Design & Development of a Re-Programmable Embedded Systems : A Practical Approach based on 8051 architecture

(Experimental Manual For

B.Tech & M.Tech Students)

for SoC with support of ITRA

Designed & Developed By: Ms. Nidhi Agarwal

Under the Guidance of: Dr. SRN Reddy, Associate Professor, CSE

Computer Science & Engineering Department

Indira Gandhi Delhi Technical University for Women

Kashmere Gate, Delhi-110006

LIST OF EXPERIMENTS

Exp No.	Description of experiment	Page No.
Exp. 1	Design and develop a reprogrammable embedded computer using 8051 microcontrollers and to show the following aspects. 1. Programming 2. Execution 3. Debugging	4-13
Exp. 2	To Demonstrate the Tool Chain for Keil IDE (Embedded Systems Development Tool Chain) with the example of LED Blinking Program.	14-16
Exp. 3	To demonstrate the procedure for flash programming for reprogrammable embedded system board using NXP's FlashMagic.	17-20
Exp. 4	To demonstrate the procedure of flash programming source code for reprogrammable embedded system board using Sunrom technology's 89SXX USB flash programmer, **ISP Model: 1315.**	21-26
Exp. 5	To demonstrate the procedure and connections for multiple controllers programming of same type of controller with same source code in one go, using flash magic.	27-28
Exp. 6	To interface 8 LEDs at Input-output port and create different patterns.	29-30
Exp. 7	To demonstrate use of general purpose port i.e. Input/ output port of two controllers for data transfer between them.	31-32
Exp. 8	To demonstrate block wise programming of memory for micro controller using flash magic.	33-36
Exp. 9	To demonstrate block wise erasing of memory for micro controller using flash magic.	37-39
Exp. 10	To demonstrate timer working in timer mode and blink LED without using any loop delay routine.	40
Exp. 11	To demonstrate interfacing of seven-segment LED display and generate counting from 0 to 99 with fixed time delay.	41-42

Exp.12	To demonstrate serial communication between PC and controller using serial UART of controller.	43-47
Exp. 13	To demonstrate communication between two controllers using SPI in master mode. Only master will send and Slave will receive.	48-50
Exp. 14	To demonstrate communication between two controllers using SPI in master and slave mode. Master will send, Slave will receive and vice versa.	51-53
Exp. 15	To demonstrate interfacing of 16x2 LCD and print some welcome message on it.	54-57

Experiment 1

Design and development of a Reprogrammable Embedded System (Computer) (RES) using 8051 Microcontroller (MC)

1.1 Objective: Design and develop a reprogrammable embedded system board using 8051 microcontrollers and to show following aspects.

1. Programming
2. Execution
3. Debugging

1.2.1 Software Requirement: Editor for schematic drawing like Eagle.
https://www.cadsoftusa.com/download-eagle/freeware/

Or Protel.
http://protel-pcb.software.informer.com/1.5/
1.2.2 Hardware Requirement: Soldering Iron, Tweezer, Cutter, Multimeter, Components as per table1.1.

1.3 Description:

1.3.1 Embedded System:- Embedded systems are those systems that are similar to computer (they can be termed as computer on a chip) but are designed for some specific task, they may have lesser components (be in size or in count) associated to it, then PC. They may or may not contain all components of a computer system. For more definitions one may refer links below.

http://www.dauniv.ac.in/downloads/EmbsysRevEd_PPTs/Chap01Lesson_1Emsys.pdf
http://en.wikipedia.org/wiki/Embedded_system
Unlike PC, Embedded systems are designed to perform some specific task and generally are not designed for performing multiple tasks.

1.3.2 Block Diagram of an Embedded System:

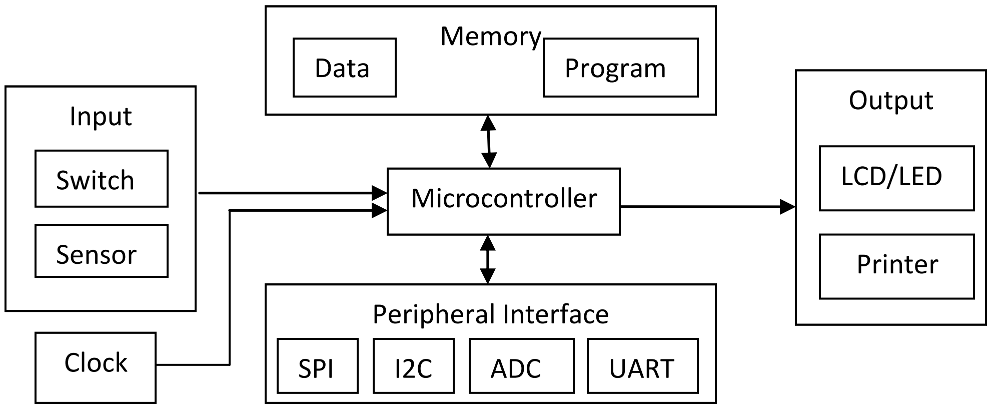


Figure 1.1: Block diagram of Embedded System

1.3.3 “Components of a Computer”

Basic component of an embedded system is its controller which could be a microprocessor unit (MPU) or a microcontroller unit (MCU). MPU needs more peripherals to accomplish a task and hence results in complex circuit and higher power consumption, whereas MCU units mostly have on chip peripherals that includes memory elements like ROM/RAM, basic function elements like Timers/ Counters/ Interrupts and special interfaces like UART/ SPI/ I2C/CAN etc. and thus resulting in lesser component count and lesser power consumption. For more information on difference between MPU and MCU one may visit below mentioned sites.

http://www.atmel.in/Images/MCU_vs_MPU_Article.pdf

http://maxembedded.com/2011/06/05/mcu-vs-mpu/

1.3.4 Re-programmable Embedded System (RES):- In its easiest definition, a re-programmable embedded system is one which can be re programmed a number of times easily while in system or in application and with minimum component requirement i.e. there is no need to pull out MCU every time one wants to program it and hence provide flexibility in programming and operations. It is developed using 8051 compatible microcontrollers manufactured by NXP/ Atmel. In addition to execution of intended program RES also provide debugging facility and chip programming for other users.

The Reprogrammable embedded system consists of:

- Sockets for placing microcontroller- 40 pin
- DC socket for external power supply (DC 5V)
- 1 LED for power on indication and 1 push button for reset
- 11.0592 MHz Quartz Crystal Oscillator
- 8 LEDs for output pin state indication at port P0
- 1 DIP switch (8 switch) for input pin activation
- Connector and driver for serial communication RS232
- Multiple-pin connectors for direct access to I/O ports
- Connector for SPI programming
- 1 Piezo buzzer for audio/frequency output
- Additional power supply connectors

1.4 Selection of component for a given application

Every application circuit is build around some components which should be selected as per the functionality of the application, availability of components, cost of entire system, procurement time for components and most importantly meeting of some critical parameters of intended application.

1.4.1 Selection of Processor:

Selection should be based mainly on architecture, availability, cost, time to prototype and market, testability and debug-ability. As per the requirement a microcontroller will be suitable for this purpose. Intel/Atmel 8051 architecture is suitable for beginners due to its easy understandability, easy

availability of architecture description and instruction set. Some advance controllers of 8051 architecture provide boot-loader, in system programming and in application programming.
NXP's P89V51RD2 and Atmel's AT89S52 are such general purpose controllers, based on 8051 architecture. Re-programmability is achieved using ISP (In-System-Programming) feature provided by NXP P89V51RD2 or by Atmel AT89S52. P89V51RD2 uses ISP by Atmel AT89S52. P89V51RD2 provides ISP feature using UART pins (RxD, TxD, RST, PSEN) while AT89S52 uses SPI pins (MOSI, MISO, SCK, RST) for ISP functionality.

NXP's P89V51RD2 and Atmel's AT89S52 features include;

- 8-bit, 40-pin controller in DIP package
- Operating voltage +5V
- Operating frequency 0 to 40 MHz
- 32-Input / Output pins
- 3-16 bit Timers
- 8- Interrupt levels
- 1-UART
- 1-SPI
- 1 KB of user RAM
- 64 KB of Flash

1.4.2 Selection of other components:

1.4.2.1 Serial communication interface

UART (Universal Asynchronous Receiver Transmitter) is required for boot-loader/ ISP/IAP programming and also for applications that include PC interfacing.
MAX232 is one such chip which provide serial communication interface between personal computer and microcontroller chip. It is selected due to its easily availability and low cost. Operating voltage requirement is +5V.

1.4.2.2 Oscillator

Oscillator is used as a clock signal generator. Crystal oscillators are used for their frequency stability and hence should be chosen over other type of oscillators.
Piezo electric crystal oscillator of 11.0592MHz frequency is used here as this frequency is most suitable for generation of precise baud rate and easy interfacing with PC. Besides, it is also possible to select internal RC oscillator during chip programming/Operations.

1.4.2.3 Connector

DB9 Female PCB Mount:- 3 pins of DB9 connector (pin 2-RD, pin 3-TXD and pin 5-GND) are used for connections between PC and UART IC i.e. MAX232.
Connectors for direct access to Ports
In order to enable microcontroller ports to be directly connected to additional components, each of them is connected to 8 pin, on-board connector.
* Upper Port P1 is also used for providing SPI interface for flash programming.

1.4.2.4 Input Selection

8-DIP switches are provided on board here for interfacing with any of input port. Inputs from sensors/ADC/PC may also be connected through port connectors.

1.4.2.5 Output Selection:

LED: 8-LEDs are connected at port0 with 1KOhm resistor network RN1. They may be used for initial configurations and testing as well as to view outputs.
LCD: 16x2, LCD may be connected using I/O port connectors. They may be used for displaying messages/values. LCD supports ASCII display.
Output at PC/DAC/Motors (through drivers) is also supported.

1.4.2.6 Power Supply

There is a connector on the development board enabling connection to external power supply source (DC-5V). Besides, voltage necessary for device operation can also be obtained from PC via USB cable at connector J7/J8.

1.5 Selection of tools

Some tools and editors are required to prepare assembly language program and its compiling i.e. hex file generation, and writing this hex file to flash memory.
Free downloadable Keil µvision version 4, editor is used for writing assembly language program and its compiling.
Free downloadable Flash Magic or USB programmer is used for flash programming.
Hyper terminal available with windows is used for debugging purpose.

1.6 Schematic Diagram

Discussion and explanation: Refer schematic diagram figure 1.6

1. Microcontroller 89V51RD2 is biased with +5V power supply connected at pin 40, GND connected at pin 20. A 0.1MFD ceramic capacitor is connected between pin 40 and GND to suppress supply spikes.
2. Enable Access (EA), pin 31 and PSEN pin 29 are all connected with Vcc. PSEN bar is connected to high logic as only internal flash memory is in use.
3. Cathode of all 8 LEDs are connected at different pins of port0 i.e. from pin 32 to 39 of controller, LED anode will be connected to Vcc through 1KOhm resistance network RN1. These LEDs will be used in program to view outputs or to check proper functioning by blinking them with different delays.
4. A 16 pin DIP switch (8 on/off switches) can be connected through 10KOhm resistance network RN3 at any port for switch inputs. At on condition port will be at low level.
5. A 11.0592 MHz crystal oscillator is connected between pin 18 and 19 of controller, with two 22pf ceramic capacitors connected between pin 18, 19 and GND.
6. As controller requires logic high voltage for short duration to get itself reset, a reset circuit is connected at RST pin i.e. pin 9 of controller. It consists of a push-to-on switch connected between Vcc and pin 9, a 10K resistor connected between pin 9 and GND and an electrolytic capacitor of 10MFD/25V, connected between Vcc and pin9 of controller.
7. For serial UART working, pin 10 of controller i.e. receive pin at port3 (P3.0) and pin 11 of controller i.e. transmit pin at port3 (P3.1) are connected with serial UART IC, MAX232 pin 9

and 10 respectively. Pin 9 of MAX232 is R2OUT i.e. receive out pin, which outputs data received from PC through serial cable via pin 8 i.e. R2IN of MAX232. Pin 10 of MAX232 is T2IN i.e. transmit input, which inputs data from controller. This input data is then sent to PC through serial cable via pin7 i.e. T2OUT of MAX232.

8. IC MAX232 is biased with +5V supply at pin 16, GND at pin 15. Rest of its biasing is done as per recommended circuitry. Four number 10 MFD/63V electrolytic capacitors are connected as recommended.
9. DB9 connector is connected between MAX232 and PC.

Refer table 1 for complete list of components.

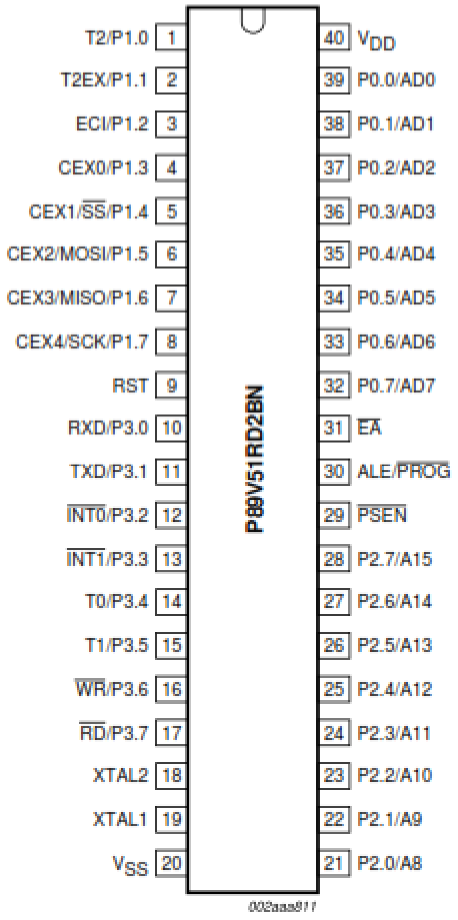


Figure 1.2: Pin diagram of P89V51RD2/AT89S52

*Note: Some instructions or names of SFRs may be changed in different processors of different manufacturers, e.g. ATMEL NXP for same architecture. Care must be taken here.

