	CLOSED
4	JP7	CLOSED
5	JP9 (1-2) JP9 (3-4) JP9 (5-6)	CLOSED CLOSED CLOSED
6	JP10 (2-3)	CLOSED
7	JP12	CLOSED
8	JP13	CLOSED
9	JP15 (2-3)	CLOSED
10	SW1 (2-3)	CLOSED

3.3 The use of different jumper pins is explained below.

3.3.1 INTERRUPTS

The CPU interrupts pins INTR, RST5.5, RST6.5, RST7.5 and TRAP can be connected to external devices through the bus expansion connector or on card I/O devices as given below.

- ⇒ Shorting JP13 (1-2), JP9 (5-6) in 85MEL the trap pin is used for the single step function.
- ⇒ Shorting JP10 (pins 2 & 3), the INTR pin is connected to 8259 INT pin (pin 17)
- ⇒ Shorting JP11 **ITR*** line is connected to CPU INTR through inverter.
- ⇒ Shorting JP7 the RXRDY pin of USART 8251A can be connected to RST6.5 to indicate availability of a received character from the serial port.
- ⇒ Shorting SW2 (pins 2 & 3 0, the RST7.5 pin is connected to the onboard VECT INTR key.
- ⇒ Shorting SW2 (pins 1 & 2), the RST7.5 pin is connected to expansion connector P6/36.

NOTE:
In the case of jumpers if the shorts are not made, the interrupt pins of the CPU are automatically connected, through inverters, as shown, to the expansion connector. The inverters ensure that the interrupt pins of the CPU are at a low level, even if no external connections are made.

Port lines PC3 of IC's 8255 are brought to expansion connectors. These lines provide PPI interrupt capability during operation in mode 1 and mode 2.

3.3.2 MEMORY ADDRESS DECODING FOR 85MEL

JUMPER PIN	27128 (U29)	27128 (U27)	6264 (U22)
JP14 OPEN	0-3FFFH	4000-7FFFH	E000-FFFFH
JP15 CLOSE (2-3)			
JP12 CLOSE			
JUMPER PIN	**27256 (U29)**		**62256 (U22)**
JP14 OPEN	0-7FFFH		8000-FFFFH
JP15 CLOSE(1-2)			
JP12 OPEN			

3.3.3 I/O ADDRESS DECODING

I/O DEVICE	ADDRESS
Timer, 8253 (U11)	C8 TIMER 0 C9 TIMER 1 CA TIMER 2 CB CONTROL
PPI,8255 (U4)	D8 PORT A D9 PORT B DA PORT C DB CONTROL
PPI,8255 (U3)	F0 PORT A F1 PORT B F2 PORT C F3 CONTROL
8251 (U6)	C0 DATA C1 CONTROL
8259 (U12)	F8 CONTROL F9 DATA

3.3.4 PROGRAMMABLE TIMER 8253

1. Timer 0 of the 8253 has been used for the single step function. To enable this short JP9 (1-2).
2. Timer 1 of the 8253 has been used on card for generation of the TXD and RXD baud clock required by USART (8251A). To enable this short JP9 (3-4).

NOTE: That all timer gate, clock and out lines are terminated at 26 pin connector P5.

3.3.5 SERIAL I/O 8251A

The user can operate the serial mode using connector P1 "DB9", connecting RS232C cable between the computer and the trainer.

3.3.6 BATTERY BACK-UP FOR RAM

By shorting JP16 RAM U22 can be battery backed.

3.4 THE CONNECTOR PIN DETAILS ARE GIVEN BELOW.

3.4.1 BUS EXPANSION CONNECTOR-P6 (50 PIN)

PIN NO	DESCRIPTION	PIN NO	DESCRIPTION
1	GND	2	GND
3	Address line A0	4	Address line A1
5	Address line A2	6	Address line A3
7	Address line A4	8	Address line A5
9	Address line A6	10	Address line A7
11	Address line A8	12	Address line A9
13	Address line A10	14	Address line A11
15	Address line A12	16	Address line A13
17	Address line A14	18	Address line A15
19	CPU status line, S1	20	CPU SOD line

21	INTA line I/P pin 2	22	CPU status line, S0
23	Data line D0	24	Data line D1
25	Data line D2	26	Data line D3
27	Data line D4	28	Data line D5
29	Data line D6	30	Data line D7
31	CPU SID line	32	Bus clock to inverter (U28A pin 2)
33	INTR interrupt I/P to inverter (U24D pins 12 & 13)	34	RST6.5 interrupt I/P to inverter (U28A pin 4)
35	RST5.5 interrupt to inverter pin 6 of U28A	36	RST7.5 (U28A pin 8)
37	Hold signal inverter (U24C pins 9 & 10)	38	HLDA line to CPU pin 38 of U31
39	VCC=5V	40	VCC=5V
41	RD line	42	WR line
43	NC	44	IO/M line
45	ALE line	46	Ready line
47	GND	48	RESET signal
49	GND	50	CLK signal

3.4.2 AUXILIARY CONNECTOR-P5 (26 PIN)

PIN NO	DESCRIPTION	PIN NO	DESCRIPTION
1	Gate0 i/p of timer 8253	2	Gate1 i/p of timer 8253
3	Gate2 i/p of timer 8253	4	OUT2 o/p of timer 8253
5	CLK2 o/p of timer 8253	6	OUT1 o/p of timer 8253
7	OUT0 o/p of timer 8253	8	I/P to 8259 through JP3 (IRQ0) U12/18
9	CLK1 i/p of timer 8253	10	I/P to 8259 through JP3 (IRQ1) U12/19
11	CLK0 i/p of timer 8253	12	GND
13	NC	14	Through JP4 to inverter U8 pin 10 (PC3)
15,25	GND	16,26	NC
17	TXD of USART 8251 (U6/19)	18	TXRDY pin of USART 8251 (U6/15)
19	I/P of 8259A (IRQ2) (U12/20)	20	I/P of 8259A (IRQ3) (U12/21)
21	I/P of 8259A (IRQ4) (U12/22)	22	I/P of 8259A (IRQ5) (U12/23)
23	I/P of 8259A (IRQ6) (U12/24)	24	I/P of 8259A (IRQ7) (U12/25)

3.4.3 I/O CONNECTOR P2, P3 (8255) PIN DETAILS – (26 PIN)

PIN NO	DESCRIPTION	PIN NO	DESCRIPTION
1	Port line PC4, IC pin 13	2	Port line PC5, IC pin 12
3	Port line PC2, IC pin 16	4	Port line PC3, IC pin 17
5	Port line PC0, IC pin 14	6	Port line PC1, IC pin 15
7	Port line PB6, IC pin 24	8	Port line PB7, IC pin 25
9	Port line PB4, IC pin 22	10	Port line PB5, IC pin 23
11	Port line PB2, IC pin 20	12	Port line PB3, IC pin 21
13	Port line PB0, IC pin 18	14	Port line PB1, IC pin 19
15	Port line PA6, IC pin 38	16	Port line PA7, IC pin 37
17	Port line PA4, IC pin 40	18	Port line PA5, IC pin 39
19	Port line PA2, IC pin 2	20	Port line PA3, IC pin 1
21	Port line PA0, IC pin 4	22	Port line PA1, IC pin 3
23	Port line PC6, IC pin 11	24	Port line PC7, IC pin 10
25	JP1 pin 2 in 85MEL	26	GND

NOTE: INTR signal is driven through inverter U24 coming from P3/25.

3.4.4 SERIAL I/O CONNECTOR-P1

PIN NO	DESCRIPTION
3	RS-232C O/P - buffered TXD line
2	RS-232C I/P - RXD line to buffer
4	RS-232C O/P - buffered DTR line
6	RS-232C I/P - DSR line to buffer
5	GND

3.4.5 POWER CONNECTOR –P4

PIN NO	DESCRIPTION
4,5	GND
9	+5V

3.5 CIRCUIT DESCRIPTION

With reference to the enclosed schematics, the 8085 CPU is connected to on card memory, I/O and bus expansion connector. The CPU interrupt pins are either connected through interrupts, to ensure an active low level.

Address decoder provides memory & I/O address decoding. By suitable selection of jumpers provided, different type of memory devices like 27128, 27256, 6264 and 62256 can be used. The system is supplied with 16K bytes of system monitor EPROM and 8K bytes of RAM.

Oncard RAM sockets are battery backed with the aid of a circuit comprising of transistors. The user can solder NI-CD 3.6V PCB mountable battery.

The programmable timer 8253 lines are terminated in a 26 pin auxiliary connector P5. Timer 0 of the 8253 is used for the single step function and is operated in the interrupt on terminal count mode. This can be disabled when external connections are to be made. Timer 1 of the 8253 is operated in the square-wave mode to provide the TXD/RXD clock for USART. This function can be disabled when not required. All the 3 Timer i/p & o/p lines are terminated to the header.

Programmable peripheral interface (8255) U3 & U4 are terminated in 26-pin headers P2 & P3.

IC U5-8255 interacts with the 16X2 LCD display and computer keyboard. The RS232C is provided through a USART 8251A and interface IC MAX232C.

The clock I/P to the 8251A and the 8253 is bus clock divided by 2 i.e., 1.536MHz. To branch to the serial monitor commands the following initialization of memory is required to select the baud rate.

FFA6H	FFA7H	BAUD RATE
05	00	19200
0A	00	9600
14	00	4800
28	00	2400
50	00	1200
A0	00	600
40	01	300

-5-5
-5 to USB

The RESET key is then pressed followed by E & 0 key to transfer control to a PC (functioning as a terminal emulator) connector to connector DB9. The above values are based on the 1.536 MHz clock i/p to the 8253.

3.6 TROUBLE SHOOTING

The 85MEL is fully tested, prior to dispatch and the system should give the user trouble free service. To ensure this, carefully observe the caution sheet enclosed. On apply of power, the sign-on message does not appear disconnect the power supply and check the following.

⇒ Power supply voltages are of the correct magnitude and polarity.

⇒ All IC's are properly seated in their sockets and the pin 1 positions are as per the component layout.

⇒ The IC part numbers corresponds to those in the component layout.

⇒ All jumper connections are as per the details given in sec.3.2.

⇒ The monitor ROM is in the recommended IC position (U29). If all the above conditions are satisfied, still sign-on does not appear when power is again supplied then observe the clock signal at test point marked as **CLK***. The frequency should be approximately 3MHz. If it is not present, either the CPU or the crystal could be defective or a loose contact could be present.

⇒ If CLK signal is present then check for the ALE at the test point marked as **ALE**. In the same way check for the chip select signals at the test points marked as **/CS** with respect to that socket number. Example to check the chip select of the monitor socket use the test point marked as **/CSROM1**. Check the read and write signals at the test points marked as **/RD** and **/WR.**

⇒ Same as chip select signals all the 3 clocks I/P, O/P & gate signals are brought as the test points near the connector P1. If the CLK signals are present then problem could be because of a defective decoder or a defective monitor EPROM or RAM.

3.7 TEST PROGRAMS

The following test programs can be run to check the various components of the system. The listing of all the following test programs is given at the end of manual.

3.7.1 To test monitor ROM

Procedure:

Press the **G** key on the keyboard. Enter the address 2640<cr>. Enter the start address, end address, followed by the checksum of the EPROM <cr>.

Example:

G 2640<Cr>0000, 3FFF, 12BC<Cr>.

Result:

OK:
MONITOR ROM TEST
CHECKSUM IS OK

The above message will be displayed on the LCD display if the checksum is matched.

ERROR:

An ****ERROR**** will be displayed on the LCD display. One can check the checksum in the location FF0E and FF0FH.

3.7.2 TO TEST THE MEMORY RAM

Procedure:

Press the **G** key on the keyboard. Enter the address 26E0<Cr>. Enter the start address and end address of the RAM<Cr>.

Example:

G 26E0<Cr>E000, FFFE<Cr> (This address is for 6264 RAM IC at the socket U22)

G 26E0<Cr>8000, FFFE<Cr> (This address is for 62256 RAM IC at the socket U22)

Result:

RAM IS TESTED OK

The above message will be displayed on the LCD display if the IC is ok.

ERROR:

An ****ERROR**** will be displayed on the LCD display.

3.7.3 TO TEST KEYBOARD

Procedure:

Press the **G** key on the keyboard. Enter the address 2730<Cr>.

EXAMPLE: G 2730<Cr>

Result:

KEYBOARD TEST
CODE=

The above message will be displayed on the display on the LCD & it will be waiting for a key to be pressed. The corresponding key will be displayed on the LCD display.

3.7.4 To test the LCD on the trainer kit

Procedure: G<2770><Cr>

Result:

WELCOME TO A.L.S TESTING OF LCD DISPLAY will be displayed on the LCD display in a loop.

3.7.5 To test vector interrupt key

Procedure:

Press **G** key on the keyboard and enter the address 27C0H<Cr>. You will see a message, now the kit will be waiting for the vector interrupt key to be pressed.

Example: G<27C0><Cr>

Result:

VECTOR INTERRUPT
KEY TEST

The above message will be displayed & waits for the VECT key to be pressed. Once the interrupt occurs the sign-on message will be displayed. Examine the register A=11, B=22, C=33 will be displayed.

3.7.6 To test PPI 8255's IC

Procedure:

Short the output of the both ICs at P2 & P3 by means of a 26-core flat cable assembly. Press **G** key on the keyboard and enter the address 28C0<Cr>.

Example: G<28C0><Cr>

Result:

OK.

8255 TEST PASS will be displayed on the LCD display if both the ICs are OK.

ERROR:

An ****ERROR**** will be displayed on the LCD display in case of any problem in the 8255 ICs.

3.7.7 To test programmable interrupt Controller 8259

Procedure:

Short the jumper JP2, JP3/1 & 2. Press the G key on the keyboard and enter the address 29D0<Cr>. You will be prompted with a message **"8259 INTR TEST"** on the first line of the LCD display. Momentarily short any external interrupt line to GND. The particular interrupt message will appear on the LCD display. This program will be in loop till you reset the system.

Example: G<29D0><Cr>

Result:

8259 INTR TEST
The above message will be displayed. Momentarily short any one external interrupt line to GND. The particular interrupt message will appear on the LCD display. This program will be in loop until you reset the system. For instant, you have taken the IR32 pin to LOW (P5/20), you will see the message **"8259 INTERRUPT3"** on the LCD display. The message will change for each interrupt lines.

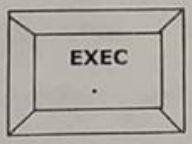
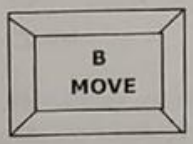
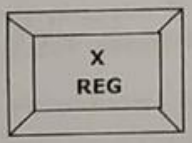
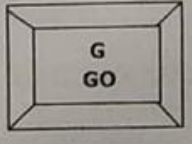
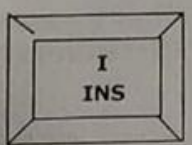
CHAPTER 4

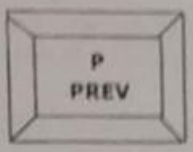
4.1 GENERAL DESCRIPTION

The **System Design Aid** SDA-85MEL is a single board system complete with CPU, Memory and Interrupt Controller Timer & I/O. The system components are mounted on a double-sided glass epoxy PCB with legend printing to provide easy component identification. A computer keyboard provides access to the system monitor enabling entry, verification, debugging and execution. The labeling of the keys clearly indicates their function.

4.2 KEYBOARD MONITOR COMMANDS

The function of the keys in the keyboard is explained and the uses of the monitor commands are covered in detail.

KEY LABEL	DESCRIPTION
	Transfers control to the monitor at location OH key is connected to the RESET input pin of the CPU.
	A user interrupt key connected to the RST7.5 input of the CPU. Transfers control to location FFB1H in RAM if the interrupt is unmasked and interrupts are enabled. **The Vector key is mounted on the kit.**
	The monitor interprets this key as a delimiter. Its use in the different commands and is explained later.
	The command terminator (ENTER KEY)
	Selects the Block Move command
	Selects the Examine/modify CPU register
	Selects the GO command (program execution)
	Selects the Insert data bytes function



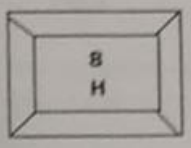
A delimiter key. Its use is explained in the detailed command explanation

Selects the single step function STEP

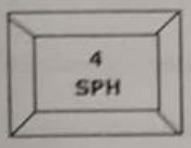
Selects memory Examine/modify function

Selects the Delete Data bytes function

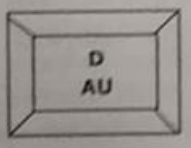
Hex key, the letter C, selects the memory complement command. In combination with Examine Register function, allows display of C register



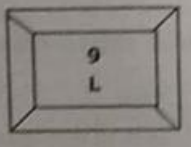
Hex key, the numeral 8. Allows the contents of the H register to be displayed in combination with Examine Register Command.



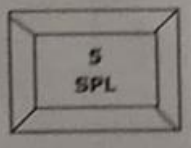
Hex key, the numeral 4, allows the high byte of the stack pointer to be in combination with Examine Register Command.



Hex key, the letter D. selects the auxiliary command set foe EPROM programmer. In combination with the Examine register function, allows display of D register.

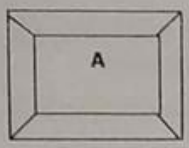


Hex key, the numeral 9. In combination with the Examine Register Function, allows display of the L register contents.

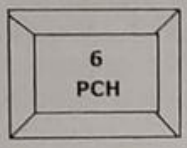


Hex key, the numeral 5, in combination with Examine register function, allows display of the low byte of the stack pointer.

Hex key, the letter E. Selects the CRT mode when pressed after Reset. In combination with the Examine register function, allows display of E register.

