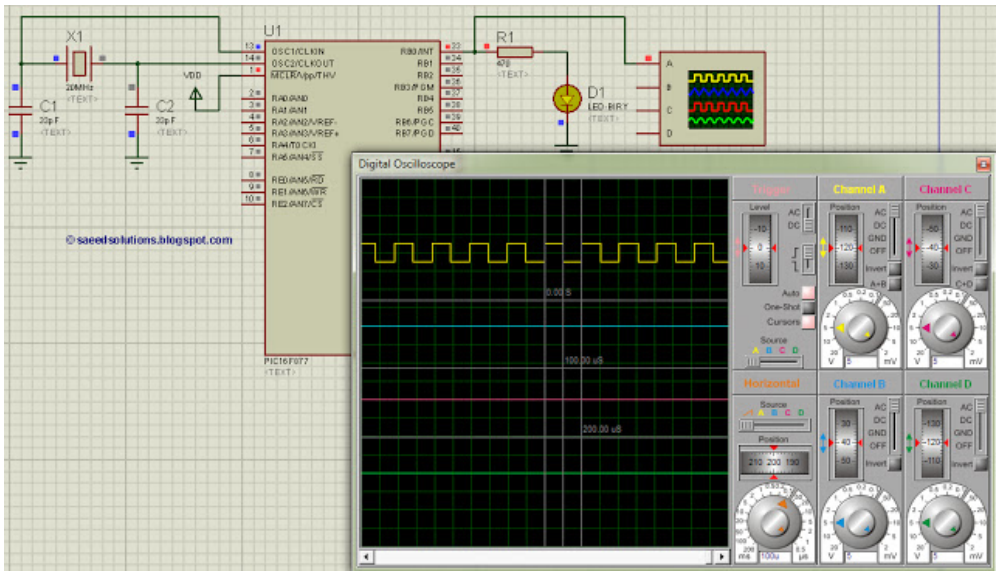
PIC16F877 timer0 code + Proteus simulation

This PIC16F877 microcontroller tutorial answers the question,
" How to use timer0 of PIC16F877 and how to handle its interrupts? "

Using PIC16 simulator (*Proteus*) you can verify this PIC timer0 code and change it according to your needs. This code is written in C language using MPLAB with HI-TECH C compiler. You can download this code from the '*Downloads*' section at the bottom of this page.

It is assumed that you know how to blink an LED with PIC16F877 microcontroller. If you don't then please read this page [http://saeedsolutions.blogspot.com/2012/11/pic16f877-led-blinking-code-proteus.html] first, before proceeding with this article.

The following diagram (made in *Proteus*) shows the PIC microcontroller circuit diagram.



[http://3.bp.blogspot.com/-EUKrngZ0cgc/ULWwd1-LvbI/AAAAAAAACD8/eKxMiXbBFJw/s1600/PIC16F877+Timer0+CCT.bmp]

Figure 1. PIC16F877 timer0 simulation

In this circuit, PIC16F877 is running on external crystal of 20MHz value. RB0 pin is toggled every time timer0 expires and executes it's ISR[1] code. In the above figure, it is clear that after approximately every 100usec, RB0 pin is toggled i-e timer0 expires. You can easily change this value in the code.

Code

The main function code is shown below.

```
// Main Function
void main(void)
{
    TRISB0 = 0;          // Make RB0 pin output
    RB0    = 0;          // Make RB0 low

    InitTimer0();        // Initialize timer0

    while(1)
    {
    }
}
```


[http://2.bp.blogspot.com/-xOC66wDwMuk/ULWxtm14ehI/AAAAAAAACEE/v3XCmzytlLI/s1600/PIC16F877+TIMER0+MAIN+CODE.bmp]

Figure 2. Main function for timer0 of PIC16F877

In the main function, firstly TRISB0 is made zero to make RB0 pin an output pin, also RB0 pin is made zero. After that, *InitTimer0()* function is called which initializes timer0. Rest of the work is done in timer0 interrupt service routine. Every time timer0 expires RB0 pin is toggled.

