

## [1]RBO INT

- RBO, must be initialized for input.
- The interrupt source must be set to take place either on the falling or the rising edge of the signal.[using option reg]
- (INTF in the **INTCON Register**) must be initially cleared
- **INTE** & **GIE** bit in the **INTCON Register** must be enables

### OPTION\_REG REGISTER (ADDRESS 81h, 181h)

|       |        |       |       |       |       |       |       |
|-------|--------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1  | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| RBPU  | INTEDG | T0CS  | T0SE  | PSA   | PS2   | PS1   | PS0   |
| bit 7 |        |       |       |       |       |       | bit 0 |

### INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

|       |       |        |       |       |        |       |       |
|-------|-------|--------|-------|-------|--------|-------|-------|
| R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-x |
| GIE   | PEIE  | TMR0IE | INTE  | RBIE  | TMR0IF | INTF  | RBIF  |
| bit 7 |       |        |       |       |        |       | bit 0 |

## [2]Port-B Bits 4-7 Status Change Interrupt

- Port-B lines4-7 must be initialized for input
- The interrupt source must be set to take place either on the falling or the rising edge of the signal.[in **option REG**]
- pull-ups on port-B should be disabled in the OPTION register.
- (**RBIF** in the **INTCON Register**) must be initially cleared.
- **GIE** & **RBIE** bit in the **INTCON Register** must be enabled

### OPTION\_REG REGISTER (ADDRESS 81h, 181h)

|       |        |       |       |       |       |       |       |
|-------|--------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1  | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| RBPU  | INTEDG | T0CS  | T0SE  | PSA   | PS2   | PS1   | PS0   |
| bit 7 |        |       |       |       |       |       | bit 0 |

### INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

|       |       |        |       |       |        |       |       |
|-------|-------|--------|-------|-------|--------|-------|-------|
| R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-x |
| GIE   | PEIE  | TMR0IE | INTE  | RBIE  | TMR0IF | INTF  | RBIF  |
| bit 7 |       |        |       |       |        |       | bit 0 |

## TMR0 interrupt

- clrwdt
- GIE & T0IE must be enabled in **INTCON** reg
- Instruction cycle is selected using **T0CS** in **Option Reg**
- PSA is assigned to TMR0
- Initial value is put in TMR0 Reg
- Prescaler is chosed

### [3]ADC

ADRESH - Result High Register

ADRESL - Result Low Register

1. Configure the PIC I/O lines to be used in the conversion. All analog lines are initialized as **input** in the corresponding TRIS registers.
2. Select the ports to be used in the conversion by setting the PCFGx bits in the **ADCON1** register. Selects **right- or left-justification**.
3. Select the analog channels, select the A/D conversion clock, and enable the A/D module.
4. Wait the acquisition time.
5. Initiate the conversion by setting the GO/DONE bit in the ADCON0 register.
6. Wait for the conversion to complete.
7. Read and From the above two regs.

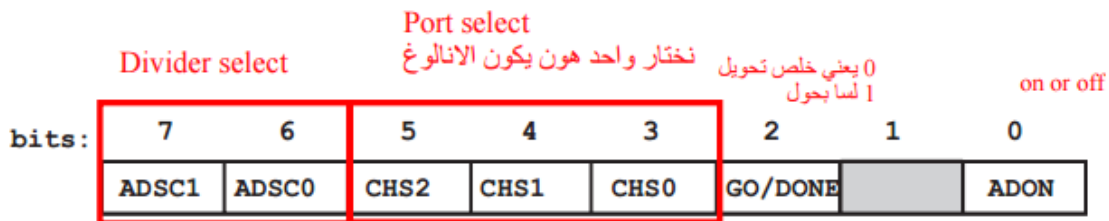


Figure 1: ADCON0

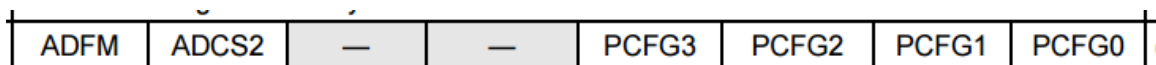


Figure 2: ADCON1

## [4] Capture Module

Note that it has **CCP1CON**(for configuration), (**CCPR1L** and **CCPR1H**)(for data)

To capture either two consecutive rising or falling edges and calculate their difference in order to measure the period.

1. CCP1(RC2) pin: Configured for input.
2. Timer1: 16-bit operation, use instruction cycle clock as clock source, prescaler set to 1
3. Capture on every rising edge. Write the value 0x05 into the CCP1CON,
4. Disable the CCP1 interrupt. Clear the CCP1iE bit of the PIE1 register.
5. Wait until **CCP1IF** is enabled inside **PR1** Register

### CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

|       |     |       |       |        |        |        |        |
|-------|-----|-------|-------|--------|--------|--------|--------|
| U-0   | U-0 | R/W-0 | R/W-0 | R/W-0  | R/W-0  | R/W-0  | R/W-0  |
| —     | —   | CCPxX | CCPxY | CCPxM3 | CCPxM2 | CCPxM1 | CCPxM0 |
| bit 7 |     |       |       |        |        |        | bit 0  |

Figure 3: Step 3

|       |     |         |         |         |        |        |        |
|-------|-----|---------|---------|---------|--------|--------|--------|
| U-0   | U-0 | R/W-0   | R/W-0   | R/W-0   | R/W-0  | R/W-0  | R/W-0  |
| —     | —   | T1CKPS1 | T1CKPS0 | T1OSCEN | T1SYNC | TMR1CS | TMR1ON |
| bit 7 |     |         |         |         |        |        | bit 0  |

0  
1  
on

Figure 4: Enable the timer1 (T1Con)

## [5] Compare Mode

The C language version of the program is as follows:

```
#include <p18F8720.h>
void main (void)
{
    TRISCbits.TRISC2 = 0;           /* configure CCP1 pin for output */
    T3CON = 0xC9;                  /* turn on TMR3 in 16-bit mode, prescaler is 1 */
    CCP1CON = 0x09;                /* configure CCP1 pin to set high initially but pull low on match*/
    CCPR1 = TMR3 + 0x0640;         /* start CCP1 compare with delay equals 1600*/
    PIR1bits.CCP1IF = 0;          /* clear CCP1IF flag */
    while (1) {
        while (!(PIR1bits.CCP1IF));
        PIR1bits.CCP1IF = 0;
        CCP1CON = 0x08;
        CCPR1 += 0x0960;           /* start CCP1 compare with delay equals 2400*/
        while (!(PIR1bits.CCP1IF));
        PIR1bits.CCP1IF = 0;
        CCP1CON = 0x09;
        CCPR1 += 0x0640;           /* start CCP1 compare with delay equals 1600*/
    }
}
```

## [6] The PWM period can be calculated using the following formula:

**PWM period = [(PR2) + 1] x 4 x T<sub>osc</sub> x (TMR<sub>y</sub> prescale factor)**

The PWM duty cycle can be calculated using the following formula:

**PWM duty cycle = (CCPR<sub>x</sub>L:CCP<sub>x</sub>CON<5:4>) x T<sub>osc</sub> x (TMR<sub>y</sub> prescale factor)**

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the **PR2**
2. Set the PWM duty cycle by writing to the **CCPR1L** register and **CCP1CON<5:4>** bits.
3. Make the CCP1(RC2) pin an output.
4. Set the **TMR2 prescaler** value and enable TMR2 by writing to **T2CON** register.
5. Configure the **CCP1CON** module for PWM operation.

