University of San Carlos | Department of
COMPUTER ENGINEERING

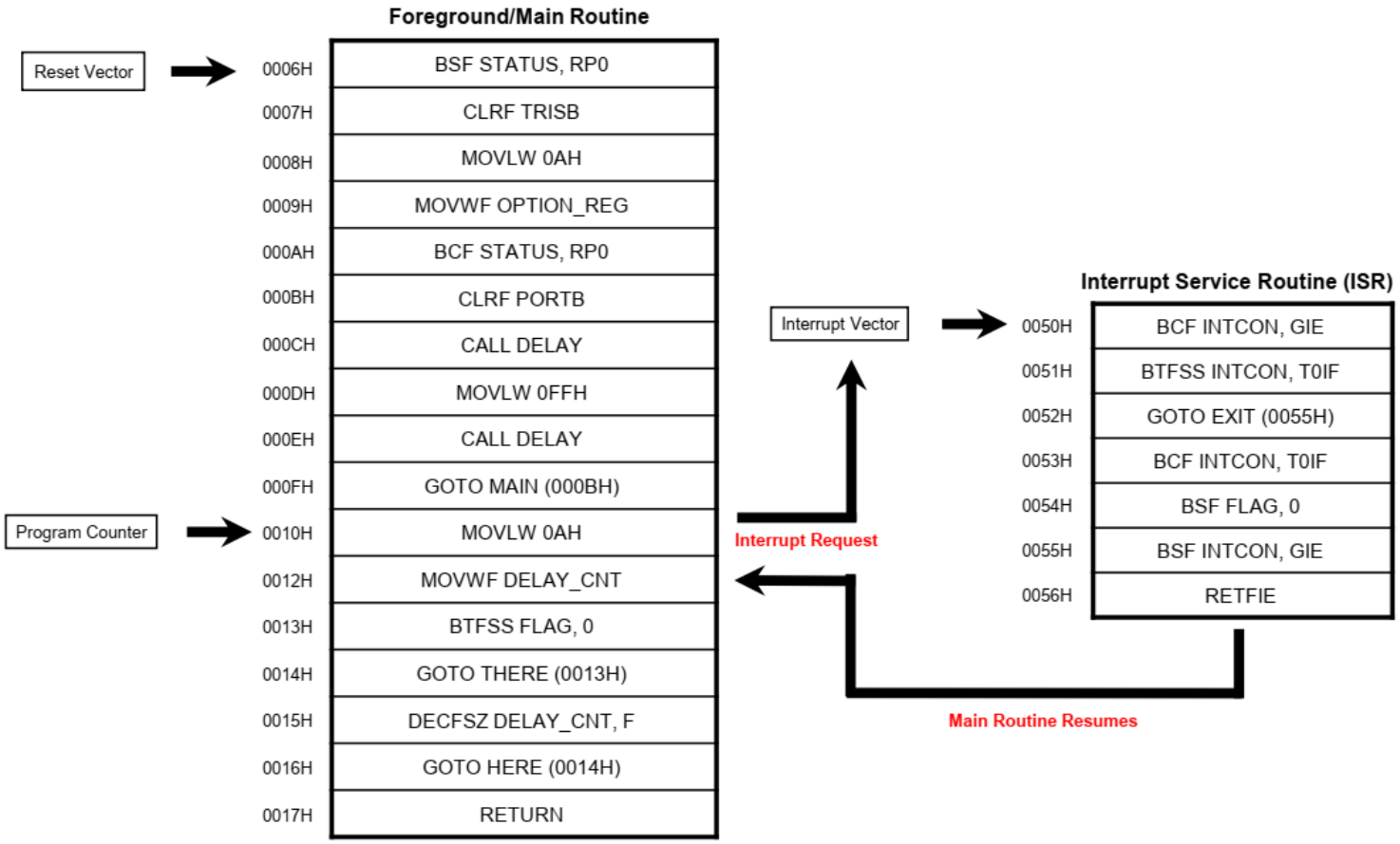
CpE 3201 (Embedded Systems)

Interrupts

What is an Interrupt?

- signal generated by some event external to the CPU (Interrupt Request)
- causes the CPU to stop what it is currently doing (stop executing the code it is currently running) and jump to a separate piece of code (designed to deal with the event which generated the interrupt request)

What is an Interrupt?