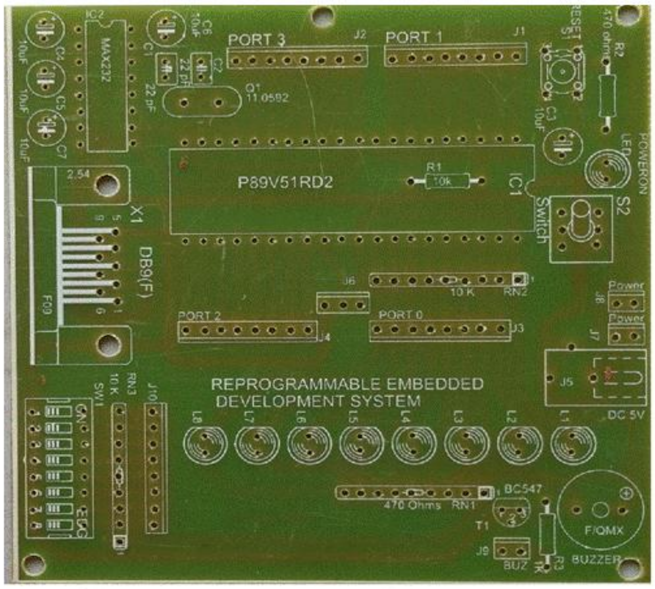


Figure 1.3: Bareboard PCB for Reprogrammable Embedded System

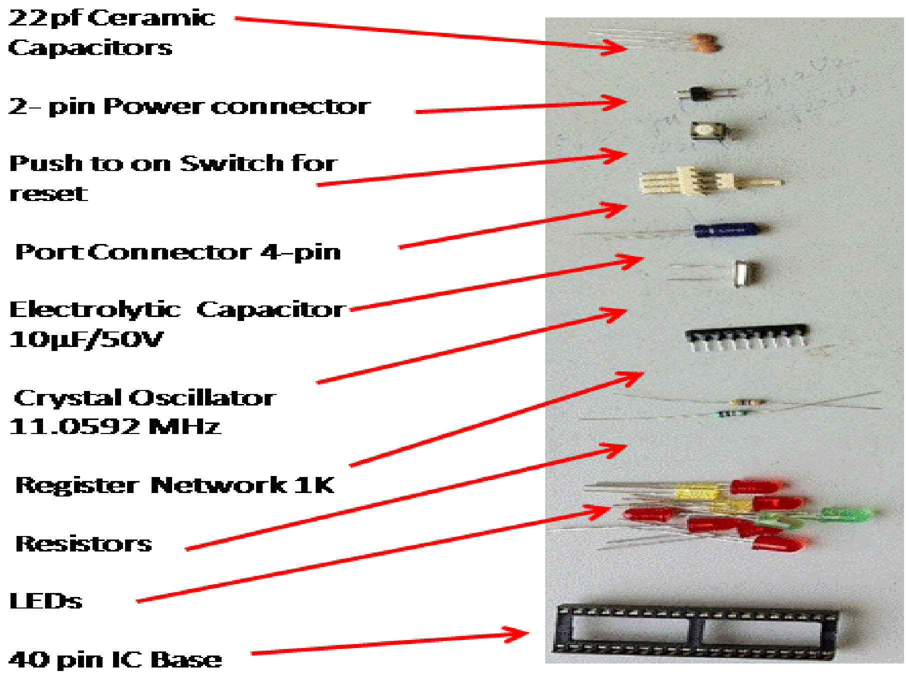


Figure 1.4: Components for Reprogrammable Embedded System

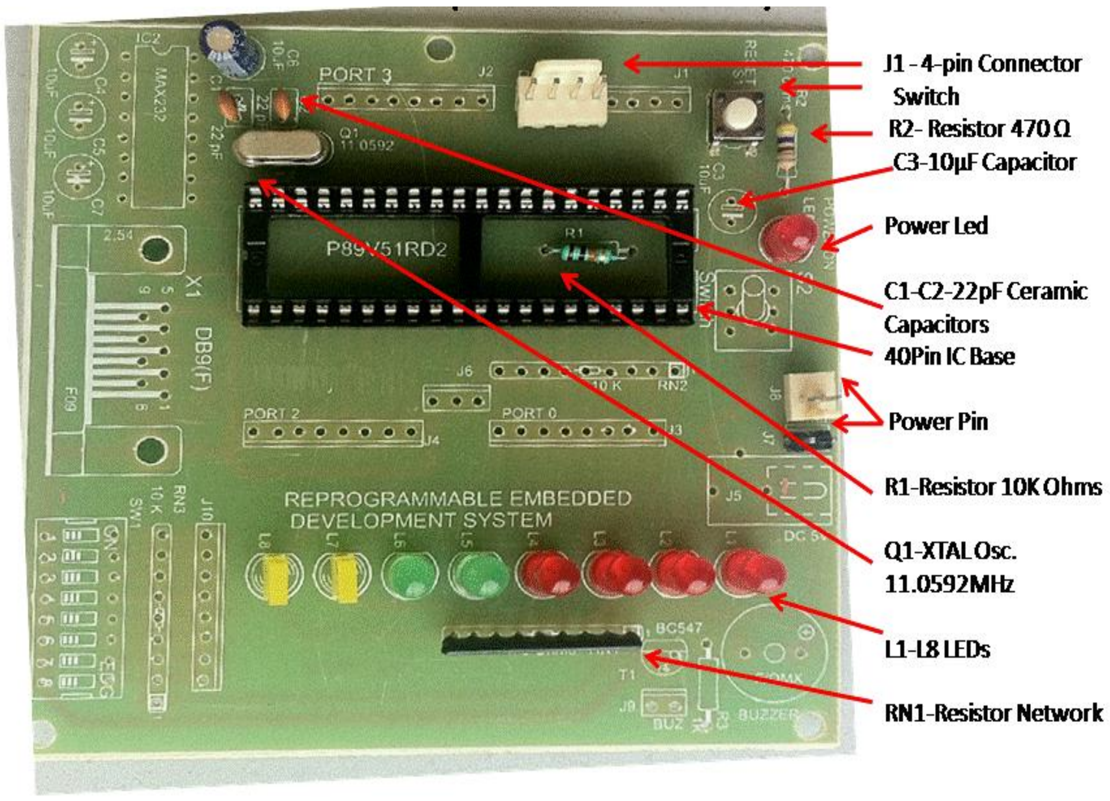


Figure 1.5: Assembled PCB for Reprogrammable Embedded System

Table 1.1

1.7 LIST OF COMPONENTS FOR THE EMBEDDED DEVELOPMENT SYSTEM

		Components	Qty./board
ICs			
	1	P89V51RD2/ AT89S52	1
	2	MAX232	1
CAPACITORS			
	1	10uF(electrolytic)/63V	5
	2	22pF(ceramic)	2
KEYS			
	1	DIP SWITCH	1
	2	ON/OFF(push to On push to OFF)	1
CRYSTAL			
	1	11.0592 MHz	1
LEDS			
	1	LED(3mm)	9
RESISTANCES			
	1	10K RESISTANCE NETWORK(9 -pin)	2
	2	1K RESISTANCE NETWORK(9 -pin)	1
	3	470E	1
	4	1K	1
CONNECTORS			
	1	SERIAL PORT(DB-9) RIGHT ANGLED FEMALE	1
	2	SERIAL PORT(DB-9) MALE	1
	3	SERIAL PORT (DB-9) FEMALE	1
	4	CONNECTOR (8-PIN) MALE+ FEMALE	2
	5	CONNECTOR (2-PIN) MALE+FEMALE	2
	6	BERGSTICK MALE	1
	7	CASING (DB-9 CONNECTOR)	2
	8	FEMALE TO FEMALE SINGLE CONNECTING WIRE	10
TRANSISTORS			
	1	BC547	1
IC base			
	1	40-pin	1
	3	16-pin	2
		PCB	1

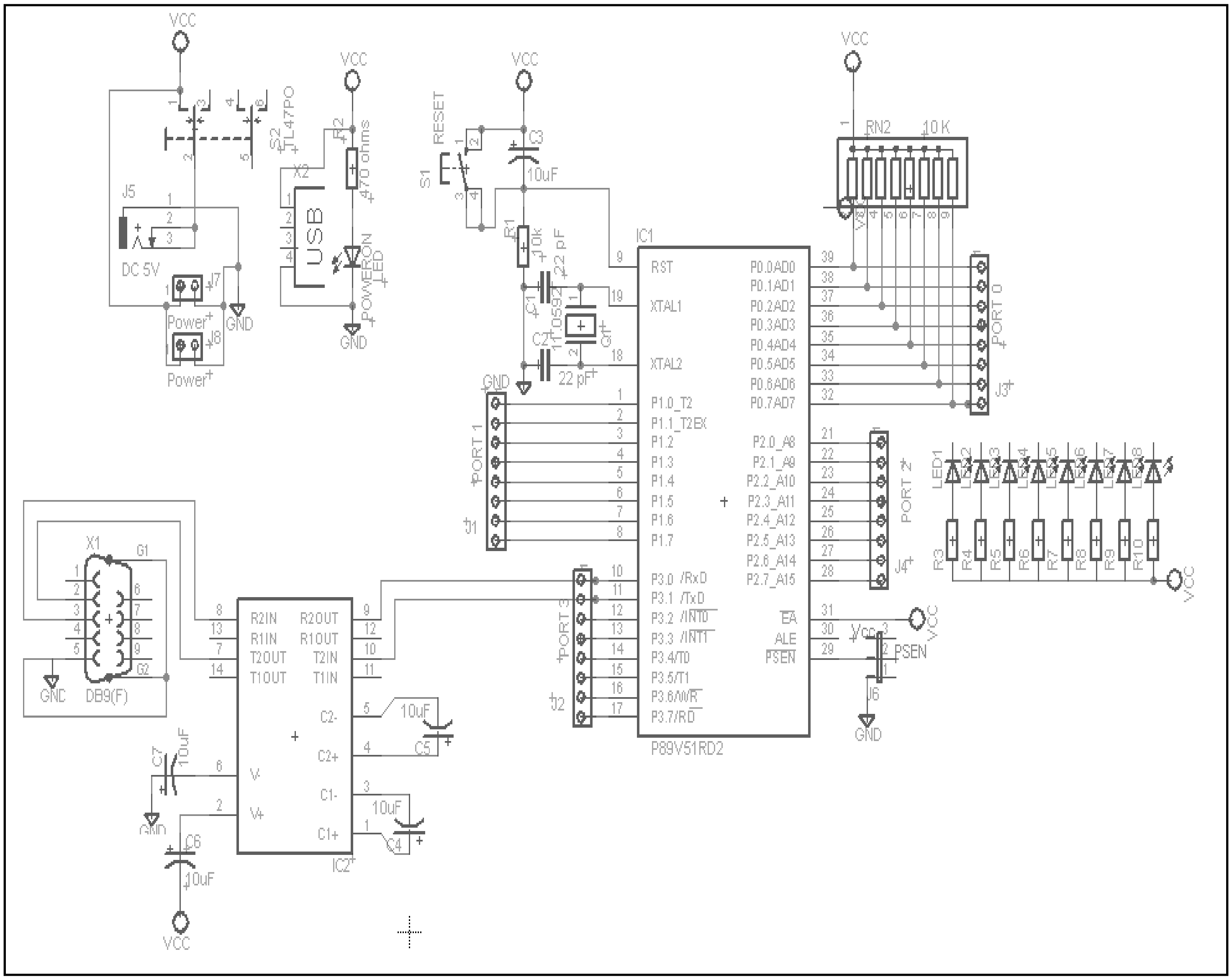


Figure1.6: Schematic Diagram for reprogrammable Embedded Board

1.7 Programmer

The purpose of the programmer is to transfer HEX code from PC to appropriate pins and provide regular voltage levels during chip programming as well. For this development system, the programmer is freely available FlashMagic (for P89V51RD2) or 89SXX from Sunrom Technologies (for AT89S52) connected to PC via Serial cable/ USB cable. When the process of programming is completed, microcontroller pins used for it are automatically available for other application.

1.7.1 Source Code: Table 1.2

; Blinking of LED at port 2.7. Some delay is generated to see LED blinking(LA and LB loop)

```
            ORG 0000
START:      NOP
            MOV R0, #0FFH;
LB:         MOV R1, #0FFH;
```

```
LA:          NOP;
             DJNZ R1, LA;
             DJNZ R0, LB;
             CPL P2.7
             SJMP START
             END
```

1.8 Result: Embedded system board (as shown in figure 1.5) is soldered, checked and found working.

1.9 Conclusion:- Target board of such types can be designed using very less amount of components and can be used for many applications.

1.10 Remarks:- Other controllers of 8051 architecture and having same pinouts can also be tested and used on same target board .

1.11 References:-

1. Datasheet Max232
2. Datasheet P89v51RD2/AT89S52
3. *http://www.cadsoftusa.com/shop/eagle-hobbyist-and-education/*
4. *https://www.cadsoftusa.com/download-eagle/*
5. *http://www.cadsoft.de/wp-content/uploads/2011/05/V6_tutorial_en.pdf*
6. *http://www.mikrocontroller.net/attachment/17909/Protel_99_SE_Traning_Manual_PCB_Design.pdf*

Experiment 2

Tool Chain of Keil IDE (Embedded Development Tool Chain) with the example of LED Blinking Program

2.1 Objective: To understand the procedure of creating source code for reprogrammable embedded system board using IDE such as Keil µVision.

2.2 Software Requirement: Editor like Keil µVision Ver 4 or less.

2.3 Description:

Understanding any processor or controller needs familiarity with its architecture and instruction set. Any architecture can be best understood using its instruction set through different programs.

One may use assembly language or embedded C for writing programs. Programs written in assembly language are completely processor dependent and need major changes when converting to other processor. While programs written in C are generally independent of processor and needs minor changes during conversion to other processors.

C is thus preferred for programming. But to know and understand a processor better, one must be familiar with assembly language.

All source code written in this document will be written using assembly language for 8051 architecture.

Some development environment is needed to prepare any application. An editor is needed first to provide a platform for writing programs i.e. source code.

A source code written in assembly/C language is needed to be converted to machine language (hex code) before programming into processor. This conversion is done by compiler which converts assembly/C language code to hex code.

IDE i.e. Integrated development Environment, serves both these purposes as well as provide debugging facility.

Assembly language file will be stored by extension **.asm**, C file by extension **.c** and hex file by extension **.hex**.

2.4 Procedure:

Many free software are available for educational purpose e.g. Keil, SDCCDown load free tools for IDE from

*www.**keil**.com/**download**/product*, *https://www.keil.com/demo/eval/c51.htm*

IDE for 8051 architecture can be downloaded using these links. It’s an integrated development environment for creation and compilation of assembly/C source code for any 8051 architecture based target boards. It also provides debugging facility [1].

Steps:

2.4.1.1 Click on Keil µVision4 icon for getting started.

2.4.1.2 Click on **Project tab**>Make new project> Select target device.

2.4.1.3 Click on **File**>New file.

2.4.1.4 Prepare a test code in assembly language as shown in editor window. Save it with **.asm** extension.

2.4.1.5 Add this created file to project. One may add one or more than one file in a single project.

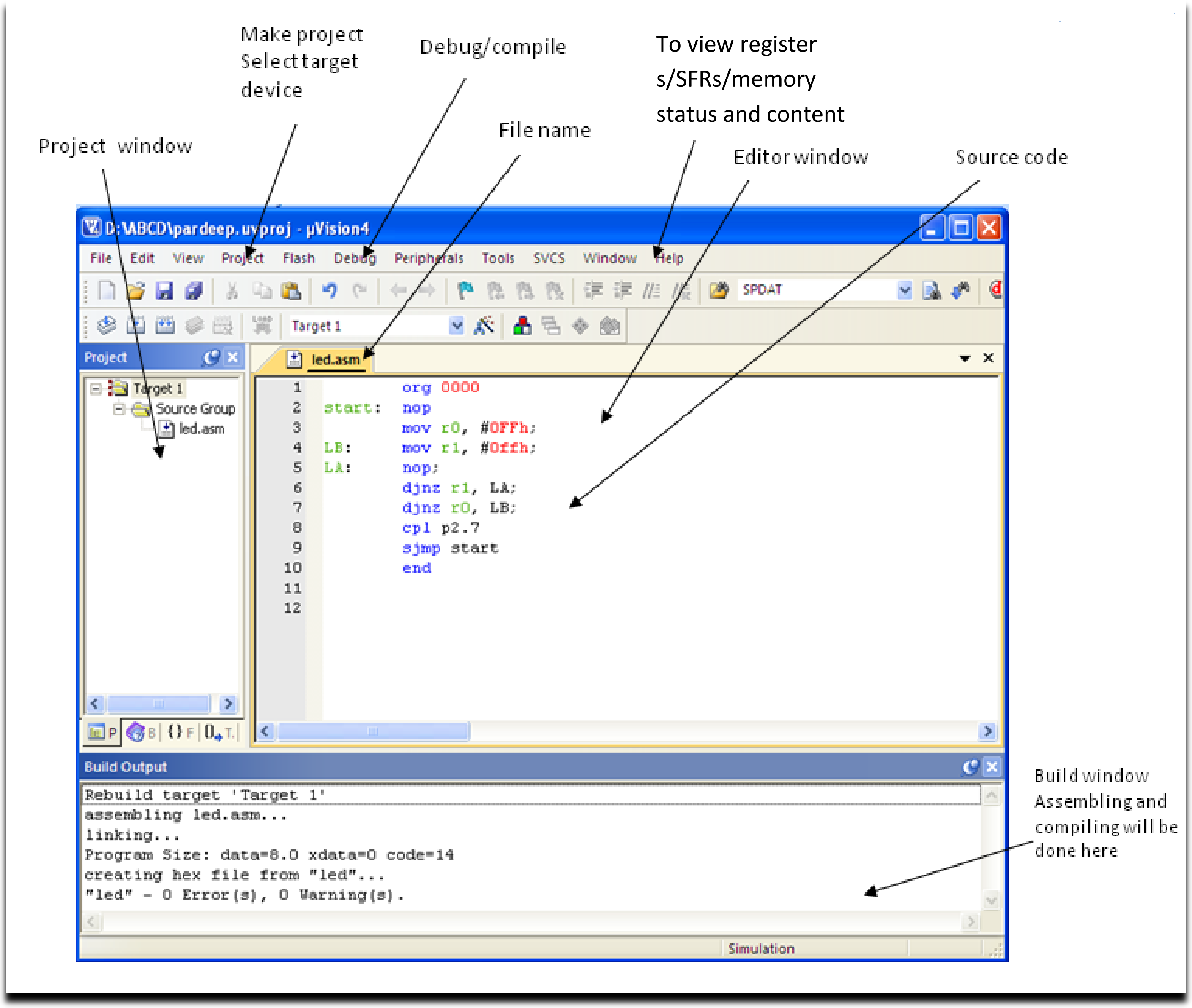
2.4.1.6 Click **Target1** (at left side pane)>Source Group> Right click to add code file.

2.4.1.7 Open **Project tab**> Options for target target1> Output tab>**check ‘create hex file’ option.**

2.4.1.8 Open **Project tab**> Build target. This will generate compiled **.hex** file from the **.asm** or **.C** file, in the project created.

One may refer Help tab for further help for using this tool.

*Many video are there in you tube which may be referred to understand the procedure better.



```
        org 0000
start:  nop
        mov r0, #0FFh;
LB:     mov r1, #0ffh;
LA:     nop;
        djnz r1, LA;
        djnz r0, LB;
        cpl p2.7
        sjmp start
        end
```

```
Rebuild target 'Target 1'
assembling led.asm...
linking...
Program Size: data=8.0 xdata=0 code=14
creating hex file from "led"...
"led" - 0 Error(s), 0 Warning(s).
```

Figure 2.1 IDE- Keil µVision

2.5 Result: Sample program for LED blinking is written, compiled and hex file generated.

2.6 Conclusion:- Different programs can be written, debugged and simulated using IDE.

2.7 Remarks:- Different programs should be written and tested using assembly/C language for better understanding of the tool .

2.8 References:-

[1] *https://www.keil.com/demo/eval/c51.htm*

Experiment 3

Flash Programming Using Serial UART Flash Programmer- FlashMagic

3.1 Objective: To understand the procedure of flash programming source code for reprogrammable embedded system board using NXP's FlashMagic.

3.2.1 Software Requirement: FlashMagic.

3.2.2 Hardware Requirement: Target board with P89V51RD2 controller as per circuit given in figure 1.6, Serial Cable with DB9 Connector, Power Cable.

NOTE: Choose target processor which supports In-system programming (ISP) e.g. Phillips P89V51RD2.

3.3 Procedure:

1. Down load free tools for flash programmer from

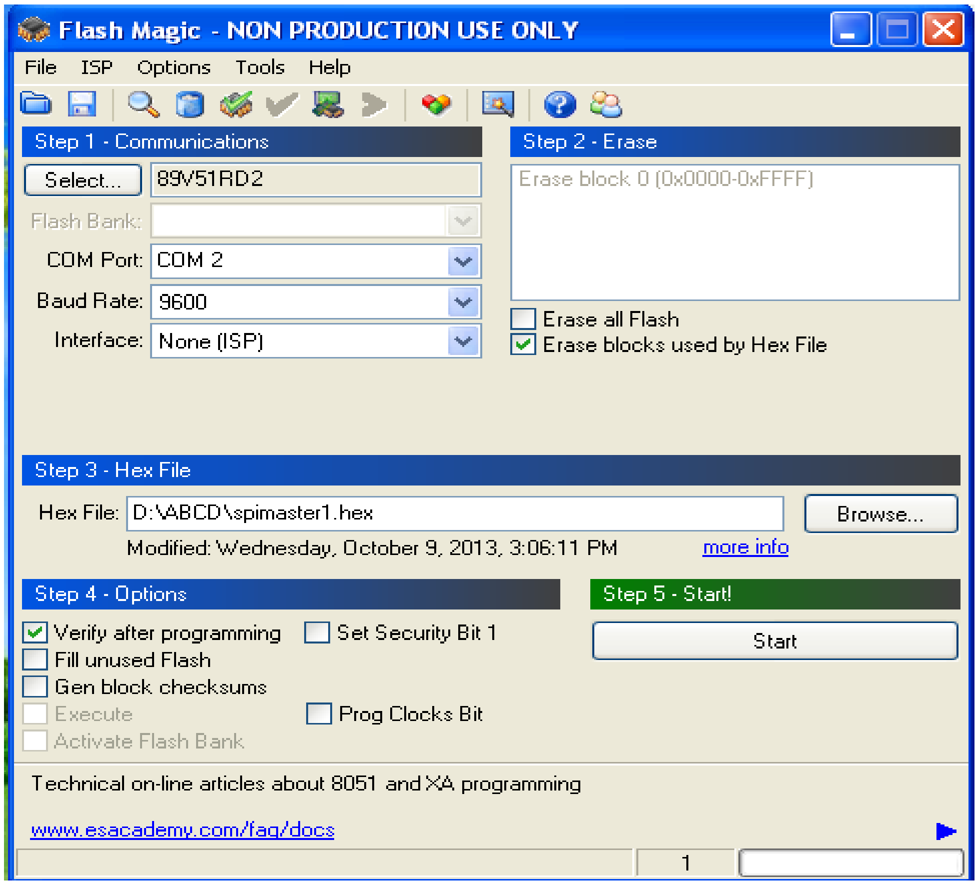
*www.**flashmagic**tool.com*
Save and run **FlashMagic.exe**
A serial port flash programmer will be downloaded then.

