

LABORATORY MANUAL

**EMBEDDED SYSTEMS
AND
REAL TIME PROGRAMMING**

**B.E. (Electronics)
(Sem. VIII)**

INDEX

Serial No.	Title	Page No.
1	To Study 8051 Simulator	2
2	To Write a Program for Scanning a Key Pad and Displaying the Pressed Key on a LCD	4
3	To Write a Program to drive a Stepper Motor	5
4	To Write a Program to Drive a DC Motor with Simulation using Proteus	6
5	To Implement Mouse Driver Program using Dos Interrupts.	7
6	To Write a C Program to Interface Real Time Clock IC DS1307 with a Microcontroller Using I ² C Bus Protocol	9
7	To Write a C Program to Generate PWM Using ATMEL AVR ATMEGA-8535	11
8	To study testing tools with KEIL Compiler	13

Laboratory Session 1

Title: To write assembly language programs for the 8051 using the simulator S51

- i. Block transfer
- ii. Toggling of port pin
- iii. To transfer a character serially
- iv. To generate a square wave on the port pin
- v. To drive a LED connected on the port pin (using the external interrupt pin)

Pre-Requisites:

1. Instructions of Microcontroller 8051.
2. Addressing modes of Microcontroller 8051.
3. Flag register of Microcontroller 8051
4. Knowledge of S51 simulator.

Theory:

The 8051 Microcontroller has 128 bytes of RAM , 4KB of on –chip ROM, two timers, one serial port and 4 parallel ports (32 I/O lines) all on a single chip. The 8051 is a 8 bit microcontroller with 6 interrupt sources. There are two timers in 8051, depending on the value of M1 and M0 bits in the TMOD register. The timers can operate in 4 modes

Mode 0: 13 bit timer, 8 bit timer/counter THX with TLX as 5 bit pre-scaler

Mode 1: 16 bit timer, THX and TLX are cascaded, there is no pre-scaler

Mode 2: 8 bit auto reload, THX holds a value which is to be reloaded into TLX each time it overflows

Mode 3: Split time mode

8051 has a serial port and can operate in 4 modes depending on SM0 and SM1 bits in the SCON register

SM0	SM1	Mode	
0	0	0	shift register, baud = f/12
0	1	1	8 bit UART, baud = variable
1	0	2	9 bit UART, baud f/32 or f/64
1	1	3	9 bit UART , baud = variable

Five interrupts are provided in the 8051. Three of these are generated automatically by internal operations, Timer flag 0, Timer flag 1 and serial port (TI and RI). Two interrupts are triggered by external signals provided by circuitry that is connected to pins INTO and INT1

References:

1. The 8051 Microcontroller and Embedded Systems- Muhammad Mazidi
2. The 8051 Microcontroller – Kenneth Ayala

Algorithm: Block Transfer:

1. Start
2. Initialize R0 & R1 with source and destination memory addresses respectively.
3. initialize counter R2 with value 10
4. copy content of memory pointed by R0 in A
5. Copy content of A to memory pointed by R1.
6. Increment R0, R1 by 1.
7. Decrement counter R2 by 1 & if R2 ≠ 0, go to step 3.

8. Stop

Algorithm: Toggling of port pin:

- 1.Start
- 2.Set port 1.3
- 3.Initialize timer 0 in mode 1 using TMOD.
- 4.load timer 0 with FF + 1
- 5.run timer 0
- 6.Check for timer overflow.
- 7.On overflow complement P1.3.
- 8.Go to step 2
- 9.Stop.

Algorithm: To Transfer a character serially.

- 1.Start
- 2.Initialize R1 with counter.
- 3.Initialize timer 1 in mode 0 using TMOD.
- 4.Load count in timer 1.
5. Initialize serial port in mode 1 using SCON.
- 6.Transfer data to be send serially to SBUF.
- 7.Decrement counter R1 & if not zero then jump to step 2.
- 8.Stop

Algorithm: To generate a square wave on port pin.

- 1.Start
- 2.Initialize timer 1 in mode 1.
- 3.Load timer zero with FF H
- 4.Complement P 1.5
- 5.Start timer 0
- 6.Wait for timer overflow.
- 7.Create overflow flag.
- 8.Go to step 2.
- 9.Stop.

Algorithm: To drive a LED connected on the port pin (using the external interrupt pin)

- 1.Start
- 2.Enable external interrupt INT 1 make it level triggered using IE and TCON.
- 3.clear port 1.3
- 4.wait for INT 1

Post Lab Assignments: 1. Explain how the priority of interrupts can be changed in the 8051
2. Explain how multiprocessor communication can be used in the 8051

Laboratory Session 2

Title: To Write a Program for Scanning a Key Pad displaying the pressed key on a LCD with the hardware implementation using 89C51RD2 kit.

