ASSEMBLY & C - PROGRAMMING WITH AT89S52 (8051)

Workshop Manual

TEAM FALCON

NAME:

SEMESTER:

Contents

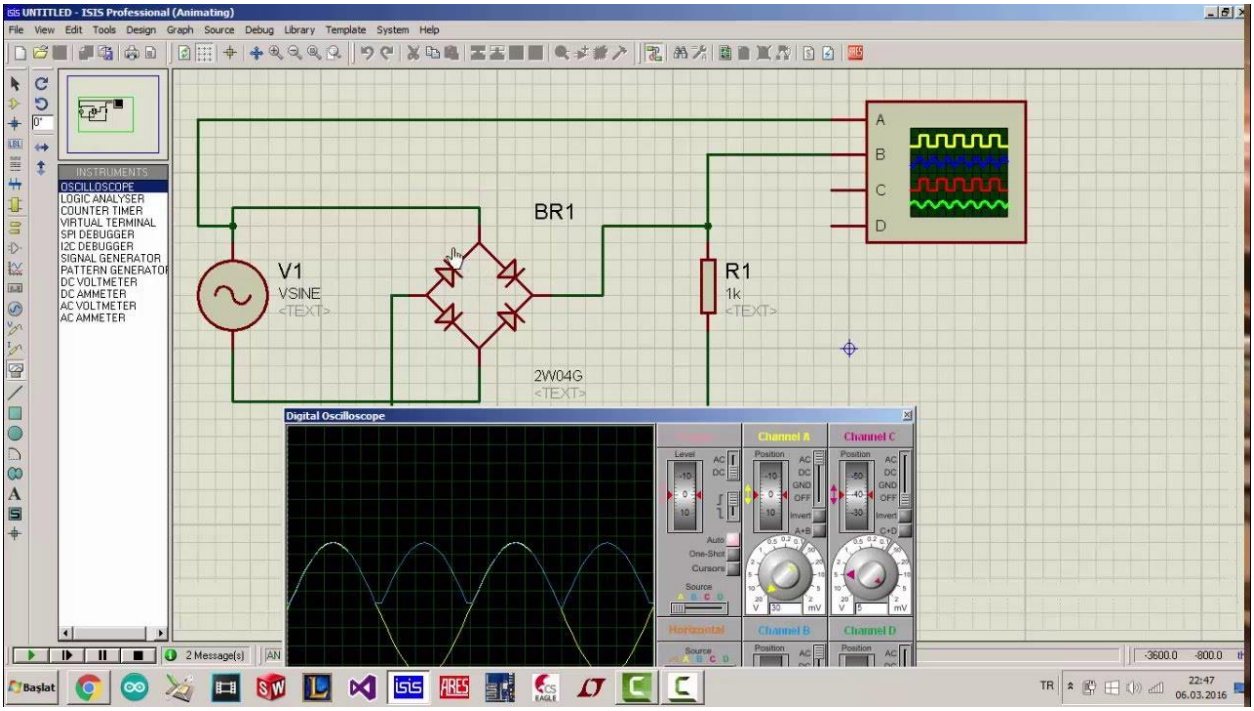
Topic	*Page*
Proteus	*2*
Keil	*6*
Intel-8051	*9*
8051 I/O Programming	*11*
Assembly Programming	*12*
C Programming	*15*
Timers	*18*
Mini-Project	*23*

Proteus

Proteus Design Suite, is an **EDA** tool used by professionals and students for Schematic development and PCB design.

The major advantage of this simulator is it provides a variety of MCU for simulation via coding. This feature reduces testing time on the hardware design.

For further information visit https://www.labcenter.com/



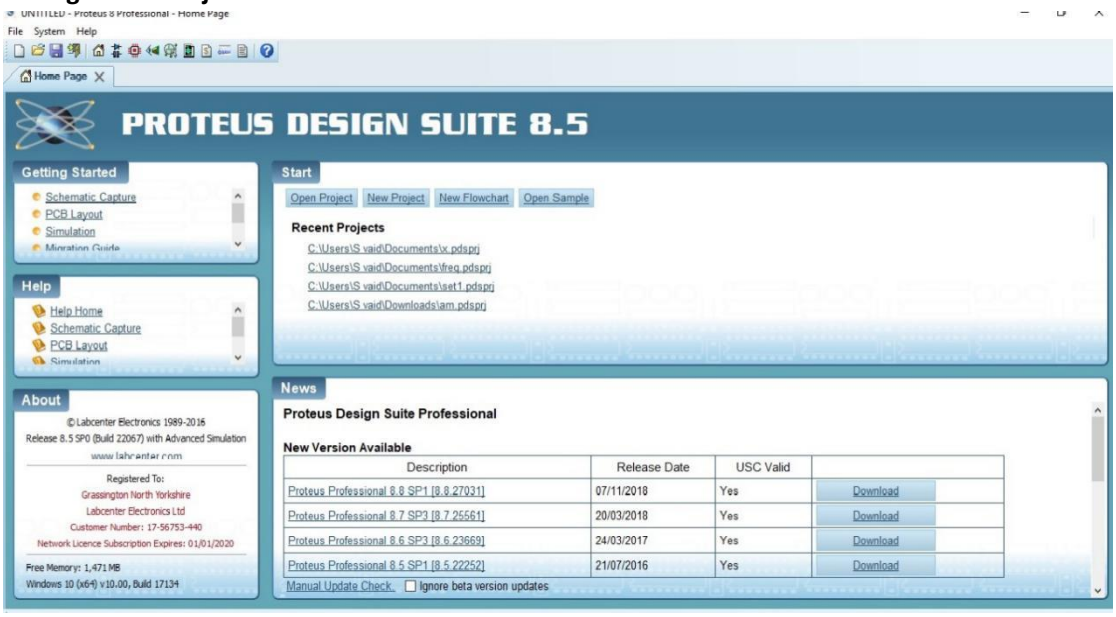
Offers coding environment in Assemble, C/C++, Energia ide, Arduino ide and many more.

Supports many Processors from Atmel, Ti, and Intel.

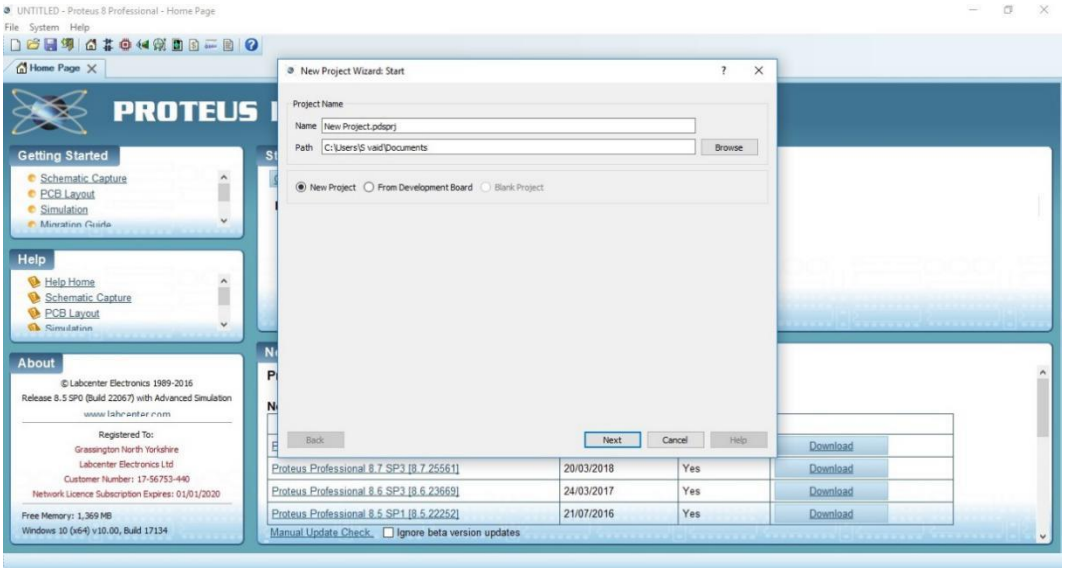
In the upcoming Assembly coding and Embedded C coding of 8051, we extensively use Proteus Design Suite professional edition 8. This helps you to render the simulation of circuits in short time. Make alterations and the final working code can be incorporated in the 8051.

