MPLAB8 and Flow Charts

ECE 376 Embedded Systems

Jake Glower - Lecture #3

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Boot Loaders

Programming a PIC Chip

- Option 1: External Programmer such as PICStart-Plus
- Option 2: Boot Loader

Boot Loader

- Program on the PIC Chip
- Located at 0x000 to 0x799
  - Code must be offset by 0x800
- Watches the serial port
- If it sees a carriage return within 3 seconds of reset
  - It clears out the old program
  - It waits for a new program to be send via the serial port

Assembler in MPLAB8

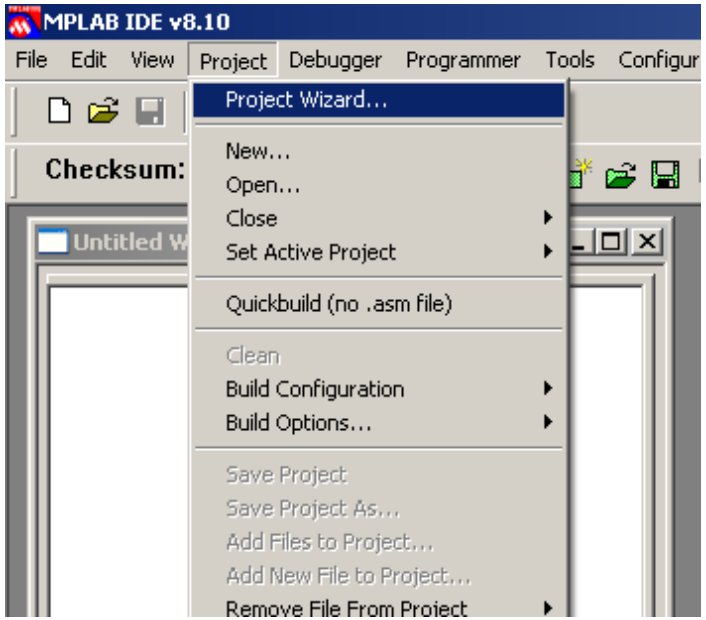
Step 1. Create a new directory.

