Chapter 2 The First Program

2.1 Blink a LED by software delay

The first program is a simple one: just to blink a LED. Figure 2.1.1 is an extract from the full schematic. The driving port of LED D3 is P2.0.

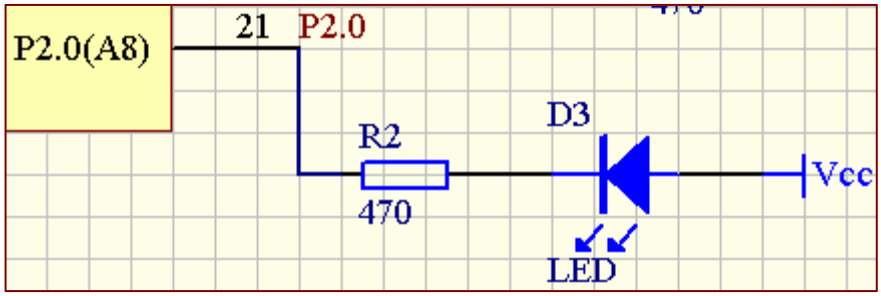

Figure 2.1.1

The source code is located under cd:\src\chp2\src2_1\. Project name is blinkLED.uv2. Launch µVision2 IDE, under **Project** manual (Figure 2.1.2), browse to blinkLED.uv2 and click open.

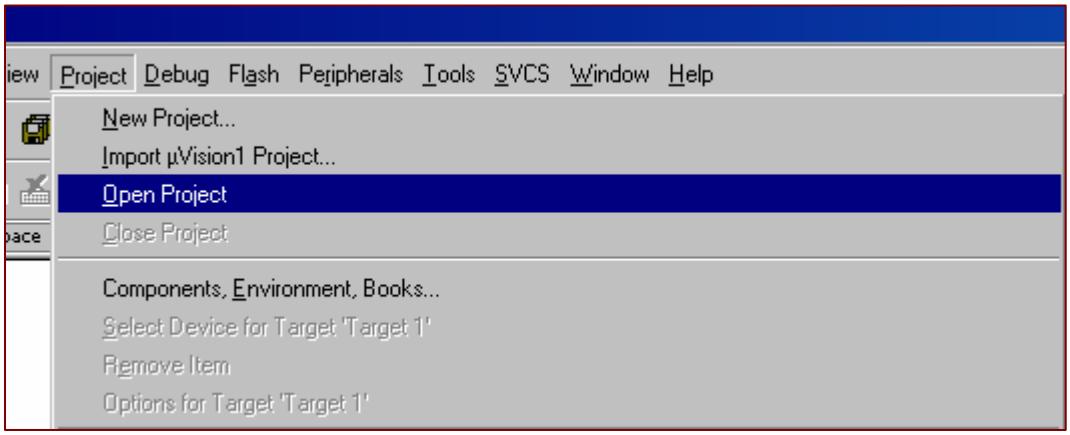

Figure 2.1.2

```
#include <reg52.h>                                          (1)

#define ON    0                                             (2)
#define OFF   1

sbit LED = P2^0;                                            (3)
void DelayMs(unsigned int msec);                            (4)

void DelayMs(unsigned int msec)                             (5)
{
        unsigned int x,y;
        for(x=0; x<=msec;x++)
                {
                        for(y=0;y<=110;y++);
                }
}

void main(void)
{
        for(;;)
                {
                        LED = ON;                           (6)
                        DelayMs(50);
                        LED = OFF;                          (7)
                        DelayMs(50);
                }
}
```

Listing 2.1

Line (1) is the register definition file provided by Keil C. It is a map of the registers of 89C52 compatible mcu.

Line (2) is a constant definition for portability.

Line (3) defines the hardware pin P2.0 to be the LED driving port. The data type 'sbit' is provided by Keil C for the 8051 family and is used to access an individual I/O pin.

Line (4) is the prototype for a delay function.

Line (5) is the body of the delay function. It has been adjusted for an 11.0592MHz crystal.

Line (6) is the heart of this program. AT89S52 is not the same as other microcontrollers like Microchip's PIC or Texas Instrument's MSP430. There is no register to control I/O direction. To turn on the LED, we need writing 0 to P2.0 to output low.

