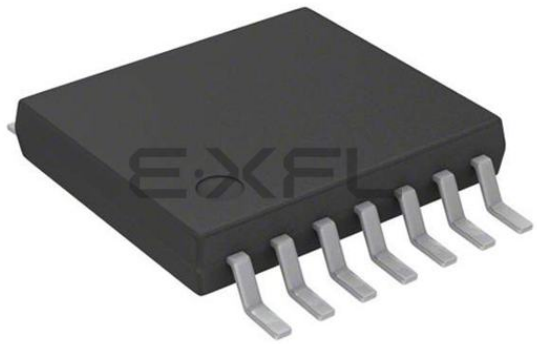


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	12
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	FLASH
EEPROM Size	224 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 11x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	14-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	14-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf15323t-i-st

TABLE 1: PIC16(L)F153XX FAMILY TYPES

Device	Data Sheet Index	Program Flash Memory (KW)	Program Flash Memory (KB)	Storage Area Flash (B)	Data SRAM (bytes)	I/OPins	10-bit ADC	5-bit DAC	Comparator	8-bit/ (with HLT) Timer	16-bit Timer	Window Watchdog Timer	CCP/10-bit PWM	CWG	NCO	CLC	Zero-Cross Detect	Temperature Indicator	Memory Access Partition	Device Information Area	EUSART/ I^2C-SPI	Peripheral Pin Select	Peripheral Module Disable	Debug (1)
PIC16(L)F15313	**(C)**	2	3.5	224	256	6	5	1	1	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	1/1	Y	Y	I
PIC16(L)F15323	**(C)**	2	3.5	224	256	12	11	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	1/1	Y	Y	I
PIC16(L)F15324	**(D)**	4	7	224	512	12	11	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/1	Y	Y	I
PIC16(L)F15325	**(B)**	8	14	224	1024	12	11	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/1	Y	Y	I
PIC16(L)F15344	**(D)**	4	7	224	512	18	17	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/1	Y	Y	I
PIC16(L)F15345	**(B)**	8	14	224	1024	18	17	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/1	Y	Y	I
PIC16(L)F15354	**(A)**	4	7	224	512	25	24	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I
PIC16(L)F15355	**(A)**	8	14	224	1024	25	24	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I
PIC16(L)F15356	**(E)**	16	28	224	2048	25	24	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I
PIC16(L)F15375	**(E)**	8	14	224	1024	36	35	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I
PIC16(L)F15376	**(E)**	16	28	224	2048	36	35	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I
PIC16(L)F15385	**(E)**	8	14	224	1024	44	43	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I
PIC16(L)F15386	**(E)**	16	28	224	2048	44	43	1	2	1	2	Y	2/4	1	1	4	Y	Y	Y	Y	2/2	Y	Y	I

Note 1: I - Debugging integrated on chip.

Data Sheet Index:

- **A: DS40001853** PIC16(L)F15354/5 Data Sheet, 28-Pin
- **B: DS40001865** PIC16(L)F15325/45 Data Sheet, 14/20-Pin
- **C: DS40001897** PIC16(L)F15313/23 Data Sheet, 8/14-Pin
- **D: DS40001889** PIC16(L)F15324/44 Data Sheet, 14/20-Pin
- **E: DS40001866** PIC16(L)F15356/75/76/85/86 Data Sheet, 28/40/48-Pin

Note: For other small form-factor package availability and marking information, visit www.microchip.com/packaging or contact your local sales office.

1.1 Register and Bit Naming Conventions

1.1.1 REGISTER NAMES

When there are multiple instances of the same peripheral in a device, the peripheral control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance, and control identifier. The control registers section will show just one instance of all the register names with an 'x' in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

1.1.2 BIT NAMES

There are two variants for bit names:

- Short name: Bit function abbreviation
- Long name: Peripheral abbreviation + short name

1.1.2.1 Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is *RegisterName*bits.*ShortName*. For example, the enable bit, EN, in the COG1CON0 register can be set in C programs with the instruction `COG1CON0bits.EN = 1`.

Short names are generally not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When this occurs, during the include file generation, all instances of that short bit name are appended with an underscore plus the name of the register in which the bit resides to avoid naming contentions.