3.3.1 Requirements: Flash Magic works on Windows 2000, XP, Vista and 7. 10Mb of disk space is required for reliable operation of the said tool.

Once install, Flash magic will look like figure 3.1.

Recommended settings for flash magic are as shown in figure 3.1.
Connect target board with serial port of PC using Serial cable (standard 9 pin cable is recommended). Flash magic tool steps are described as under.

- Step1 :- Select target device, choose serial COM port, baud rate and Interface as None.
- Step2:- Choose desired option for flash erase.
- Step3:- Browse desired hex file that to be flash programmed into target controller.
- Step4:-Select desired options for programming. Verify after programming is recommended. Beginners should not use other options as they may lock their target device.
- Step 5:- After all setting are done press Start programming button.



89V51RD2

Flash Bank:

COM Port: COM 2

Baud Rate: 9600

Interface: None (ISP)

Erase block 0 (0x0000-0xFFFF)

[ ] Erase all Flash

[✓] Erase blocks used by Hex File

Hex File: D:\ABCD\spimaster1.hex

Modified: Wednesday, October 9, 2013, 3:06:11 PM

[✓] Verify after programming

[ ] Set Security Bit 1

[ ] Fill unused Flash

[ ] Gen block checksums

[ ] Execute

[ ] Prog Clocks Bit

[ ] Activate Flash Bank

Technical on-line articles about 8051 and XA programming

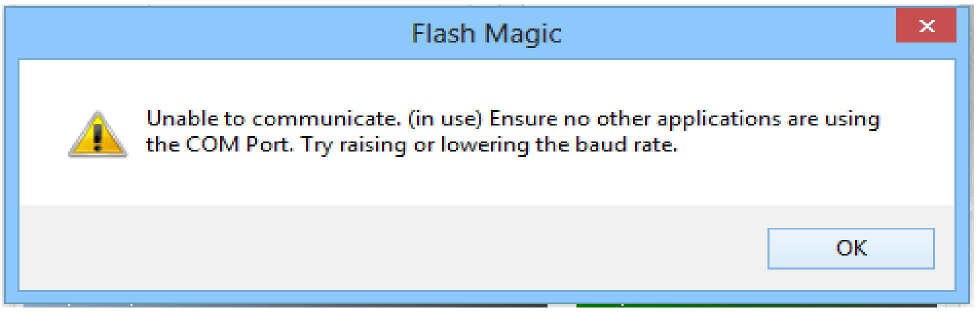
www.esacademy.com/faq/docs

1

Figure 3.1 FlashMagic Programmer

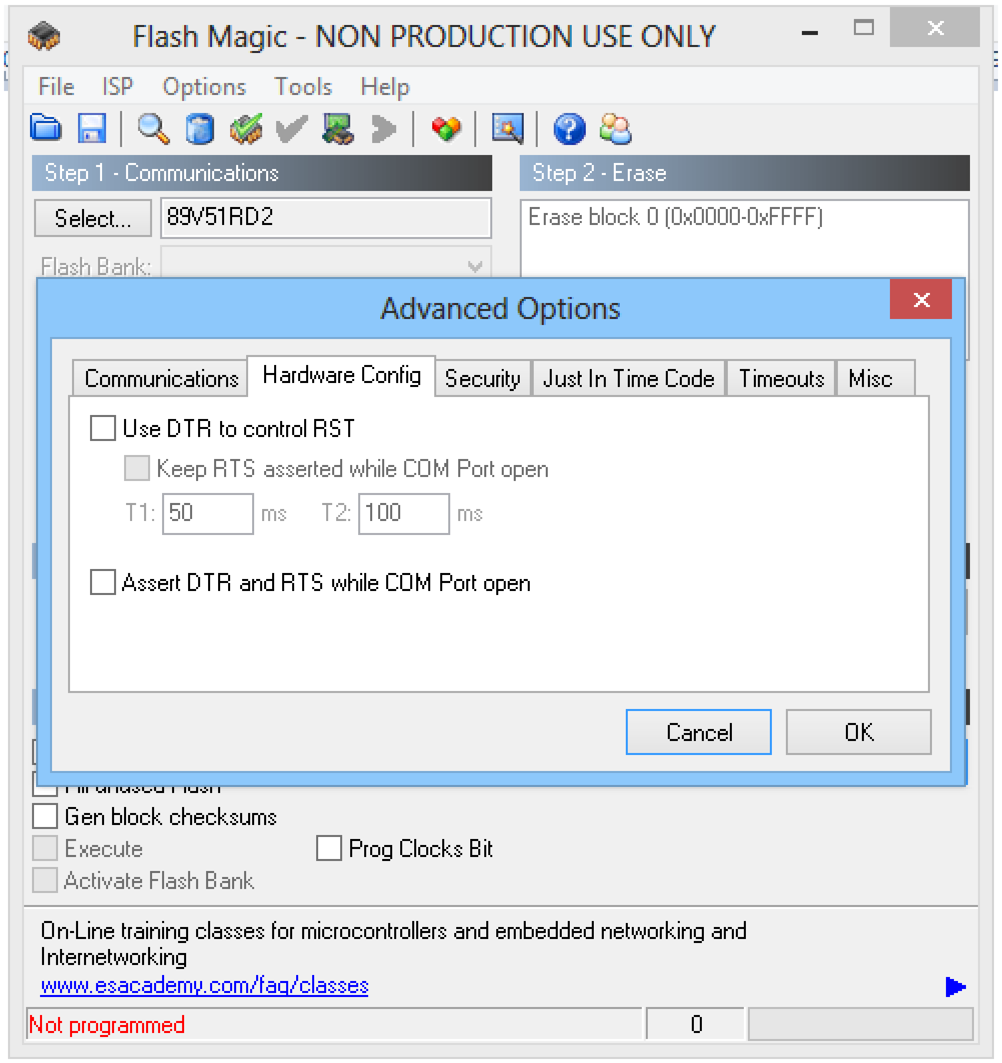
If any error message, as shown in figure 3.2 appears then click

Options->Advance Options->Hardware Config and uncheck both the options as shown in figure 3.3



Unable to communicate. (in use) Ensure no other applications are using the COM Port. Try raising or lowering the baud rate.

Figure 3.2 FlashMagic Programmer Communication Error Message



89V51RD2

Erase block 0 (0x0000-0xFFFF)

- [ ] Use DTR to control RST
- [ ] Keep RTS asserted while COM Port open

T1: 50 ms T2: 100 ms

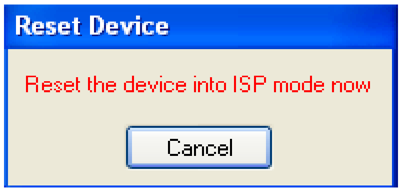
- [ ] Assert DTR and RTS while COM Port open

On-Line training classes for microcontrollers and embedded networking and Internetworking
www.esacademy.com/faq/classes

Not programmed 0

Figure 3.3 FlashMagic Programmer Communication Advance Options

Once communication between flash magic and target board established, after pressing start button, flash programmer will show message as in figure 3.4



Reset the device into ISP mode now

Figure 3.4 Reset Message

Reset processor at target board.
Flash programming will be started then.

After successful programming Finished message can be seen at bottom left corner of flash programmer screen.
Target board is now ready to be used. It will run as per the hex file loaded.

3.4 Result: Sample program for LED blinking is programmed and LED blinking is observed as per program at target board.

3.5 Conclusion:- Different hex files can be programmed and checked using flash programmer.

3.6 Remarks:- Different programs should be programmed and tested using assembly/C language for better understanding the tool .

3.7 References:-

[1] *www.**flashmagic**tool.com*

Experiment 4

Flash Programming Using SPI Flash Programmer- 89SXX

4.1 Objective: To understand the procedure of flash programming source code for reprogrammable embedded system board using Sunrom technology's 89SXX USB flash programmer, **ISP Model: 1315.**

4.2.1 Software Requirement: 89SXX programmer frontend.

4.2.2 Hardware Requirement: Target board with Atmel AT89S52 controller as per circuit given in figure 1.12, Power Cable.

NOTE: Choose target processor which supports In-system programming (ISP) using SPI e.g. Atmel AT89S52.

4.3 Procedure:

1. Down load free tools for flash programmer application from *http://www.sunrom.com/337*
2. For 89SXX USB flash programmer, **ISP Model : 1315** (figure 4.1)
3. **Save and run** **1100_setup.exe**
4. A SPI flash programmer application will be downloaded then (figure 4.2).
5. **89SXX USB flash programmer (figure 4.1) will program target board (figure 4.5) using 1100_setup.exe (figure 4.2).**

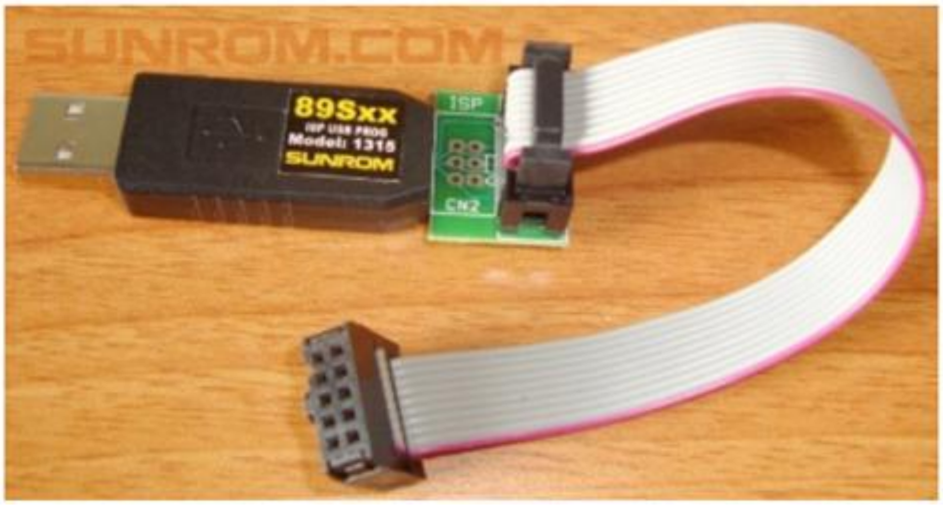


Figure 4.1 89SXX USB ISP Flash Programmer (SPI based)

Click icon (figure 4.2) to start programming application (figure 4.3).

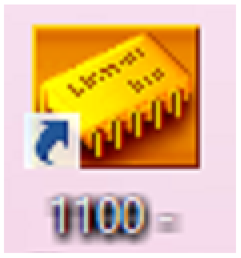
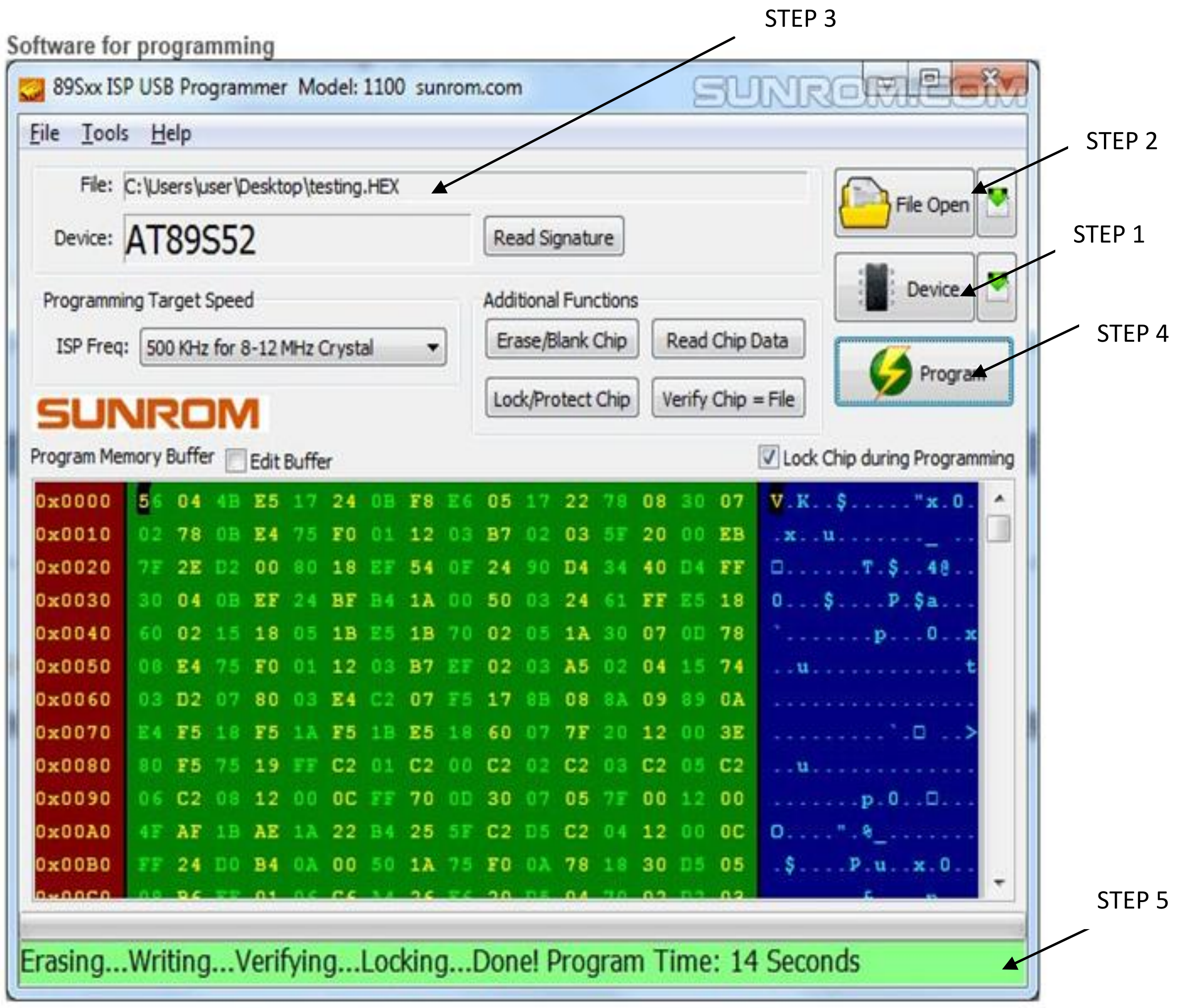


Figure 4.2 Application icon



File: C:\Users\user\Desktop\testing.HEX

Device: AT89S52

Programming Target Speed

ISP Freq: 500 KHz for 8-12 MHz Crystal

Additional Functions

☐ Edit Buffer

✓ Lock Chip during Programming

```
0x0000  56 04 4B E5 17 24 0B F8 E6 05 17 22 78 08 30 07  V.K..$....."x.0.
0x0010  02 78 0B E4 75 F0 01 12 03 B7 02 03 5F 20 00 EB  .x..u......._ ..
0x0020  7F 2E D2 00 80 18 EF 54 0F 24 90 D4 34 40 D4 FF  □......T.$..4@..
0x0030  30 04 0B EF 24 BF B4 1A 00 50 03 24 61 FF E5 18  0...$....P.$a...
0x0040  60 02 15 18 05 1B E5 1B 70 02 05 1A 30 07 0D 78  `.......p...0..x
0x0050  08 E4 75 F0 01 12 03 B7 EF 02 03 A5 02 04 15 74  ..u............t
0x0060  03 D2 07 80 03 E4 C2 07 F5 17 8B 08 8A 09 89 0A  ................
0x0070  E4 F5 18 F5 1A F5 1B E5 18 60 07 7F 20 12 00 3E  .........`.□ ..>
0x0080  80 F5 75 19 FF C2 01 C2 00 C2 02 C2 03 C2 05 C2  ..u.............
0x0090  06 C2 08 12 00 0C FF 70 0D 30 07 05 7F 00 12 00  .......p.0..□...
0x00A0  4F AF 1B AE 1A 22 B4 25 5F C2 D5 C2 04 12 00 0C  O....".%_.......
0x00B0  FF 24 D0 B4 0A 00 50 1A 75 F0 0A 78 18 30 D5 05  .$....P.u..x.0..
```

Erasing...Writing...Verifying...Locking...Done! Program Time: 14 Seconds

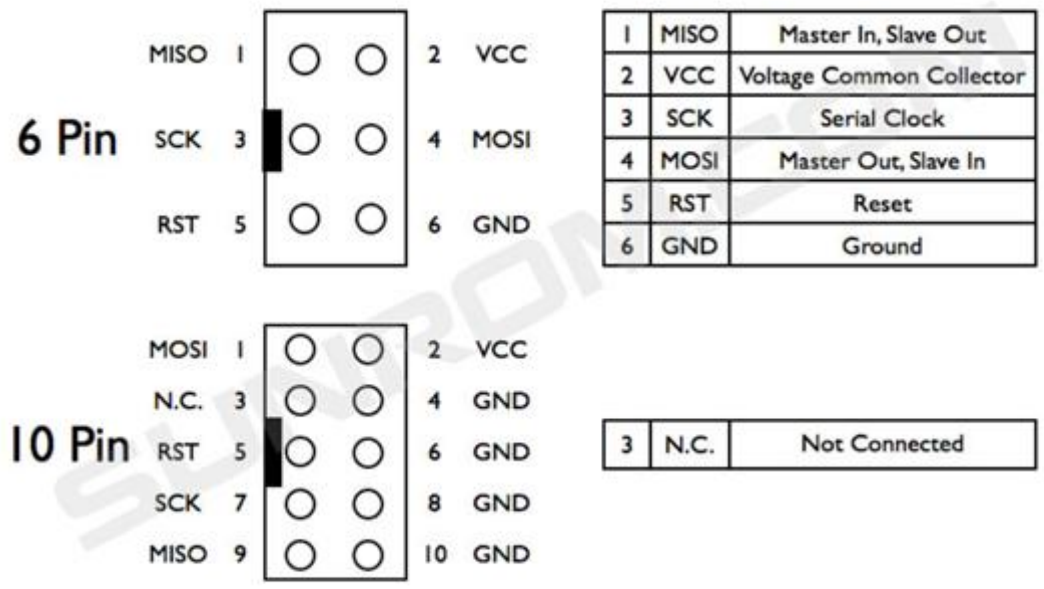
Figure 4.3 Application 89SXX USB ISP Flash Programmer (SPI based)

Connect USB programmer with computer and target board. Steps to be followed are as described under.

Step1:- Select Device> 8051 MCU>Atmel>AT 89S52>OK

Step2:- Browse intended hex file.

Step3:- Selected file will appear here.
Step4:- Click Program to flash program the selected device.
Step5:- During programming, different programming status can be seen here.
Pin configuration for ISP connectors are as shown in figure 4.4.



