Using the PortB Interrupt on Change as an External Interrupt

INTRODUCTION

The PIC16/17 family of RISC-like microcontrollers has been designed to provide advanced performance and a cost-effective solution for a variety of applications. To address these applications, there is the PIC16CXX microcontroller family of products. This family has numerous peripheral and special features to better address user applications.

One feature is the interrupt on change of the PORTB pins. This “interrupt on change” is caused when any of the RB<7:4> pin, configured as input, changes levels. When 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 the PIC16CXX devices to support multiple external interrupts, in addition to 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 the interrupt to function the same as the interrupt (INT) pin. The characteristic of the interrupt signal must also be known to develop the microcontrollers hardware/software. After we know this, we can determine the best way to structure the program to handle this signal. These characteristics include:

1. Trigger interrupt on rising, falling, or both edges
2. What is the pulse width of the interrupt (high time / low time)

It is easy to understand the need of knowing which edge triggers the external interrupt service routine. 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 pulse width of the interrupt. This determines the amount of additional overhead that the software routine may need.

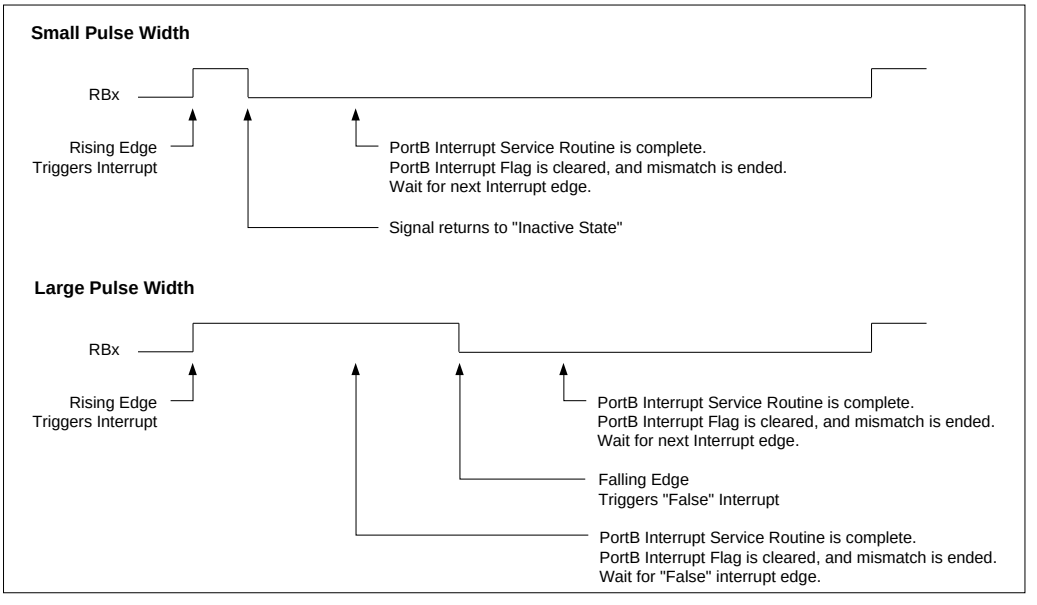
Using the PortB Interrupt on Change as an External Interrupt

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 then 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
```

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 3 shows a 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 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: 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 4 external interrupts. To do this requires additional software overhead, to determine which of the PortB pins (RB<7:4>) 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 interrupt. Of course hybrid solutions are also possible. That is, for example, using PORTB<6:1> as a 3x3 keypad, with PORTB7 as an external interrupt and PORTB0 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
```

Author: Mark Palmer, Logic Products Division

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770 640-0034 Fax: 770 640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408 436-7950 Fax: 408 436-7955

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

Korea
Microchip Technology
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku,
Seoul, Korea
Tel: 82 2 554 7200 Fax: 82 2 558 5934

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 Muenchen, Germany
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95