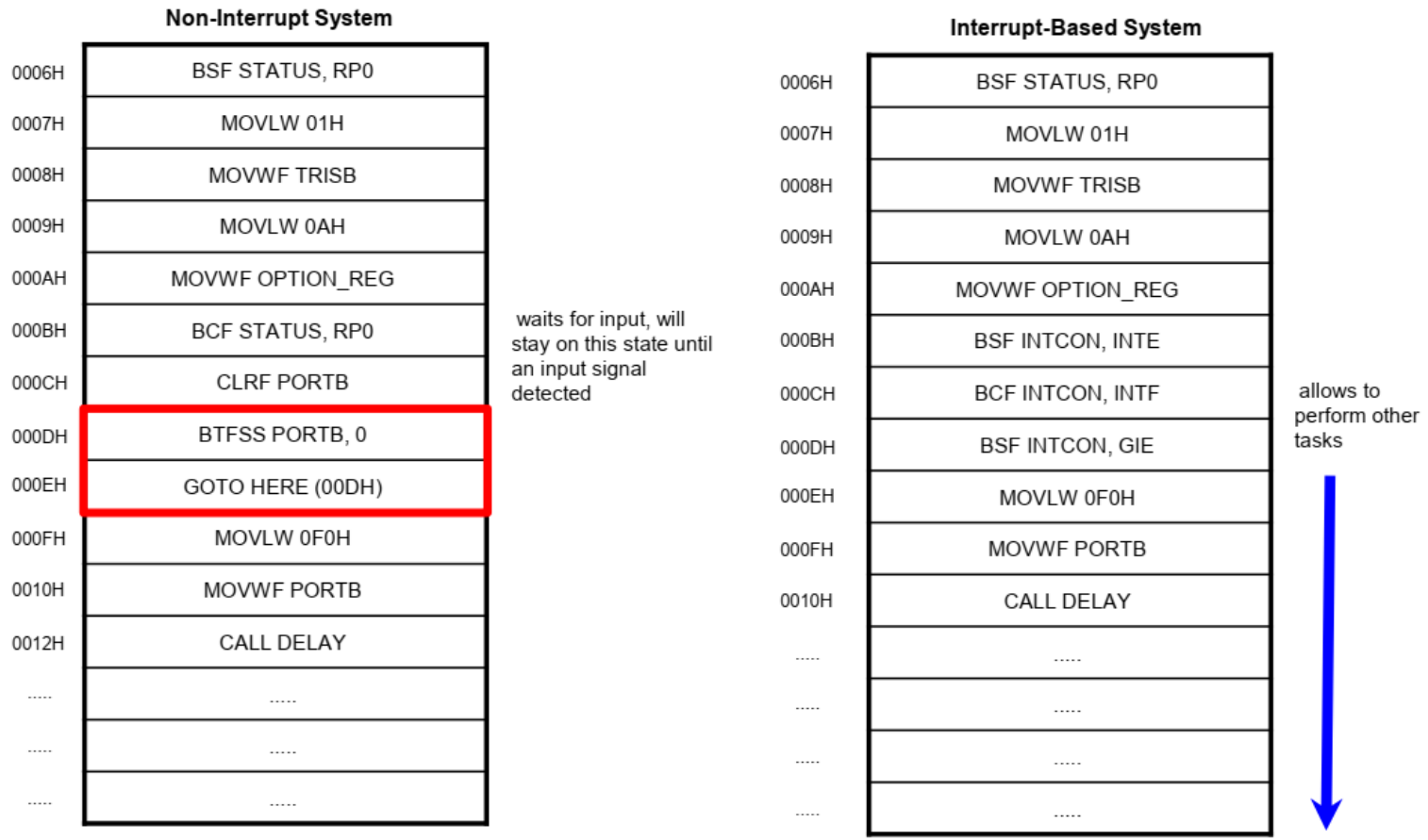
Interrupt Service Routine

- the code that handles the interrupt request and is executed
- when finished, returns to the instruction that was running prior to the interrupt, which then resumes running with no awareness that it has been pre-empted by the interrupt code
- must be as short as possible

Why use Interrupts?

- one useful feature in embedded systems; makes systems efficient and responsive to critical events
- makes code easier to manage