Working with Proteus

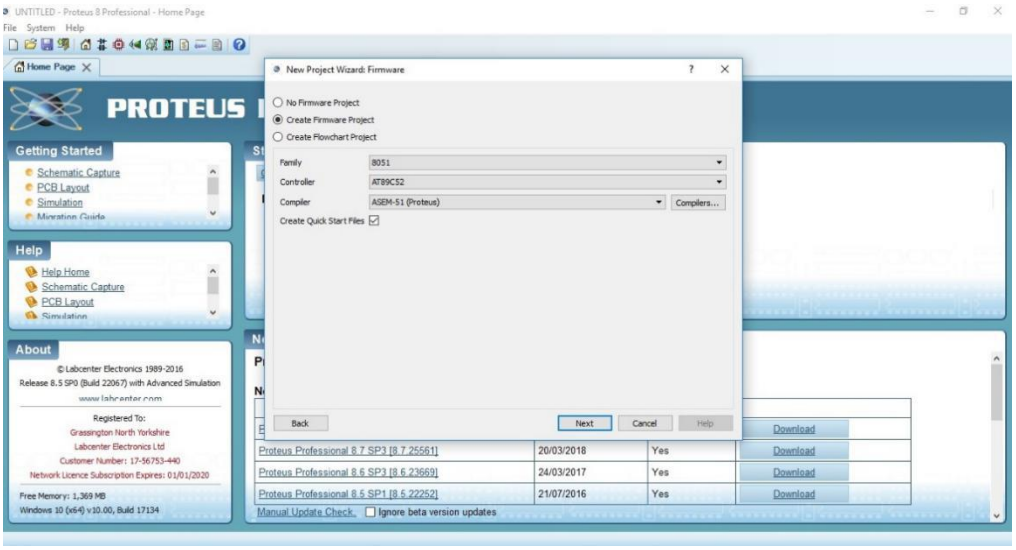
1. Creating a new Project



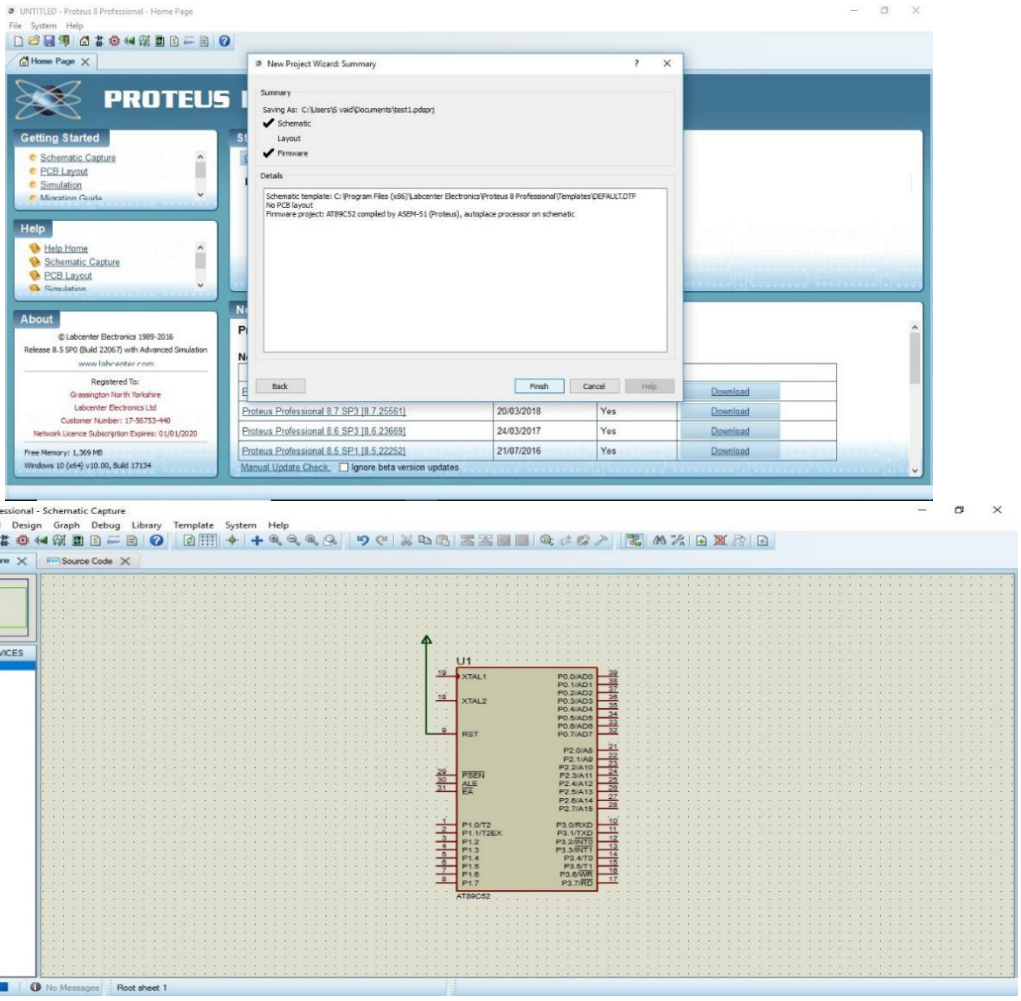
2. Enter the **Name and Project file path**.



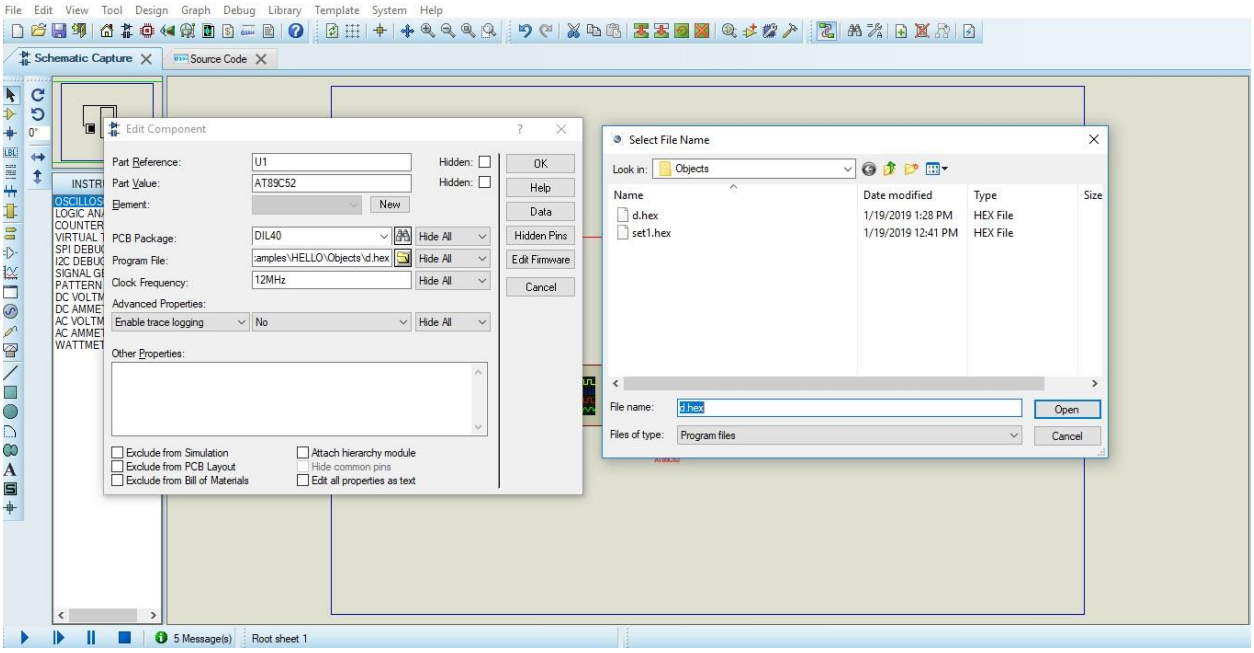
3. Selection of **Template size**, (Default size can be picked).
4. Based on preference **PCB firmware template** can be chosen.
5. For 8051, select create a Firmware project.
   Family: 8051.
   Controller: AT89C52 (Similar version of **AT89S52**, board provided).
   Complier: ASEM-51.



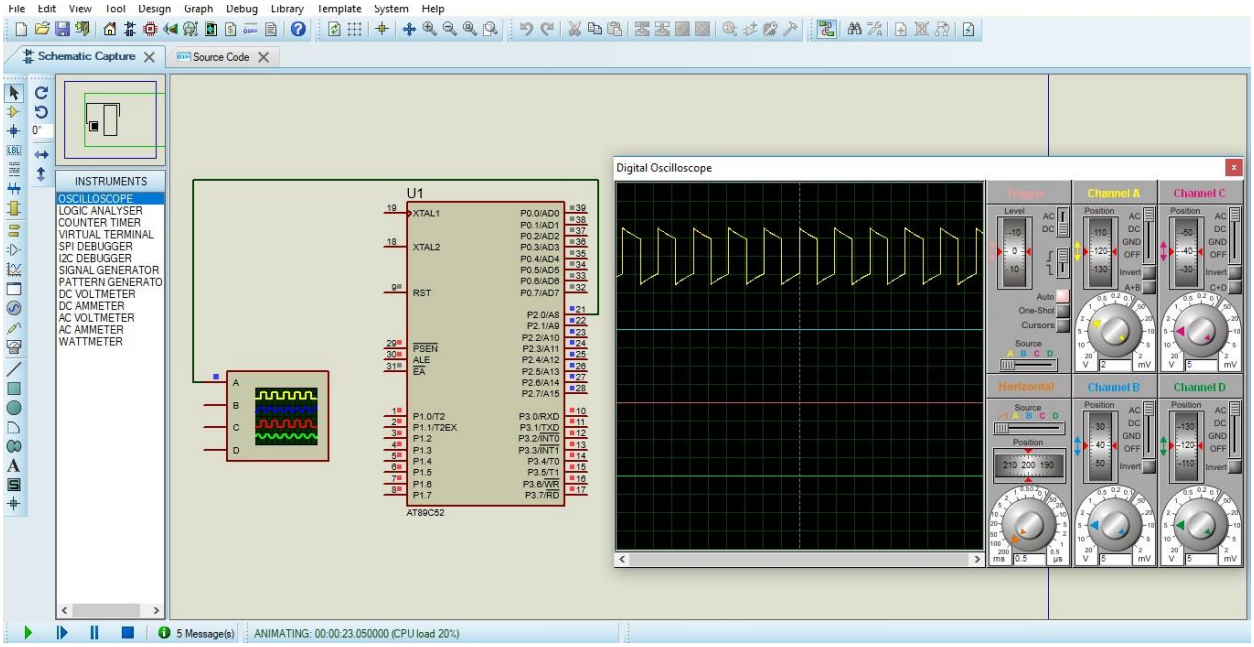
6. Summary is given, Click **Finish**.
7. **Schematic and Source code** appears on the screen.



8. Read program from **External file**, or enter the code in the **Source code** tab.



9. Make the **circuit** connections and **run** the code.



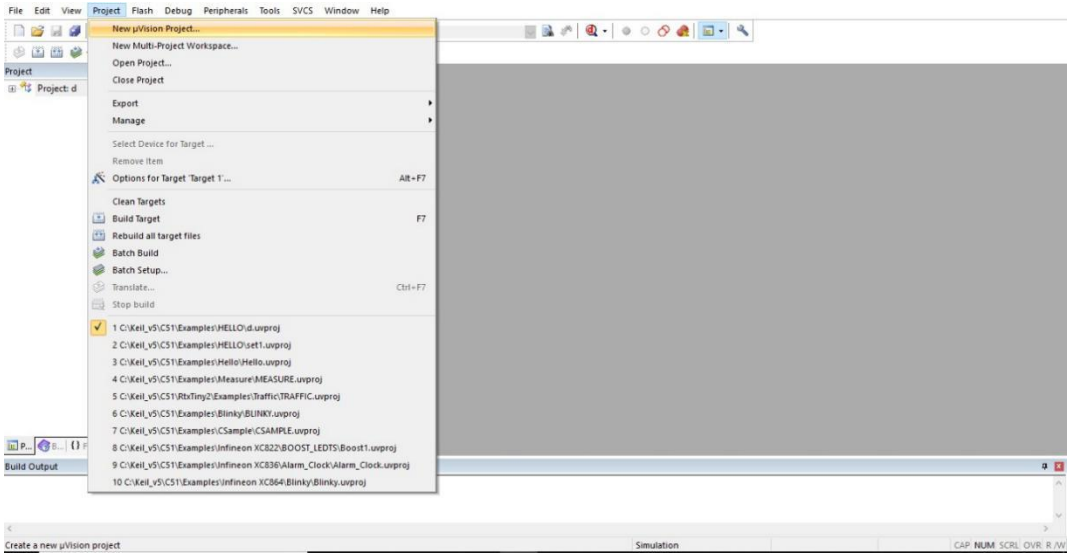
KEIL

Keil is an embedded programming tool. This is used to code compliers, Macro assemblers, Emulators, Debuggers, Programming ARM Cortex, Embedded C programing, and Creation of INTEL HEX files for microcontrollers and microprocessors.

