User's Manual

FOR

ET-8051LCD-APL
8051 MICROCONTROLLER KIT

Excel Technologies

C-92, Sector - 63, Noida, U.P. 201309, India
Ph : 0120 - 4318572, 08860106750
www.exceltechnologiesonline.in
Email : exceltechnologies.piplani@gmail.com

INDEX

CHAPTER – 1
KNOW YOUR SYSTEM

1.1 BRIEF INTRODUCTION

ET-8051 LCD-APL is a single board Micro-controller Training Kit configured around Intel's 8 bit Micro-controller 8031. This kit has been designed to provide ease in interaction with the 8-bit processor based Micro-controller to enable the students to learn about its architecture, Its capabilities and Applications. 8031 CPU can also be replaced by 8051 CPU. Various applications like Stepper Motor Control, Digital Input and output and Traffic Light Controller and Graphical LCD have been provided on the Board of the Kit.

The Kit communicates with the outside world through a keyboard having an ASCII Keyboard and a LCD display. The Kit can also interact with CRT Terminal or IBM compatible PC/AT Computer System.

The Kit is packed up with powerful monitor in 64K Bytes of factory programmed EPROM and 8K / 32K Bytes of Read/Write Memory and scratch pad memory. The total memory on the board can be easily expanded further using the same sockets as the chips of higher capacity can be used. The system has 48 programmable I/O lines. The serial I/O Communication is made possible through RS232C using 8251.

For control applications, three 16bit Timers / Counters are available through 8253. The Kit provides onboard battery back up for on board RAM. This saves the user's program in case of power failure.

The onboard resident system monitor software is very powerful. It provides various software commands like BLOCK MOVE, Examine/Substitute Memory/Register, FILL, Break Point, Single Step etc. which are very helpful in debugging/ developing software. An onboard line assembler is also provided on the kit to allow the students to directly assemble their programs.

The Kit also provides on board Optional interfaces for Real Time Clock Chip. The Address, data and Control Signals are available on the 50 pin FRC Connector for further expansion of the system.

1.2 SYSTEM SPECIFICATIONS (HARDWARE)

CPU:	Intel 8031/8051, 8bit Micro-controller
EPROM:	64K Bytes of EPROM Loaded with monitor expandable

	Further using 27010
RAM:	32K Bytes of CMOS RAM using 62256.
I/O:	48 Programmable I/O lines through Two Nos. of 8255.
Serial:	EIA RS-232-C through 8251.(On Demand) USB Serial Interface (Standard)
Timer / Counter:	Three 16bit timer/counters through 8253.
Keyboard:	IBM compatible Keyboard.
Display:	16*2 LCD Display, **(20*2 / 20*4 LCD / 40*2 Display are optional).**
OPTIONAL Interfaces:	EPROM PROGRAMMER FOR 2764/27128/27256 Speaker Interface Cassette Interface RTC (Real Time Clock)
BUS:	All address, data and control signals are available at 50 pin FRC Connector.
Power Supply:	5V, 1.5 Amps for kit and Serial operations.
Temperature	0 to 50^0C

1.3 <u>SYSTEM SPECIFICATIONS (SOFTWARE)</u>

The Kit provides various software commands to achieve the following:

A) <u>KEYBOARD MODE:</u>

1. Examine / Modify the External memory Byte locations.
2. Examine / Modify the Internal Memory Byte locations.
3. Examine / Modify the contents of any of Internal Register of 8031/51.
4. Move a block of Data / Program from one location to another location.
5. Fill a particular memory area with a constant.

6. Compare a Memory Block with another Block
7. Compare a Memory Block with a constant.
8. Test the RAM area for being OK.
9. To execute the program in full clock speed.
10. To execute the program instruction at a time (Single Step Mode).
11. To execute the Program with Break Point.
12. UP Loading a Program from the Kit to the PC/AT System.
13. Down loading a program form the PC/AT to the kit.
14. Assemble a program in the memory Area.
15. Check the contents of an EPROM for Blank (Blank Check) (Optional)
16. List the contents of an EPROM in to RAM area. (Optional)
17. Verify the contents of an EPROM with any memory area. (Optional)
18. Program/Duplicate an EPROM. (Optional)

B) SERIAL MODE:

1. Display / Modify an External memory location.
2. Display / Modify an Internal Memory location.
3. Display / Modify Internal Registers of 8031/51.
4. Dump a Block of memory data on the Screen.
5. Move a block of Data/Program from one location to another location.
6. Fill a constant Byte in the Memory Block.
7. Execute the program in full clock speed mode.
8. Execute a program in single instruction mode.

9. Execute a program with Break Point.

10. Compare a Block of memory with another Block.

11. Compare a Block of memory with a Data Byte.

12. Assemble / Disassemble a program on the Screen

13. To Search for a String of Byte in a Memory Block.

1.3.1 INTRODUCTION TO HARDWARE

a) MICRO – CONTROLLER

The system has got 8031/8051 Microcontroller as the CPU Operating at a Crystal frequency of 20 Mhz. The 8031/8051 Microcontroller is an 8 bit CPU with on chip oscillator and clock circuit, 8 input / output lines, two 16 bit Timer / Counters, a fire source interrupt structure, full duplex serial port and built in Boolean processor. The chip has the capacity to address 64 K bytes of external data memory and 64K bytes of external program memory.

b) CLOCK GENERATION

The clock generator circuit is in-built in the micro-controller. The circuit accepts a crystal input, which operates at a fundamental frequency of 20.000 MHz (20.0 MHz was selected since this frequency is a multiple of the baud rate clock and also provides a suitable frequency for the CPU). Additionally, the clock generator performs a further divide-by-two output called PCLK (peripheral clock) which is the primary clock signal used by the remainder of the circuits.

c) MEMORY

The Kit provides 64K Bytes of EPROM loaded with monitor and 8K bytes of CMOS RAM. The RAM area can be further expanded to incorporate 32K Bytes of memory. Out of this total on board memory the RAM area from 4000 to 7FFF is backed up by the battery. The memory mapping available on the board is given in chapter-5.

d) I/O DEVICES

1) 8255 (Programmable Peripheral Interface)

The 8255 is a programmable peripheral interface (PPI) designed to use with 8031/51 Micro-controller. This basically acts as a general purpose I/O component to interface peripheral equipment to the system bus. It is not necessary to have an external logic interface with peripheral devices since the system software programs the functional configuration of 8255. It has got three input/out ports of 8 lines each (PORT-A, PORT-B AND PORT-C). Port-C can be divided into two ports of 4 lines each named as Port-C upper and Port –C lower. Any Input/ Output combination of Port-A, Port-B, Port-C upper and Port-C lower can be defined using the appropriate software commands. The port addresses for these ports are given in chapter-5. The Kit provides six input/output ports of 8 lines each using two 8255 chips. These ports are brought out at 26 pin FRC connectors J1 & J2.

The 8255 (PPI) can be programmed and tested using program given in sample programs.

2) 8253 (Programmable Interval Timer)

This chip is a programmable interval timer / counter and can be used for the generation of accurate time delays under software control. Various other functions that can be implemented with this chip are programmable rate generator, Event Counter, Binary rate multiplier, real time clock etc. This chip has got three independents 16bit counters each having a count rate of up to 2MHz. The CLK, Gate & OUT signals of these timers are brought out at the J5 connector.

The 8253 (Timer / Counter) can be programmed and tested using program given in sample programs.

3) 8251 (USART)

This chip is a programmable communication interface and is used as a peripheral device. This device accepts data characters from the CPU in parallel form and then converts them into a continuous serial data stream for transmission. Simultaneously it can receive serial data stream and converts them into parallel data characters for the CPU. This chip will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of it at any time 8251 has been utilized in the Kit for RS-232-C interface and 20mA current loop. The Signals of the Serial Interface provided through the 8251 are brought out on the connector J6.

Interfacing CRT terminal / IBM compatible PC/AT with the kit using the 8251 chip is given in chapter-3.

4) 58167 (Real Time Clock Chip) Optional

The Kit provides an on Board RTC Chip which can be used to run the Real Time Clock on the Board. The students can understand the functioning of the Real Time Clock. The other details of the RTC are given in Annexure.

The 58167 (RTC) can be programmed and tested using program given in chapter-5 (sample programs).

5) DISPLAY

The Kit provides 16*2 Character LCD (Liquid Crystal Display). The user can therefore directly work on the alphanumeric mode. Each line of the display has an address, which can be used by the user to display the data on the display. Some of the sample programs given later in the chapter-5 use the LCD Display. The user should refer these for reference. The Kit also has provision for adding 20*2/20*4 LCD display instead of standard 16*2 Display.

6) INTERFACE

The Kit has provision for the flowing interfaces. The detail descriptions about these are given in chapter-3.

RS-232-C interface through 8251 (USART) (On Demand)

USB Serial Interface (Standard)

Real Time Clock chip Interface (Optional)

EPROM Programmer interface for 2764/27128/27256. (Optional)

Cassette Interface (Optional)

Speaker Interface (Optional)

7) BATTERY BACK UP

The Kit provides a battery back up for the onboard RAM area. The battery backed up RAM has an address from 4000 to 7FFF. The program / data Stored in this area will remain stored even if the power is switched off.

CHAPTER – 2

COMMAND DESCRIPTION

2.1 KEYBOARD DESCRIPTION

The Kit provides 101/104 Keys standard IBM type Keyboard as an Input Device. As a standard, the IBM compatible keyboard is provided unless until specifically mentioned in the order. A socket is provided on the kit to connect the IBM compatible keyboard. The keyboard gets its power from the Kit.

The various commands available on the system are assigned an alphabet each. Where ever possible, various commands are actually grouped together and are represented by an alphabet. The system accepts the alphabet in capital or in small letters.

The system monitor is very interactive and displays a message for each action. When user press <CR> or <SPACE> key system will display "Cmd_Wrd=" message or "?" respectively.

In certain Command Words, there are Sub commands available. These sub commands wherever available, can be selected by pressing <Space Bar> key. Pressing of <Space> key each time will change the sub command. The Sub command is selected when you press <CR> key. If you keep pressing, the <Space> key, the display of the name of Sub command is repeated in a loop.

Pressing of **ESC** key at any time aborts the Command and brings the system back to the command mode and displays a message " Cmd_Wrd=" indicating that a new command can be given now.

2.2 LIST OF COMMANDS:

The various commands with the list of Sub Commands and their command Keys are listed here for the Key Board Mode.

S. No.	Key Used	Command Name	Command Description with list of Sub Command Options	Sub commands Available
1	S	Sub_M/R?	Substitute - **Extr_Mem** - **Register** - **Intr_Mem**	-Read / Write into/ from external Memory -Read / Write into / from register - Read / Write into/ from internal

				memory
2	M	Move	Move - **Mem?** - **Data**	- Move a Block of memory to another Destination address
3	C	Compare	Compare - **Mem** - **Data**	- Compare two Blocks of Memory for being equal. - Compare the data of the Master Socket with the EPROM placed in the ZIF (this Command is only for optional EPROM programmer)
4	Z	Mem_Tst	Memory Test	- Test a Block of RAM memory for being OK.
5	E	Ex_Mon	Expand Monitor	- Expands the Monitor - Command Set for working in assembler mode
6	G	Go To	Execute in **- Burst** **- Sing_Stp** **- Break _Pt**	- Execute a program in Full Speed Mode - Execute a program in Single Step mode - Execute a program with Break Point.
7	S	Ser_Out	Serial Out	- Up Loads a program/data to the PC/AT in Hex or Binary Format.
8	I	Ser_In	Serial Input	- Down Loads a Program/Data into the RAM area from the PC/AT
9	B	Blank	Blank Check	- Blank Check an EPROM in the ZIF Socket for being Blank (this Command is only for optional EPROM programmer)

10		List	List	- List the contents of the EPROM in the Master or ZIF Socket of the programmer into the desired RAM area. (this command is only meant for the optional EPROM programmer)
11	P	Program	Program	- Program the EPROM in the ZIF Socket with the data from the RAM or the master EPROM or the constant Byte. (this command id only valid for the optional EPROM programmer).

2.3 COMMAND DESCRIPTION

2.3.1 Sub_M/R: Substitute Memory, Register Command.

This Command can be used for the following purposes.

1) Enter a program / Data in to the memory Area
2) Examine / Modify a Register of the 8031/51 Microcontroller.

2.3.1.1 ENTER A PROGRAM / DATA IN THE MEMORY AREA.

Procedure:-

Press the **"S"** key on the keyboard. A message "Sub_M/R? " is displayed on the LCD. Press <CR> key. The system displays a message "Extr_Mem." Press <CR> key. A message " Addr " is displayed. Now enter the starting address of the memory area followed by <CR> key. The system displays the address of the location along with the present contents of the location. If you want to change the content of the location, enter the required data followed by the <CR>, the system will display the content of the next location and so on. In case at any point you do not want to change the content of a memory location, then just press <CR> key. Once the required data has been entered,

press " Esc " key to come out of this command, Now system will display "Cmd_wrd=" on LCD display.

EXAMPLE : Enter the following program in location 6000h onward using the Sub_M/R Command.

Mem. Location	OP-CODE	Mnemonics
6000	E5 20	MOVA 20
6002	**D2 20**	**SETB 20**

KEY PRESSED	**DISPLAY ON LCD**
S	Sub_M/R?
<CR>	Extr_Mem
<CR>	Addr
6000<CR>	6000 XX
E5<CR>	6000 XX E5
20<CR>	6001 YY 20
	6002 ZZ
D2 <CR>	6002 ZZ D2
	6003 XX
20 <CR>	6003 XX 20
	6004 YY
<ESC>	Cmd_Wrd=

NOTE:

Here XX, YY, ZZ indicates the random values stored in the memory locations. If you want to come out of Sub-M/R? Command press ESC key of the keyboard.

2.3.1.2 EXAMINE / MODIFY A REGISTER OF 8031 / 8051

Procedure:-

Press the **"S"** key on the Keyboard. A message "Sub_MIR?" is displayed on the LCD. Press <CR> key. The system displays a message "Extr_Mem". Press <Space> key, the system displays a message "Register" is displayed. Press <CR> key, the system displays a message "General ". If you want to examine any of the registers like A, B, DPL, DPH etc. press <CR> key or else press <space> key for examining thew other Bank of Registers like R0, R1 etc.

The system shows following message when <CR> is pressed.

1. System will display a message "Name ". Enter the register you want to examine and press <CR> e.g. A, B, D, etc.
2. System will show content of register and again ask for the name of the next register that you want to examine.
3. If you do not to examine any other register, just press <Esc> key. The system will come out of the Examine Register Command and display a message "Cmd_Wrd="

If in the above example, <space> is pressed instead of <CR> then the system will display a message "Bank"

1. If <CR> is pressed now, system displays a message "Name " i.e. it asks the name of register like R0, R1, R2, etc.
2. If <CR> is pressed again system will show content of register R0 onward.
3. If <space> is pressed the also the system will ask name of register whose content, user want to see or change.

EXAMPLE: Enter the value 34 in register A and value 78 in register B and then verify the same with out resetting the system .

Key Pressed	**Display on LCD Screen**
S	Sub_MIR?
<CR>	Extr_Mem
<Space>	Register
<CR>	General
<CR>	Name

<CR>	A XX
34 <CR>	Name
<CR>	B YY
78<CR>	Name
Esc	Cmd_Wrd=

NOTE:

1. Here the XX, YY indicates the random valued stored in the Registers.

2. DO Not Press RESET at this stage.

Now again press **S** key and repeat the above process to verify that the Registers A and B have the same value, which you entered.

2.3.2 Move Command : Move Block , Constant Command

This Command can be used for the following purposes

1) Move a Block of memory from one location to another location
2) Move a constant Data Byte in to a Block of memory

2.3.2.1 MOVE A BLOCK OF MEMORY FROM ONE LOCATION TO ANOTHER LOCATION

Procedure:

Press **M** key and the system will display a message "Move?". Press <CR< key and the system will display a message "Mem?". Press <CR> key and the system will display a message "Strt". Enter the segment address of the Block of memory to be moved followed by <CR>. The system will display a message " End". Now enter the End address followed by <CR>. A message "Dstn" is displayed. Now enter the address of the destination Block i.e. where the block of data is to be moved followed by a <CR>. The system displays a message "Done". The system is ready to take the new Command now.

EXAMPLE: Move the data lying at Block address 6000H – 6005H to 7000H.

User can enter data using Sub_M/R command at address 6000H as explained in command 2.3.1.1, please do that now using sub_M/R Command as explained earlier.

Key pressed	Display on LCD Screen
M	Move?
<CR>	Mem?
<CR>	Strt
6000<CR>	End
6005<CR>	Dstn
7000<CR>	Done

The system is ready to take another command. Verify that the data at address 7000 to 7005 is also lying at location 6000 using Sub_MIR Command as explained earlier.

2.3.2.2 MOVE A CONSTANT DATA BYTE IN A BLOCK OF MEMORY

Procedure:

Press **M** key and the system will display a message "Move?". Press <CR> key and the system will display a message "Mem?". Press <space> key and a message "Data" is displayed. Press <CR> key and the system prompt a message "Byte". Enter the Byte with which you want to fill the memory Block followed by <CR>. A message "Strt" is displayed. Enter the address of the memory location where you want to fill followed by <CR>. The system will display a message "End". Now enter the End address followed by <CR>. The system displays a message "Done". The system is ready to take the new Command now.

EXAMPLE: Move a Constant Data Byte AA in memory Block 6000H to 6005

Key Pressed	Display on LCD Screen
M	Move?
<CR>	Mem?
<Space>	Data
<CR>	Byte
AA<CR>	Strt

6000 <CR> End

6005 <CR> Done

The system is ready to take another command now. You can verify the above by Examining the locations 6000H onward by using the Sub_M/R Command.

2.3.3 Compare: Compare Block, Constant, Master Command

This Command can be used for the following purposes.

1) Compare two Blocks of memories for being equal.
2) Compare a Block of Memory with a constant Data Byte.
3) Compare a Master EPROM with the EPROM in the ZIF Socket. This command is useful for Optional EPROM Programmer.

2.3.3.1 COMPARE TWO BLOCKS OF MEMORY

This command will compare two Blocks of Data for being equal. If the data are same, it displays a message done where as if there is a discrepancies between thr two Blocks, then the address where they are not same is displayed, with the content of the second Block at that location. On pressing <CR> again, the next location where they are not same will be displayed along with the data Byte. On pressing <CR> each time all such locations can be displayed where as at any time on pressing <CR>, if there is no other location of discrepancy, then the system displays a message "Done".

