

Introduction to PIC Micro

Author: Tom Varghese
tomvarghese@ymail.com

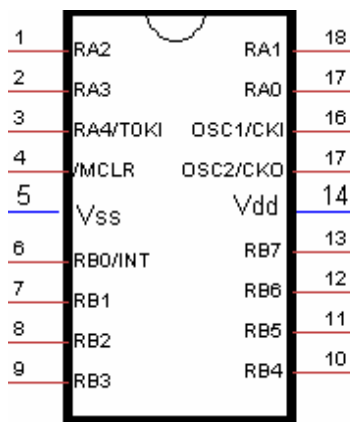
If you are not aware of Microcontrollers in this micro age, and still alive as an electronics student, it is better to stop your studies. Nowadays everything surrounded with us contains at least one microcontroller. Usually it is written as uC (spoken as mue C). There are lot of microcontroller vendors are there, but leading vendors are motorolla, PIC, AVR, Maxim, Freescale etc. For the simplicity's sake, I am starting with PIC.

A lot of PIC Microcontrollers are exist and new are producing day by day. Again simplicity's sake I opt for PIC16F84A. This is a low cost, simple one suitable for beginners. PIC microcontrollers are introduced my Microchip. They provide an IDE named MPLAB for free. Datasheet for PIC16F84A is downloadable. MPLAB uses an assembler called MPASM for the development of PIC. Since is much low level, we use a C compiler for the same. Here we are using sourceboost IDE. It is free for download. But updating requires money. A free version is only needed for us. After compilation, we should program the intel hex code created by sourceboost to our PIC Micro. For this purpose we use again a free software named Picprog. So we are starting our mission. Before, you require the following stuff:

- [PIC16F84A Micro](#)
- [Sourceboost IDE](#)
- [Picprog programmer](#)

Some passive components such as resistor, capacitor, LEDs. Details later.

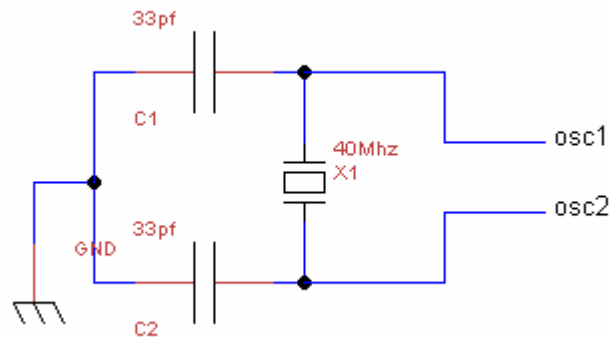
Ok, I am starting without preamble. A basic pin diagram of PIC16F84A is as follows.



and pin17 as shown.

A constant 5V supply is needed for smooth working of the micro. Therefore a voltage

It can be powered by a 5V battery. Positive terminal is connected to Vss and negative terminal is connected to Vdd. A system clock is needed which is the heart beat of a Microcontroller. A crystal oscillator is used for this purpose. Other oscillators like RC oscillators can also be used. An oscillator network should be connected to pin16



series network using IC7805 should be added.

connect all circuits properly. Then PIC Micro is ready to work. But we didn't tell what to do. This process is called programming. So we want to program the PIC. PIC16F84A contains 1024 bytes of program memory(flash memory) and 68 bytes of RAM and 64 bytes of data EEPROM. The program is written in the program memory and instructions are fetched directly from the program memory. The actual work is done on the primary memory or RAM. But all data will be killed when the power is switched OFF. Therefore another non volatile memory called EEPROM (Electrically Erasable Programmable Read Only Memory) is added to store the data for future use. These memory ratings are varied for different Micros.