1.1.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral thereby making every long bit name unique. The long bit name for the COG1 enable bit is the COG1 prefix, G1, appended with the enable bit short name, EN, resulting in the unique bit name G1EN.

Long bit names are useful in both C and assembly programs. For example, in C the COG1CON0 enable bit can be set with the `G1EN = 1` instruction. In assembly, this bit can be set with the `BSF COG1CON0,G1EN` instruction.

1.1.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. Bit fields adhere only to the short bit naming convention. For example, the three Least Significant bits of the COG1CON0 register contain the mode control bits. The short name for this field is MD. There is no long bit name variant. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the COG1 to the Push-Pull mode:

```
COG1CON0bits.MD = 0x5;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant mode bit has the short bit name MD2 and the long bit name is G1MD2. The following two examples demonstrate assembly program sequences for setting the COG1 to Push-Pull mode:

Example 1:

```
MOVLW  ~(1<<G1MD1)
ANDWF  COG1CON0,F
MOVLW  1<<G1MD2 | 1<<G1MD0
IORWF  COG1CON0,F
```

Example 2:

```
BSF    COG1CON0,G1MD2
BCF    COG1CON0,G1MD1
BSF    COG1CON0,G1MD0
```

1.1.3 REGISTER AND BIT NAMING EXCEPTIONS

1.1.3.1 Status, Interrupt, and Mirror Bits

Status, interrupt enables, interrupt flags, and mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

1.1.3.2 Legacy Peripherals

There are some peripherals that do not strictly adhere to these naming conventions. Peripherals that have existed for many years and are present in almost every device are the exceptions. These exceptions were necessary to limit the adverse impact of the new conventions on legacy code. Peripherals that do adhere to the new convention will include a table in the registers section indicating the long name prefix for each peripheral instance. Peripherals that fall into the exception category will not have this table. These peripherals include, but are not limited to, the following:

- EUSART
- MSSP

4.5 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figure 4-4 through Figure 4-7). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when CALL or CALLW instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer if the STVREN bit is programmed to '0' (Configuration Words). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

Note 1: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, CALLW, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

4.5.1 ACCESSING THE STACK