- Located on thumb drive works well
- X:\ECE376\ASM\Count

Step 2. Start MPLAB8

Step 3. Click on File New Project

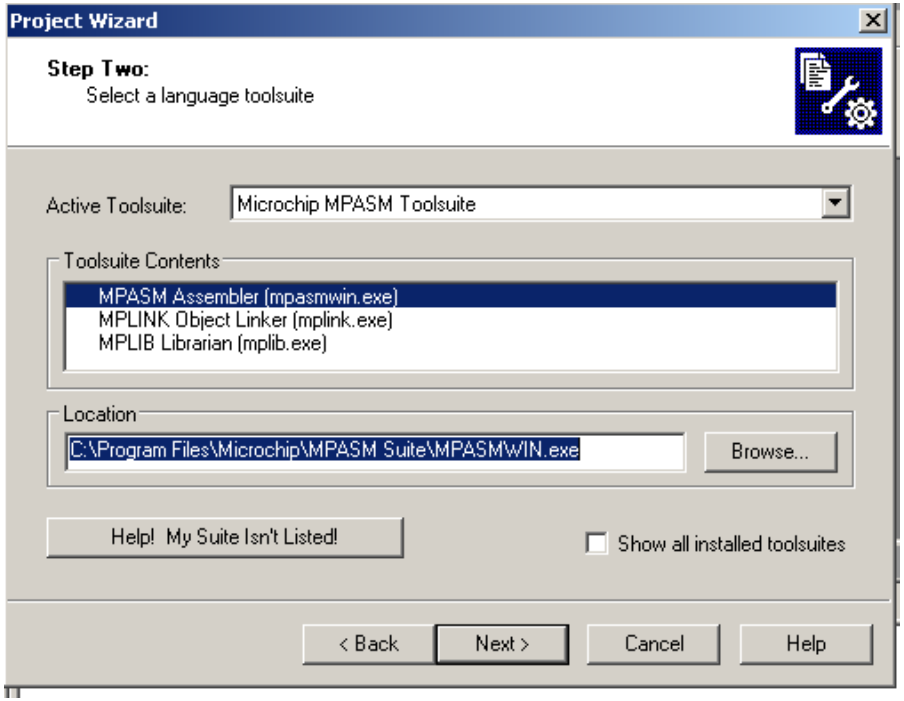
Project Wizard if this is a new project



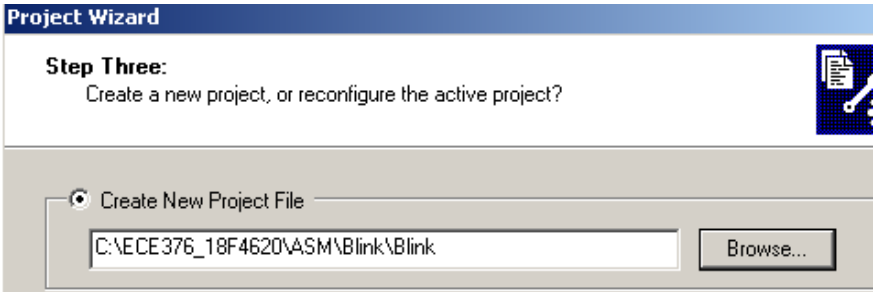
This takes you through the process of starting a new project (i.e. a new program). Click OK

Device = PIC18F4620 (next)

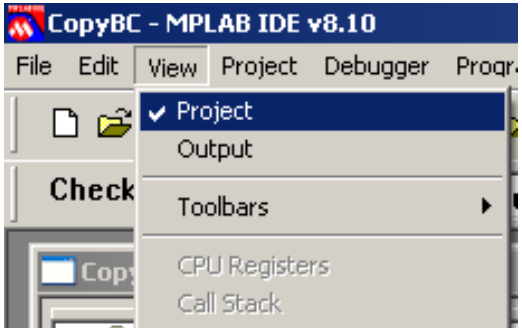
Program Language is MPASM



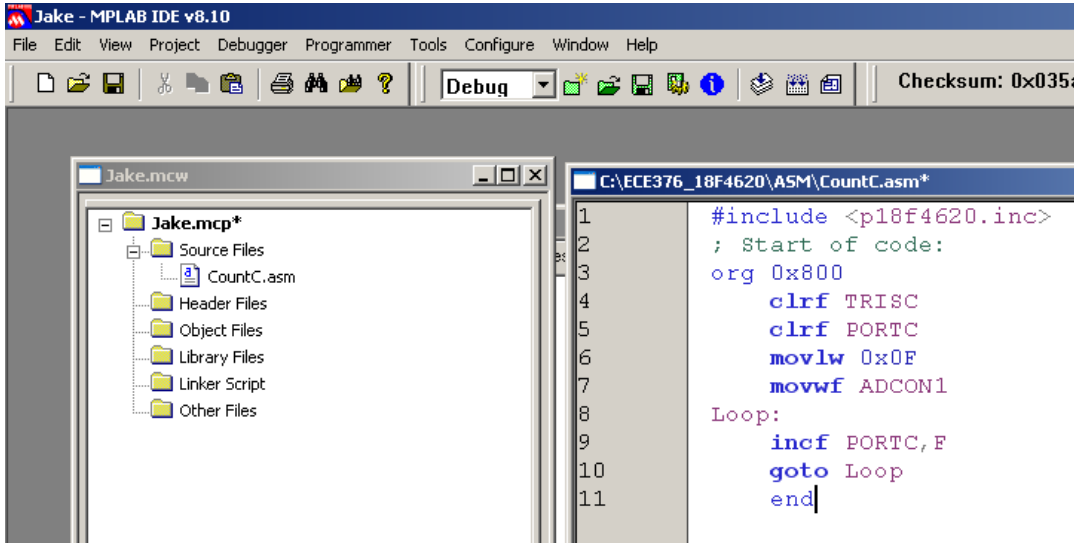
Directory for Files: Select your directory