Procedure:

Press **"C"** key, the system displays a message "Compare?". Press <CR> key and a message is prompted "Mem?". Press <CR> key. The system will display a message "Strt". Enter the starting address of the first block followed by <CR>. The system will display a message "End". Now enter the End address of the first block followed by <CR>. The system asks for "Dstn". Enter the address of the next Block followed by a <CR>. If the Blocks are equal, a message "Done" is displayed or else the address of the first location where they are not same will be displayed along with the data. At any time you want to quit the command, press Esc key. A message "Cmd_wrd=" is displayed and the system is ready to take new command.

EXAMPLE 1: In the Move Block Command we have moved the data Block lying at location 6000H – 6005 to Block address 7000H. Now use the Compare Command to compare the two Blocks for being same.

Key Pressed	**Display on LCD Screen**
C	Compare?
<CR>	Mem?
<CR>	Strt
6000<CR>	End
6005 <CR>	Dstn
7000 <CR>	Wait
	Done

EXAMPLE 2: Now change the contents of memory location 6000 to 55 using Sub_M/R Command and then repeat the Compare Command as explained above. You will see that the system displays the discrepancy at address 6000 along with the Data.

2.3.3.2. COMPARE A BLOCK OF MEMORY WITH A CONSTANT DATA BYTE.

This command will compare a memory Block with a constant data value. If the entire Block of memory has the same value, the system will prompt a message "Done" where as if there is a difference at some memory location, then the system will display the first location where there is a difference along with its Data Value. On pressing <CR> again, the system will show "Done" message if rest of the locations has the same Data or else the system will display the next location where there is a discrepancy.

Procedure:

Press **"C"** key, the system displays a message "Compare?". Press <CR> key and a message is prompted "Mem?. Press <Space> key. The system will display a message "Data". Press <CR> key and the system will ask for the "Byte". Enter the Byte with which you want to compare and press <CR>. A message "Strt" is displayed. Enter the start address of the Memory followed by a <CR>. The system will display a message "End". Now enter the End address of the Memory followed by <CR>. If the entire Block has the same Data Byte, a message "Done" is displayed or else the address of the first location where they are not same will be displayed along with the data. Press <CR> again and the system will display either "Done" or the next location where the data is not same along with its data. At any time you want to quit the command, press "Esc" key. A message "Cmd_wrd=" is displayed and the system is ready to take new command.

EXAMPLE 1: In the Explanation of the Move Constant Command, we had moved a constant data AA in locations 6000H to 6005H. Compare this memory Block with the Constant Byte AA. It should show "Done" as the result.

Key Pressed	***Display on LCD Screen***
C	Compare?
<CR>	Mem?
<space>	Data
<CR>	Byte
AA<CR>	Strt
6000<CR>	End
6005<CR>	Wait
	Done

EXAMPLE 2: Now change the content of location 6000 to 23 using the Sub_M/R command and again use the compare constant command to see the result.

Key Pressed	**Display on LCD screen**
S	Sub-M/R
<CR>	Mem?
<CR>	Addr
6000<CR>	6000 AA
23<CR>	6000 AA 23
<CR>	6001 AA
Esc	Cmd_Wrd=
C	Compare?

<CR>	Mem?
<Space>	Data
<CR>	Byte
AA<CR>	Strt
6000<CR>	End
6005<CR>	Wait 6000 23
<CR>	Wait
	Done

2.3.4 Ex_Mon : Expanded Monitor Command

Note: Please refers to Appendix B for the proper instruction format.
e.g. For MOV b A,#20; the syntax for the in-built assembler is
MOVA, #20

This command expands the scope of the monitor program to include the assembly of the 8031/8051programs.Although certain other command are also added but these are valid for the serial monitor and will be explained in chapter for the serial command.

Procedure:

Press **"E"** keyk and the system will display a message "Ex_Mon?". Press <CR> key and a "C=" will be displayed. One can now Assemble the Program by pressing "A" or "A" and starting address of the memory location. If you want to come out of the Expanded Mode, just press "Q" followed by a <CR>, the system will prompt "Cmd_Wrd=".

EXAMPLE: Enter the following Sample program using on Board Line Assembler
In Expanded Mode.

6000	74 20	MOVA, #20
6002	79 03	MOV b R1, #03
6004	29	ADD b A, R1
6005	02 00 06	LJMP b 0006

The Program adds two numbers lying in A and R1 register and stores the result in Register A. Verify the contents of the memory location 6000 onward with the codes given above using Sub_M/R Command

Key Pressed	**Display on LCD Screen**
E	Ex-Mon?
<CR>	C=
A6000	**6000_**
	6000 MOVA, #20<CR> 6002 MOV b R1, #03 <CR> 6004 ADD b A, R1 <CR> 6005 LJMP b 0006 <CR>
Q	PRTN OFF
	Cmd_Wrd=
S	Sub_MIR?
<CR>	Extr_Mem
<CR>	Strt
6000<CR>	6000 74
<CR>	6001 20
<CR>	6002 79
<CR>	6003 03
<CR>	6004 29

(Keep Pressing <CR> till you have verified all the memory locations). Press Esc key to come out of this command.

Note: One can also dissemble the program by pressing 'U' and giving the starting address of memory location where program is entered. Then press F9 function key followed by <CR>. After that use either F9 or F10 to see further.

2.3.5 Go To : Go To Command

This command can be used for the following purpose.
1) Go To – Burst: Execute a program in full speed mode.

2) Go To - Step: Execute a program in single step mode.
3) Go To – Break: Execute a program with a set Break point

2.3.5.1: Go To – Burst

The following sequence is to be followed for the above command.

Procedure:

Press **"G"** Key and the system will display a message "Go To?". Press <CR> key and you are prompted with a message "Burst". Press <CR> and the system display a message "Addr". Enter the address followed by a <CR>.

The program will be executed and the system ;will display you a message according to your program.

EXAMPLE: Execute the program entered above at location 6000H and verify that the result in Register A.

Key Pressed	**Display on LCD Screen**
G	Go To?
<CR>	Burst
<CR>	Addr
6000 <CR>	Wait Done

The program has been executed now. Do not press RESET key and verify the contents of the Register A using Sub_M/R Command as explained earlier. The register A will have the result. Verify the result in register A using the Sub_M/R Command as explained earlier.

2.3.5.2 GoTo - Step

The following sequence is to be followed for the above command

Procedure:

Press **"G"** key and the system will display a message "Go To?. Press <CR> key and you are prompted with a message "Burst". Press <Space> key and the system will display a message " Sing_Stp". Press <CR> key and the system will ask for the address of the memory location. Enter the address followed by a <CR>

The program will execute one instruction and stop at the address of the 2nd instruction with the OP-Code of the instruction displayed. Press <step> key on the board and the system will execute the next instruction and stop again at the next instruction and so on. Please not that TIMER is used for Single step and the instructions of this routine will be executed step by step.

EXAMPLE: Execute the same program as given above in Single Step mode.

Key Pressed	**Display on LCD Screen**
G	Go To?
<CR>	Burst
<Space>	Sing_Step
<CR>	Strt
6000<CR>	6000 74
<STEP>	6002 79
<STEP>	6004 29
<STEP>	6005 02

At this stage, the program will jump to the subroutine address called here. So do not be confused here. There is no point in continuing the execution in single step as you will have to execute all the instructions of the subroutine. As the concept of single instruction has been understood. You can come out of this process by pressing Esc key.

<Esc> Cmd_Wrd=

the program has now been executed and one can check the contents of register A using the Sub_M/R Command.

2.3.5.3: - Break Pt

The command allows us to execute our programs in full speed mode but with a break point in between where the program will stop. It is very use full command in debugging our programs as every time the program stops at a break point, we can examine the contents of various registers and memory location and find our as to how the program is behaving. In the serial mode, the program also displays the contents of all the registers. In this case we must ensure that the break address given by us should be the starting of the instruction and not the in between address i.e. suppose an instruction starts at address 6000in the above sample program, then we should not give the break point at 6005. We should either give break point address as 6004 or as 6005,

which will be the beginning of the new instruction as the instruction at 6005 was of two bytes. If by mistake you give the break address in between the addresses, then the system will write a code CC at that address and execute the program without stopping at the desired address. In this case the user should change he content of the address from CC to the original code.

Procedure:

Press **"G"** Key and the system will display a message "Go To?". Press <CR> keyk and you are prompted with a message "Burst". Press <Space>key and the system will display a message "Sing_Stp". Press <Space> key and the system will display a message "Break_Pt". Press <CR> key and the system will ask for the address of the memory location by displaying a message "Addr". Enter the starting address from where yu want to execute the program followed by a <CR>. The system will now display a message "End". Enter the Break-point Address at this juncture followed by <CR>. The program will execute instructions till the break point and stop at the address of the Break-point but the display will have a message "wait".

The system is now ready to take any new command. Without pressing Reset one can examine any memory area or any register.

EXAMPLE: In the above example explained in the Go To command, we shall execute the program in Break mode with the Break Address i.e. the End address as 6004.

Key Pressed	**Display on LCD Screen**
G	Go To?
<CR>	Burst
<Space>	Sing_Stp
<Space>	Break_Pt
<CR>	Addr
6000<CR>	End
6004<CR>	wait
	Done

One can now verify the contents of register and see that the A and R1 registers have a value of 20 and 03 respectively but the A register has not been updated with the result (i.e. addition of A with R1) as this instruction has not been executed.

2.3.6 : Ser_Out : Serial Out Command

This command sends the Data of the Memory Blocks on to the serial Port. So if a serial device is connected to this port, The Data will go there. For instance if a PC/AT System is connected to the serial port of the Kit through an Terminal Emulation software, then the data send out from the Kit will be displayed on the screen of the PC. The Kit can send the Data either in the Intel Hex Format or in the Binary Format. This command can also be useful for uploading the program / data to the PC/AT system hard Disk or Floppy Disk. This is possible through the save data Features of XTALK Software available in the market.

This command is explained in the chapter –3 on the Interfaces under the heading UPLOADING to the PC/AT System.

Procedure:-

Press **"U"** key and the system will display a massage "Ser_out". Press <CR> key and the system displays "PRTY" to set the parity. Press <CR> or else Press <Space>key, the system will display a message "Hex" or "Bin" respectively. Press <CR> key. A message "Strt" is displayed asking for the Starting address of the memory. Enter the address followed by a <CR> and you will get the message "End" asking for the End Address till where you want to send to the system PC/AT. Enter the required address followed by a <CR>. The system will send the data and show blinking cursor.

2.3.7: Ser_in : Serial in Command

This Command receives the data from the serial ort of the Kit and stores it in the Memory of the Kit. The Kit can receive the Data either in the Intel Hex format or in the Binary Format. This command is very useful for Downloading the program lying in the Hard Disk or Floppy Disk of the PC/AT System. The Command needs a XTALK software which can send a file in the Intel Hex or in the Binary format to the PC/AT Serial Port.

This command is explained in the chapter – 3 on the Interfaces under the heading DOWNLOADING from the PC/AT system.

Procedure:-

Press **"I"** key and the system will display a message **"Ser_in**". Press <CR> key and the system displays "PRTY" to set the parity. Press <CR> or else press <Space> key,

the system will display a message “Hex” or “Bin” respectively. Press <CR> key. A message “Strt” is displayed asking for the starting address of the memory. Enter the address followed by a <CR> and you will get the message “End” asking for the End Address till where you want to receive to the system PC/AT. Enter the required address followed by a <CR> . The system will receive the data and show message “Wait” and Done.

2.3.8: Blank? : Blank Check for the EPROM in the Master socket or the ZIF Socket (Optional command Valid only for optional Eprom Programmer)

This Command checks the EPROM in the master socket or the ZIF Socket for being Blank or not. If the EPROM is blank, it shows done or else it will show the address where the EPROM is not blank with the data at that address.

Procedure:-

Press **“B”** key and the system will display a message “Blank?”. Press <CR> key and the system displays “MAST”. If you want this socket to be checked, press <CR> or else press <Space> key, the system will display a message ”PRST”. Press <CR> key. A message “Strt” is displayed asking for the starting address of the EPROM. Enter the address followed by a <CR> and you will get the message “End” asking for the End address till where you want to check for the Blank. Enter the required address followed by a <CR>. The system will perform the Blank Check and display the result on the screen.

2.3.9: Program? : Program an EPROM Command (Optional)

This command is used to program an EPROM using the optional EPROM Programmer interface Module. The command allows us to program the Eprom from the contents of either the RAM A read or with a constant Byte or from the EPROM Placed in the Master socket of the Programmer Module. The EPROM number to be programmed can also be entered. In case we choose to program from the RAM area, the system also allows us the option of choosing the Normal way i.e. Continuos Bytes.

Procedure:-

Press **“P”** key and we get a message “Program?” . Press <CR> key the system asks for the “Mem?” of the EPROM to be programmed. Enter the type as per the list given in the chapter – 3 on Interfaces. We can program from the block of memory or we can choose the Constant option by pressing <Space> or the third option i.e. the master option by pressing the <Space> key again.

If you choose the “Data” option then on pressing <CR> we will choose to program the EPROM with a constant Byte. The system will now display a message “Byte”. Enter the Byte with which you want to program the Eprom followed by a <CR>. The system will now display a message “Strt”. Enter the starting address of the EPROM followed by a <CR>. The system will display a message “End” asking for the End address. Enter the same followed by a <CR>. The programmer will now start programming and will show the result on the screen.

If you choose the Master option then on pressing <CR> when the LCD display had the message “MAST”, we will choose to program the Eprom with the contents of the Master EPROM in the Programmer Module. The system will display a message “Strt”. Enter the starting address of the Master EPROM followed by a <CR>. The system will display a message “End” asking for the End address of the Master EPROM. Enter the same followed by a <CR> The programmer will now display a message “Dstn”. Enter the starting address of the Eprom in the ZIF socket from where you want to start programming. Enter the same followed by a <CR> key and the programming will start the system will show the result on the screen.

CHAPTER-3

ON BAORD INTERFACES

The System provides the following on Board interfaces.

1. Serial Interface through USART 8251 chip.(On Demand)
2. USB Serial Interface (Standard)
3. Real Time Clock through 58167 RTC Chip. (Optional)
4. EPROM Programmer Interface (Optional)

4.1 Serial Interface (RS-232-C)

The system provides a serial Interface on the board of the kit. The user can connect a CRT terminal or PC/AT Computer System using any terminal emulation Software.

The On Board Serial Interface can be used for the following two purposes.

1. Uploading / Downloading the program from / to the kit to / from the PC/AT System. This is possible through the **"U" and "I"** Commands as explained below.

2. Executing the Commands of the Kit explained in chapter-2 through the Serial Interface. In this case the software monitor changes the Output Device of the kit to CRT terminal or Monitor of the PC/AT instead of the LCD Display. This is done by the **"R"** Command.

The Kit provides a 9-pin D Type male Connector where all the signals of RS232C are coming. The detail about this connector is given in chapter-6 of the manual. To connect a CRT terminal or the PC/AT system, one needs a cross cable as follows.

9 Pin D Type male of Kit	**To**	**9 Pin D type male of PC**
2		3
3		2
4		6
5		5
6		4
7		8
8		7

9 Pin D Type male of Kit	**To**	**25 Pin D type male of PC**
2		2
3		3
4		6

5	1 & 7
6	20
7	5
8	4

4.1.1 Connecting the kit to the CRT Terminal

The Kit 8051LCD can be connected to the CRT Terminal or the PC using Serial RS-232-C Interface or the USB Interface.

For connecting a CRT terminal to the kit, using RS-232-C, connect the two through a cross cable as given above, 9 pin Male to 9 pin Female or 9 pin to 25 pin. Choose the appropriate cable depending upon the type of Serial connector is there on the Terminal. If you want to connect the Kit to a CRT Terminal or PC using a USB Interface , the you will need a USB Male (A Type) to USB Male (A type). In both the cases one needs to use a Terminal Emulation Software like XTAL or PC Term or real Term

Set the Communication parameters on the software as follows:

Baud rate : 2400
Parity : None
Stop Bits : one
Data Bits : 8

Switch on the two devices and press RESET on the kit. A message ET-31 LCD appears on the kit. Now press "R" key. The system flashes a message "Serial? " on the screen. Press <CR> key and you will see a message ET-51 LCD on the screen of the monitor. The system is now ready to communicate with the CRT terminal. All the commands explained in the chapter on serial commands can now executed.

4.1.2 Connecting the Kit to the PC/AT Computer:

Using terminal Emulator Software available from the market, an IBM compatible PC/AT computer can be connected to the kit. The terminal emulator software makes a computer work like a Terminal.

XTALK is one such simple terminal emulator software for IBM-PC/AT compatible computers. It allows the user to communicate with the computer through serial port with the facility of downloading & uploading of data-between the computer and the other serial devices. The various communication parameters like baud rate (speed), number of data bits, stop bits, parity etc. can be changed. The package communicates through COM1: as well as COM2: ports of the IBM compatible PC/AT system.

4.1.2.1) INSTALLATION OF XTALK:

Although the manual, which is supplied by the XTALK vendors, does mention about the procedure of installation, it has been given here for the convenience of the users. One can as such use any communication package available from the market.

a) Insert the CD containing XTALK in you CD drive.
b) If you have hard disk, makes a directory in the name of XTALK in your C: drive.
c) Copy the diskette in your directory as given below:
   D>COPY *.* C:\XTALK <CR>
d) Execute XTALK as given below:
   XTALK <CR> (either from A: or C:)

4.1.2.2 SELECTION OF COMMUNICATION PARAMETERS

For setting the parameters, press <HOME> key. The system will show you a 'COMMAND? Prompt which means that the system is asking you to give commands. Write 'SP-2400' to set the baud rate at 2400.

COMMAND? SP 2400

The baud rates available in XTALK are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 etc. If you want to set the baud rate at say 2400, then in "COMMAND?" Prompt, write SP-2400 and press <CR>, Rest of the parameters viz. Parity, Duplex Mode, Data Parity, Full duplex Mode, 8 Data Bits and one Stop Bit. If you want to change any of the parameters write the first two letters e.g., PA for Parity, 'DU' for Duplex Mode and then write as per requirement. Suppose you want to change Parity, you should write in the COMMAND? PA EVEN, for odd Parity write PA ODD. Similarly one can change Duplex mode, port etc.