Line (7) turns off the LED by writing 1 to P2.0 to make it a high impedance input.

To get a better idea of how lines 6&7 work, we need to refer to the basic port structure of 8051. There is an Atmel 8051 Microcontrollers Hardware Manual Rev. 4316D-8051-05105 available from Atmel web site. Figure 2.1.3 is an extract from the manual on Port 2.

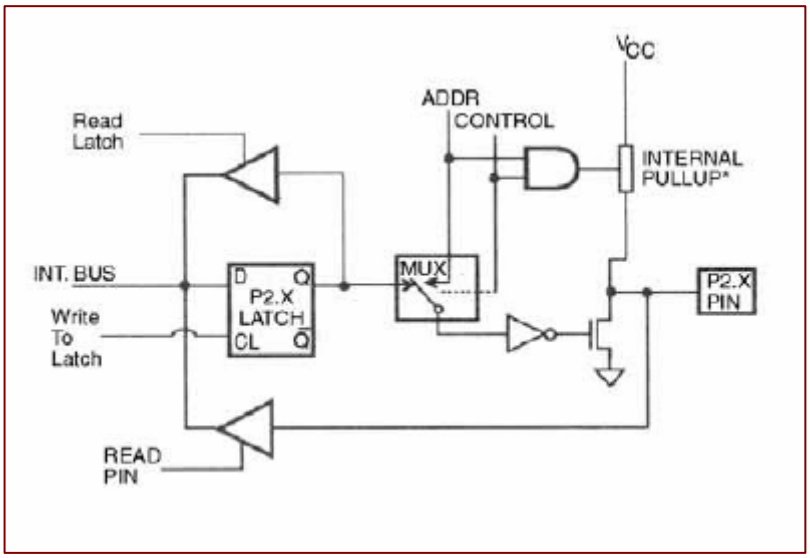

Figure 2.1.3 P2 structure

Connect a LED's cathode via a current limiting resistor to P2.0 is like Figure 2.1.4.

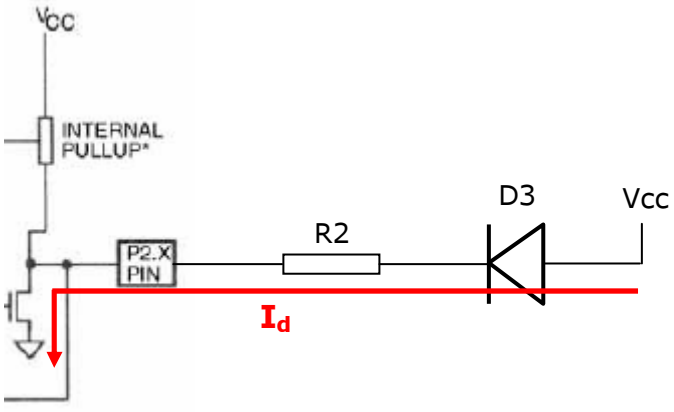

Figure 2.1.4

By running the code **LED= ON (eq. P2^0=0)**, we write 0 to the P2.0 latch and thus turning ON the output driver FET. Current flows from Vcc to anode of D3, passing through the current limiting resistor R2 (470Ω) and finally through the internal FET to ground; therefore, turning LED ON.

On the contrary, writing 1 to the port latch by the code **LED=OFF (eq. P2^0=1)** will turn off the output driver FET. Port P2^0 is pulled up by the internal pull-up resistor to Vcc. No current flow through D3 thus turning off the LED.

Now, let's check all options for proper HEX file generation. Right click on **Target 1** under the current Project Workspace (Figure 2.1.5).

