INT Interrupts:

Goal:

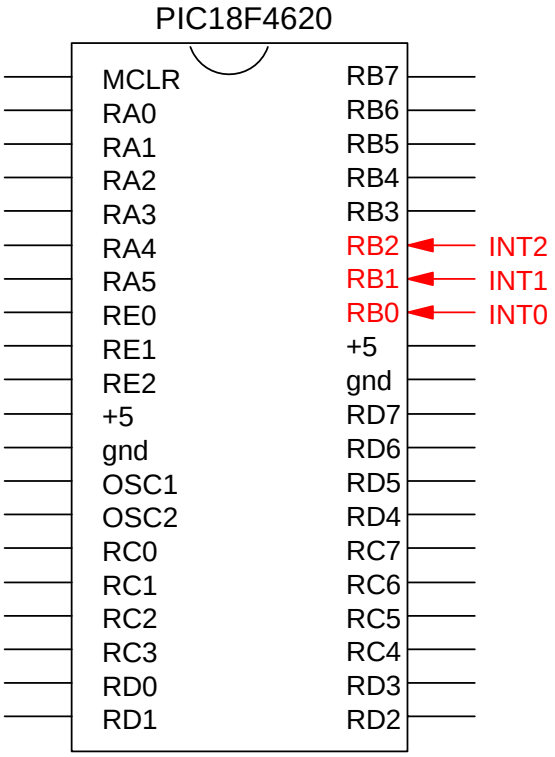
- Trigger an interrupt when a rising edge or falling edge is detected.

Why:

- Efficiency. Don't waste time looking for an edge. Only call a routine when it happens.
- Speed: Accurate to 1 clock (200ns). Start dealing with the event 7 clocks after it happens (time to call the interrupt subroutine).
- Interface an optical encoder to a PIC. This is a digital version of a potentiometer where the outputs are 1/0 as you turn the knob (upcoming).

How: Hardware

- Make sure your device outputs 0V / 5V
- Connect to RB0/INT0 pin on the PIC, or
- Connect to RB1/INT1 pin on the PIC, or
- Connect to RB2/INT2 pin on the PIC



How: Software:

1. Set up RB0/RB1/RB2 as an input pin

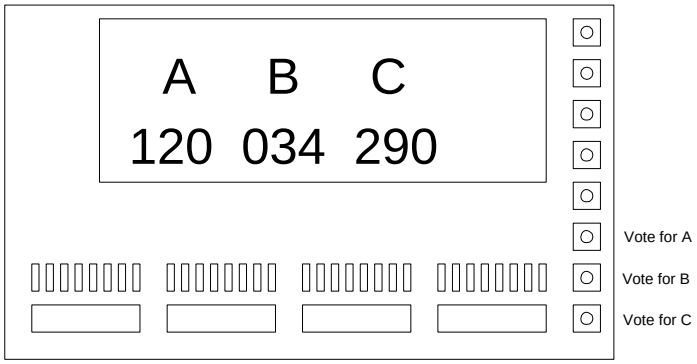
2. Set up the conditions for the interrupt
   - INTEDGx = 1: interrupt on a rising edge
   - INTEDGx = 0: interupt on a falling edge