3.1.12.3 COMMUNICATION BETWEEN PC/AT & THE KIT USING XTALK

a) Connect the kit and Power Supply. Make sure that the +5V, GND are connected properly.
   Connect the serial port of PC with the Serial Port of the Kit (Connector J8) using a cross cable as mentioned above.
b) Switch on the computer and the Kit.
c) After loading XTALK , set the parameters as shown in the table below:

PARAMETERS TO SET		COMMANDS TO GIVE IN XTALK MODE	
SPEED	**2400 BAUD**	**SP**	**2400**

DUPLEX MODE	**FULL**	**DU**	**FULL**
PARITY	**NONE**	**PA**	**NONE**
STOP BITS	**1**	**ST**	**1**
DATA BITS	**8**	**DA**	**8**
CWAIT	**DELAY 20**	**CW**	**20**
LWAIT	**DELAY 20**	**LW**	**20**
EMLATION	**NONE**	**EM**	**NONE**

Write **AT b PGDN** in Command prompt and Press <CR>

Write **GO LO** in Command prompt and press <CR>.

To save the above settings to a file, give save command. SAVE 31KIT and press <CR>. After this command when one uses XTALK again the parameters can be loaded either using LOAD 'LO' command or as soon as XTALK is executed, it displays all the transmission files which has extension *.XTK. One can enter the number of the file directly. Now the kit is ready for serial communication.

d) Press **"R"** key of the kit and **ET-31LCD** will be displayed on the screen indicating that communication is established between kit and IBM PC.
e) Your PC & Kit are now linked for communication and all your instructions of the serial command mode explained in the chapter-4 can now be operated from your PC Keyboard.

   All serial commands mentioned in this chapter will be fed through PC Keyboard . Apart from this Uploading & Downloading from/to the kit can also be done through this software.

4.1.2.3 UPLOADING TO PC/AT COMPUTER USING XTALK:

Connect a **serial cable** between **connector J8** of the Kit and **COM1** port of the Computer (if not already connected as described above). The kit can now be used to store the data from the kit to the computer. This can be achieved by following the instructions given below:

Suppose the data lying between 6000H to 600F is to be stored in the computer.

a) Press **"U"** key on the Kit keyboard. A message **"Ser_Out"** is displayed on the screen of the Kit.
b) Press Enter and the system will display **"PRTY"** on the screen
c) Press Enter and the system will display **"Hex"** on the screen. Since we want to use Intel Hex Format, just press **<CR>** key again . The system ask for **"strt".**
d) Enter the Start address as **6000** followed by **<CR>**
e) You will be prompted by a message **"End"**
f) Just type the end address as **600F (DO NOT PRESS ENTER NOW)**
g) Press **HOME key on PC/AT keyboard**. A message **"Command?"** appears on the bottom most line.
h) Write **CA ON** and press **<CR>** key two times. Now Press **<CR>** on the **keyboard of the Kit.**
i) The data from address 6000 to 600F will be displayed on the screen of the **computer**
j) Press HOME key again and a message "Command?" appears again on the lower most line.
k) Write **CA OFF** and press **<CR>.**The following message will appear on the bottom line.
"Information is still in the capture.
Do you want to save it (Y/S)".
Since we want to save the data, press **"Y"**. After pressing **"Y"** it will ask you:

" Write capture buffer to what file? _____"

Name the file in which you want to save the data following. DAT e.g., **"ABC.DAT"** and press <CR>.
l) After pressing <CR> on the screen, you will get:
" File successfully written, press <Enter>".
m) Data captured will be stored in the file as defined by the file name.

4.1.2.4 DOWNLOADING FROM PC/AT TO 8031/51 KIT USING XTALK

The file from the PC/AT memory can be downloaded to the kit also. This command is very useful in Development of the software. One can store the program in the system hard Disk at the end of the day and start again from the same point the next day. After you have connected the kit with the PC/AT as explained earlier, one can down load into the kit using the **"I"** Command i.e. the **"Ser_In"** command.

The following procedure is to be adopted for downloading the above file from PC/AT to the kit.

Suppose we want to download the file named **"ABC.DAT"** stored earlier in the uploading example.

a) Press **"I"** key on the keyboard of the Kit and press <CR>.
b) The system will display a message **" PRTY"** on the screen. Press **<CR>** key again.
c) The system will display a message **"Hex"** on the screen. Press **<CR>** key again.
d) The system will now display a message **"Strt".** Enter the address of the memory where you want to store the data. In this example enter **6000** followed by a **<CR>.**
e) The system will ask for the end address of the memory by displaying a message **"End"** . Enter the address as 600F followed by a **<CR>.**
f) The system will display a message **"wait".**
g) Now press HOME key the PC/AT keyboard. A **"Command?"** will appear on the bottom line of the computer screen. Type **"SEND ABC..DAT"** and press <CR> KEY **TWICE**.
h) The programs stored in the file will be loaded into memory of your kit at the address specified in the program ie 6000 AND A MESSAGE **" Done"** appears on the screen of the Kit.

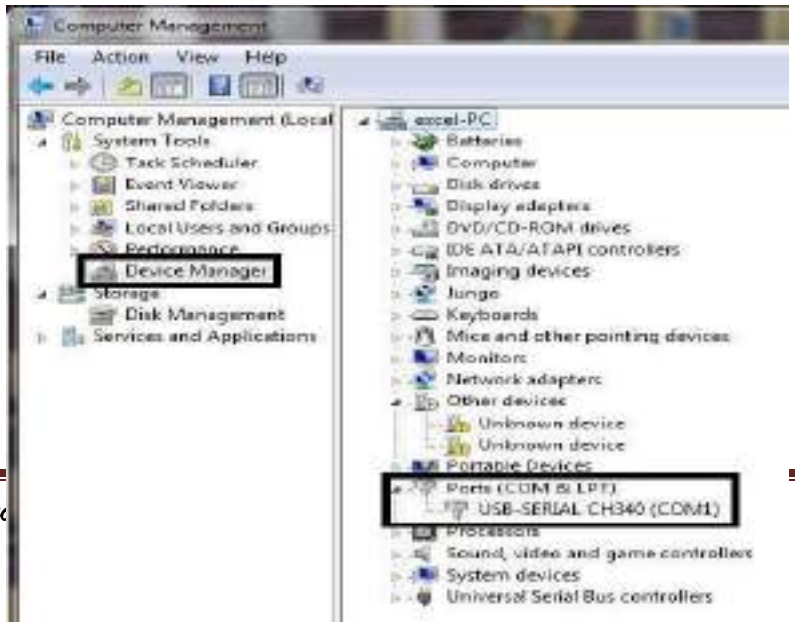
UPLOADING AND DOWNLOADING FILE THROUGH USB IN ET-8051LCD KIT USING REALTERM

1. Connect The 8051µC kit to a PC through USB cable
2. Connect a keyboard to 8051 kit
3. Install "Real Term" application on your PC and open it.
4. Click on "port" tab, set the Baud rate->2400

Set the Port ->Port no (e.g.= 1)

NOTE:-To check the connected COM PORT

- Right click on "Computer" icon which is in desktop
- Go to : Manage –>Device Manager
- Click on -> Ports
- Then it will show the connected COM PORT (e.g. USB-SERIAL CH340(COM1))

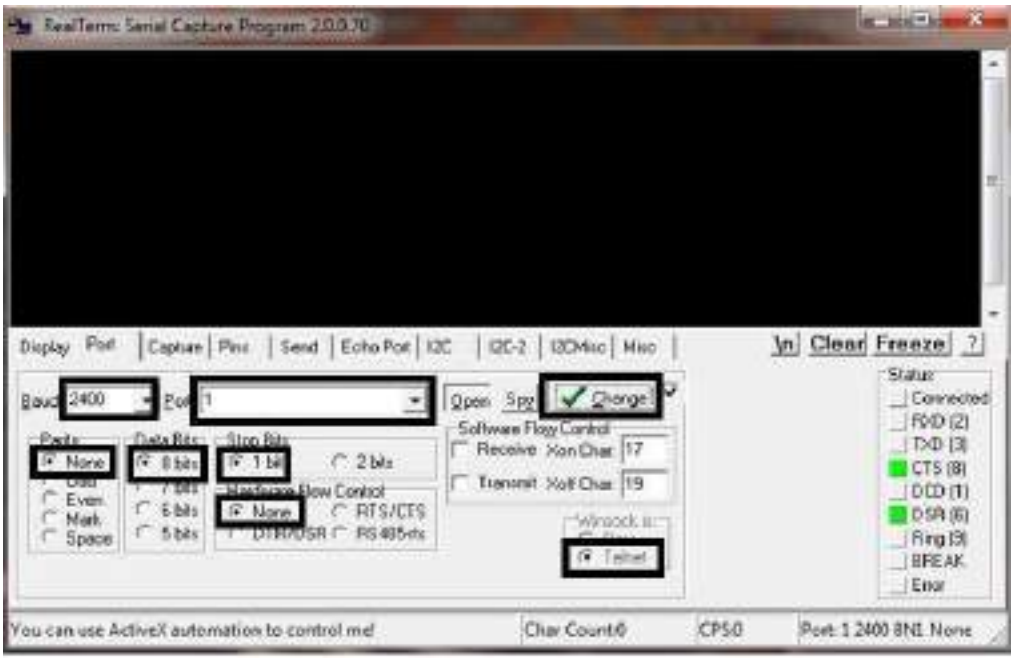


5. Then set Parity ->None
   Data Bits ->8bits
   Stop Bits ->1
   Hardware flow control -> None
   Winsock -> Telnet
6. Then click on "Change " to apply it. (Refer picture on the Next Page)

7. Reset the kit and press "R" key followed by <CR> from the keyboard which is connected to the kit.
8. Then a "**ET-51LCD**" Will display on the window (it means kit is connected with PC and ready to communicate)

UPLOADING

1. Press "**U**" key on the PC key board. A message "**Ser_Out**" is displayed on the screen of the PC.
2. Press enter and the system will display "**PRTY**" on the screen.
3. Press enter and the system will display "**Hex**" on the screen.
4. Press enter and the system ask for "**strt**".
5. Enter the start address as 6000 followed by <CR>
6. You will prompted by a message "**End**".
7. Just type the end address as 60FF (don't press < CR> at this moment).
8. Then switch to "capture" tab
9. Click on "…" browser button from the "FILE" menu
10. Select a location (e.g. Desktop) and give a name followed by an extension.DAT, then save it.
11. Check Bytes->000000
    Time stamp -> None
    Delimiter -> comma
12. Then click on "Start : overwrite"(after clicking the capture section becomes red)
13. Then press "enter" key (by clicking inside the black screen)
14. Then click "Stop capture" button from the capture section
15. Then go to the saved location (e.g. Desktop) and open the generated .DAT file (e.g. capture.dat) through Notepad and check whether the data is uploaded or not to PC



DOWNLOADING

1. Press "**I**" key on the PC key board.
2. Press enter and the system will display "**PRTY**" on the screen.
3. Press enter and the system will display "**Hex**" on the screen.
4. Press enter and the system ask for "**strt**".
5. Enter the start address as 6000 followed by <CR>
6. You will prompted by a message "**End**".
7. Just type the end address as 60FF and press enter.
8. The system will display "**wait**"
9. Now switch to "Send" tab
10. Select the file which you want to download to kit (e.g. capture.dat from desktop) by clicking on the"…" browser button.
11. Then click on "send file" to dump the file to port
12. It will show "done" after completing the sending process and two dots followed by *CRLF on the black screen
13. Then list the memory location and very that the data is downloaded in the Kit memory

4.1.2.5 GO BACK TO KEYBOARD MODE

If you want to go back to keyboard mode from the Serial Mode operation, press "O" key, the kit display a message "Cmd_Wrd=" and the serial operation will end. One can now operate from the keyboard of the kit.

4.1.2.6 ABORTING THE XTALK SOFTWARE

For aborting the XTALK, press Esc key write **"QUIT"** at COMMAND? Prompt.

4.2 REAL TIME CLOCK (Optional)

A Real Time Clock is provided on the Board of the kit. The user can write in to the clock chip the data of the HOUR, MINUTE and SECOND . The different addresses of the chip for various operations are given here. The Chip is connected to the Higher Byte (H_Byte) of the Data Bus. The RTC operates from a crystal of 32.768 KHz.

The Interrupt of the RTC is connected to the INTR (pin-18) of the 8031 CPU.

General Description of 58167 Chip

The MM58167B is a low threshold metal gate CMOS circuit that functions as a real time clock in bus oriented microprocessor systems. The device includes an addressable real time clock counter, 56 bits of RAM, and two interrupt outputs. A POWER DOWN# input allows the chip to be disabled from the rest of the system for standby low power operation. The time base is a 32.768 KHz crystal oscillator.

Features

- Microprocessor compatible (8-bit data bus)
- Milliseconds through month counters
- 56 bits of RAM with comparator to compare the real time counter to the RAM data.
- 2 INTERRUPT OUTPUTS with 8 possible interrupt signals
- POWER DOWN# input that disables all inputs and outputs except for one of the interrupts
- Status bit to indicate rollover during a read
- 32.768 KHz crystal oscillator
- Four-year calendar (no leap year)
- 24-hour clock

The address Bits as connected to the chip for reading or writing from / to the chip for various operations are given here for the user to understand.

A5	A4	A3	A2	A1	Function
0				0	Counter – Milliseconds
0	0	0	0	1	Counter – Hundredths and Tenths of Seconds
0	0	0	1	0	Counter – Seconds
0	0	0	1	1	Counter – Minutes
0	0	1	0	0	Counter - Hours
0	0	1	0	1	Counter – Day of week

0	0	1	1	0	Counter – Day of Month
0	0	1	1	1	Counter – Month
0	1	0	0	0	RAM – Milliseconds
0	1	0	0	1	RAM – Hundredths and Tenths of Seconds
0	1	0	1	0	RAM – Seconds
0	1	0	1	1	RAM – Minutes
0	1	1	0	0	RAM – Hours
0	1	1	0	1	RAM – Day of week
0	1	1	1	0	RAM – Day of Month
0	1	1	1	1	RAM – Month
1	0	0	0	0	Interrupt Status Register
1	0	0	0	1	Interrupt Control Register
1	0	0	1	0	Counters Reset
1	0	0	1	1	RAM Reset
1	0	1	0	0	Status Bit
1	0	1	0	1	Go Command
1	0	1	1	0	STANDBY INTERRUPT
1	1	1	1	1	Test Mode

EXAMPLE: Write the data of Hour, Minute and Sec on the RTC using the Sub_M/R Command in the out Port mode and then Read the values back using the same Command in the Input Port mode.

4.3 EPROM Programmer Interface (Optional):

The kit has an optional EPROM Programmer, which is available in the form of a module connected to the kit through a cable. The various commands available for this are explained in the chapter –2 of keyboard description.

The various chips which can be programmed on the programmer are 2716, 2732, 2732A, 2764, 27128, 27256, 27512. The device number to be programmed is to be entered when asked by the system "Type". Enter the last four digits of the IC e.g. Enter 2764 for 2764, enter 7128 for 27128, enter 732A for 2732A etc. The programming voltage is generated automatically by the programmer unit.

While the system is programming the EPROM, it also displays the address of the EPROM being programmed and its corresponding data. The system also verifies after every programming of the Byte. If there is a mismatch, It stops the operation.

CHAPTER – 4

SERIAL MODE COMMANDS

The system provides two modes of command i.e. the **Keyboard mode** and the **Serial mode**. The commands in the keyboard Mode are explained in the chapter-2. However this chapter covers the commands in the Serial mode. Most of the commands in the keyboard mode and the Serial mode are exactly the same syntax and procedures excepts one or two commands like dump memory, disassemble program and menu command etc.

However if the system is in Expanded mode, then most of the commands like, Move, Compare, Fill, Examine Register and Memory etc. also can be executed directly instead of going through the usual way of going through the usual way of going through the Main command. Say e.g. the Move Command is executed in the normal way through the Sub_M/R Command. However in the expanded mode, it can be executed directly by giving a command. The syntax for the various commands in is mode is explained later in this chapter.

4.1 HOW TO ENTER IN THE SERIAL MODE:

The user can use either a CRT terminal or a PC/AT System for this purpose.

Please refer to the instructions given in **Chapter – 3 at Serial No. 3.1.1 to 3.2.2.3** to connect the kit with CRT or Computer PC/AT. Once the kit has got connected to the PC/AT or CRT terminal using the "R" Command, you will get a message **"ET-51LCD".** This chapter will explain the commands from this onward. **The system in Serial mode accepts the commands in Capital Letters only.**

4.2 : MENU COMMAND: Z

Enter "Z" on the Serial Keyboard and the system displays a set of menu as follows.

List of commands

Press Enter Key to Select any Command or
Sub Command or to End Address/Data Fields
To Decrement Address Press Ctrl U
To Select Sub Commands within any

Command Press Space/Esc Key
Key Pressed Commands Executed

S (Sub_M/R) Modify/Examine Memory or Register

G (Go To) Run Prgm in Burst Or Step Or with Brk Point

M Move Mem_Blk Or Fill a Constant Data in Mem_Blk

E Expand the Monitor command Set

D / V Select 7 Segment / Video

Esc Abort the Command and Return to command Mode

**********Expanded Monitor Commands Set**********

Debug Commands Except File & Execution

A Line Assembler
U disassemble
Q Return to System Monitor

Refer Manual for Further Detailed

Description Of the commands in Key

Board or in Serial Mode

Cmd_Wrd =

The Commands **"S", "G", "M", "E"** Commands have the same meaning as explained in the Keyboard Mode. However the **"A", "U"** and the **"Q"** commands needs to be explained here in the new environment of serial output. Here for the clarity of the new user, the letters outputted by the system have been high lighted in BOLD letters.

ET- 31LCD ; Press **"E"** key. The letter **"E"** is not displayed here but the system asks
A massage
Ex_Mon? ; Press <CR> and you get a message for line assembler with a "C=" in The next line.

Line Assembler / Disassembler for 8031/51

Note: Please refers to Appendix B for the proper instruction format, e.g.

For MOV b A, #20 ; the syntax for the in-built assembler is MOVA, #20

-A <CR> ; Press "A" and <CR>, the system will prompt with a default address of 6000 for starting the assembly

6000 MOVA, B <CR>
6002 MOVB, R1 <CR>
6004 MOVR1, R2 <CR>
6006<CR> Entering a <CR> without giving any command will terminate the Assembly command and a **"C="** is displayed

A6000<CR> If you want to Assemble from specific address say from 6000 then Follow the **"A"** Command with the Address
6000 MOV b R1,#20 <CR>
6002 MOVA, #50 <CR>
6004 MOV b R1, R2 <CR>
6006 <CR>

U6000<CR> ; The command U6000 will disassemble the Program from this address Onward till 16 instructions are displayed. Since we have not assembler Beyond the address 6005, the unassembled portion represents what ever Is lying in the memory.