1	MISO	Master In, Slave Out
2	VCC	Voltage Common Collector
3	SCK	Serial Clock
4	MOSI	Master Out, Slave In
5	RST	Reset
6	GND	Ground

3	N.C.	Not Connected

Figure 4.4 Pin Configuration for USB 89SXX ISP Programmer [3]

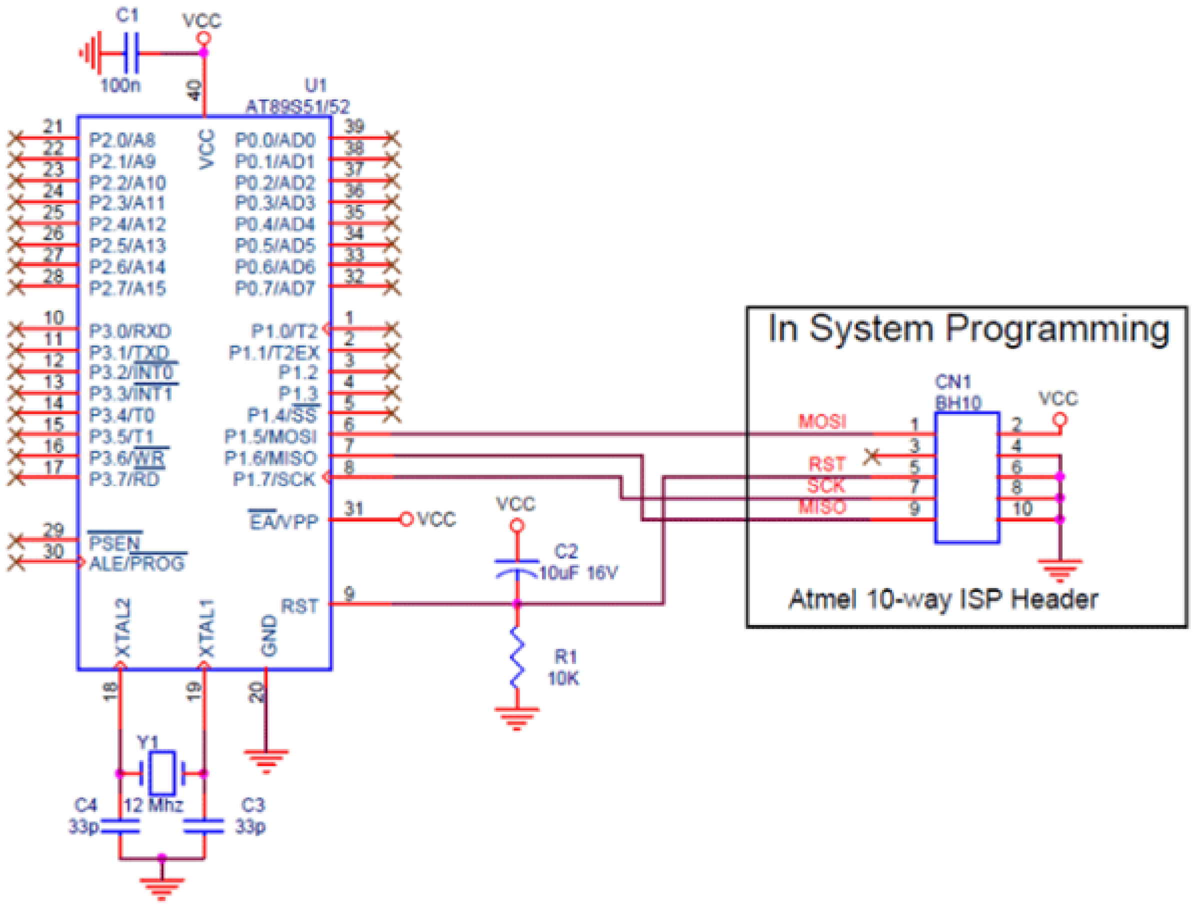


Figure 4.5 Schematic diagram showing connection between microcontroller and programmer [3]

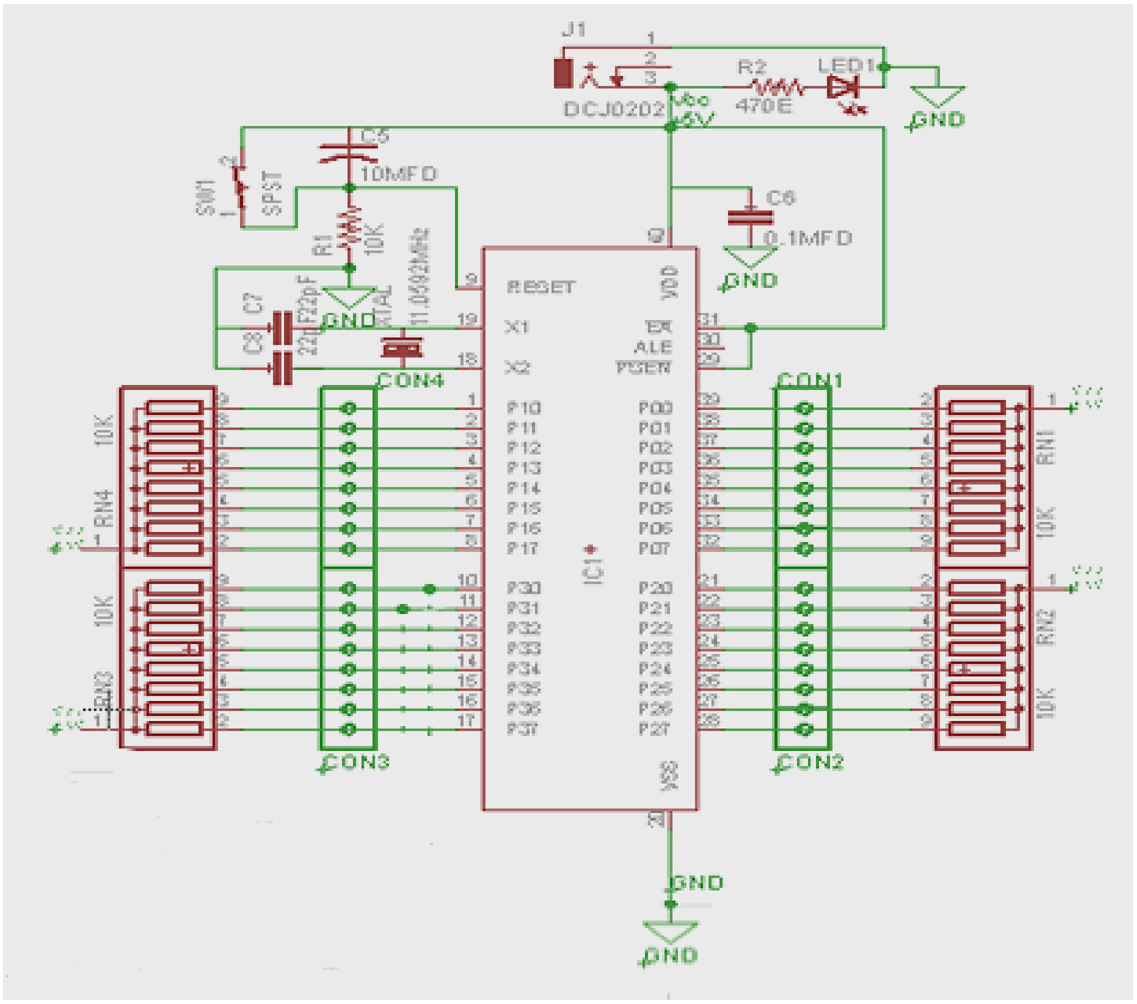


Figure 4.5 Schematic diagram of target board using ATMEL microcontroller

4.4 Source Code: Table 4

;Blinking of LED at port 2.7. Some delay is generated to see LED blinking(LA and LB loop)

```
            ORG 0000
START:      NOP
            MOV R0, #0FFH;
LB:         MOV R1, #0FFH;
LA:         NOP;
            DJNZ R1, LA;
            DJNZ R0, LB;
            CPL P2.7
            SJMP START
            END
```

4.5 Result: Target board is programmed and blinking of LED at port p2.7 i.e. pin no 28 of microcontroller AT89S52 is obtained.

4.6 Conclusion: With very low component count and easily and freely available tools, designing and programming of small and low cost systems can be achieved.

4.7 Remarks: Any IDE and any flash programmer can be used. Keil µVision, FlashMagic and 89SXX USB programmers are recommended and tested here, for their respective purposes. All these tools are selected due to their easy accessibility, low cost and wide acceptance. Help on these tools are easily available.

As newer PC/laptops are equipped with USB port, one can also think for USB base programmers. One option is to choose Universal programmer, they are highly flexible in terms of device selection but are higher in cost. For simple projects, one can easily rely on chip specific programmer which may not provide flexibility in device selection, but are quite cheep.

***Note:-** USB to RS232 converter will be needed to use Flash magic from PC's USB port.

4.8 Reference:-

1. *www.keil.com*
2. *www.flashmagictools.com*
3. *http://www.sunrom.com/337*

Experiment No. 5

Multiple Controllers Programming Using Flash Magic

5.1 Objective: To understand the procedure and connections for multiple controllers programming of same type of controller with same source code in one go, using flash magic.

5.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

5.2.2 Hardware Requirement: 2 Sets of target boards with P89V51RD2 controller as per circuit given in figure 1.6, Serial Cable with DB9 Connector.

5.3 Procedure:

1. Connect circuit as per given block diagram of figure 5.1. These connections will facilitate programming of multiple controller of same type, simultaneously. Here it is checked for 2 controllers.
2. Use source code of experiment 1.
3. Set flash programmer for programming as per given figure 5.2.
4. Open Options-> advance option-> Misc->Check, disable device signature checking.
5. Start flash programming.
6. **Reset both the target controllers simultaneously when prompted.**
7. Click ok for warning message.

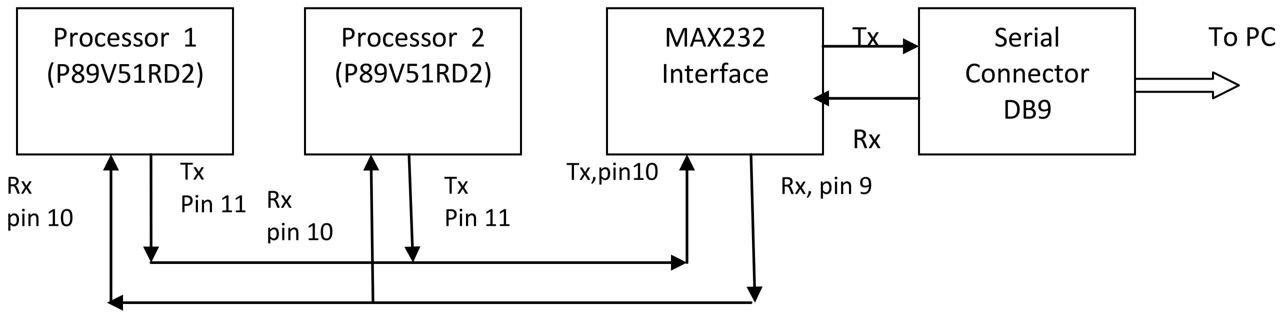
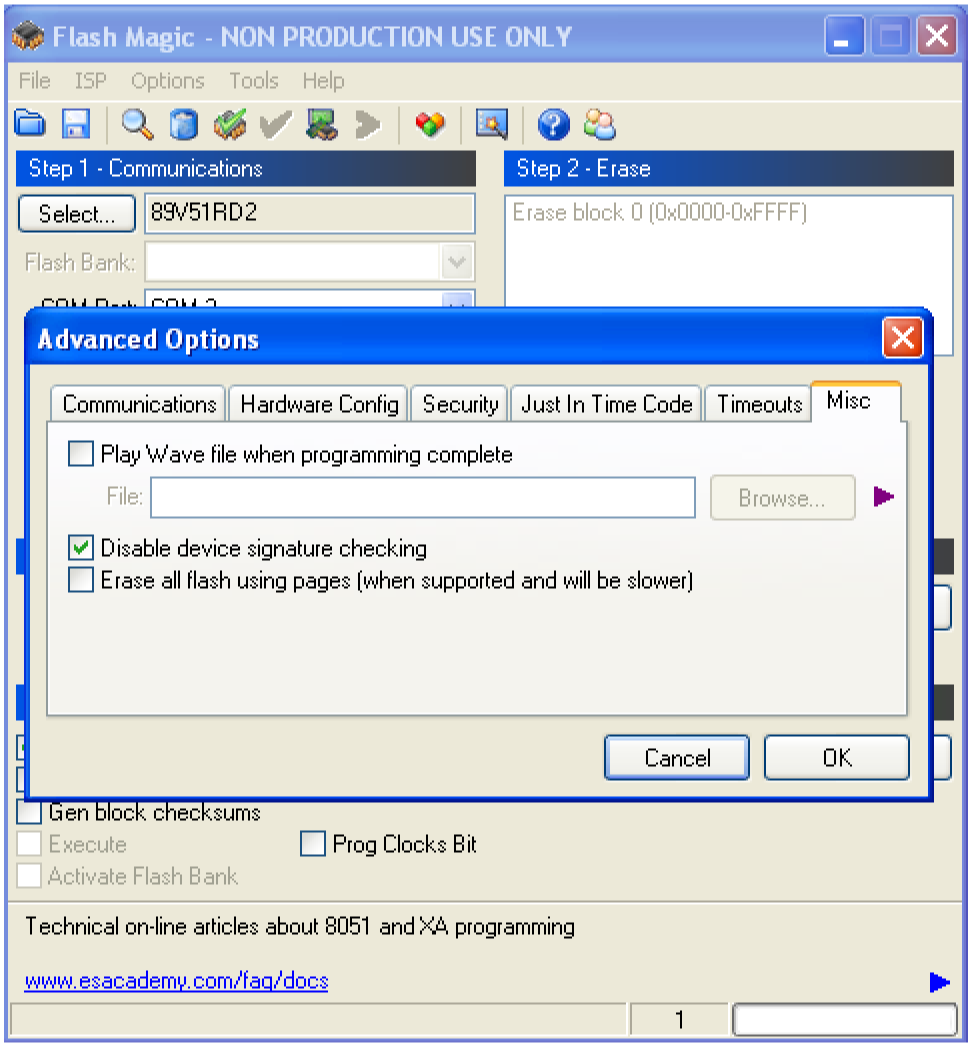


Figure 5.1 Block diagram for multi controller programming, connections


Flash Magic - NON PRODUCTION USE ONLY

Step 1 - Communications

Select... 89V51RD2

Flash Bank:

Step 2 - Erase

Erase block 0 (0x0000-0xFFFF)

Advanced Options

Communications | Hardware Config | Security | Just In Time Code | Timeouts | Misc

- [ ] Play Wave file when programming complete
- [x] Disable device signature checking
- [ ] Erase all flash using pages (when supported and will be slower)

- [ ] Gen block checksums
- [ ] Execute
- [ ] Prog Clocks Bit
- [ ] Activate Flash Bank

Technical on-line articles about 8051 and XA programming

www.esacademy.com/faq/docs

Figure 5.2 Flash programmer settings for multi controller programming

5.4 Result: Both the controllers are programmed simultaneously with same program.

5.5 Conclusion:- Multiple controllers can be programmed in one go with same source code and hence beneficial in time saving for large production. Target board there can be designed with some jumper settings to facilitate multiple programming connections.

5.6 Remarks:- Single max232 interface is used here to program multiple controllers and hence care must be taken as this may increase voltage levels of max232 which may result in its damage.

5.7 References:-

1. Datasheet Max232
2. Datasheet P89v51RD2

3.

Experiment 6

Interfacing LEDs at Input/ Output Port

6.1 Objective: To interface 8 LEDs at Input-output port and create different patterns.

6.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

6.2.2 Hardware Requirement: Target boards with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Serial Cable with DB9 Connector/USB 89SXX Programmer with cable.

6.3 Procedure:

1. Connect 8 LEDs at port 2 (pin 21 to 28) as per given block diagram of figure6.1. Rest of the circuit will remain same as figure 1.6/4.5
2. Write program for LED blinking at port.
3. Build/Compile project.
4. Flash program and observe results.

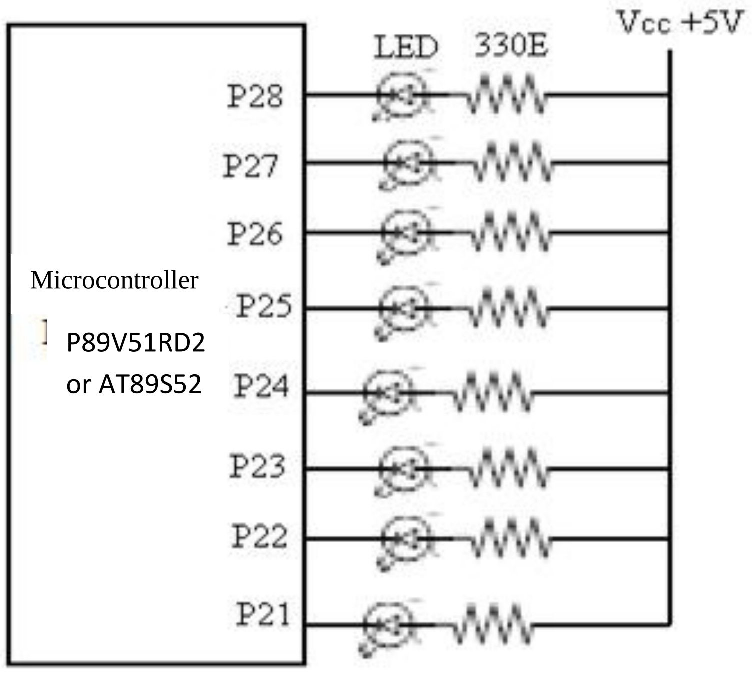


Figure 6.1 LED connections at port2

6.4 .1 Source Code : Table 6A

```
;Program written using assembly language of
8051 architecture for P89V51RD2
;This program will lit one LED at a time
;starting from port 2.0 to port 2.7 and create
;LED rotation pattern. All other LEDs will
;remain off.
                ORG 0100

MAIN:           MOV P2,#0FEH
                LCALL DELAY
                MOV P2,#0FDH
                LCALL DELAY
                MOV P2,#0FBH
                LCALL DELAY
                MOV P2,#0F7H
                LCALL DELAY
                MOV P2,#0EFH
                LCALL DELAY
                MOV P2,#0DFH
                LCALL DELAY
                MOV P2,#0BFH
                LCALL DELAY
                MOV P2,#7FH
                LCALL DELAY
                LJMP MAIN

; Subroutine for some delay generation, to
view ;LED blinking

DELAY:          MOV R0,#0FFH
LOOP1:          MOV R1,#05H
LOOP2:          MOV A,#05H
LOOP3:          DEC A
                JNZ LOOP3
                DJNZ R1,LOOP2
                DJNZ R0,LOOP1
                RET
                END
```

***Bold lines shows change in instructions**

6.4.2 Source code : Table6B

Same function can be achieved using source code 3B with less number of instructions

```
;Rotate accumulator left command is use to
;create same rotating effect.
                ORG 0100
MAIN:           MOV A,#0FEH
MAIN1:          MOV P2,A
                LCALL DELAY
                LCALL DELAY
                RL A
                LJMP MAIN1

; subroutine for some delay generation

DELAY:          MOV R0,#0FFH
LOOP1:          MOV R1,#05H
LOOP2:          MOV R2,#05H
LOOP3:          DJNZ R2,LOOP3
                DJNZ R1,LOOP2
                DJNZ R0,LOOP1
                RET
                END
```

6.5 Result: Output at LEDs observed as per programs, i.e. LED rotating in circular manner is achieved at port2.

6.6 Conclusion:- with smart use of instructions memory area of controller can be saved as can be seen from above example.