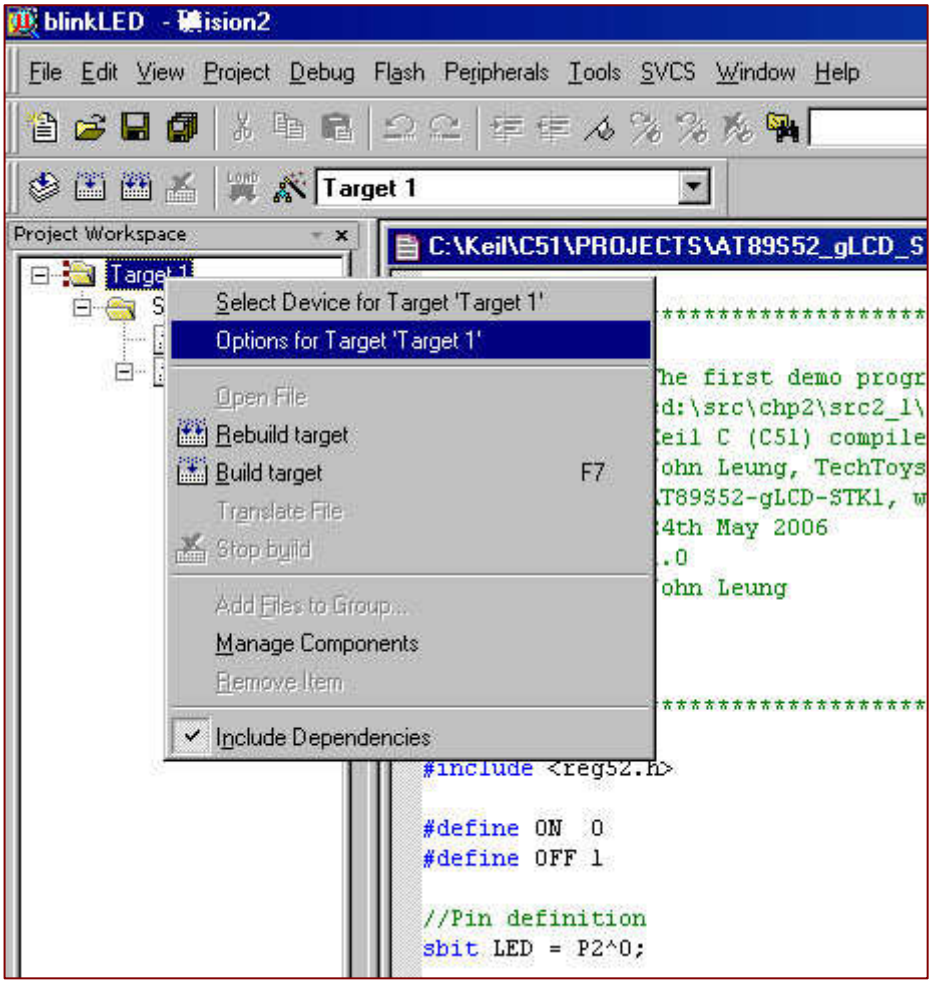

Figure 2.1.5

The Xtal frequency should be 11.0592MHz which is the crystal frequency onboard. Check **Use On-chip ROM** option because we are running code from the internal flash of AT89S52. Memory Model is Small and Code Rom Size is also Small.