6000	79 20	MOV	R1,#20
6002	74 50	MOV	A,#50

4.3 The Syntax for Serial commands in the Expanded Mode:

The Following commands can also be executed in the expanded mode with syntax as explained below:

4.3.1 Fill Command:

The Fill command will fill a constant Byte in a memory Block. This command can be executed in serial mode using Move command as explained in Chapter-2. However the syntax for this command in expanded mode is same as explained earlier.

4.3.2 Move Command:

This command will move a block of Data or Program from one location to another location. The syntax for the Command is given here.

-M6000 60FF 7000 <CR>

This command will move the block of data from location 6000 to 60FF to location 7000.

4.3.3 Compare Memory

This command will compare two Blocks of data for being equal. If they are not equal, then it displays the addresses where they are not equal with the corresponding data. The syntax of the command is given below.
-C6000 60FF 7000 <CR>

This command will compare the block of memory 6000, 60FF to the Block 7000.

4.3.4 Get out of Expanded Mode:

The Command letter **"Q"** will bring you out of the Expanded Mode of commands.

-Q <CR> ; Q Command brings you out of Expanded Mode and a message

; "Cmd_Wrd =" appears on the screen.

Cmd_Wrd=

One can now use the normal serial mode commands of the kit.

4.3.5 Get Out of Serial Mode:

To come out of the serial mode, a **"O"** command is provided. So press **"O"** key and on the LCD Screen, a message "Cmd_Wrd=" is displayed. The system will now work in the keyboard mode.

CHAPTER – 5

I/O ADDRESSES, SOFTWARE AND MEMORY MAPPING

5.1) PORT ADDRESS:

The port addresses of the various I/O devices used in the kit are given below:

Peripheral Description	IC Name	Register / Port	Address	Remark
Programmable Peripheral Interface	8255-I UPPER	A B C Control	280CH 280DH 280EH 280FH	4 BYTE PAGE
Programmable Peripheral Interface	8255-II LOWER	A B C Control	2808H 2809H 280AH 280BH	4 BYTE PAGE
Real Time Clock (Optional)	58167	Second Minute Hours Week Day Date Month INT status INT Control	2842H 2843H 2844H 2845H 2846H 2847H 2850H 2851H	2840 TO 285F IN ALL 32 BYTES PAGE
Programmable Timer / Counter	8253	Timer 0 Timer 1 Timer 2 Control	2818H 2819H 281AH 281BH	4 BYTE PAGE
Universal Synchronous Asynchronous Receiver Transmitter	8251	Data Control	2828H 282AH 281AH 281BH	2 OVERLAPPED LOCATION IN 4 BYTES PAGE

5.2) SAMPLE PROGRAMS:

PROGRAMMING EXAMPLE

The following sample programs are given here to make the user familiarize with the operation of the kit.

Note: The Letter b indicates a BLANK Space here in the Program Mnemonics

PROGRAM – 1:

PROGRAM TO DISPLAY " SUPERB " ON LCD DISPLAY.

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	75 81 58		MOV b 81, #58	Initialize SP
6003	12 06 1D		LCALL b 061D	Clear display
6006	90 60 28	AGN:	MOV b DPTR, #6028	Read 1st message
6009	12 06 06		LCALL b 0606	Display message
600C	79 FF	XX:	MOV b R1, # FFH	Call Delay routine lying at 0114 in system
600E	7A FF		MOV b R2, #FFH	
6010	12 01 14		LCALL b 0114	
6013	90 20 10		MOV b DPTR, #2010	Set cursor position
6016	E0		MOVX b A, @DPTR	Clear display
6017	12 06 1D		LCALL b 061D	
601A	12 01 14		L CALL b 0114	
601D	B4 00 05		CJNE b A, # 00, 05	Jump to CHK if not 00
6020	74 07		MOVA,# 07	
6022	F0	CHA:	MOVX b @ DPTR, A	
6023	80 D9		SJMP b D9	Jump to 6006
6025	14	CHK	DECA	
6026	80 FA		SJMP b FA	Jump to 6022
6028	20 20 20 20 53 55 50 45 52 42 03			MESSAGE. Enter this data using S Command
Execute the program from address 6000 and observe the message "SUPERB" on LCD				

PROGRAM – 2

TO CALCULATE ADDITION OF TWO NUMBERS STORED AT 6200H & 6201H MEMORY LOCATIONS AND SHOW THE RESULT ON LCD DISPLAY

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	75 81 58		MOV b 81, # 58	Initialize SP
6003	90 62 00		MOV b DPTR, #6200	Read 1st data
6006	E0		MOVX b A, @DPTR	
6007	F9		MOV b R1, A	Store in R1
6008	A3		INC b DPTR	Read next data
6009	E0		MOVX b A, @DPTR	

600A	29		ADD b A, R1	Add nos.
600B	A3		INC b DPTR	Store result
600C	F0		MOVX b @DPTR, A	
600D	12 06 1D		LCALL b 061D	Clear display
6010	12 05 D2		LCALL b 05D2	Call display
6013	02 00 06		LJMP b 0006	Return to command mode
Now enter the data at location 6200H onward using S Command				
6200	03			1st data
6201	04			2nd data
Execute the program from address 6000 and observe the result on LCD:- 6202 07				

PROGRAM – 3

TO CALCULATE MULTIPLLICATION OF TWO NUMBERS STORED AT 6200H & 6201H MEMORY LOCATIONS AND SHOW THE RESULT ON LCD DISPLAY.

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	75 81 58	START:	MOV b 81, # 58	Initialize SP
6003	90 62 00		MOV b DPTR, # 6200	Read 1st data
6006	E0		MOVX b A, @DPTR	
6007	78, 00		MOV b R0, # 00	
6009	F9		MOV b R1, A	Store in R1
600A	A3		INC b DPTR	
600B	E0		MOVX b A, @DPTR	Read Next Data
600C	FA		MOV b R2, A	Store in R2
600D	E8	XX	MOVA, R0	
600E	29		ADD b A, R1	Add No,s
600F	F8		MOV b R0, A	
6010	DA, FC		DJNZ b R2, FC	Jump to XX if R1 is zero
6012	12 06 1D		LCALL b 061D	Clear display
6015	E8		MOVA, R0	
6016	A3		INC b DPTR	
6017	F0		MOVX b @DPTR, A	Store result
6018	12 05 D2		LCALL b 05D2	Call display
601B	02, 00, 06		LJMP b 0006	Return to command mode.
NOTE:				

6200	First no.			1st data
6201	Second no.			2nd data
6202	Result			
Execute the program from address 6000 and observe the result at address 6202 on LCD Display				

PROGRAM– 4

PROGRAM TO SHOW ASCII CODE OF ANY KEY PRESSED ON THE KEYBOARD USING USEFUL ROUTINE GIVEN IN THE MANUAL. THIS PROGRAM HELPS USER TO USE ROUTINES OF THE KIT.

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	75 81 58	START:	MOV b 81, #58	Initialize SP
6003	12 06 1D	START1:	LCALL b 061D	Clear display
6006	12 05 59		LCALL b 0559	
6009	12 05 D2		LCALL b 05D2	
600C	80 F5		SJMP b F5	Jump to start1
SEE THE CODE OF KEY PRESSD :- E8C8 XX				
WHERE XX IS CODE OF KEY PRESSED AND E8C8 IS VALUE IN DPTR.				

PROGRAM – 5

PROGRAM TO CHECK THE NUMBER AT MEMORY LOCATION 6200H FOR BEING ODD OR EVEN. THIS PROGRAM DISPLAYS THE MESSAGE "ODD" OR "EVEN" FOR ODD NUMBER AND EVEN NUMBER RESPECTIVELY.

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	75 81 58	START:	MOV b 81, #58	Initialize SP
6003	12 06 1D		LCALL b 061D	
6006	90 62 00		MOV b DPTR, #6200	Read the data
6009	E0		MOVX b A, @DPTR	
600A	44 FE		ORLA, #FE	Compare for odd or even
600C	B4 FE 05		CJNE b A, #FE,05	
600F	90 60 20		MOV b DPTR, #6020	Display message
6012	80 03		SJMP b 03	Address for message
6014	90 60 25		MOV b DPTR, #6025	
6017	12 06 06		LCALL b 0606	
601A	02 00 06		LJMP b 0006	

Now Enter the data from address 6020 onward				
6020	45 56 45 4E 03		DB "EVEN", 03	Data for Even
6025	4F 44 44 03		DB "ODD", 03	Data for Odd
Now Enter the data at location 6200H				
6200	04			
SEE THE MESSAGE :- EVEN				
User can change the no. and see the result according to number				

PROGRAM – 6

PROGRAM TO DISPLAY EXCEL TECHNOLOGIES ON GIVING INTERRUPT (INTO) THROUTGH STEP KEY AVAILABLE ON THE KIT BOARD.

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	75 81 58	START:	MOV b 81, #58	Initialize interrupt
6003	D2 88		SETB b 88	Enable interrupt
6005	90 60 25		MOV b DPTR, #6025	
6008	AB 82		MOV b R3, 82	Mov DPL contents to R3
600A	AC 83		MOV b R4, 83	Mov DPH contents to R4
600C	90 20 20		MOV b DPTR, #2020	
600F	12 06 CD		LCALL b 06CD	
6012	12 06 1D		LCALL b 061D	Clear display
6015	90 60 47		MOV b DPTR, #6047	Display message
6018	12 06 06		LCALL b 0606	
601B	20 B2 FD		JB b B2, FD	Check in interrupt occurs
601E	75 A8 81		MOV b A8, #81	
6021	00		NOP	
6022	00		NOP	
6023	80 ED		SJMP b ED	SJMP to 6012
6025	75 A8 00		MOV b A8, #00	
6028	30 B2 FD		JNB b B2, FD	
602B	78 FF		MOV b R0, #FF	Delay
602D	D8 FE		DJNZ b R0, FE	
602F	30 B2 F6		JNB b B2,F6	JNB to 6028
6032	12 06 1D		LCALL b 061D	Clear Display
6035	90 60 4E		MOV b DPTR, #604E	
6038	12 06 06		LCALL b 0606	Display message
603B	78 05		MOV b R0, #05	
603D	79 FF		MOV b R1,# FF	Delay for change in

603F	7A FF		MOV b R2, #FF	message
6041	12 01 14		LCALL b 0114	
6044	D8 F8		DJNZ b R0, F8	DJNZ R0 to 6041
6046	32		RETI	
6047	45 58 43 45 4C 20 03		DB "EXCEL", 03	Enter the data for Excel using S Command
Now enter the message "Technologies" from location 6052H onward				
604E	54 45 43 48 4E 4F 4C 4F 47 49 45 53 03		DB "TECHNOLOGIES", 03	
Execute the program from 6000 and observe message "EXCEL" on the display. Now press "Step" key and see that the message "TECHNOLOGIES" is also added on the display. After some delay, the message EXCEL comes again.				

PROGRAM – 7

PROGRAM TO DISPLAY TIME USING RTC-58167 ON LCD. USER CAN CHANGE THE TIME ACCORDING TO HIM USING Sub_M/R COMMAND AT MEMORY LOCATION 2842 (Second), 2843 (minute), 2844 (hour).

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	75 81 58	START:	MOV b 81, #58	Initialize SP
6003	12 06 1D		LCALL b 061D	Clear display
6006	90 28 44		MOV b DPTR,#2844	
6009	E0		MOVX b A, @DPTR	Read hour
600A	FB		MOV b R3, A	
600B	7D 02		MOV b R5, #02	
600D	12 05 9E		LCALL b 059E	Display Hr.
6010	74 20		MOVA,#20	
6012	12 20 06		LCALL b 2006	Put space between hr. and minute
6015	90 28 43		MOV b DPTR, #2843	Read min.
6018	E0		MOVX b A, @DPTR	
6019	FB		MOV b R3, A	
601A	7D 02		MOV b R5, #02	
601C	12 05 9E		LCALL b 059E	Display min.
601F	74 20		MOVA, #20	
6021	12 20 06		LCALL b 2006	Put space between min. & sec.
6024	90 28 42		MOV b DPTR, #2842	

6027	E0		MOVX b A, @DPTR	Read seconds
6028	FB		MOV b R3, A	
6029	7D 02		MOV b R5, #02	
602B	12 05 9E		LCALL b 059E	Display seconds
602E	12 06 1D		LCALL b 061D	Clear Display
6031	7F, FF	DELAY	MOV b R7, # FF	
6033	7E FF		MOV b R6, # FF	
6035	DE FE		DJNZ b R6, FE	
6037	DF FA		DJNZ b R7, FA	
6039	02 60 03		LJMP b 6003	Loop Back
OBSERVE THE CLOCK ON THE LCD DISPLAY IN THE FORMAT:- Hr Mn Sc.				

PROGRAM -8

OUT ASCII CHARACTER ON CRT

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	90 62 00		MOV b DPTR, # 6200	
6003	E0		MOVX b A, @ DPTR	Load ASCII code
6004	12 07 31		LCALL b 0731	Display character on CRT
6007	02 00 06		LCALL b 0006	Return to command mode

Description

Load ASCII Code at location 6200. Execute the program from location 6000 and see the character displayed on CRT for corresponding ASCII code.

PROGRAM- 9

OUT ASCII DATA ON CRT

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	90 62 00		MOV b DPTR, # 6200	
6003	E0		MOVX b A, @ DPTR	Load ASCII code
6004	12 01 3A		LCALL b 013A	Display code on CRT
6007	02 00 06		LCALL b 0006	Return to command

				mode

Description

Load ASCII Code at location 6200. Execute the program from location 6000 and see the same data displayed on CRT for corresponding data.

PROGRAM –10

SPLITTING A BYTE INTO TWO NIBBLE

ADDRESS	OP-CODE	LABEL	MNEMONICS	COMMENTS
6000	90 62 00		MOV b DPTR, # 6200	
6003	E0		MOVX b A, @ DPTR	Load Data
6004	F8		MOV b R0, A	
6005	54, F0		ANLA, # F0	Mask lower nibble
6007	C4		SWAP b A	Place higher nibble at lower nibble position
6008	A3		INC b DPTR	
6009	F0		MOVX b @ DPTR, A	Store higher nibble
600A	12 06 1D		LCALL b 061D	Clear LCD
600D	12 05 D2		LCALL b 05D2	Display higher nibble
6010	74 20		MOVA,# 20	
6012	12 07 31		LCALL b 0731	Display blank space
6015	E8		MOVA, R0	Take data
6016	54 0F		ANLA, # 0F	Mask higher nibble
6018	A3		INC b DPTR	
6019	F0		MOVX b @DPTR, A	Store lower nibble
601A	12 05 D2		LCALL b 05D2	Display lower nibble
601D	02 00 06		LJMP b 0006	Return to command mode

Description

Load the data at location 6200. Execute the program from location 6000 and observe result on LCD.

5.3) MEMORY MAPPING:-

The System has total two sockets on the Board. Out of it one socket is allocated for the EPROM. This socket is of 28 Pin and at factory setting, this socket is defined to have 27C512 EPROM having the Monitor of the Kit.

The system provides onboard Scratch Pad RAM of 8K Byte using 6264. The Monitor uses part of this Area for temporary storage. The memory can be expanded to 32K

Apart from this a Battery backed up RAM area is also provided on board so that the user can write his programs in it. Although the scratch Pad RAM area (leaving aside the portion used by the monitor) can also be used for writing the programs, but in case of power failure, the program will get washed out.

The memory addresses of various chips on the board are given here for your reference.

S.NO.	DESCRIPTION	CHIPS USED	MEMORY ADDRESS	CAPACITY
1	EPROMs	27512	0000H – 1FFFH	64K Byte
2	SCRATCH PAD	6264	2000H – 27FFH	2 K Byte (The other area of this RAM is used by monitor
3	Battery Backed up RAM	62256	4000H 0 7FFFH	32K Byte

IMPORTANT NOTE:

The Monitor uses the RAM area from 8000H to EFFFH (4 k Bytes) for its temporary storage. It is therefore advised to the user not to use this area for writing the programs.

5.4) USEFUL ROUTINES AND THEIR DESCRIPTION

The ET-8051LCD kit has its monitor program and ROM based assembler and disassembler. User can write program from memory Location says 6000h onwards. Our monitor program also have some software routine, user can use those. To use these routines in user's programs, user may need their addresses and parameter (I/P or O/P) in registers / memory locations. These routines may output (O/P) some value in registers and wile doing so may destroy (D) previous contents of these registers.

Some additional information about the addresses and short forms used are as follows:

SHORT FORMS USED AND THEIR MEANINGS

I/P = Input value
O/P = Output value
D = Demolish

Control + U = ACK,
Pressing Esc returns to command mode.

ADDRESS	ROUTINE	DISCRIPTION
0102	INVERT	If called, this routine will twist the Acc through 8 places i.e. inverts bit string of accumulator
0114	DELAY	The routine which introduces delay. Delay equation is R2 (R1*2+3) +12 machine cycles.
0123	ESCAPE	Polls I/P key once if Esc pressed then aborts and goes to command mode, else returns destroy accumulator.
013A	ASCII	Converts accumulator in two ASCII, MSB first, & O/Ps it to serial port. & Updates additive checksum at 19th byte destroys accumulator.
0162	BINARY	Converts ASCII code in acc to binary. If accumulator has valid bin data in acc, else shows "Format" error & goes to command mode. Destroys accumulator.
017F	SERIAL 2 BYTE	Gets two ASCIIs from serial link & gives compact hex number in accumulator & updates additive checksum at 19th byte.
019A	SUBTRACTION	Subtraction, R6-T5 minus 1D-1C→1D-1C answer is correct if R6-R5> 1D – 1C. Destroy accumulator.
01B3	WAIT	Gives WAIT message. Of course, if called, destroys DPTR.
01BD	COMPARE	Compares R4-R3 & R6-R5, (R4, R6 EQUMSD). If R4-R3< R6 R5 →C = 1, OD5 = 0 IF R4-R3> = R6-R5 → C = 0 OD5 = 0 IF R4-R3> = R6-R5 → C = 0, OD5 = 1. Destroys accumulator.
01D5	SPACE	Gives 2 spaces (ASCII 20H) at Console Out.
01E4	PULSE	Useful utility which detects a pulse on into. I.e. returns if you press a key at right corner of the CPU board, unless waits.
01EB	SERIAL OUT	A = ASCII data to be outputted through serial link.
01F3	S OUT PRINT	A = ASCII, O/P to remote device through serial link also write to printer if enabled.
0230	READ SERIAL	Polls serial port till a character is received & fetches it in accumulator.
0242	CHECK	Polls serial port once if C = 1, A ← valid code, C

	SERIAL	= 0, No char received. Destroys accumulator.
0248	CHECK DPTR	Checks if DPTR = 1F-1E? , If not decrements DPTR. R6 & R5 destroyed.
027A	DECREMENT	Decrements 1DH –1CH BY 1.
028E	COMPARE MEMORY	Compares memory blocks with each other. DPTR = SRC STRT address R4-R3 = SRC end address R6-R5 = DST STRT address if locks matched →CY = 1 DPTR – mismatch address if any destroys DPTR.
02BC	COMPARE DATA	Compares given byte in acc with given memory area. DPTR = start address R4-R3 = END address C = 1 →matched. Destroys DPTR.
02EF	INCREMENT	Checks if DPTR = R4R3? And increments DPTR if not destroys R6 & R5.
0302	MOVE BLOCK	Moves a page (R2) long. DPTR = SRC STRT address R5-R6 = DST STRT address, destroys DPTR.
031F	FILL DATA	Fills a buffer with a given data in accumulator DPTR = STRT address R4-R3 = END address destroys DPTR.
032E	TRANSFER	Moves a given page from top to bottom DPTR = STRT adds, R4-R3 = end address R6-R5 = DSTN STRT address destroys DPTR.
034A	REVERSE TRANSFER	Moves a given page for bottom to top DPTR = STRT adds, R4-R3 = end adds R6-R5 = DSTN STRT address. Destroys DPTR.
038E	COMPLETE	If your task is done call this address to show done. Destroys DPTR.
0398	BRANCH	With A = code, DPTR = jmp table pointer & R4-R3 = CODE TABLE BASE IT FINDS A BRANCH ADDS c= 1, if code found & DPTR is jump pointer. Destroys DPTR.
03C6	WRITE	A write care routine with R6-R5 as pointer.
03D8	WRITE CARE	A write care routine with DPTR as pointer gives error if PC is in I/P mode. Destroys DPTR.
041F	WRITE PROTECT	Enables write protect of RAM at 6000H.
0451	PROTECT DATA	After calling this, RAM at 6000 H is protected from hazardous writes.
045F	RECEIVE	Receives a no IR4-R3 with given name pointed to by DPTR. R4-R3 default set value, R5 = limit no, R6 = length.