6.7 Remarks:- Different LED patterns can be generated and tested e.g. blinking of all LEDs, Blinking of alternate LEDs, 8 bit/4bit binary pattern generation, BCD number pattern generation, dancing LED etc.

6.8 Reference :- Datasheet 89V51RD2/ AT89S52

Experiment 7

Use of I/O pins for data transfer between controllers

7.1 Objective: To use general purpose port i.e. Input/ output port of two controllers for data transfer between them.

7.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

7.2.2 Hardware Requirement: 2 sets of target boards with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Serial Cable with DB9 Connector/ USB 89SXX Programmer with cable, some connectors.

7.3 Procedure:

1. Connect target boards as per given block diagram of figure 7.1.
2. Write programs, for both the controllers.
3. Build/Compile project.
4. Flash program both the controllers with respective source codes and observe results.

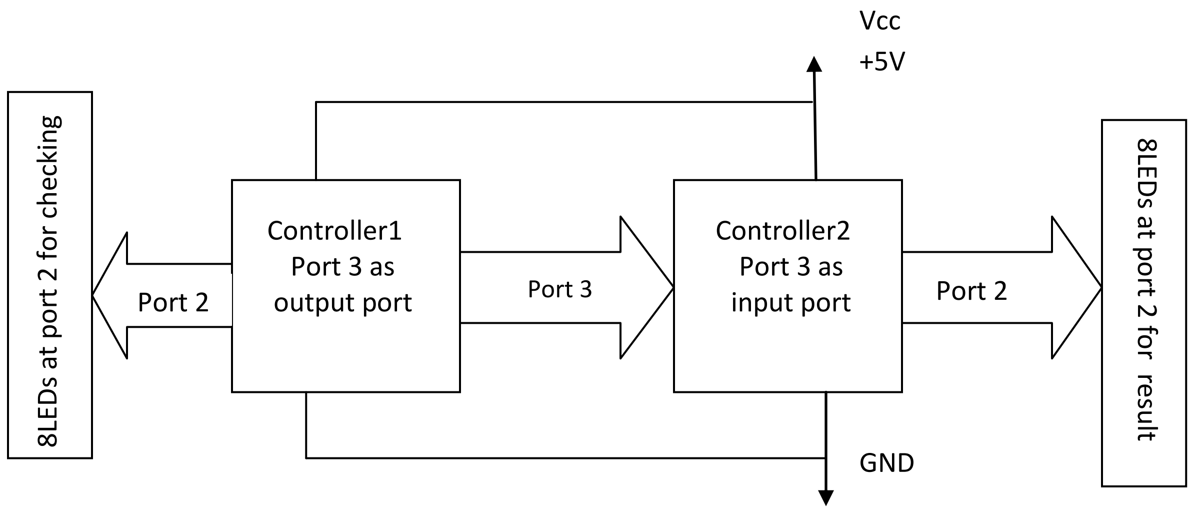

Vcc
+5V
8LEDs at port 2 for checking
Port 2
Controller1 Port 3 as output port
Port 3
Controller2 Port 3 as input port
Port 2
8LEDs at port 2 for result
GND

Figure 7.1 Two microcontrollers connected through port3

7.4.1 Source code: Table 7A

```
;Program for microcontroller1, i.e. for Sender
;controller
            ORG 0000H
START:      MOV A,#0FEH
LOOP:       RR A
            MOV P3,A
            MOV P2,A
            LCALL DELAY
            SJMP LOOP
```

```
;Subroutine for delay
DELAY:      MOV R0,#02
LOOP1:      MOV R1,#255
LOOP2:      MOV R2, #255
LOOP3:      DJNZ R2, LOOP3
            DJNZ R1, LOOP2
            DJNZ R0, LOOP1
            RET
            END
```

7.4.2 Source code: Table 7B

```
;Program for microcontroller2, ie for receiver controller. These controllers are connected
;through port3. No delay program is written here as delay will be taken care by controller1
;routine.
            ORG 0000H
START:      NOP
LOOP1:      MOV A,P3
            MOV P2,A
            SJMP LOOP1
            END
```

7.5 Result: Output observed at LEDs, of both the target boards as per programs, i.e. LED lighting in circular manner is achieved at port2 of controller2 same as of controller1 at port2.

7.6 Conclusion:- General purpose Input/Output pins can also be used to transfer data between ports. However this process will be slow but benefit lies in its easiness.

7.7 Remark:- Both controllers can send and receive data from other controller at different as well as at same port but not at the same time. Different programs can be tested for such functionalities.

7.8 Reference :-

1. Datasheet 89V51RD2/AT89S52

Experiment 8

Memory Block Programming Using Flash Magic

8.1 Objective: To achieve block wise programming of memory for controller using flash magic.

8.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

8.2.2 Hardware Requirement: Target board with P89V51RD2 controller as per circuit given in figure 1.6, Serial Cable with DB9 Connector.

8.3 Procedure:

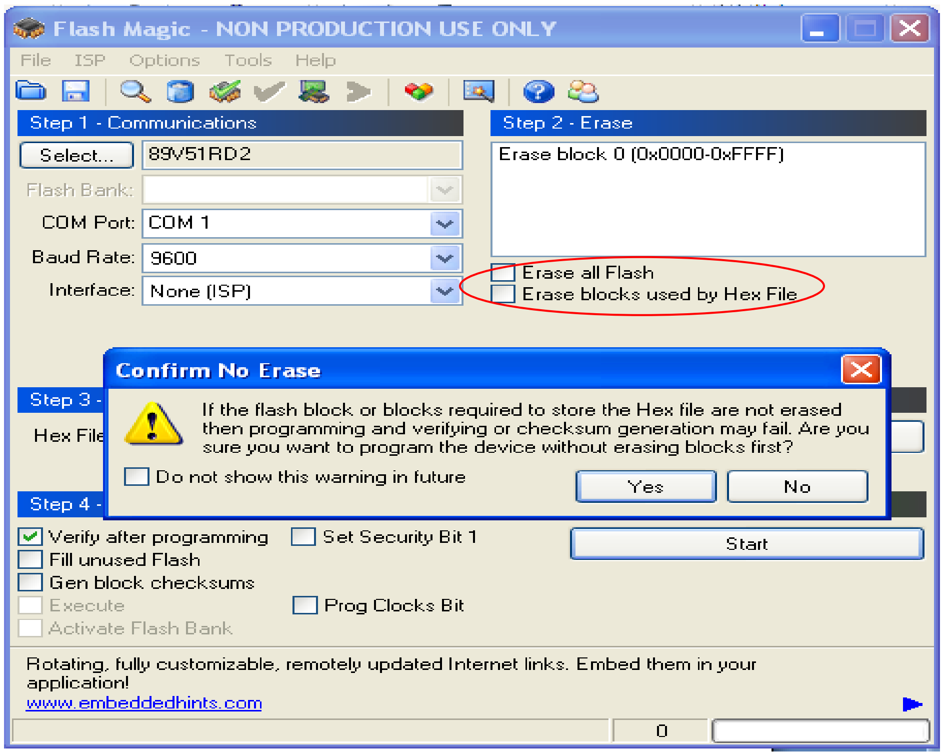
1. Write source codes (e.g. LED blinking at port) , for different blocks of memory.
2. Add all source codes into one µvision project (as described in figure 8.2).
3. Build/Compile project. Compiled file i.e. hex file will contains all source codes into one file.
4. Before programming the controllers, un-check all memory erase options (as described in figure 8.1) in flash programmer.
5. Flash program the controller and observe results.

8.4 Source Code: Table 8

;First block of program starts from location ;50h, on/off LED at port2.0 for some time	;Third block of program starts from location ;200h, on/off LED at port2.2 for some time
; jump to second block of memory	; jump to fourth block of memory
ORG 0050H	ORG 0200H
LJMP START1	LJMP START1
START1: MOV R3,#20	START1: MOV R3,#20
LED1: CPL P2.0	LED3: CPL P2.2
LCALL DELAY	LCALL DELAY
DJNZ R3,LED1	DJNZ R3,LED3
LJMP 0100H	LJMP 0300H
DELAY: MOV R0,#02	DELAY: MOV R0,#02
LOOP1: MOV R1,#255	LOOP1: MOV R1,#255
LOOP2: MOV R2,#255	LOOP2: MOV R2,#255
LOOP3: DJNZ R2,LOOP3	LOOP3: DJNZ R2,LOOP3
DJNZ R1,LOOP2	DJNZ R1,LOOP2
DJNZ R0,LOOP1	DJNZ R0,LOOP1
RET	RET
END	END
;Second block of program starts from location ;100h, on/off LED at port2.1 for some time	;Fourth block of program starts from location ;300h, on/off LED at port2.3 for some time
; jump to third block of memory	; jump to first block of memory
ORG 0100H	ORG 0300H

```
         LJMP START1
START1:  MOV R3,#20
LED2:    CPL P2.1
         LCALL DELAY
         DJNZ R3,LED2
         LJMP 0200H
DELAY:   MOV R0,#02
LOOP1:   MOV R1,#255
LOOP2:   MOV R2,#255
LOOP3:   DJNZ R2,LOOP3
         DJNZ R1,LOOP2
         DJNZ R0,LOOP1
         RET
         END
```

```
         LJMP START1
START1:  MOV R3,#20
LED4:    CPL P2.3
         LCALL DELAY
         DJNZ R3,LED4
         LJMP 0050H
DELAY:   MOV R0,#02
LOOP1:   MOV R1,#255
LOOP2:   MOV R2,#255
LOOP3:   DJNZ R2,LOOP3
         DJNZ R1,LOOP2
         DJNZ R0,LOOP1
         RET
         END
```



89V51RD2
Flash Bank:
COM Port: COM 1
Baud Rate: 9600
Interface: None (ISP)
Erase block 0 (0x0000-0xFFFF)
☐ Erase all Flash
☐ Erase blocks used by Hex File
Hex File

Confirm No Erase

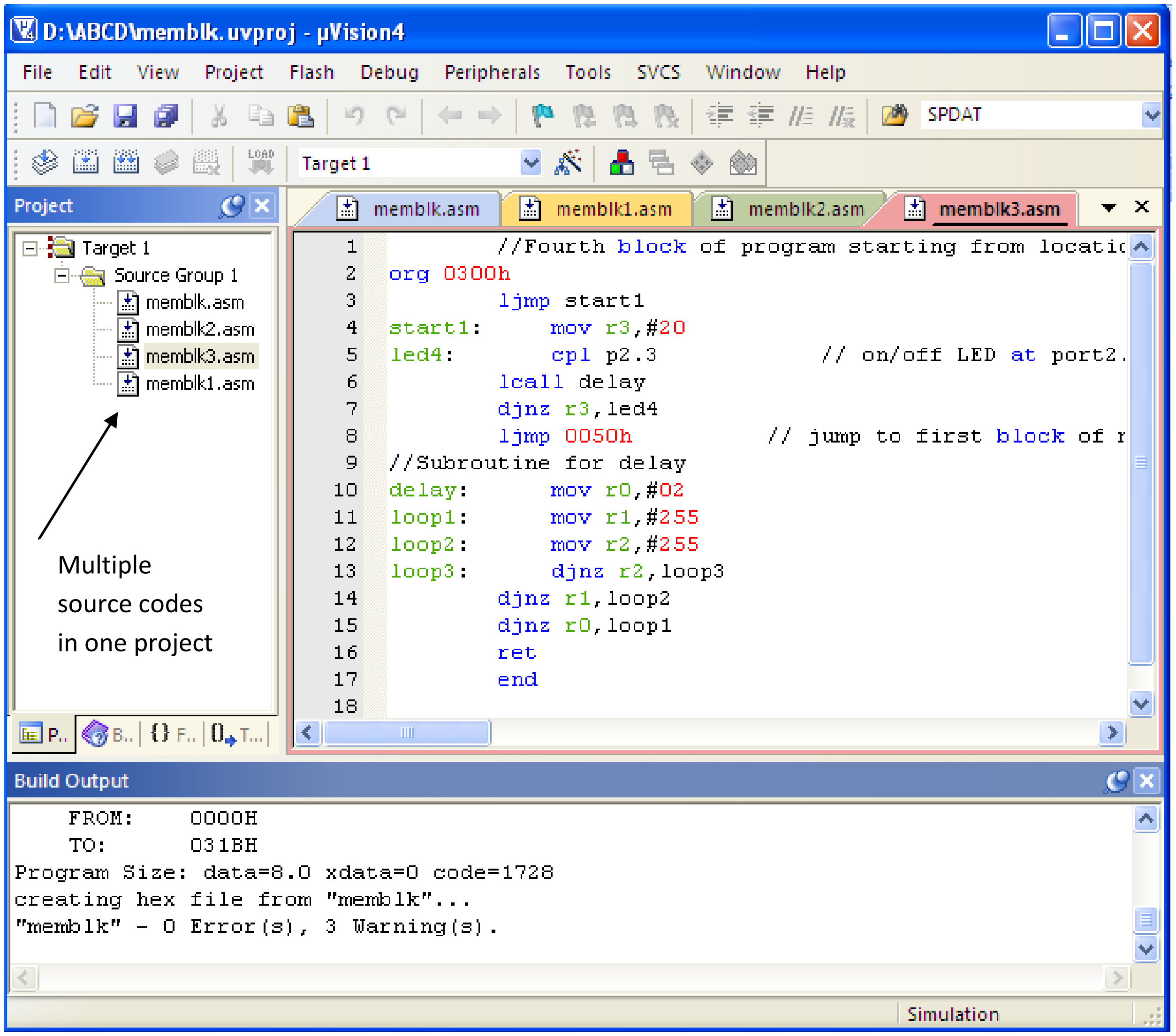
If the flash block or blocks required to store the Hex file are not erased then programming and verifying or checksum generation may fail. Are you sure you want to program the device without erasing blocks first?

☐ Do not show this warning in future

☑ Verify after programming
☐ Set Security Bit 1
☐ Fill unused Flash
☐ Gen block checksums
☐ Execute
☐ Prog Clocks Bit
☐ Activate Flash Bank

Rotating, fully customizable, remotely updated Internet links. Embed them in your application!
www.embeddedhints.com

Figure 8.1 Memory block programming selection in flash magic



```
        //Fourth block of program starting from locatic
org 0300h
        ljmp start1
start1:     mov r3,#20
led4:       cpl p2.3            // on/off LED at port2.
        lcall delay
        djnz r3,led4
        ljmp 0050h          // jump to first block of r
//Subroutine for delay
delay:      mov r0,#02
loop1:      mov r1,#255
loop2:      mov r2,#255
loop3:      djnz r2,loop3
        djnz r1,loop2
        djnz r0,loop1
        ret
        end
```

```
    FROM:     0000H
    TO:       031BH
Program Size: data=8.0 xdata=0 code=1728
creating hex file from "memblk"...
"memblk" - 0 Error(s), 3 Warning(s).
```

Figure 8.2 Memory block programming of controller

8.5 Result: Output at LEDs observed as per programs, LED blinking at port2 of controller, at different port locations as per program. Different source codes are combined into one project as seen in figure 5.2.

Hex file of this project is as displayed under

```
:100050000200537B14B2A012005FDBF902010078AA
:0C0060000279FF7AFFDAFED9FAD8F622206
:100200000202037B14B2A212020FDBF90203007890
:0C0210000279FF7AFFDAFED9FAD8F622254
:100300000203037B14B2A312030FDBF9020050783F
:0C0310000279FF7AFFDAFED9FAD8F622253
:100100000201037B14B2A112010FDBF90202007895
:0C0110000279FF7AFFDAFED9FAD8F622255
:00000001FF
```

Yellow highlight shows start of different memory block of different source codes and green highlight shows same delay routine used in all source codes.

8.6 Conclusion:- Without erasing entire source code a part of it can be altered using memory block programming as discussed. And hence results in saving time of programming for large programs.

8.7 Remarks: Same labels can be used in different source codes though they belong to same project. Common routines e.g. .delay routines for different source codes in same project can also be placed at one location and used by all programs and thus results in memory space saving.

8.8 Reference :-

1. Datasheet 89V51RD2

Experiment 9

Memory Block Erasing Using Flash Magic

9.1 Objective: To achieve block wise erasing of memory for controller using flash magic.

9.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

9.2.2 Hardware Requirement: Target board with P89V51RD2 controller as per circuit given in figure 1.6, Serial Cable with DB9 Connector.

9.3 Procedure:

1. Write source code for LED blinking at port, for different blocks of memory. Use source code of experiment 5.
2. Build/Compile project.
3. At flash programmer, un-check all memory erase options. See figure 9.1.
4. At flash programmer open ISP, select Erase Flash pages. See figure 9.2.
5. Select desired page to be erased (here it is page 4), and then press erase. See figure9.2
6. Program controller with desired code and observe results.