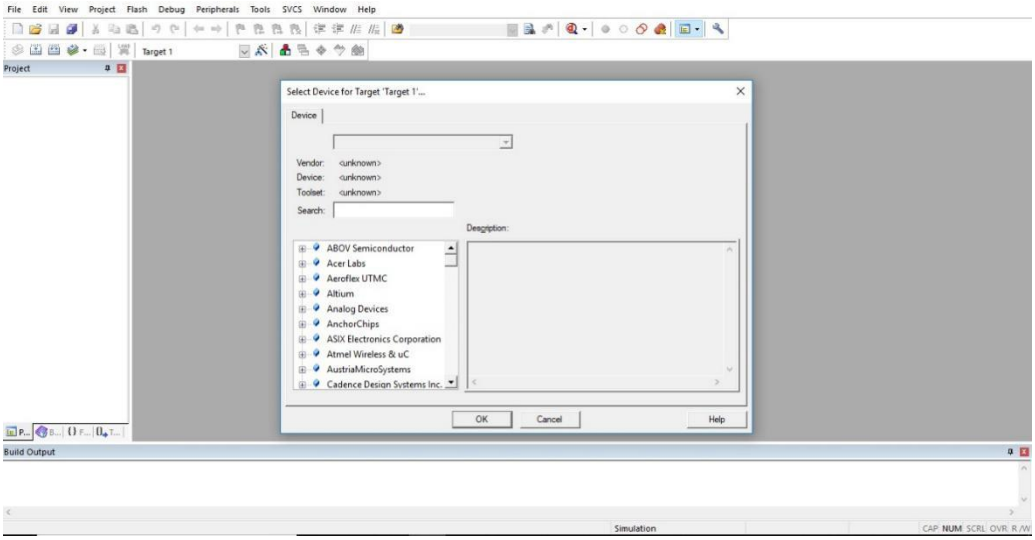
Proteus is used to simulate the design and for testing. The tested code is then written in KEIL for Embedded C (C51 version of Keil µVision50). The code is converted to Intel Hex file format to load it into the microcontroller.

Working with KEIL

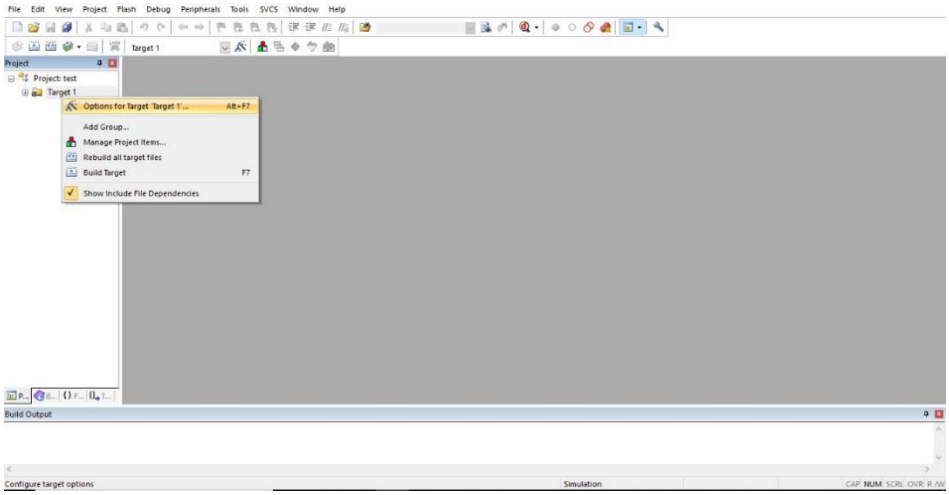
1. Open a New Project / **New µVision project**.



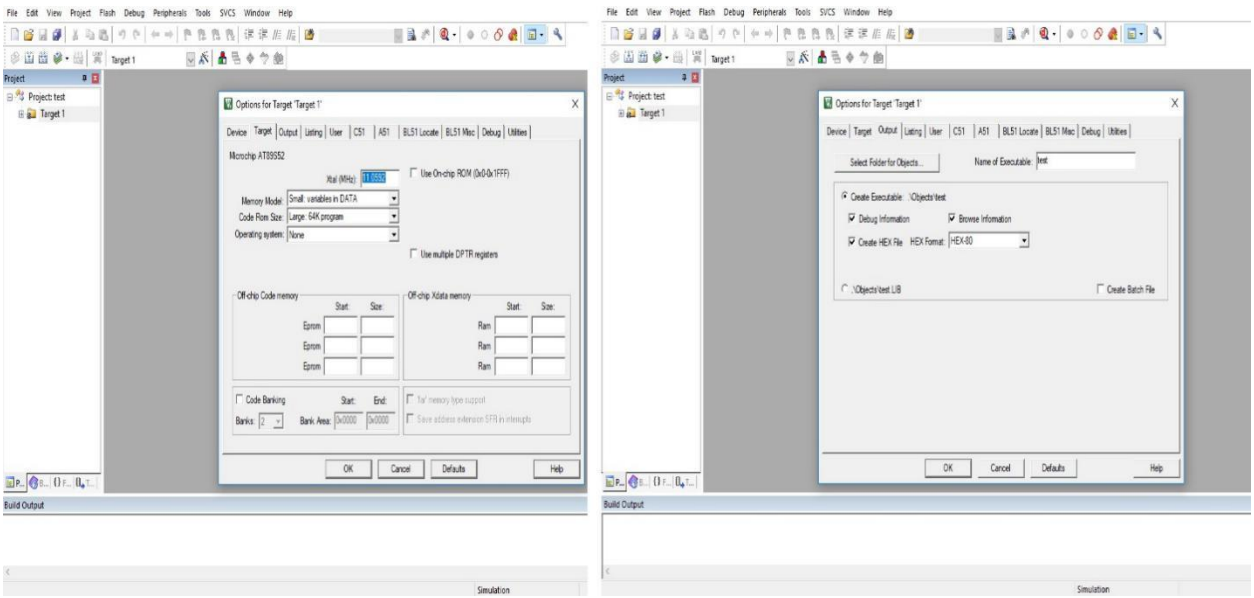
2. Search for the **device** to be programmed. (AT89S52).



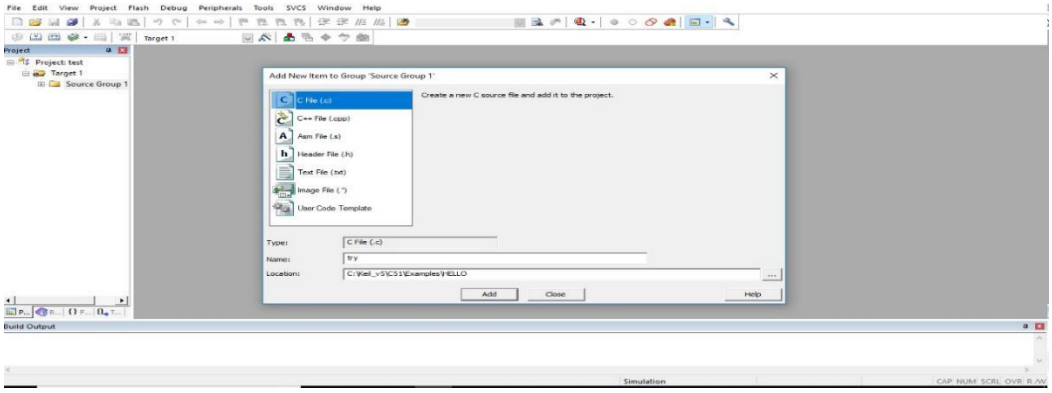
3. Got to **Target**, **Options** for Target.



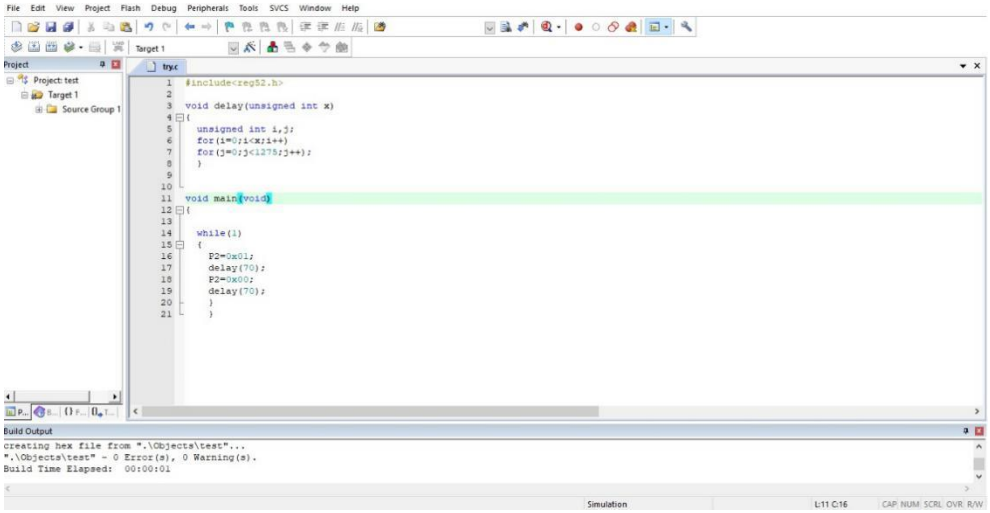
4. Make the crystal oscillator frequency into **11.0592 MHz**, and in the **Output Tab**, select the **HEX file** output (HEX-80).