047B	NUMBER	R4-R3 ←limit nom R6 = no of digits. This routine gets a no., (R6) long & 1T. or Equ. (R5) in R4-R3 from the console in. C = 1→ valid entry 1 R4 R3 destroys R5, R6, accumulator.
0503	KEY	Polls Keyboard once, if a key is pressed, A ← key-code, C=1 destroys accumulator, DPTR.
050E	READ MESSAGE	Accumulator = no of messages, DPTR ←STRT addr of message table. This routine authorizes a message with CR and gets message no. in accumulator.
0559	READ KEYBOARD	Polls keyboard controller continuously, until a key is pressed. Accumulator ←valid ASCII code of key pressed.
0571	READ CONSOLE	Reads a character from current console & checks for CR/Ctrl-U; if Esc, goes to command mode; if 0D5 = 1 then key = CR / Ctrl-U & if C = 1 then CR else Ctrl-U; if 0D5 = 0 some other key (in accumulator).
0589	OUT HEX	Acc = XN, HEX no. (LSD), O/P to console at current cursor. Cursor automatically adjusted to next location.
059E	OUT NUMBER	A right hand justified no in R4 R3 (Hex), R5 = no of digits O/P to console at current cursor.
05D2	OUT STRING	A = DATA (Hex) DPTR = address (Hex) O/P to current cursor position as four digits of address followed by two blanks followed by two digits of data
0606	DISPLAY MESSAGE	Message out routine, DPTR = ASCII message table start pointer, table ending in ETX. Outputs a message to current cursor position destroys DPTR.
061D	CLEAR	Sends carriage returns & line feed to console.
062C	Write key	A ← ASCII char to be outputted to kbd & display controller at current cursor position, cursor modified automatically.
06B3	U NEAR	Linear search routine A = I/P no., DPTR = pointer to table to search through A = O/P no. CY = 1 ← valid code 1 accumulator destroys DPTR.
06CD	TERMINATE	This utility will terminate any routine or interrupt by putting 02 XY (jmp instruction), at the location pointer to by DPTR. I/p – 1)DPTR = location at which you want to initialize the vector 2) R4 = X R3 = Y ; this is the address where you want to get

		vectored.
0731	PRINT ASCII	Acc. Data O/P to serial Port

Notes:-

1. In normal cases user may put 22H (RET) at the end of his/her program, which will return user to command mode with saving most of the registers except PC & SP.
2. In single stepping user have to use 32H (RET1), instead of 22H.
3. If user write 02 00 06 (LJMP 0006) at the end, then it will go to command mode with saving all registers.
4. A) –02 06D8H – (LJMP 06D8) jump to command mode.
   B) – 02 06DBH – (LJMP 06DB) jump to command mode, without disturbing the display.

CHAPTER –6
DETAILS OF CONNECTORS

The 8031 Based Kit here provides lot of on Board interfaces for the user interacts. These interfaces can be accessed through various connectors. The details of these connectors are given here for your reference.

6.1 DETAIL OF CONNECTOR J1: (8255 UPPER)

The various signals of the first 8255 are brought out on this connector J1.

PIN	SIGNALS	PINS	SIGNALS
1	P1C4	14	P1B1
2	PIC5	15	P1A6
3	P1C2	16	P1A7
4	P1C3	17	P1A4
5	P1C0	18	P1A5
6	P1C1	19	P1A2
7	P1B6	20	P1A3
8	PIB7	21	P1A0
9	P1B4	22	P1A1
10	P1B5	23	P1C6
11	P1B2	24	P1C7
10	PIB3	25	GND
13	PIB0	26	GND

6.2 DETAIL OF CONNECTOR J2: (8255 LOWER)

The various signals of 8255-2 are brought out at this connector J2:

PIN	SIGNALS	PIN	SIGNALS
1	P2C4	14	P2B1
2	P2C5	15	P2A6
3	P2C2	16	P2A7
4	P2C3	17	P2A4
5	P2C0	18	P2A5
6	P2C1	19	P2A2
7	P2B6	20	P2A3
8	P2B7	21	P2A0
9	P2B4	22	P2A1
10	P2B5	23	P2C6
11	P2B2	24	P2C7

12	P2B3	25	GND
13	P2B0	26	GND

6.3 DETAILS OF CONNECTOR J3: (BUS)

PIN	SIGNAL	PINS	SIGNAL
1	VCC	2	VCC
3	GND	4	GND
5	D0	6	D1
7	D2	8	D3
9	D4	10	D5
11	D6	12	D7
13	A0	14	A1
15	A2	16	A3
17	A4	18	A5
19	A6	20	A7
21	A8	22	A9
23	A10	24	A11
25	A12	26	A13
27	A14	28	A15
29	RD*	30	WR*
31	NC	32	PSEN
33	INT0	34	INT1
35	1/373	36	NC
37	NC	38	NC
39	RXD	40	TXD
41	NC	42	NC
43	T0	44	T1
45	RST	46	CLK
47	EA*	48	ALE
49	JP4 000	50	NC

Note: * indicates Active Low Logic

6.4 DETAILS OF CONNECTOR J4:

This connector is for the further expansion by the manufacturer and not for the user or student.

6.5 DETAILS OF CONNECTOR J5: (8253 TIMER/COUNTER)

PINS	SIGNALS	PINS	SIGNALS
1	GATE0	2	OUT0
3	GATE1	4	OUT1
5	GATE2	6	OUT2
7	NU	8	NU
25	GND	28	GND

The remaining pins of this connector are not used, so not shown in this detail.

6.6 DETAIL OF 9 PIN SERIAL CONNECTOR USING 8251 (OPTIONAL)

This connector is supplied at the back of the Box only on Demand

The Details about the 9-pin Serial interface using the USART chip (Universal Synchronous Asynchronous Receiver Transmitter) are given here for your reference. The Kit uses this chip for the serial interface with the PC/AT or any other serial device.

PINS	SIGNALS	PINS	SIGNALS
1	DCD	6	DSR
2	RX	7	RTS
3	TX	8	CTS
4	DTR	9	NC
5	GND		

6.7 DETAILS OF CONNECTOR J6: (POWER CONNECTOR)

PINS	SIGNALS	PINS	SIGNALS
1	+5V	3	NC
2	GND	4	NC

CHAPTER –7
PROGRAM FOR ON BOARD APPLICATIONS

The Kit has following applications on the Board of the Kit.

1) **Digital Input through DIP Switches.**
2) **Digital Outputs through eight LEDs**
3) **Traffic Light Controller**
4) **Stepper Motor Controller**
5) **Graphical LCD Display**

These on Board applications are connected through the 8255-II provided on the Board. These applications can be selected through a DIP Switch provided on the Board and named as APL-SELECT SWITCH. The selection can be done as follows

Sl. No.	DIP Switch Position	Application Selected
1	DIP SW-1-------**ON** DIP SW 2 to 8--OFF	Digital Input & Digital Output
2	DIP SW-1-------OFF DIP SW-2-------**ON** DIP SW-3 to 8--OFF	Stepper Motor
3	DIP SW-1-------OFF DIP SW-2-------OFF DIP SW-3-------**ON** DIP SW-4 to 8--OFF	Graphical LCD
4	DIP SW-1to 3---OFF DIP SW-4 to 5 –**ON** DIP SW-6 to 8--OFF	Traffic Light Controller

APLICATION-1

DIGITAL INPUT & DIGITAL OUTPUT

The Digital input in this application is provided through a 8 way DIP switch named as Digital Input. The inputs are normally pulled Down and will give logic Zero when switch is OFF. These Digital inputs gets connected to PORT A of the 2nd 8255. The addresses of these are given in chapter-6 of the manual.

The Digital outputs have eight LEDs connected to them. The outputs are provided through PORT B of 2nd 8255. The LED glows when the Digital output is High.

EXAMPLE:

The program written here will take the Digital Input from DIP Switches and will output the Logic to output LED. That means if any input is ON, the corresponding LED will also be ON.

ADDRESS	CODES	LABEL	MNEMONICS	REMARK
6000	90 28 0F		MOV DPTR ,# 280F	Initialize 8255-1; Port B & C as Output, Port A as Input.
6003	74 90		MOV A, # 90	
6005	F0		MOVX @ DPTR ,A	
6006	90 28 0C	START	MOVX DPTR,# 280C	Read port A switches.
6009	E0		MOVX A,@ DPTR	
600A	90 28 0D		MOV DPTR,# 280D	Output the scanned switches on port B LED's
600D	F0		MOVX @ DPTR,A	
600E	80 F6		SJMP F6	Jump to START

APLICATION-2

STEPPER MOTOR CONTROLLER:

The stepper Motor controller is implemented using the current driver ULN2803 which is given input through PORT B of the 2nd 8255. The Driver can drive two stepper Motors named as stpr1 and stpr2 on the Board at the left hand side.

Connect the motor on the connector stpr1 and enter the program given below. The power supply points of the Motor need to be +5V or +12V as the case may be.

Enter the following program and execute the same.

ADDRESS	CODES	LABEL	MNRMONICS	REMARK
6000	75 81 65	START	MOV 81,# 65	Initialize stack
6003	90 28 0F		MOV DPTR,# 280F	Initialize 8255 – I;All Ports as Output port.
6006	74 80		MOVA,# 80	
6008	F0		MOVX @DPTR,A	
6009	90 28 0D		MOV DPTR,# 280D	
600C	74 **FA**	CLK-1	MOVA,# **FA**	
600E	F0	OUT	MOVX @DPTR,A	Out Data FA on Port C
600F	11 22		ACALL 6022	Give Delay
6011	74 **F6**		MOVA,# **F6**	
6013	F0	OUT	MOVX @DPTR,A	Out Data F6 on Port C
6014	11 22		ACALL 6022	Give Delay
6016	74 **F5**		MOVA# **F5**	Read no. of steps

6018	F0	OUT	MOVX @DPTR,A	Out Data F5 on Port C
6019	11 22		ACALL 6022	Give Delay
601B	74 **F9**		MOVA, **F9**	
601D	F0	OUT	MOVX @DPTR,A	Out Data F9 on Port C
601E	11 22		ACALL 6022	
6020	80 DE		Loop Back	
6022	7E **FF**		MOV R6,FF	**Create delay**
6024	7F FF	LOOP	MOV R7,# FF	
6026	DF FE		DJNZ R7,FE	
6028	DE FA		DJNZ R6,FA	
602A	22		RET	Return

Note: On executing the program, the motor will run continuously, with the step delay. However the delay can be changed by changing the content of location 6023 to say 80.

The Direction of the motor can be changed by reversing the sequence of the codes as F9, F5, F6 and FA instead of FA, F6, F5 and F9 as used earlier.

The Motor can also be programmed to move certain fixed no. of steps by changing the program suitably

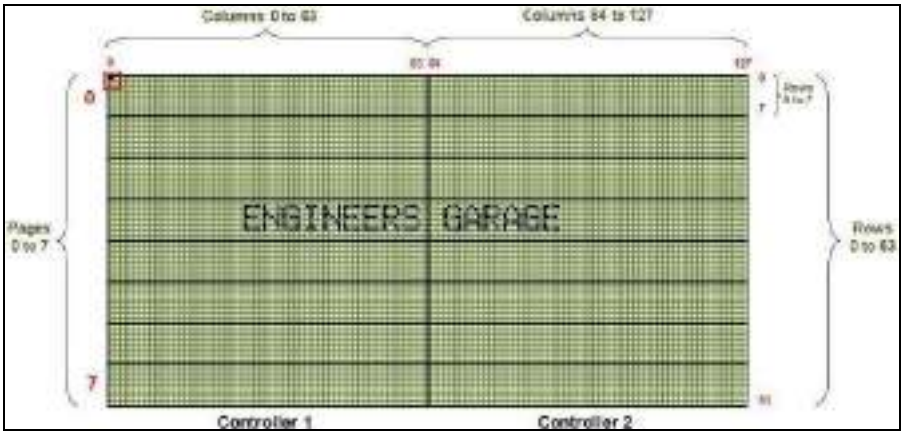
APLICATION-3

GRAPHICAL LCD DISPLAY:

In today's environment Graphical LCDs are very commonly used in various applications like Mobile phones, computer Display, vending Kiosks etc. This Kit uses a graphical LCD with resolution of 128*64. The eight Data lines of the display are connected to the port B of 2nd 8255. The controls signals are connected to the Port A lines like PA0, PA1 and PA2 of the 2nd 8255 .

BASIC ABOUT GRAPHICAL LCD

It use 2 KS0108 Controller to executes its internal operation 128x64 graphical lcd is divided into two equal halves with each half being controlled by a separate KS0108 controller. This division is called paging scheme and is explain as follow:

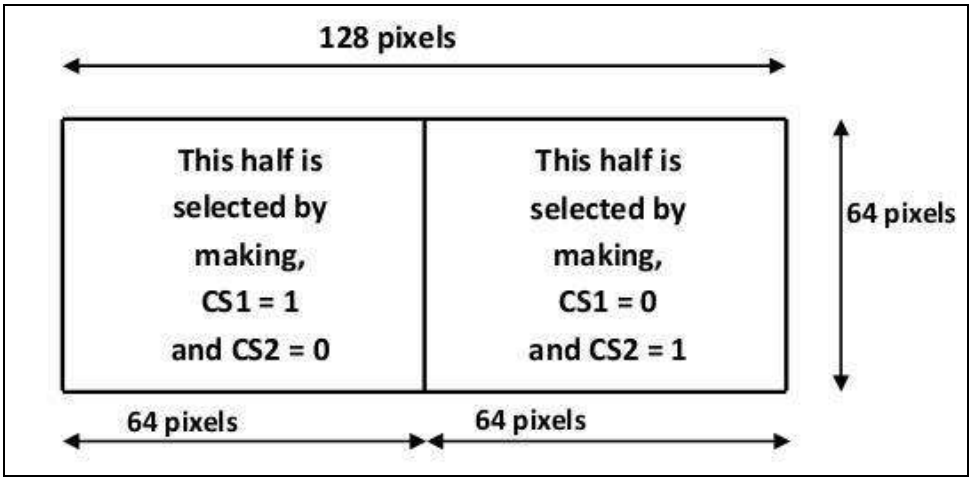


Each half consists of 8 pages and each page consist of 8 row and 64 columns.

Row 0	Row 0
Row 1	Row 1
Row 2	Row 2
Row 3	Row 3
Row 4	Row 4
Row 5	Row 5
Row 6	Row 6
Row 7	Row 7

Each row is 8 pixel thick

- So two horizontal pages make 128 (64x2) columns and 8 vertical pages to makes 64 rows (8x8) and hense 128x64 = 8192 pixels.
- To select each half separately we have to use as following values of CS1 and CS2:



PIN FUNCTION OF GLCD:

- **Graphical Lcd has Following Pins and their Functions are:**

PIN NO.	SYMBOL	DESCRIPTION	FUNCTION
1	VSS	GROUND	0V (GND)
2	VDD	POWER SUPPLY FOR LOGIC CIRCUIT	+5V
3	V0	LCD CONTRAST ADJUSTMENT	
4	RS	INSTRUCTION/DATA REGISTER SELECTION	RS = 0 : INSTRUCTION REGISTER RS = 1 : DATA REGISTER
5	R/W	READ/WRITE SELECTION	R/W = 0 : REGISTER WRITE R/W = 1 : REGISTER READ
6	E	ENABLE SIGNAL	
7	DB0	DATA INPUT/OUTPUT LINES	8 BIT: DB0-DB7
8	DB1		
9	DB2		
10	DB3		
11	DB4		
12	DB5		
13	DB6		
14	DB7		
15	CS1	CHIP SELECTION	CS1=1,CHIP SELECT SIGNAL FOR IC1
16	CS2	CHIP SELECTION	CS2=1,CHIP SELECT SIGNAL FOR IC2
17	RST	RESET SIGNAL	RSTB=0,DISPLAY OFF,DISPLAY FROM LINE 0.
18	VEE	NEGATIVE VOLTAGE FOR LCD DRIVING	-10V
19	Vout	Connected to Port	
20	LED-	SUPPLY VOLTAGE FOR LED-	0V

FUNCTION OF R/W AND RS:

R/W	RS	FUNCTION
0	0	SEND INSTRUCTION TO DDRAM
0	1	DATA WRITE FROM INPUT REGISTER TO DDRAM (ON LCD)
1	0	STATUS CHECK (BUSY READ)
1	1	DATA READ FROM DDRAM TO OUTPUT REGISTER

DDRAM: DATA DISPLAY RAM

BASIC PROCEDURE TO WORK ON GLCD:

1. LCD initialization

2. Page selection
3. Column selection
4. Data Display

1. LCD INTIALIZATION

a. Put these value in data resister

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	**0**	**1**	**1**	**1**	**1**	**1**	**D**

D = 1 , LCD ON

D = 0 , LCD OFF (DISSAPPEARS DATA)

b. CS1 = 1, CS2 = 1 (Both halves Selected)

c. RS = 0, R/W = 0 (To select Instruction mode)

d. EN = 1 (ENABLE = 1)

e. DELAY

f. EN = 0 (To latch data into the input register)

2. PAGE SELECTION

a)Put these value in data resister

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	**0**	**1**	**1**	**1**	**X3**	**X2**	**X1**

X3	X2	X1	PAGE NUMBER
0	0	0	PAGE 0
0	0	1	PAGE 1
0	1	0	PAGE 2
0	1	1	PAGE 3
1	0	0	PAGE 4
1	0	1	PAGE 5
1	1	0	PAGE 6
1	1	1	PAGE 7

a. CS1 = 1, CS2 = 1 (Both halves Selected)

b. RS = 0, R/W = 0 (To select Instruction mode)

c. EN = 1 (ENABLE = 1)

d. DELAY

e. EN = 0 (To latch data into the input register)

3. COLUMN SELECTION

a) Put these value in data resister

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	**1**	**Y5**	**Y4**	**Y3**	**Y2**	**Y1**	**Y0**

Range of Column Selection is Column 0 to Column 63 for each halves.