9.4 Source Code: Table 9

```
; Block of program starting from location 200h
; On/off LED at port2.6 for some time
; Jump to different block of memory
                ORG 0200H
                LJMP START1
START1:         MOV R3,#20
LED3:           CPL P2.6
                LCALL DELAY
                DJNZ R3,LED3
                LJMP 0300H
```

```
DELAY:          MOV R0, #02
LOOP1:          MOV R1, #255
LOOP2:          MOV R2, #255
LOOP3:          DJNZ R2, LOOP3
                DJNZ R1, LOOP2
                DJNZ R0, LOOP1
                RET
                END
```

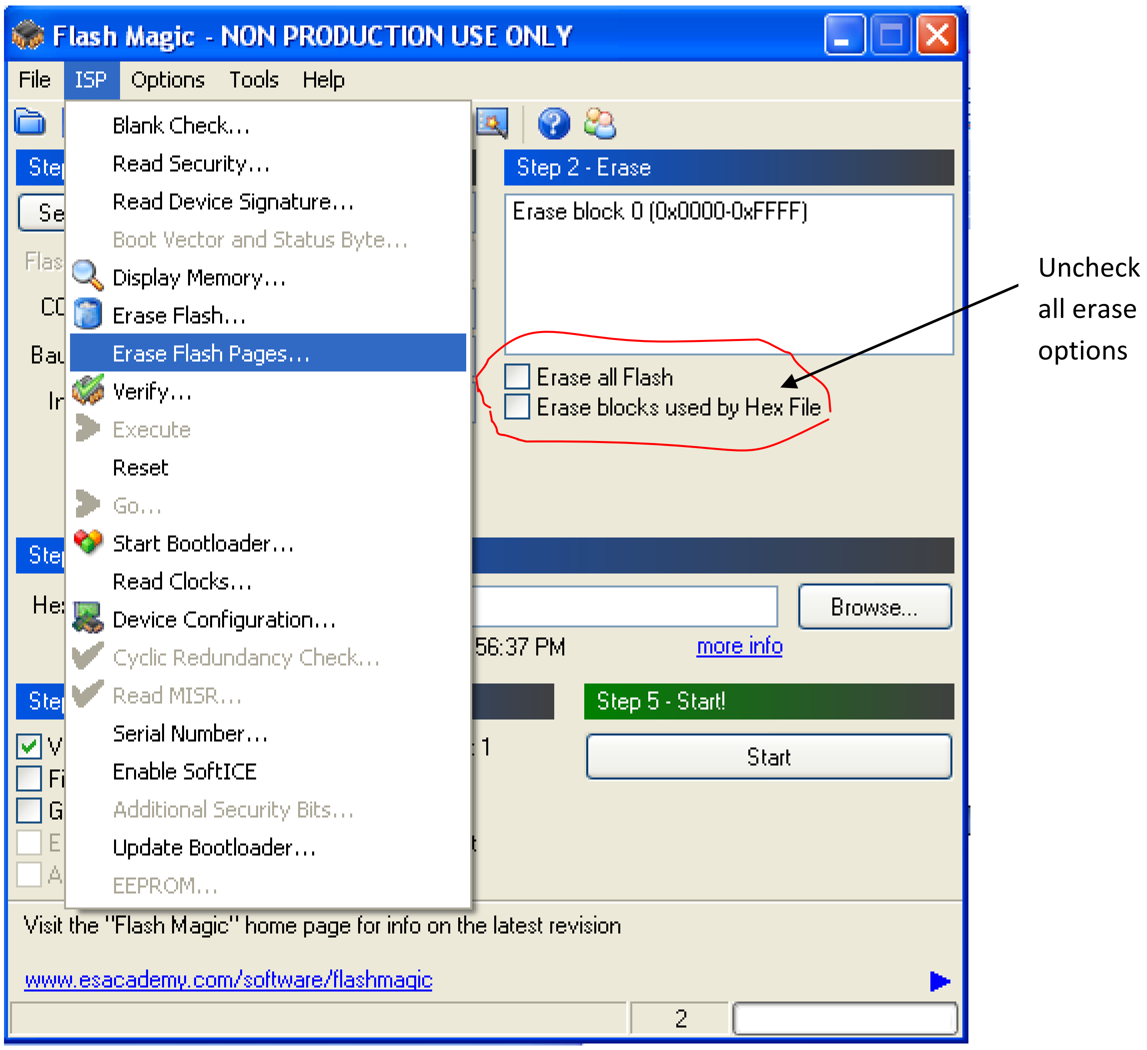
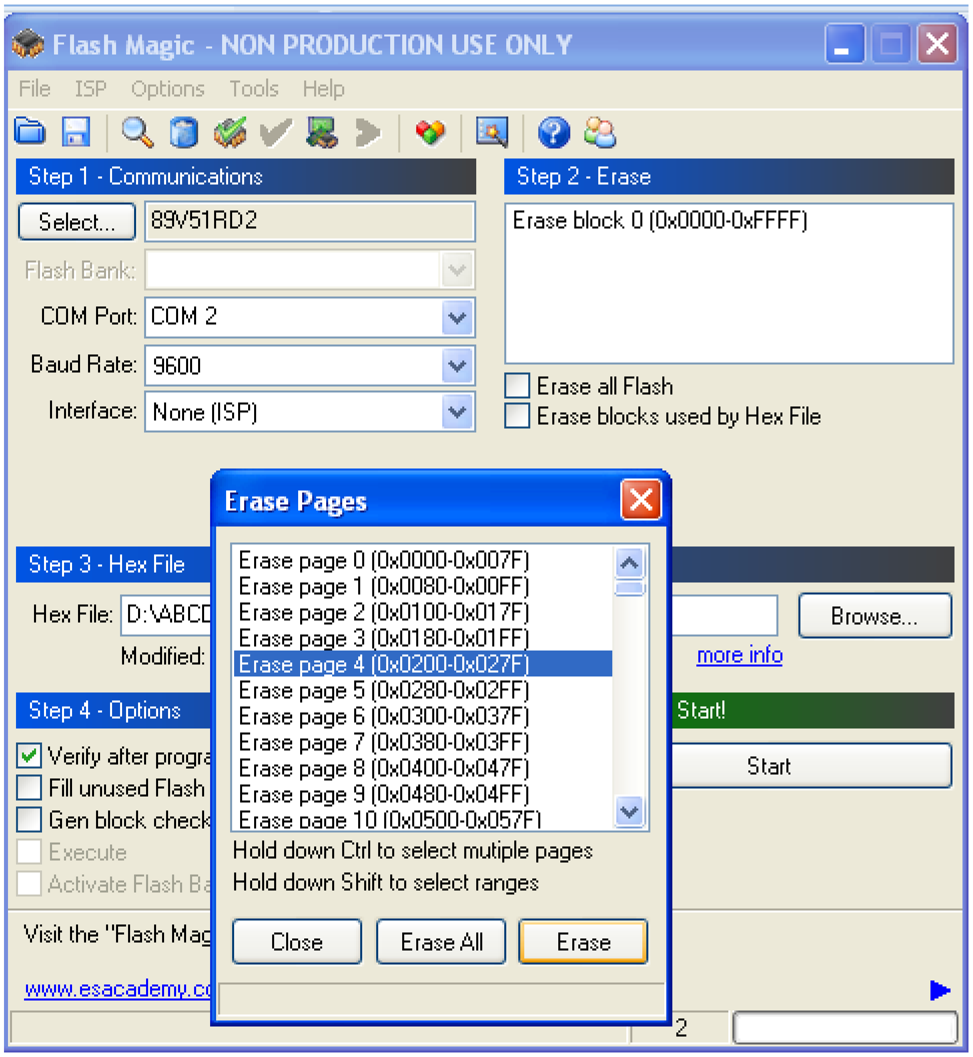


Figure 9.1 Settings for memory block programming



89V51RD2
Flash Bank:
COM Port: COM 2
Baud Rate: 9600
Interface: None (ISP)

Erase block 0 (0x0000-0xFFFF)
Erase all Flash
Erase blocks used by Hex File

Hex File: D:\ABCD
Modified:

✓ Verify after progra
Fill unused Flash
Gen block check
Execute
Activate Flash Ba

Erase page 0 (0x0000-0x007F)
Erase page 1 (0x0080-0x00FF)
Erase page 2 (0x0100-0x017F)
Erase page 3 (0x0180-0x01FF)
Erase page 4 (0x0200-0x027F)
Erase page 5 (0x0280-0x02FF)
Erase page 6 (0x0300-0x037F)
Erase page 7 (0x0380-0x03FF)
Erase page 8 (0x0400-0x047F)
Erase page 9 (0x0480-0x04FF)
Erase page 10 (0x0500-0x057F)
Hold down Ctrl to select mutiple pages
Hold down Shift to select ranges

Figure 9.2 Erase desired page of flash memory

9.5 Result: In source code of experiment 8, only third block is erased using flash page erase and then flash is programmed with new program for LED output at port2.6.
Results observed as per programs, i.e. LED blinking at port2 of controller, at different port locations as per program. LED blinking first at port2.0, then port2.1, and then at port 2.6 instead of port2.2 as per experiment 5 and then at port 2.3.

9.6 Conclusion:- Same as block writing of memory, block erase of memory can also be done for different blocks.

9.7 Remark:- Each memory location cannot be erased or programmed. This can be done only in block defined. Means modifications will be done on entire block of memory and not on single location.

9.8 Reference :- Datasheet 89V51RD2

Experiment 10

Timer in Timer Mode

10.1 Objective: To achieve timer working in timer mode and blink LED without using any loop delay routine.

10.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

10.2.2 Hardware Requirement: Target board with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Serial Cable with DB9 Connector/ USB 89SXX programmer with cable.

10.3 Procedure:

1. Write source code e.g. LED blinking at port.
2. Build/Compile project.
3. Program controller with desired code and observe results.

10.4 Source Code: Table 10

```
;Timer0 is used in timer mode. Runs from
;0000 to ffffh and blink LED at port 2.2, Timer
;0 ISR,Timer0 in mode 1

            ORG 0000H
            LJMP START
            ORG 000BH
START:      MOV TMOD,#01H
            MOV TL0,#00H
            MOV TH0,#00H
            SETB EA
            SETB ET0
            SETB TR0
            LJMP $
```

```
; Timer0 ISR routine

TMR0_INT:   CLR TR0
            CLR TF0
            CPL P2.2
            MOV TL0,#00H
            MOV TH0,#00H
            SETB TR0
            RETI
            END
```

10.5 Result: Results observed as per programs, i.e. one LED blinking at port2.2 of controller, as per program.

10.6 Conclusion:- As timer is used to generate delay, controller is free to complete other tasks while generating delay.

10.7 Remarks:- Using timers different functions can be achieved. E.g. generation of fixed time delay, generation of frequency at, any I/O port etc.
Counter function of timers allow measurement of unknown frequency, event counting etc.

10.8 Reference :- Datasheet 89V51RD2/AT89S52

Experiment 11

Seven segment LED display interfacing

11.1 Objective: To achieve interfacing of seven segment LED display and generate counting from 0 to 99 with fixed time delay.

11.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

11.2.2 Hardware Requirement: Target board with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Display circuit as per figure 11.1, Serial Cable with DB9 Connector/ USB 89SXX programmer with cable.

11.3 Procedure:

1. Connect circuit as per figure 11.1. Connect J1 and J2 between points 2 and 3.Connect point3 of J1 to port1.6 and point3 of J2 to port1.7 of target board controller (figure1.6/4.5).
2. Short CON5 and CON6 and connect them to CON2 of target board controller (figure1.6/4.5).
3. Write desired source code.
4. Build/Compile project and program controller with desired code and observe results.

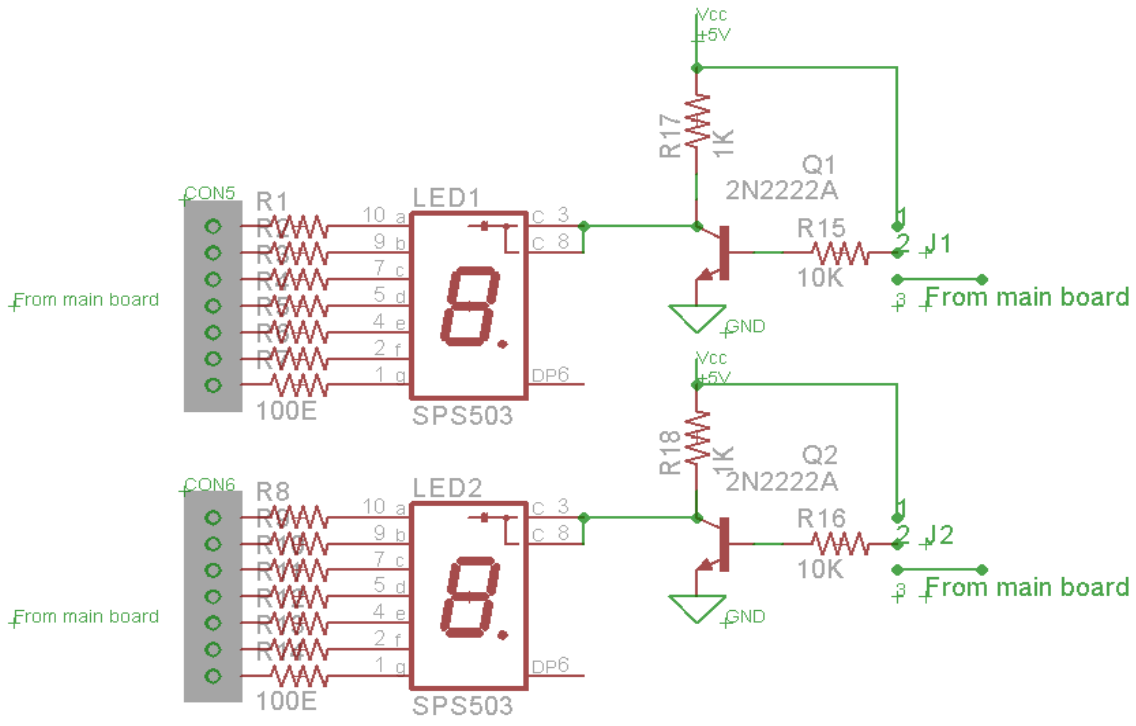


Figure 11.1 Seven Segment LED display connections with controller

11.4 Source Code: Table 11

```
;Only for two seven segment connected at
;port2, ;control pins for common cathode type
;Seven segment is connected at port 1.6 and
;1.7

            ORG 0000H
            LJMP START1
START1:     SETB P1.7
            SETB P1.6
            MOV R5,#00H
            MOV R4,#00H
            MOV R6,#0FFH
NEXTSEG:    MOV R3,#0AH
            MOV DPTR,#1000H
NEXTDIG:    MOV A,R4
            MOVC A,@A+DPTR
            MOV P2,A
            LCALL DELAY
            CLR P1.7
            SETB P1.6
            MOV A,R5
            MOVC A,@A+DPTR
            MOV P2,A
            LCALL DELAY
            CLR P1.6
            SETB P1.7
            DJNZ R6,NEXTDIG
            MOV R6,#0FFH

            INC R4
            DJNZ R3,NEXTDIG
            INC R5
            MOV A,R5
            MOV R4,#00H
            SETB P1.6
            CJNE R5,#0AH,NEXTSEG
            SJMP START1
;DELAY ROUTINE
DELAY:      MOV R0,#02
LOOP1:      MOV R1,#10
LOOP2:      MOV R2,#100
LOOP3:      DJNZ R2,LOOP3
            DJNZ R1,LOOP2
            DJNZ R0,LOOP1
            RET

ORG 1000H
DB          03FH, 06H, 05BH, 04FH, 66H,
            06DH,  07DH, 07H, 07FH, 067H
            END
```

11.5 Result: Two common cathode seven segments LED displays are showing counting starting from 0 to 99 with fixed time delay.

11.6 Conclusion:- Interfacing of seven segment LED displays is achieved.

11.7 Remark:- More than two seven segment LED displays can be connected at same port with different control pins. Alpha-numeric displays can also be connected in similar manner.

11.8 Reference :-

1. Datasheet 89V51RD2/AT89S52
2. http://en.wikipedia.org/wiki/Seven-segment_display

Experiment 12

RS232 Serial Communication between PC and controller

12.1 Objective: To achieve serial communication between PC and controller using serial UART of controller.

12.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer, Hyper terminal at PC (download from http://www.hilgraeve.com/hyperterminal-trial/).

12.2.2 Hardware Requirement: Target board with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Serial Cable with DB9 Connector/ USB 89SXX programmer with cable.

12.3 Procedure:

1. Write desired source code.
2. Build/Compile project.
3. Program controllers with desired code.
4. Close Flash programmer.
5. Connect target board with PC through DB9 serial connector, open hyper terminal for desired communication.
6. Set hyper terminal as shown in figures 12.1 to 12.4. Select Com port, baud rate as 9600, data bits as 8, parity as none, stop bit as 1 and flow control as none.
7. Observe results.

12.4 Source Code: Table 12

```
; This code enables controller to communicate
;with PC at 9600 baud-rate. Data sent from PC
;will be echoed ;back to PC. Timer1 of
;controller is used for generating baud rate.
;Serial interrupt routine resides at 0023h. LED
;at port 2.0 is used for indication. Stack
;pointer set at 08h. Timer1 is used in mode 2
;i.e. auto reload mode.

            ORG 0000H
            JMP 0100H
            ORG 0023H
            CALL SR_INT
            RETI

            ORG 0100H
START:      SETB P2.0
            MOV SP,#08H
            MOV SCON, #50H
            ANL PCON,#7FH
            SETB TR1
AGAIN:      JMP AGAIN

;Serial interrupt routine for checking transit or
;receive. Only if RI flag is set, receive will
;occur else transmit. Receive indication is
;given by LED at port 2.0. Read data from
;SBUF, this is data received from PC, clear RI
;flag and load SBUF with same data to sent
;back to PC

SR_INT:     JNB RI,CHKTX
            CLR P2.0
            MOV A,SBUF
            CLR RI
            MOV SBUF,A
```

```
                CLR TR1
                MOV TMOD, #20H
; Enable interrupt servicing and serial port
;interrupt, Load fdh in timer1 for 9600 baud
;rate. Set 8 bit UART mode and enable
;reception. After then start timer1.
                MOV IE,#90H
                MOV TH1,#0FDH
                MOV TL1,#0FDH
```

```
                RET
;After transmit complete give indication at
;LED connected at port 2.0 and clear transmit
;flag TI

CHKTX:          SETB P2.0
                CLR TI
                RET
                END
```

Hyper Terminal settings will be as described under.