First PIC Program

We write program in sourceboost C. The program code can be downloaded from here. The C code is shown below. This is a simple program to blink an externally connected LED.

```
#include <system.h>

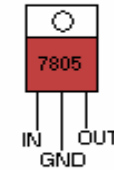
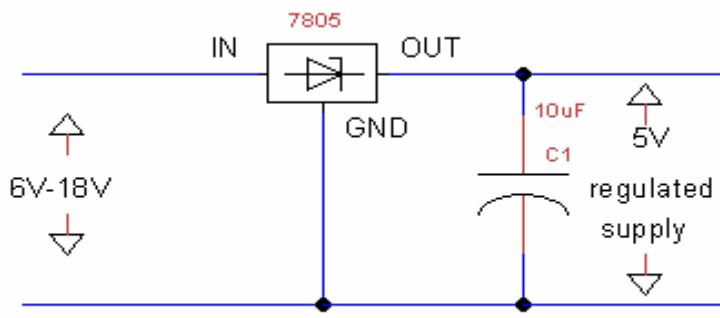
//Target PIC16F84 configuration word
#pragma DATA _CONFIG, _PWRTE_ON & _WDT_OFF & _HS_OSC & _CP_OFF

//Set clock frequency
#pragma CLOCK_FREQ    4000000

void main( void )
{
    //Configure port A
    trisa = 0x00;
    //Configure port B
    trisb = 0x00;

    //Initialize port A
    porta = 0x00;
    //Initialize port B
    portb = 0x00;

    while(1)
    {
        portb.0=1;
        delay_ms(100);
        portb.0=0;
        delay_ms(100);
    }
}
```



```
}
}
```

Step
by
step
analysis
of
the
Prog

ram

```
#include <system.h>
```

This is common for all programs. No need to learn deep.

```
//Target PIC16F84 configuration word
#pragma DATA _CONFIG, _PWRTE_ON & _WDT_OFF & _HS_OSC & _CP_OFF
```

```
//Set clock frequency
#pragma CLOCK_FREQ 4000000
```

The first line is the configuration word of PIC16F84A. Here `_PWRTE_ON` means power up timer is on. So it will take a moment to set up all when power switch is ON. In the next line, clock frequency is set to 4000000Hz. I.e., micro expect 4MHz frequency Oscillator for it's work. Therefore we should place a 4MHz crystal in the above shown circuit.

Next is main function. This function is actually executed in the run time. Therefore the work which is done by the PIC should written in `main()`. To start writing the actual program, we should tell that which of the 13 I/O pins are input and which are output. One pin can never state as input and output at the same time.

```
//Configure port A
trisa = 0x00;
//Configure port B
trisb = 0x00;
```

Here a number is assigned starting with 0x (in 0x00) 0x means it is a hexadecimal number. I.e., 00 is hexadecimal number. The equivalent binary number is 0000 0000.

Here 0 corresponds to the output pin and 1 corresponds to the input pin.

Here is an example:

PortB:	0	1	2	3	4	5	6	7
In/out:	out	out	in	in	out	out	in	In
Corresponding Binary value	0	0	1	1	0	0	1	1

Therefore the corresponding in/ out pattern of portB is 0011 0011. Converting into Hex, The number is 33. Then it can be represented as 0x33. Then we can write `trisb=0x33`

Similarly we can write `trisa` for portA.

In the above code all pins of the micro is configured as output pins.

```
//Initialize port A
    porta = 0x00;
//Initialize port B
    portb = 0x00;
```

Here we assigns all pins of `porta` and `portb` as 0(off). Consider `portb=0x55`. Then convert it into binary, 0101 0101.

Binary value	0	1	0	1	0	1	0	1
OFF/ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON

OFF means there is no voltage occurs at the corresponding pin while ON means there occur a constant 5V output. In the above shown program all pins are assigned as zero and there is no output. We can individually control the pins of `Porta` and `Portb` (which is configured as output) by the statement `porta.0`, `porta.1`,....That means, If you want to assign 4th output pin of `portb` to 1, simply add the statement `portb.3=1`; (note: here `portb.0` is the first pin). This method is implemented in the next stage of the program.

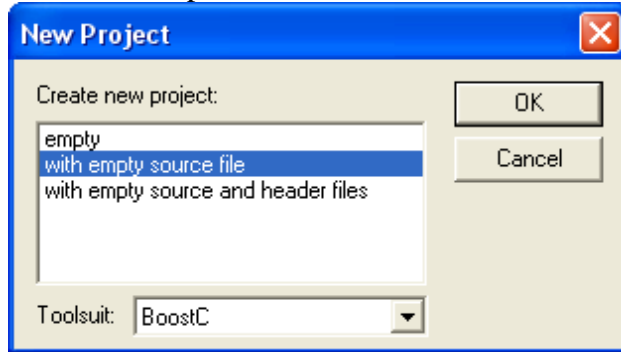
```
while(1)
{
    portb.0=1;
    delay_ms(100);
    portb.0=0;
    delay_ms(100);
}
```

Here the statements in the braces of `while (1)` repeats infinitely (until power is off). `Portb.0=1` makes the first pin of `portb` changes to 1 (ie 5V output). `delay_ms(100)` make a delay of 100ms of time. That is the micro do nothing useful for 100ms. After 100ms, it executes the next statement `portb.0=0`. Again the first pin of `portb` becomes off. Again a delay of 100ms for off time. After 100ms, it again goes to `while (1)`. Thus the statements in the braces of `while (1) {}` is repeated. If we connect an LED in series with a 330Ω resistor across `portb.0` (i.e. RB0 – 6th pin of the micro - refer pin diagram shown in the first page) it continuously become ON and OFF (Both ON time and OFF time of 100mS).