3. Enable the INT interrupt
   - INTxE = 1: enable INTx interrupts
   - INTxE = 0: disable INTx interrupts

4. Enable all interrupts:
   - GIE = 1: enable all interrupts

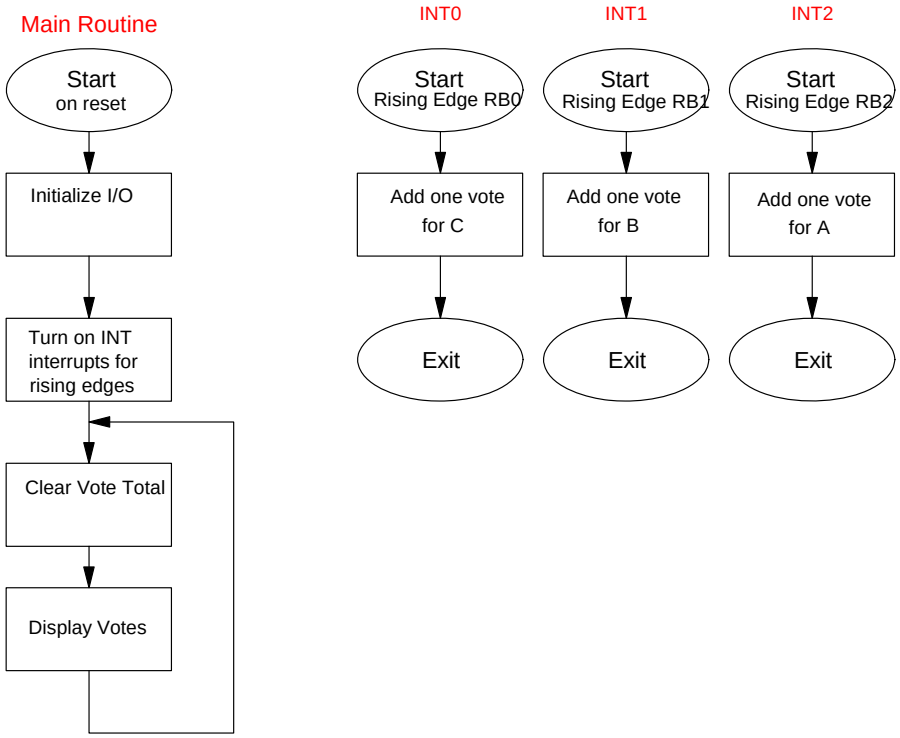
Address	Register Name	Bit							
		7	6	5	4	3	2	1	0
0xF81	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
0xF93	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
0xFF0	INTCON3	INT2IP	INT1IP	–	INT2IE	INT1IE	–	INT2IF	INT1IF
0xFF1	INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	–	TMR0IP	–	RBIP
0xFF2	INTCON	GIE	PEIE	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF

Example 1: Voting Machine. Count how many times you press buttons RB0, RB1, and RB2. Display the total number of button presses on the LCD display.



Hardware: Nothing is needed. Push buttons on PORTB are already there.

Software (Vote.C):



Increment three counters inside the interrupt service routine

```
// Global Variables
unsigned int N0, N1, N2;

void interrupt IntServe(void) {
   if (INT0IF) {
      N0 += 1;
      INT0IF = 0;
      }
   if (INT1IF) {
      N1 += 1;
      INT1IF = 0;
      }
   if (INT2IF) {
      N2 += 1;
      INT2IF = 0;
      }
   }
```

Set up INT0, 1, and 2 for rising edge interrupts.

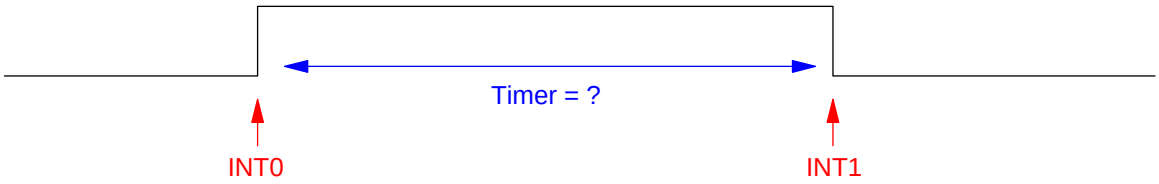
```
// initialize INT0 interrupts for rising edges
   INT0IE = 1;
   TRISB0 = 1;
   INTEDG0 = 1;

// initialize INT1 interrupts for rising edges
   INT1IE = 1;
   TRISB1 = 1;
   INTEDG1 = 1;

// initialize INT2 interrupts for rising edges
   INT2IE = 1;
   TRISB2 = 1;
   INTEDG2 = 1;
```