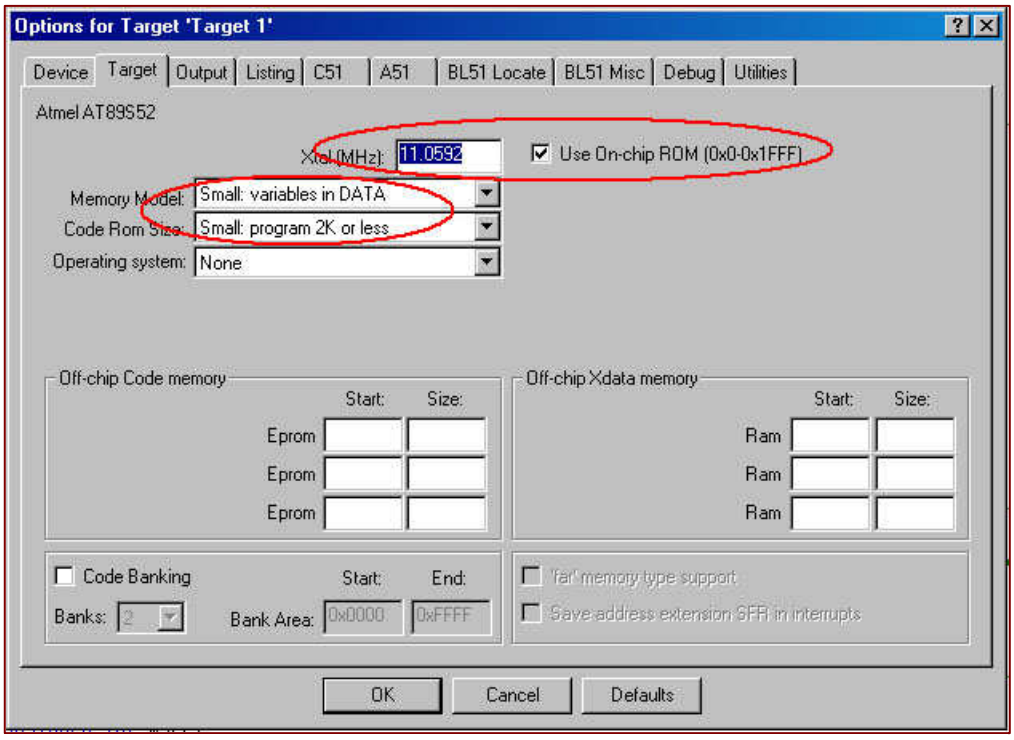

Figure 2.1.6 Target option

The next important option is the **Output** option. Make sure the **Create HEX File** radio button is checked, otherwise, no hex file will be generated.

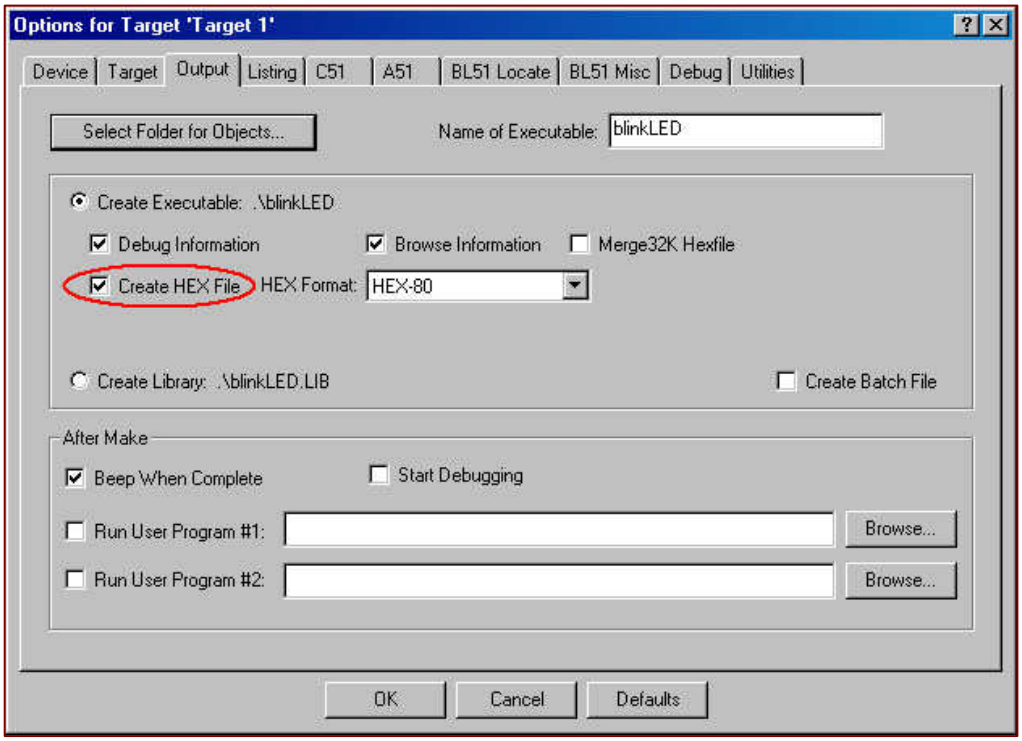

Figure 2.1.7 Output option

Click OK to close the option window. Under **Project**, select **Rebuild all target files**. Browse to the project folder and look for the blinkLED.hex file. This is the execute file for our microcontroller! Program this file to AT89S52 by your favorite programmer. An alternative is an ISP cable.