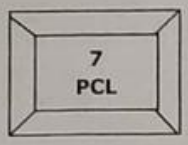


Hex key, the letter A. In combination with the Examine register function, allows display of Accumulator.

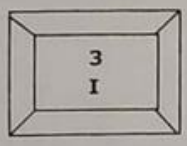


Hex key, the numeral 6. In combination with the Registers allows display of the high byte of the program counter.

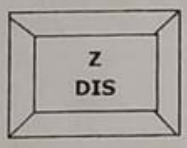
Hex key, the letter F. Selects the memory Fill Command. In combination with the Examine register function, allows display of flog status.



Hex key, the numeral 7. In combination with the Examine Register function allows display of the program counter low byte.



Hex key, the numeral 3. In combination with the Examine Register Function allows display of the Interrupt status.



Selects Disassembler command

NOTE: In the following explanation <, or space bar> is the NEXT key and <Cr> is the EXEC key.

4.2.1 BLOCK MOVE COMMAND

Syntax: **B MOVE<starting source address> <,> <ending source address> <,> <starting destination address><.>**

A block of memory specified by the starting and ending source addresses can be copied to a location specified by the starting destination address. The starting destination address can be less than, equal to or greater than the starting source address. The source region can be in RAM or ROM but the destination region must be in RAM as it is not possible to write into ROM as the move involves a write operation.

EX: The following key sequence will move the memory contents in the region 0000H to 000FH to the region starting at F000H.
BLOCK MOVE 0, F0, F000.

4.2.2 EXAMINE REGISTER COMMAND

Syntax: **EXAM REG <Register Designator> <,> ..... <.>**

It is necessary to know the contents of the CPU registers to develop and debug a user program. The use of this command allows the contents to be displayed in the data field from the save locations in RAM and modified, if required. As mentioned above each register has described by a key, which is its designator. When the command is selected and a register designator is pressed the register name is displayed in the address field

and the contents in the data field. If the NEXT key is then pressed the next register is displayed and goes on until the last register (program counter). The EXEC key will terminate this command.

EX: To display the contents of the H register follows the key sequence given below.

EXAMREG<8> <,>.....<.>

If the L registers contents are now desired press NEXT etc., a new value can be entered if required, in the data field. The PREV key is an invalid terminator.

4.2.3 GO COMMAND

Syntax: **GO <starting program address> <.>**

The execution of user program is possible through the command. When the command is selected the program counter contents, from the save location in RAM, are displayed in the address field and the corresponding contents in the data field. A new address can be entered to specify a new user program starting point. The terminator EXEC causes the monitor to transfer control to the user program. All the CPU registers are updated from their save locations in RAM before execution of the first user instruction. Any terminator other than EXEC is an invalid terminator.

EX: A user program written from location F850 can be executed as given below.

GO F850

The sequence GO, causes the program starting at location specified by the contents of the program counter save location to be executed.

4.2.4 SUBSTITUTE MEMORY COMMAND

Syntax: **M MEM <address of memory location> <,> <new byte if required><,><.>**

To execute a user program it is necessary to have a means whereby the program can be entered into SDA-85-MEL RAM at a specified starting address. This command, when exercised, allow the user to access and modify specified memory locations. Programs when used with this command allow the user to access and modify specified memory locations. Programs are entered and executed in RAM (addresses F800 to FFFFH). The delimiter PREV, when used with this command, allows the memory location previous to the one displayed, to be examined and modified.

EX: If program is to be entered starting at location F870H then proceeds as follows.

KEY	ADDRESS FIELD DISPLAY	DATA FIELD DISPLAY
M		
F	000F	
8	00F8	
7	0087	
0	0F87	
NEXT	F870	32
3	F870	03
E	F870	3E
NEXT	F871	44
1	F871	01
1	F871	11
PREV	F870	3E
<Cr>		

The delimiter EXEC (.) terminates the command and causes a branch to the main command recognizer routine. The contents of ROM locations can be examined, but not modified from the keyboard.

4.2.5 STEP COMMAND

Syntax: **S STEP<start addr of user program><,>...<.>**

To debug a program it is very useful to have a facility where by a program can be executed one instruction at a time. This command provides the user with the necessary software, which in combination with the hardware onboard allows program execution instruction by instruction. The non-mask able TRAP interrupt is used for this purpose. The delimiter NEXT allows a single instruction execution and the delimiter EXEC terminates the command. Whenever the next key is pressed the register contents are updated from their save locations, a single instruction in the user program is executed, the registers are once again saved and the monitor returns to the user with the program counter and the byte that it addresses displayed in the address and data fields respectively.

EX: The key sequences STEP F850, causes the instructions at locations F850H to be executed and since the EXEC (.) key was pressed control is returned to the command recognizer.

4.2.6 INSERT COMMAND

Syntax: **I<start address of the user program><,><end address of the user program><,><address at which data is to be inserted<,><number of bytes to be inserted><,>...<.>**

This is a mini editing command, which allows the user to insert a specified number of bytes into his program at a specified address. Addresses in his program, corresponding to 3-byte memory reference instructions, are altered to take at which the bytes are to be inserted is not altered. The start and end address of user program is to be entered first. Any reference in the user program to an address greater than the user program address is not altered. The bytes are to be inserted in a manner similar to the substitute memory command. The terminator EXEC terminates the command.

NOTE:

- ⇒ When inserting the bytes care should be taken to see that in the portion that is inserted any 3-byte memory reference should have the address altered, to take into account the number of bytes that are being inserted.
- ⇒ The monitor accepts only hex values and the number of bytes to be inserted should be in hex.
- ⇒ As the size of the programs grows when a number of bytes are inserted, the user should ensure that there is sufficient space in RAM beyond the last user programs address, to allow for the programs or monitor stack region.

EX: I F000, F0FF, F026, 13,

The above key sequence displays F026 in the address field and the contents of the location in the data field, to allow insertion of the first byte. A total of 19 bytes can be inserted [13H] and the terminator EXEC returns control to the command recognizer routine.

4.2.7 DELETE COMMAND

Syntax: **K<start address of the user program><,><end address of the user program><,><starting address><ending address><,><Cr>**

A mini editing command that allows a portion of a user program to be deleted specified by a starting address and an editing address. All 3 byte memory reference instructions have their address modified to allow for the number of bytes that have been deleted from the user program. Addresses less than the ending address are not altered. The starting and ending user program addresses have to be entered into locations FFBA, FFBB, FFBC and FFBD as indicated in the case of the INSERT command.

EX: K F000, F0FF, FE04, FE33, <Cr>

The command displays the next address of the deleted block and its contents. Press EXEC to main command recognizer routine.

4.2.8 AUXILIARY COMMAND SET

Syntax: **D/AU**

On reset, the SDA-85-MEL initializes itself to the keyboard mode. The D/AU key, when pressed, transfers control to an auxiliary command set, including the cassette interface routines and EPROM programmer interface routine.

4.2.9 EPROM READ COMMAND

Syntax: **D9<starting EPROM address><,><ending EPROM address> <,><starting system RAM addr.><last 2 digits of EPROM type><.>**

EX: D9<,>200<,>2FF<,>F000<,>32<Cr>

The above key sequence will transfer the contents of a 2732 a type EPROM between address 200H and 2FFH to system RAM starting F000H.

4.2.10 EPROM PROGRAM COMMAND

Syntax: **D8<starting system ROM/RAM addr.><,><ending system ROM/ RAM addr.> <starting addr of EPROM in ZIF socket<,><last two digits of EPROM type> <.>**

EX: D8 F000, F0FF, 0000, 56

The above key sequence will program a 27256 EPROM starting from address 00H with contents of the system memory between addresses F000H and F0FFH inclusive.

4.2.11CRT COMMAND

Syntax: **E0**

Press **E** followed by **0** key this command allows the working mode from keyboard to serial for communication with a computer. RAM locations FFA6H and FFA7H should be initialized to the values corresponding to the required baud rate.

To branch to the serial monitor commands the following initialization of memory is required to select the baud rate.

FFA6H	FFA7H	BAUD RATE
05	00	19200
0A	00	9600
14	00	4800
28	00	2400
50	00	1200
A0	00	600
40	01	300

4.2.12LINE ASSEMBLER COMMAND

Syntax: **A [start-addr]<cr>...<ESC>**

Where start-address is the RAM address of the trainer, starting from which the program is to be entered. Rather than entering the programs or program changes in machine code, the line assembler can be used to enter the program in assembly language. The assembler accepts and converts mnemonic inputs machine code.

The converted code is then stored in the processor's program memory starting from the indicated start-address. After storing the codes in memory, memory is verified and if there is a mismatch, an error message is output as follows.

The assembler does not recognize symbolic labels; constants and addresses are entered as hexadecimal values. The instruction mnemonics accepted are those that had been adopted by the original manufacture of the processor. **Please note that <,> has to be pressed after entering the start address and not <Cr>.** The LCD display will be cleared. the line assembler then waits for an assembly language line input from the user & press <Cr>. The line is then assembled and the machine codes are stored from the indicated start address. The instruction mnemonics accepted is displayed on the first line of the display, the memory location and the data stored at that location would be displayed on the second line of the LCD display.

If any key is pressed, the disply once again goes blank and waits for the next entry. At this point if ESC key is pressed, the command is ended or the next mnemonics can be entered.

EX: Enter a 5-line program starting at the location F100H with an error on the third input line. The entries by the user are shown in bold phase

A

Starting ADDRESS: F100<,>

MVI A, 23<cr>	F100	3E23<Cr>
MVI C, 00<cr>	F102	0E00<Cr>
MVO B.C<cr>		
ERROR CODE, TRY AGAIN		
MOV B,C	F104	41<Cr>
INR A	F105	3C<Cr>
ORA B	F106	B0<Cr>

Press ESC to terminate the assembler process

TOTAL LINES ASSEMBLED 005

LINE ASSEMBLY COMPLETED

4.2.13DISASSEMBLE MEMORY COMMAND

Syntax: **Z**
Start- address: [starting source address]
End-address [ending source address]

Start source address: Is the hexadecimal address of the processor's program memory from where the disassembled memory content is displayed.

End source address: Is the hexadecimal address of the processor's program memory indicating the last memory location of the range to be disassembled for display.

If only the start-address is specified, the ****ERROR**** will be displayed. The user has to enter both start & end address. The content of that memory location is disassembled and displayed. More data are read from subsequent locations to complete the instruction if necessary. On the other hand, an end address specification causes the memory contents in the defined memory range to be disassembled and displayed. The end-address must be greater-than or equal to the start-address or an error message is printed and the command ended. If an illegal machine code is encountered an error message is displayed and the command is aborted.

The Disassembler is a two-pass Disassembler with all branch and subroutine call addresses first identified and converted to labels; a total of 248 labels are remembered for all disassembly. Depending on the range to be disassembled, there may be a pause before any line is displayed.

EX: Dissemble a previously entered 8085 program. Note that the byte following the end address is read in order to complete the instruction.

>Z
Starting address: 3000<cr>
Ending address: 300A<cr>

LABEL	SOURCE CODE		LOC	OBJ
	MVI	A, 00	3000	3E00
	OUT	DB	3002	D3DB
	LXI	H, ABCD	3004	21CDAB
	MVI	A, 10	3007	3E10
	JMP	3002	3009	C30230

DISASSSEMBLY COMPLETED

Please note that the Disassemble user the RAM in the region FD00-FEFF. Hence this area of RAM is not available to the user when the Disassembler is being run and if accessed the result are unpredictable.

4.3 SERIAL MONITOR COMMANDS

This section details the commands that are accessible from a computer in terminal emulation mode connected to the serial port. Characters (either I/P or O/P) are assumed to correspond to the ISO-ASCII-bit code. I/P & O/P routines effect the necessary conversion between HEX and ASCII. No action is taken if a NULL or LF character is input. SPACE and COMMA are interrupted as NEXT. The period is interpreted as EXEC. COLON (:) is used by the monitor as a record or line mark. As the command structure and function is essentially the same as that keyboard mode, only in the serial mode are explained in detail.

4.3.1 HELP MENU

On pressing the key **'H'** the following menu will be displayed.

EX: >H

****SDA-85-MEL Monitor****
***Summary of Monitor Commands**

***Memory Commands**		***Utility Commands**	
<D>	**Display**	**<X>**	**Examine Register**
<M>	**Modify**	**<A>**	**Assemble**
<B>	**Block Move**	**<Z>**	**Disassemble**
<I>	**Insert**	**<E>**	**EPROM Programmer**
<K>	**Delete**	**<R>**	**Cassette Save**
<F>	**Block Fill**	**<P>**	**Cassette Load**
<C>	**Block Complement**	**<H>**	**Help**

***Execution Commands**

<G>	**Execute**	**<S>**	**Single Step**

4.3.2 DISPLAY MEMORY COMMAND

Syntax: **>D**

In the serial mode this allows a formatted display of the contents of memory specified by a starting address and an ending address.

EX: **>D**
Starting Address: C000<,>
Ending Address: C0FF<,>

The above command produces a line display with a ":" colon for the line (record) mark, followed by the starting address of the line and then the contents of the memory locations. The last location is followed by a $.

4.3.3 MODIFY MEMORY COMMAND

Syntax: **M [start-addr.]<,>...<Cr>**
EX: **M**
Starting Address: FE10<,>3E-32 11-54 06-1C 22-CA 32-<CR>

The syntax is clearly indicated. The original contents are displayed requesting the user to make changes if required. The delimiter (, or space bar) displays the contents of the next location. The delimiter <Cr> terminates the command.

4.3.4 BLOCK MOVE COMMAND

Syntax: **B [start-addr] <cr>[end addr]<cr>[destination addr]**
EX: **B 0<cr>FF<cr>F000<cr>**

The above key sequence moves the contents of memory between addresses 00H and FFH to RAM starting at location F000H.

4.3.5 INSERT COMMAND

Syntax: I
[Start- addr.]<Cr>
[End- addr.]<Cr>
[Addr.]<Cr>
[No. of bytes] <,>.... <Cr>

EX: **I**
FC20<cr>
FC40<cr>
FC2D<cr>
F<,>32-11 06...<cr>

The above allows the user to insert 0FH bytes starting from location FC20H. The delimiter cr terminates the command.

4.3.6 DELETE COMMAND

Syntax: **K** [Start- addr.]<Cr>
[End- addr.]<Cr>
#Delete Block# [Start addr.]<Cr>
[End addr.] <,>

EX: K
Starting Address of user program: **FC20<cr>**
Ending Address of user program: **FC40<cr>**
#Delete Block#
Starting Address: **FC30<cr>**
Ending Address: **FC40<,>**
-33

The above deletes the portion between FC30H and FC40H, inclusive. 33 are the content of the location, after the last user program location. CR terminates the command.

4.3.7 BLOCK FILL COMMAND

Syntax: **F**
[Start-addr.]<Cr>
[End-addr]<cr>
[Constant]<cr>

EX: **F**
F400<cr>
F600<cr>
55<cr>

The above command fills the memory region between F400 and F600 with 55H. A<Cr> instead of the constant 55 fills the region with FFH.

4.3.8 BLOCK COMPLEMENT COMMAND

Syntax: **C**
[Start-addr.]<Cr>
[End-addr]<cr>

EX: **C**
F400<cr>
F600<cr>

The above command complements the content of the memory region between F400 and F600.

4.3.9 EXAMINE REGISTER COMMAND

Syntax: X [Register identifier]{new value}<space>....<cr>

EX: **X A=00-55<SP> B=00-20<SP> C=05 <SP> D=01<cr>**

The command indicates that the contents of a register are 00H and the hyphen informs the user that if he wants to change it, he can input the necessary character. 55H is the new data for A. If the space bar is next pressed the contents of B register is displayed. Cr terminates the command.

4.3.10 GO COMMAND

Syntax: **G [start-addr.]<Cr>**

The delimiter CR allows program execution. When the **G** key is pressed to activate this command the display on the console is,
G FB22-32/FE30<cr>
The above executes the program at location FE30H

4.3.11 SINGLE STEP COMMAND

Syntax: **S [start-addr.] <Space> <cr>**

A typical display using this command is given below.
S FD53: 4F/ FE11 FE11-32/ FE14-06/ FE16-3C/ <cr>

After issuing the prompt by the monitor, if the **S** key pressed the content of the program connect same location and the corresponding byte of information is displayed. A new address can then be typed in followed by a SPACE or just a SPACE executes the instruction at the location displayed and contents. As long as the SPACE key is pressed the program is executed one instruction at a time. The delimiter CR terminates the command.

4.3.12 FILE UPLOAD

Syntax: **U**

After entering the **U** key go to main menu by pressing F10 and select option 4.

Note one should not press the <cr> key after keying in the "U" key while invoking the file upload command.

4.3.13 FILE DOWNLOAD

Syntax: **L<cr>**

After entering the L<Cr> go to main menu by pressing F10 and select option 3.

Note one should press the <cr> key after keying in the "L" key while invoking the file download command.

4.4 USER PROGRAM TERMINATION

The op-code 'CF' when introduced into a program module, causes the program to break at that address and to save all the CPU registers. The EXAM REG command can then be used to study the contents of the CPU registers, there by assisting in program debugging.

4.5 UTILITY ROUTINES

The UTILITY ROUTINES available in the monitor are listed below.

4.5.1 CONSOLE INPUT-CI

Starting address: **04A5H**
Outputs: (A) = character that was received
Destroys: A & Flags
The console input routine is used in the serial mode to input a character from the terminal.

4.5.2 CONSOLE OUTPUT-CO

Starting address: **0511H**
Outputs: (C) = character to be output
Destroys: A & Flags
The console output routine is used in the serial mode to output a character that is in the register C to the external terminal.