(Optional): Display the count total on the LCD display

```
// Main Loop

   while(1) {
      LCD_Move(1,0);   LCD_Out(N0, 0, 3);
      LCD_Move(1,5);   LCD_Out(N1, 0, 3);
      LCD_Move(1,10);  LCD_Out(N2, 0, 3);
      Wait_ms(100);
      }
```

Example 2: Measure Pulse Width

Measuring the time from the rising edge to the falling edge of a pulse



Hardware: Connect the pulse to both pins RB0 and RB1

Software:

- Set up Timer0 for measuring time to 100ns (PS = 1)
- Set up INT0 for a rising edge interrupt
- Set up INT1 for a falling edge interrupt

Interrupt Service Routines:

- Timer0: Interrupts every 2^{16} clocks. Keep track of time as a 32-bit number.
- INT0: Record the time of the rising edge
- INT1: Record the time of the falling edge and the time difference

```
unsigned long int TIME, T0, T1, dT;

void interrupt IntServe(void)
{
   if (TMR0IF) {
      TIME += 0x10000;
      TMR0IF = 0;
      }
   if (INT0IF) {
      T0 = TIME + TMR0;
      INT0IF = 0;
      }
   if (INT1IF) {
      T1 = TIME + TMR0;
      dT = T1 - T0;
      INT1IF = 0;
      }
   }
```

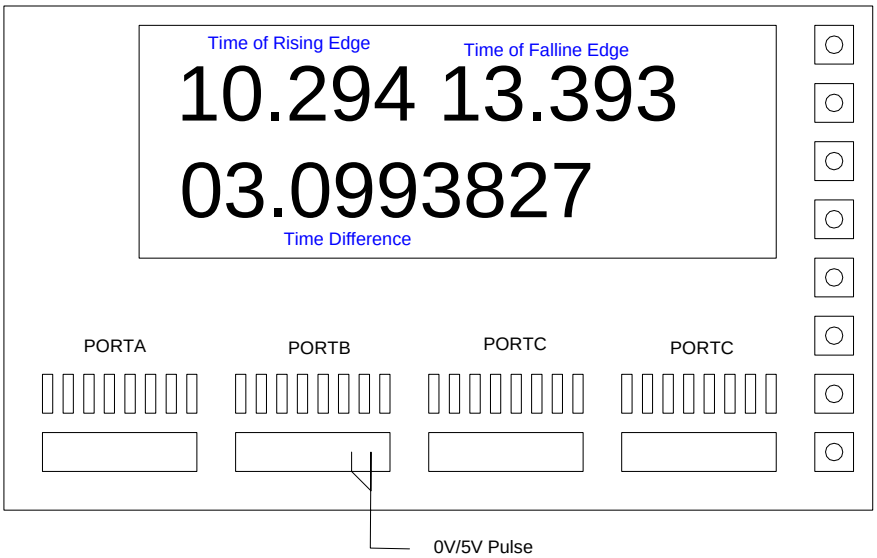
Interrupt Initialization: Set up

- Timer0 for PS=1 (100ns) and interrupt every 216 clocks
- INT0: Rising edge interrupt on RB0
- INT1: Falling edge interrupt on RB1

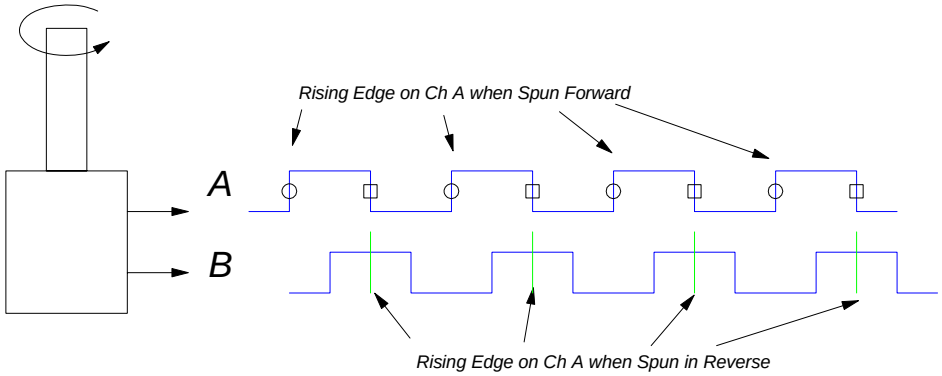
```
// set up Timer0 for PS = 1
   T0CS = 0;
   T0CON = 0x88;
   TMR0ON = 1;
   TMR0IE = 1;
   TMR0IP = 1;
   PEIE = 1;

// Turn on INT0 interrupt
   INT0IE = 1;
   TRISB0 = 1;
   INTEDG0 = 1;

// Turn on INT1 interrupt
   INT1IE = 1;
   TRISB1 = 1;
   INTEDG1 = 0;
```

