Example 1 (Polling – i.e. not using interrupt)

// Int_**INT0_a.c** Polling based program

```c
#include <xc.h>
#include "delays.h"

unsigned char j;
unsigned char press;

void LED_RD7_RD0(void)        // The f                                      ft to right
{
       j = 0x80;              // Initi
                              // ie th

       while(j!=0x01)         // Check
                              // to th
       {
         PORTD = j;           // Displ
         delay_ms(250);       // Calli
         j = j>>1;            // Makin                                      se
       }                      // opera

       PORTD = j;             // Displ
}                             // Stop
```

Pattern in PORTD (also in j):

1000 0000 = 0x80
0100 0000
0010 0000
0001 0000
0000 1000
0000 0100
0000 0010
0000 0001 = 0x01

```c
void main(void)               // Main Function
{
       ADCON1 = 0x0F;
       CMCON = 0x07;

       TRISBbits.TRISB0 = 1;       // RB0 is the push button switch for INT0
       TRISCbits.TRISC2 = 0;       // RC2 connects to a DC motor
       TRISD = 0x00;          // PortD connects to a bar LEDs

       PORTD = 0x00;          // LEDs all off
       press = 0;             // Not pressing yet

       while(1)               // Main Process
       {
         LED_RD7_RD0();       // Move Port D LEDs from bit7 to bit0

         // polling the switch at RB0
         if (PORTBbits.RB0 == 0)
         {
           press++;           // To track first or second time pressing RB0 switch

               if (press == 1)            // First press
               {
                   PORTCbits.RC2 = 1;     // Turn On Motor
               }
               else
               if (press == 2)            // Second press
               {
                   PORTCbits.RC2 = 0;     // Else turn Off Motor
                   press = 0;             // Reset the pressing counter
               }
           }
       }
}
```

Initialisation

Calls

*The program can only check whether the button is pressed after the "LED_RD7_RD0( )" has been completed – **slow response**!*

*Example 2 (Interrupt – using **INT0** interrupt)*

// Int_INT0_b.c

```c
// Int INT0 b.c Interrupt based program
// ISR activated by INT0 from an active low switch from RB0

#include <xc.h>
#include"delays.h"

unsigned char i, j;
unsigned char press, a, b;
```

Indicates to the compiler this is an ISR.

*This **interrupt service routine** (ISR) is called automatically whenever INT0 (RB0) is triggered.*

```c
void interrupt ISR_PortB0_low(void)      // Interrupt Service Routine for INT0
{
  if (INTCONbits.INT0IF)// External Interrupt Flag Bit = 1 when interrupt occurs
  {// Variable “press” is either 0 or 1 here.
      press++;                // To track first or second time pressing RB0 switch

      if (press == 1)                // First press
      {
        PORTCbits.RC2 = 1;           // Turn On Motor
      }
      else
      if (press == 2)                // Second press
      {
        PORTCbits.RC2 = 0;           // Else turn Off Motor
        press = 0;                   // Reset the pressing counter
      }

       INTCONbits.INT0IF = 0;   //Clearing the flag at the end of the ISR
  }
}
```

Otherwise, the ISR will be called again.

```c
void LED_RD7_RD0(void)// The function to shift a set-bit from the MSB to LSB
{                     (Same as in Example 1.)
      j = 0x80;           // Initialise j with B1000 0000
                          // ie the leftmost bit (or MSB)

      while(j!=0x01)      // Check that the bit has not been shifted
                          // to the right-most bit (LSB) ie B00000001
      {
        PORTD = j;        // Display j at PORTD
        delay_ms(250);    // Calling a delay function from delays.h
        j = j>>1;         // Making use of LOGICAL-RIGHT-SHIFT bit-wise
      }                   // operator to shift data to the right

      PORTD = j;          // Display j at PORTD
}                         // Stop at B00000001
```

```c
void main(void)                // Main Function
{
```