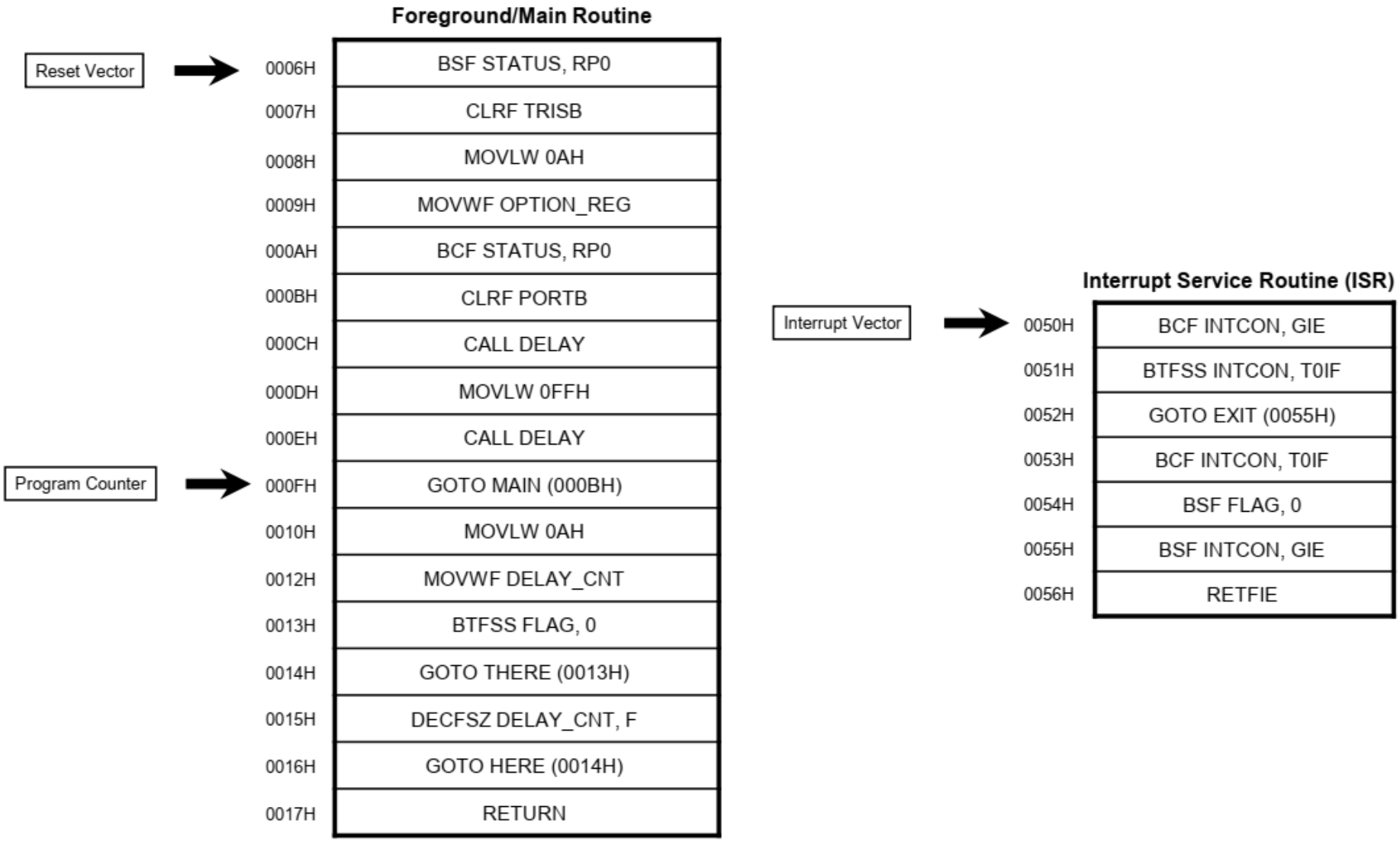
Why use Interrupts?