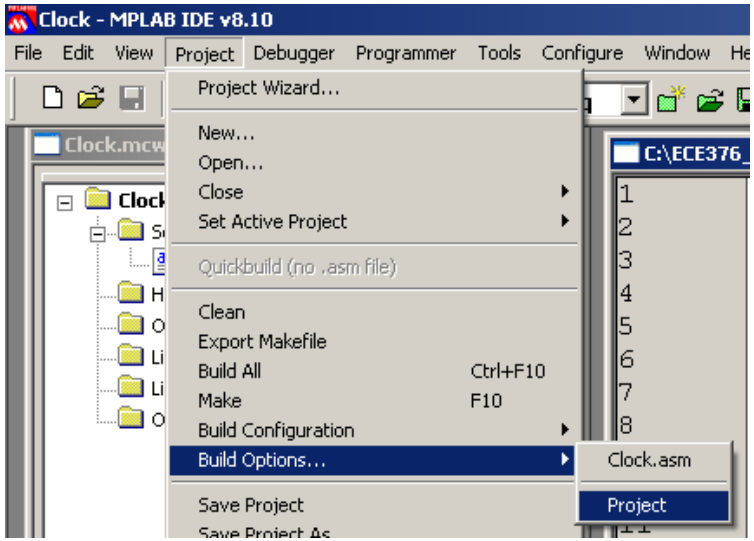
Click on View Project



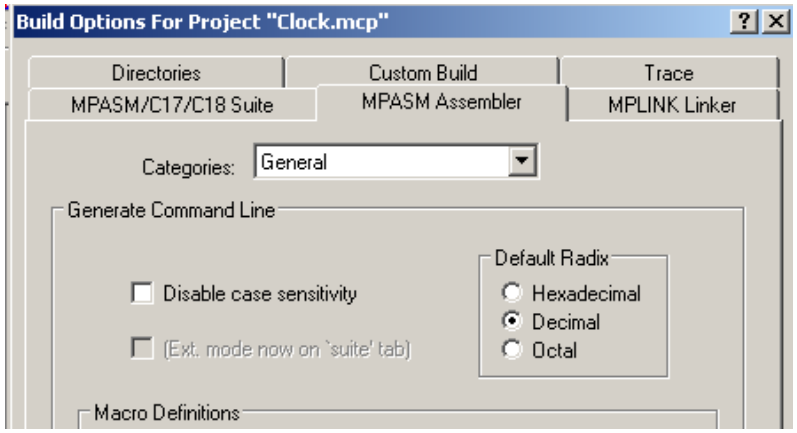
You should see the following:



Change the default to decimal. Click on Project Build Options Project

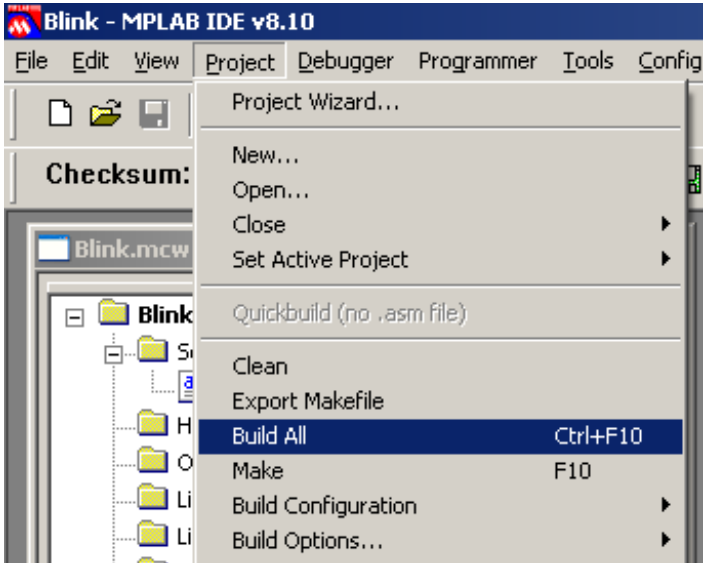


Click on MPASM anc select Decimal. This results in numbers like 100 representing 100 base 10.