Figure 2.1.8

By using an ISP cable, we will get rid of the labour to move the 40-pin mcu to-and-fro our target board and the programmer. Just plug in the ISP cable as shown in Figure 2.1.9, launch the download software and click program. No hassle to relocate the mcu. ISP cable is available from us or directly from Atmel.

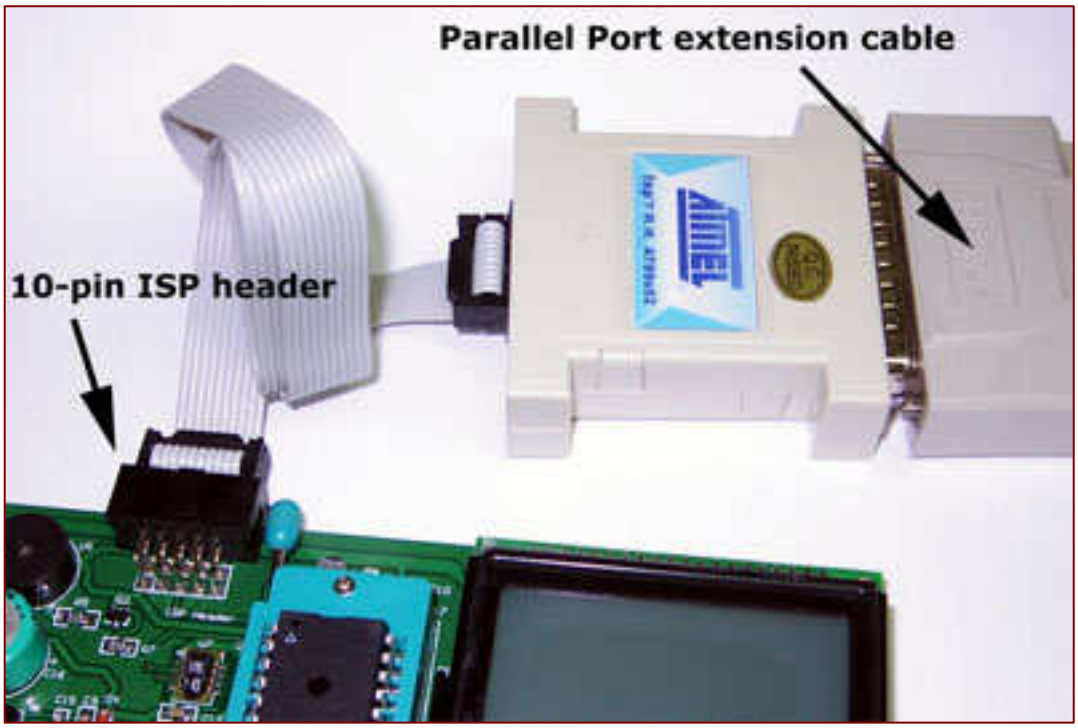

Figure 2.1.9

After code download, look at LED D3 and see it is blinking. Adjust the parameter in DelayMs( ) to see what happen to the blink rate.

There is a serious problem with this program. When the code line DelayMs(50) is executed, all processing power is spent inside the for-loop of DelayMs(). To get around this loop-hole, we can make use of interrupt in section 2.2.

2.2 Blink a LED by hardware delay

The source code is located under cd:\src\chp2\src2_2\. Project name is hblinkLED.uv2.

```
#include <reg52.h>

#define ON    0
#define OFF   1

sbit LED = P2^0;

void Tmr0Init();

void Tmr0() interrupt 1                                          (1)
{
        unsigned char cnt;
        cnt++;
        if(cnt==5)     {LED = ~LED; cnt=0;}                      (2)
        TH0 = 0xDC;                                              (3)
        TL0 =  0x00;
}

void Tmr0Init(void)                                              (4)
{
        TMOD=TMOD&0xF0;                                          (5)
        TMOD=TMOD|0x01;
        TH0 = 0xDC;
        TL0 =  0x00;
        ET0 =  1;                                                (6)
        TR0 =  1;                                                (7)
}

void main(void)
{
        Tmr0Init();                                              (8)
        EA = 1;                                                  (9)
        for(;;)
        {
                ;                                                (10)
        }
}
```

Listing 2.2

Line (1) is the interrupt service routine.