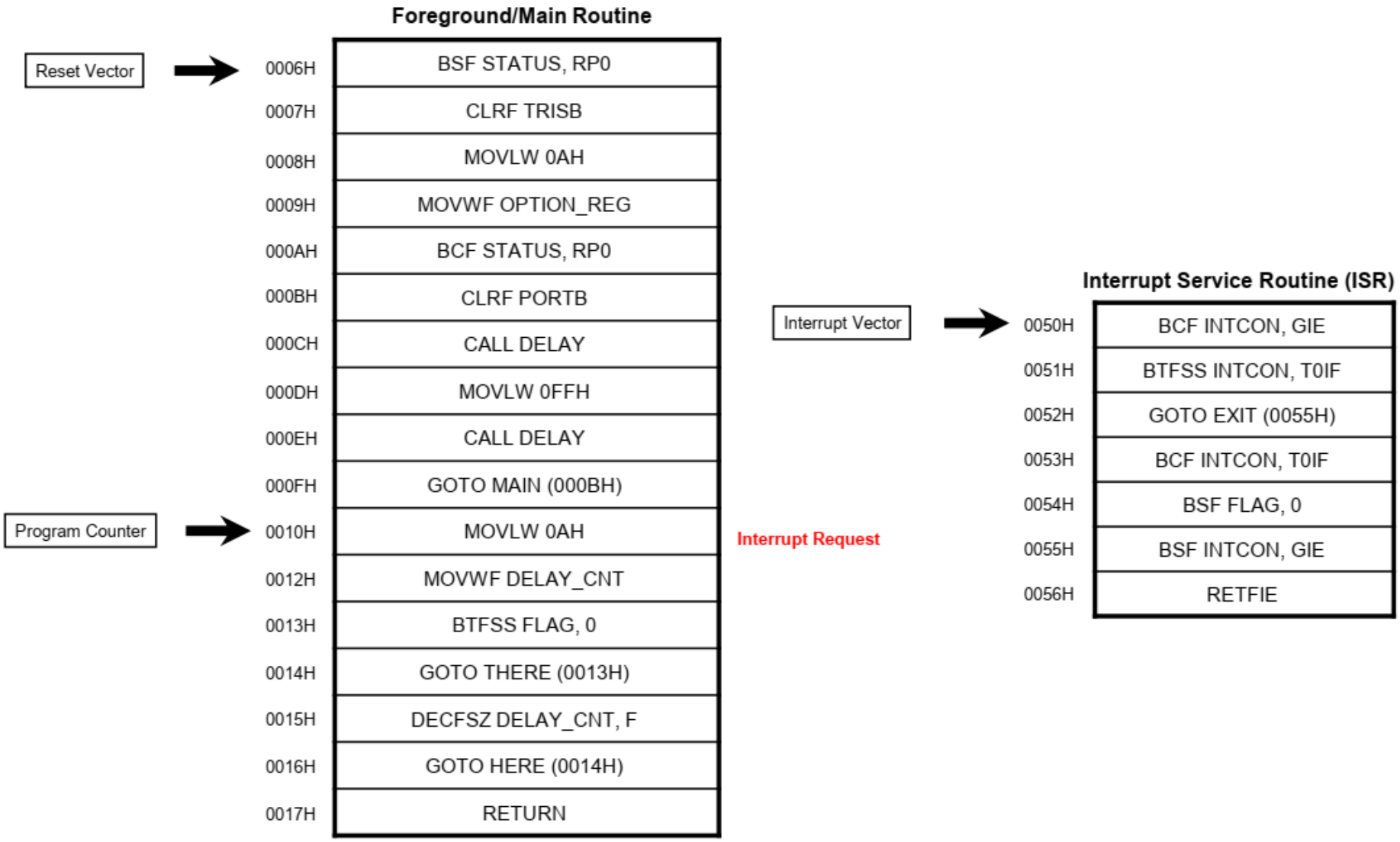


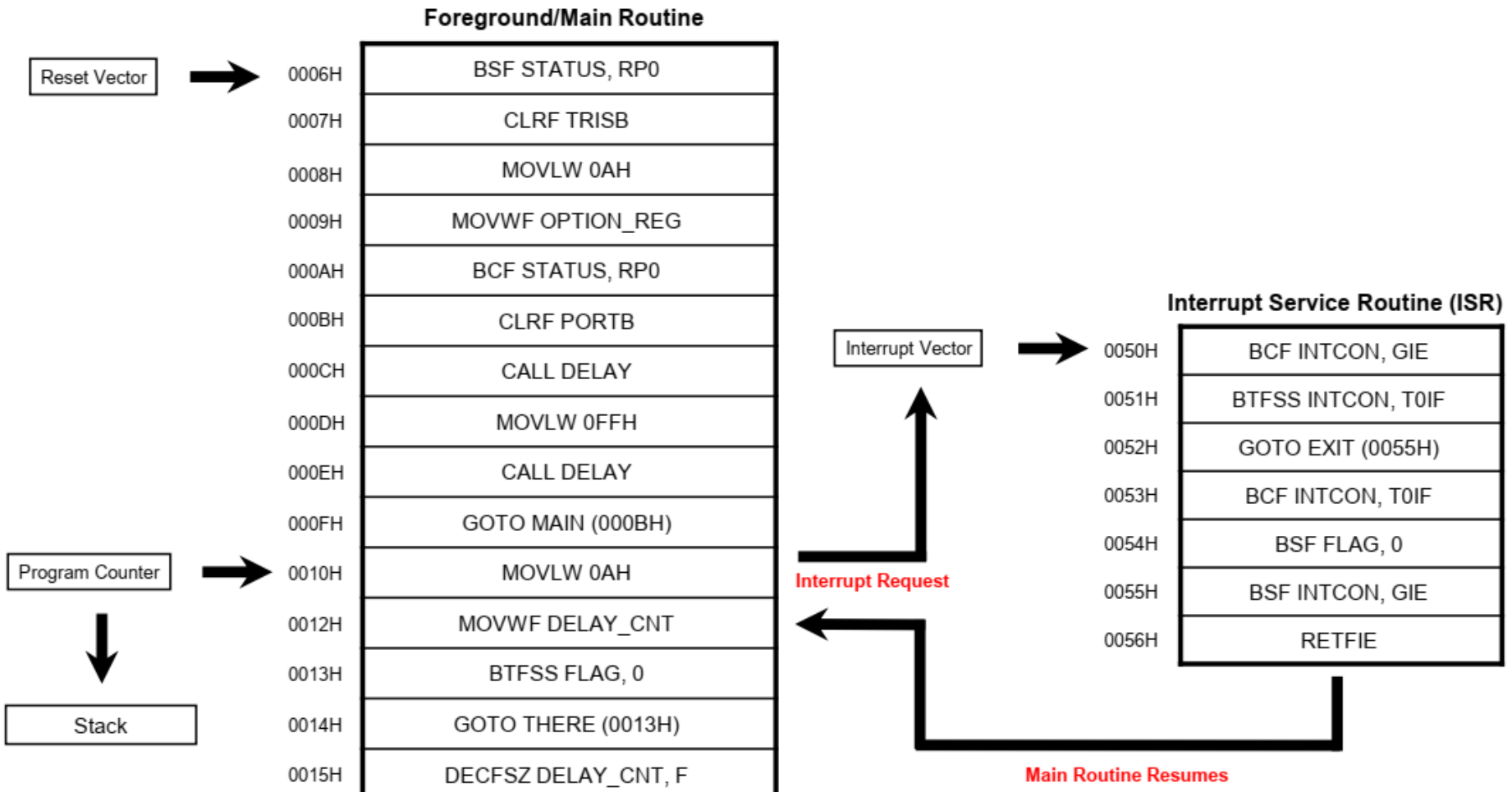
Disadvantages of Interrupts

- Interrupt can be confusing and error-prone
- Overlapping of operations may happen together with the main routine
- Should be tightly regulated and must consider the length of the ISR as to not to further delay the foreground routine

How an Interrupt works?