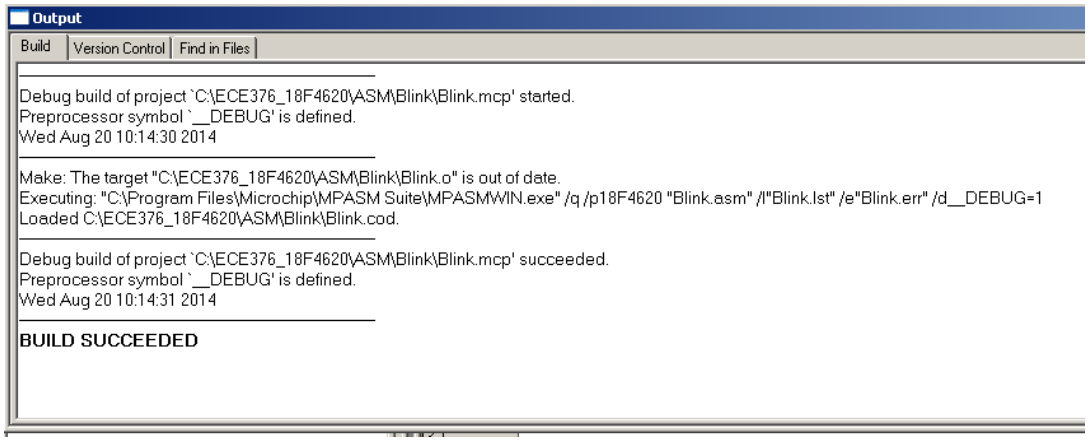
The source file is what you compile.

- If this is blank, right click on Source File and select the ASM file you wish to compile.
- If you don't have an ASM file yet, select File New edit a file, and save it as .ASM