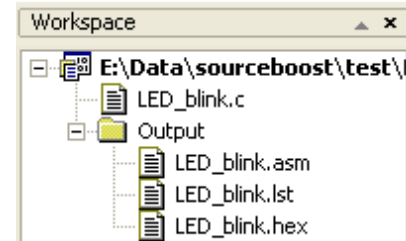
Compiling and building

The above written programs human interpretable language, but a PIC micro cannot. We should convert it into the PIC interpretable binary language. For this purpose first of all open sourceboost IDE. Then select `project>new` .Then select in which

directory, you want to build a project. Select any directory and give name 'LED_blink'. Then click 'open'. A window will occur as shown.

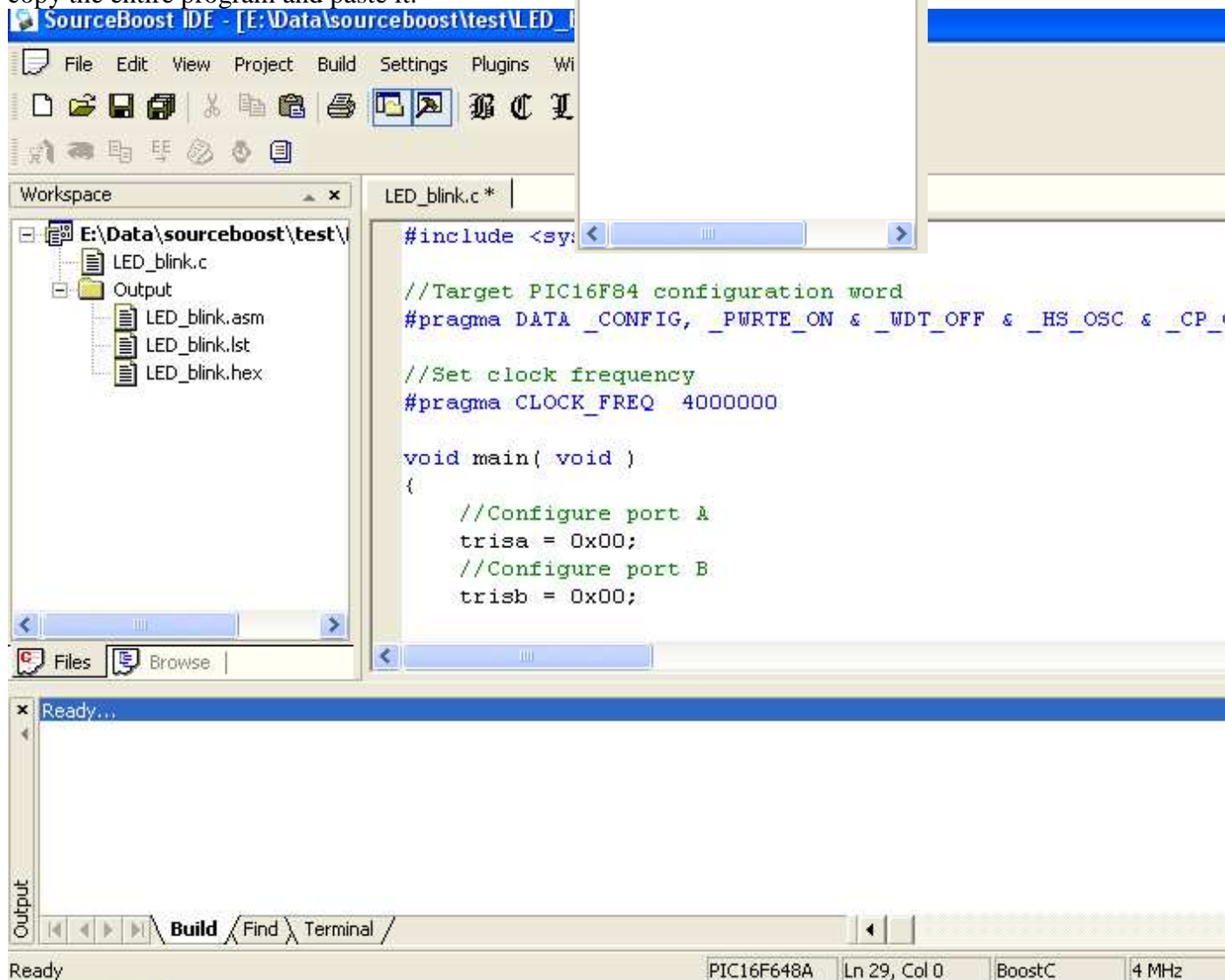


Select 'with empty source file' and click OK. Then click LED_blink.c in the workspace (Left side).



Then some writings and