Line (2) toggles the LED between ON and OFF by using a counter 'cnt' to count a multiple of Timer 0 overflow period. In this case, LED=~LED when cnt reaches 50ms (5x10ms), 10 ms being the Timer 0 overflow period.

Line (3) reloads Timer 0 (TH0 & TL0). Because crystal freq=11.0592MHz => 1 machine cycle at 1.085usec. Number of cycles to delay 10ms = 10000/1.085 = 9216.
T0 in 16-bit mode => T0 preload with a value (65536-9216)=56,320, i.e. 0xDC00

Line (4) is the timer 0 initialization code.

Line (5) initializes Timer 0 to 16-bit timer mode.

Line (6) enables Timer 0 interrupt.

Line (7) turns on Timer 0.

Line (8) calls Tmr0Init() in main() for Timer 0 initialization.

Line (9) enables global interrupt; otherwise, Tmr0Init() becomes useless.

Line (10) is an infinite loop, "hanging" the program there forever. However, the program is still running illustrated by a blinking LED.

How to make sure the blink period is really 50msec? How to make sure the Timer 0 interrupt period is 10msec?

1. Use a good oscilloscope to probe the P2.0 pin.
2. If there is no oscilloscope, use Performance Analyzer built-in µVision 2 as below.

Under **Debug** -> **Start/Stop Debug Session** or **Ctrl+F5** to bring up the Debug Windows.
Under **Debug** -> **Performance Analyzer**->**Setup Performance Analyzer**->double click on main and **Define** to put main() as the Current PA Ranges->Close the window

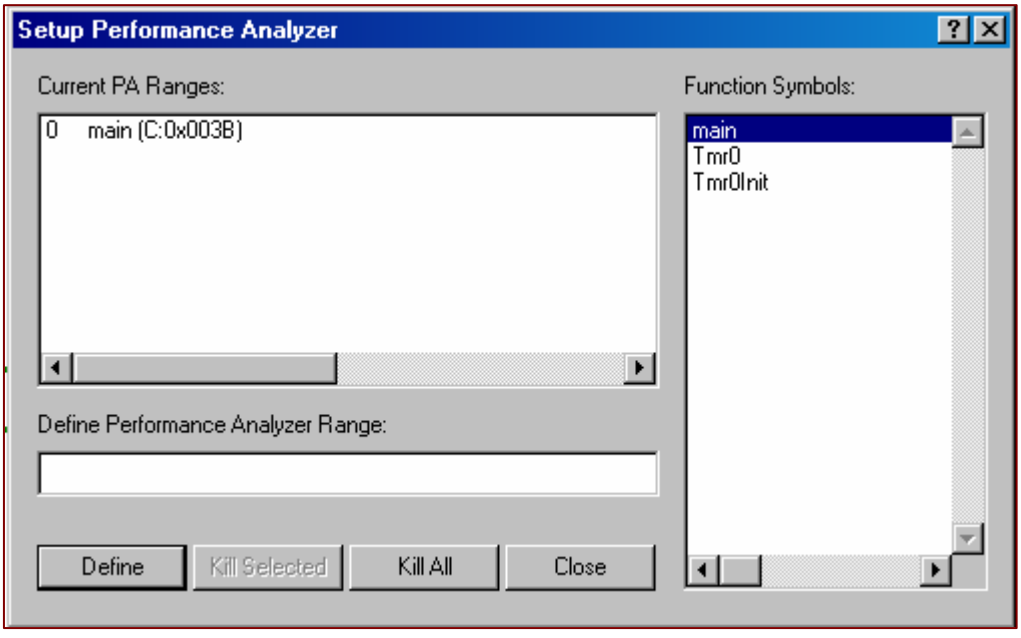

Figure 2.2.1

Double click at the code line cnt++ in Tmr0() interrupt 1 to set a break point there.

