AN566

Using the PORTB Interrupt on Change as an External Interrupt

Author: *Mark Palmer*
Microchip Technology Inc.

INTRODUCTION

The PICmicro™ families of RISC microcontrollers are designed to provide advanced performance and a cost-effective solution for a variety of applications. To address these applications, there is the PIC16CXXX microcontroller family of products. This family has numerous peripheral and special features to better address user applications.

The feature this application note will focus on is the Interrupt on Change of the PORTB pins. This “interrupt on change” is triggered when any of the RB7:RB4 pins, configured as an input, changes level. When this interrupt is used in conjunction with the software programmable weak internal pull-ups, a direct interface to a keypad is possible. This is shown in application note AN552, Implementing Wake-up on Key Stroke. Another way to use the “interrupt on change” feature would be as additional external interrupt sources. This allows PIC16CXXX devices to support multiple external interrupts, in addition to the built-in external interrupt on the INT pin.

This application note will discuss some of the issues in using PORTB as additional external interrupt pins, and will show some examples. These examples can be easily modified to suit your particular needs.

USING A PORTB INPUT FOR AN EXTERNAL INTERRUPT

The interrupt source(s) cannot simply be directly connected to the PORTB pins, and expect an interrupt to occur the same as on the interrupt (INT) pin. To develop the microcontrollers hardware/software to act as an interrupt by an external signal, we must know the characteristics of the external signal. After we know this, we can determine the best way to structure the program to handle this signal. The characteristics that we need to consider when developing the interrupt include:

1. The rising edge and falling edges.
2. The pulse width of the interrupt trigger (high time / low time).

It is easy to understand the need of knowing about which edge triggers the interrupt service routine for the external interrupt. This allows one to ensure that the interrupt service routine is only entered for the desired edge, with all other edges ignored. Not so clear is the pulse width of the interrupt’s trigger. This characteristic helps determine the amount of additional overhead that the software routine may need.

© 1997 Microchip Technology Inc.

Figure 1 shows the two cases for the interrupt signal verses the time to complete the interrupt service routine. The first waveform is when the signal makes the low-to-high-to-low transitions before the interrupt service routine has completed (interrupt flag cleared). When the interrupt flag has been cleared, the interrupt signal has already returned to the inactive level. The next transition of the signal is due to another interrupt request. An interrupt signal with this characteristic will be called a small pulse width signal.

The second waveform is when the signal only makes the low-to-high transitions before the interrupt service routine has completed (interrupt flag cleared). The next transition (high-to-low) will return the interrupt signal to the inactive level. This will generate a “false” interrupt, that will need to be cleared. Then the following transition (low-to-high) will be a “true” interrupt. An interrupt signal with this characteristic will be called a wide pulse width signal.

An interrupt pulse with a small pulse width requires less overhead than a wide pulse width. A small pulse width signal must be less than the minimum execution time of the interrupt service routine, while a wide pulse width must be greater then the maximum time through the interrupt service routine.

Example 1 shows a single interrupt source on PORTB (RB7), which executes the interrupt service routine on a rising edge. The interrupt source has a small pulse width. In this case, since the interrupt pulse width is small, the pulse has gone high and then low again before PORTB is read to end the mismatch condition. So when PORTB is read it will read a low signal and will again be waiting for the rising edge transition.

FIGURE 1: INTERRUPT STEPS FOR SMALL AND WIDE PULSE WIDTHS

EXAMPLE 1: SINGLE INTERRUPT WITH A SMALL PULSE WIDTH

```
PER_INT       BTFSS   INTCON, RBIF          ; PortB interrupt?
              GOTO    OTHER_INT             ; Other interrupt
                      :                     ; Do task for INT on RB7
                      :                     ;
CLR_RBINTF    MOVF    PORTB, 1              ; Read PortB (to itself) to end
                                            ; mismatch condition
              BCF     INTCON, RBIF          ; Clear the RB interrupt flag.
              RETFIE                        ; Return from interrupt
OTHER_INT             :                     ; Do what you need to here
                      :
              RETFIE                        ; Return from interrupt
```


© 1997 Microchip Technology Inc.

Example 2 shows a single interrupt source on PORTB (RB7), which executes the interrupt service routine on a rising edge. The interrupt source has a wide pulse width. In this case since the interrupt pulse width is large, the pulse is still high before PORTB is read to end the mismatch condition. So when PORTB is read it will read a high signal and will generate an interrupt on the next falling edge transition (which should be ignored).