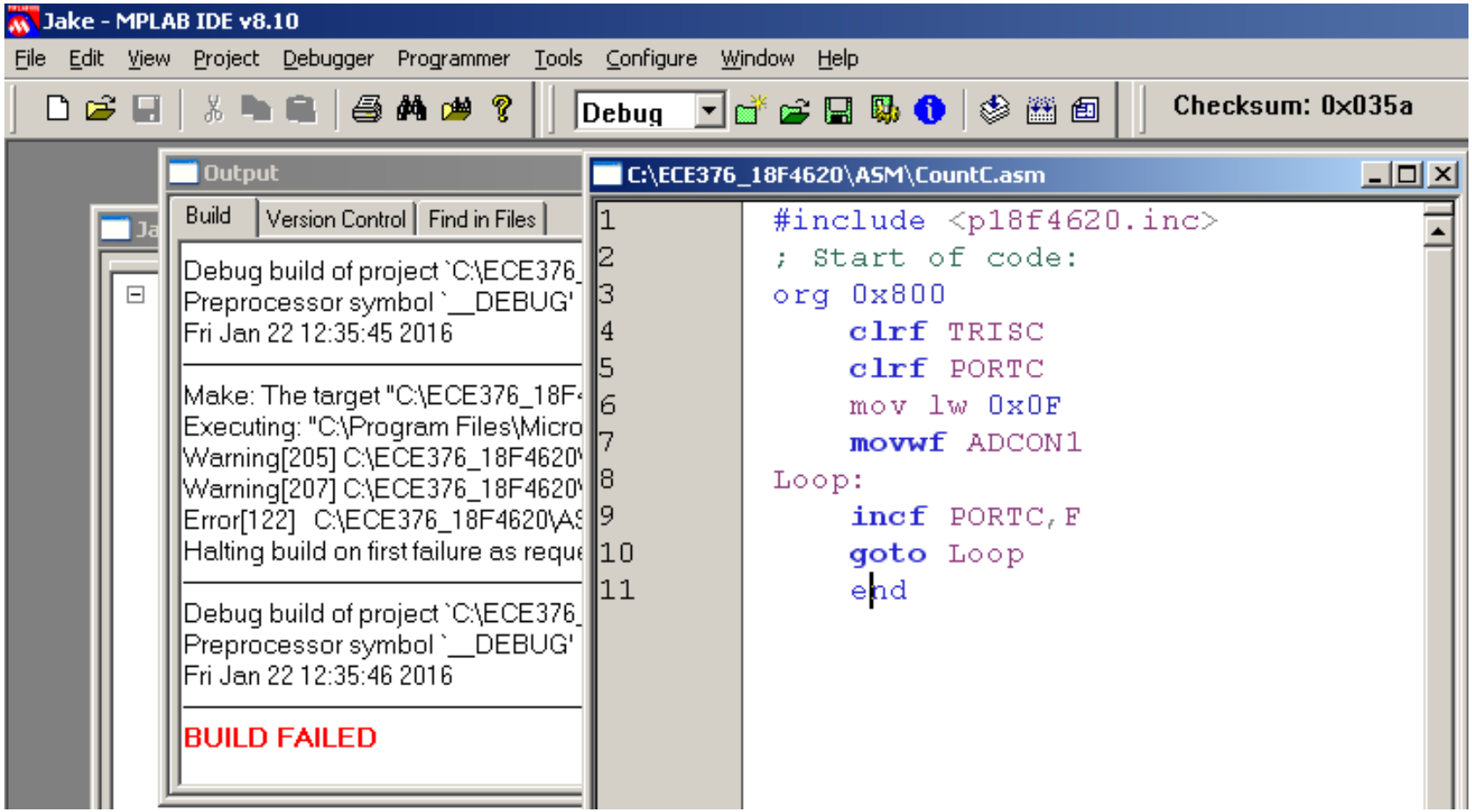


To compile your code, click on Project Bulid All (or hit key F10)

If your program compiles correctly, you get the message 'Succeed'



If there is an error in your code (such as a space in line 13 below), you will get an error message along with a notice which line has a problem



If you want to see what your program looks like, click on View Program Memory

Jake - MPLAB IDE v8.10 - Program Memory

File Edit View Project Debugger Programmer Tools Configure Window Help

Debug

Checksum: 0xfb31

- Project
- Output
- Toolbars
- CPU Registers
- Call Stack
- Disassembly Listing
- EEPROM
- File Registers
- Flash Data
- Hardware Stack
- LCD Pixel
- Locals
- Memory
- Program Memory
- SFR / Peripherals
- Special Function Registers
- Watch
- 1 Memory Usage Gauge

Program Memory

Line	Address	Opcode	Label	Disassembly
1020	07F6	FFFF		NOP
1021	07F8	FFFF		NOP
1022	07FA	FFFF		NOP
1023	07FC	FFFF		NOP
1024	07FE	FFFF		NOP
1025	0800	6A94		CLRF TRISC, ACCESS
1026	0802	6A82		CLRF PORTC, ACCESS
1027	0804	0E0F		MOVLW 0xf
1028	0806	6EC1		MOVWF ADCON1, ACCESS
1029	0808	2A82	Loop	INCF PORTC, F, ACCESS
1030	080A	EF04		GOTO Loop
1031	080C	F004		NOP
1032	080E	FFFF		NOP
1033	0810	FFFF		NOP
1034	0812	FFFF		NOP
1035	0814	FFFF		NOP