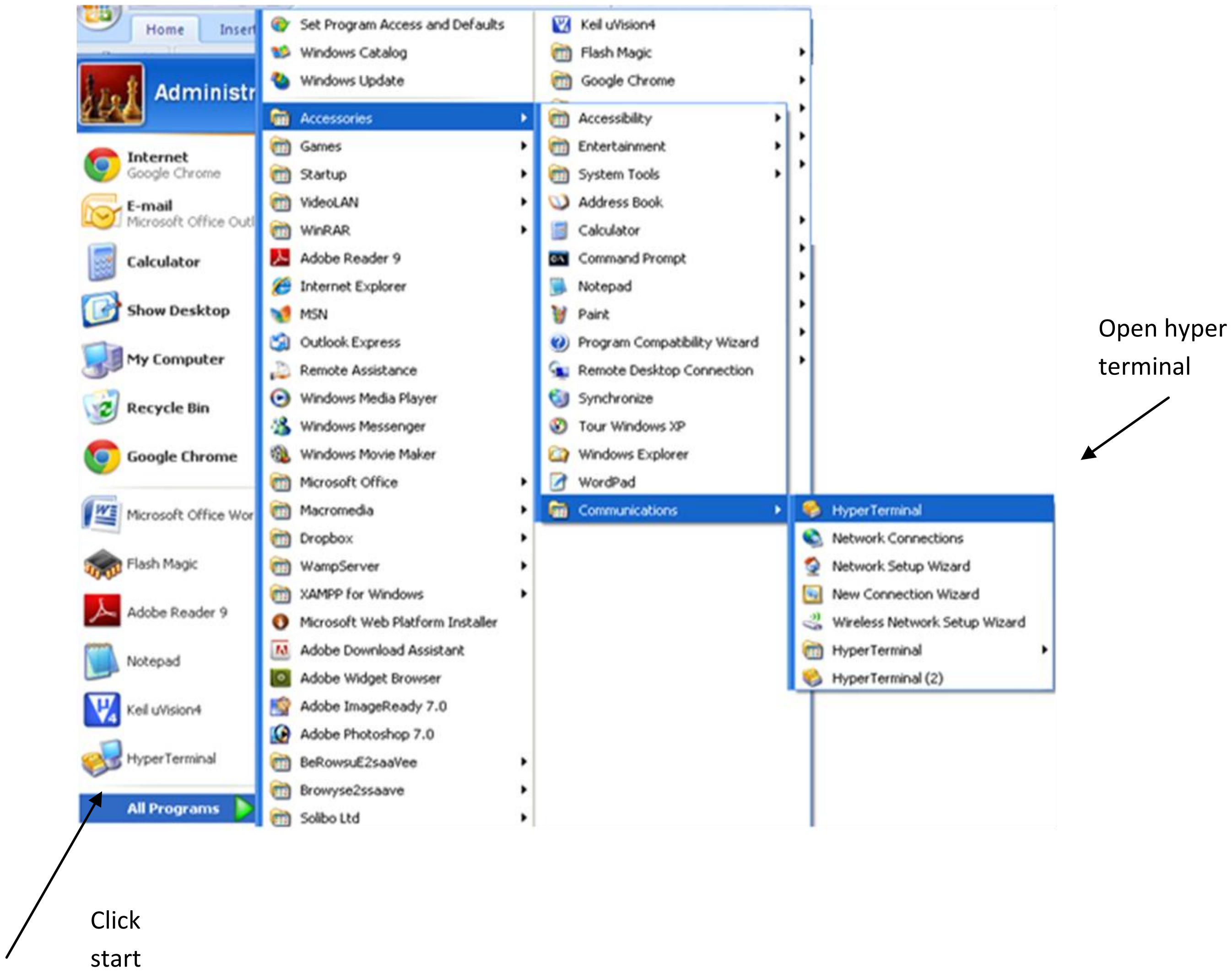
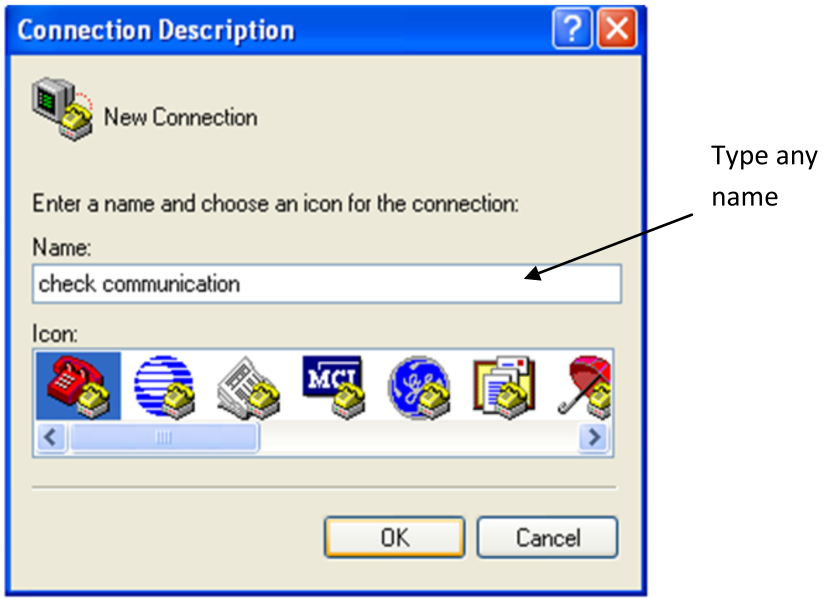


Figure 12.1 Open hyper terminal for checking serial communication between PC and Target board

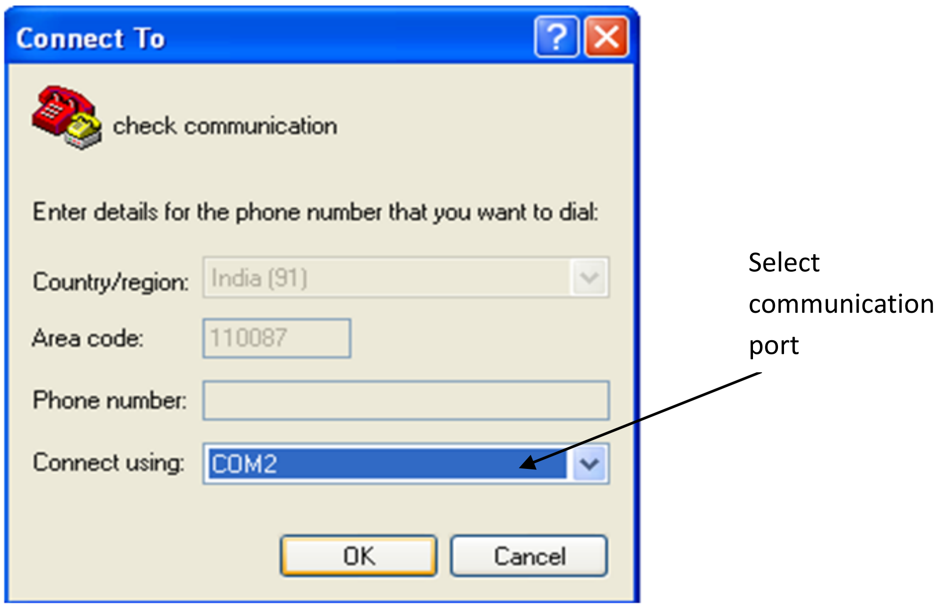

Connection Description

New Connection

Enter a name and choose an icon for the connection:

Name: check communication

Figure 12.2 Name hyper terminal for connections in serial communication


Connect To

check communication

Enter details for the phone number that you want to dial:

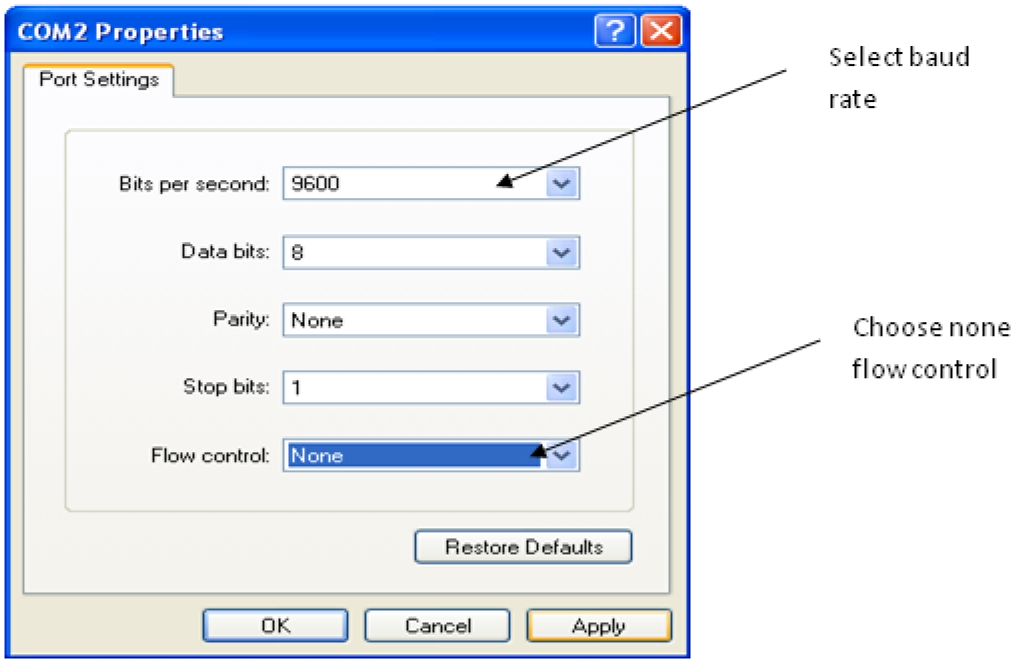
Country/region: India (91)

Area code: 110087

Phone number:

Connect using: COM2

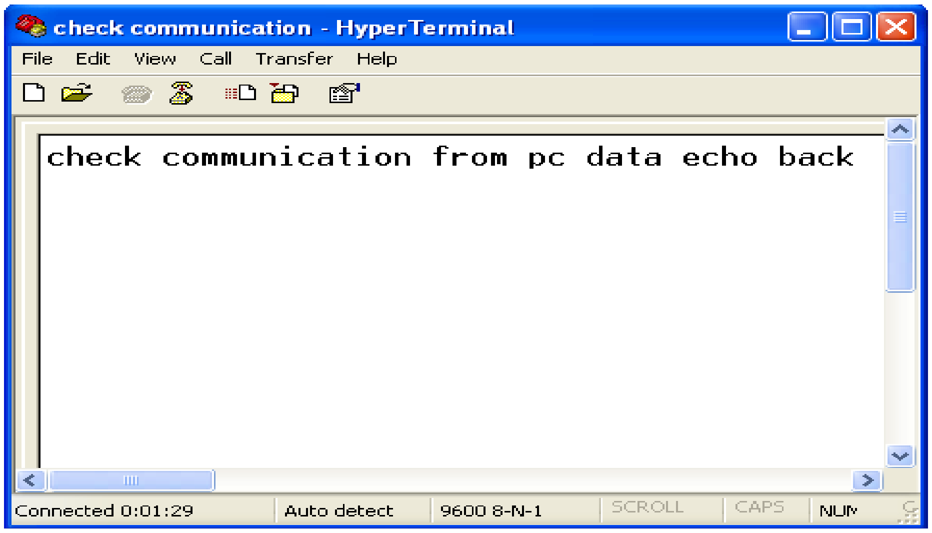
Figure 12.3 Select communication port for checking serial communication



Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

Figure 12.4 Select options for checking serial communication

12.5 Result: Target processor is receiving from PC, and then sent back received data to PC. Results are observed as per figure 12.5.


check communication - HyperTerminal
File Edit View Call Transfer Help

```
check communication from pc data echo back
```

Connected 0:01:29 | Auto detect | 9600 8-N-1 | SCROLL | CAPS | NUM

Figure 12.5 Output received while checking serial communication

12.6 Conclusion:- Controller is communicating with PC using serial UART.

12.7 Remark:- Communication between two controllers can also be achieved using serial UART.

12.8 Reference :-

Datasheet 89V51RD2/AT89S52

http://en.wikipedia.org/wiki/RS-232

http://www.aggsoft.com/rs232-pinout-cable/serial-port-db9.htm

http://www.hilgraeve.com/hyperterminal/

Experiment 13

SPI-Serial Peripheral Interface-Master mode

13.1 Objective: To achieve communication between two controllers using SPI in master mode. Only master will send and Slave will receive.

13.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

13.2.2 Hardware Requirement: Target board with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Display circuit as per figure 11.1, Serial Cable with DB9 Connector/USB 89SXX programmer with cable.

13.3 Procedure:

1. Connect circuit as per figure13.1.
2. In circuit of figure 11.1, connect J1 between 1 and 2, connect CON5 (of figure 11.1 circuit) to CON2 of target board circuit as per figure 1.6/4.5.
3. Write desired source code.
4. Build/Compile project.
5. Program controllers with desired code and observe results.

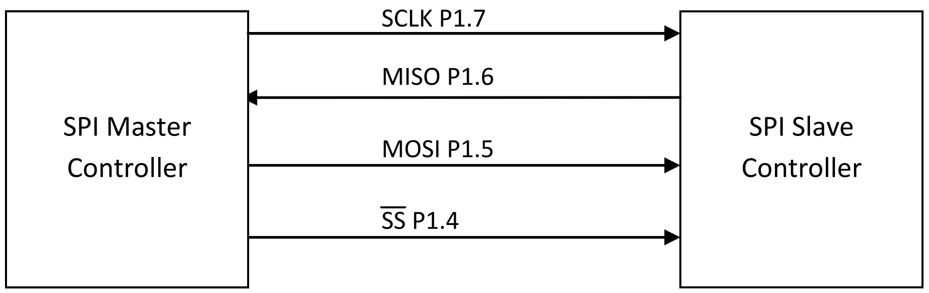


Figure 13.1 SPI connections between two controllers as Master-Slave

13.4.1 Source Code: Table 13A For Master processor

;Processor as master and used only for sending	LOOP:	MOV A,#00H
;data to slave, Fclk Peripheral/128 as baud rate		MOV SPDAT,DATA_EX
;and with slave select pin, P1.6(MISO) serial		JNB TRANSMIT_OK,$
;input, P1.5(MOSI) serial output, P1.4		CLR TRANSMIT_OK
;(SS_bar) Slave select, P1.7 (SCK) Serial		LCALL DELAY
;Clock		INC R5
;Define some RAM locations		CJNE R5,#05H,LP2
		MOV R5,#00H
TRANSMIT_OK BIT 20H.1		MOV DPTR,#1500H

```
SERIAL_DATA       DATA 08H
DATA_SAVE         DATA 09H
DATA_EX           DATA 0AH
;     Define     SPI     Control     Register,
;Configuration/status  register;
;Data register and Interrupt registers

SFR           SPCR      = 0XD5;
SFR           SPSR       = 0XAA;
SFR           SPDAT    = 0X86;
SFR           IEN0       = 0XA8;
SFR           IEN1       = 0XE8;
              ORG 0000H
              LJMP BEGIN
              ORG 004BH
              LJMP IT_SPI
              ORG 0100H
BEGIN:        CLR P2.7
              MOV R5,#00H
              MOV A,#00H
              MOV DPTR,#1500H
              MOVC A,@A+DPTR
              MOV DATA_EX,A
              ORL SPCR,#10h
              SETB P1.4
              ORL SPCR,#83h
              ANL SPCR,#0FFh
              ORL SPCR,#04h
              SETB EA
              SETB ES
              ORL SPCR,#40h
              CLR TRANSMIT_OK
```

```
              MOVC A,@A+DPTR
              MOV DATA_EX,A
LP2:          MOV A,R5
              MOVC A,@A+DPTR
              MOV DATA_EX,A
              LJMP LOOP
; Interrupt routine, interrupt at address 0x004B

IT_SPI:       CPL P2.7
              MOV R7,SPSR
              MOV ACC,R7
              JNB ACC.7,BREAK1
              SETB TRANSMIT_OK
              MOV SPSR, #00H
BREAK1:       JNB ACC.6, BREAK2
BREAK2:       RETI

;Delay routine
DELAY:        MOV R0,#20
LOOP1:        MOV R1,#255
LOOP2:        MOV R2,#255
LOOP3:        DJNZ R2,LOOP3
              DJNZ R1,LOOP2
              DJNZ R0,LOOP1
              RET

              ORG 1500H
DB 6DH, 38H, 77H, 3EH, 79H
              END
```

13.4.2 Source Code: Table 13B For Slave processor

```
;Processor as slave and used only for receiving
;data from master, Fclk Peripheral/128 as baud
;rate and with slave select pin,  P1.6(MISO)
;serial  input,  P1.5(MOSI)  serial  output,  P1.4
;(SS_bar)  Slave  select,   P1.7  (SCK)  Serial
;Clock
;Define some RAM locations

TRANSMIT_OK      BIT 20H.1
SERIAL_DATA      DATA 08H
```

```
              ORG 0100H
BEGIN:        MOV DATA_EX,#55h
              CLR P1.4
              ORL SPCR,#83h
              ANL SPCR,#0FFh
              ORL SPCR,#04h
              SETB EA
              SETB ES
              ORL SPCR,#40h
              CLR TRANSMIT_OK
```

```
DATA_SAVE         DATA 09H
DATA_EX           DATA 0AH
; Define SPI Control Register,
;Configuration/status register;
;Data register and Interrupt registers

SFR          SPCR     = 0XD5;
SFR          SPSR     = 0XAA;
SFR          SPDAT    = 0X86;
SFR          IEN0     = 0XA8;
SFR          IEN1     = 0XE8;
             ORG 000H
             LJMP BEGIN
             ORG 4BH
             LJMP IT_SPI
```

```
LOOP:        JNB TRANSMIT_OK,$
             CLR TRANSMIT_OK
             LJMP LOOP

; Interrupt routine, interrupt at address 0x004B
IT_SPI:      MOV R7,SPSR
             MOV ACC,R7
             JNB ACC.7,BREAK1
             MOV P2,SPDAT
             SETB TRANSMIT_OK
             MOV SPSR, #00H
BREAK1:      JNB ACC.6, BREAK2
BREAK2:      RETI

             END
```

13.5 Result: Slave processor is showing data as received from master processor. Seven segment LED display connected at Port 2 of slave controller shows alphabets SLAUE one by one repeatedly. While one LED connected at Port2.7 of master process toggles every time it transmits.

13.6 Conclusion:- SPI interfacing is achieved where master controls the slave.

13.7 Remark:- A master can control a number of slaves using slave select. Same source code of master and slaves can be used for communication between one master and four slaves with changes only in slave select.

13.8 References:-

1. 89V51Rd2 datasheet/AT89S52
2. *http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus*
3. *http://www.ecse.rpi.edu/courses/CStudio/Silabs/Appnotes/AN028.pdf*

NOTE:- Program written here are as per P89V51RD2. Please refer Atmel datasheet for any changes in SFRs used for SPI.

Experiment 14

SPI-Serial Peripheral Interface-Master/Slave mode

14.1 Objective: To achieve communication between two controllers using SPI in master and slave mode. Master will send, Slave will receive and vice versa.

14.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer.

14.2.2 Hardware Requirement: Target board with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Display circuit as per figure 11.1, Serial Cable with DB9 Connector/USB 89SXX Programmer with cable.

14.3 Procedure:

1. Connect circuit as per figure13.1.
2. In circuit of figure 11.1, connect J1 between 1 and 2, connect CON5 (of figure 11.1 circuit) to CON2 of target board circuit as per figure 1.6/4.5
3. Write desired source code.
4. Build/Compile project.
5. Program controllers with desired codes and observe results.