4.5.3 DELAY

Starting address: **057FH**
Outputs: (DE) = 16 bit quantity, no of times the loop is executed
Loop time: 7.85micro-secs

4.5.4 DISUB

Starting address: **05AFH**
Outputs: (BC) = address 1, (DE) = address 2
Destroys: A, B, C, D, E, H, L & Flags
DISUB gets two addresses, the first in BC, the second in DE and the final delimiter in A.

4.5.5 ECHO

Starting address: **05D6H**
Outputs: (C) = character to be output
Destroys: A & E
ECHO outputs a character on the C register to the external serial device. IF CR was the character the LF is output.

4.5.6 GTHEX

Starting address: **0605H**
Outputs: (B) = 0; use the address field, (B) = 1; use the data field
Outputs: (A) = last character read (terminator)
: (DE) = hex digits evaluated module 2** 16 carry = 0 if no character was received
Destroys: A, B, C, D, E, H, L & Flags

GTHEX gets hex digits from the keyboard or serial device and displays them as they are received. Only the last four digits are retained and returned in D, E, if B is then the last two digits are retuned. A valid terminator like NEXT, EXEC, SPACE, PREV< CR and $ terminates the digits and is returned in A as an output of the routine. If no digits were received the carry flag is reset.

4.5.7 NMOUT

Starting address: **06E2H**
Inputs: (A) = byte that is to be output
Destroys: A, C, H, L & Flags
NMOUT outputs the byte, which is in A, as two ASCII, characters corresponding to the two nibbles to the serial device.

4.5.8 OUTPUT

Starting address: **0042H**
Inputs: (A) = displays flag; 0 = use address field; 1 = use data field
(B) = dot flag; 0 = no dot; 1 = dot
Destroys: A, H, L & Flags
The OUTPUT routine is used in the keyboard mode to output characters to the display. Either 4 characters or 2 characters are output using the H, L register as a pointer to the characters. The output flags determine whether a dot appears with the last character or not. The display table is 06F6H is used to translate the code to the character to be displayed.

4.5.9 RDKBD

Starting address: **071FH**
Inputs: (A) = character read from keyboard
Destroys: A, H, L & Flags
RDKBD fetches a character from the keyboard. Prior to using this routine RST5.5 is to be unmasked.

4.5.10 SCHRI

Starting address: **0763H**
Inputs: (A) = character received from serial device
Destroys: A, C & Flags
SCHRI inputs a character from the serial device. No action is taken if a NULL or LF was received. Otherwise the character is echoes to the console.

4.5.11 UPDAD

Starting address: **07A7H**
Inputs: (FFF7), (FFF8) = address to be displayed (B) = dot flag,
0 = not dot, 1 = dot
Destroys: A, B, C, D, E, H, L & Flags
UPDAD is used in the keyboard and serial modes to update the address field display using the current address stored at location FFF7H & FFF8H.

4.5.12 UPDDT

Starting address: **07D1H**
Inputs: (FFF9) = data to be (B) = dot flag; 0 = no dot 1= dot
Destroys: A, B, C, D, E, H, L & Flags
UPDDT is used in the keyboard and serial modes to update the data field display using the current data at location FFF9H.

CHAPTER 5

PC/XT/AT DRIVER PROGRAM

5.1 INTRODUCTION

PCLINK.EXE is a driver program that has be used for interfacing SDA-85-MEL trainer to a computer through an RS-232C link. It allows you to use the computer as a simple display terminal, to load or to save an Intel-hex file from computer to or from SDA-85-MEL memory.

5.2 GETTING STARTED

Connect the SDA-85-MEL to the computer auxiliary serial port (either COM1 or COM2 :) through an RS232C cable. The RS232C cable must have the DTR and DSR lines connected, in addition to the TXD, RXD and GND lines. Insert the disk supplied into your computer and type. **PClink**

The PCLINK program first displays a welcome message press <cr>, then display the main menu, as shown below.

MAIN MENU

1. Terminal mode
2. Disk catalog
3. File download
4. File upload
5. Configuration
6. Exit program

Enter your choice [1-6]?

Select an option of your choice. The significance and use of the different options is given below.

5.2.1 TERMINAL MODE

This option allows you to use the computer as an ordinary display terminal. Whatever you type on the computer keyboard will appear on its screen and will be sent to the SDA-85-MEL. Whatever the SDA-85-MEL transmits will appear on the computer screen. To go to this mode, select item 1 on the main menu.

While you are working in this mode, press function key **"F10"** to go back to main menu.

5.2.2 DISK CATALOG

This option allows you to display the directory of any drive. To select this option, press 2 from the main menu. The system then asks for the drive name by displaying the message.
ENTER DRIVE [A, B...]?
After you key in the drive specified, the system displays the directory of that drive and then displays the message,
PRESS<ENTER>
When you press, the main menu will be displayed

5.2.3 FILE DOWNLOAD

This option allows you to transfer an Intel-hex file, stored on a floppy diskette, to SDA-85-MEL memory. To select this option, press 3 from the main menu.

As soon as you select this option, the following prompt will be displayed.

FILE DOWNLOAD
CURRENT TRANSLATION FORMAT: INTEL HEX
Name of the file to download

Enter the name of the file which you want to transfer and press <ENTER>. The program assumes default extension of hex for the file. The system then starts the transfer operation. While the transfer operation is in progress, the system displays the number of the record being transferred.

NOTE:

1. Whenever you select this option, you must always make sure that (file download) the SDA-85-MEL is ready to receive an Intel-hex file. This is achieved by executing the following FILE DOWNLOAD command from the terminal mode.
   L<Cr> go to main menu by pressing **"F10"** and select option 3
2. After the file download operation is complete, you must always goto the terminal mode.

5.2.4 FILE UPLOAD

You may have modified or written a program in the SDA-85-MEL memory. This option allows you to save it as a file in hex-hex format. To go to this option, select item 4 from main menu. As soon as you select this item, the following message is displayed.

FILE UPLOAD
TRANSLATION FORMAT: INTEL HEX

Name of the file to store data in:
Enter the name of the file in which you want to store the data <CR>. The program assumes default extension for the file is HEX. You have to be careful in specifying the file name, as this program will overwrite an already existing file.
While the transfer operation is in progress, the system displays the number of the records being received. The program stores the data received on the disk and return to main menu after the transfer is complete.

NOTE:

1. You must always make sure that whenever you select this option (file upload), the SDA-85-MEL trainer is ready to transmit system memory data in hex format. This is achieved by executing the upload command from the terminal mode U. Go to main menu by pressing "F10" and select option 4.
2. After the file upload operation is complete, you must always go to the terminal mode. This is necessary because the system displays the result of the transfer operation on the screen.

5.2.5 CONFIGURATION

This option allows you to select the active serial port (either COM1 or COM2). It allows you to interactively select the following parameters.

Baud rate	: 300, 1200, 4800 and 9600
Word length	: 7, 8
Stop bits	: 1, 2
Parity	: none, odd or even
Port	: 1, 2

5.2.6 EXIT PROGRAM

You have finished using the PC85; select item 6 from the menu to exit. Selection of this item will return you to the operation system.

CHAPTER 6

TALK-USER'S GUIDE

WINDOWS BASED COMMUNICATION PACKAGE-

6.0 Introduction

Talk is a n RS-232C serial communication program for Microsoft Windows 98, Windows ME, Windows NT, Windows 2000 and Windows XP Operating systems. This serial communication program is meant to be used with the Advanced Electronic Systems (ALS) trainer kits like 8031, 8085, 8086 etc.

6.1 System Requirements

Hardware	:	an Intel x86 based computer
Software	:	Microsoft Windows 98, Windows NT, Windows 2000 or Windows XP
Hard-Disk space	:	~10MB
Memory	:	64KB

6.1.1 Installation Procedure

If you are installing on windows NT/2000/XP, Login as an Administrator else go to step 2.
Insert the Talk CD in to the CDROM drive.
Using Windows Explorer, Locate the file Setup Talk.exe on the CD.
Double- click on the setup Talk.exe file.
Follow the on- screen instructions.
If the setup program prompts for a REBOOT of the machine, you must reboot the system in order to use the Talk program.

6.1.2 Uninstallation Procedure

Method 1: Using Add/Remove from the control panel
Method 2: Click on Start → Programs → Talk → Uninstall

6.2 How to connect the ALS Trainer Kit to your Computer?

1. Locate a free COMM port (COM1, COM2, etc) connector (usually it is located at the back of the computer)
2. Connect one end of the serial cable (shipped with the kit) to the identified PC COMM port.
3. Connect the other end of the serial cable to the trainer kit.
4. Connect the trainer kit power supply cable (Refer to your ALS Trainer kit manual).
5. Set the baud rate, Data bits, Serial mode etc for the trainer kit as given in the kit manual (Refer to the ALS kit manual).
6. Run the talk program on your computer, start→programs→talk→talk →(see section 2.1).
7. From Talk's Options→ Settings menu, set the baud, data bits, etc to the same parameters as that of the trainer kit (See section 2.20).
8. Select the type of trainer kit (8085, 8031/51, and 8086) from options→ Target board menu (see section 2.3).
9. Connect to the COMM port using Talk's Options→ Connect menu→ (see section 2.4).
10. Switch on the power supply for the Trainer kit.

11. Put the Trainer kit into serial mode (Refer to your ALS Trainer Kit Manual).
12. If you can see the Sign-On message of the Trainer Kit on the Talk's terminal window, then you have successfully established the connection.
13. Once the connection is established successfully, you are ready to use the Talk program with your ALS Trainer Kit.
14. Whenever you type a command for the Trainer Kit, make sure that the Cursor is in the Talk's terminal window.

6.2.1 How to Run the Talk program?

As part of the Talk Program's Installation, a shortcut is placed on the start menu program. Click on the Start→ Programs→ Talk→Talk

When the Talk program is launched, a window is displayed similar to the one shown below:

6.2.2 How to close the Talk Program?

Once you have completed working with the Talk Program, you can close it by clicking on the File→ Exit menu.

6.2.3 Configuring COMM port Settings

From the 'Options' menu select "Settings' (Click on Options→Settings).

Select the desired parameters like Com port, Baud rate, etc and then press OK button.

6.2.4 Selecting the Trainer Kit

The talk program can be used with the ALS 8085, 8031/51 and 8086 Trainer Kits. So, it is important to select the type of Trainer Kit for which you are establishing connection with the Talk program.

Click on Options→ Target Board
Select the desired Kit and press OK button.

6.2.5 Connecting/ Disconnecting

To connect the Talk program to the selected PC COMM port, click on Options→ Connect.

To disconnect the Talk program from the PC COMM port, Click on Option→Disconnect.
Once it is disconnected, the Talk programs will not send/receive anything from the Trainer Kit.

6.2.6 Downloading an Intel HEX file to the Trainer Kit

Before downloading/uploading an Intel Hex file, make sure that you have selected the correct Trainer Kit (Options→ Target board menu) for which you have established the connection (8085, 8031/51 or 8086). Also make sure that you have a valid Intel Hex file.

To download, click on File→ Download Intel Hex file menu. The Talk programs will pop-up an "Open File" Dialog Box. Using this "Open File" Dialog Box, you can navigate to any drive/ folder to select your Intel Hex File. Select the desired file and click on "Open". Once the File Download operation is completed, the Talk program will return to the terminal window.

6.2.7 Uploading to an Intel HEX file from the Trainer Kit

To upload, Click on file→ upload Intel Hex file menu. The Talk program will pop-up a file "Save As" Dialog Box. You can type a file name or select the existing file. If the file doesn't exist, it will be created, if the file exists, you are prompted for over-writing on the existing file.

Next, the program prompts for starting memory address and Ending memory address. Enter the Start address and End address in hexadecimal and click the OK button.

Once the File Upload operation is completed, the Talk program will return to the terminal window.

6.2.8 About Properties Menu

When using ALS make trainer kit, the user need not change any settings in the properties menu. This is not relevant to trainer kit.

CHAPTER 7

PROGRAMMING EXAMPLES

7.1 USING INSERT AND DELETE COMMANDS

Insert command prompts for start address of the program, end address of the program, start address of the insertion point & number of bytes to insert. It will display data at that address & allows the user to enter new data.

Delete command prompts for start address of the program, end address of the program, start address & end address to be deleted. It will display data at next address after deletion.

```
FE90    0601          (0601) ;two instruction no change
        3E00          (3E00) ;
        CD A0 FF      (CD B0 FE)    ;three-byte non-mem. ref. change addr
        21 9F FF      (21 9F FF)    ;three-byte on-mem. ref. no change
        01 FF 07      (01 FF 07)    ;
FE90    64            (64)          ;single byte-no change
        65            (65)          ;
FE9F    66            (66)          ;new data can be inserted from here
        C3 94 FE      (C3 94 FE)    ;mem. ref-address INSERT
        C2 9D FE      (C2 9D FE)    ;address, hence no change
        D2 B0 FE      (D2 C0 FE)    ;address changed
        DA D0 FE      (DA D0 FE)    ;mem ref-addr last user prog addr
        33C           (CC 30 3C)    ;mem ref-address INSERT 0 address
        2F            (66)          ;original data has moved here after
        27            (C3 94 FE)    ;using the INSERT command
        FF
        FA A9 FE      (C2 9D FE)
        EC C0 FE      (D2 C0 FE)
        E60F
        2E10          (DA D0 FF)
        D4 9D 3E      (CC 30 FE)
        C4 90 3E      (2F)
        EA 94 3E      (37)
        01 9F 3E      (FF)          ;the program has grown in length
                      (FA B9 3E)    ; and occupies space upto FFD7H
                      (EC D0 3E)
                      (E6 0F)
                      (2E 10)
                      (D4 9D 3E)
                      (C4 90 3E)
                      (EA 94 3E)
        FFD5          (01 9D 3E)
```

The above program is not executable and is only meant to illustrate the use of the INSERT and DELETE commands.

7.2 USING INTERRUPT

This example shows in a simple way how the line Assembler and Disassembler are used and how the USART (8251 A) is used in an interrupt driven mode using RST6.5, short JP7 and reset the system. If a PC is being used in the terminal emulation mode, use the line Assembler program provided in the monitor to enter the following program at location F800. (The sequence of commands is shown below).

```
>A
Starting address:     F800
MVI A, 0D
SIM
LXI H, FA00
EI
JMP $
```

($ Represents the address of this jump instruction- while using the line Assembler use 0 wherever addresses are referred and enter these values later by reassembling only these lines. Keep a hand written copy of the program with labels as entry for subsequent alteration). The RST6.5 interrupt transfer point is next initialized and the interrupt program is written at location FF90H.

```
>A
Starting address;     FFAB
JMP F900
>A
Starting address;     F900
PUSH PSW;             save Acc and Flags
IN C0;          read a character from the USART
MOV M, A;             store it in the location FA00H upwards
INX H;          increment the ptr to the next locn
MOV A, H;             obtain hi-byte of pointer
CPI FB;         have 256 characters been received?
JNZ 0;          if not go ahead and obtain another
POP PSW;              0=temp addr retrieve acc and flags
RST 1;          all done-return to monitor
POP PSW;              0 addr jump here if more chrs
EI;                   re-enable interrupts for next USART
RET;                  int and return to main routine
```

To execute the above enter. The processor is now in a loop. For every key press on the system console (PC keyboard) will cause the UASRT to interrupt the CPU, because the RXRDY pin connected to the RST6.5 I/p of the CPU. If a key is kept pressed the program will terminate after 256 characters are received.

These character scan then be viewed in the memory region FA00 to FAFF using the display memory command.