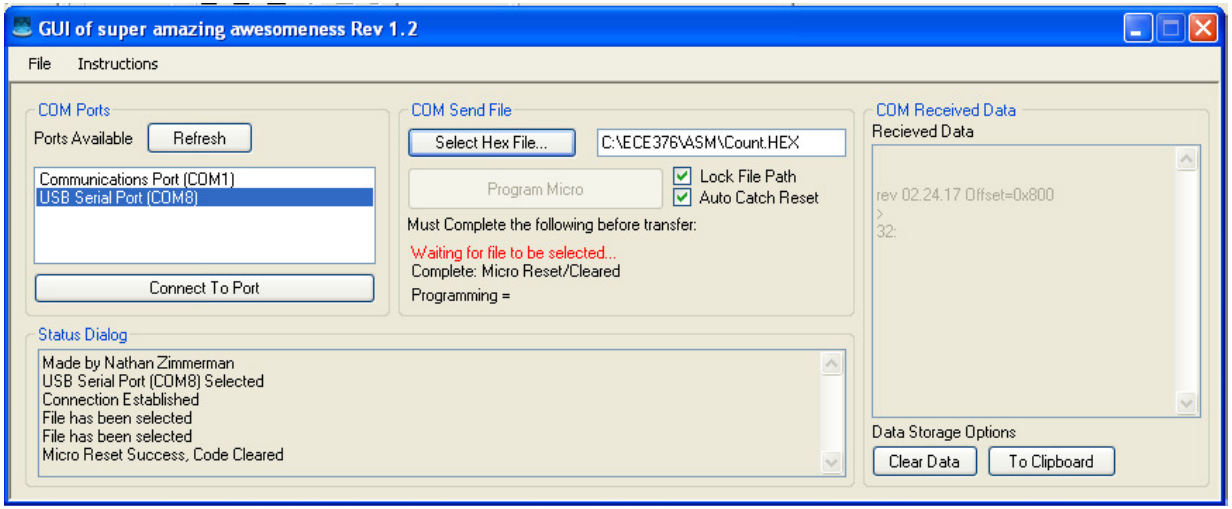
The program is stored in the file .HEX This is a text files that contains the program in machine language (the OP-Code above)

C:\ECE376_18F4620\ASM\Blink\Blink.HEX

```
:020000040000FA
:10030000926A936A946A956A966A150EC16E822AF9
:0403100087EF01F082
:00000001FF
```

To download your code to your PIC board,

- Power up your PIC board (i.e. plug it in)
- Connect the serial cable to a PC
- Run a terminal program, such as Hyperterminal or PIC_Flash_Tool
- Select the USB Serial Port (COM number varies)
- Select the .hex file to download (must be lower case letters)
- Hit RESET on your PIC board.
- Wait for Program Micro to light up
- Click on Press Program Micro.



Flow Charts:

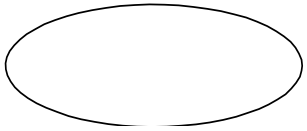
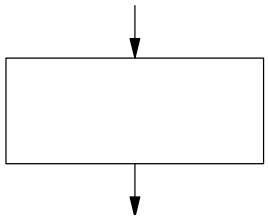
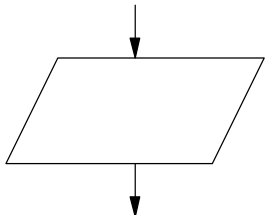
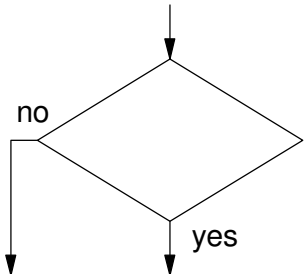
Graphical way to explain how a program works

- Keep it simple (less than 20 blocks), but
- Keep it informative (more than one block)

It also helps if you follow a few rules:

- Flow charts should start at the top of the page
- The program execution should move down towards the bottom of the page
- There should be a single exit point