CS1	CS2	FUNCTION
0	0	Neither Half Selected
0	1	Second Half Selected
1	0	First Half Selected
1	1	Both Half Selected

Y5	Y4	Y3	Y2	Y1	Y0	COLUMN NUMBER
0	0	0	0	0	0	COLUMN 0
0	0	0	0	0	1	COLUMN 1
0	0	0	0	1	0	COLUMN 2
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	0	1	COLUMN 61
1	1	1	1	1	0	COLUMN 62
1	1	1	1	1	1	COLUMN 63

a. RS = 0, R/W = 0 (To select Instruction mode)
b. EN = 1 (ENABLE = 1)
c. DELAY
d. EN = 0 (To latch data into the input register)

4. DISPLAY DATA

- After Page and Column are Selected Data can be written to that particular location of GLCD. For example if you want to display all 8 pixel of that particular column you can give data as “FF” So all 8 pixel will be darken.
- Data must be written by first selecting column address and then data can be given to it.

a. Put the any desire value in data resister

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
X	X	X	X	X	X	X	X

b. Select CS1, CS2 based on requirement.
c. RS = 1, R/W = 0 (To select Data write mode)
d. EN = 1 (ENABLE = 1)
e. DELAY
f. EN = 0 (To latch data into the input register)

CONTROL WORD GENERATION

1.FOR LCD INITIALIZATION

DATA WORD

PA7	**PA6**	**PA5**	**PA4**	**PA3**	**PA2**	**PA1**	**PA0**	**PORT A**
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX DW
0	0	1	1	1	1	1	D	
0	0	1	1	1	1	1	1	**3F**

COMMAND WORD

PC7	**PC6**	**PC5**	**PC4**	**PC3**	**PC2**	**PC1**	**PC0**	**PORT C**
NC	NC	NC	CS2	CS1	EN	R/W	RS	FUNCTION
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX CW
0	0	0	1	1	1	0	0	**1C**

PC7	**PC6**	**PC5**	**PC4**	**PC3**	**PC2**	**PC1**	**PC0**	**PORT C**
NC	NC	NC	CS2	CS1	EN	R/W	RS	FUNCTION
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX CW
0	0	0	1	1	0	0	0	**18**

2.FOR PAGE 0 SELECTION

DATA WORD

PA7	**PA6**	**PA5**	**PA4**	**PA3**	**PA2**	**PA1**	**PA0**	**PORT A**
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX DW
1	0	1	1	1	X3	X2	X1	
1	0	1	1	1	0	0	0	**B8**

COMMAND WORD

NOTE: Command Word Remain Same For Lcd Initialization, Page Selection and Column Selection. It changes only for data display.

3.FOR COLUMN 0 SELECTION

DATA WORD

PA7	**PA6**	**PA5**	**PA4**	**PA3**	**PA2**	**PA1**	**PA0**	**PORT A**
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX DW
0	1	Y5	Y4	Y3	Y2	Y1	Y0	
0	1	0	0	0	0	0	0	**40**

4.FOR DATA DISPLAY "|"

i.e Straight line of 8 pixel on FIRST halves

DATA WORD

PA7	**PA6**	**PA5**	**PA4**	**PA3**	**PA2**	**PA1**	**PA0**	**PORT A**
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX DW
X	X	X	X	X	X	X	X	
1	1	1	1	1	1	1	1	**FF**

DATAWR1 WORD/CONTROL WORD

PC7	**PC6**	**PC5**	**PC4**	**PC3**	**PC2**	**PC1**	**PC0**	**PORT C**
NC	NC	NC	CS2	CS1	EN	R/W	RS	FUNCTION
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX CW
0	0	0	0	1	1	0	1	**0D**

PC7	**PC6**	**PC5**	**PC4**	**PC3**	**PC2**	**PC1**	**PC0**	**PORT C**
NC	NC	NC	CS2	CS1	EN	R/W	RS	FUNCTION
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	HEX CW
0	0	0	0	1	0	0	1	**09**

DATAWRITE CONTROL WORD

HEX CODE	**FUNCTION**
0D,09	SELECT FIRST HALF OF LCD
1D,19	BOTH HALF SELECTED
15,11	SECOND HALF SELECT

HARDWARE CONNECTION

LCD PIN FUNCTION	**LCD PIN NO.**	**8255 26 PIN CONNECTOR/PIN NO.**	**8255 PIN FUNCTION**	**8255 IC PIN NO.**
DB0	7	21	PORT A PA0	4
DB1	8	22	PA1	3
DB2	9	19	PA2	2
DB3	10	20	PA3	1
DB4	11	17	PA4	40
DB5	12	18	PA5	39
DB6	13	15	PA6	38
DB7	14	16	PA7	37
LEDK	20	GND		
GND	1	GND		
VCC	2	+5V		
V(contrast adust)	3	+5V (Variable through POT)		
LEDA	19	+5V(through Resister)		

RS	4	5	PORT C PC0	14
R/W	5	6	PC1	15
EN	6	3	PC2	16
CS1	15	4	PC3	17
CS2	16	1	PC4	13
RESET(active low)	17	ON SWITCH		
VOUT	18	ON PORT		

DATA ON MEMORY LOCATION 2000:2000 TO 2000:2080

CHARECTER	**ADDRESS**						
SPACE/BLANK	2000:2000	00	00	00	00	00	00
SPACE/BLANK	2000:2006	00	00	00	00	00	
E	2000:200B	FE	92	92	92	82	00
X	2000:2011	C6	28	10	28	C6	00
C	2000:2017	7C	82	82	82	44	00
E	2000:201D	FE	92	92	92	82	00
L	2000:2023	FE	80	80	80	80	00
SPACE/BLANK	2000:2029	00	00	00	00	00	
T	2000:202E	02	02	FE	02	02	00
E	2000:2034	FE	92	92	92	82	00
C	2000:203A	7C	82	82	82	44	00
H	2000:2040	FE	10	10	10	FE	00
N	2000:2046	FE	08	10	20	FE	00
O	2000:204C	7C	82	82	82	7C	00
L	2000:2052	FE	80	80	80	80	00
O	2000:2058	7C	82	82	82	7C	00
G	2000:205E	7C	82	92	92	F4	00
I	2000:2064	00	82	FE	82	00	00
E	2000:206A	FE	92	92	92	82	00
S	2000:2070	8C	92	92	92	62	00
SPACE/BLANK	2000:2076	00	00	00	00	00	00
SPACE/BLANK	2000:207C	00	00	00	00	00	00

NOTE: We have used 7 row and 5 columns to create a character.

PROGRAM :

PROGRAM TO DISPLAY "EXCEL TECHNOLOGIES" at center with square border on top and bottom of the GRAPHICAL LCD USING 8085 KIT THROUGH 8255.

ADDRESS	**OP**	**LABEL**	**MNEMONICS**	**COMMENTS**

	CODE			
6000	90 28 0F	START:	MOV DPTR,#280F	CWR ADDRESS
6003	74 80		MOV A,#80	CW OF 8255
6005	F0		MOVX @DPTR,A	OUT CW ON CWR
6006	90 28 0D		MOV DPTR,#280D	PORT B ADDRESS
6009	74 3F		MOV A,#3F	CW TO INTIALISE LCD
600B	F0		MOVX @DPTR,A	OUT CW ON PORT A
600C	11 51		ACALL 6051	ACALL COMMAND
600E	51 00		ACALL 6200	ACALL CLEAR LCD
6010	74 B8		MOV A,#B8	PAGE 0 SELECT
6012	F0		MOVX @DPTR,A	OUT DATA
6013	11 51		ACALL 6051	ACALL COMMAND
6015	7A 40		MOV R2,#40	COLUMN ADDRESS
6017	7B 08		MOV R3,#08	COUNTER
6019	71 00	LP1:	ACALL 6300	ACALL SQUARE
601B	0A		INC R2	INCREMENT COLUMN ADDRESS
601C	DB FB		DJNZ R3,FB	DJNZ R3,LP1
601E	00		NOP	NO OPERATION
601F	74 BF		MOV A,#BF	PAGE 7 SELECT
6021	F0		MOVX @DPTR,A	OUT DATA
6022	11 51		ACALL 6051	ACALL COMMAND
6024	7A 40		MOV R2,#40	COLUMN ADDRESS
6026	7B 08		MOV R3,#08	COUNTER
6028	71 00	LP2:	ACALL 6300	ACALL SQUARE
602A	0A		INC R2	INCREMENT COLUMN ADDRESS
602B	DB FB		DJNZ R3,FB	DJNZ R3,LP2
602D	00		NOP	NO OPERATION
602E	91 00		ACALL 6400	ACALL EXCEL
6030	02 00 06		LJMP 0006	STOP PROGRAM

SUBROUTINE 1 FOR DATAWRITE

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6040	90 28 0C	DATAWR:	MOV DPTR,#280C	PORT A SELECTED
6043	74 1D		MOV A,#1D	DW
6045	F0		MOVX @DPTR,A	OUT DATA WORD
6046	11 80		ACALL 6080	ACALL DELAY
6048	74 19		MOV A,#19	DW

604A	F0		MOVX @DPTR,A	OUT DATA WORD
604B	11 80		ACALL 6080	ACALL DELAY
604D	90 28 0D		MOV DPTR,#280D	PORT B SELECT
6050	22		RET	RETURN TO MAIN

SUBROUTINE 2 FOR COMMAND

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6051	90 28 0C	COMMAND:	MOV DPTR,#280C	PORT A SELECTED
6054	74 1C		MOV A,#1C	CW
6056	F0		MOVX @DPTR,A	OUT COMMAND WORD
6057	11 80		ACALL 6080	ACALL DELAY
6059	74 18		MOV A,#18	CW
605B	F0		MOVX @DPTR,A	OUT WORD
605C	11 80		ACALL 6080	ACALL DELAY
605E	90 28 0D		MOV DPTR,#280D	PORT B SELECT
6061	22		RET	RETURN TO MAIN

SUBROUTINE 3 FOR DATAWRITE1

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6062	90 28 0C	DATAWR1:	MOV DPTR,#280C	PORT A SELECTED
6065	74 0D		MOV A,#0D	DW
6067	F0		MOVX @DPTR,A	OUT DATA WORD
6068	11 80		ACALL 6080	ACALL DELAY
606A	74 09		MOV A,#09	DW
606C	F0		MOVX @DPTR,A	OUT DATA WORD
606D	11 80		ACALL 6080	ACALL DELAY
606F	90 28 0D		MOV DPTR,#280D	PORT B SELECT
6072	22		RET	RETURN TO MAIN

SUBROUTINE 4 FOR DELAY

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6080	00	DELAY:	NOP	NO OPERATION
6081	00		NOP	

6082	00		NOP	
6083	22	RET	NOP	RETRUN
6084	00			

SUBROUTINE 5 FOR DATAWRITE2

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6085	90 28 0C	DATAWR2:	MOV DPTR,#280C	PORT A SELECTED
6088	74 15		MOV A,#15	DW
608A	F0		MOVX @DPTR,A	OUT DATA WORD
608B	11 80		ACALL 6080	ACALL DELAY
608D	74 11		MOV A,#11	DW
608F	F0		MOVX @DPTR,A	OUT DATA WORD
6090	11 80		ACALL 6080	ACALL DELAY
6092	90 28 0D		MOV DPTR,#280D	PORT B SELECT
6095	22		RET	RETURN TO MAIN

SUBROUTINE 6 FOR CLEAR LCD

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6200	7C BF	CLEAR LCD:	MOV R4,#BF	PAGE 7 ADDRESS
6202	7B 08		MOV R3,#08	PAGE COUNTER
6204	EC	LOP2:	MOV A,R4	LOAD PAGE ADDRESS
6205	F0		MOVX @DPTR,A	OUT PAGE ADDRESS ON PORT A
6206	11 51		ACALL 6051	CALL COMMAND
6208	7A 7F		MOV R2,#7F	COLUMN 63 ADDRESS
620A	7D 40		MOV R5,#40	COLUMN COUNT
620C	EA	LOP1:	MOV A,R2	LOAD COLUMN ADDRESS
620D	F0		MOVX @DPTR,A	OUT ADDRESS
620E	11 51		ACALL 6051	CALL COMMAND
6210	74 00		MOV A,#00	DATA FOR CLEAR LCD
6212	F0		MOVX @DPTR,A	OUT DATA
6213	11 40		ACALL 6040	CALL DATAWR

6215	1A		DEC R2	DECREMENT COLUMN ADDRESS
6216	DD F4		DJNZ R5,F4	DJNZ R5,LOP1
6218	1C		DEC R4	DECREMENT PAGE ADDRESS
6219	DB E9		DJNZ R3,E9	DJNZ R3,LOP2
621B	22		RET	RETURN TO MAIN PROGRAM

SUBROUTINE 7 FOR SQUARE PRINT []

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6300	7C 04	SQUARE:	MOV R4,#04	LOAD COUNT
6302	EA	AG1:	MOV A,R2	LOAD COLUMN ADDRESS
6303	F0		MOVX @DPTR,A	OUT ADDRESS ON PORT A
6304	11 51		ACALL 6051	CALL COMMAND
6306	74 00		MOV A,#00	LOAD DATA
6308	F0		MOVX @DPTR,A	OUT DATA ON PORT A
6309	11 40		ACALL 6040	CALL DATAWR
630B	0A		INC R2	INCREMENT COLUMN ADDRESS
630C	DC F4		DJNZ R4,F4	DJNZ R4,AG1
630E	00		NOP	NO OPERATION
630F	EA		MOV A,R2	LOAD COLUMN ADDRESS
6310	F0		MOVX @DPTR,A	OUT ADDRESS ON PORT A
6311	11 51		ACALL 6051	CALL COMMAND
6313	74 FF		MOV A,#FF	LOAD DATA
6315	F0		MOVX @DPTR,A	OUT DATA ON PORT A
6316	11 40		ACALL 6040	CALL DATAWR
6318	0A		INC R2	INCREMENT COLUMN ADDRESS
6319	7C 02		MOV R4,#02	LOAD COUNT
631B	EA	AG2:	MOV A,R2	LOAD COLUMN ADDRESS
631C	F0		MOVX @DPTR,A	OUT ADDRESS ON PORT A
631D	11 51		ACALL 6051	CALL COMMAND

631F	74 81		MOV A,#81	LOAD DATA
6321	F0		MOVX @DPTR,A	OUT DATA ON PORT A
6322	11 40		ACALL 6040	CALL DATAWR
6324	0A		INC R2	INCREMENT COLUMN ADDRESS
6325	DC F4		DJNZ R4,F4	DJNZ R4,AG2
6327	00		NOP	NO OPERATION
6328	EA		MOV A,R2	LOAD COLUMN ADDRESS
6329	F0		MOVX @DPTR,A	OUT ADDRESS ON PORT A
632A	11 51		ACALL 6051	CALL COMMAND
632C	74 FF		MOV A,#FF	LOAD DATA
632E	F0		MOVX @DPTR,A	OUT DATA ON PORT A
632F	11 40		ACALL 6040	CALL DATAWR
6331	22		RET	RETURN TO MAIN

SUBROUTINE 8 TO PRINT "EXCEL TECHNOLOGIES"

ADDRESS	OP CODE	LABEL	MNEMONICS	COMMENTS
6400	90 28 0D	EXCEL:	MOV DPTR,#280D	PORT B ADDRESS
6403	74 BB		MOV A,#BB	PAGE 3 ADDRESS
6405	F0		MOVX @DPTR,A	OUT PAGE ADDRESS
6406	11 51		ACALL 6051	CALL COMMAND
6408	78 65		MOV R0,#65	LOAD HIGHER 8 BIT OF MEMORY ADDRESS
640A	79 00		MOV R1,#00	LOAD LOWER 8 BIT OF MEMORY ADDRESS
640C	7C 40		MOV R4,#40	COLUMN COUNT
640E	7D 40		MOV R5,#40	COLUMN ADDRESS
6410	ED	LP1:	MOV A,R5	LOAD COLUMN ADDRESS
6411	F0		MOVX @DPTR,A	OUT ADDRESS ON PORT A
6412	11 51		ACALL 6051	CALL COMMAND
6414	88 83		MOV 83,R0	MOV DPH,R0
6416	89 82		MOV 82,R1	MOV DPL,R1
6418	E0		MOVX A,@DPTR	LOAD DATA FROM MEMORY
6419	90 28 0D		MOV DPTR,#280D	LOAD PORT B ADDRESS
641C	F0		MOVX @DPTR,A	OUT DATA FROM MEMORY TO PROT A USING A
641D	11 62		ACALL 6062	ACALL DATAWR1
641F	09		INC R1	INCREMENT MEMORY ADDRESS

6420	0D		INC R5	INCREMENT COLUMN ADDRESS
6421	DC ED		DJNZ R4,ED	DJNZ R4,LP1
6423	00		NOP	NO OPERATION
6424	7C 40		MOV R4,#40	COLUMN COUNT
6426	7D 40		MOV R5,#40	COLUMN ADDRESS
6428	ED	LP2:	MOV A,R5	LOAD COLUMN ADDRESS
6429	F0		MOVX @DPTR,A	OUT ADDRESS ON PORT A
642A	11 51		ACALL 6051	CALL COMMAND
642C	88 83		MOV 83,R0	MOV DPH,R0
642E	8982		MOV 82,R1	MOV DPL,R1
6430	E0		MOVX A,@DPTR	LOAD DATA FROM MEMORY
6431	90 28 0D		MOV DPTR,#280D	LOAD PORT B ADDRESS
6434	F0		MOVX @DPTR,A	OUT DATA FROM MEMORY TO PROT B USING A
6435	11 85		ACALL 6085	ACALL DATAWR2
6437	09		INC R1	INCREMENT MEMORY ADDRESS
6438	0D		INC R5	INCREMENT COLUMN ADDRESS
6439	DC ED		DJNZ R4,ED	DJNZ R4,LP2
643A	00		NOP	NO OPERATION
643C	22		RET	RETURN TO MAIN PROGRAM

DATA ON MEMORY LOCATION 6500 TO 6580

CHARECTER	**ADDRESS**						
SPACE/BLANK	6500	00	00	00	00	00	00
SPACE/BLANK	6506	00	00	00	00	00	
E	650B	FE	92	92	92	82	00
X	6511	C6	28	10	28	C6	00
C	6517	7C	82	82	82	44	00
E	651D	FE	92	92	92	82	00
L	6523	FE	80	80	80	80	00
SPACE/BLANK	6529	00	00	00	00	00	
T	652E	02	02	FE	02	02	00
E	6534	FE	92	92	92	82	00
C	653A	7C	82	82	82	44	00
H	6540	FE	10	10	10	FE	00
N	6546	FE	08	10	20	FE	00
O	654C	7C	82	82	82	7C	00
L	6552	FE	80	80	80	80	00
O	6558	7C	82	82	82	7C	00

G	655E	7C	82	92	92	F4	00
I	6564	00	82	FE	82	00	00
E	656A	FE	92	92	92	82	00
S	6570	8C	92	92	92	62	00
SPACE/BLANK	6576	00	00	00	00	00	00
SPACE/BLANK	657C	00	00	00	00	00	00

NOTE: We have used 7 row and 5 columns to create a character.