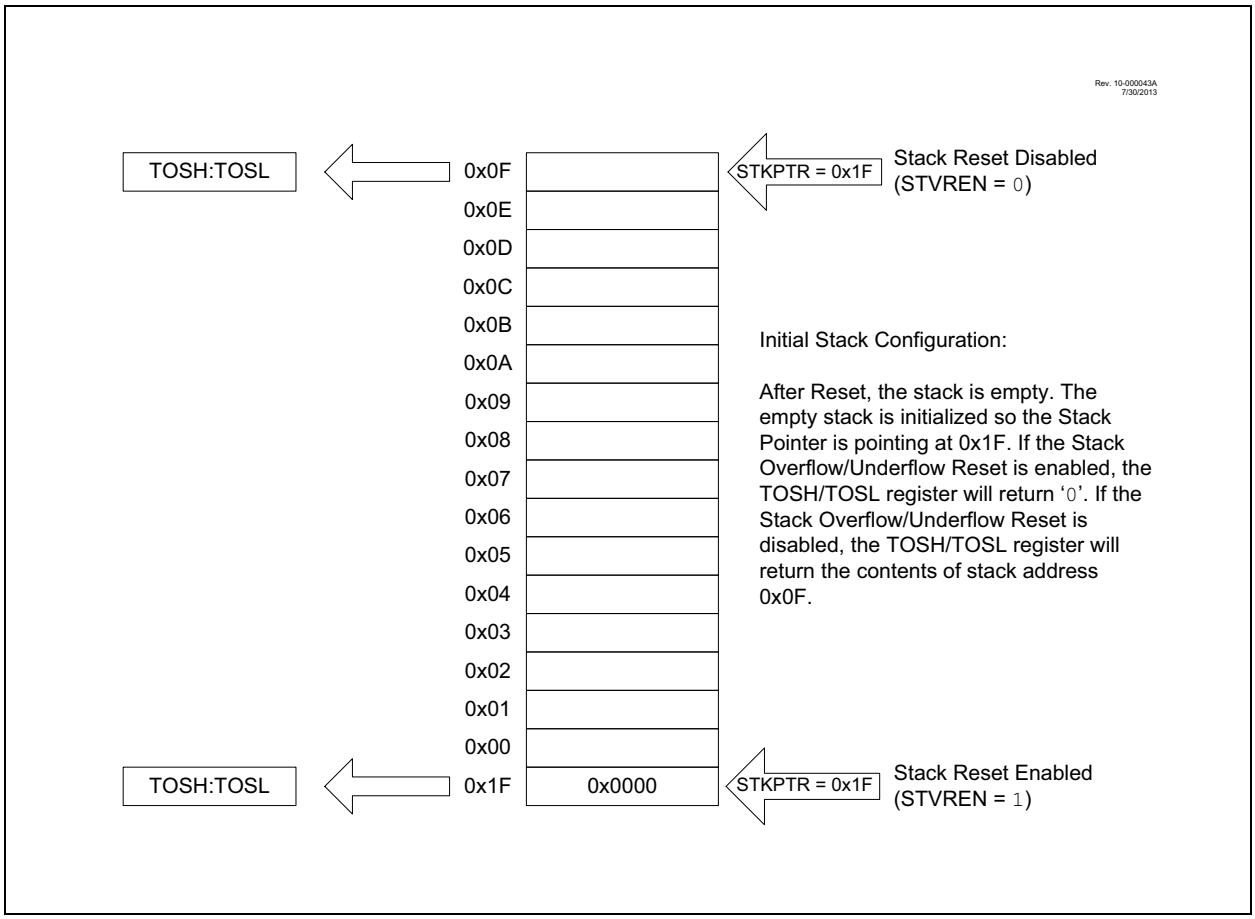
The stack is accessible through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is five bits to allow detection of overflow and underflow.

Note: Care should be taken when modifying the STKPTR while interrupts are enabled.

During normal program operation, CALL, CALLW and interrupts will increment STKPTR while RETLW, RETURN, and RETFIE will decrement STKPTR. STKPTR can be monitored to obtain to value of stack memory left at any given time. The STKPTR always points at the currently used place on the stack. Therefore, a CALL or CALLW will increment the STKPTR and then write the PC, and a return will unload the PC value from the stack and then decrement the STKPTR.

Reference Figure 4-4 through Figure 4-7 for examples of accessing the stack.

FIGURE 4-4: ACCESSING THE STACK EXAMPLE 1



8.3 Register Definitions: Brown-out Reset Control

REGISTER 8-1: BORCON: BROWN-OUT RESET CONTROL REGISTER

R/W-1/u	U-0	U-0	U-0	U-0	U-0	U-0	R-q/u
SBOREN(1)	—	—	—	—	—	—	BORRDY
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

bit 7 **SBOREN:** Software Brown-out Reset Enable bit(1)
If BOREN <1:0> in Configuration Words ≠ 01:
SBOREN is read/write, but has no effect on the BOR.
If BOREN <1:0> in Configuration Words = 01:
1 = BOR Enabled
0 = BOR Disabled

bit 6-1 **Unimplemented:** Read as '0'

bit 0 **BORRDY:** Brown-out Reset Circuit Ready Status bit
1 = The Brown-out Reset circuit is active
0 = The Brown-out Reset circuit is inactive

Note 1: BOREN<1:0> bits are located in Configuration Words.

REGISTER 10-11: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

R/W/HS-0/0	R/W/HS-0/0	U-0	U-0	U-0	U-0	U-0	R/W/HS-0/0
OSFIF	CSWIF	—	—	—	—	—	ADIF
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HS = Hardware set

bit 7 **OSFIF**: Oscillator Fail-Safe Interrupt Flag bit
1 = Oscillator fail-safe interrupt has occurred (must be cleared in software)
0 = No oscillator fail-safe interrupt

bit 6 **CSWIF:** Clock Switch Complete Interrupt Flag bit
1 = The clock switch module indicates an interrupt condition and is ready to complete the clock switch operation (must be cleared in software)
0 = The clock switch does not indicate an interrupt condition

bit 5-1 **Unimplemented:** Read as '0'

bit 0 **ADIF**: Analog-to-Digital Converter (ADC) Interrupt Flag bit
1 = An A/D conversion or complex operation has completed (must be cleared in software)
0 = An A/D conversion or complex operation is not complete

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Enable bit, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY

```
; This write routine assumes the following:
; 1. 64 bytes of data are loaded, starting at the address in DATA_ADDR
; 2. Each word of data to be written is made up of two adjacent bytes in DATA_ADDR,
;    stored in little endian format
; 3. A valid starting address (the least significant bits = 00000) is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in common RAM (locations 0x70 - 0x7F)
; 5. NVM interrupts are not taken into account

        BANKSEL     NVMADRH
        MOVF        ADDRH,W
        MOVWF       NVMADRH             ; Load initial address
        MOVF        ADDRL,W
        MOVWF       NVMADRL
        MOVLW       LOW DATA_ADDR       ; Load initial data address
        MOVWF       FSR0L
        MOVLW       HIGH DATA_ADDR
        MOVWF       FSR0H
        BCF         NVMCON1,NVMREGS     ; Set Program Flash Memory as write location
        BSF         NVMCON1,WREN        ; Enable writes
        BSF         NVMCON1,LWLO        ; Load only write latches

LOOP
        MOVIW       FSR0++
        MOVWF       NVMDATL             ; Load first data byte
        MOVIW       FSR0++
        MOVWF       NVMDATH             ; Load second data byte

        MOVF        NVMADRL,W
        XORLW       0x1F                ; Check if lower bits of address are 00000
        ANDLW       0x1F                ; and if on last of 32 addresses
        BTFSC       STATUS,Z            ; Last of 32 words?
        GOTO        START_WRITE         ; If so, go write latches into memory

        CALL        UNLOCK_SEQ          ; If not, go load latch
        INCF        NVMADRL,F           ; Increment address
        GOTO        LOOP

START_WRITE
        BCF         NVMCON1,LWLO        ; Latch writes complete, now write memory
        CALL        UNLOCK_SEQ          ; Perform required unlock sequence
        BCF         NVMCON1,WREN        ; Disable writes