Foreground/Main Routine
Reset Vector
0006H
BSF STATUS, RP0
0007H
CLRF TRISB
0008H
MOVLW 0AH
0009H
MOVWF OPTION_REG
000AH
BCF STATUS, RP0
000BH
CLRF PORTB
000CH
CALL DELAY
000DH
MOVLW 0FFH
000EH
CALL DELAY
Program Counter
000FH
GOTO MAIN (000BH)
0010H
MOVLW 0AH
0012H
MOVWF DELAY_CNT
0013H
BTFSS FLAG, 0
0014H
GOTO THERE (0013H)
0015H
DECFSZ DELAY_CNT, F
0016H
GOTO HERE (0014H)
0017H
RETURN
Interrupt Service Routine (ISR)
Interrupt Vector
0050H
BCF INTCON, GIE
0051H
BTFSS INTCON, T0IF
0052H
GOTO EXIT (0055H)
0053H
BCF INTCON, T0IF
0054H
BSF FLAG, 0
0055H
BSF INTCON, GIE
0056H
RETFIE


Foreground/Main Routine
Reset Vector
0006H
BSF STATUS, RP0
0007H
CLRF TRISB
0008H
MOVLW 0AH
0009H
MOVWF OPTION_REG
000AH
BCF STATUS, RP0
000BH
CLRF PORTB
000CH
CALL DELAY
000DH
MOVLW 0FFH
000EH
CALL DELAY
000FH
GOTO MAIN (000BH)
Program Counter
0010H
MOVLW 0AH
0012H
MOVWF DELAY_CNT
0013H
BTFSS FLAG, 0
0014H
GOTO THERE (0013H)
0015H
DECFSZ DELAY_CNT, F
0016H
GOTO HERE (0014H)
0017H
RETURN
Interrupt Request
Interrupt Vector
Interrupt Service Routine (ISR)
0050H
BCF INTCON, GIE
0051H
BTFSS INTCON, T0IF
0052H
GOTO EXIT (0055H)
0053H
BCF INTCON, T0IF
0054H
BSF FLAG, 0
0055H
BSF INTCON, GIE
0056H
RETFIE


Foreground/Main Routine
Reset Vector
0006H
BSF STATUS, RP0
0007H
CLRF TRISB
0008H
MOVLW 0AH
0009H
MOVWF OPTION_REG
000AH
BCF STATUS, RP0
000BH
CLRF PORTB
000CH
CALL DELAY
000DH
MOVLW 0FFH
000EH
CALL DELAY
000FH
GOTO MAIN (000BH)
Program Counter
0010H
MOVLW 0AH
0012H
MOVWF DELAY_CNT
0013H
BTFSS FLAG, 0
Stack
0014H
GOTO THERE (0013H)
0015H
DECFSZ DELAY_CNT, F
0016H
GOTO HERE (0014H)
0017H
RETURN
Interrupt Vector
Interrupt Request
Interrupt Service Routine (ISR)
0050H
BCF INTCON, GIE
0051H
BTFSS INTCON, T0IF
0052H
GOTO EXIT (0055H)
0053H
BCF INTCON, T0IF
0054H
BSF FLAG, 0
0055H
BSF INTCON, GIE
0056H
RETFIE
Main Routine Resumes

Enabling/Disabling Interrupts

- Every interrupt source will have some way to enable or disable it - usually an enable bit in a configuration register
- Interrupt disabling is also called ***interrupt masking***, but some interrupts are non-maskable meaning the source cannot be disabled and is persistent
- At the end of the interrupt configuration sequence for that peripheral, the interrupt will be enabled but this will not cause interrupts to begin to be serviced

Enabling/Disabling Interrupts

- A **Global Interrupt Enable (GIE)** mechanism of the CPU will enable all unmasked interrupts; this is very useful if you want to disable the all unmasked interrupts temporarily
- GIE is used to disable all unmasked interrupts during ISR to prevent ***interrupt overlapping***

Common Interrupt Sources

- Input pin state change
- Timer overflow
- Timer capture
- Timer compare match
- UART
- SPI
- I2C
- Analog to Digital Conversion
- Watchdog