are displayed on the white screen. Just delete it copy the entire program and paste it.



Then opt for build>compile. If you done all correctly, you get a success message and else an error message in the bottom side of the window.

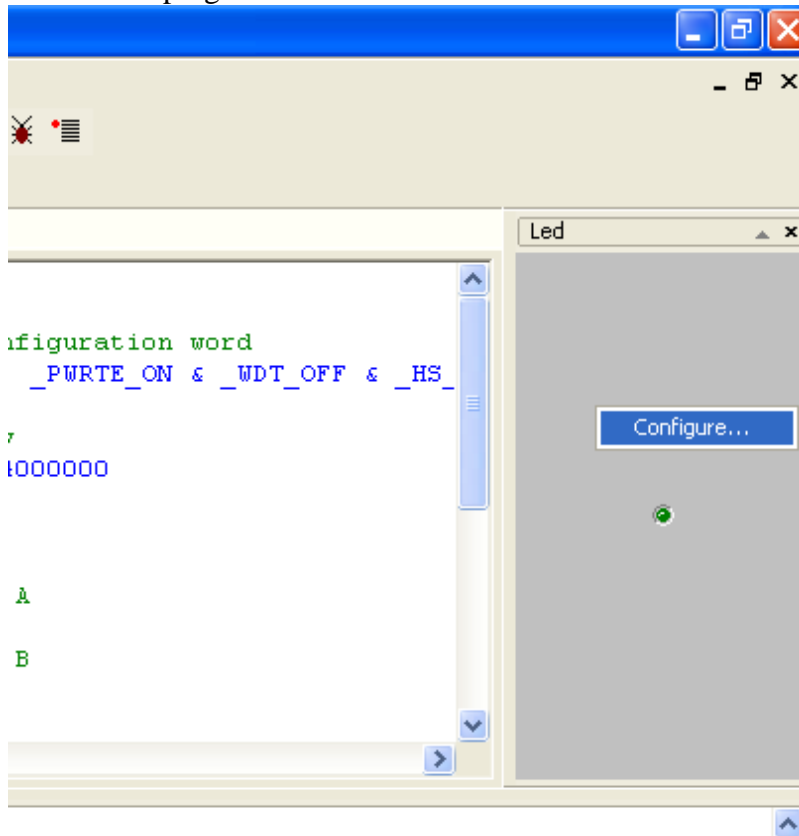
Then select build>build (or F7). Again a success result will be shown.

Yes, You did it. You converted your code into machine interpretable language.

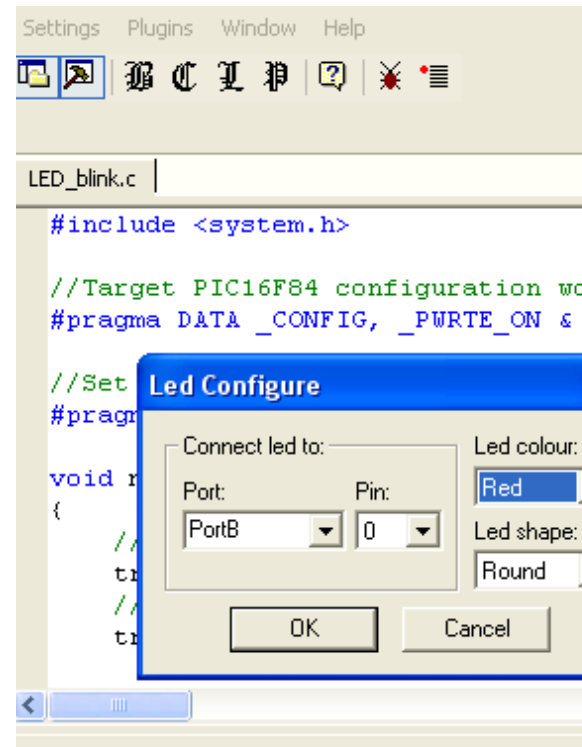
Debugging

Debugging is the process of clearing the run time errors. Here you can see a blinking of LED virtually on your monitor without the help of any hardware. Select 'Debugger' tool from the toolbar (below menu bar) (second from right).