5. Under Target go to source group and **add new file** (C or ASM file).



6. Program and **Build the code** using **F7**.

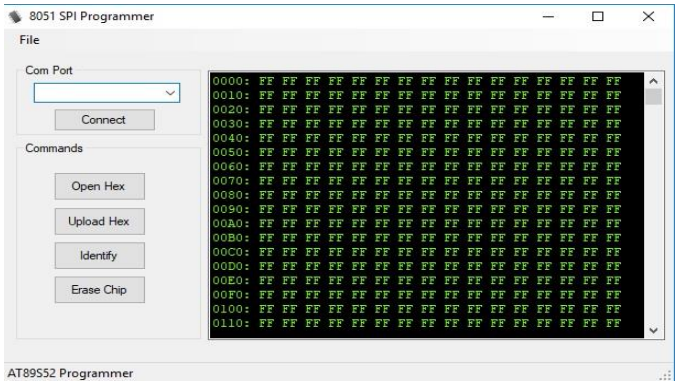


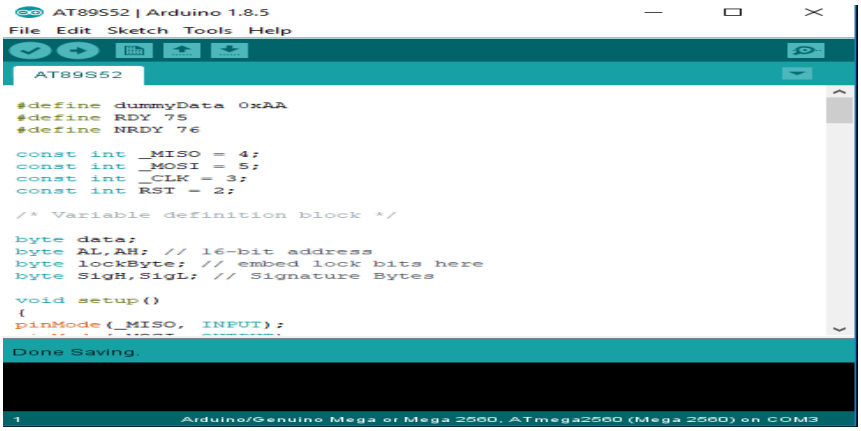
7. The HEX file will be in the target folder, inside **Objects folder**.

SPI Programming With Arduino to 8051

Program the Arduino to read the hex file, and send it serially to 8051

89S52Programmer.exe is used for this purpose to detect, erase, and upload the hex file.

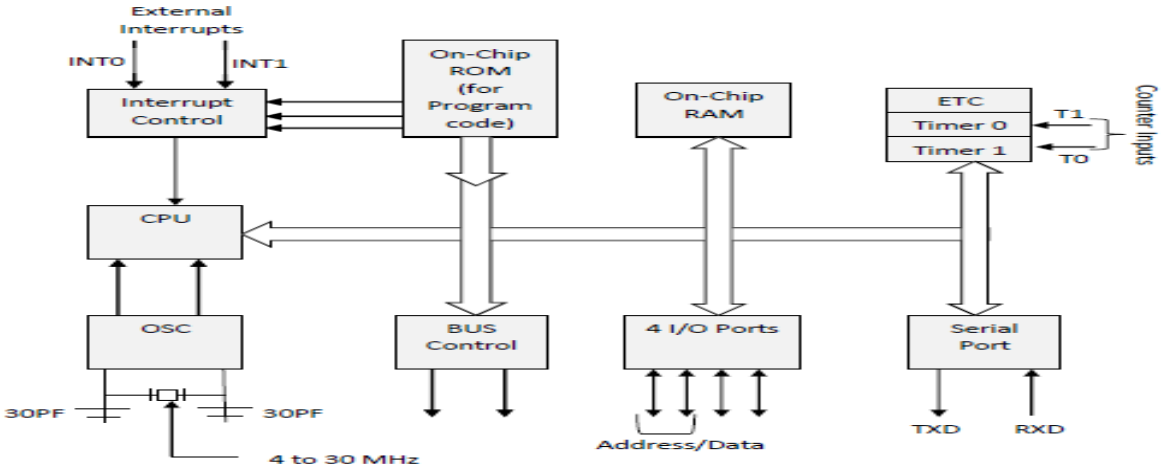




8051- Intel

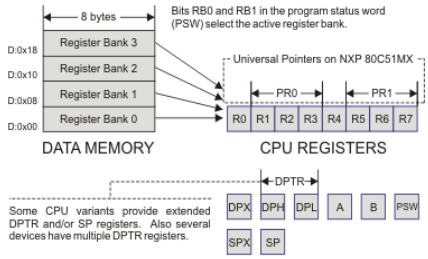
(AT89S52-Atmel, used in the text)

Architecture



- Clock for the Processor can be given from 4Mhz to 40Mhz range.
- There are two 16 bit timers: Timer0 and Timer1 in 8051 (The board used here AT89S52 has three Timer0, Timer1, and Timer2
- On chip 8KB of ROM (ISP programmable: Programming via Arduino, also known as In System Programming with SPI derail interfacing. Using MOSI, MISO, SCLK.) , and 256B of RAM.
- Full Duplex UART.
- For a more detailed description visit http://ww1.microchip.com/downloads/en/DeviceDoc/doc1919.pdf

Registers of 8051



- Most widely used are A (Accumulator), B, and R0 to R7. Other important register is PC (Program Counter, to point to the current address in the program from memory).
- For mathematical applications Status Register is used.

Simulation of Assembly Programs

- Can be done via Proteus. There are other options like ProView32, Edsim51, and MCU 8051.
- Edsim51 requires Java Runtime Environment (JRE) to simulate. This a good tool as it has inbuilt functions of LCD display, Shows all registers, ADC and DAC outputs, and many other features.

Instructions and their uses in Assembly 8051

1. **MOV**: Used for moving data of 8 bit at a time, as 8051 is an 8bit architecture.

   Ex: MOV destination, source;

2. **Arithmetic**:
   a) ADD Destination , Source

   This adds the values in the source and destination registers, and stores the value in the Destination.

   b) ADDC :Add with carry
   c) BCD packed addition

   d) SUBB destination , source

      Subtract with borrow, Subtracts source from destination along with borrow.

      NOTE: to use it as SUB, make CY flag 0 in Status register.

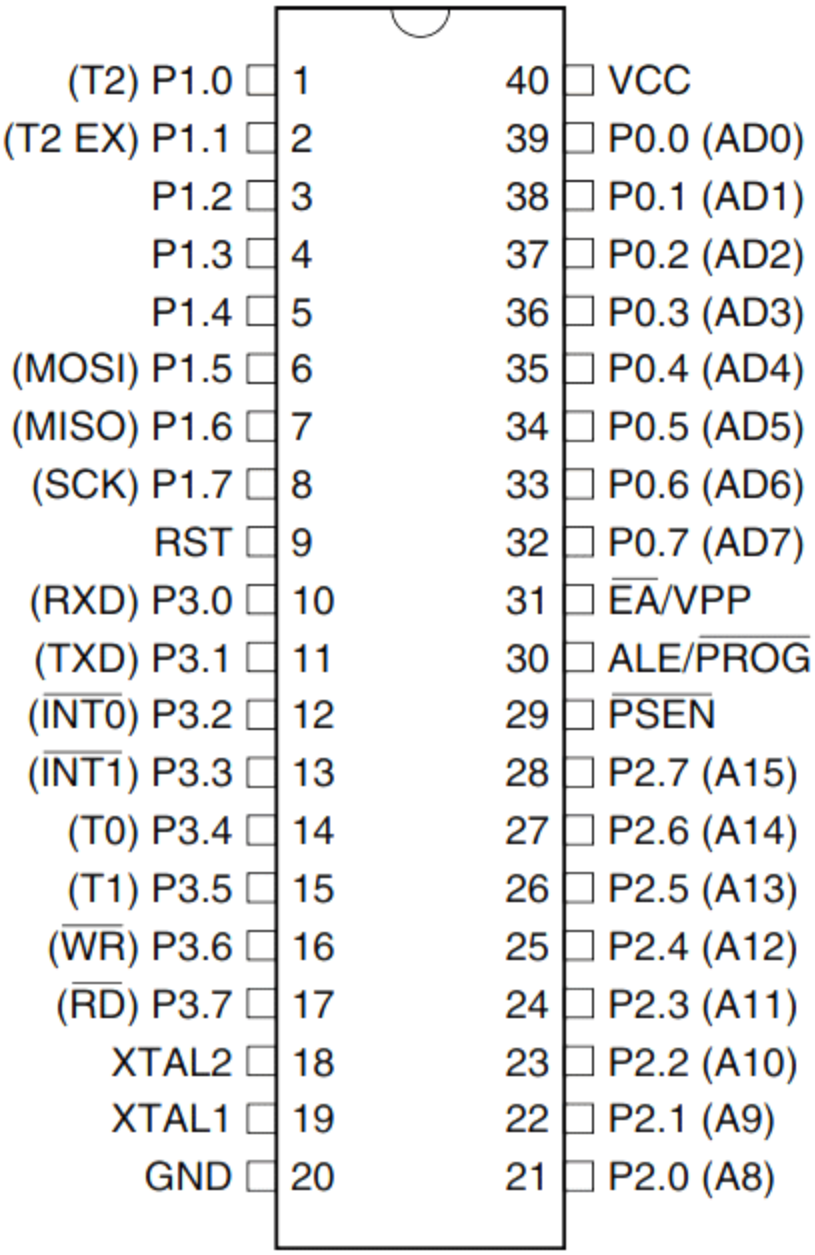
   e) MUL AB: Multiplication of A and B values. Result lower byte in A and higher byte in B.
   f) DIV AB: Numerator in A, denominator in B. Result, quotient in A and remainder in B.