14.4.1 Source Code: Table 14A For Master processor

```
;Processor as master and used  for sending and
;receiving data to and from slave, Fclk
;Peripheral/128 as baud rate and with pin,
;P1.6(MISO) serial input, P1.5(MOSI) serial
;output, P1.4 (SS_bar) Slave select,  P1.7
;(SCK) Serial Clock
;Define some RAM locations

TRANSMIT_OK      BIT 20H.1
SERIAL_DATA      DATA 08H
DATA_SAVE        DATA 09H
DATA_EX          DATA 0AH
; Define SPI Control Register,
;Configuration/status  register
;Data register and Interrupt registers

SFR          SPCR     = 0XD5;
SFR          SPSR     = 0XAA;
SFR          SPDAT    = 0X86;
SFR          IEN0     = 0XA8;
SFR          IEN1     = 0XE8;
             ORG 000H
```

```
LOOP:        MOV A,#00H
             MOV SPDAT,DATA_EX
             JNB TRANSMIT_OK,$
             CLR TRANSMIT_OK
             LCALL DELAY
             INC R5
             CJNE R5,#05H,LP2
             MOV R5,#00H
             MOV DPTR,#1500H
             MOVC A,@A+DPTR
             MOV DATA_EX,A

LP2:         MOV A,R5
             MOVC A,@A+DPTR
             MOV DATA_EX,A
             LJMP LOOP
; Interrupt routine, interrupt at address 0x004B

IT_SPI:      CPL P1.4
             MOV R7,SPSR
             MOV ACC,R7
             JNB ACC.7,BREAK1
```

```
            LJMP BEGIN
            ORG 4BH
            LJMP IT_SPI
            ORG 0100H
BEGIN:      CLR P2.7
            MOV R5,#00H
            MOV A,#00H
            MOV DPTR,#1500H
            MOVC A,@A+DPTR
            MOV DATA_EX,A
            MOV SPCR,#0D7h
            SETB P1.4
            SETB EA
            SETB ES
            CLR TRANSMIT_OK
```

```
            MOV P2,SPDAT
            SETB TRANSMIT_OK
            MOV SPSR, #00H
BREAK1:     JNB ACC.6, BREAK2
BREAK2:     RETI
;Delay routine
DELAY:      MOV R0,#20
LOOP1:      MOV R1,#255
LOOP2:      MOV R2,#255
LOOP3:      DJNZ R2,LOOP3
            DJNZ R1,LOOP2
            DJNZ R0,LOOP1
            RET

            ORG 1500H
DB 6DH, 38H, 77H, 3EH, 79H
            END
```

14.4.2 Source Code: Table 14B For Slave processor

```
;Processor as slave and used for sending and
;receiving data to and from master, Fclk
;Peripheral/128 as baud rate and with pin,
;P1.6(MISO) serial input, P1.5(MOSI) serial
;output, P1.4 (SS_bar) Slave select, P1.7
;(SCK) Serial Clock
;Define some RAM locations
TRANSMIT_OK     BIT 20H.1
SERIAL_DATA     DATA 08H
DATA_SAVE       DATA 09H
DATA_EX         DATA 0AH
; Define SPI Control Register,
;Configuration/status register;
;Data register and Interrupt registers

SFR         SPCR    = 0XD5;
SFR         SPSR    = 0XAA;
SFR         SPDAT   = 0X86;
SFR         IEN0    = 0XA8;
SFR         IEN1    = 0XE8;
            ORG 000H
            LJMP BEGIN
            ORG 4BH
            LJMP IT_SPI
```

```
            ORG 0100H
BEGIN:      MOV DATA_EX,#0Fh
            MOV SPCR,#0C7h
            CLR P1.4
            SETB EA
            SETB ES
            CLR TRANSMIT_OK
LOOP:       MOV SPDAT, DATA_EX
            JNB TRANSMIT_OK,$
            CLR TRANSMIT_OK
            MOV A,DATA_EX
            CPL A
            MOV DATA_EX,A
            LJMP LOOP
; Interrupt routine, interrupt at address 0x004B
IT_SPI:     CPL P1.4
            MOV R7,SPSR
            MOV ACC,R7
            JNB ACC.7,BREAK1
            MOV P2,SPDAT
            SETB TRANSMIT_OK
            MOV SPSR, #00H
BREAK1:     JNB ACC.6, BREAK2
BREAK2:     RETI
            END
```

14.5 Result: Slave processor is showing data as received from master processor. Seven segment LED display connected at Port 2 of slave controller show alphabets SLAUE one by one repeatedly. While eight LEDs connected at Port2 of master process shows 4 LEDs on/off every time it receives data from slave after it transmits.

14.6 Conclusion:- Bidirectional communication between master and slave controllers using SPI interface is achieved. It shows that either of the two controller can work as master/slave.

14.7 Remark:- Data communicated between two controllers can be stored in internal /external memory block. Any of the participating controller can control other controller's operations.

14.8 References:-

1. 89V51Rd2 datasheet/ AT89S52
2. *http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus*
3. *http://www.ecse.rpi.edu/courses/CStudio/Silabs/Appnotes/AN028.pdf*

NOTE:- Program written here are as per P89V51RD2. Please refer Atmel datasheet for any changes in SFRs used for SPI.

Experiment 15

Interfacing of 16x2 LCD with microcontroller

15.1 Objective: To achieve interfacing of 16x2 LCD and print some welcome message on it.

15.2.1 Software Requirement: Editor like Keil µvision ver 4 or less, Flash programmer

15.2.2 Hardware Requirement: Target board with P89V51RD2/AT89S52 controller as per circuit given in figure 1.6/4.5, Serial Cable with DB9 Connector/USB 89SXX programmer with cable, 16x2 LCD , 10K POT, connecting wires.

15.3 Procedure:

1. Connect LCD with target board (figure 1.6/4.5) as per figure 15.1.
2. Write desired source code as per table 15. Refer figure 15.2.
3. Build/Compile project.
4. Program controllers with desired code.
5. Close Flash programmer
6. Observe results.

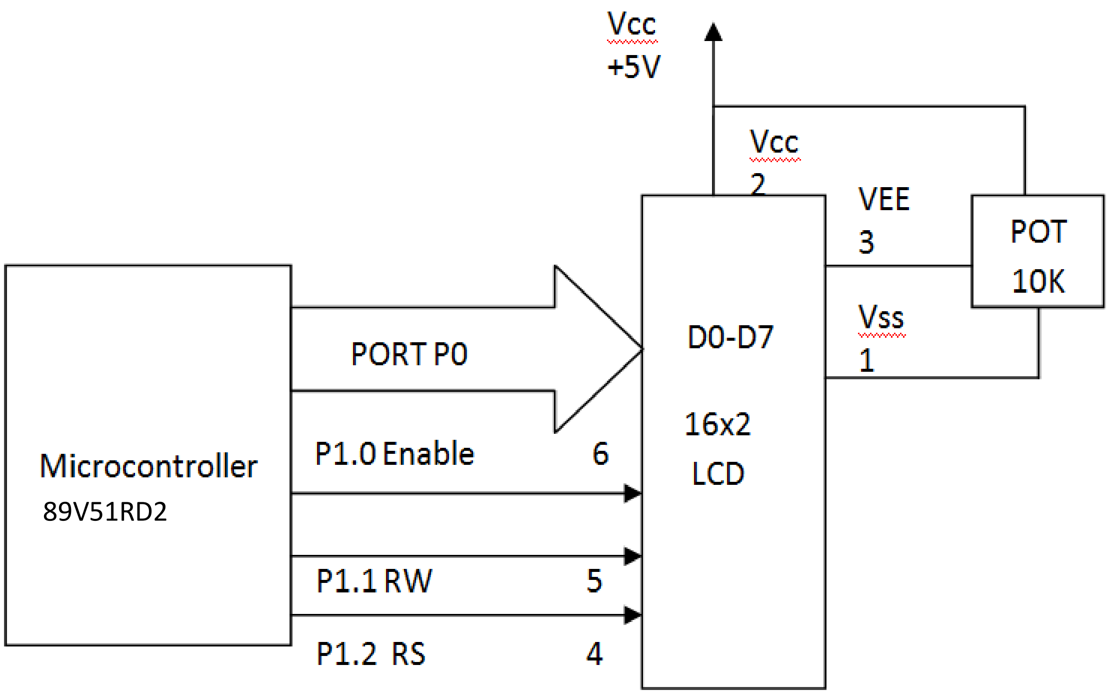
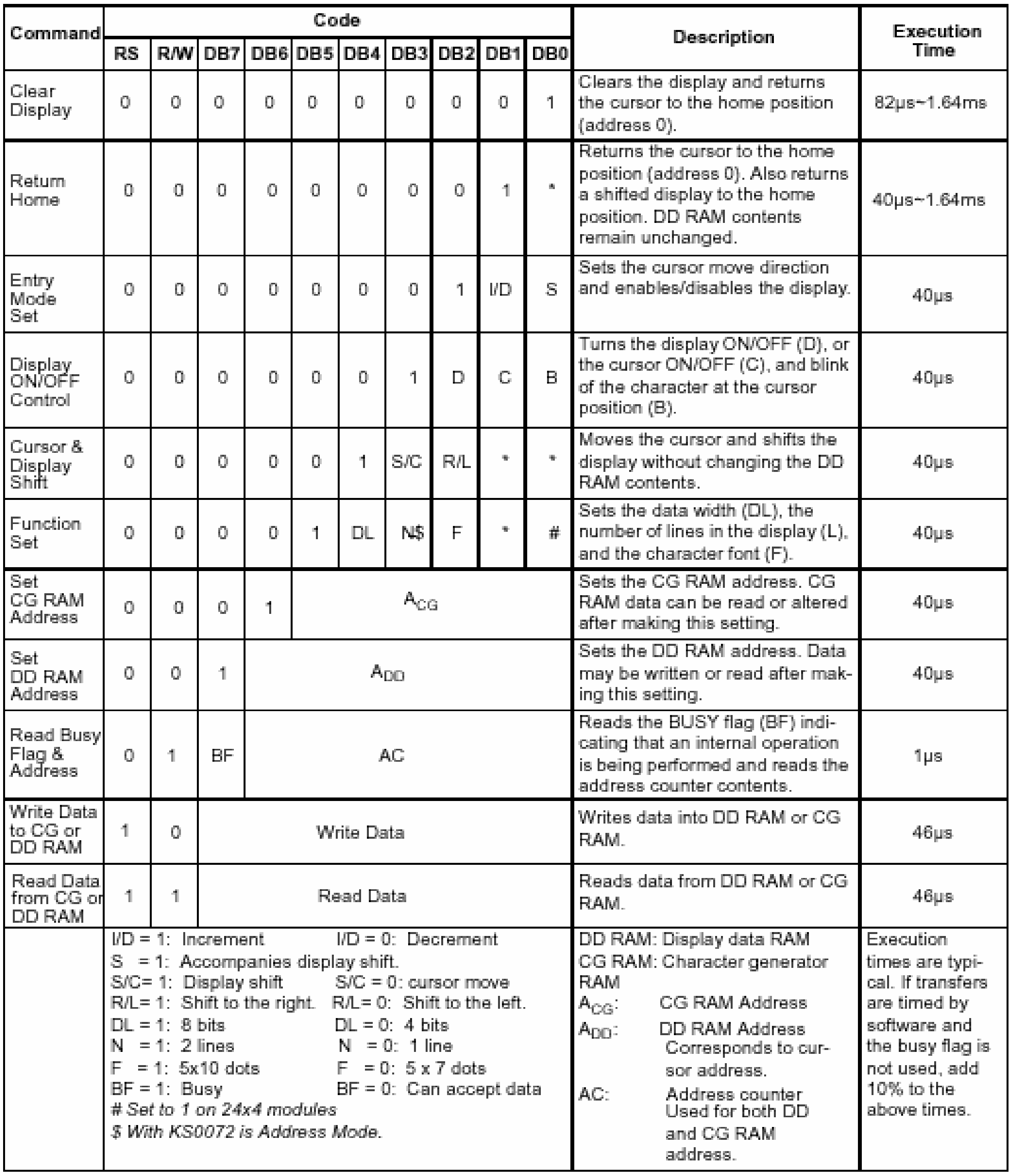


Figure 15.1 16x2 LCD module connections with microcontroller



Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged.	40μs~1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and enables/disables the display.	40μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B).	40μs
Cursor & Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing the DD RAM contents.	40μs
Function Set	0	0	0	0	1	DL	N$	F	*	#	Sets the data width (DL), the number of lines in the display (L), and the character font (F).	40μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data can be read or altered after making this setting.	40μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. Data may be written or read after making this setting.	40μs
Read Busy Flag & Address	0	1	BF	AC							Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents.	1μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	46μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	46μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift. S/C= 1: Display shift S/C = 0: cursor move R/L= 1: Shift to the right. R/L= 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5x10 dots F = 0: 5 x 7 dots BF = 1: Busy BF = 0: Can accept data *# Set to 1 on 24x4 modules* *$ With KS0072 is Address Mode.*										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address.	Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times.

Figure 15.2 LCD module commands and instruction set

15.4 Source Code: Table 15

```
; This code enables controller to send data on
;LCD. Port 0 is used for data bus, P1.0 for
;enable, P1.1. for RW and P1.2 for RS.

            ORG 0000H
            LJMP START
;subroutine to display welcome, use ASCII
;values corresponding to alphabets
START:      LCALL LCDINT
WELCOME:    MOV A,#20H
            LCALL DISDT1
            MOV A,#57H
            LCALL DISDT1
            MOV A,#45H
            LCALL DISDT1
            MOV A,#4CH
            LCALL DISDT1
            MOV A,#43H
            LCALL DISDT1
            MOV A,#4FH
            LCALL DISDT1
            MOV A,#4DH
            LCALL DISDT1
            MOV A,#45H
            LCALL DISDT1
            JMP  $

//subroutine for display data on LCD

DISDT1:     SETB P1.2
            SETB P1.0
            NOP
            NOP
            MOV P0,A
            LCALL DELAY
            CLR P1.0
            CLR P1.2
            LCALL DELAY1
            RET

;one time usable routine for LCD initialization

LCDINT:     MOV P1, #00H
            SETB P1.0
```

```
;SET ENTRY MODE
            SETB P1.0
            NOP
            NOP
            MOV P0,#06H
            NOP
            CLR P1.0
            LCALL DELAY
;SET DISPLAY ON
            SETB P1.0
            NOP
            NOP
            MOV P0,#0CH
            NOP
            CLR P1.0
            LCALL DELAY
;SET CURSOR AND DISPLAY SHIFT
            SETB P1.0
            NOP
            NOP
            MOV P0,#01CH
            NOP
            CLR P1.0
            LCALL DELAY
;SET FUNCTION SET
            SETB P1.0
            NOP
            NOP
            MOV P0,#038H
            NOP
            CLR P1.0
            LCALL DELAY
            RET

//Subroutine for delay used in LCD delay
DELAY:      MOV R0,# 09
LOOP1:      DJNZ R0,LOOP1
            RET

//Subroutine for delay used in display
DELAY1:     MOV R0,#01
LOOP11:     MOV R1,#0100
LOOP12:     MOV R2,#0255
```

```
            NOP                     LOOP33:     DJNZ R2,LOOP33
            NOP                                 DJNZ R1,LOOP12
            MOV P0,#01H                         DJNZ R0,LOOP11
            NOP                                 RET
            CLR P1.0                            END
            LCALL DELAY
```

15.5 Result: 16x2 LCD interfaced with controller is showing message as ‘WELCOME’ on its first line as per the source code.

15.6 Conclusion:- LCD is interfaced with controller and is showing messages.

15.7 Remark:- With the same procedure 16x4, 20x2, 20x4, 40x4 etc LCD can also be interfaced. Here care must be taken in selecting LCD display segment address. Through proper segment address one can display data anywhere on LCD display panel.
Bidirectional communication between LCD and controller can also be done, where one can use LCD memory space, can create own characters using CGRAM data space of LCD eg hindi character set.

15.8 References:-

1. Datasheet 89v51RD2/AT89S52
2. *https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf*

Design & Development of a mobile design using Raspberry Pi : A Practical Approach

(Experimental Manual for M.Tech Students)

for SoC and mobile design(Version 1, 2013-14)

with support of MHRD and NOKIA projects

Designed & Developed By: Ms. Nidhi Agarwal

Under the Guidance of: Dr. SRN Reddy, Associate Professor, CSE

Computer Science & Engineering Department

Indira Gandhi Delhi Technical University for Women

Kashmere Gate, Delhi-110006

Appendix A-List of Experiments

2nd Semester M.TECH (MPC)

L P C

0 2 1

Paper Code: MMC-520
Paper Title: Embedded System Design Based on ARM/Atmel Lab

1. Introduction to programming tool chain for Embedded Application Development Environment (IDE) i.e. KEIL µversion4 and Flash Magic.
2. Design and develop a re-programmable embedded computer using 8051 architecture. Give its schematic diagram and explain the function of each block.
3. Explain the procedure for Multiple Processor programming with Flash Magic & draw its schematic.
4. Write a program to interface LEDs at Input/ Output Ports, write delay routine and generate 10 different patterns with specified delay.
5. Use of I/O pins for data transfer between two controllers and perform Master and Slave Communication.
6. Write a program to use different Memory Block for different purpose.(e.g. each block for different LED patterns). Use switches for selecting different LED patterns.
7. Write a program to erase a particular memory block (as in experiment no. 5) and re-write with some other LED pattern code. Use switches for selecting different LED patterns.
8. Write a program to use of Timers 0/1 in Timer Mode and generate delay and use this delay to blink LED as per experiment no.3. Also calculate the delay.
9. Write a program to interface seven segment display
    a) Display the numerals from 0-9 at regular interval.
    b) Display one character of your name at a time at regular interval
10. Write a program to communicate between PC and Controller using RS232 interface at 9600 baud rate using Window XP hyper-terminal application.
11. Write a program to communicate between two controllers using SPI (Serial Peripheral Interface) and draw its schematic.
12. Write a program to print Hello World/ Your name using Intel Atom board editor and check the output on the terminal.
13. Write a program to display MAC address of Intel Atom Board on the terminal and initialize the serial port.
14. Study the architecture and peripherals of MBED ARM development board and write a program to display various system parameters.
15. Write a program to interface buzzer and LEDs at I/O ports of ARM MBED board.
16. Write a program to interface LCD to ARM board and display moving message.
17. Write a program to generate variable duty cycle of Pulse Width Modulation (PWM) using ARM board. Select duty cycle using 8 switches D/P switch.
18. Minor project based on 8051/ATMEGA/ARM/ATOM. Choose a project as per your choice it should include at least five interfaces from following list-

External interrupt	Timer/Counter	Sensor: Motion/IR/Sound/ Humidity/Pressure/Temperature	ADC
LCD	4 X 4 keyboard	RS232 Interface/I2C/SPI	GSM Module
Relay	Qwerty Keyboard	RTC	Bluetooth Module