Things to consider when using Interrupts

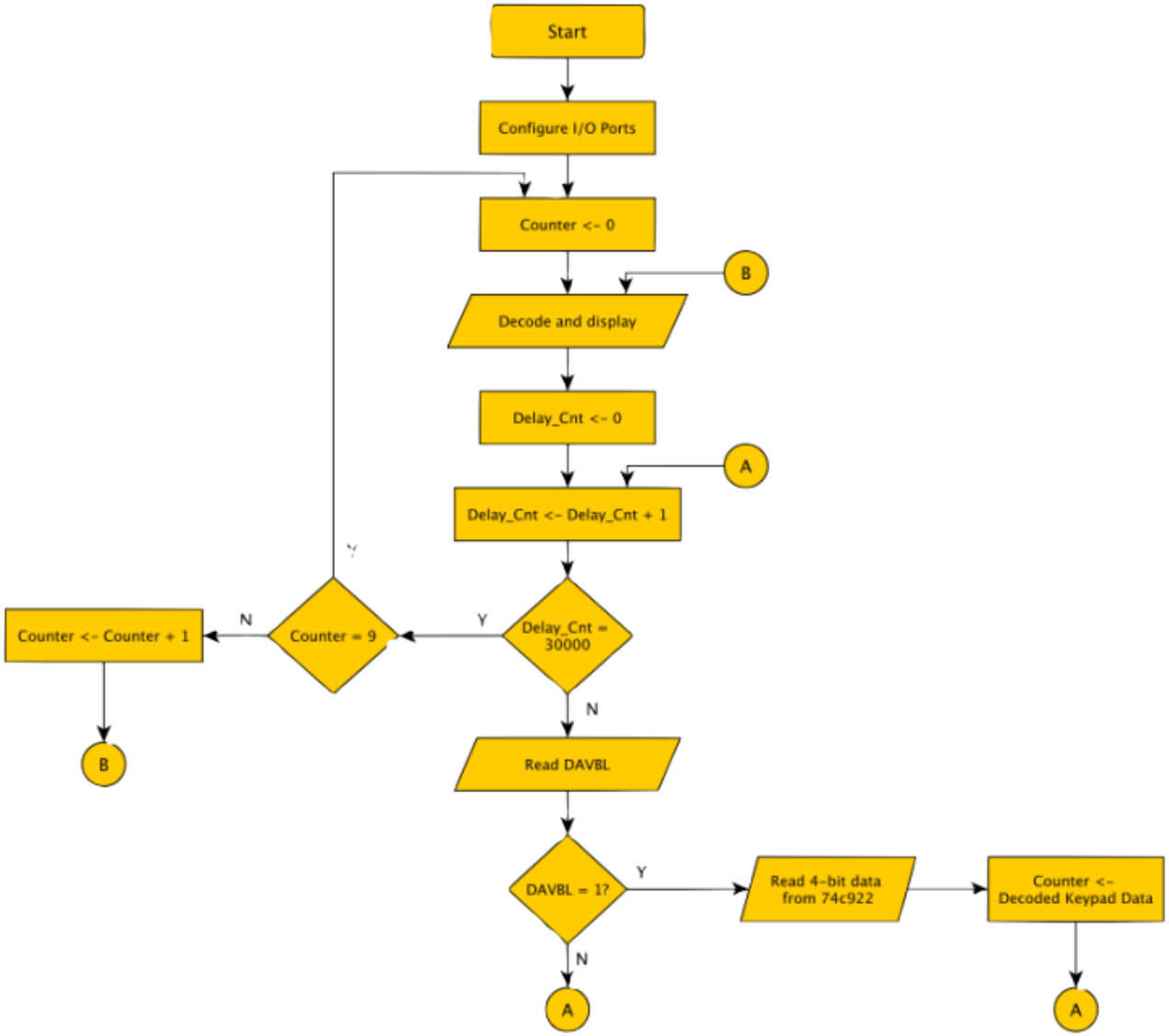
- ISR should be a short as possible
- Data corruption
- Interrupt Priorities

Example (External Interrupt Source)

- A numeric keypad (through 74C922) and a 7-segment display is connected to the PIC16F877A MCU.
- Upon start-up, the display would count-up from 0-9 then recycles at an interval of 1 second. When keys 1-9 in the keypad are pressed, the key number will be displayed and continues to count from that number.