3. **Logical**:
   a) ANL: And operation of dest, src.
   b) ORL: Or operation.
   c) XRL: Xor operation.
   d) RR source: right rotate source.
   e) RL source: left rotate source.
   f) RRC source: Right rotate carry.
   g) RLC source: Left Rotate carry.

4. **Jump** and **loop** statements, these along with **Subroutines** and **Interrupts** can be further read in the Documentation of 8051.

DATA TRANSFER	*ARITHMETIC*	*LOGICAL*	*BOOLEAN*	*PROGRAM BRANCHING*
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ
XCH	MUL	RL	JB	CJNE
XCHD	DIV	RLC	JNB	DJNZ
	DA A	RR	JBC	NOP
		RRC	ANL	LCALL
		SWAP	ORL	ACALL
			CPL	RET
				RETI
				JMP

8051 I/O Programming



- Rxd and Txd is from UART communication
- INT0 and INT1 are external interrupts.
- Pin 31 is connected to Vcc.
- XTAL 1&2 are for crystal oscillator +22pf ceramic capacitor.
- T0, T1, T2 are timer inputs for counter mode.
- MOSI (Master out slave in), MISO (Master in slave out) and SCK are used for Serial Peripheral Interface for ISP programming with the Arduino.

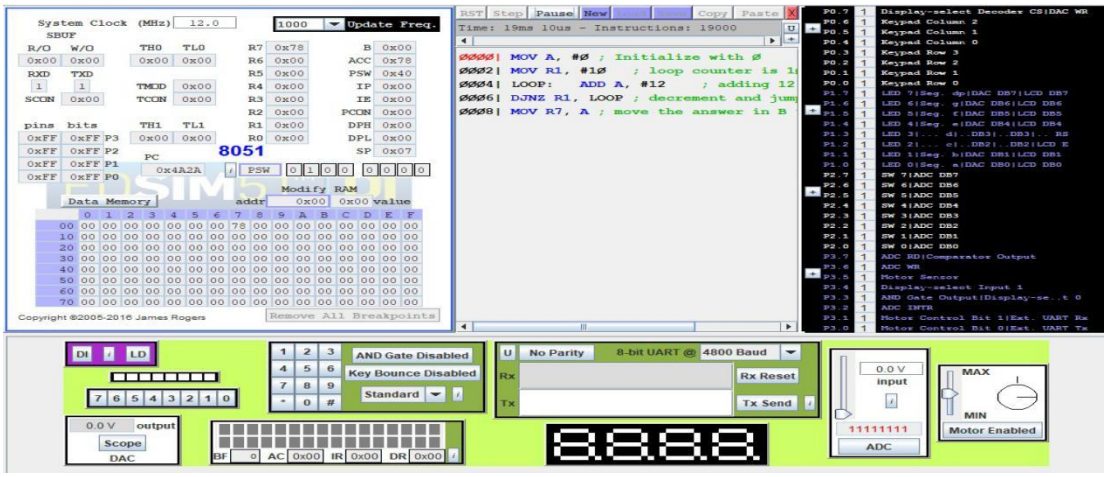
Assembly Programming

Program-1: Moving contents of a register

```
MOV SP, #1FH          ; Stack pointer is added with the address 1F
MOV R0, #25           ; move 25 to R0 register
MOV R1, #3CH          ; move 3C to R1 register
MOV R2, #09H          ; move 9 to R2 register
MOV B, #0DEH          ; move DE to B register
END
```

Program-2: Multiplication as Repeated addition

```
MOV A, #0             ; Initialize with 0
MOV R1, #10           ; loop counter is 10
LOOP:   ADD A, #12    ; adding 12 each iteration
DJNZ R1, LOOP         ; decrement and jump if non zero
MOV R7, A             ; move the answer in B to R7.
END
```



Program-3: Machine cycle and delay calculation

```
                      MACHINE CYCLE
DELAY: MOV R3, #19    ; 1
HERE:  MOV R4, #255   ; 1
LOOP: DJNZ R4, LOOP   ; 2
       DJNZ R3, HERE  ; 2
       RET            ; 2
```

We are using 11.0596 MHz Crystal from AT89S52, hence one cycle will be 1.085 µs
LOOP takes (2x255) x 1.085 µs = 553.35 µs.

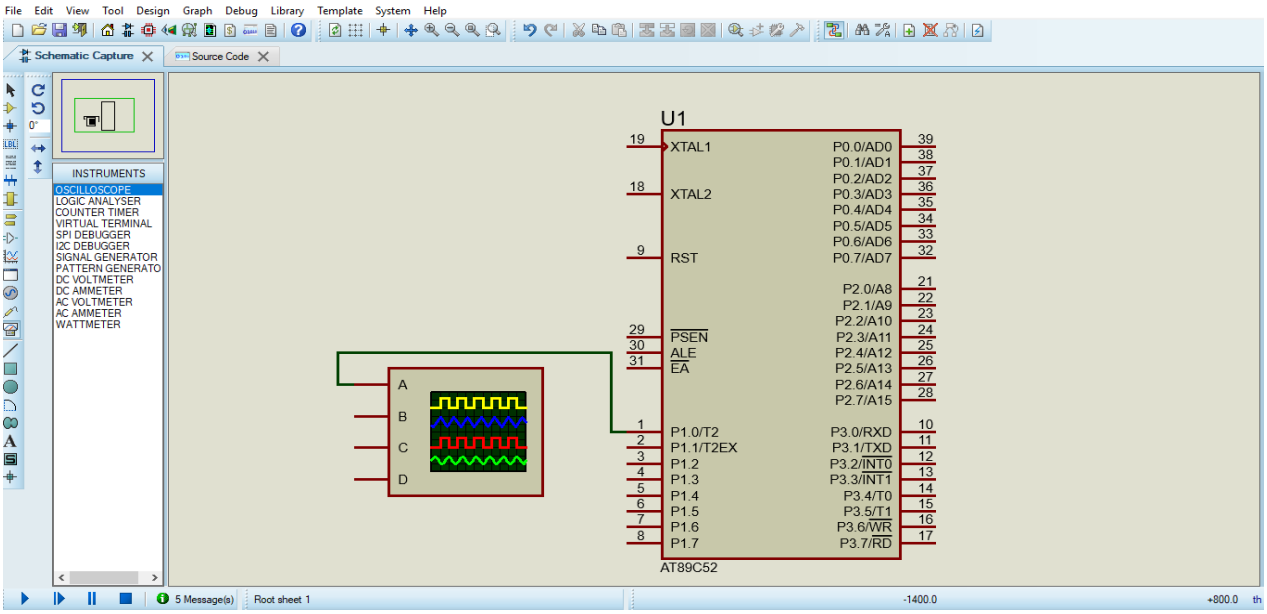
HERE repeats the LOOP 19 times, hence the total delay is
19x 553.35= 10513.65 µs, **approximately 10ms.**
These are approximates, not considering the subroutine instructions.

Program-4: Addition of values in register

```
MOV A, R0              ; move values in R0 to A
ADD A, R1              ; add values of R11 and values in A
JC TEXT                ; if sum is greater than FFh then CY=1 (Carry Flag)
SJMP EXIT              ; if CY=0 then exit
TEXT:  MOV A, #'Y'     ; if CY=1 move 'Y' in to A
       MOV P1, A       ; send output via Port-1
EXIT:  NOP             ; No Operation
       END             ; Terminate
```

Program-5: 50% duty cycle generation

```
BACK: CPL P1.0         ; Complement the value in p1.0
      LCALL DELAY      ; call delay subroutine
      SJMP BACK        ; run in infinite loop
DELAY:
      MOV R3, #19      ; 1
HERE: MOV R4, #255     ; 1
LOOP: DJNZ R4, LOOP    ; 2
      DJNZ R3, HERE    ; 2
      RET              ; 2
END
```



Program-6: Read serial data and placed in output port

```
        MOV R0, #08          ; counter value 8
        SETB P0.0            ; make P0.0 as input
LOOP:   MOV C, P0.0          ; move data from P0.0 to carry bit
        RRC A                ; right rotate with carry
        DJNZ R0, LOOP        ; repeat until all 8 bits are moved
        MOV P1, A            ; transfer in a parallel format
        END
```

C Programming Of 8051

Advantages:

1. Platform Independent.
2. Easier than Assembly.
3. Many Libraries can be used.

Disadvantages:

1. Limited memory in Microcontrollers, as C code takes more space than Assembly.
2. High precision programming is better done in Assembly.

Data Type used in 8051 series:

The most preferred data type is 'Unsigned Char', as it occupies 8bit data. This is very much suitable for the 8bit architecture of 8051.

Avoid using other data types like 'int' as they occupy large memory in the microcontroller.

- Unsigned Char
- Unsigned Int
- Signed Char
- Signed Int
- Sbit
- Bit and sfr

Default the data type int or char will be signed, for unsigned we have to mention it explicitly.