Then select plugins>LED.



Then right click on plug in and choose 'configure'. A dialogue box will appear as shown.

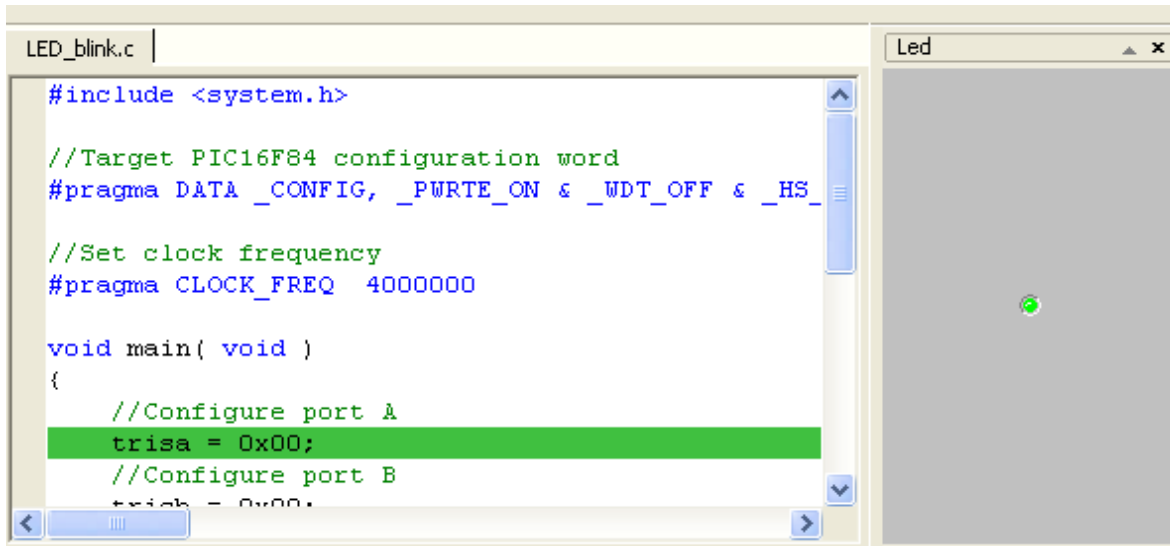


(0.8%), free:254 bytes (99.2%),
loc:95 bytes

Then select port as PortB an pin as 0. Then click OK.

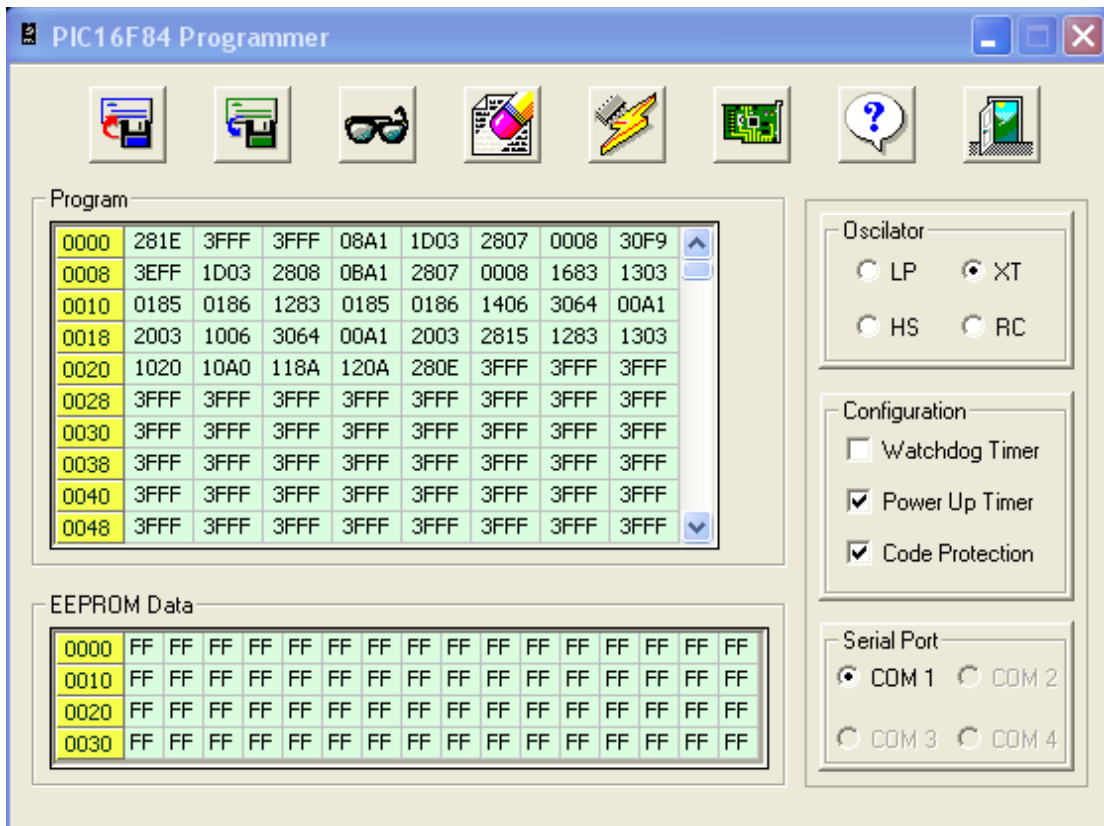
Press F5 to run the program. Then you are able to see the a blinking LED on the screen.