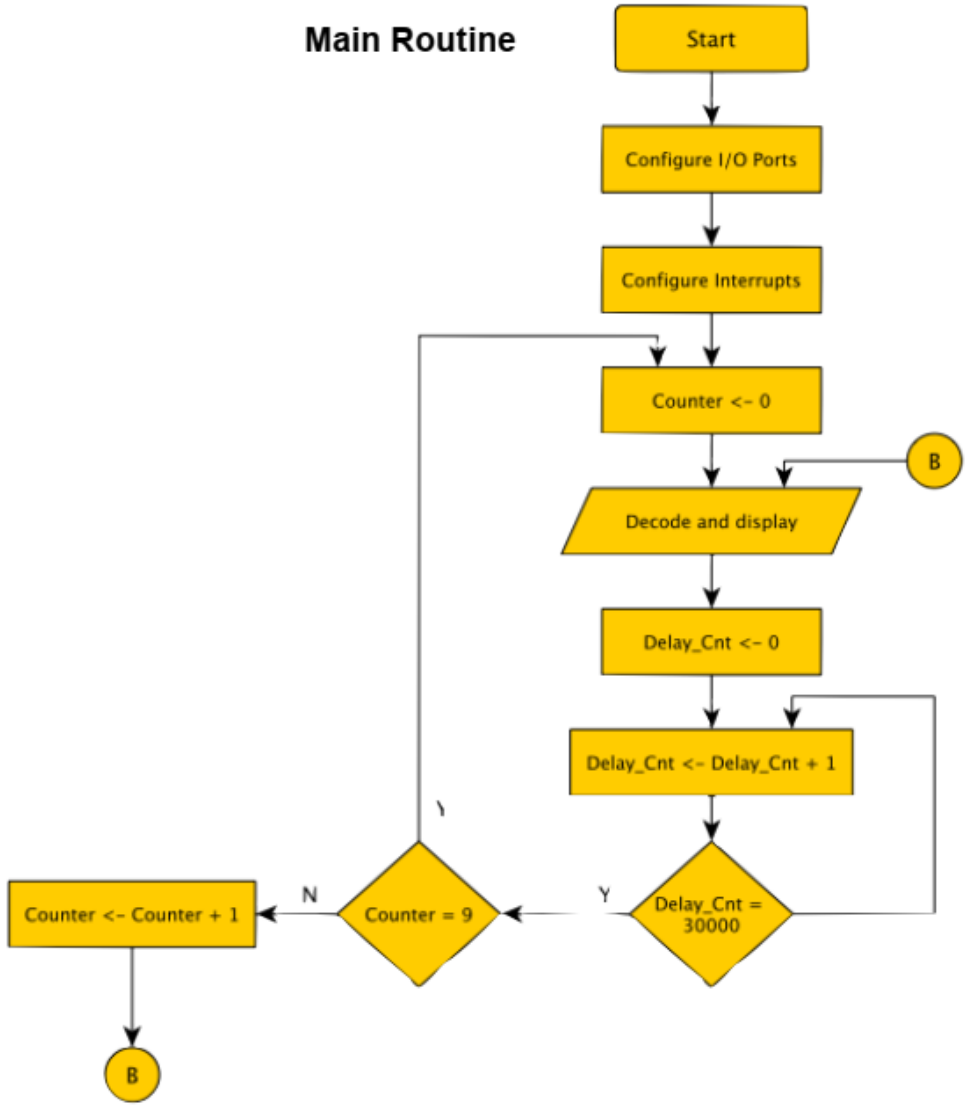
Conventional Approach

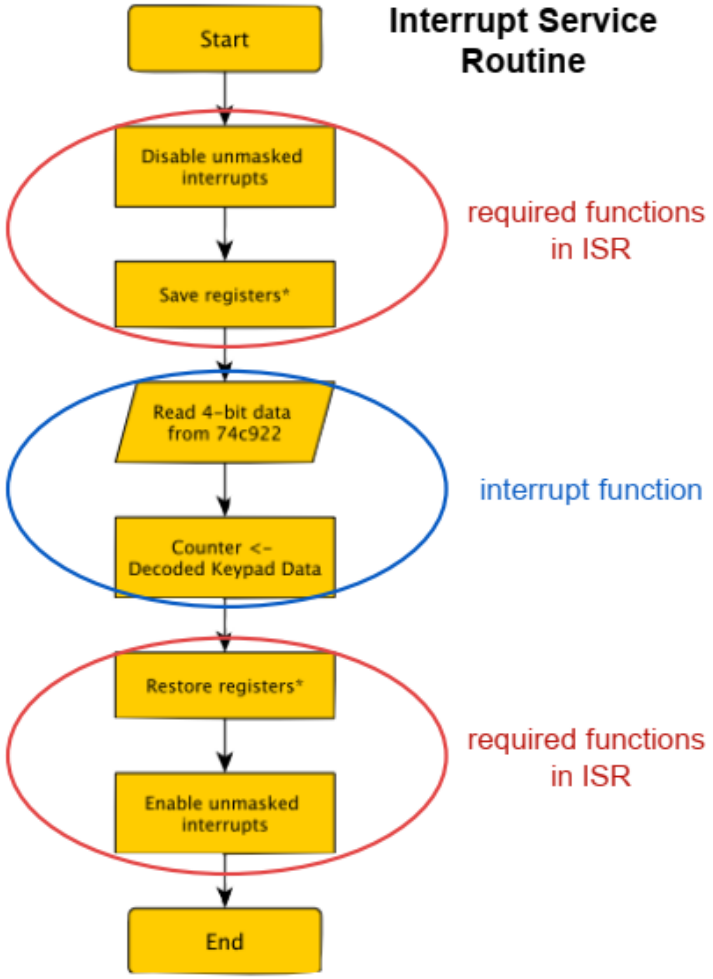
- The conventional approach is to wait for the keypad input. In this case, waiting for key press is not possible since the counter would be updated every second.
- The solution is to read the keypad data in between delay.