7.3 TWENTY-FOUR HOURS DIGITAL CLOCK

This program simulates a twenty-four hour digital clock. The software uses the monitor routines DELAY, UPDDT, UPDAD to provide hours, minutes and seconds display in the address and data fields.

```
E000   21 00 00     STRT:     LXI    H, 0H
E003   22 F7 FF     TM00:     SHLD   CURAD
E006   CD A9 14               CALL CLR
E009   3E 83                  MVI    A, 83H
E00B   32 C2 FF               STA    FFC2H
E00E   CD 42 E0               CALL   UPDD2
E011   0E 00                  MVI    C, 00H
E013   79           TM11:     MOV    A, C
E014   32 F9 FF               STA    CURDT
```

```
E017  3E 85              MVI   A, 85H
E019  32 C2 FF           STA   FFC2H
E01C  CD 4E E0           CALL  UPDD1
E01F  CD 60 E0           CALL  DLY
E022  79                 MOV   A, C
E023  3C                 INR   A
E024  27                 DAA
E025  4F                 MOV   C, A
E026  FE 60              CPI   60H
E028  C2 13 E0           JNZ   TM11
E02B  7D                 MOV   A,L
E02C  3C                 INR   A
E02D  27                 DAA
E02E  6F                 MOV   L, A
E02F  FE 60              CPI   60H
E031  C2 03 E0           JNZ   TM00
E034  AF                 XRA   A
E035  6F                 MOV   L, A
E036  7C                 MOV   A, H
E037  3C                 INR   A
E038  27                 DAA
E039  67                 MOV   H, A
E03A  FE 24              CPI   24H
E03C  C2 03 E0           JNZ   TM00
E03F  C3 00 E0           JMP   STRT
E042  C5          UPDD2: PUSH  B
E043  E5                 PUSH  H
E044  D5                 PUSH  D
E045  F5                 PUSH  PSW
E046  CD A7 07           CALL  UPDAD
E049  F1                 POP   PSW
E04A  D1                 POP   D
E04B  E1                 POP   H
E04C  C1                 POP   B
E04D  C9                 RET
E04E  C5          UPDD1: PUSH  B
E04F  E5                 PUSH  H
E050  D5                 PUSH  D
E051  F5                 PUSH  PSW
E052  CD D1 07           CALL  UPDDT
E055  21 6D E0           LXI   H, HMS
E058  CD 5C 00           CALL  OUT2LSTR
E05B  F1                 POP   PSW
E05C  D1                 POP   D
E05D  E1                 POP   H
E05E  C1                 POP   B
E05F  C9                 RET
E060  11 FF FF     DLY:  LXI   D, 0FFFFH
E063  CD 7F 05           CALL  DELAY
E066  11 00 90           LXI   D, 9000H
E069  CD 7F 05           CALL  DELAY
E06C  C9                 RET
E06D  48 3A 4D 3A 53 HMS: DB   'H:M:S', 0DH
E072  20 0D
```

7.4 DECIMAL MULTIPLICATION

This is an example to multiply two BCD numbers not exceeding 99. The numbers to be multiplied are stored in locations FFA6H and FFA7H. The result is displayed.

```
1.   F000  21 00 00        LXI   H, 0H   ;initialize the result to 0
2.   F003  3A A6 FF        LDA   FFA6H   ;move the multiplier to B
3.   F006  47              MOV   B, A
4.   F007  3A A7 FF        LDA   FFA7H   ;move the multiplicand to C and to A
5.   F00A  4F              MOV   C, A
6.   F00B  79        STR5: MOV   A, C
7.   F00C  85              ADD   L       ;low byte of the partial result
8.   F00D  27              DAA           ;adjust for BCD arithmetic
9.   F00E  6F              MOV   L, A
10.  F00F  3E 00           MVI   A, 0H   ;compute high byte of the partial
11.  F011  BC              ADC   H
12.  F012  27              DAA           ;adjust for BCD arithmetic
13.  F013  67              MOV   H, A    ;partial result in HL
14.  F014  78              MOV   A, B    ;decrement multiplier by 1
15.  F015  C6 99           ADI   99H     ;tens compliment of 1=99
16.  F017  27              DAA           ;adjust for BCD operation
17.  F018  47              MOV   B, A    ;multiplier restored
18.  F019  C2 0B F0        JNZ   STR5    ;multiplier not=0,so loop
19.  F010  22 F7 FF        SHLD  FFF7H   ;all done, display the result
20.  F01F  CD BC 06        CALL  UPDAD   ;in the address field
21.  F022  76              HLT           ;halt the processor
```

TEST PROGRAM DETAILS:

We are providing a few of the programs, which can be entered and executed by the user.
The following procedure is followed to execute a program using a SDA-85-MEL kit:

1. Switch on the supply. → ASM.
2. Now the sign on message "-SDA-8085-STA" will be displayed on the display.
3. Now press the "SUBST MEM" key & enter the starting address of the program i.e., the address from where you want to start the program. (EX: For 2K RAM the starting address is F000). For 8K RAM the starting address is E000. Every Instruction ENTER key is Pressed.
4. Then the "NEXT" key is pressed & the data is entered in the consecutive memory location till the end of the program.
5. Make sure you press the "NEXT" key after entering the last data byte into the memory location.
6. Press the "RESET" key.
7. a. If the program requires certain data to be accessed from certain Memory locations, it has to be entered in the respective memory location before executing the program.
   b. Then press the "GO" key and enter the starting address of the program & then press the ~~"EXEC" key~~. → ENTER key.
8. Now you can observe the results in the memory locations or in the registers (using the "EXAM REG" key) as per your program.

TO ENTER Data TO De Memory location
Reset → (M) MEM → Enter the address → NEXT → Data to Entered. → NEXT → one More data → NEXT → Reset → e10 → (F000)

SAMPLE PROGRAM FOR SDA-85-MEL TRAINER KIT

1. Program to interchange contents of two memory locations.
2. Program to count number of ones in data byte.
3. Program to find two's complement of a number.
4. Program to add two 8 bit binary numbers with carry.
5. Program to add two consecutive numbers.
6. Program to add 3 bytes.
7. Program to multiply two 8-bit numbers.
8. Program to divide 16-bit number by 8-bit number.
9. Program to find if a given number is odd or even.
10. Program to find if a given number is positive or negative.
11. Generating file Fibonacci series number.
12. Program to find the largest number.
13. Program to subtract two 16 bit numbers.
14. Program to arrange numbers in sorted order.
15. Program to implement mod 16 down counter.
16. Program to implement mod 10 down counter.
17. Program to add two BCD Numbers.
18. Program to add five BCD numbers.
19. BCD to binary conversion.
20. Binary to BCD conversion.
21. Program to display message in seven segment display of the 85kit.
22. Program to add 16 bit binary numbers.
23. Program to subtract two 8 bit BCD numbers.
24. Program to multiply two 16 bit binary numbers.
25. Program to multiply two 8 bit BCD numbers.
26. Waveform generations.
    - # Sine waveform with time period of 10msec.
    - # Square waveform of 50% duty cycle and the time period of 12msec.
    - # Program to find the square of a number (0 to 7) using lookup table.
    - # Rectangular waveform with 60% duty cycle and time period of 10msec.
27. Program to arrange in ascending & descending order of given list of 10 numbers.
28. Program to subtract two 8 bit binary numbers.

29. Program to find the number of non blank characters in an ASCII string.
30. Program to test zero conditions of a byte in memory.
31. Program to add two 16 bit binary numbers.

1. Prg1

PROGRAM TO EXCHANGE CONTENTS OF TWO MEMORY LOCATIONS. THE CONTENTS OF MEMORY LOCATIONS F100 AND F101 ARE INTERCHANGED.

```
F000  21 00 F1        LXI   H,F100
F003  11 01 F1        LXI   D,F101
F006  46              MOV   B,M
F007  1A              LDAX  D
F008  77              MOV   M,A
F009  EB              XCHG
F00A  70              MOV   M,B
F00B  CF              RST 1
```

2. Prg2

PROGRAM TO COUNT NO OF 1'S IN A DATA BYTE PROGRAM READS THE CONTENT OF F100H NO OF 1'S ARE STORED IN C REGISTER IN THE END.

```
F000  3A 00 F1              LDA   F100H
F003  0E 00                 MVI   C,00
F005  06 08                 MVI   B,08
F007  07              RPT:  RLC
F008  D2 0A F0              JNC   F00C
F00B  0C                    INR   C
F00C  05              LBL1: DCR   B
F00D  C2 05 F0              JNZ   F007
F010  CF                    RST 1
```

3. Prg3

PROGRAM TO FIND 2'S COMPLEMENT OF A GIVEN BYTE INPUT A BYTE IN F100 LOCATION OUTPUT A REGISTER

```
F000  3A 00 F1        LDA   F100
F003  2F              CMA
F004  3C              INR   A
F005  CF              RST 1
```

4. Prg4

PROGRAM ADDS TWO 8 BIT NUMBERS USING ADD INSTRUCTION
TAKE CARE OF CARRY ALSO
INPUT: REGISTER A & REGISTER B
OUTPUT: REGISTER A & B

```
F000  0E D7           MVI   C,D7
F002  16 44           MVI   D,44
F004  79              MOV   A,C
F005  82              ADD   D
F006  47              MOV   B,A
```

```
F007  3E 00        MVI   A,0
F009  CE 00        ACI   0
F00B  CF           RST   1
```

5. Prg5

PROGRAM TO ADD TWO CONSECUTIVE NUMBERS IN MEMORY LOCATIONS.

```
F000  21 00 F1     LXI   H,F100
F003  46           MOV   B,M
F004  23           INX   H
F005  7E           MOV   A,M
F006  80           ADD   B
F007  47           MOV   B,A
F008  3E 00        MVI   A,0
F00A  CE 00        ACI   0
F00C  CF           RST 1
```

6. Prg6

PROGRAM TO ADD 3 BYTE NUMBERS WITH ANOTHER SET OF 3 BYTE NUMBERS IN MEMORY. 1ST SET OF 3 BYTES IS IN W1 W2 & W3 LOCATIONS. 2ND SET OF 3 BYTES IS IN W4 W5 & W6 LOCATIONS. RESULT IS STORED IN W7, W8, W9 & W10 LOCATIONS.

Load HL Pair

DAD - Add reg. Pair to HL Pair

store HL Pair using direct address

```
F000  2A 1D F0     LHLD  F01D
F003  EB           XCHG
F004  2A 20 F0     LHLD  F020
F007  19           DAD   D
F008  22 23 F0     SHLD  F023
F00B  21 1F F0     LXI   H,F01F
F00E  3A 22 F0     LDA   F022
F011  8E           ADC   M
F012  32 25 F0     STA   F025
F015  3E 00        MVI   A,0
F017  CE 00        ACI   0
F019  32 26 F0     STA   F026
F01C  CF           RST   1
F01D  00     W1:   DB    0
F01E  00     W2:   DB    0
F01F  00     W3:   DB    0
F020  00     W4:   DB    0
F021  00     W5:   DB    0
F022  00     W6:   DB    0
F023  00     W7:   DB    0
F024  00     W8:   DB    0
F025  00     W9:   DB    0
F026  00     W10:  DB    0
```

7. Prg7

MULTIPLIES TWO 8 BIT NUMBERS. MULTIPLIER AND MULTIPLICAND IN F100 & F101 OUTPUT IN HL REGS.

```
F000  21 00 F1     LXI   H,F100
F003  5E           MOV   E,M
```

```
F004  16 00                MVI   D,0
F006  23                   INX   H
F007  7E                   MOV   A,M
F008  0E 08                MVI   C,08
F00A  21 00 00             LXI   H,00
F00D                ST05:
F00D  0F                   RRC
F00E  D2 12 F0             JNC   F012
F011  19                   DAD   D
F012                ST10:
F012  EB                   XCHG
F013  29                   DAD   H
F014  EB                   XCHG
F015  0D                   DCR   C
F016  C2 0D F0             JNZ   F00D
F019  CF                   RST   1
```

8. Prg8

Program to divide 16 bit no. by 8 bit no.
Input : dividend 16 bit no. in location f100,f101
divisor is in location f102
Output : reg L

```
F000  2A 00 F1             LHLD  F100     ; load 2nd 16 bit no. to hl
F003  3A 02 F1             LDA   F102     ;load 1st 16 bit no. to hl
F006  47                   MOV   B,A
F007  0E 08                MVI   C,08
F009                ST05:
F009  29                   DAD   H
F00A  7C                   MOV   A,H
F00B  90                   SUB   B
F00C  DA 11 F0             JC    F011
F00F  67                   MOV   H,A
F010  2C                   INR   L
F011                ST10:
F011  0D                   DCR   C
F012  C2 09 F0             JNZ   F009
F015  CF                   RST   1
```

9. Prg9

PROGRAM TO FIND WHETHER A GIVEN BYTE IS ODD OR EVEN
IF THE NUMBER IS ODD F100 = FF ELSE = 00

```
F000  3A 00 F1             LDA   F100
F003  0F                   RRC
F004  DA 0C F0             JC    F00C
F007  3E 00         EVEN:  MVI   A,00
F009  C3 0E F0             JMP   F00E
F00C  3E FF         ODD:   MVI   A,FF
F00E  32 01 F1      L1:    STA   F101
F011  CF                   RST   1
```

10. Prg10

PROGRAM TO FIND WHETHER A GIVEN BYTE IS POSITIVE OR NEGATIVE
IF THE NUMBER IS NEGATIVE F100 = FF ELSE = 00

For negative number enter 2's complement.

```
F000  3A 00 F1          LDA   F100
F003  07                RLC
F004  DA 0C F0          JC    F00C
F007                    POSITIVE:
F007  3E 00             MVI   A,00
F009  C3 0E F0          JMP   F00E
F00C                    NEGATIVE:
F00C  3E FF             MVI   A,FF
F00E  32 01 F1          L1:   STA   F101
F011  CF                RST   1
```

11. Prg11

PROGRAM TO GENERATE FABONICCI SERIES

```
F000  21 00 F1          LXI   H,F100
F003  16 0A             MVI   D,10      ;10 numbers are generated
F005  06 01             MVI   B,01
F007  70                MOV   M,B
F008  23                INX   H
F009  0E 01             MVI   C,01
F00B  71                MOV   M,C
F00C  23                INX   H
F00D  3E 00             RPT:  MVI   A,00
F00F  80                ADD   B
F010  27                DAA
F011  81                ADD   C
F012  27                DAA
F013  48                MOV   C,B
F014  47                MOV   B,A
F015  77                MOV   M,A
F016  23                INX   H
F017  15                DCR   D
F018  C2 0D F0          JNZ   F00D
F01B  CF                RST   1
```

F100 5
101 0
102 5
103 1
4
5
6

12. Prg12

PROGRAM TO FIND LARGEST NUMBER IN AN UNSORTED ARRAY
INPUT F100 : NUMBER OF ELEMENTS IN ARRAY
F101 ONWARDS ARRAY ELEMENTS
OUTPUT: REGISTER A

```
F000  21 00 F1          LXI   H,F100
F003  4E                MOV   C,M       ; no. of elements to sort
F004  0D                DCR   C
F005  23                INX   H
F006  7E                MOV   A,M
F007  23                ST05: INX   H
F008  BE                CMP   M
```

```
F009  D2 0D F0              JNC   F00D
F00C  7E                    MOV   A,M
F00D  0D            ST10:   DCR   C
F00E  C2 07 F0              JNZ   F007
F011  CF                    RST   1
```

13. Prg13

PROGRAM TO SUBTRACT 2 16 BIT NUMBERS
(W1 W2) - (W3 W4) = (W5 W6)

```
F000  21 15 F0      ST:     LXI   H,F015
F003  11 13 F0              LXI   D,F013
F006  01 17 F0              LXI   B,F017
F009  1A                    LDAX  D
F00A  96                    SUB   M
F00B  02                    STAX  B
F00C  23                    INX   H
F00D  13                    INX   D
F00E  03                    INX   B
F00F  1A                    LDAX  D
F010  9E                    SBB   M
F011  02                    STAX  B
F012  CF                    RST   1
F013  00            W1:     DB    0
F014  00            W2:     DB    0
F015  00            W3:     DB    0
F016  00            W4:     DB    0
F017  00            W5:     DB    0
F018  00            W6:     DB    0
```

14. Prg14

PROGRAM TO SORT AN ARRAY IN ASCENDING ORDER
INPUT: F100 - F107 GIVEN ARRAY
OUTPUT: F100 - F107 SORTED ARRAY IN ASCENDING ORDER

```
F000  06 00       START:    MVI   B,0
F002  0E 08                 MVI   C,08
F004  21 00 F1              LXI   H,F100
F007  7E            ST05:   MOV   A,M
F008  23                    INX   H
F009  BE                    CMP   M
F00A  DA 16 F0              JC    F016
F00D  CA 16 F0              JZ    F016
F010  5E                    MOV   E,M
F011  2B                    DCX   H
F012  73                    MOV   M,E
F013  23                    INX   H
F014  77                    MOV   M,A
F015  04                    INR   B
F016  0D            ST10:   DCR   C
F017  C2 07 F0              JNZ   F007
F01A  78                    MOV   A,B
F01B  FE 00                 CPI   00
F01D  C2 00 F0              JNZ   F000
F020  CF                    RST   1
```

15. Prg15

PROGRAM IS MOD 16 DOWN- COUNTER FFFF TO 0000

```
07A7   UPDAD        EQU   7A7H

F000   21 FF FF             LXI   H,FFFF
F003   E5           ST05:   PUSH  H
F004   22 F7 FF             SHLD  FFF7
F007   CD A7 07             CALL  07A7H
F00A   CD 12 F0             CALL  F012
F00D   E1                   POP   H
F00E   2B                   DCX   H
F00F   C3 03 F0             JMP   F003
F012   06 02        DELAY:  MVI   B,02
F014   21 FF FF      D100:  LXI   H,FFFF
F017   2B             D105: DCX   H
F018   7C                   MOV   A,H
F019   B5                   ORA   L
F01A   C2 17 F0             JNZ   D105
F01D   05                   DCR   B
F01E   C2 14 F0             JNZ   D100
F021   C9                   RET
```

16. Prg16

PROGRAM IS DECIMAL UP-COUNTER 00 TO 99

```
07D1   UPDDT EQU   7D1H

F000   3E 00                MVI   A,00
F002   F5           ST05:   PUSH  PSW
F003   32 F9 FF             STA   FFF9
F006   CD D1 07             CALL  07D1H
F009   CD 13 F0             CALL  F013
F00C   F1                   POP   PSW
F00D   C6 01                ADI   01
F00F   27                   DAA
F010   C3 02 F0             JMP   F002
F013   06 0A        DELAY:  MVI   B,10
F015   21 FF FF      D100:  LXI   H,FFFF
F018   2B             D105: DCX   H
F019   7C                   MOV   A,H
F01A   B5                   ORA   L
F01B   C2 18 F0             JNZ   F018
F01E   05                   DCR   B
F01F   C2 15 F0             JNZ   F015
F022   C9                   RET
```

17. Prg17

PROGRAM TO ADD 2 BCD NUMBERS IN F100 AND F101.THE RESULT IS IN 'A' REGISTER

```
F000  21 00 F1          LXI   H,F100
F003  7E                MOV   A,M
F004  23                INX   H
F005  86                ADD   M
F006  27                DAA
F007  CF                RST   1
```

18. Prg18

PROGRAM TO ADD 5 DECIMAL NUMBERS CONSECUTIVELY FROM F100
IGNORING OVERFLOW
THE RESULT IS IN 'A' REGISTER

```
F000  06 04             MVI   B,04
F002  21 00 F1          LXI   H,F100
F005  7E                MOV   A,M
F006  23          RPT:  INX   H
F007  86                ADD   M
F008  27                DAA
F009  05                DCR   B
F00A  C2 06 F0          JNZ   F006
F00D  CF                RST   1
```