Pre-Requisites:

1. Knowledge of C or Assembly Language.
2. Address map of microcontroller as well as LCD.
3. Knowledge of Keil Compiler

Theory: Scanning programs for matrix keypads involve generating patterns for the rows and reading the columns, deciphering the key pressed depending on the pattern. Fig shows 4 x 4 matrix keypad. The columns are pulled up to VCC via pull up resistors. The rows are connected to the lower order four bits of the microcontroller while the columns are connected to the higher order port bits. Each row is brought low, when one row at a time and the column pattern is read. For no key press, the column will read all high. If a key pressed, then when corresponding row is brought low the column is pulled low. The pressed key is recognized using its unique row-column pattern. The universal key characteristic is the ability to bounce. The key contact vibrates open and closed for a number of milliseconds when the key is hit and often when it is released. To prevent these rapid pulses from registering multiple presses, key may be debounced with RS flip Flops or de-bounced in software with time delays.

References:

1. The 8051 Microcontroller and Embedded Systems- Muhammad Mazidi
2. The 8051 Microcontroller – Kenneth Ayala
3. Manual for KEIL Compiler

Algorithm:

1. Scan port of microcontroller to which the key pad is interfaced and checks for Key Press. i.e port pin being low.
2. De bounces the switch by incorporating some amount of software delay.
3. Recognize the key pressed from row column pattern read.
4. Call LCD display function to display the pressed key on the LCD.
5. Initialize serial UART port for 8 bit UART transmission and load display data on to SBUF for serial communication.

LCD Display Module:

1. Initialize LCD device for no of links, characters, screen conditions, cursor condition etc.
2. Send display address to command register.
3. Send data to be displayed to data register.

Post laboratory Questions:

1. Explain hardware debouncing of keys.
2. Explain the principle of working of the LCD device.

Laboratory Session 3

Title: To Write a Program to Drive a Stepper Motor

Pre-Requisites:

1. Knowledge of C or Assembly Language.
2. Addressing Details of microcontroller as well as LCD.
3. Knowledge of Keil

Theory : Stepper motors rotate or step from one fixed position to the next in small increments. Common step size range from 0.9° to 30° . Stepper motors are stepped from one position to the next by changing the currents to the field of the motor. The two common field connections are required to as two phases and four phases. Stepper motor control involves interfacing the motor coil connections to the microcontroller port via a driver circuit that provides the necessary drive current for the motor. The microcontroller generates patterns on the port providing the excitation to the field coils which cause it to rotate. The tables show the switching sequence for the typical stepper motor for full step mode and half state mode. In full step mode motor steps through the step angle each time it is excited. In half step mode motor steps through half the mode angle each time. This is accomplished by maintaining one field excitation at a time and changing the other field. Each the microcontroller output one step code it must wait a few milliseconds before the next step code is given out because the motor can only step so fast.

References:

1. Manual for KEIL Compiler
2. Stepper motor card Manuals (XPO –EST)

Algorithm:

1. Scan Microcontroller port pin for key press.
2. Output patterns for port lines interfaced to the stepper motor for forward or reverse directions, depending on key press.
3. Generate milliseconds software delay after each pattern is outputted.

Post Lab Assignments:

1. Study the hardware setup of the motor. Explain why the clamp diodes are used across the driving transistors.
2. Why is a slight jerking motion induced in the motor motion in the full step drive?

Laboratory Session 4

Title: To Write a Program to Drive a DC Motor with simulation using Proteus.

Pre-Requisites:

1. Knowledge of C or Assembly Language.
2. Addressing Details of microcontroller as well as LCD.
3. Knowledge of Keil Compiler
4. Knowledge of PROTEUS

Theory: DC Motor ideally needs a constant voltage to rotate. Thus, to achieve speed control signal conditioning is necessary. To generate a variable dc voltage using Microcontroller either DAC is used or in some advanced application PWM is used. In PWM, control average DC voltage is varied by varying the ratio of ON time to OFF time.

$$V_{DC} \propto \frac{T_{on}}{T_{on} + T_{off}}$$

With a bridge connection, it is also possible to control the direction of motor. As shown in the figure, when T 1 & T4 transistors are ON motor rotates in one direction and T2 & T3 are ON motor rotates in reverse direction. Special care must be taken so as not to turn T1 & T2 or T3 & T4 simultaneously. This care is taken by PAL IC which allows only one set and transistors to turn ON. It also provides speed control in both directions with single variable duty cycle output.