Start
Configure I/O Ports
Counter <- 0
B
Decode and display
Delay_Cnt <- 0
A
Delay_Cnt <- Delay_Cnt + 1
Delay_Cnt = 30000
Y
Counter = 9
N
Counter <- Counter + 1
B
N
Read DAVBL
DAVBL = 1?
Y
Read 4-bit data from 74c922
Counter <- Decoded Keypad Data
A
N
A

Interrupt Approach

- Using interrupts, the CPU is tasked only of updating the Counter value as well as decoding and displaying.
- When the keypad is pressed, it generates an interrupt request and executes the ISR according to the interrupt vector address, in this case reading the keypad data and updating the Counter value.





*DAVBL is the interrupt source (external interrupt)

Interrupt Sources of PIC16F877A

- External interrupts
- Overflow/underflow interrupts
- A/D conversion interrupts
- Communication interrupts

RB0/INT External Interrupt

- An external interrupt source is a signal (state change for the interrupt to be detected) sent to MCU from an I/O device or other MCU.
- State changes – rising edge, falling edge, or both
- Interrupt will be generated if enabled/unmasked together with GIE; ISR will be executed.

Interrupt Configuration (OPTION_REG)

REGISTER 2-2: 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
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Interrupt Configuration (INTCON)

REGISTER 2-3: 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

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

RB0/INT External Interrupt

- To enable/unmask the interrupt, **INTE** must be set to '1'. However, **GIE** will enable or disable all unmasked interrupt.
- Therefore to enable **RB0/INT** external interrupt, both **INTE** and **GIE** must be set to '1'.
- When an interrupt is generated, **INTF** will be set to '1' by the MCU and has to be *cleared manually* via software.

Timer0 Module

- The Timer0 module is a readable and writable 8-bit timer/counter.
- It has a programmable prescaler* and can be powered using the internal clock cycle and external clock.
- If enabled, it can generate an interrupt upon overflow.

* a prescaler is basically a clock divider in 2n

Interrupt Configuration (OPTION_REG)

REGISTER 2-2: 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
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Interrupt Configuration (INTCON)

REGISTER 2-3: 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

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Timer0 Overflow Interrupt

- A timer overflow occurs after a counter reach its terminal count. Incrementing a counter on its terminal count will make the value overflow and it automatically resets the counter.
- The PIC16F877A MCU has several counters/timers that can be configured to generate an overflow interrupt, one of which is the **Timer0**.

Timer0 Overflow Interrupt

- To enable/unmask the interrupt, **TMR0IE** must be set to '**1**'. However, **GIE** will enable or disable all unmasked interrupt.
- Therefore to enable **Timer0** overflow interrupt, both **TMR0IE** and **GIE** must be set to '**1**'.
- When an interrupt is generated, **TMR0IF** will be set to '**1**' by the MCU and has to be *cleared manually* via software.

Calculating a Delay Period using Overflows

- A delay period can be accurately implemented using timer overflows.

$$F_{overflow} = \frac{F_{osc}}{(4 \times prescaler \times \max count)}$$

- For a 1 second delay at 1:32 prescaler:

$$F_{overflow} = \frac{4\ MHz}{(4 \times 32 \times 256)} \approx 122.070312$$

- Therefore approximately122 overflows are needed for a 1 second delay to be achieved.

Calculating a Delay Period using Overflows

- Period per overflow:

$$T_{overflow} = \frac{1}{F_{overflow}}$$

$$T_{overflow} = \frac{1}{122.070312} \approx 8.192ms$$

PIC16F877A Interrupt Handling

- The PIC16F877A has only a **single interrupt vector** (**00004H**) which means that all interrupt sources is pointed only to a single address in memory to handle the interrupt(s).
- To handle the **RB0/INT external** and/or **Timer0 overflow interrupt**, an **Interrupt Service Routine (ISR)** function must be written.

PIC16F877A Interrupt Handling

```
void interrupt ISR()
{
        GIE = 0; // disables all unmasked interrupts to prevent interrupt overlap
        if (INTF) // check the interrupt flag for RB0/INT
        {
                INTF = 0; // clears the interrupt flag
                /* write the interrupt service routine here
                for RB0/INT external interrupt */
        }
        else if(T0IF) // check the interrupt flag for Timer0
        {
                T0IF = 0; // clears the interrupt flag
                /* write the interrupt service routine here
                for Timer0 overflow interrupt */
        }
        GIE = 1; // enable interrupts again
}
```

University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3201
Embedded Systems

End of Lecture

References:

- De Leon, Hilary L., Microcontroller Programming and Interfacing, 2006.
- Goankar, Ramesh, Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 MCU Family),2007.
- http://www.scriptoriumdesigns.com/embedded/interrupts.php
- PIC16F87X Data Sheet, Microchip Technology Inc. 2003.