Program Set-1

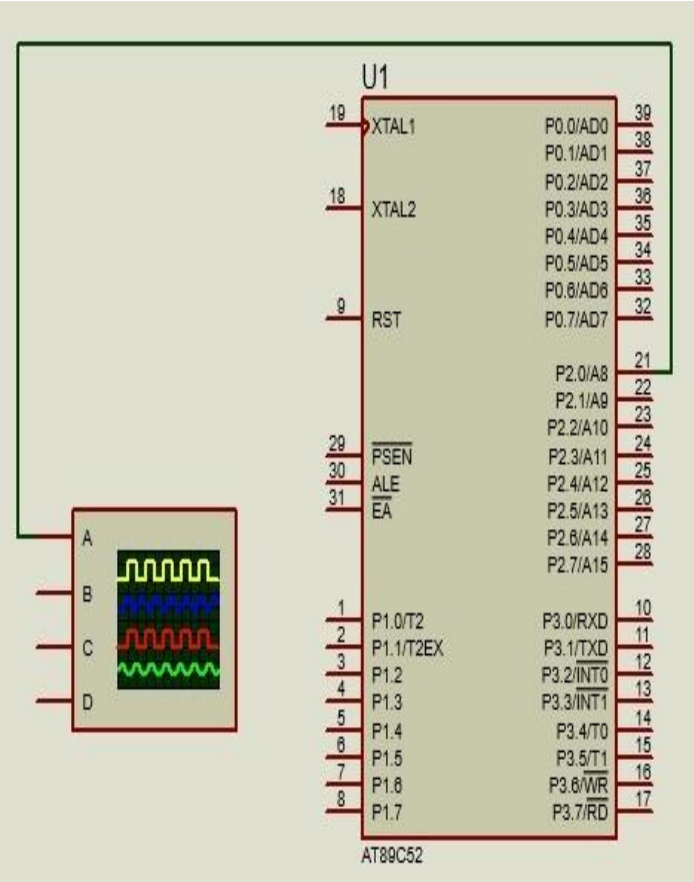
Program -1: To send Hex values of ASCII characters

```
#include <reg52.h>                                  // reg51.h for 8051, reg52.h for AT89S52

void main (void)

{

        unsigned char a[]= "0123456789ABCDEF";

        unsigned char z;

        for(z=0;z<10;z++)

                P1=a[z];

}
```

Program-2: Toggle LED delay version 1

```
#include<reg52.h>
void main(void)
{       unsigned int z;
        while(1)                    // Run Always, can also use for(;;)
        {       P1=0x02;
                for(z=0;z<50000;z++);     // Delay count
                P1=0x00;
                for(z=0;z<50000;z++);
        }
}
```

Program-3: Toggle LED delay version 2



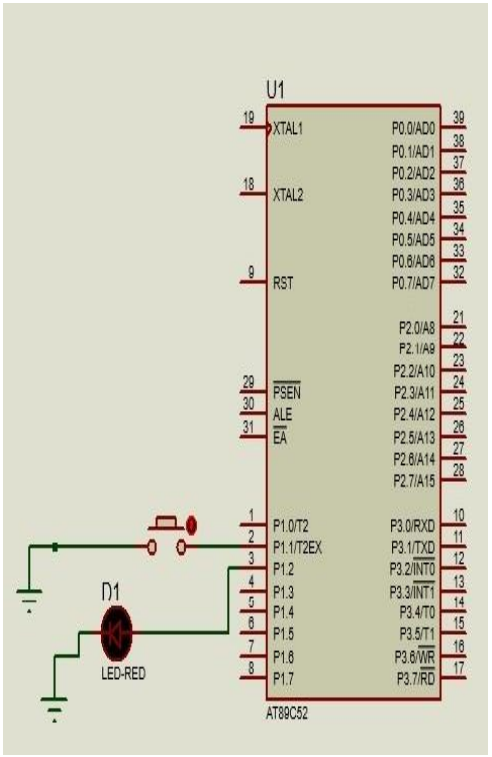

```
#include<reg52.h>
void delay(unsigned int x)
{       unsigned int i,j;
        for(i=0;i<x;i++)
                for(j=0;j<1275;j++);
}
void main(void)
{       while(1)
        {       P2=0x01;
                delay(70);
                P2=0x00;
                delay(70);
        }
}
```

Program-4: Push button LED turn on

```
#include<reg52.h>
sbit x = P1^1;                          //sbit to declare port
sbit led =P1^2;
void main(void)
{       x=1;                            //make x as input
        while(1)                // to keep checking the status of the pin
        {       if(x==0)                // active low input
                        led=1;
                else
                        led=0;
        }       }
```

Program-5: Switch or Sensor based alarm

```
#inlcude<reg52.h>
sbit in = P1^1;
sbit out = P1^2;
void delay(unsigned char z)
{       unsigned int m,n;
        for(m=0;m<z;m++)
for(n=0;n<1275;n++);
}
void main(void)
{       while(in==0)
        {       out=1;
                delay(70);
                out=0;
                delay(70);
        }       }
```



TIMER

Two timers Timer0 and Timer1 (in AT89S52 we have Timer2)

The timer register is divided into 2. The timer register is 16 bit long, but 8051 can access 8bit at a time, so the upper and lower byte are divided. Lower byte is TL0 & TL1 for timer0 and 1 respectively. Similarly higher byte TH0 & TH1.

Timer modes:

Mode 0: 13 bit timer / counter (8+5).

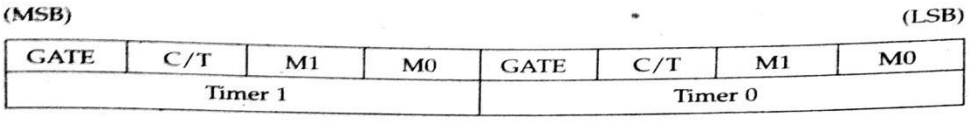
Mode 1: 16 bit timer / counter (max values FFFFh)

Mode 2: 8 bit timer/ counter (max value FFh)

Mode 3: Split timer mode

C/T bit allows one to select whether the module should act as a timer or a counter. The 16 bit counter is very useful, if coupled along with 1Hz input, it can serve as a calendar for 136 years!!

Timer will use 1/ 12 th the frequency of the crystal oscillator, regardless of the machine cycle. To change the speed of the timer one must change the crystal.



GATE Gating control when set. The timer/counter is enabled only while the INTx pin is high and the TRx control pin is set. When cleared, the timer is enabled whenever the TRx control bit is set.

C/T Timer or counter selected cleared for timer operation (input from internal system clock). Set for counter operation (input from Tx input pin).

M1 Mode bit 1

M0 Mode bit 0

M1	M0	Mode	Operating Mode
0	0	0	13-bit timer mode 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	16-bit timer mode 16-bit timer/counters THx and TLx are cascaded; there is no prescaler
1	0	2	8-bit auto reload 8-bit auto reload timer/counter; THx holds a value that is to be reloaded into TLx each time it overflows.
1	1	3	Split timer mode

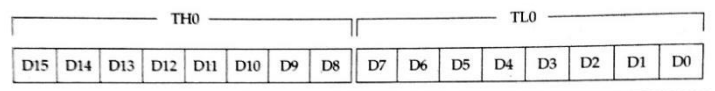
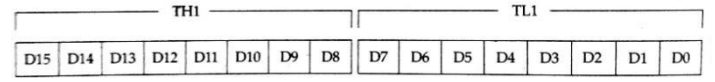


Figure 9-1. Timer 0 Registers



Program Set -2

Program-1: LED blinking at exactly 1 second with timer module

```
#include<reg52.h>
void delay(void)
{       TMOD=0x10;
        TL1=0xFE;
        TH1=0xA5;       // this will give 500ms delay
        TR1=1;
        while(TF1==0);
        TR1=0;
        TF1=0;
}
void main(void)
{       unsigned int x;
        while(1)
        {
                P0 =P0 ^ 0x01;
                for(x=0;x<20;x++)
                        delay();
        }
}
```

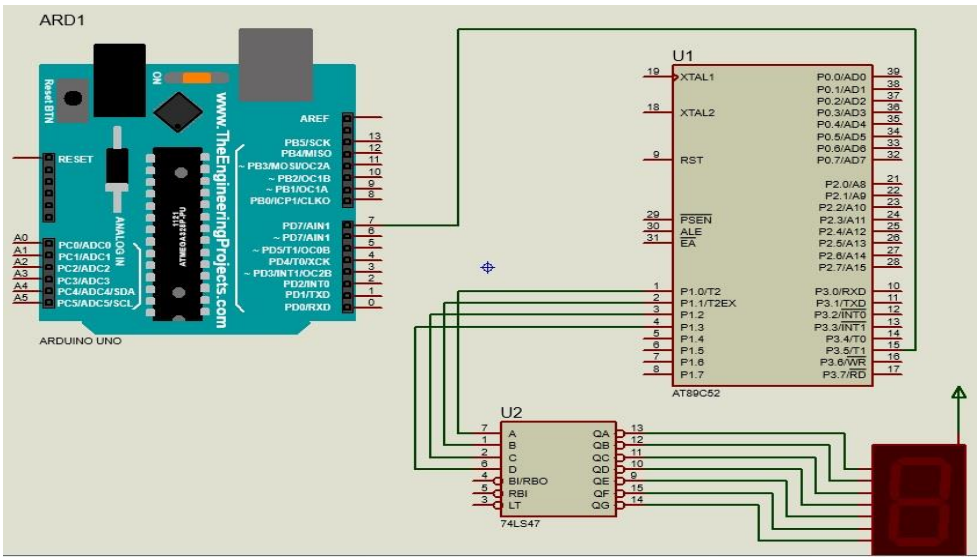
Program-2: Selection of frequency with button. Using timer for frequency generation

```
#include<reg52.h>
sbit led=P2^0;                  // led pin
sbit button=P1^2;               // switch pin


void delay(unsigned char y)
{       TMOD=0x10;
        if(y==1)
        {       TL1=0xFE;
                TH1=0xA5;
        }                               // this will give 500ms delay
        else
        {       TL1=0xFE;
                TH1=0x55;
        }                               //this will give 250ms delay
        TR1=1;
        while(TF1==0);
        TR1=0;
        TF1=0;
}
void main(void)
{       while(1)
        {       led ^= 0x01;
                if(button==0)
                        delay(0);
                else
                        delay(1);
        }       }
```

Program-3:

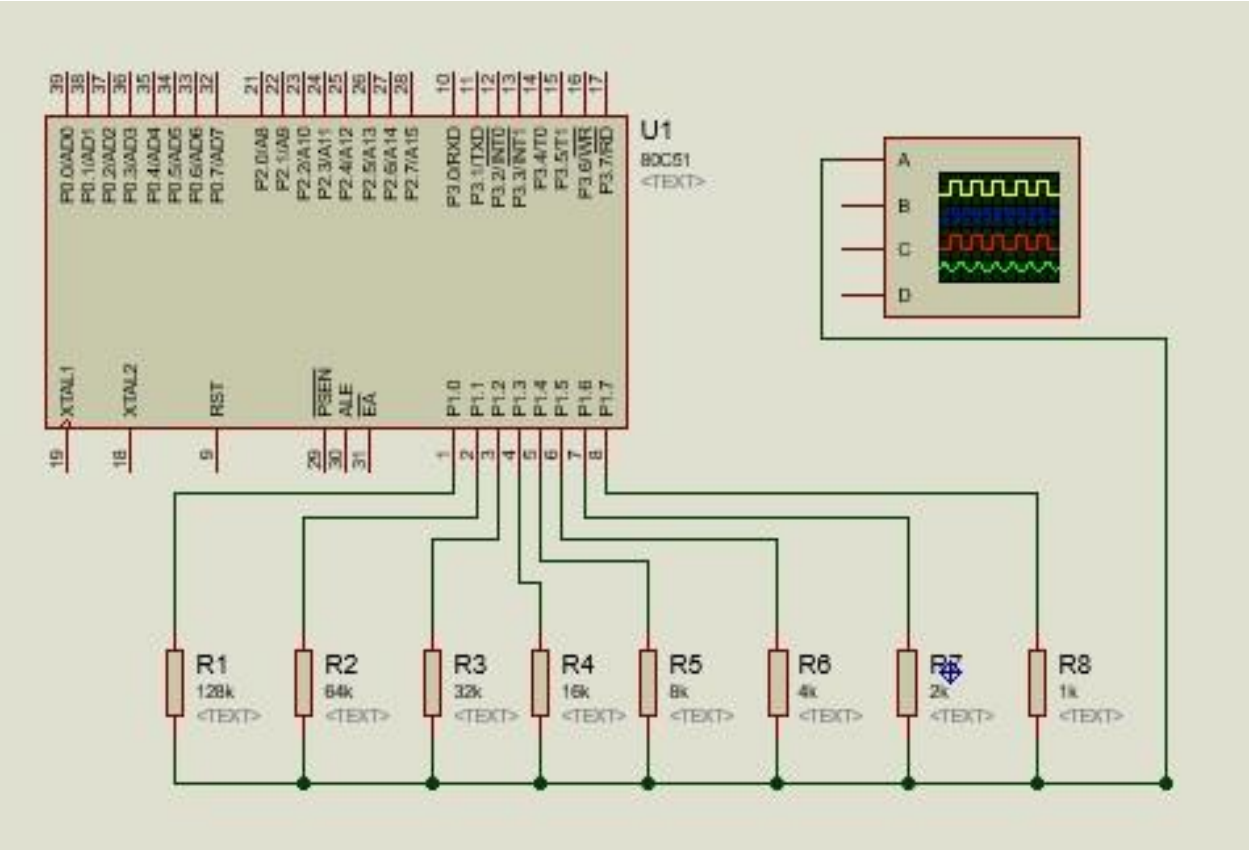
Using timer as a counter, to keep tracking the input pulse and display in output port. The SIGNAL can be provided from Arduino generating a 1Hz square wave.

```
#include<reg52.h>

sbit freq=P3^5;          // For Timer1 INPUT PIN, can't be replaced with any other (Pin 3.5)


void main(void)
{       freq=1;
        TMOD=0x61;
        TH1=0;
        while(1)
        {       do
                {       TR1=1;
                        P1=TL1;
                }
                while(TF1==0);
                TR1=0;
                TF1=0;
        }       }
```



Program-4: Generation of sine wave in C

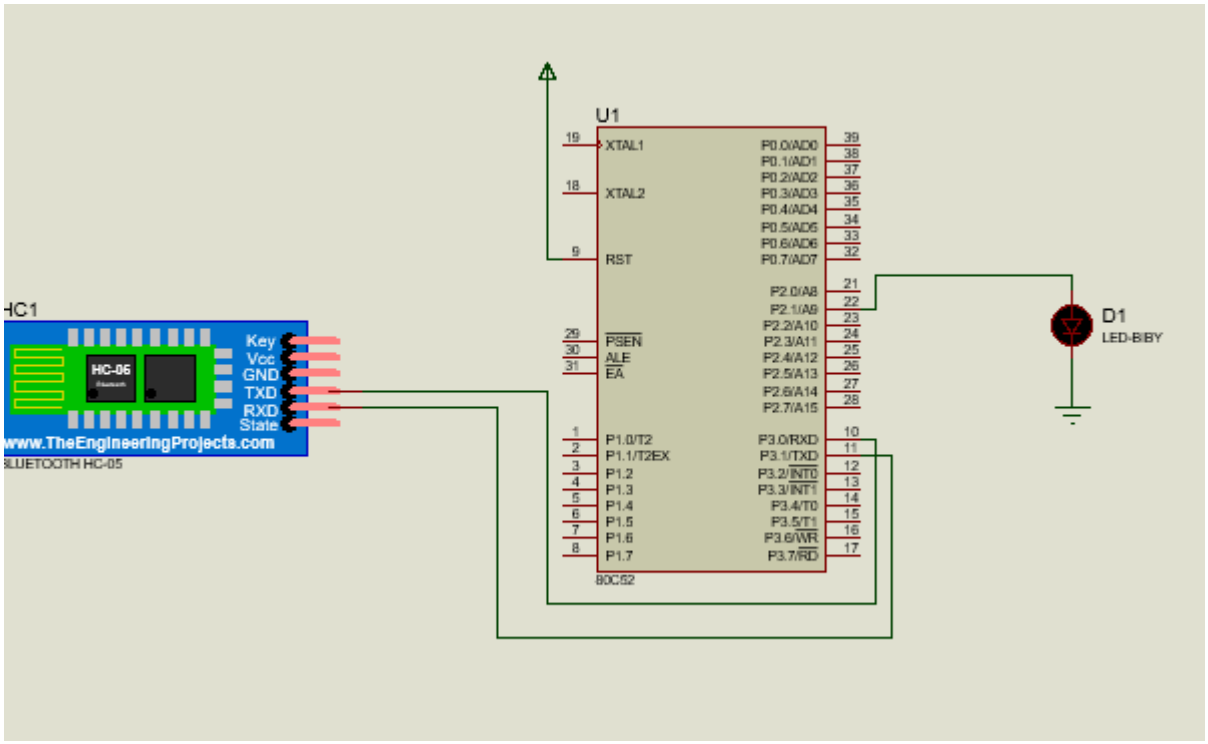
```
#include<reg51.h>
void delay();
void main()
{       unsigned int i;
        unsigned int wavelength[16]={ 127,176,218,245,255,245,218,176,128,79,27,10,0,10,37,79 };
        while(1)
        {       for(i=0;i<16;i++)
                {       P1=wavelength[i];
                }
        }
}
```



Program-5: UART

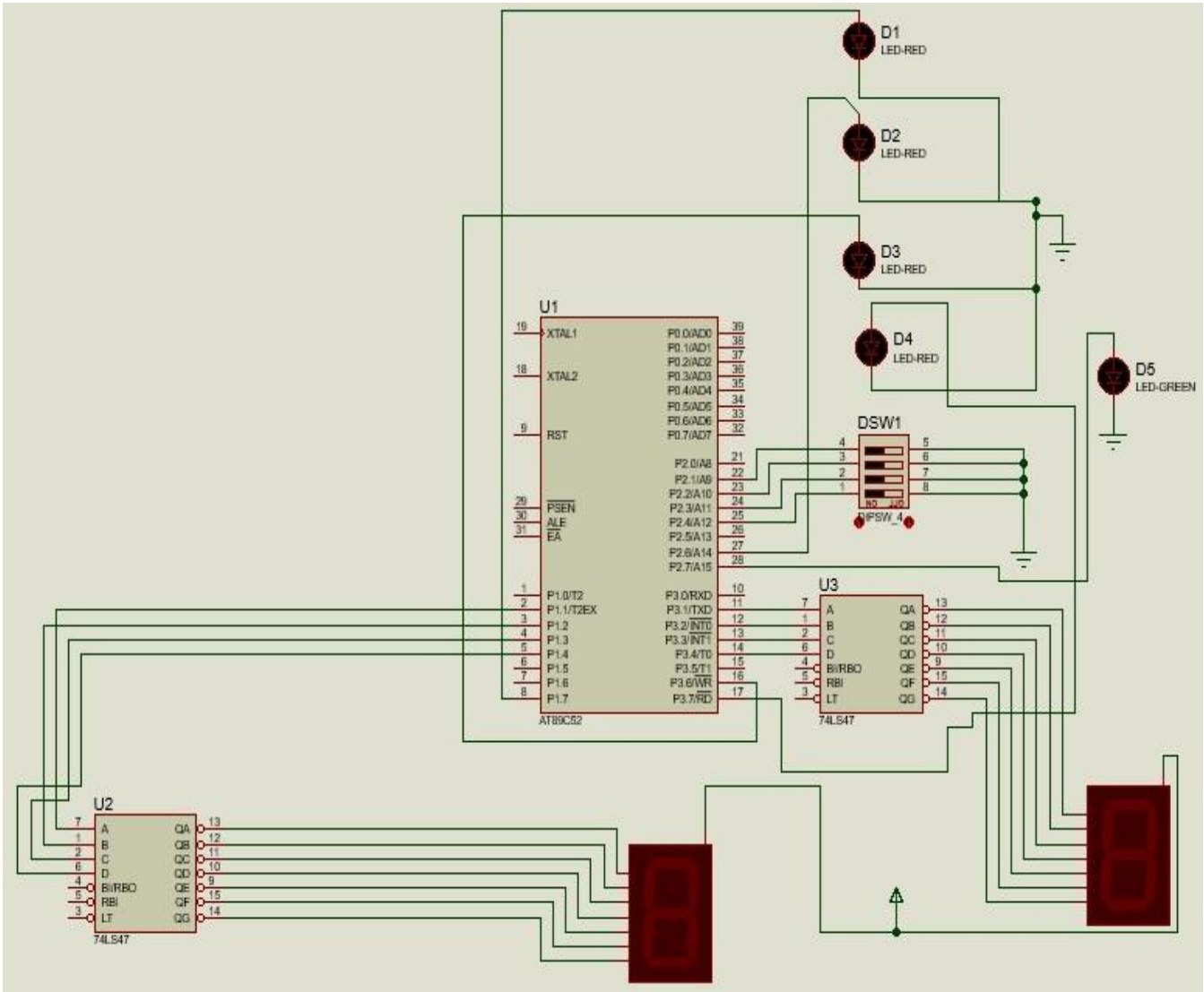
```
#include<reg51.h>
sbit x=P2^1;
void write(char p)
{       SBUF=p;
        while(TI==0);
        TI=0;
}
void serial(char *p)
{       while(*p)
        {       write(*p);
                p++;
        }
        write('\r');
}
void main()
{       char o;         TMOD=0x20;
         TH1=0xfd;      SCON=0x50;
        TR1=1;
        serial("begin");
        while(1)
        {       while(RI==0);
                o=SBUF;
                RI=0;
                if(o=='A')
                {       x=1;
                        serial("LIGHT IS ON");
                }
```

```
                else if(o=='B')
                {x=0;
                        serial("light is off");
                }
        }
}
```