Is it really thrilling? 😊

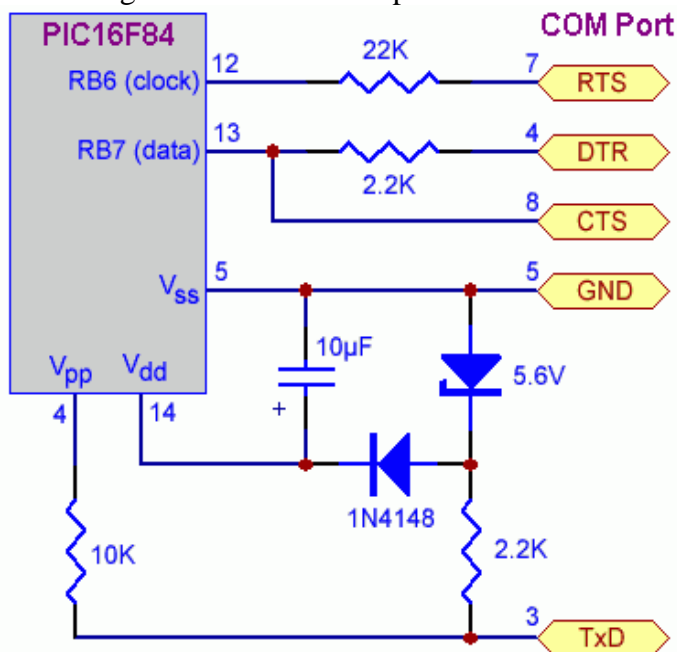


Programming Your PIC

If you successfully done all above steps, you are able to program the stuff into your microcontroller. You should have hardware to interface with PIC. You can directly program from sourceboost or MPLAB but hardware is costly. There are so many free programmers are available on web. Many can program multiple PICs, AVR, 8051 etc. Examples are ponyprog, icprog etc. But for the simplicity's sake, I opt for picprog- a very simple programmer. It's hardware contains a few low cost components only. Hope you downloaded and installed picprog. A screenshot of Picprog is given below.



There are a few buttons on the top of the window. Click 'open' button and change directory to our project's directory. Here you can find a file named LED_blink.hex. Open this file with picprog. Then some hexadecimal numbers are displayed on the program section. Remaining memory is filled with binary 1's (ie 3FFF in hexadecimal). Oscillator is selected to XT which means crystal oscillator and check 'Power Up Timer' in the Configuration section. You can program your micro only if your PC has a serial port. Almost all IBM model PCs have 2 serial ports. Select your serial port. If you have more than one serial port, you must do some trial error method and mark it which is COM1 and which is COM2 etc. The hardware is simple circuit containing a few low cost components. The circuit diagram is shown below.



I am not including more details about serial port which is not a part of this article. Many articles about serial port is available on the web. Finally you want to program the code. Connect the hardware to COM port and insert PIC properly. Then click 'program' button. In a few seconds, program will be completed and you are ready to use your PIC.