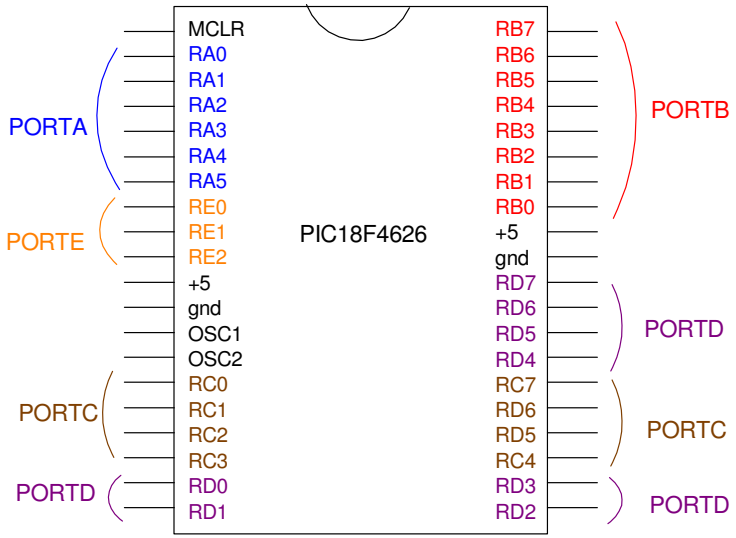
Flow Chart Symbols

Symbol	Image	Meaning
Oval		Start / End of routine or program
Rectangle		Function or operation
Parallelogram		Input / Output
Diamond		Decision

PIC I/O

The PIC18f4620 chip has 33 I/O lines split into five ports:

	PORTA	PORTB	PORTC	PORTD	PORTE
Pins	2..7	33..40	15..18, 24..26	19..22, 27..30	3
Binary Input	5	8	8	8	3
Binary Output	5	8	8	8	3
Analog Input	5	5	-	-	3



Setting Up I/O Ports for Binary I/O

Three registers are associated with each port

- PORTx: Defines whether the pin is 0V (0) or 5V (1)
- TRISx: Defines whether the pin is input (1) or output (0)
- LATx: "Read-modify-write operations on the LATC register read and write the latched output value for PORTC."

In addition, you need to initialize ADCON1 to 15. This sets all I/O pins to binary.

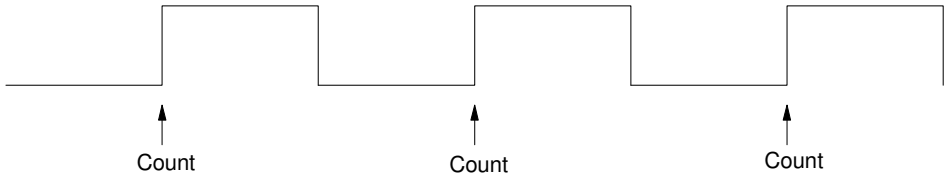
```
movlw 0x0F
movwf ADCON1
```

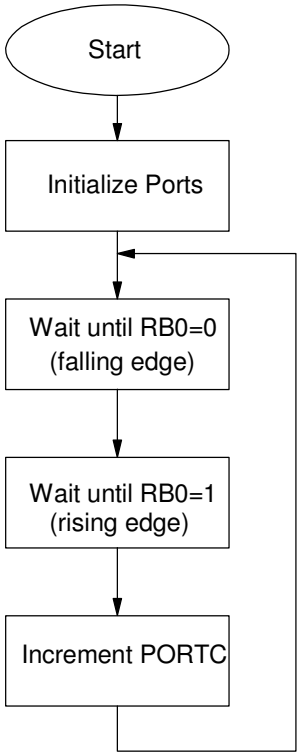
Count Edges on RB0

```
#include <p18f4620.inc>
    org 0x800
    clrf  TRISA
    movlw 0xFF

    movwf TRISB
    clrf  TRISC
    clrf  TRISD
    clrf  TRISE
    movlw 0x0F
    movwf ADCON1
    clrf  PORTC

Loop1:
   btfsc PORTB,0
   goto  Loop1
Loop2:
   btfss PORTB,0
   goto  Loop2
   incf  PORTC,F
   goto  Loop1
   end
```