References:

1. Manual for KEIL Compiler
2. Manual for DC Motor card (XPO- EST)

Algorithm:

- 1.To achieve speed control in both directions for each set of transistors, one transistor is given constant voltage while other transistor is given variable pulse width voltage.
2. This on time of variable pulse width voltage is user programmable.
3. Forward and reverse direction is set by key press or no key press.
4. Thus the motor interprets the average voltage and has corresponding speed output.

Post Lab Assignments:

- 1.Explain how speed control of DC motor is achieved using the PTC 8253
- 2.Explain the various modes of operation of the 8253

Laboratory Session 5

Title: To Implement Mouse Driver Program using DOS Interrupts.

Pre-Requisites: 1. Knowledge of various DOS interrupts
2. Knowledge of Assembly Language.

Theory :

Device drivers are software codes which provide interface between the processor and the actual hardware. It does the function of initializing the device, securing it and detaching the device after its use. All operating systems provide built-in functions that can be used to write device drivers. Thus function takes of the address and hardware requirements of the devices. Microsoft Mouse driver provides many functions for using the mouse which can be accessed using INT 33H DOS interrupt. Some of these functions are shown for exa. Pointers getting button status and pointer position etc.

Function 01 H : shows mouse pointer. Displays the mouse pointer and cancels any mouse pointers exclusion and previously designed by INT 33 H.
Function 01 H call with AX = 0001H returns nothing.

Function 03 H: Get mouse position and pointer position. Returns current mouse button status and pointer position.

Call with AX 0003H

Returns: BX= Mouse button status

Bit significant (if set)

0 - left button is down

1 - Right button is down

2 – Centre button is down

3 – 15 reserved (0)

CX - horizontal X co-ordinate

DX - vertical Y co-ordinate

Similar to INT 33 H, Microsoft provides functions for using the video display using INT 21 H.

INT 21H function 02H: character outputs the character to the currently active video display.

Call with AH = 02 H

DL = 8 bit data for output

Returns: Nothing

E.g. Send the character “*” to the standard output device

```
MOV AH, 02
```

```
MOV DL, '*'
```

```
INT 21H
```

Reference: MS-DOS by Ray Duncan

Algorithm:

1. Make mouse pointer visible using INT 33H function 01h.
2. Cut button status and pointer position using INT 33h function 03h.
3. Convert X & Y co-ordinate value into ASCII.
4. Display values of X & Y on the screen using INT 21h function 01h.
5. If button status is zero display “left click” on the screen of button status.
& if it is one display “right click” on the screen.
6. Go to step two.

Post Lab Assignments:

1. Explain the principle of working of the Mouse
2. What is the procedure for loading any device driver in a system

Laboratory Session 6

Title: To Write a C Program to Interface Real Time Clock IC DS1307 with a Microcontroller Using I²C Bus Protocol.

Pre-Requisites:

1. Knowledge of C or Assembly Language.
2. Knowledge of I²C protocol.
3. Knowledge of Keil Compiler
4. Knowledge of Proteus

Reference:

1. Embedded Systems Architecture, Programming and Design –Raj Kamal
2. Manual for RTC DS 1307

Theory :

DS1307 serial real time clock is a low power, full binary coded decimal (BCD) clock/calendar plus it has 56 bytes of NVSRAM. Address and data are transferred via a two – wire bidirectional I²C bus. The clock/calendar provides seconds, minutes, hours, day, date, and month and year information. Each of the above information is stored in dedicated registers and is updated periodically. The end of the month is automatically adjusted for month with lower than 31 days, including leap year corrections. The clock operates in either 24 hours or 12 hours format with AM / PM indicator. It has built in power sense circuit that detects power failure and auto switches to the battery supply. The DS1307 operates as a slave device and its connections to the microcontroller are as shown in the figure. Access is obtained by implementing a START condition and providing a device ID code, following by register address. Subsequent register can be accessed sequentially until a stop condition is executed.

I²C bus (Inter IC Connect)

I²C uses only two bidirectional open – drain lines, serial data (SDA) and serial clock (SCL) pulled up with resistors. The I²C reference design has a seven bit address spaced with 16 reserved addresses so a maximum of 112 nodes can communicate on the same bus.

I²C defines three basic types of message each of which begins with a START and ends with a STOP.

1. Single message where a master writes data to slave.
2. Single message where a master reads data from slave.
3. Combined a message where a master issue at least two reads and / or writes one or more slaves.