Initialisation

```c
    ADCON1 = 0x0F; // No analog inputs. (All digital.)
    CMCON = 0x07;  // Set Comparator Mode to Off – not really necessary.

    TRISBbits.TRISB0 = 1; // RB0 is the push button switch for INT0
    TRISCbits.TRISC2 = 0; // RC2 connects to a DC motor
    TRISD = 0x00;         // PortD connects to a bar LEDs

    PORTD = 0x00;         // LEDs all off
    press = 0;            // Not pressing yet
    j = 0;

    RCONbits.IPEN =1;     // Bit7 Interrupt Priority Enable Bit
                          // 1 Enable priority levels on interrupts
                          // 0 Disable priority levels on interrupts

    INTCONbits.GIEH =1;   // Bit7 Global Interrupt Enable bit
                          // When IPEN = 1                 (Default value)
                          // 1 Enable all high priority interrupts
                          // 0 Disable all high priority interrupts

    INTCON2bits.INTEDG0 = 0;// Bit4 External Interrupt2 Edge Select Bit
                          // 1 Interrupt on rising edge
                          // 0 Interrupt on falling edge

    INTCONbits.INT0IE = 1;  // Bit4 INT0 External Interrupt Enable bit
                          // 1 Enable the INT0 external interrupt
                          // 0 Disable the INT0 external interrupt

    INTCONbits.INT0IF = 0;  // Clearing the flag

    while(1)              // Main Process
    {
        LED_RD7_RD0();    // Move Port D LEDs from bit7 to bit0
    }
}
```

Normal tasks in the while (1) loop.

// Int_TMR0.c

```c
#include <xc.h>

unsigned char j;

void interrupt ISR_Timer0_Int()  // Timer0 Interrupt Service Routine (ISR)
{
 if (INTCONbits.TMR0IF)          // TMR0IF:- Timer0 Overflow Interrupt Flag Bit
                                 // 1 = TMR0 reg has overflowed
                                 // 0 = TMR0 reg has not overflowed
    {
        TMR0H = 0x48;  // Timer0 start value = 0x48E5 for 1 second
        TMR0L = 0xE5;

        j++; // Increase count by 1
        PORTD = j; // Show count value at Port D Leds

        INTCONbits.TMR0IF = 0;  // Reset TMR0IF to "0" since the end of
                                // the interrupt function has been reached
    }
}
```

*This **in**terrupt service routine (ISR) is called automatically whenever **TMR0IF =1** (i.e. TMR0 rolls over).*

==1

Do the following only if TMR0 is interrupting.

***TMR0IF** will become 1 when roll-over occurs (i.e. **TMR0** = 0x0000.)*
At which time, the above ISR will be called automatically.

```
void main(void)                         // Main Function
{
                     Initialisation
      ADCON1 = 0x0F;
      CMCON = 0x07;

      TRISAbits.TRISA3 = 1;             // RA3 is the On/Off switch
      TRISCbits.TRISC2 = 0;             // RC2 connects to a DC motor
      TRISD = 0x00;                     // PortD connects to a bar LEDs

      PORTD = 0x00;                     // LEDs all off
      j = 0;                            // Start count from 0

      RCONbits.IPEN =1;                 // Bit7 Interrupt Priority Enable Bit
                                        // 1 Enable priority levels on interrupts
                                        // 0 Disable priority levels on interrupts


      INTCONbits.GIEH =1;               // Bit7 Global Interrupt Enable bit
                                        // When IPEN = 1
                                        // 1 Enable all high priority interrupts
                                        // 0 Disable all high priority interrupts

      T0CON = 0b00000111;               // bit7:0 Stop Timer0
                                        // bit6:0 Timer0 as 16 bit timer
                                        // bit5:0 Clock source is internal
                                    // bit4:0 Increment on lo to hi transition on TOCKI
                                        // bit3:0 Prescaler output of Timer0
                                        // bit2-bit0:111 1:256 prescaler


      INTCON2 = 0b10000100;             // bit7 :PORTB Pull-Up Enable bit
                                        //      1 All PORTB pull-ups are disabled
                                        // bit2 :TMR0 Overflow Int Priority Bit
                                        //    1 High Priority

      TMR0H = 0x48;                     // Initialising TMR0H
      TMR0L = 0xE5;                  // Initialising TMR0L for 1 second interrupt

      T0CONbits.TMR0ON = 1;             // Turn on timer
      INTCONbits.TMR0IE = 1;            // bit5 TMR0 Overflow Int Enable bit
                                        // 0 Disable the TMR0 overflow int

      INTCONbits.TMR0IF = 0;         // bit2 TMR0 Overflow Int Flag bit
                                        // 0 TMR0 register did not overflow

    while (1) // Main Process
    {
       if (PORTAbits.RA3 == 0) // If RA3 switch is ON

            PORTCbits.RC2 = 1; // Turn On Motor

       else

            PORTCbits.RC2 = 0; // Else turn Off Motor

    }
       Normal tasks in the while (1) loop.
}
```