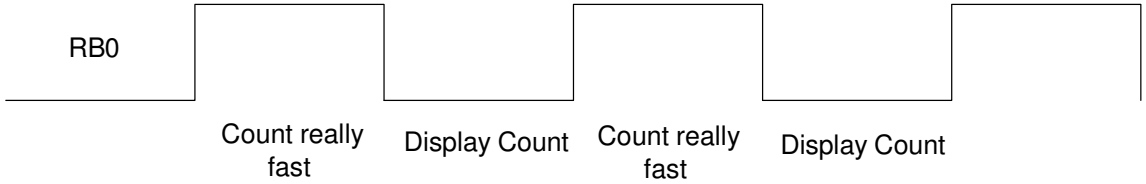


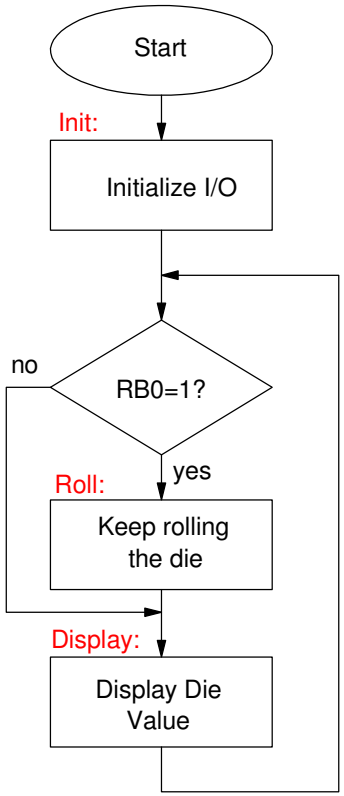
Random Number Generator

```
#include <p18f4620.inc>
DIE   EQU   0
   org 0x800

   clrf  TRISA
   movlw 0xFF
   movwf TRISB
   clrf  TRISC
   clrf  TRISD
   clrf  TRISE
   movlw 0x0F
   movwf ADCON1

Main:
   btfsc PORTB,0
   incf   DIE,W
   andlw  0x07
   movwf  DIE
   movwf  PORTC
   goto   Main
```





Top Down Programming:

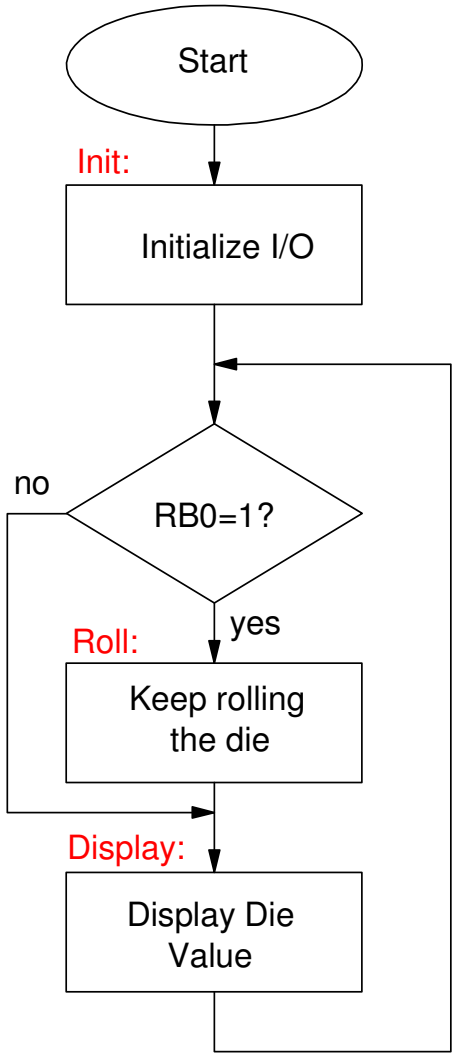
```
#include <p18f4620.inc>

; Variables

DIE    EQU    0

; --- Main Routine ---

       org 0x800
       call   Init
Main:
       btfsc PORTB,0
       call   Roll
       call   Display
       goto   Main
```



```
; ---  Subroutines ---

Init:
      clrf  TRISA
      movlw 0xFF
      movwf TRISB
      clrf  TRISC
      clrf  TRISD
      clrf  TRISE
      movlw 0x0F
      movwf ADCON1
      return

Roll:
      incf   DIE,W
      andlw  0x07
      movwf  DIE
      return

Display:
      movf   DIE,W
      movwf  PORTC
      return

      end
```