19. Prg19

PROGRAM TO CONVERT BCD NUMBER TO BINARY
INPUT: BCD NUMBER IN F100H
OUTPUT: A REGISTER

```
F000  3A 00 F1          LDA   F100
F003  47                MOV   B,A
F004  E6 0F             ANI   0F
F006  4F                MOV   C,A
F007  78                MOV   A,B
F008  E6 F0             ANI   F0
F00A  0F                RRC
F00B  0F                RRC
F00C  0F                RRC
F00D  0F                RRC
F00E  47                MOV   B,A
F00F  CD 14 F0          CALL  F014
F012  81                ADD   C
F013  CF                RST   1
F014  AF          ADDR: XRA   A
F015  C6 0A       AD05: ADI   0A
F017  05                DCR   B
F018  C2 15 F0          JNZ   F015
F01B  C9                RET
```

20. Prg20

PROGRAM TO CONVERT BINARY NUMBER TO BCD
INPUT: BCD NUMBER IN F100H
OUTPUT: HL REGISTER PAIR

```
F000  3A 00 F1            LDA   F100
F003  06 64               MVI   B,64
F005  0E 00               MVI   C,0
F007  CD 1C F0            CALL  F01C
F00A  51                  MOV   D,C          ; 1000S IN D
F00B  06 0A               MVI   B,0A
F00D  0E 00               MVI   C,0
F00F  CD 1C F0            CALL  F01C
F012  6F                  MOV   L,A          ; 10S IN L
F013  79                  MOV   A,C
F014  07                  RLC
F015  07                  RLC
F016  07                  RLC
F017  07                  RLC
F018  B5                  ORA   L
F019  6F                  MOV   L,A          ;10S & UNITS INL
F01A  62                  MOV   H,D          ; 1000S IN  H
F01B  CF                  RST   1
F01C  90          CONV:   SUB   B
F01D  DA 24 F0            JC    F024
F020  0C                  INR   C
F021  C3 1C F0            JMP   F01C
F024  80          LOOP:   ADD   B
F025  C9                  RET
```

21. Prg21

THE PROGRAMS DISPLAYS "GOOD BAD UGLY" ON 7 SEGMENT DISPLAY ON 85 ME KIT

```
F000  3E 10               MVI   A,10         ;mode command
F002  D3 D1               OUT   D1           ;control register address of 8279
F004  21 00 F1STRT:       LXI   H,F100 ;data displayed stored at 8100h
F007  E5                  PUSH  H            ;onwards
F008  0E 06       REP1:   MVI   C,06
F00A  06 87               MVI   B,87         ;display deice address
F00C  78          REP:    MOV   A,B
F00D  D3 D1               OUT   D1
F00F  7E                  MOV   A,M
F010  D3 D0               OUT   D0           ;data register address of 8279
F012  23                  INX   H
F013  05                  DCR   B
F014  0D                  DCR   C
F015  C2 0C F0            JNZ   F00C
F018  C5                  PUSH  B
F019  0E 02               MVI   C,02
F01B  11 FF FF    DEL1:   LXI   D,FFFF
F01E  1D          DEL:    DCR   E
F01F  C2 1E F0            MVI   E,FF
F024  15                  DCR   D
```

```
F025  C2 1E F0          JNZ   F01E
F028  0D                DCR   C
F029  C2 1B F0          JNZ   F01B
F02C  C1                POP   B
F02D  E1                POP   H
F02E  23                INX   H
F02F  E5                PUSH  H
F030  7D                MOV   A,L
F031  FE 14             CPI   14
F033  C2 08 F0          JNZ   F008
F036  C3 04 F0          JMP   F004
F100                    ORG   F100
F100  FF          TBL:  DB    FF    ;code for the data to be displayed
F101  FF                DB    FF
F102  FF                DB    FF
F103  FF                DB    FF
F104  FF                DB    FF
F105  FF                DB    FF
F106  82                DB    82
F107  A3                DB    A3
F108  A3                DB    A3
F109  C0                DB    C0
F10A  FF                DB    FF
F10B  80                DB    80
F10C  88                DB    88
F10D  C0                DB    C0
F10E  FF                DB    FF
F10F  C1                DB    C1
F110  82                DB    82
F111  C7                DB    C7
F112  91                DB    91
F113  FF                DB    FF
F114  FF                DB    FF
F115  FF                DB    FF
F116  FF                DB    FF
F117  FF                DB    FF
F118  FF                DB    FF
F119  FF                DB    FF
F11A  FF                DB    FF
F11B  FF                DB    FF
F11C  FF                DB    FF
F11D  FF                DB    FF
```

22. Prg22

Program to add 2, 16 bit binary nos.
Input : E000 - E001 1st 16 bit no.
E002 - E003 2nd 16 bit no.
Output : D E register pair.

```
F000  2A 00 E0          LHLD  E000  ;load 1st 16 bit no. to hl
F003  EB                XCHG        ; move it to de regs
F004  2A 02 E0          LHLD  E002  ;load 2nd 16 bit no. to hl
F007  7D                MOV   A,L
F008  83                ADD   E
```

```
F009  5F                        MOV   E,A
F00A  7C                        MOV   A,H
F00B  8A                        ADC   D
F00C  57                        MOV   D,A
F00D  CF                        RST   1
E000                            ORG   E000
E000  00 00               W1    DW    0
E002  00 00               W2    DW    0
```

23. Prg23

PROGRAM TO SUBTRACT 2 8 BIT NUMBERS
INPUT : 8000 AND 8001 LOCATION
OUTPUT :8002H

```
F000  21 00 80                  LXI   H,8000 ; subtrahend 8000h
F003  11 01 80                  LXI   D,8001 ;minuend 8001h
F006  4E                        MOV   C,M
F007  1A                        LDAX  D
F008  47                        MOV   B,A
F009  CD 14 F0                  CALL  F014
F00C  E5                        PUSH  H
F00D  21 02 80                  LXI   H,8002 ;result at 8002h
F010  77                        MOV   M,A
F011  23                        INX   H
F012  72                        MOV   M,D
F013  CF                        RST   1
F014  16 00         S_BCD:      MVI   D,00
F016  3E 99                     MVI   A,99
F018  91                        SUB   C
F019  3C                        INR   A
F01A  80                        ADD   B
F01B  27                        DAA
F01C  D2 20 F0                  JNC   F020
F01F  14                        INR   D
F020  C9                  S05:  RET
```

24. Prg24

16 bit multiplication program
input 8200(msb) 8201(lsb) multiplicand
8202(msb) 8203(lsb) multiplier
output : 8210(msb),8211,8212.8213(lsb)
repeated addition is used to perform multiplication
it takes some time to perform computation
at the end kit gets reset and user can verify the
result at approprite locations.

```
F000  21 01 82          ST:     LXI   H,8201
F003  01 10 82                  LXI   B,8210
F006  3E 00                     MVI   A,00
F008  16 04                     MVI   D,04
F00A  02                  RPT:  STAX  B
F00B  03                        INX   B
F00C  15                        DCR   D
```

```
F00D  C2 0A F0                  JNZ   F00A
F010  0B                        DCX   B
F011  11 03 82      CHK:        LXI   D,8203
F014  1A                        LDAX  D
F015  FE 00                     CPI   00
F017  CA 1D F0                  JZ    F01D
F01A  C3 2D F0                  JMP   F02D
F01D  1B            MSB:        DCX   D
F01E  1A                        LDAX  D
F01F  FE 00                     CPI   00
F021  CA 5A F0                  JZ    F05A
F024  3D                        DCR   A
F025  12                        STAX  D
F026  13                        INX   D
F027  3E FF                     MVI   A,FF
F029  12                        STAX  D
F02A  C3 2F F0                  JMP   F02F
F02D  3D            ADD:        DCR   A
F02E  12                        STAX  D
F02F  0A                  L2:   LDAX  B
F030  86                        ADD   M
F031  02                        STAX  B
F032  2B                        DCX   H
F033  0B                        DCX   B
F034  0A                        LDAX  B
F035  8E                        ADC   M
F036  D2 54 F0                  JNC   F054
F039  C5                        PUSH  B
F03A  F5                        PUSH  PSW
F03B  01 11 82                  LXI   B,8211
F03E  0A                        LDAX  B
F03F  FE FF                     CPI   FF
F041  CA 48 F0                  JZ    F048
F044  3C                        INR   A
F045  C3 51 F0                  JMP   F051
F048  AF                  L5:   XRA   A
F049  02                        STAX  B
F04A  0B                        DCX   B
F04B  0A                        LDAX  B
F04C  3C                        INR   A
F04D  02                        STAX  B
F04E  C3 52 F0                  JMP   F052
F051  02                  L4:   STAX  B
F052  F1                  L3:   POP   PSW
F053  C1                        POP   B
F054  02                  L1:   STAX  B
F055  03                        INX   B
F056  23                        INX   H
F057  C3 11 F0                  JMP   F011
F05A  CF            SUB05:      RST   1
```

25. Prg25

BCD multiplication (8 bit)
One of the operand is to be at F100h and the other at F101h
The result is stored at F150h and F151h

```
F000  21 00 F1     ST:    LXI   H,F100 ;multplicand   (45)
F003  11 01 F1            LXI   D,F101 ;multiplier            ( 36)
F006  7E                  MOV   A,M
F007  E6 0F               ANI   0F
F009  4F                  MOV   C,A    ;c<-5
F00A  1A                  LDAX  D
F00B  E6 0F               ANI   0F
F00D  47                  MOV   B,A    ;b<-6
F00E  CD AB F0            CALL  F0AB
F011  01 00 F3            LXI   B,F300 ; F300<-30
F014  02                  STAX  B
F015  7E                  MOV   A,M
F016  E6 F0               ANI   F0
F018  0F                  RRC
F019  0F                  RRC
F01A  0F                  RRC
F01B  0F                  RRC
F01C  4F                  MOV   C,A    ;c<-4
F01D  1A                  LDAX  D
F01E  E6 0F               ANI   0F
F020  47                  MOV   B,A    ;b<-6
F021  CD AB F0            CALL  F0AB
F024  01 01 F3            LXI   B,F301
F027  02                  STAX  B
F028  E5                  PUSH  H
F029  21 00 F3            LXI   H,F300
F02C  4E                  MOV   C,M
F02D  23                  INX   H
F02E  46                  MOV   B,M
F02F  E1                  POP   H
F030  CD B3 F0            CALL  F0B3
F033  E5                  PUSH  H
F034  21 00 F2            LXI   H,F200        ;f200<-02
F037  70                  MOV   M,B
F038  23                  INX   H      ;f201<-70
F039  71                  MOV   M,C
F03A  E1                  POP   H
F03B  7E                  MOV   A,M
F03C  E6 0F               ANI   0F
F03E  4F                  MOV   C,A    ;c<-05
F03F  1A                  LDAX  D
F040  E6 F0               ANI   F0
F042  0F                  RRC
F043  0F                  RRC
F044  0F                  RRC
F045  0F                  RRC
F046  47                  MOV   B,A    ;b<-03
F047  CD AB F0            CALL  F0AB
F04A  01 02 F3            LXI   B,F302
```

```
F04D  02                  STAX  B
F04E  7E                  MOV   A,M
F04F  E6 F0               ANI   F0
F051  0F                  RRC
F052  0F                  RRC
F053  0F                  RRC
F054  0F                  RRC
F055  4F                  MOV   C,A     ;c<-04
F056  1A                  LDAX  D
F057  E6 F0               ANI   F0
F059  0F                  RRC
F05A  0F                  RRC
F05B  0F                  RRC
F05C  0F                  RRC
F05D  47                  MOV   B,A     ;b<-03
F05E  CD AB F0            CALL  F0AB
F061  01 03 F3            LXI   B,F303          ;F303<-12
F064  02                  STAX  B
F065  E5                  PUSH  H
F066  21 02 F3            LXI   H,F302
F069  4E                  MOV   C,M
F06A  23                  INX   H
F06B  46                  MOV   B,M
F06C  E1                  POP   H
F06D  CD B3 F0            CALL  F0B3
F070  E5                  PUSH  H
F071  21 02 F2            LXI   H,F202  ;F302<-01
F074  70                  MOV   M,B
F075  23                  INX   H
F076  71                  MOV   M,C     ;F303<-35
F077  E1                  POP   H
F078  21 01 F2            LXI   H,F201
F07B  11 51 F1            LXI   D,F151
F07E  06 00               MVI   B,00
F080  4E                  MOV   C,M     ;c<-70
F081  23                  INX   H
F082  23                  INX   H
F083  7E                  MOV   A,M
F084  E6 0F               ANI   0F
F086  07                  RLC
F087  07                  RLC
F088  07                  RLC
F089  07                  RLC
F08A  81                  ADD   C
F08B  27                  DAA
F08C  D2 90 F0            JNC   F090
F08F  04                  INR   B
F090  12            ZZZ:  STAX  D             ;f251<-20
F091  1B                  DCX   D
F092  7E                  MOV   A,M
F093  E6 F0               ANI   F0
F095  0F                  RRC
F096  0F                  RRC
F097  0F                  RRC
F098  0F                  RRC
```

```
F099  4F                    MOV   C,A     ;c<-03
F09A  2B                    DCX   H
F09B  7E                    MOV   A,M
F09C  0F                    RRC
F09D  0F                    RRC
F09E  0F                    RRC
F09F  0F                    RRC
F0A0  B1                    ORA   C
F0A1  80                    ADD   B
F0A2  27                    DAA
F0A3  4F                    MOV   C,A           ;c<-14
F0A4  2B                    DCX   H
F0A5  2B                    DCX   H
F0A6  7E                    MOV   A,M
F0A7  81                    ADD   C             ;a<-16
F0A8  27                    DAA
F0A9  12                    STAX  D
F0AA  CF                    RST   1
F0AB  AF              S1:   XRA   A             ;rtn to perform bcd multiplication
F0AC  81              RPT:  ADD   C
F0AD  27                    DAA
F0AE  05                    DCR   B
F0AF  C2 AC F0              JNZ   F0AC
F0B2  C9                    RET
F0B3  E5              S2:   PUSH  H             ;rtn to store the result of multiplication
F0B4  26 00                 MVI   H,0           ; in such a sway as to perform addition
F0B6  78                    MOV   A,B
F0B7  E6 0F                 ANI   0F
F0B9  07                    RLC
F0BA  07                    RLC                 ; 45  ;rhis rtn covers the
F0BB  07                    RLC                 ; X36 30 & 24 obtained as product
F0BC  07                    RLC                 ;     6X5 and 6X4 t0 0270
F0BD  81                    ADD   C             ;-------
F0BE  27                    DAA                 ; 0270
F0BF  D2 C3 F0              JNC   F0C3    ; 0135
F0C2  24                    INR   H             ;-------
F0C3  4F              CONT: MOV   C,A           ; 1620
F0C4  78                    MOV   A,B           ;-------
F0C5  E6 F0                 ANI   F0
F0C7  0F                    RRC
F0C8  0F                    RRC
F0C9  0F                    RRC
F0CA  0F                    RRC
F0CB  84                    ADD   H
F0CC  47                    MOV   B,A
F0CD  E1                    POP   H
F0CE  C9                    RET
```

26. Prg26

Program to generate triangular waveform
Time period 10 msecs
Connect nifc-06 to 26 pin connector P2 in 8085 trainers

```
00F0                        PORTA  EQU   F0
00F3                        CNTRL  EQU   F3

F000  3E 80                 MVI    A,80
F002  D3 F3                 OUT    F3
F004  3E 00          LOOP2: MVI    A,00
F006  D3 F0          LOOP:  OUT    F0
F008  00                    NOP
F009  00                    NOP
F00A  00                    NOP
F00B  00                    NOP
F00C  00                    NOP
F00D  3C                    INR    A
F00E  FE FF                 CPI    FF
F010  C2 06 F0              JNZ    F006
F013  D3 F0          LOOP1: OUT    F0
F015  00                    NOP
F016  00                    NOP
F017  00                    NOP
F018  00                    NOP
F019  00                    NOP
F01A  3D                    DCR    A
F01B  C2 13 F0              JNZ    F013
F01E  C3 04 F0              JMP    F004
```

27. Prg27

Program to generate sine waveform with magnitude of 5volts
Time period 10msecs
X = 128 + 128 SIN 0 -90 DEGREES TO +90 DEGREES
EVERY 5 DEGREE 'X' IS CALCULATED
Connect nifc-06 to 26 pin connector 'P2' in 8085 trainer

```
00F0                        PORTA  EQU   F0
00F3                        CNTRL  EQU   F3

F000  3E 80          ST0:   MVI    A,80
F002  D3 F3                 OUT    F3
F004  21 2E F0       RPT:   LXI    H,F02E
F007  0E 24                 MVI    C,36
F009  7E             LOOP1: MOV    A,M
F00A  D3 F0                 OUT    F0
F00C  CD 24 F0              CALL   F024
F00F  23                    INX    H
F010  0D                    DCR    C
F011  C2 09 F0              JNZ    F009
F014  0E 24                 MVI    C,36
F016  2B             LOOP2: DCX    H
F017  7E                    MOV    A,M
```

```
F018  D3 F0              OUT   F0
F01A  CD 24 F0           CALL  F024
F01D  0D                 DCR   C
F01E  C2 16 F0           JNZ   F016
F021  C3 04 F0           JMP   F004
F024  16 0E       DELAY: MVI   D,0E
F026  00             L:  NOP
F027  00                 NOP
F028  00                 NOP
F029  15                 DCR   D
F02A  C2 26 F0           JNZ   F026
F02D  C9                 RET
F02E            LKUPTBL:
F02E  00 01 02 04 08 DB  0,1,2,4,8,12,17,23,30,37,46,54,64,74,84,95,105,116,128
F033  0C 11 17 1E 25
F038  2E 36 40 4A 54
F03D  5F 69 74 80
F041  8B 96 A1 AB B6     DB
                         139,150,161,171,182,192,201,210,218,226,232,238,244,248
F046  C0 C9 D2 DA E2
F04B  E8 EE F4 F8
F04F  FB FE FF           DB    251,254,255
```