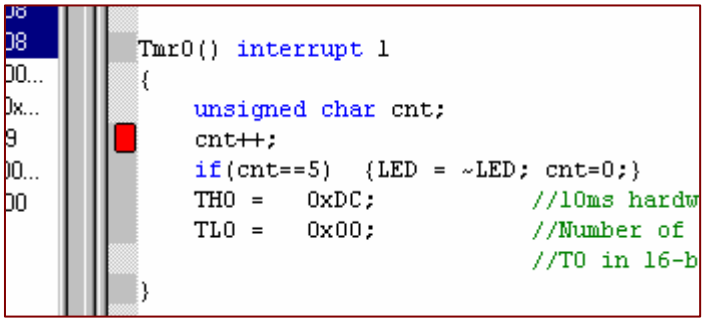

Figure 2.2.2

Look at Port2.0 by clicking on **Peripherals**->**I/O-Ports**->**Port2**.

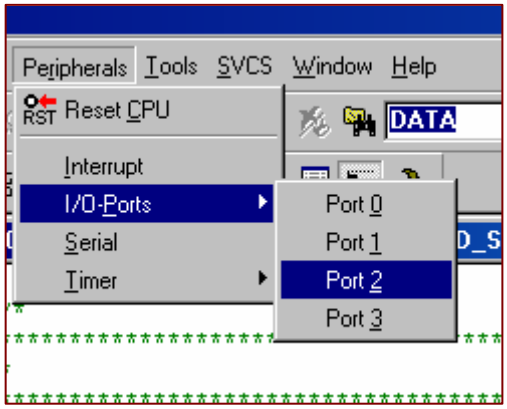

Figure 2.2.3

Under **View**->select **Performance Analyzer Window** to bring it up.
Press F5 once. Double click on main to update the total time data (10.015ms) and this data increment on each F5 key press, until 5 key presses then bit 0 of P2 toggle.

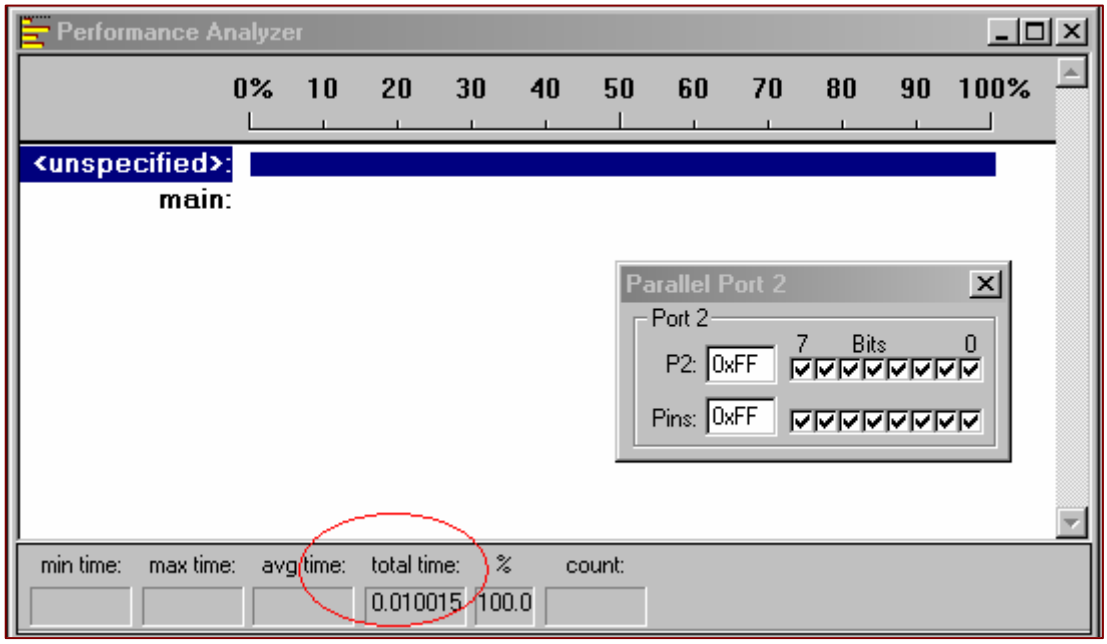

Figure 2.2.4

⇒ Chapter 3