The code used to initialize timer0 is shown below.

```
void InitTimer0(void)
{
    // Timer0 is 8bit timer, select T0CS and PSA to be zero
    OPTION_REG &= 0xC0;     // Make Prescalar 1:2

    T0IE = 1;               // Enable Timer0 interrupt
    GIE = 1;                // Enable global interrupts
}
```


[http://2.bp.blogspot.com/-RqpupTVx-FA/ULWyrmWyEDI/AAAAAAAACEM/F-jR1oS-CRE/s1600/PIC16F877+INIT+TIMER0+CODE.bmp]

Figure 3. InitTimer0 function code for PIC16F877

In this function, OPTION_REG is initialized to make timer0 prescalar to be 1:2. Timer0 is an 8bit timer, so it expires after reaching a value of 255. When timer0 prescalar is made 1:2 then it means that timer0 value will increment after every two clock cycles. Since PIC16F877 is running at 5MIPS[2] speed, this means that timer0 will expire after every 256*2/5 = 102 usec[3]. T0IE bit enables timer0 interrupts and GIE bit enables global interrupts.

Timer0 interrupt service routine code is shown below.

```
void interrupt ISR(void)
{
    if(T0IF)  //If Timer0 Interrupt
    {
        // XTAL frequency is 20Mhz, so CPU frequency is 5Mhz.
        // RB0 toggles after every 256 cycles, so it completes
        // its cycle in 2*256 = 512 cycles. A prescalar of 1:2
        // is being used, so a frequecny of 5M/(512*2) = 4.9KHz
        // is being generated at RB0 pin.

        RB0 = ~RB0; // Toggle RA0 pin
        T0IF = 0;   // Clear the interrupt
    }
}
```


[http://3.bp.blogspot.com/-p3HlwrCSEyw/ULWzq6q7a4I/AAAAAAAACEU/3samBhLfgBg/s1600/PIC16F877+TIMER0+ISR+CODE.bmp]

Figure 4. Timer0 interrupt service routine

Whenever timer0 expires, then an interrupt is generated which executes the function shown above in the *Figure 4*. In this function, RB0 pin is toggled every time and timer0 interrupt flag is cleared. This means that in *Figure 1* when ever RB0 pin toggles than timer0 interrupt has occurred.

This is just an example code for timer0 of PIC16F877 microcontroller. You can modify it to fulfill your circuit requirements. You can leave your comments in the comment section below.

Notes and References

[1] Interrupt service routine i-e when ever timer0 expires, it generates an interrupt and the code which is executed due to that interrupt is called it's ISR.

[2] I am using external crystal of 20MHz for this PIC16F877 microcontroller. Since PIC microcontrollers execute every instruction in 4CPU cycles, so effectively this PIC16F877 is running at 20MHz/4 = 5 MIPS (million of instructions per second) speed. You can use any crystal value from 4 to 20MHz with PIC16F877 in HS mode.

[3] You can confirm this value from *Figure 1*.

Downloads

Timer0 code for PIC16F877 was compiled in MPLAB v8.85 with HI-TECH C v9.83 compiler and simulation was made in Proteus v7.10. To download code and *Proteus* simulation click here [https://dl.dropbox.com/u/22020714/PIC16F877%20Timer0%20Code.rar] .

Further Reading Suggestions

You may also like,

- How to set PIC16F877 configuration bits in code ? [http://saeedsolutions.blogspot.com/2012/11/how-to-set-pic16f877-configuration-bits.html]

- PIC16F877 UART code + Proteus simulation [http://saeedsolutions.blogspot.com/2012/11/pic16f877-uart-code-proteus-simulation.html]
- PIC16F877 LCD code + Proteus simulation [http://saeedsolutions.blogspot.com/2012/11/pic16f877-lcd-code-proteus-simulation.html]
- PIC16F877 PWM code + Proteus simulation [http://saeedsolutions.blogspot.com/2012/11/pic16f877-pwm-code-proteus-simulation.html]

Posted 28th November 2012 by Saeed Yasin

Labels: PIC micro-controller Projects, PIC16F877 Projects

14 View comments