APLICATION-4

TRAFFIC LIGHT CONTROLLER

The fourth application provided on the Board of the Kit is Traffic Light controller. The description of the application is given here.

CIRCUIT DESCRIPTION:

The hardware design for this module is very simple as only the Buffers are used to drive an LED's through the I/O lines of 8255. The LED to be ON is to be fed with logic 1 through 8255 port. The light arrangement in the crossing are used as given below:

The traffic light controller simulates the operation of traffic light in a busy crossing. The Red, Yellow and Green colour LED's are used for indication of stop, Get Ready and Go signal. The card has been designed in such a way that students can write different programs to simulate different patterns of traffic control followed in different cities. The duration of the traffic movement is controlled by delay routines which are called by the main program.

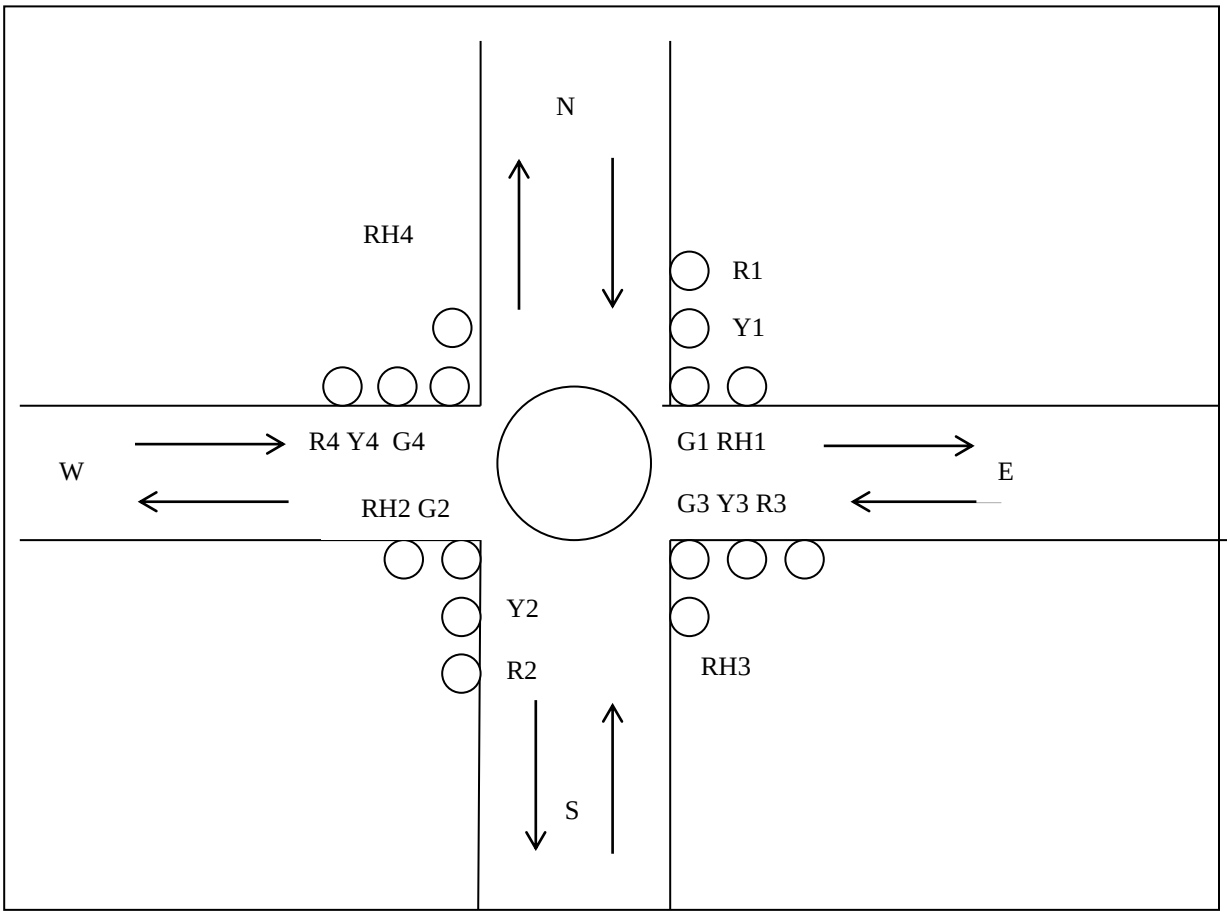
The Port A and Port B of 8255 are used as given below:

PA0	PA1	PA2	PA3	PA4	PA5	PA6	PA7
For R1	For G2	For Y1	For Y2	For G1	For R2	For RH1	For RH2

PB0	PB1	PB2	PB3	PB4	PB5	PB6	PB7
For R3	For R4	For Y3	For Y4	For G3	For G4	For RH3	For RH4

EXERCISES:

1. **Write a program to control the traffic using following scheme of traffic movement as per the priority given in a, b, c, etc.**



a) North to South and South to North Traffic
b) South to North and South to East Traffic
c) North to South and North to West Traffic
d) East to West and West to East Traffic
e) East to North and East to West Traffic
f) West to South and West to East Traffic
g) Repeat from (a)

2. **Writ a program to control the traffic using following scheme of traffic movement as per the priority given in a, b, c, etc.**

a) North to South and North to West Traffic
b) South to North and South to East Traffic

c) North to South and South to North Traffic
d) East to West and West to East Traffic
e) East to West and East to North Traffic
f) West to East and West to South Traffic
g) Repeat from (a)

3. Write a program to control the traffic using following scheme of traffic movement as per the priority given in a, b, c, etc.

a) South to North and South to East Traffic
b) Flashing Warning on RH1 for South to East Traffic before Stopping South to East Traffic
c) North to South and South to North Traffic
d) North to South and North to West Traffic
e) East to West and East to North Traffic
f) Flash Warning on RH3 for East to North Traffic before stopping
g) East to West and West to East Traffic
h) West to East and West to South Traffic
i) Repeat from (a)

NOTE; The delay for duration of the ;movement and stop of traffic as well as the duration of flash etc. or duration of Yellow between Red and Green can be selected suitably by changing the value of counters used in delay routines.

The user can imagine new ways of controlling Traffic and give the students such exercise to perform.

EXERCISE:-1

The exercise number one is solved here for the students to understand the way the program can be written.

SEQUENCE OF SWITCHING:-

The sequence of switching i.e. outputting of data for example-1 is given here. The students once understands this, can then write programs for other exercises.

Sl. No.	LED's to be Switched ON	Binary Data to be outputted	Word /Data to be outputted
1	PortA-R2, R1 Port B- G4, G3	0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0	21 30
2	PortA-R2, R1 Port B-Y4, Y3	0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0	21 0C
3	Port A-R2, R1	0 0 1 0 0 0 0 1	21

	Port B- RH4, RH3, R4, R3	1 1 0 0 0 0 1 1	C3
4	Port A- Y2, Y1 Port B-Y4, Y3	0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0	0C 0C
5	Port A-G1, G2 Port B-R4, R3	0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1	12 03
6	Port A- Y2, Y1 Port B- R4, R3	0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1	0C 03
7	Port A-RH2, RH1, R2, R1 Port B- R4, R3	1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 1	E1 03
8	Port A- Y2, Y1	0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0	0C 0C

LISTING OF THE PROGRAM FOR TRAFFIC LIGHT CONTROLLER AAPLICATION ON THE BOARD OF ET-8051LCD-APL

Select the DIP Switch SW1-3 & Sw6-8 in OFF position and Keep SW3-4 in ON position.

Enter the program given below and execute from Address 6000.

Address	Code	Label	Mnemonics	Remark
6000	74 80	START:	MOVA, #80H	All ports as O/P
6002	90 28 0F		MOV DPTR, #280FH	
6005	F0		MOVX @DPTR, A	
6006	79 21	CONT:	MOV R1 #21H	
6008	7A 30		MOV R2, #30H	
600A	11 44		ACALL OUT	
600C	11 4D		ACALL DELAY.1	Delay for 40 Sec.
600E	7A 0C		MOV R2, #0CH	
6010	11 44		ACALL OUT	
6012	11 54		ACALL DELAY.2	Delay for 2 Sec.
6014	7A C3		MOV R2, #C3H	
6016	11 44		ACALL OUT	
6018	11 5B		ACALL DELAY.3	Delay for 10 Sec.
601A	79 0C		MOV R1, #0CH	
601C	7A 0C		MOV R2, #0CH	
601E	11 44		ACALL OUT	
6020	11 54		ACALL DELAY.2	Delay for 2 Sec.
6022	79 12		MOV R1, #12H	
6024	7A 03		MOV R2, #03H	
6026	11 44		ACALL OUT	
6028	11 4D		ACALL DELAY.1	Delay for 40 Sec.
602A	74 0C		MOV A, #0CH	

602C	90 28 0C		MOV DPTR, #280CH	
602F	F0		MOVX @DPTR, A	
6030	11 54		ACALL DELAY.2	Delay for 2 Sec.
6032	79 E1		MOV R1, #E1H	
6034	7A 03		MOV R2, #03H	
6036	11 44		ACALL OUT	
6038	11 5B		ACALL DELAY.3	Delay for 10 Sec.
603A	79 0C		MOV R1, #0CH	
603C	7A 0C		MOV R2, #0CH	
603E	11 44		ACALL OUT	
6040	11 54		ACALL DELAY.2	
6042	80 C2		SJMP CONT	
6044	90 28 0C	OUT:	MOV DPTR, #280CH	
6047	E9		MOV A, R1	
6048	F0		MOVX @ DPTR, A	Data out at port A
6049	A3		INC DPTR	
604A	EA		MOV A, R2	
604B	F0		MOVX @DPTR, A	Data out at port B
604C	22		RET	
604D	7B 10	DELAY1	MOV R3, #40H	Delay for 10 Sec.
604F	11 62	KNT1	ACALL DELAY	
6051	DB FC		DJNZ R3, KNT.1	
6053	22		RET	
6054	7B 03	DELAY2	MOV R3, #03H	Delay for 2 Sec.
6056	11 62	KNT2	ACALL DELAY	
6058	DB FC		DJNZ R3, KNT.2	
605A	22		RET	
605B	7B 10	DELAY3	MOV R3, #10H	Delay for 10 Sec.
605D	11 62	KNT3	ACALL DELAY	
605F	DB FC		DJNZ R3, KNT.3	
6061	22		RET	

6062	7C 04	DELAY:	MOV R4, #04H	Delay for 0.68 sec.
6064	7D FF	REP.2:	MOV R5, #FFH	
6066	7E FF	REP.1:	MOV R6, #FFH	
6068	DE FE		DJNZ R6, $	
606A	DD FA		DJNZ RR, REP.1	
606C	DC F6		DJNZ R4, REP.2	
606E	22		RET	

NOTE: - Students can observe Different Traffic Pattern by Writing Their Own Program and See the Desired Traffic Pattern.

ANNEXTURE-1

LIQUID CRYSTAL DISPLAY

ET-8051LCD-APL provides 16 x 2 Liquid Crystal Displays. LCD stands for **LIQUID CRYSTAL DISPLAY** and is a commonly used o/p device for the microprocessor based systems. AS compared to the other commonly used o/p device i.e. Light Emitting Diode (LED) displays, they consume very less power and they don't emit their own light but use ambient light.

The LCD Display used here is CTS-1602 (16 characters x 2 lines) 1/16 duty, 1/5 bias. It is a 16 character x 2 line display, with back light. A Character is displayed using 5x7 dots format. It is compatible with both 4 and 8 bit microprocessor.

The LCD display has 14 pins. The various signals on them can be classified into 3 groups, namely Power signals, Control signals and Data signals.

Signals constituting the power signals are:
1) VSS : Ground
2) VDD : Supply voltage for logic and LCD. It should be 5V +/-5%
3) V0 : Supply voltage for LCD. By varying this, the contrast can be varied.

Signals forming the control group are:]

4) RS : It stands for Register selection. Depending upon its status, the contents of the data bus are treated as an instruction code or data. A High on this pin means that the contents of the data bus are to be treated as the data while a Low on this means that their contents are to be treated as an instruction code.

5) R/W : Status of this pin indicates whether the MPU will read or write data, from or to, the LCD. A High on this pin will initiate a read cycle, while a Low on it will initiate write cycle.

6) E : This is the enable signal. While performing either read or write operation, a High and then High to Low transition has to be provided on this pin.

7-14. DB0-DB7: These can be connected to 8 I/O port lines. DB0 is the LSB & DB7 is the MSB.

Dot matrix LCD controller used in this LCD display is KS 0066. It automatically initializes (reset), when the power is turn on, using the internal reset ckt. However when the power supply conditions are not met, MPU will reset the LCD display by

issuing a sequence of instructions shown under the heading "Initialization by instructions". It has internal memory called Display Data RAM (80 x 8 bits).

HOW TO DISPLAY A STRING OF CHARACTERS

A character can be display by writing its ASCII code into the DISPLAY DATA RAM (DDRAM), which can hold a maximum of 80 character codes. Ram is divided in two groups. Starting address of the 1st group is (refer to table2) 80H while that of 2nd group is C0.

Also the DDRAM is more of a sequential memory than random memory. It is so since, memory locations can be accessed serially only. For e.g. suppose we want to write something at 87H which belongs to 1st group of DDRAM. To get there we have to initialize address counter of DDRAM at 80H (start of 1st group of DDRAM), perform 7 dummy write operations and then write what we intended to write at 87H, which is presently pointed by the address counter for the DDRAM. Same is true for the 2nd group of memory, whose starting address is C0.

INITIALIZING BY INSTRUCTION

If the power supply conditions for correctly operating the internal reset circuit are not met, initialization by instruction is required.

Use the following STEPS for initialization:

TABLE – 1

Power ON	When interface is 8bits long.
Wait for more than 15 m.s. After VDD rises to 4.5V	
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 1 1	Function set (interface is 8 bits long)
Wait for more than 4.1 m.s.	
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 1 1	Function set (interface is 8 bit long)
Wait for more than 100μs	
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 1 1	Function set (interface is 8 bits long)
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0	Function set (interface is 8 bits long. Specify the number of display lines and

0 0 0 0 1 1 N F	character font) The number of display lines and character font can not be changed afterwards.
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 0 0 1 0 0 0	Display OFF
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 0 0 0 0 0 1	Display ON
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 0 0 0 1 I/D S	Entry Mode Set

INITIALIZATION ENDS

1) Display Clear
2) **Function Set**
   DL = 1: 8 bit interface data
   DL = 0: 4 bit
   F = 0 : 5 x 7 dot character font
   N = 1:1/16 Duty
   N = 0 : 1/8 Duty, 1/11 Duty

3) **Display ON/OFF Control**
   D = 0 : Display OFF
   C = 0 : Cursor OFF
   B = 0 : Blink OFF

3) **Entry Mode Set**
   I/D = 1 : +1 (increment)
   S = 0 : No shift

INSTRUCTIONS

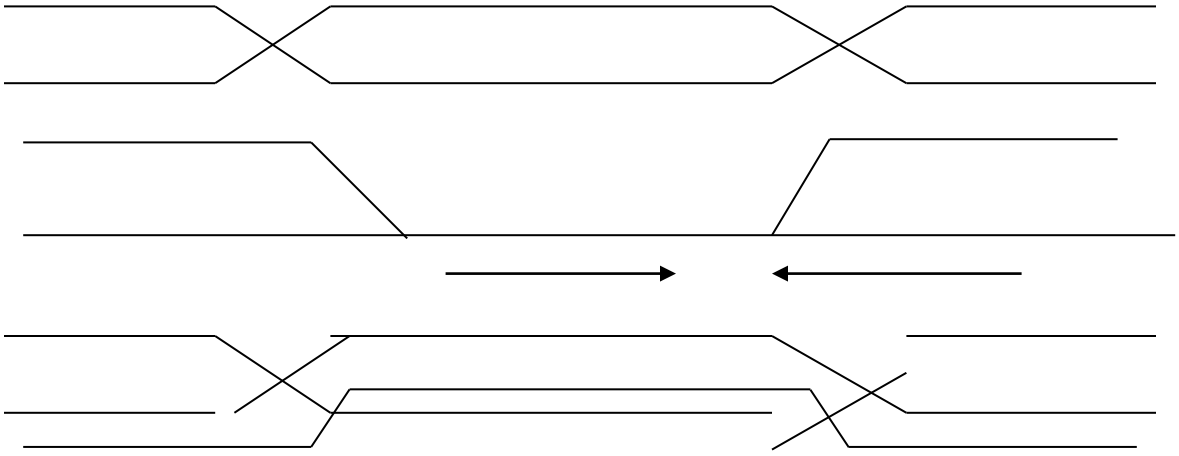
TABLE – 2

INSTRUCTION	CODE RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0	DESCRIPTION
1. Clear Display	0 0 0 0 0 0 0 0 0 1	Clears all display and returns the cursor to the home position (Address 0).