28. Prg28

Program to generate square waveform of 50% duty cycle
Time period 12 msecs magnitude 5Volts
Duty cycle = Ton/(Ton+Toff)=0.5 = 0.5X12msec = 6msec = Toff
Connect nifc-06 to 26 pin connector 'P2' in 8085 trainers

```
00F0                     PORTA  EQU   F0
00F3                     CNTRL  EQU   F3

F000  3E 80              MVI    A,80
F002  D3 F3              OUT    F3
F004  3E FF       LOOP:  MVI    A,FF
F006  D3 F0              OUT    F0
F008  CD 15 F0           CALL   F015
F00B  3E 00              MVI    A,00
F00D  D3 F0              OUT    F0
F00F  CD 15 F0           CALL   F015
F012  C3 04 F0           JMP    LOOP
F015              DELAY:
F015  16 03              MVI    D,03
F017  1E 00              MVI    E,00
F019  1B          DL05:  DCX    D
F01A  7A                 MOV    A,D
F01B  B3                 ORA    E
F01C  C2 19 F0           JNZ    F019
F01F  C9                 RET
```

29. Prg29

Program to generate rectangular waveform of 60% duty cycle
Time period 10 msecs magnitude 5Volts
Duty cycle = Ton/(Ton+Toff)=0.6 = 0.6X10msec = 6msec Toff=4msec
Connect nifc-06 to 26 pin connector 'P2' in 8085 trainers

```
00F0                        PORTA  EQU    F0
00F3                        CNTRL  EQU    F3

F000  3E 80                 MVI    A,80
F002  D3 F3                 OUT    F3
F004  3E FF          LOOP:  MVI    A,FF
F006  D3 F0                 OUT    F0
F008  CD 15 F0              CALL   F015
F00B  3E 00                 MVI    A,00
F00D  D3 F0                 OUT    F0
F00F  CD 20 F0              CALL   F020
F012  C3 04 F0              JMP    F004
F015                DELAY1:
F015  16 03                 MVI    D,03
F017  1E 00                 MVI    E,00
F019  1B            DL05:   DCX    D
F01A  7A                    MOV    A,D
F01B  B3                    ORA    E
F01C  C2 19 F0              JNZ    F019
F01F  C9                    RET
F020                DELAY2:
F020  16 02                 MVI    D,02
F022  1E 00                 MVI    E,00
F024  1B            DL15:   DCX    D
F025  7A                    MOV    A,D
F026  B3                    ORA    E
F027  C2 24 F0              JNZ    F024
F02A  C9                    RET
```

30. Prg30

Program to generate saw tooth waveform
Time period 5 msecs magnitude 5Volts
Connect nifc-06 to 26 pin connector 'P2' in 8085 trainers

```
00F0                        PORTA  EQU    F0
00F3                        CNTRL  EQU    F3

F000  3E 80                 MVI    A,80
F002  D3 F3                 OUT    F3
F004  3E 00          LOOP2: MVI    A,00
F006  D3 F0          LOOP:  OUT    F0
F008  CD 15 F0              CALL   F015
F00B  00                    NOP
F00C  3C                    INR    A
F00D  FE FF                 CPI    0FF
F00F  C2 06 F0              JNZ    F006
F012  C3 04 F0              JMP    F004
F015                DELAY:
```

```
F015  0E 03              MVI   C,03
F017  00          DL05:  NOP
F018  0D                 DCR   C
F019  C2 17 F0           JNZ   F017
F01C  C9                 RET
```

31. Prg31

Program to arrange in ascending and descending order of given list of no note: The ascending and descending of given no. is done through BUBBLE SORT method i.e., if there are 'n' no. to be sorted then (n-1)*n times it has to be sorted

```
F000  AF            ST:    XRA   A
F001  06 0A                MVI   B,0AH     ;the value of 'n' the values
F003  21 00 E1      STAT:  LXI   H,E100    ;are stored for ascending
F006  0E 0A                MVI   C,0A
F008  0D                   DCR   C         ;value of 'n-1'
F009  7E            CONT:  MOV   A,M       ;ACC has the contents of memory
F00A  23                   INX   H         ;increment HL
F00B  BE                   CMP   M         ;ACC is compared with
                                           ;memory content
F00C  DA 17 F0             JC    F017      ;if content of ACC is small then go
                                           ;to next sort
F00F  CA 17 F0             JZ    F017
F012  56                   MOV   D,M       ;if ACC content is greater than
                                           ;memory content
F013  77                   MOV   M,A       ;place the smaller value in x and
                                           ;larger value in x+1h mem. locn.
F014  2B                   DCX   H
F015  72                   MOV   M,D
F016  23                   INX   H         ;increment x position
F017  0D            NOCH:  DCR   C         ;decrement the counter after the
                                           ;placing is over
F018  C2 09 F0             JNZ   F009
F01B  05                   DCR   B         ;decrement the counter
F01C  C2 03 F0             JNZ   F003
F01F  CF                   RST   1

F020  21 50 F0      STT:   LXI   H,F050    ; the values are stored for descending
F023  0E 0A                MVI   C,0A
F025  0D                   DCR   C
F026  7E            REPEAT: MOV  A,M
F027  23                   INX   H
F028  BE                   CMP   M
F029  D2 34 F0             JNC   F034
F02C  CA 34 F0             JZ    F034      ;NOTE: the conditional check(JC) is
                                           ;replaced by conditional check
F02F  56                   MOV   D,M
F030  77                   MOV   M,A
F031  2B                   DCX   H
F032  72                   MOV   M,D
F033  23                   INX   H
F034  0D            NOXHG: DCR   C
F035  C2 26 F0             JNZ   F026
F038  05                   DCR   B
```

```
F039  C2 20 F0                  JNZ   F020
F03C  CF                        RST   1

F100                            ORG   F100H
F100  23 76 02 56 89      ARR:  DB    23H,76H,02H,56H,89H,33H,99H,69H,98H,45H
F105  33 99 69 98 45
```

INPUT: given at location F100H.
RESULT: Ascending order are at location E100H after executing the program at F000H
RESULT: Descending order are at location F050H after executing the program at F020H

32. Prg32

8 bit subtraction program
Input x 8100 y 8101
Output: x - y at 8102 8103 =1-> the result is in 2's compliment
Repeated addition is used to perform multiplication

```
F000  21 00 81                  LXI   H,8100    ;the no. to be subtracted is stored
F003  4E                        MOV   C,M       ;at F100H and the minuend is stored
F004  23                        INX   H         ;at F101H. the result is stored at
F005  7E                        MOV   A,M       ;8102H. A 1 at F103H indicates that
F006  2F                        CMA             ;the result is in 2's complement form
F007  C6 01                     ADI   01
F009  81                        ADD   C
F00A  D2 0E F0                  JNC   F00E
F00D  04                        INR   B
F00E  23                CONT:   INX   H
F00F  77                        MOV   M,A
F010  23                        INX   H
F011  70                        MOV   M,B
F012  CF                        RST   1
```

33. Prg33

Data transfer two bytes of data in consecutive memory location
Starting at f020h
Input: F010 & F011
Output: F020 & F021

```
F000  21 10 F0                  LXI   H,F010    ;memory location
F003  7E                        MOV   A,M       ;get the data from accumulator
F004  32 20 F0                  STA   F020      ;store at this location
F007  23                        INX   H         ;get the next address
F008  7E                        MOV   A,M
F009  32 21 F0                  STA   F021
F00C  CF                        RST   1
```

If data at F010=10, F011=20 result at F020=10, F021=20

34. Prg34

Data tests zero condition of a data byte memory location
Starting at f020h
Input: F100
Output: F101 = 00 if the byte is zero else =FF

```
F000  21 00 F1              LXI   H,F100
F003  7E                    MOV   A,M
F004  B7                    ORA   A
F005  CA 0A F0              JZ    F00A
F008  3E FF                 MVI   A,FF
F00A  32 01 F1    ST05:     STA   F101
F00D  CF                    RST   1
```

35. Prg35

Program to find the square of a number (between 0 and 7)
Using lookup table
Input: F100
Output: F101 = square of the number

```
F000  3A 00 F1              LDA   F100    ;get the no. whose square is to be found
F003  21 30 F0              LXI   H,F030  ;get the start address of the lookup table
F006  85                    ADD   L       ;add data to the starting address
F007  6F                    MOV   L,A     ;point H,L to appropriate answer
F008  7E                    MOV   A,M     ;get the required square
F009  32 01 F1              STA   F101    ;store it in the memory location
F00C  CF                    RST   1

F030                        ORG   F030
F030  00 01 04 09 10 SQRS:  DB    0,1,4,9,16,25,36,49
F035  19 24 31
```

EX:
Data at F100 = 05H
Result at F101 = 19H

36. Prg36

Program to find the number of nonblank character in a string
The string is terminated by ASCII period (2EH)
An ASCII blank is 20h
Input: F101 onwards
Output: F100 = number of blanks in the string

```
F000  21 01 F1              LXI   H,F101    ;initialise the pointer
F003  06 00                 MVI   B,00      ;initialise the counter
F005  7E          LOOP:     MOV   A,M       ;get the data
F006  23                    INX   H         ;point to the next location
F007  FE 20                 CPI   20        ;check whether the data is blank
F009  CA 14 F0              JZ    F014
F00C  FE 2E                 CPI   2E        ;check for end of string
F00E  CA 18 F0              JZ    F018
```

```
F011  C3 05 F0            JMP   F005
F014  04          NEXT1:  INR   B
F015  C3 05 F0            JMP   F005
F018  78          NEXT2:  MOV   A,B        ;load the count to accumulator
F019  32 00 F1            STA   F100       ;store no. of non-blank character
F01C  CF                  RST   1

F101                      ORG   F101H
F101  4A 41 43 4B 20      STRS: DB    'JACK AND THE JILL WENT UP THE HILL.'
F106  41 4E 44 20 54
F10B  48 45 20 4A 49
F110  4C 4C 20 57 45
F115  4E 54 20 55 50
F11A  20 54 48 45 20
F11F  48 49 4C 4C 2E
```

; Listing of test programs

; ROM Test Routine

; This program has been written to check the SDA-85MEL system monitor using a checksum comparison method. This program can also be used to check a region of mem specified by a starting and ending addr when the chsum is known. The starting and ending address are entered at the kbd as also the chsum and an error is reported if it does not agree. The computed chsum is available in locations FF9E, FF9F lo-byte first.

```
                          ORG   F000H       ;starting addrs to test ROM
ROMCK:
F000  CD A9 14            CALL  CLDIS1
F003  CD AF 05            CALL  DISUB      ;get two addrs in BC & DE
F006  C5                  PUSH  B
F007  E1                  POP   H          ;first addr to HL
F008  22 70 FF            SHLD  STADD      ;store it
F00B  EB                  XCHG             ;get second addr to HL
F00C  23                  INX   H          ;inc to adjust for term
F00D  22 72 FF            SHLD  ENADD1     ;store this end addr
F010  CD A9 14            CALL  CLDIS1
F013  3E C7               MVI   A,0C7H
F015  32 C2 FF            STA   NADD1
F018  21 68 F0            LXI   H,CHMSG
F01B  16 09               MVI   D,09
F01D  06 80               MVI   B,80H
F01F  CD 42 00            CALL  LBL5
F022  06 02               MVI   B,02H
F024  CD 86 05            CALL  DESUB      ;get chsum from kbd
F027  EB                  XCHG             ;get chsum to HL
F028  22 74 FF            SHLD  CHSUAD     ;store chsum
F02B  2A 70 FF            LHLD  STADD      ;retrieve start addr
F02E  06 00               MVI   B,CONST
F030  3E 00               MVI   A,CONST    ;set A and B to zero
F032  86          ROM05:  ADD   M          ;add mem byte to A
F033  D2 37 F0            JNC   ROM10      ;no overflow
F036  04                  INR   B          ;overflow-inc B
F037  23          ROM10:  INX   H          ;inr mem ptr
F038  EB                  XCHG             ;save mem ptr temporarily
F039  2A 72 FF            LHLD  ENADD1     ;obtain end addr in HL
```

```
F03C  4F                      MOV   C,A          ;store sum temporarily in C
F03D  CD 83 06                CALL  HILO         ;det.if it is the last addr
F040  D2 48 F0                JNC   ROM15        ;last addr so go to term
F043  EB                      XCHG               ;more to add-retriv ptr to HL
F044  79                      MOV   A,C          ;ret temp sum
F045  C3 32 F0                JMP   ROM05        ;loop to add
F048  79              ROM15:  MOV   A,C
F049  32 0E FF                STA   TEMP1        ;store lo byte of chsum
F04C  78                      MOV   A,B
F04D  32 0F FF                STA   TEMP1+1      ;store hi byte of chsum
F050  2A 74 FF                LHLD  CHSUAD       ;obtain chsum stored-kbd i/p
F053  CD 37 05                CALL  COMPL        ;complement it to check
F056  09                      DAD   B            ;16 bit add to computed chsum
F057  D2 F1 05                JNC   ERR          ;if it doesn't match it is an err
F05A  7D                      MOV   A,L          ;
F05B  B4                      ORA   H            ;
F05C  C2 F1 05                JNZ   ERR          ;err if result of two chsm add is
F05F  21 72 F0                LXI   H, ROMMSG
F062  06 80                   MVI   B,80H        ;
F064  CD 42 00                CALL  LBL5
F067  76                      HLT                ;not zero-ret to mon if OK
F068  43 48 45 43 4B  CHMSG DB      'CHECK SUM',0DH
F06D  20 53 55 4D 0D
F072  4D 4F 4E 49 54  ROMMSG: DB    'MONITOR ROM TEST CHECKSUM IS OK',0DH
F077  4F 52 20 52 4F
F07C  4D 20 54 45 53
F081  54 20 43 48 45
F086  43 4B 53 55 4D
F08B  20 49 53 20 4F
F090  4B 0D
```

; RAM Test Routine

```
                              ORG   F0A0H  ;starting addrs to test RAM
F0A0  3E 0E           RAMCK:  MVI   A,0EH  ;unmask RST5.5
F0A2  30                      SIM
F0A3  CD A9 14                CALL  CLDIS1
F0A6  CD AF 05          KBD:  CALL  DISUB  ;get two addrs
F0A9  C5                      PUSH  B
F0AA  E1                      POP   H      ;first addrs to HL
F0AB  CD BD F0        STR05:  CALL  STR15
F0AE  CD 83 06                CALL  HILO   ;see if it is last addrs
F0B1  D2 AB F0                JNC   STR05
F0B4  21 CF F0                LXI   H,RAMMSG
F0B7  06 80                   MVI   B,80H        ;
F0B9  CD 42 00                CALL  LBL5
F0BC  76                      HLT          ;all done halt
F0BD  46              STR15:  MOV   B,M    ;store RAM byte temporarily
F0BE  0E 08                   MVI   C,08H  ;C=no of bits to test
F0C0  3E 01                   MVI   A,01H  ;make first bit=1
F0C2  77              STR20:  MOV   M,A    ;write to RAM
F0C3  BE                      CMP   M      ;check if properly written
F0C4  C2 F1 05                JNZ   ERR    ;no-err exit
F0C7  07                      RLC          ;write ok-make next bit=1
F0C8  0D                      DCR   C      ;decr counter
```

```
F0C9   C2 C2 F0                    JNZ    STR20  ;if byte not chcked loop
F0CC   70                          MOV    M,B    ;retrieve original RAM data
F0CD   23                          INX    H      ;move to next RAM location
F0CE   C9                          RET           ;before returning
F0CF   52 41 4D 20 49    RAMMSG:   DB     'RAM IS TESTED OK',0DH
F0D4   53 20 54 45 53
F0D9   54 45 44 20 4F
F0DE   4B 20 20 20 20
F0E3   20 20 20 20 20
F0E8   20 20 20 20 20
F0ED   20 20 0D
```

; Keyboard Test Routine

; This program allows the keys in the SDA 85-MEL to be tested. Each key has a code, which is displayed in the data field when that key is pressed refer to the manual for the code details. The program runs in a loop and is terminated by pressing the reset key.

```
                                   ORG    F100H          ; starting addrs to test keyboard
F100   3E CB             KBT:      MVI    A,0CBH
F102   32 C2 FF                   STA    NADD1
F105   21 1D F1                   LXI    H,KBDMSG  F11D
F108   06 80                      MVI    B,80H
F10A   CD 42 00                   CALL   LBL5
F10D   CD D0 3E          KBT05:    CALL   GET_ASCII      ;get a key
F110   F5                         PUSH   PSW
F111   06 C7                      MVI    B,C7H
F113   CD C2 14                   CALL   SET_DDRAM
F116   F1                         POP    PSW
F117   CD FF 14                   CALL   SND_MSG
F11A   C3 0D F1                   JMP    KBT05  F10D ;loop back to get next key
F11D   4B 45 59 42 4F KBDMSG:     DB     'KEYBOARD TEST CODE = ',0DH
F122   41 52 44 20 54
F127   45 53 54 20 20
F12C   20 20 43 4F 44
F131   45 20 3D 20 0D
```