MINI PROJECT

Development of a System that reads a 4 bit password, once the correct password is fed a countdown timer starts from 60. The timer runs for every 1 second, so countdown of 60 seconds precisely. The display is shown via Seven Segment with a decoder. After the countdown is completed, another task can be made to run. Here few LED glow in a certain pattern. (Password is 2314)



Code

```
#include<reg52.h>
void password(); void wait();      void output();          void delay(int x);
sbit error=P2^7;          sbit pass0=P2^1;
sbit pass1=P2^2;          sbit pass2=P2^3;
sbit pass3=P2^4;          sbit out1=P1^7;
sbit out2=P2^6;           sbit out3=P3^6;          sbit out4=P3^7;
char seven[10]={0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12};
void main()
{       int p=9,k=5;
         error=0;
        out1=0;           out2=0;
        out3=0;           out4=0;
        password();
        P3=seven[6];      P1=seven[0];
        delay(100);
        P3=seven[5];
        while(1)
        {       for(p=9;p>=0;p--)
                {       P1=seven[p];
                        delay(100);
                }
        k=k-1;
                if(k==-1)
                {       output();       }
        P3=seven[k];
                }
}
```

```
void password()
{       while(1)
        {       if(pass1==pass0==pass2==pass3==1)
                {       wait();
                        if(pass2==0&&pass1==pass0==pass3==1)
                        {       wait();
                                if(pass1==0&&pass0==pass3==1)
                                {       wait();
                                        if(pass3==0&&pass0==1)
                                        {       wait();
                                                if(pass0==0)
                                                {       error=1;
                                                        return;
                                                }

                                        }

                                }

                        }
                }
        }
}
void wait()
{       int passwait0=pass0;    int passwait1=pass1;    int passwait2=pass2;    int passwait3=pass3;
        while(passwait0==pass0&&passwait1==pass1&&passwait2==pass2&&passwait3==pass3)
        {       }
}
```

```
void delay(int x)
{       if(x==0)
        {       return;
        }
        TMOD=0x10;
        TL1=0xff;
        TH1=0xdb;
        TR1=1;
        while(TF1==0);
        TR1=0;
        TF1=0;
        delay(x-1);
}
void output()
{       while(1)
        {       out1=1;
                        delay(10);
                out3=1;
                        delay(10);
                out2=1;
                        delay(10);
                out4=1;
                        delay(10);
                out1=0;
                        delay(10);
                out3=0;
                        delay(10);
```

```
                out2=0;
                        delay(10);
                out4=0;
                        delay(5);
                out1=1;
                out2=1;
                out3=1;
                out4=1;
                        delay(5);
                out1=0;
                out2=0;
                out3=0;
                out4=0;
                        delay(5);
        }
}
```

MINI PROJECT *(Version-2)*

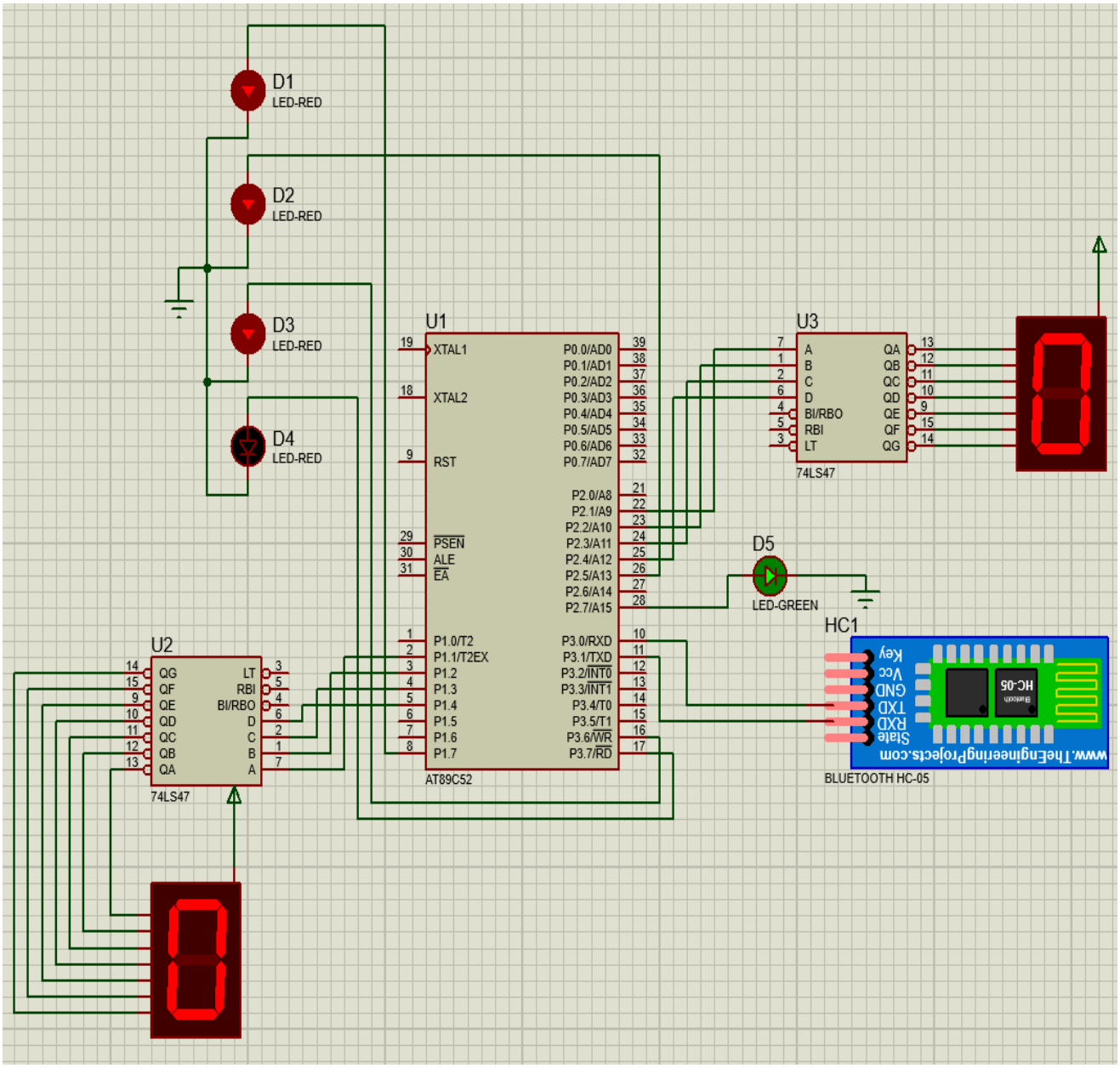
Development of a System that reads a 4 bit password, once the correct password is fed a countdown timer starts from 60. The timer runs for every 1 second, so countdown of 60 seconds precisely. The display is shown via Seven Segment with a decoder. After the countdown is completed, another task can be made to run. Here few LED glow in a certain pattern. (Password is rnsit1). The password is fed to a Bluetooth module from a mobile phone.

```
#include<reg52.h>
#include<string.h>
void password();        void wait();        void output();        void delay(int x);
sbit error=P2^7;        sbit out1=P1^7;     sbit out2=P2^6;
sbit out3=P3^6;         sbit out4=P3^7;
char seven[10]={0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12};
char a;
char k[10]="rnsit1";
char b[10];
void x(char v)
{       SBUF=v;
        while(!TI);
        TI=0;
}
void write(char *p)
{ while(*p)
        {       x(*p);
                p++;
        }
        x(0x0d);
}
void recieve()
{       int j=0;
```

```
        write("enter the password");
        while(SBUF!='\r')
        {       while(RI==0);
                RI=0;
                if(SBUF!='\r')
                {       b[j]=SBUF;
                        j++;
                }
        }
}
void password()
{       while(1)
        {       recieve();
                if(strcmp(b,k)==0)
                {       write("password correct");
                        error=1;
                        return;
                }
                write("password incorrect /n please reset");
                while(1);
        }
}
void main()
{       int p=9,k=5;
        error=0;
        out1=0;
        out2=0;
        out3=0;
```

```
        out4=0;
        TMOD=0x20;
        SCON=0x50;
        TH1=0xfd;
        TR1=1;
        write("begin");
        password();
        P2=seven[6];
         P1=seven[0];
         delay(100);
         P2=seven[5];
        while(1)
        {
                for(p=9;p>=0;p--)
                {       P1=seven[p];
                        delay(100);
                }
                k=k-1;
                if(k==-1)
                {       output();
                }
                P2=seven[k];
                }
}
void delay(int x)
{       if(x==0)
        {       return;
        }
```

```
        TMOD=0x10;
        TL1=0xff;
        TH1=0xdb;
        TR1=1;
        while(TF1==0);
        TR1=0;
        TF1=0;
        delay(x-1);
}
void output()
{       while(1)
        {
                out1=1;         delay(10);
                out3=1;         delay(10);
                out2=1;         delay(10);
                out4=1;         delay(10);
                out1=0;         delay(10);
                out3=0;         delay(10);
                out2=0;         delay(10);
                out4=0;         delay(5);
                out1=1;         out2=1;         out3=1;         out4=1;
                delay(5);
                out1=0;         out2=0;         out3=0;         out4=0;
                delay(5);
        }
}
```


D1
LED-RED
D2
LED-RED
D3
LED-RED
D4
LED-RED
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
AT89C52
U2
QG
QF
QE
QD
QC
QB
QA
LT
RBI
BI/RBO
D
C
B
A
74LS47
U3
74LS47
D5
LED-GREEN
HC1
Key
Vcc
GND
TXD
RXD
State
BLUETOOTH HC-05
www.TheEngineeringProjects.com