Implementing your program


```

#include <system.h>

//Target PIC16F84 configuration word
#pragma DATA _CONFIG, _PWRTE_ON & _WDT_OFF & _HS_OSC & _CP_OFF

//Set clock frequency
#pragma CLOCK_FREQ 4000000

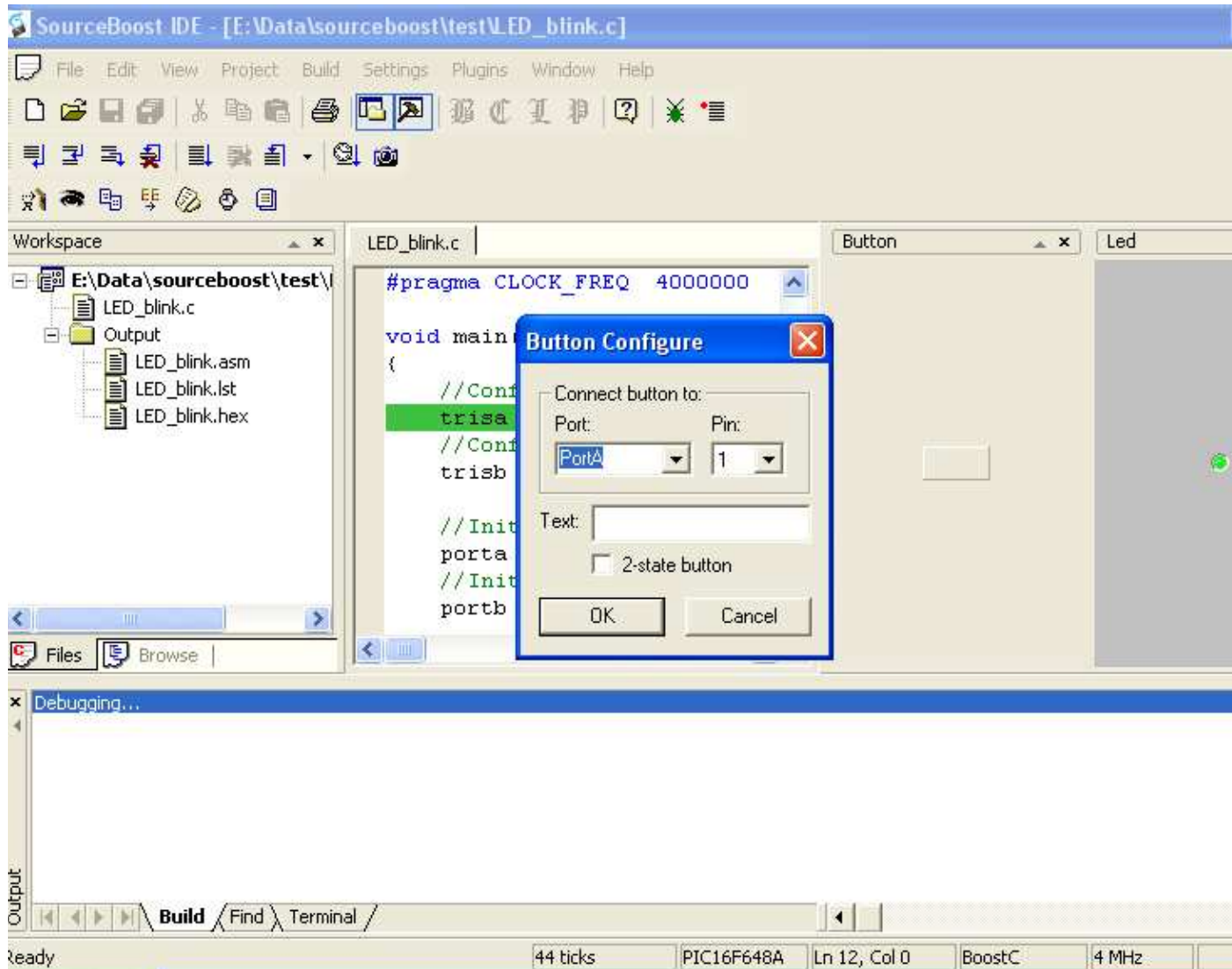
void main( void )
{
    //Configure port A
    trisa = 0x02;
    //Configure port B
    trisb = 0x00;

    //Initialize port A
    porta = 0x00;
    //Initialize port B
    portb = 0x00;

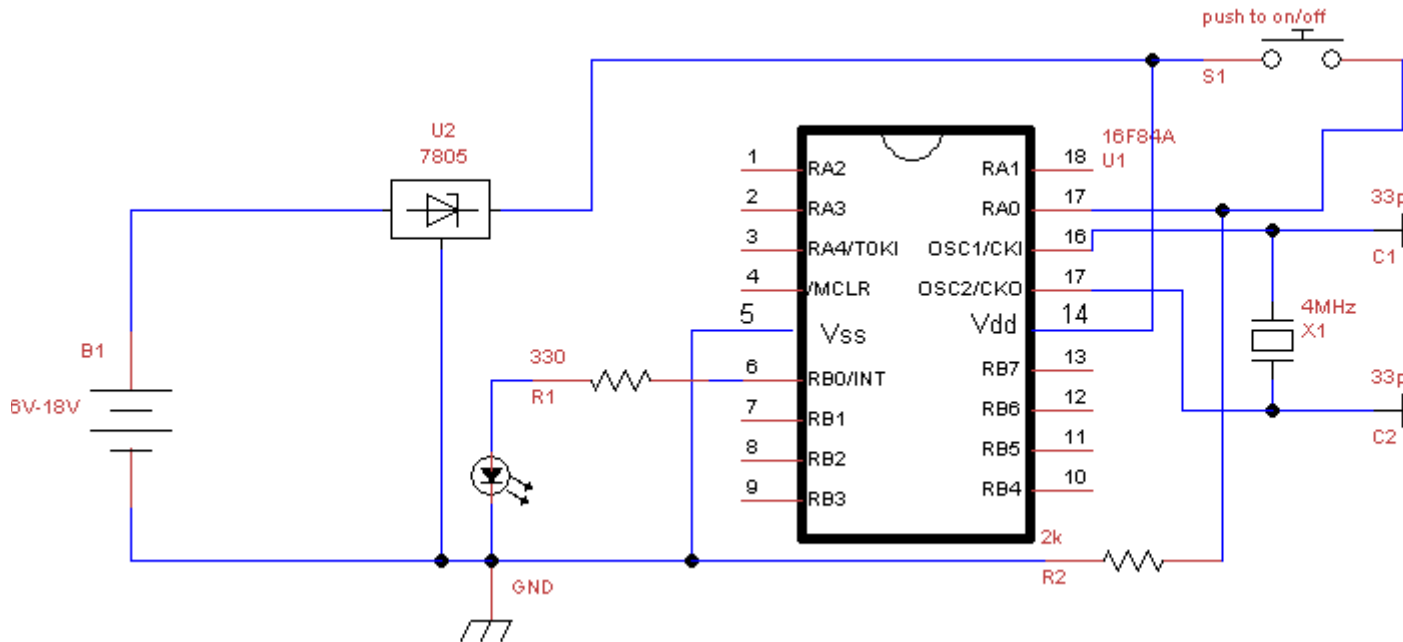
    while(1)
    {
        if(porta.1==1)
        {
            delay_s(2);
        }
        else
        {
            portb.0=1;
            delay_ms(100);
            portb.0=0;
            delay_ms(100);
        }
    }
}

```

There is only a slight change from the previous program so that you can easily understand what is going on. Compilation and building are done as in the previous program. In debugging section, you need a special plug-in called 'button' .Select debug mode (as in the previous project) and select plugins>Button. Configure it for porta.1 (shown in figure below.)



Press F5 to Run. Then you see a blinking LED. Then press click button. Then LED becomes OFF for 2 seconds and again starts to blink. The circuit is also slightly changed.



Do it yourself

Try yourself to make your own programs. After doing these simple programs, you will get ability to make your own complex programs.

- ❖ A program to simulate logic gates AND, OR, NOT (one LED and Two buttons)
- ❖ A program to simulate an 8 bit chaser (8 LEDs)
- ❖ A program to show numbers 1 to 9 LED display (You should know the basics of LED display)

Now you are stepped in to the world of microcontrollers. There are other microcontrollers like AVR, 8051, Maxim, Freescale etc. One is not much different than other. If you are able to do with one, you can easily learn others also. Any comment about this article is welcome.

tomvarghese@gmail.com

Additional information you required

Another microcontrollers [AVR](#), [8051](#), [Basic stamp](#)
[Serial](#) and [parallel](#) ports controlling
[Basics of C programming](#)
[Electronics circuit designing](#)