; Display Test Routine

This program display **WELCOME TO A.L.S TESTING OF LCD DISPLAY** keeps scrolling the program is terminated by pressing RESET key.

```
                                   ORG    F140H          ; starting addrs to test display
F140   21 62 F1          START:    LXI    H,DISP_TEST_MSG
F143   3E 08                      MVI    A,08H
F145   F5                 RPT:    PUSH   PSW
F146   E5                         PUSH   H
F147   06 80                      MVI    B,80H
F149   CD 42 00                   CALL   LBL5
F14C   E1                         POP    H
F14D   23                         INX    H
F14E   11 FF FF                   LXI    D,FFFFH
F151   CD 7F 05                   CALL   DELAY
F154   F1                         POP    PSW
F155   3D                         DCR    A
```

```
F156   C2 45 F1                    JNZ    RPT
F159   11 FF FF                    LXI    D,FFFFH
F15C   CD 7F 05                    CALL   DELAY
F15F   C3 40 F1                    JMP    START
F162                  DISP_TEST_MSG:
F162   57 45 4C 43 4F        DB     'WELCOME TO A.L.S TESTING OF LCD DISPLAY',0DH
F167   4D 45 20 54 4F
F16C   20 41 2E 4C 2E
F171   53 20 54 45 53
F176   54 49 4E 47 20
F17B   4F 46 20 4C 43
F180   44 20 44 49 53
F185   50 4C 41 59 0D
```

; Vector Interrupt Test Routine

; Routine to test VECTR-INTR key. This program stores immediate data in the reg A, B and C whenever a GO to location. 'VECT' is executed and the VECTR-INTR key is pressed. The contents of the reg can then be examined through the EXAM REG command.

```
                                   ORG    F190H        ; start addrs to test VECT.INTR key
F190   21 B1 FF        VECT:      LXI    H,FFB1H      ;initialise RST7.5 transfer pt.
F193   36 C9                      MVI    M,C9H        ;to ret instrn. so that a branch
F195   3E 1B                      MVI    A,1BH        ;to the routine str05 occurs
F197   30                         SIM                 ;unmask RST7.5, enable interrupts
F198   FB                         EI                  ;and halt for the VECT-INTR key
F199   21 A9 F1                   LXI    H,VECTMSG    F1A9 F1
F19C   06 80                      MVI    B,80H
F19E   CD 42 00                   CALL   LBL5  0042
F1A1   76                         HLT                 ;to be pressed
F1A2   3E 11            VECT05:   MVI    A,11H        ;branches here when the key is
F1A4   06 22                      MVI    B,22H        ;pressed, initializes registers A
F1A6   0E 33                      MVI    C,33H        ;B and C to 11,22 and 33 respectely
F1A8   CF                         RST    1            ;ret to monitor
F1A9   56 45 43 54 4F VECTMSG:    DB     'VECTOR INTERRUPT KEY TEST',0DH
F1AE   52 20 49 4E 54
F1B3   45 52 52 55 50
F1B8   54 20 4B 45 59
F1BD   20 54 45 53 54
F1C2   20 20 20 0D
```

; 8255 Test Routine

```
EQUATES
PORT_1A          EQU    F0H          ;8255(u17) port addresses
PORT_1B          EQU    F1H
PORT_1C          EQU    F2H
CTL1             EQU    F3H
PORT_2A          EQU    D8H          ;8255 (u16) port addresses
PORT_2B          EQU    D9H
PORT_2C          EQU    DAH
CTL2             EQU    DBH
                                   ORG    F290H        ; starting addrs to test 8255
F290   3E 0E                       MVI    A,0EH
F292   30                          SIM
```

```
F293  3E 80          TEST:   MVI   A,80H       ;configure u17 ports as o/p ports
F295  D3 F3                  OUT   CTL1
F297  3E 9B                  MVI   A,9BH       ;configure u16 ports as i/p ports
F299  D3 DB                  OUT   CTL2
F29B  3E 55                  MVI   A,55H       ;output alternate 1's and 0's to the
F29D  D3 F0                  OUT   PORT_1A     ;port lines
F29F  D3 F1                  OUT   PORT_1B
F2A1  D3 F2                  OUT   PORT_1C
F2A3  DB D8                  IN    PORT_2A     ;read back and compare
F2A5  FE 55                  CPI   55H
F2A7  C2 F1 05               JNZ   ERR         ;if data output does not tally with the
F2AA  DB D9                  IN    PORT_2B     ;data read back, it is an error
F2AC  FE 55                  CPI   55H
F2AE  C2 F1 05               JNZ   ERR
F2B1  DB DA                  IN    PORT_2C
F2B3  FE 55                  CPI   55H
F2B5  C2 F1 05               JNZ   ERR
F2B8  3E 80                  MVI   A,80H       ;configure u16 ports as o/p ports
F2BA  D3 DB                  OUT   CTL2
F2BC  3E 9B                  MVI   A,9BH       ;configure u16 ports as i/p ports
F2BE  D3 F3                  OUT   CTL1
F2C0  3E AA                  MVI   A,AAH       ;output alternate 1's and 0's to the
F2C2  D3 D8                  OUT   PORT_2A     ;port lines
F2C4  D3 D9                  OUT   PORT_2B
F2C6  D3 DA                  OUT   PORT_2C
F2C8  DB F0                  IN    PORT_1A     ;read back and compare
F2CA  FE AA                  CPI   AAH
F2CC  C2 F1 05               JNZ   ERR
F2CF  DB F1                  IN    PORT_1B     ;if data output does not tally with the
F2D1  FE AA                  CPI   AAH         ;data read back, it is an error
F2D3  C2 F1 05               JNZ   ERR
F2D6  DB F2                  IN    PORT_1C
F2D8  FE AA                  CPI   AAH
F2DA  C2 F1 05               JNZ   ERR
F2DD  CD A9 14               CALL  CLDIS1
F2E0  06 80                  MVI   B,80H
F2E2  21 E9 F2               LXI   H,M8255H    ;display '8 2 5 5' in the address field
F2E5  CD 42 00               CALL  LBL5
F2E8  76                     HLT
F2E9  20 38 32 35 35   M8255H      DB    '8255 TEST PASS',0DH
F2EE  20 54 45 53 54
F2F3  20 50 41 53 53
F2F8  20 20 20 20 20
F2FD  20 20 20 20 20
F302  20 20 20 20 20
F307  20 0D
```

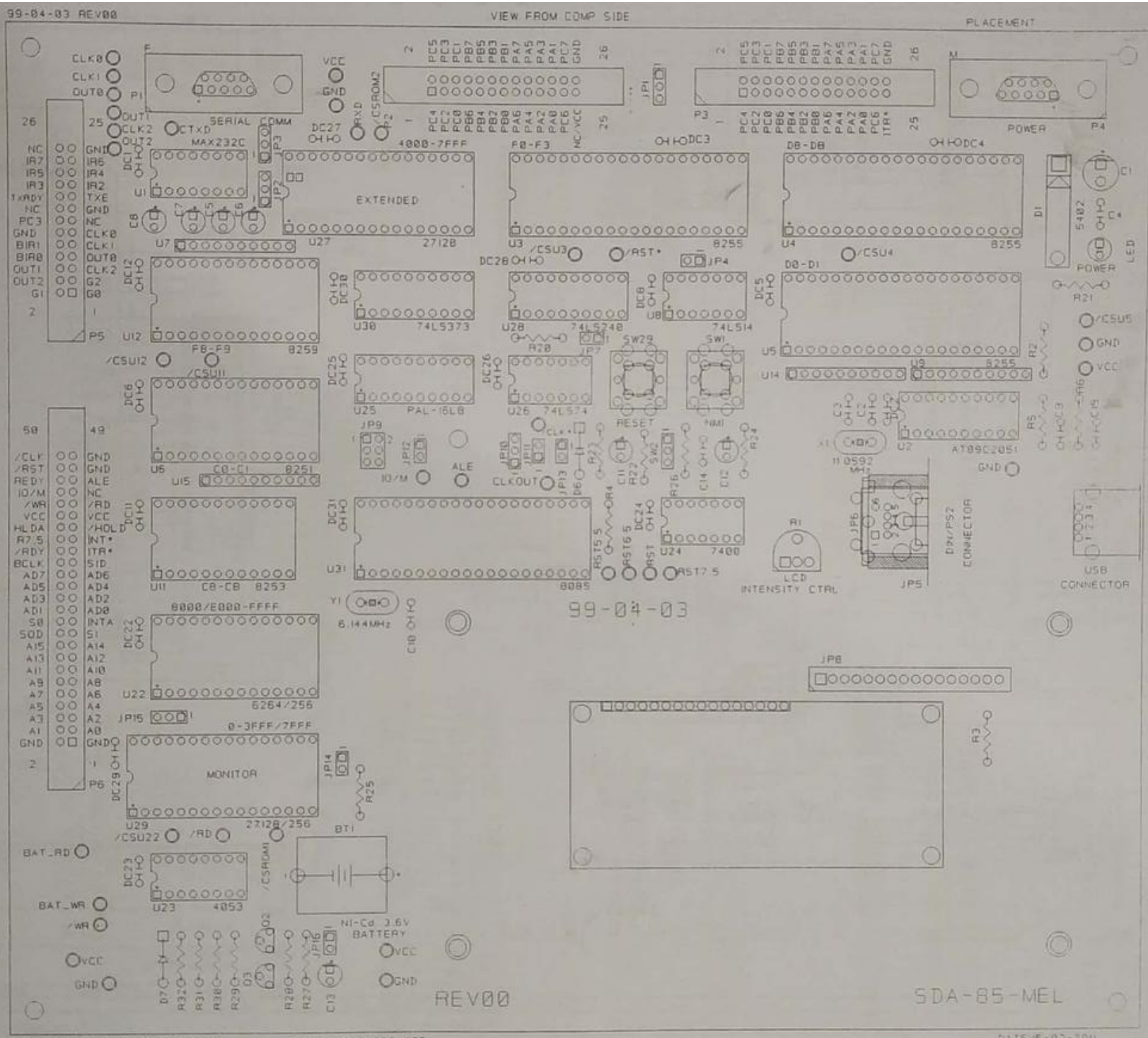
; TEST SOFTWARE TO TEST ON-BOARD 8259

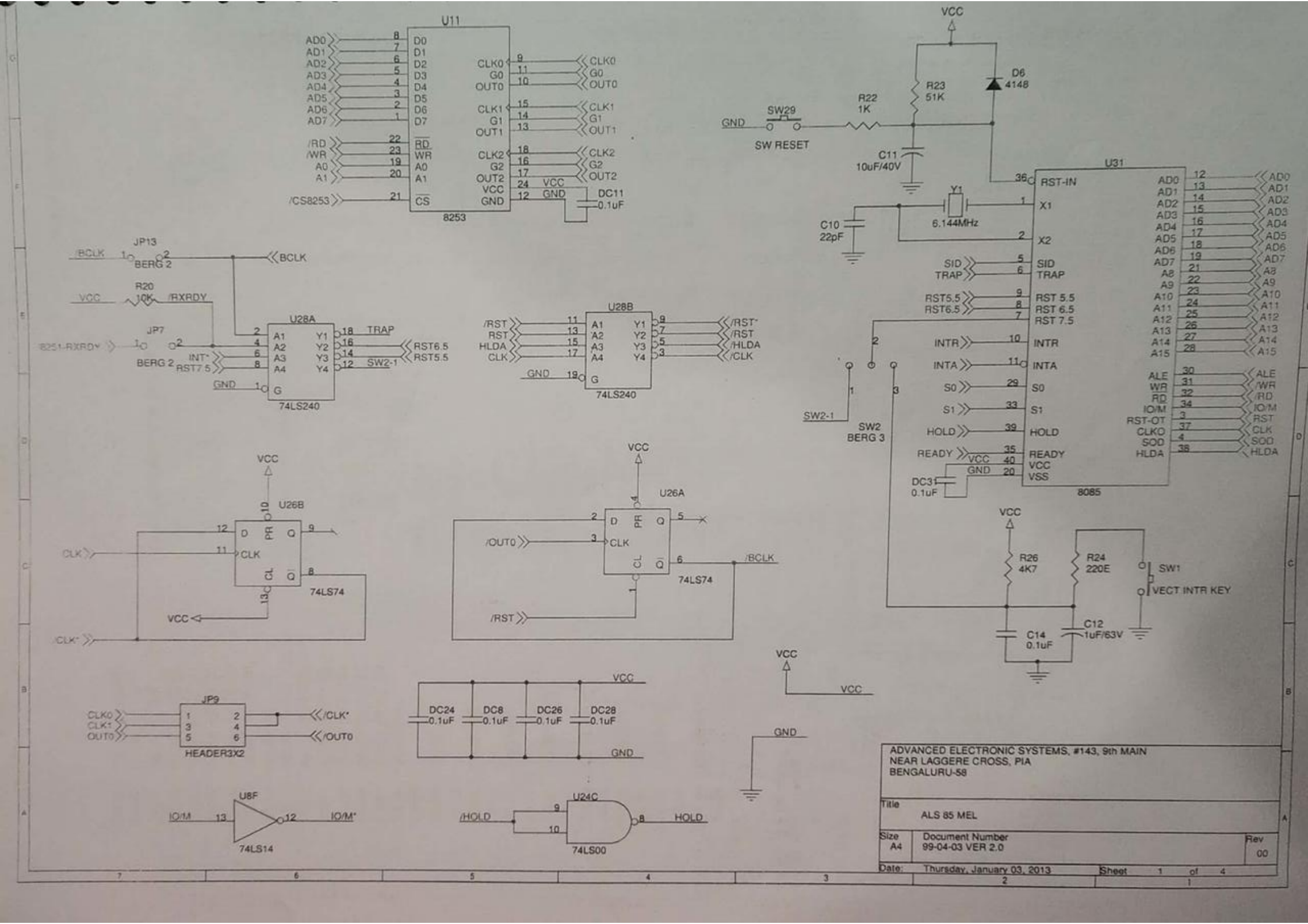
```
CTL             EQU   F8H
DTA             EQU   F9H
UPDDT     EQU   07D1H
STDT            EQU   FFF9H

                ORG   0F3A0H; starting addrs to test 8259
F3A0  06 80     MVI   B,80H
F3A2  21 0F F4  LXI   H,BLS
F3A5  CD 42 00  CALL  42H
F3A8  3E 13     MVI   A,13H
F3AA  D3 F8     OUT   CTL
F3AC  3E F6     MVI   A,F6H
F3AE  D3 F9     OUT   DTA
F3B0  3E 0A     MVI   A,0AH
F3B2  D3 F9     OUT   DTA
F3B4  3E 00     MVI   A,00H
F3B6  D3 F9     OUT   DTA
F3B8  3E 0E     MVI   A,0EH
F3BA  30        SIM
F3BB  FB        EI
F3BC  C3 BC F3  LP:   JMP   LP
F3BF  06 80     INT0: MVI   B,80H
F3C1  21 30 F4  LXI   H,BLS0
F3C4  CD 42 00  CALL  42H
F3C7  FB        EI
F3C8  C9        RET
F3C9  06 80     INT1: MVI   B,80H
F3CB  21 51 F4  LXI   H,BLS1
F3CE  CD 42 00  CALL  42H
F3D1  FB        EI
F3D2  C9        RET
F3D3  06 80     INT2: MVI   B,80H
F3D5  21 72 F4  LXI   H,BLS2
F3D8  CD 42 00  CALL  42H
F3DB  FB        EI
F3DC  C9        RET
F3DD  06 80     INT3: MVI   B,80H
F3DF  21 93 F4  LXI   H,BLS3
F3E2  CD 42 00  CALL  42H
F3E5  FB        EI
F3E6  C9        RET
F3E7  06 80     INT4: MVI   B,80H
F3E9  21 B4 F4  LXI   H,BLS4
F3EC  CD 42 00  CALL  42H
F3EF  FB        EI
F3F0  C9        RET
F3F1  06 80     INT5: MVI   B,80H
F3F3  21 D5 F4  LXI   H,BLS5
F3F6  CD 42 00  CALL  42H
F3F9  FB        EI
F3FA  C9        RET
F3FB  06 80     INT6: MVI   B,80H
F3FD  21 F6 F4  LXI   H,BLS6
```