Algorithm:

Writing to DS 1307

1. Send I²C start condition.
2. Send DS1307 slave address and write flag (00h)
3. Send address of register to write to or point to one bit at a time.
4. Send data bytes, if any one bit at a time
5. Send I²C Stop condition.

Reading Time data from DS 1307

- 1.Send a write instruction and set register address to 00h
- 2.Send **I²C** start condition
- 3.Send DS 1307 slave address and read flag.(D1h)
- 4.Read 8 **I²C** bytes from the slave device which correspond to DS 1307 register which contain Date/Time.
- 5.Send **I²C** stop condition.

Post Lab Assignments:

1. Explain the I2C protocol in detail and enumerate its application area.
2. Differentiate between the I2C and the CAN bus

Laboratory Session 7

Title: To Write a C Program to Generate PWM Using ATMEL AVR ATMEGA-8535

Pre-Requisites: 1. Knowledge of C or Assembly Language.
2. Knowledge of Keil Compiler
3. Knowledge of Proteus
4. Knowledge of ATMEGA 8535

References: 1. Manual for the ATMEGA kit (LOGSUN LG5AVR)
2. Manual for Keil Compiler

Theory : Logsun's LG5 AVR evaluation board using 8 bit ATMEGA 8535 microcontroller . It has 8KB in system programmable flash memory and 512 bytes on chip SRAM. The development board comes with RS-232, ICSP interface to allow the user to program the microcontroller directly from the PC. LGS-AVR board and related software help to rapidly design and prototype.

The kit provides

1. 2 lines * 16 character LCD
2. Real time clock interface
3. RS -232 serial interface
4. On -Chip 10 bit 8 channel ADC
5. ICSP Interface
6. JTAG interface
7. Two 8 bit timers/counters
8. One 16 bit timer/counter

All the peripherals are implemented as independent modules.

PWM generation using 16 bit timer

Timer of ATMEGA 8535 is a 16 bit timer/ counter . Phase correction PWM mode of timer 1 is used with top value of 3Fh. Duty cycle of PWM is controlled by the value set in OCR1A register which is the output compare register. The PWM output is obtained on pin OC1A in either non-inverting or inverting mode. Clock source is the system clock which is 12 Mhz. Duty cycle for non-inverting mode is given by

$$\text{Duty-cycle} = \frac{\text{OCR1A} * 100}{1023}$$

Algorithm:

1. Initialize timer 1 in phase correct PWM mode with top value -3Fh. Clock source as system clock and OC1A output in non inverting mode . Do not start timer
2. Initialise UART with 9600 baud rate and 8 bit , 1 stop bit and no parity.
3. Input duty cycle from user in percentage and count value is calculated as

$$\text{Count value} = \frac{\text{duty cycle} * 1023}{100}$$

100

4. Write the count to OCR1A register and start timer1.
5. Go to step 3.

Post Lab Assignments:

- 1.Explain the inverting and non-inverting mode for PWM generation in ATMEGA 8535
2. Explain the function of the JTAG interface in the kit used.

Laboratory Session 8

Title: To Study testing tools using KEIL compiler.

Pre-Requisites:

1. Knowledge of C or Assembly Language.
2. Knowledge of Keil Compiler
3. Knowledge of Proteus

References: Manual for the Keil Compiler

Theory: KEIL Integrated Development Environment (IDE) provides tools for testing of software codes during simulation. The tools include code coverage, performance analyzer, logic analyzer , execution profiler ,etc.

Code Coverage

Code Coverage marks code that has been executed. Code coverage helps to ensure you have thoroughly tested your application and helps you to tune your testing strategy.

- i. Lines with no code are marked with a grey checked block.
- ii. Unexecuted lines (instructions) are marked with a grey block
- iii. Fully executed lines (Instructions) are marked with a green block
- iv. Branches that have been taken are marked with a blue block
- v. Branches that have been skipped are marked with an orange block
- vi. The next line to execute is marked with a yellow arrow

One may save and restore coverage statistics for multiple debug sessions using the command line in the output window.

1. Performance Analyzer

The performance analyzer records and displays the execution times for functions and program blocks you select. Bar graphs display the time spent in part of the program. The information provided by the performance analyzer can be used to make subroutines faster by fine tuning the program

2. Logic Analyzer

The Logic analyzer graphically displays signals and program variables as they change over time. Signals recorded by the logic analyzer are easily configured using the setup button in the logic analyzer dialog or by dragging symbols from the symbol window.

Post Laboratory Assignments:

1. Explain the function of the In Circuit Emulator (ICE) in the testing of an embedded system.
2. What is meant by black box testing and White box testing?