The main routine then displays the time of the rising edge, falling edge, and pulse width

```
while(1) {
   LCD_Move(0,0);  LCD_Out(T0/10000, 3, 5);
   LCD_Move(0,8);  LCD_Out(T1/10000, 3, 5);
   LCD_Move(1,0);  LCD_Out(dT, 7, 10);
   }
```



Example 3: An rotary pulse generator (alias optical encoder, digital potentiometer) outputs two square waves as it is turned, each 90 degrees out of phase from the other. Assume there are 500 pulses per rotation (channel A had 500 pulses when you spin it all the way around. Ditto for channel B.)



Hardware:

You need to count edges on A and B. Connect these to the INTx inputs.

- Connect channel A to RB0 (INT0)
- Connect channel B to RB1 (INT1)

Software (take 1):

Let's just look at the rising edges on channel A. This will give 500 counts per rotation.

Note that

- when moving right to left, channel B is low on the rising edges (circles above)
- when moving right to left, channel B is high on the rising edges (squares above).

This lets you keep track of distance (number of edges on channel A) and direction.

Code: Just initialize INT0 interrupts for rising egdges:

```
// initialize INT0 interrupts
   INT0IE = 1;
   TRISB0 = 1;
   TRISB1 = 1;
   ADCON1 = 15;
   INTEDG0 = 1;    // rising edges
```

The interrupt service routine is

```
void interrupt IntServe(void) {
   if (INT0IF) {
      if (RB1) ANGLE += 1; else ANGLE -= 1;
      INT0IF = 0;
      }
   }
```

Software (take 2):

Let's count both rising and falling edges of channel A. This will give 1000 counts per rotation. The logic is

- If you found a rising edge and RB1 is low, you're moving forward
- If you found a falling edge and RB1 is high, you're moving forward
- Otherwise, reverse

Code: Just initialize INT0 interrupts for rising egdges:

```
// initialize INT0 interrupts
   INT0IE = 1;
   TRISB0 = 1;
   TRISB1 = 1;
   ADCON1 = 15;
   INTEDG0 = 1;    // rising edges

void interrupt IntServe(void) {
   if (INT0IF) {
      if (RB1 != INTEDG) ANGLE += 1; else ANGLE -= 1;
      INTEDG0 = !INTEDG0;
      INT0IF = 0;
      }
   }
```

Software (take 3):

Let's count both rising and falling edges of channel A and B. This will give 2000 counts per rotation.

Code: Initialize INT0 and INT1 interrupts:

```
// initialize INT0 interrupts
   INT0IE = 1;
   TRISB0 = 1;
   TRISB1 = 1;
   ADCON1 = 15;
   INTEDG0 = 1;    // rising edges

// initialize INT1 interrupts
   INT1IE = 1;
   INT1IP = 1;
   INTEDG1 = 1;    // rising edges
```

The interrupt service routine now looks for both interrupts:

```
void interrupt IntServe(void) {
   if (INT0IF) {
      if (RB1 != INTEDG0) ANGLE += 1; else ANGLE -= 1;
      INTEDG0 = !INTEDG0;
      INT0IF = 0;
      }
   if (INT1IF) {
      if (RB0 == INTEDG1) ANGLE += 1; else ANGLE -= 1;
      INTEDG1 = !INTEDG1;
      INT1IF = 0;
      }
   }
```