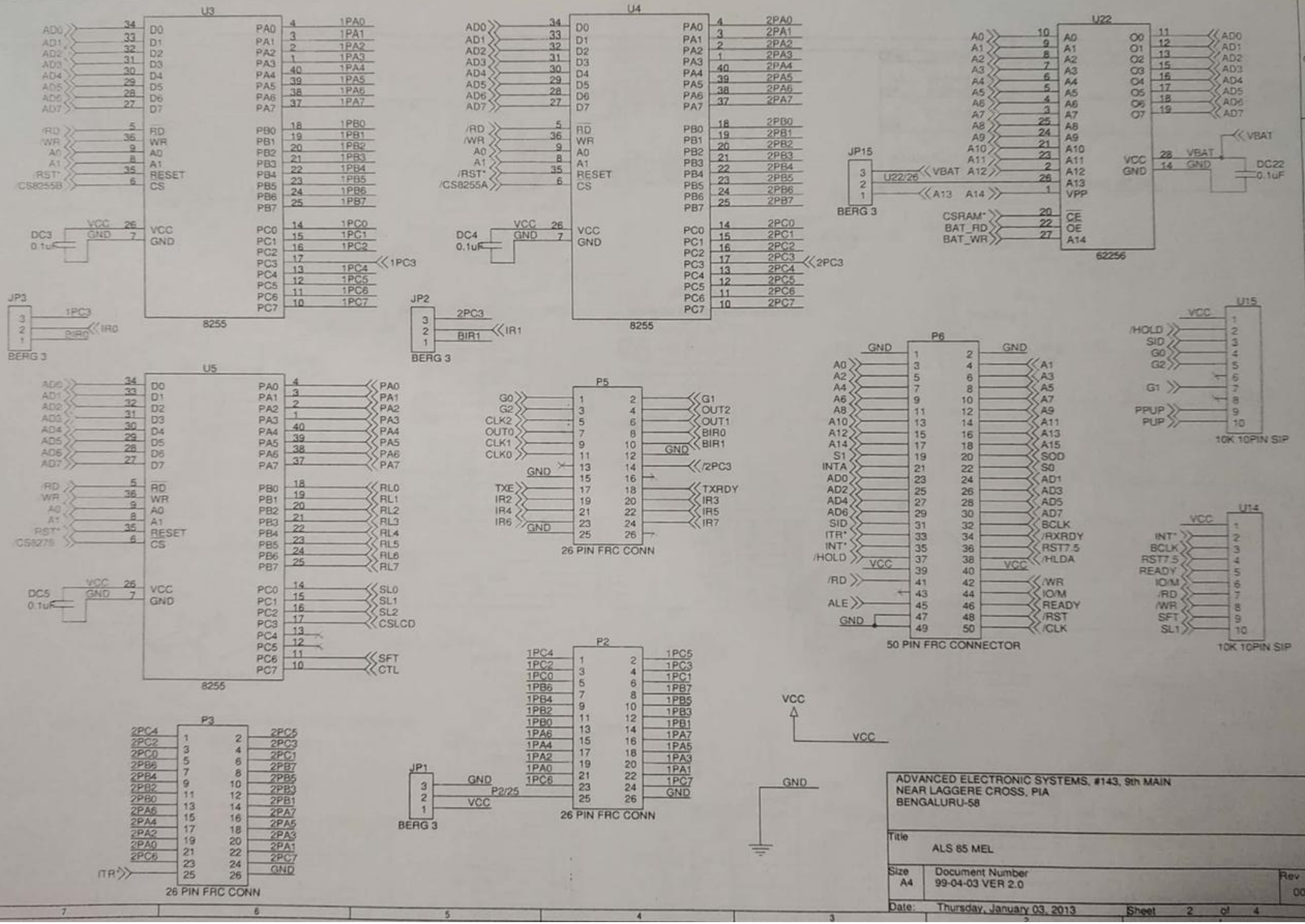
```
F400  CD 42 00                CALL   42H
F403  FB                      EI
F404  C9                      RET
F405  06 80           INT7:   MVI    B,80H
F407  21 17 F5                LXI    H,BLS7
F40A  CD 42 00                CALL   42H
F40D  FB                      EI
F40E  C9                      RET
F40F  20 38 32 35 39    BLS:  DB'8259 INTR TEST',0DH
F414  20 49 4E 54 52
F419  20 54 45 53 54
F41E  20 20 20 20 20
F423  20 20 20 20 20
F428  20 20 20 20 20
F42D  20 20 0D
F430  38 32 35 39  20         BLS0:  DB'8259 INTR TEST 0',0DH
F435  49 4E 54 45 52
F43A  52 55 50 54 20
F43F  30 20 20 20 20
F444  20 20 20 20 20
F449  20 20 20 20 20
F44E  20 20 0D
F451  38 32 35 39  20         BLS1:  DB'8259 INTR TEST 1',0DH
F456  49 4E 54 45 52
F45B  52 55 50 54 20
F460  31 20 20 20 20
F465  20 20 20 20 20
F46A  20 20 20 20 20
F46F  20 20 0D
F472  38 32 35 39 20    BLS2: DB'8259 INTR TEST 2',0DH
F477  49 4E 54 45 52
F47C  52 55 50 54 20
F481  32 20 20 20 20
F486  20 20 20 20 20
F48B  20 20 20 20 20
F490  20 20 0D
F493  38 32 35 39  20         BLS3:  DB'8259 INTR TEST 3',0DH
F498  49 4E 54 45 52
F49D  52 55 50 54 20
F4A2  33 20 20 20 20
F4A7  20 20 20 20 20
F4AC  20 20 20 20 20
F4B1  20 20 0D
F4B4  38 32 35 39  20         BLS4:  DB'8259 INTR TEST 4',0DH
F4B9  49 4E 54 45 52
F4BE  52 55 50 54 20
F4C3  34 20 20 20 20
F4C8  20 20 20 20 20
F4CD  20 20 20 20 20
F4D2  20 20 0D
F4D5  38 32 35 39  20         BLS5:  DB'8259 INTR TEST 5',0DH
F4DA  49 4E 54 45 52
F4DF  52 55 50 54 20
F4E4  35 20 20 20 20
F4E9  20 20 20 20 20
```

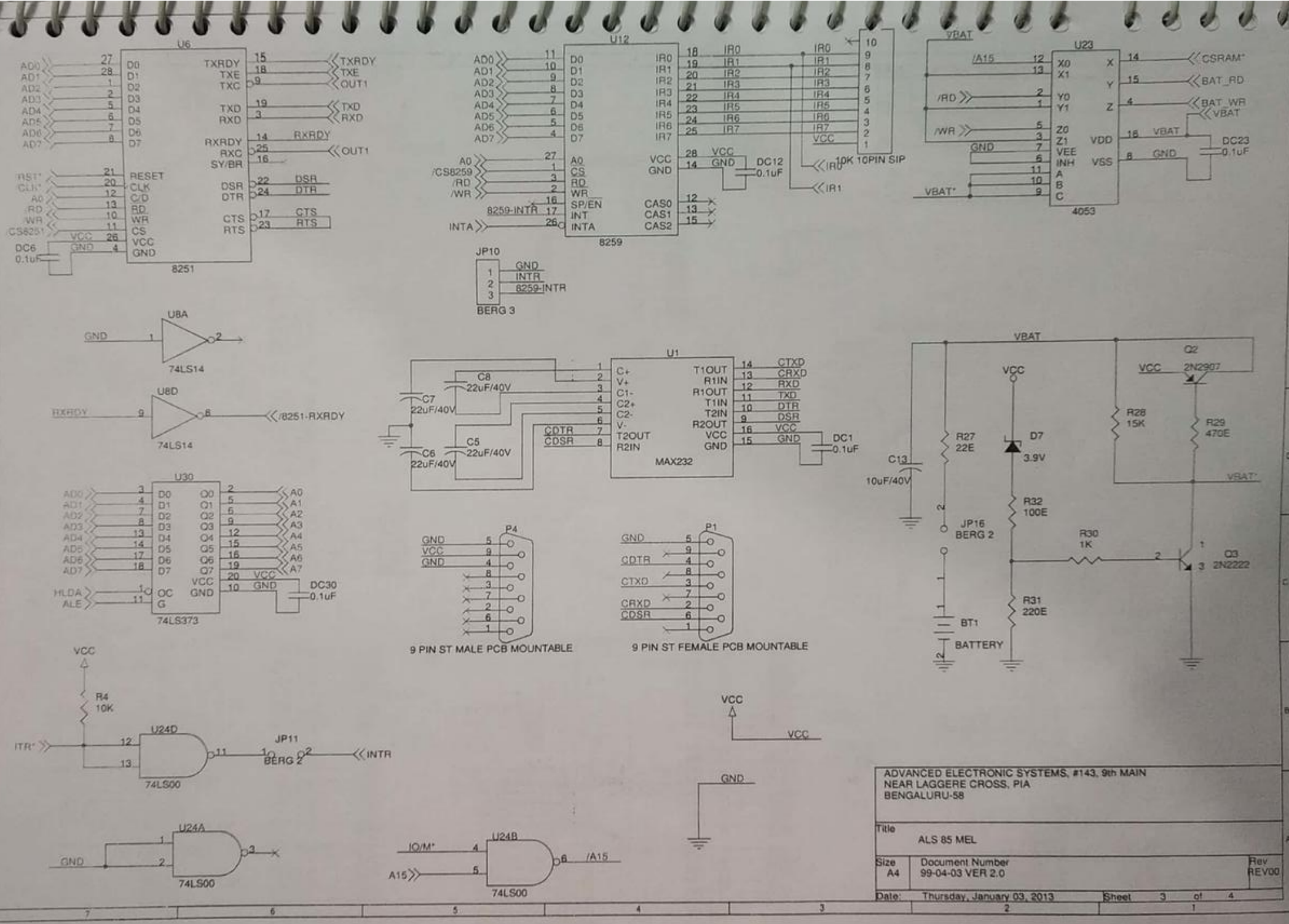
```
F4EE   20 20 20 20 20
F4F3   20 20 0D
F4F6   38 32 35 39    20          BLS6:  DB'8259 INTR TEST 6',0DH
F4FB   49 4E 54 45 52
F500   52 55 50 54 20
F505   36 20 20 20 20
F50A   20 20 20 20 20
F50F   20 20 20 20 20
F514   20 20 0D
F517   38 32 35 39    20          BLS7:  DB'8259 INTR TEST 7',0DH
F51C   49 4E 54 45 52
F521   52 55 50 54 20
F526   37 20 20 20 20
F52B   20 20 20 20 20
F530   20 20 20 20 20
F535   20 20 0D
F600                               ORG    F600H
F600   C3 BF F3                    JMP    INT0
F608                               ORG    F608H
F608   C3 C9 F3                    JMP    INT1
F610                               ORG    F610H
F610   C3 D3 F3                    JMP    INT2
F618                               ORG    F618H
F618   C3 DD F3                    JMP    INT3
F620                               ORG    F620H
F620   C3 E7 F3                    JMP    INT4
F628                               ORG    F628H
F628   C3 F1 F3                    JMP    INT5
F630                               ORG    F630H
F630   C3 FB F3                    JMP    INT6
F638                               ORG    F638H
F638   C3 05 F4                    JMP    INT7
F63B                               END
```

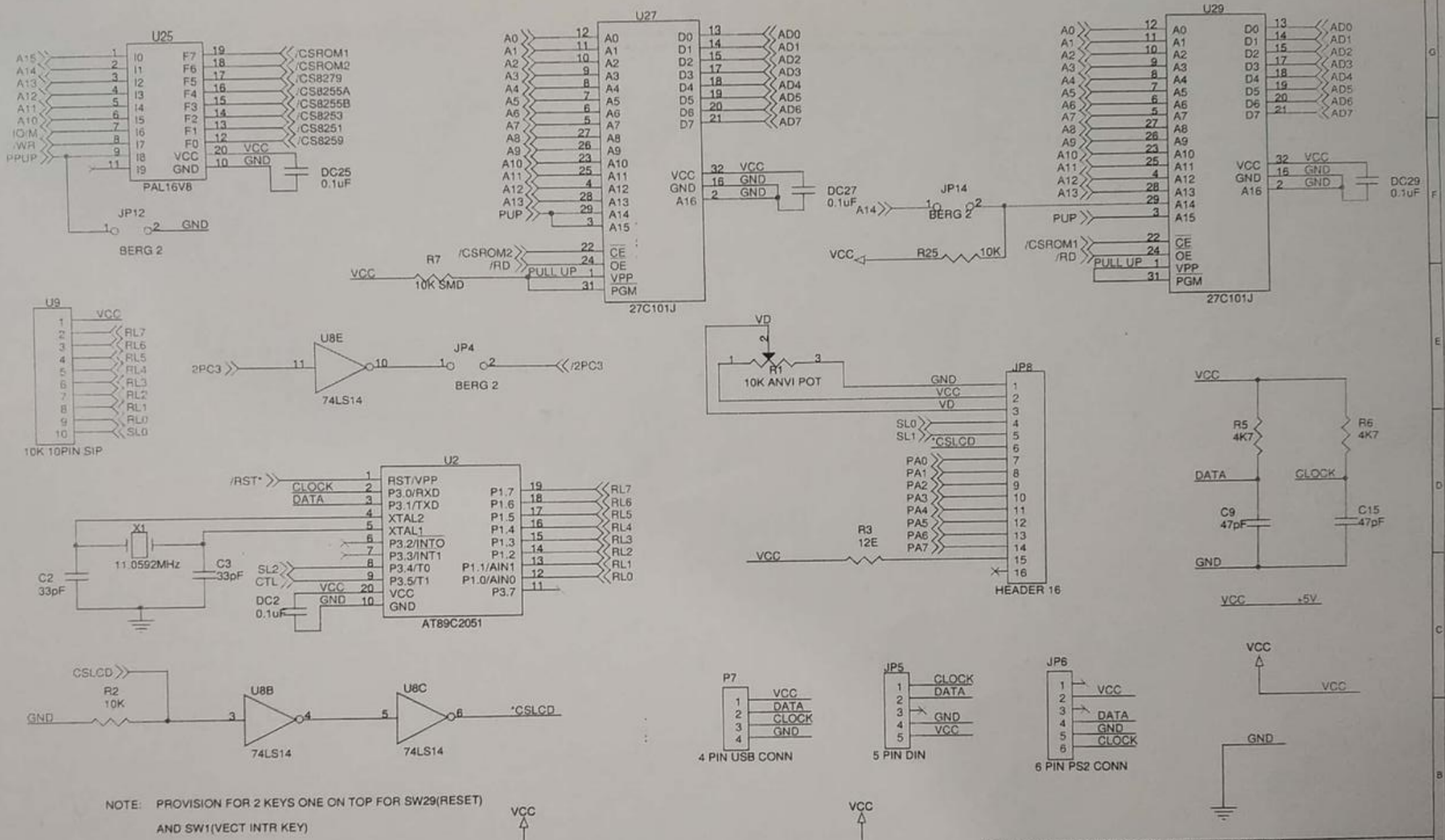
****


99-04-03 REV00
VIEW FROM COMP SIDE
PLACEMENT
SERIAL COMM
MAX232C
EXTENDED
27128
8255
74LS373
74LS240
74LS14
PAL-16L8
74LS74
RESET
NMI
AT89C2051
11.0592
8251
8259
74LS74
7400
8085
6264/256
MONITOR
27128/256
4053
NI-Cd 3.6V BATTERY
LCD INTENSITY CTRL
DIN/PS2 CONNECTOR
USB CONNECTOR
POWER
6.144MHz
99-04-03
REV00
SDA-85-MEL


U11
8253
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
/RD
/WR
A0
A1
/CS8253
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
CS
CLK0
G0
OUT0
CLK1
G1
OUT1
CLK2
G2
OUT2
VCC
GND
CLK0
G0
OUT0
CLK1
G1
OUT1
CLK2
G2
OUT2
DC11
0.1uF
VCC
R23
51K
D6
4148
R22
1K
SW29
SW RESET
C11
10uF/40V
Y1
6.144MHz
C10
22pF
U31
8085
RST-IN
X1
X2
SID
TRAP
RST 5.5
RST 6.5
RST 7.5
INTR
INTA
S0
S1
HOLD
READY
VCC
VSS
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A8
A9
A10
A11
A12
A13
A14
A15
ALE
/WR
/RD
IO/M
RST-OT
CLKO
SOD
HLDA
DC31
0.1uF
SW2-1
SW2
BERG 3
JP13
BERG 2
/BCLK
BCLK
R20
10K
/RXRDY
JP7
8251-RXRDY
INT*
RST7.5
U28A
74LS240
TRAP
RST6.5
RST5.5
SW2-1
U28B
74LS240
/RST*
/RST
/HLDA
/CLK
U26B
74LS74
CLK
/CLK*
U26A
74LS74
/OUT0
/BCLK
/RST
R26
4K7
R24
220E
SW1
VECT INTR KEY
C14
0.1uF
C12
1uF/63V
JP9
HEADER3X2
CLK0
CLK1
OUT0
/CLK*
/OUT0
DC24
0.1uF
DC8
0.1uF
DC26
0.1uF
DC28
0.1uF
GND
U8F
74LS14
IO/M
IO/M*
U24C
74LS00
/HOLD
HOLD
ADVANCED ELECTRONIC SYSTEMS, #143, 9th MAIN
NEAR LAGGERE CROSS, PIA
BENGALURU-58
Title
ALS 85 MEL
Size
A4
Document Number
99-04-03 VER 2.0
Rev
00
Date:
Thursday, January 03, 2013
Sheet
1
of
4


U3
8255
U4
8255
U22
62256
JP15
BERG 3
JP3
BERG 3
JP2
BERG 3
U5
8255
P5
26 PIN FRC CONN
P6
50 PIN FRC CONNECTOR
U15
10K 10PIN SIP
U14
10K 10PIN SIP
P3
26 PIN FRC CONN
P2
26 PIN FRC CONN
JP1
BERG 3
VCC
GND
ADVANCED ELECTRONIC SYSTEMS, #143, 9th MAIN
NEAR LAGGERE CROSS, PIA
BENGALURU-58
Title
ALS 85 MEL
Size
A4
Document Number
99-04-03 VER 2.0
Rev
00
Date:
Thursday, January 03, 2013
Sheet
2
of
4


U6
8251
U12
8259
U23
4053
U8A
74LS14
U8D
74LS14
U30
74LS373
U1
MAX232
P4
9 PIN ST MALE PCB MOUNTABLE
P1
9 PIN ST FEMALE PCB MOUNTABLE
JP10
BERG 3
JP16
BERG 2
JP11
BERG 2
10K 10PIN SIP
BT1
BATTERY
Q2
2N2907
Q3
2N2222
D7
3.9V
U24D
74LS00
U24A
74LS00
U24B
74LS00
ADVANCED ELECTRONIC SYSTEMS, #143, 9th MAIN
NEAR LAGGERE CROSS, PIA
BENGALURU-58
Title
ALS 85 MEL
Size
A4
Document Number
99-04-03 VER 2.0
Rev
REV00
Date:
Thursday, January 03, 2013
Sheet
3
of
4



ADVANCED ELECTRONIC SYSTEMS, #143, 9th MAIN
NEAR LAGGERE CROSS, PIA
BENGALURU-58

Title	ALS 85 MEL	
Size A4	Document Number 99-04-03 VER 2.0	Rev REV00
Date: Thursday, January 03, 2013	Sheet 4 of 4	