EXAMPLE 2: SINGLE INTERRUPT WITH A WIDE PULSE WIDTH

```
PER_INT       BTFSS  INTCON, RBIF         ; PortB interrupt?
              GOTO   OTHER_INT            ; Other interrupt
              BTFSC  PORTB, RB7           ; Check for rising edge
              GOTO   CLR_RBINTF           ; Falling edge, clear PortB int
              :                           ;   flag
                     :                    ; Do task for INT on RB7
                     :
CLR_RBINTF    MOVF   PORTB, 1             ; Read PortB (to itself) to end
                                          ;    mismatch condition
              BCF    INTCON, RBIF         ; Clear the RB interrupt flag.
              RETFIE                      ; Return from interrupt
OTHER_INT            :                    ; Do what you need to here
                     :
              RETFIE                      ; Return from interrupt
```

Example 3 shows an interrupt on change with the interrupt source on PORTB (RB7). This executes the interrupt service routine on a both edges. The interrupt source must have a minimum pulse width to ensure that both edges can be “seen”. The minimum pulse width is the maximum time from the interrupt edge to the reading of PORTB and clearing the interrupt flag.

EXAMPLE 3: INTERRUPT ON CHANGE

```
PER_INT       BTFSS  INTCON, RBIF         ; PortB interrupt?
              GOTO   OTHER_INT            ; Other interrupt
CLR_RBINTF    MOVF   PORTB, 1             ; Read PortB (to itself) to end
                                          ;    mismatch condition
              BCF    INTCON, RBIF         ; Clear the RB interrupt flag.
                     :                    ; Do task for INT on RB7
                     :                    ;
              RETFIE                      ; Return from interrupt
OTHER_INT                                 ; Do what you need to here
                     :
              RETFIE                      ; Return from interrupt
```

USING PORTB INPUTS FOR MULTIPLE INTERRUPTS

The previous examples have been for a single external interrupt on PORTB. This can be extended to support up to four external interrupts. To do this requires additional software overhead, to determine which of the PORTB pins (RB7:RB4) caused the interrupt. Care should be taken in the software to ensure that no interrupts are lost.

In this example, the interrupt sources on RB7, RB5, and RB4 have a small pulse width, while the interrupt source on pin RB6 is wide and should cause a trigger on the rising edge.

SUMMARY

The PORTB interrupt on change feature is both a very convenient method for direct interfacing to an external keypad, with no additional components, but is also versatile in its uses the ability to add up to four additional external interrupts. Of course hybrid solutions are also possible. That is, for example, using PORTB<6:1> as a 3x3 keypad, with PORTB<7> as an external interrupt and PORTB<0> as a general purpose I/O. The flexibility of this feature allows the user to implement a best fit design for the application.

EXAMPLE 4: MULTIPLE INTERRUPTS WITH DIFFERENT PULSE WIDTHS

```
PER_INT        BTFSS   INTCON, RBIF          ; PortB interrupt?
               GOTO    OTHER_INT             ; Other interrupt
;
; PortB change interrupt has occurred. Must determine which pin caused
; interrupt and do appropriate action. That is service the interrupt,
; or clear flags due to other edge.
;
               MOVF    PORTB, 0              ; Move PortB value to the W register
                                             ;   This ends mismatch conditions
               MOVWF   TEMP                  ; Need to save the PortB reading.
               XORWF   LASTPB, 1             ; XOR last PortB value with the new
                                             ;   PortB value.
CK_RB7         BTFSC   LASTPB, RB7           ; Did pin RB7 change
               CALL    RB7_CHG               ; RB7 changed and caused the interrupt
CK_RB6         BTFSC   LASTPB, RB6           ; Did pin RB6 change
               CALL    RB6_CHG               ; RB6 changed and caused the interrupt
CK_RB5         BTFSC   LASTPB, RB5           ; Did pin RB5 change
               CALL    RB5_CHG               ; RB5 changed and caused the interrupt
CK_RB4         BTFSC   LASTPB, RB4           ; Did pin RB4 change
               GOTO    RB4_CHG               ; RB4 changed and caused the interrupt
;
RB7_CHG             :                        ; Do task for INT on RB7
                    :                        ;
               RETURN
RB6_CHG        BTFSC   PORTB, RB6            ; Check for rising edge
               RETURN                        ; Falling edge, Ignore
                    :                        ; Do task for INT on RB6
                    :
               RETURN
RB5_CHG             :                        ; Do task for INT on RB5
                    :                        ;
               RETURN
RB4_CHG             :                        ; Do task for INT on RB4
                    :                        ;
CLR_RBINTF     MOVF    TEMP, 0               ; Move the PortB read value to the
               MOVWF   LASTPB                ;   register LASTPB
               BCF     INTCON, RBIF          ; Clear the RB interrupt flag.
               RETFIE                        ; Return from interrupt
;
OTHER_INT           :                        ; Do what you need to here
                    :
               RETFIE                        ; Return from interrupt
```


© 1997 Microchip Technology Inc.