2. Entry Mode Set	0 0 0 0 0 0 0 1 I/D S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write & read.
3. Display On/Off Control	0 0 0 0 0 0 1 D C B	Sets ON/OFF all display (D) cursor ON/OFF (c), and blink of cursor position character (B).
4.Cursor/Display Shift	0 0 0 0 0 1 S/C R/L . .	Moves the cursor and shifts the display without changing DDRAM contents.
5. Function Set	0 0 0 0 1 DL N F .	Sets interface data length (DL) number of display lines (L) and character font (F).
6. DDRAM	0 0 1 ADD	Sets the DDRAM address. DDRAM data is sent address set and received after this setting.
7. DDRAM Data Write	1 0 WRITE DATA	Writes data into DDRAM or CGRAM.
8. Cursor At Home Control	0 0 0 0 0 0 0 0 1 .	Returns the cursor to the home position (Address 0)

CODE	
I/D = 1	Increment
I/D = 0	Decrement
S = 1	With display shift
ADD	DDRAM address corresponds to cursor address

WRITE CYCLE



The address of all LCD segments are defined in the above LCD block display diagram.

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C	CB	CC	CD	CE	CF	D0	D1	D2	D3

ANNEXURE – 2

HEX CODE	NUMBER OF BYTES	MNEMONICS	OPERANDS	SYNTEX FORMAT
00	1	NOP		NOP
01	2	AJMP	CODE ADDR	AJMP b XXXX
02	3	LJAMP	CODE ADDR	LJMP b XXXX
03	1	RR	A	RR b A
04	1	INC	A	INCA
05	2	INC	DATA ADDR	INC b YY
06	1	INC	@R0	INC b @ R0
07	1	INC	@R1	INC b @ R1
08	1	INC	R0	INC b R0
09	1	INC	R1	INC b R1
0A	1	INC	R2	INC b R2
0B	1	INC	R3	INC b R3
0C	1	INC	R4	INC b R4
0D	1	INC	R5	INC b R5
0E	1	INC	R6	INC b R6
0F	1	INC	R7	INC b R7
10	3	JBC	BIT ADDR, CODE ADDR	JBC b YY; XX
11	2	ACALL	CODE ADDR	ACALL b XXXX
12	3	LCALL	CODE ADDR	LCALL b XXXX
13	1	RRC	A	RRC b A

14	1	DEC	A	DECA
15	2	DEC	DATA ADDR	DEC b YY
16	1	DEC	@R0	DEC b @ R0
17	1	DEC	@R1	DEC b @ R1
18	1	DEC	R0	DEC b R0
19	1	DEC	R1	DEC b R1
1A	1	DEC	R2	DEC b R2
1B	1	DEC	R3	DEC b R3
1C	1	DEC	R4	DEC b R4
1D	1	DEC	R5	DEC b R5
1E	1	DEC	R6	DEC b R6
1F	1	DEC	R7	DEC b R7
20	3	JB	BITADDR, CODE ADR	JB b YY,XX
21	2	AJMP	CODE ADDR	AJMP b XXXX
22	1	RET		RET
23	1	RL	A	RL b A
24	2	ADD	A, #DATA	ADD b A, #YY
25	2	ADD	A, DATA ADDR	ADD b A,YY
26	1	ADD	A, @R0	ADD b A, @ R0
27	1	ADD	A, @R1	ADD b A, @ R1
28	1	ADD	A, R0	ADD b A,R0

29	1	ADD	A, R1	ADD b A,R1
2A	1	ADD	A, R2	ADD b A,R2
2B	1	ADD	A, R3	ADD b A,R3
2C	1	ADD	A, R4	ADD b A,R4
2D	1	ADD	A, R5	ADD b A,R5
2E	1	ADD	A, R6	ADD b A,R6
2F	1	ADD	A, R7	ADD b A,R7
30	3	JNB	BIT ADDR, CODEADDR	JNB b ZZ,XX
31	2	ACALL	CODE ADDR	ACALL b XXXX
32	1	RETI		RETI
33	1	RLC	A	RLC b A
34	2	ADDC	A, #DATA	ADDC b A,#YY
35	2	ADDC	A, DATA ADDR	ADDC b A, YY
36	1	ADDC	ADDC A, @R0	ADDC b A, @ R0
37	1	ADDC	A, @R1	ADDC b A, @ R1
38	1	ADDC	A, R0	ADDC b A, R0
39	1.	ADDC	A, R1	ADDC b A, R1
3A	1	ADDC	A, R2	ADDC b A, R2
3B	1	ADDC	A, R3	ADDC b A, R3
3C	1	ADDC	A, R4	ADDC b A, R4
3D	1	ADDC	A, R5	ADDC b A, R5
3E	1	ADDC	A, R6	ADDC b A, R6

3F	1	ADDC	A, R7	ADDC b A, R7
40	2	JC	CODE ADDR	JC b XX
41	2	AJMP	CODE ADDR	AJMP b XXXX
42	2	ORL	DATA ADDR, A	ORL b YY, A
43	3	ORL	DATA ADDR, #DATA	ORL b YY,#ZZ
44	2	ORL	A, #DATA	ORLA, # YY
45	2	ORL	A, DATA ADDR	ORLA, YY
46	1	ORL	A, @R0	ORLA, @R0
47	1	ORL	A, @R1	ORLA, @R1
48	1	RL	A, R0	ORLA, R0
49	1	ORL	A, R1	ORLA, R1
4A	1	ORL	A, R2	ORLA, R2
4B	1	ORL	A, R3	ORLA, R3
4C	1	ORL	A, R4	ORLA, R4
4D	1	ORL	A, R5	ORLA, R5
4E	1	ORL	A, R6	ORLA, R6
4F	1	ORL	A, R7	ORLA, R7
50	2	JNC	CODE ADDR	JNC b XX
51	2	ACALL	CODE ADDR	ACALL b XXXX
52	2	ANL	DATA ADDR, A	ANL b YY, A
53	3	ANL	DATAADDR, #DATA	ANL b YY,#ZZ

54	2	ANL	A, #DATA	ANLA, #YY
55	2	ANL	A, DATA ADDR	ANLA, YY
56	1	ANL	A, @R0	ANLA, @R0
57	1	ANL	A, @R1	ANLA, @R1
58	1	**ANL**	A, R0	ANLA, R0
59	1	ANL	A, R1	ANLA, R1
5A	1	ANL	A, R2	ANLA, R2
5B	1	ANL	A, R3	ANLA, R3
5C	1	ANL	A, R4	ANLA, R4
5D	1	ANL	A, R5	ANLA, R5
5E	1	ANL	A, R6	ANLA, R6
5F	1	ANL	A, R7	ANLA, R7
60	2	JZ	CODE ADDR	JZ b XX
61	2	AJMP	CODEADDR	AJMP b XXXX
62	2	XRL	DATA ADDR, A	XRL b YY,A
63	3	XRL	DATA ADDR, #DATA	XRL b YY,#ZZ
64	2	XRL	A, #DATA	XRLA,#YY
65	2	XRL	A, DATA ADDR	XRLA, YY
66	1	XRL	A, @R0	XRLA @ R0
67	1	XRL	A, @R1	XRLA @ R1
68	1	XRL	A,R0	XRLA, R0
69	1	XRL	A,R1	XRLA, R1

6A	1	XRL	A,R2	XRLA, R2
6B	1	XRL	A,R3	XRLA, R3
6C	1	XRL	A,R4	XRLA, R4
6D	1	XRL	A,R5	XRLA, R5
6E	1	XRL	A,R6	XRLA, R6
6F	1	XRL	A,R7	XRLA, R7
70	2	JNZ	CODE ADDR	JNZ b XX
71	2	ACALL	CODE ADDR	ACALL b XXXX
72	2	ORL	C, BIT ADDR	ORL b C, ZZ
73	1	JMP	@A+DPTR	JMP b @ A+DPTR
74	2	MOV	A, #DATA	MOVA, # YY
75	3	MOV	DATA ADDR, #DATA	MOV b YY, #ZZ
76	2	MOV	@R0, #DATA	MOV b @ R0, # YY
77	2	MOV	@R1, #DATA	MOV b @ R1, # YY
78	2	MOV	R0, #DATA	MOV b R0,#YY
79	2	MOV	R1, #DATA	MOV b R1,#YY
7A	2	MOV	R2, #DATA	MOV b R2, #YY
7B	2	MOV	R3, #DATA	MOV b R3, #YY
7C	2	MOV	R4, #DATA	MOV b R4, #YY
7D	2	MOV	R5, #DATA	MOV b R5, #YY
7E	2	MOV	R6, #DATA	MOV b R6, #YY

7F	2	MOV	R7, #DATA	MOV b R7, #YY
80	2	SJMP	CODE ADDR	SJMP b XX
81	2	AJMP	CODE ADDR	AJMP b XXXX
82	2	ANL	C, BIT ADDR	ANL b C, ZZ
83	1	MOVC	A, @A + PC	MOVC b A,@ A+PC
84	1	DIV	AB	DIV b AB
85	3	MOV	DATA ADDR, DATA ADDR	MOV b XX,YY
86	2	MOV	DATA ADDR, @R0	MOV b XX,@ R0
87	2	MOV	DATA ADDR, @R1	MOV b XX,@ R1
88	2	MOV	DATA ADDR, R0	MOV b YY,R0
89	2	MOV	DATA ADDR, R1	MOV b YY,R1
8A	2	MOV	DATA ADDR, R2	MOV b YY,R2
8B	2	MOV	DATA ADDR, R3	MOV b YY,R3
8C	2	MOV	DATA ADDR, R4	MOV b YY,R4
8D	2	MOV	DATA ADDR, R5	MOV b YY,R5
8E	2	MOV	DATA ADDR, R6	MOV b YY,R6
8F	2	MOV	DATA ADDR, R7	MOV b YY,R7
90	2	MOV	DPTR, #DATA	MOV b DPTR, #YYYY
91	2	ACALL	CODE ADR	ACALL b XXXX
92	2	MOV	BIT ADDR, C	MOV b ZZ, C
93	1	MOVC	A, @A+ DPTR	MOVC b A, @ A+DPTR
94	2	SUBB	A, #DATA	SUBB b A,#YY

95	2	SUBB	A, DATA ADDR	SUBB b A, YY
96	1	SUBB	A, @R0	SUBB b A,@R0
97	1	SUBB	A, @R1	SUBB b A,@R1
98	1	SUBB	A, R0	SUBB b A, R0
99	1	SUBB	A, R1	SUBB b A, R1
9A	1	SUBB	A, R2	SUBB b A, R2
9B	1	SUBB	A, R3	SUBB b A, R3
9C	1	SUBB	A, R4	SUBB b A, R4
9D	1	SUBB	A, R5	SUBB b A, R5
9E	1	SUBB	A, R6	SUBB b A, R6
9F	1	SUBB	A, R7	SUBB b A, R7
A0	2	ORL	C,/BIT ADDR	ORL b C, ZZ
A1	2	AJMP	CODE ADR	AJMP b XXXX
A2	2	MOV	C, BIT ADDR	MOV b C,ZZ
A3	2	INC	DPTR	INC b DPTR
A4	2	MUL	AB	MUL b AB
A5	R RESERVED			
A6	2	MOV	@R0, DATA ADDR	MOV b @ R0, YY
A7	2	MOV	@R1, DATA ADDR	MOV b @ R1, YY
A8	2	MOV	R0, DATA ADDR	MOV b R0,YY
A9	2	MOV	R1, DATA ADDR	MOV b R1,YY

AA	2	MOV	**R2, DATA ADDR**	MOV b R2,YY
AB	2	**MOV**	R3, DATA ADDR	MOV b R3,YY
AC	2	**MOV**	R4, DATA ADDR	MOV b R4,YY
AD	2	MOV	R5, DATA ADDR	MOV b R5,YY
AE	2	MOV	R6, DATA ADDR	MOV b R6,YY
AF	2	MOV	R7, DATA ADDR	MOV b R7,YY
B0	2	ANL	C, BIT ADDR	ANL b C, ZZ
B1	2	ACALL	CODE ADDR	ACALL b XXXX
B2	2	CPL	BIT ADDR	CPL b ZZ
B3	1	CPL	C	CPLC
B4	3	CJNE	A, #DATA, CODE ADDR	CJNE b A, #YY, ZZ
B5	3	CJNE	A, DATA ADDR, CODE ADDR	CJNE b A, YY,XX
B6	3	CJNE	@R0, #DATA, CODE	CJNE b @R0, # YY, ZZ
B7	3	CJNE	@R1, #DATA, CODE	CJNE b @ R1, # YY, ZZ
B8	3	CJNE	R0, #DATA, CODE ADDR	CJNE b R0, # YY, XX
B9	3	CJNE	R1, #DATA, CODE ADDR	CJNE b R1, # YY, XX
BA	3	CJNE	R2, #DATA, CODE ADDR	CJNE b R2, # YY, XX
BB	3	CJNE	R3, #DT, CODE ADDR	CJNE b R3, # YY, XX
BC	3	CJNE	R4, #DATA, CODE ADDR	CJNE b R4, # YY, XX
BD	3	CJNE	R5, #DATA, CODE ADDR	CJNE b R5, # YY, XX
BE	3	CJNE	R6, #DATA, CODE ADDR	CJNE b R6, # YY, XX
BF	3	CJNE	R7, #DATA,	CJNE b R7, #

			CODE ADDR	YY, XX
C0	2	PUSH	DATA ADDR	PUSH b YY
C1	2	AJMP	CODE ADDR	AJMP b XXXX
C2	2	CLR	BIT ADDR	CLR b ZZ
C3	1	CLR	C	CLRC
C4	1	SWAP	A	SWAP b A
C5	2	XCH	A, DATA ADDR	XCH b A,YY
C6	1	XCH	A, @ R0	XCH b A, @ R0
C7	1	XCH	A, @ R1	XCH b A, @ R1
C8	1	XCH	A, R0	XCH b A,R0
C9	1	XCH	A, R1	XCH b A,R1
CA	1	XCH	A, R2	XCH b A,R2
CB	1	XCH	A, R3	XCH b A,R3
CC	1	XCH	A, R4	XCH b A,R4
CD	1	XCH	A, R5	XCH b A,R5
CE	1	XCH	A, R6	XCH b A,R6
CF	1	XCH	A, R7	XCH b A,R7
D0	2	POP	DATA ADDR	POP b YY
D1	2	ACALL	CODE ADDR	ACALL b XXXX
D2	2	SETB	BIT ADDR	SETB b ZZ
D3	2	SETB	C	SETB b C
D4	2	DA	A	DA b A

D5	3	DJNZ	DATA ADDR, CODE ADDR	DJNZ b YY,XX
D6	1	XCHD	A, @ R0	XCHD b A,@R0
D7	1	XCHD	A, @ R1	XCHD b A,@R1
D8	2	DJNZ	R0, CODE ADDR	DJNZ b R0, XX
D9	2	DJNZ	R1, CODE ADDR	DJNZ b R1, XX
DA	2	DJNZ	R2, CODE ADDR	DJNZ b R2, XX
DB	2	DJNZ	R3, CODE ADDR	DJNZ b R3, XX
DC	2	DJNZ	R4, CODE ADDR	DJNZ b R4, XX
DD	2	DJNZ	R5, CODE ADDR	DJNZ b R5, XX
DE	2	DJNZ	R6, CODE ADDR	DJNZ b R6, XX
DF	2	DJNZ	R7, CODE ADDR	DJNZ b R7, XX
E0	1	MOVX	A, @ DPTR	MOVX b A, @DPTR
E1	2	AJMP	CODE ADDR	AJMP b XXXX
E2	1	MOVX	A, @R0	MOVX b A,@R0
E3	1	MVX	A, @ R1	MOVX b A,@R1
E4	1	CLR	A	CLRA
E5	2	MOV	A, DATA ADDR	MOVA, YY
E6	1	MOV	A, @R0	MOVA@ R0
E7	1	MOV	A, @R1	MOVA@ R1
E8	1	MOV	A, R0	MOVA, R0
E9	1	MOV	A, R1	MOVA, R1
EA	1	MOV	A, R2	MOVA, R2

EB	1	MOV	A, R3	MOVA, R3
EC	1	MOV	A, R4	MOVA, R4
ED	1	MOV	A, R5	MOVA, R5
EE	1	MOV	A, R6	MOVA, R6
EF	1	MOV	A, R7	MOVA, R7
F0	1	MOVX	@ DPTR, A	MOVX b @DPTR,A
F1	2	ACALL	CODE ADDR	ACALL b XXXX
F2	1	MOVX	@ R0, A	MOVX b R0, A
F3	1	MOVX	@ R1, A	MOVX b R1, A
F4	1	CPL	A	CPLA
F5	2	MOV	DATA ADDR, A	MOV b YY,A
F6	1	MOV	@ R0, A	MOV b @ R0, A
F7	1	MOV	@ R1, A	MOV b @ R1, A
F8	1	MOV	R0, A	MOV b R0, A
F9	1	MOV	R1, A	MOV b R0, A
FA	1	MOV	R2, A	MOV b R0, A
FB	1	MOV	R3, A	MOV b R0, A
FC	1	MOV	R4, A	MOV b R0, A
FD	1	**MOV**	R5, A	MOV b R0, A
FE	1	MOV	R6, A	MOV b R0, A
FF	1	MOV	R7, A	MOV b R0, A

ANNEXURE – 3

The conventional format of some instructions for the in-built assembler of ET-31LCD has been changed and should be followed: -

S.NO.	INSTRUCTION	S.NO.	INSTRUCTION
1	MOVA #30	10	XRLA, # 33
2	MOVA 30	11	CLRA
3	MOVA R0	12	CPLA
4	MOVA @R0	13	CLRC
5	ANLA #33	14	CPLC
6	ANLA 33	15	DECA
7	ORLA #33	16	INCA
8	ORLA 33	17	SJMP b (Relative address)
9	XRLA #33	18	CJNE b R1, #20, (Relative address)
		19	CJNE b R0, #03, (Relative address)

Where "Relative Address" is address relative to starting address of the program e.g.

```
ADDRESS         PROGRAM

6002            MOV R1,#20
6003            MOV R2,#20
6004            CJNE b  R1#30, 02     ; where 02 is RELATIVE ADDRESS
```

User should use address of Stack Pointer insisted of SP e.g.

```
MOV     SP,#58      ; it should be use as
MOV     81,#58
```

User should change the format while using these instructions in **ASSEMBLY MODE.**

****