UNLOCK_SEQ
        MOVLW       55h
        BCF         INTCON,GIE          ; Disable interrupts
        MOVWF       NVMCON2             ; Begin unlock sequence
        MOVLW       AAh
        MOVWF       NVMCON2
        BSF         NVMCON1,WR
        BSF         INTCON,GIE          ; Unlock sequence complete, re-enable interrupts
        return
```

REGISTER 13-5: NVMCON1: NONVOLATILE MEMORY CONTROL 1 REGISTER

U-0	R/W-0/0	R/W-0/0	R/W/HC-0/0	R/W/HC-x/q	R/W-0/0	R/S/HC-0/0	R/S/HC-0/0
—	NVMREGS	LWLO	FREE	WRERR(1,2,3)	WREN	WR(4,5,6)	RD
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
S = Bit can only be set	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HC = Bit is cleared by hardware

bit 7 **Unimplemented:** Read as '0'

bit 6 **NVMREGS:** Configuration Select bit
1 = Access DIA, DCI, Configuration, User ID and Device ID Registers
0 = Access PFM

bit 5 **LWLO:** Load Write Latches Only bit
When FREE = 0:
1 = The next WR command updates the write latch for this word within the row; no memory operation is initiated.
0 = The next WR command writes data or erases
Otherwise: The bit is ignored

bit 4 **FREE:** PFM Erase Enable bit
When NVMREGS:NVMADR points to a PFM location:
1 = Performs an erase operation with the next WR command; the 32-word pseudo-row containing the indicated address is erased (to all 1s) to prepare for writing.
0 = All erase operations have completed normally

bit 3 **WRERR:** Program/Erase Error Flag bit(1,2,3)
This bit is normally set by hardware.
1 = A write operation was interrupted by a Reset, interrupted unlock sequence, or WR was written to one while NVMADR points to a write-protected address.
0 = The program or erase operation completed normally

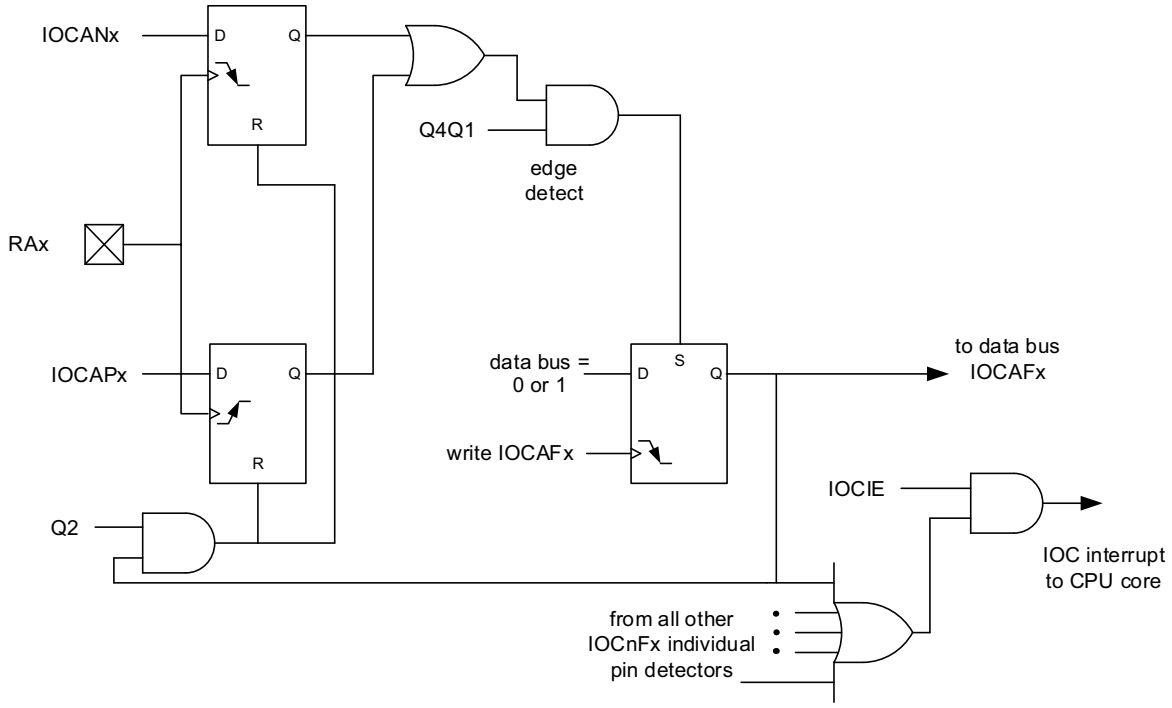
bit 2 **WREN:** Program/Erase Enable bit
1 = Allows program/erase cycles
0 = Inhibits programming/erasing of program Flash

bit 1 **WR:** Write Control bit(4,5,6)
When NVMREG:NVMADR points to a PFM location:
1 = Initiates the operation indicated by Table 13-4
0 = NVM program/erase operation is complete and inactive.

bit 0 **RD:** Read Control bit(7)
1 = Initiates a read at address = NVMADR1, and loads data to NVMDAT Read takes one instruction cycle and the bit is cleared when the operation is complete. The bit can only be set (not cleared) in software.
0 = NVM read operation is complete and inactive

Note 1: Bit is undefined while WR = 1.
2: Bit must be cleared by software; hardware will not clear this bit.
3: Bit may be written to '1' by software in order to implement test sequences.
4: This bit can only be set by following the unlock sequence of **Section 13.3.2 "NVM Unlock Sequence"**.
5: Operations are self-timed, and the WR bit is cleared by hardware when complete.
6: Once a write operation is initiated, setting this bit to zero will have no effect.

FIGURE 17-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)



Note 1: See Table 8-1 for BOR Active Conditions.

20.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting

20.1.1 PORT CONFIGURATION

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin will be configured for analog by setting the associated TRIS and ANSEL bits. Refer to **Section 14.0 "I/O Ports"** for more information.

Note:	Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

20.1.2 CHANNEL SELECTION

There are several channel selections available:

- Six Port A channels
- Six Port C channels (PIC16(L)F15323 only)
- Temperature Indicator
- DAC output
- Fixed Voltage Reference (FVR)
- AVSS (Ground)

The CHS<5:0> bits of the ADCON0 register (Register 20-1) determine which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion. Refer to **Section 20.2 "ADC Operation"** for more information.

Note:	It is recommended that when switching from an ADC channel of a higher voltage to a channel of a lower voltage, that the user selects the VSS channel before connecting to the channel with the lower voltage. If the ADC does not have a dedicated VSS input channel, the VSS selection (DAC1R<4:0> = b'00000') through the DAC output channel can be used. If the DAC is in use, a free input channel can be connected to VSS, and can be used in place of the DAC.

20.1.3 ADC VOLTAGE REFERENCE

The ADPREF<1:0> bits of the ADCON1 register provides control of the positive voltage reference. The positive voltage reference can be:

- VREF+ pin
- VDD
- FVR 2.048V
- FVR 4.096V (Not available on LF devices)

The ADPREF bit of the ADCON1 register provides control of the negative voltage reference. The negative voltage reference can be:

- VREF- pin
- VSS

See **Section 18.0 "Fixed Voltage Reference (FVR)"** for more details on the Fixed Voltage Reference.

20.1.4 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCS<2:0> bits of the ADCON1 register. There are seven possible clock options:

- FOSC/2
- FOSC/4
- FOSC/8
- FOSC/16
- FOSC/32
- FOSC/64
- ADCRC (dedicated RC oscillator)

The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11.5 TAD periods as shown in Figure 20-2.

For correct conversion, the appropriate TAD specification must be met. Refer to Table 37-13 for more information. Table 20-1 gives examples of appropriate ADC clock selections.

Note:	Unless using the ADCRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.

25.0 TIMER0 MODULE

The Timer0 module is an 8/16-bit timer/counter with the following features:

- 16-bit timer/counter
- 8-bit timer/counter with programmable period
- Synchronous or asynchronous operation
- Selectable clock sources
- Programmable prescaler (independent of Watchdog Timer)
- Programmable postscaler
- Operation during Sleep mode
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals

25.1 Timer0 Operation

Timer0 can operate as either an 8-bit timer/counter or a 16-bit timer/counter. The mode is selected with the T016BIT bit of the T0CON register.

25.1.1 16-BIT MODE

In normal operation, TMR0 increments on the rising edge of the clock source. A 15-bit prescaler on the clock input gives several prescale options (see prescaler control bits, T0CKPS<3:0> in the T0CON1 register).

25.1.1.1 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode. It is actually a buffered version of the real high byte of Timer0, which is neither directly readable nor writable (see Figure 25-1). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte was valid, due to a rollover between successive reads of the high and low byte.

Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

25.1.2 8-BIT MODE

In normal operation, TMR0 increments on the rising edge of the clock source. A 15-bit prescaler on the clock input gives several prescale options (see prescaler control bits, T0CKPS<3:0> in the T0CON1 register).

The value of TMR0L is compared to that of the Period buffer, a copy of TMR0H, on each clock cycle. When the two values match, the following events happen:

- TMR0_out goes high for one prescaled clock period
- TMR0L is reset
- The contents of TMR0H are copied to the period buffer

In 8-bit mode, the TMR0L and TMR0H registers are both directly readable and writable. The TMR0L register is cleared on any device Reset, while the TMR0H register initializes at FFh.

Both the prescaler and postscaler counters are cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset – Power-on Reset (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or
- Brown-out Reset (BOR)

25.1.3 COUNTER MODE

In Counter mode, the prescaler is normally disabled by setting the T0CKPS bits of the T0CON1 register to '0000'. Each rising edge of the clock input (or the output of the prescaler if the prescaler is used) increments the counter by '1'.

25.1.4 TIMER MODE

In Timer mode, the Timer0 module will increment every instruction cycle as long as there is a valid clock signal and the T0CKPS bits of the T0CON1 register (Register 25-2) are set to '0000'. When a prescaler is added, the timer will increment at the rate based on the prescaler value.

25.1.5 ASYNCHRONOUS MODE

When the T0ASYNC bit of the T0CON1 register is set (T0ASYNC = '1'), the counter increments with each rising edge of the input source (or output of the prescaler, if used). Asynchronous mode allows the counter to continue operation during Sleep mode provided that the clock also continues to operate during Sleep.

25.1.6 SYNCHRONOUS MODE

When the T0ASYNC bit of the T0CON1 register is clear (T0ASYNC = 0), the counter clock is synchronized to the system oscillator (FOSC/4). When operating in Synchronous mode, the counter clock frequency cannot exceed FOSC/4.

27.7 Register Definitions: Timer2 Control

REGISTER 27-1: T2CLKCON: TIMER2 CLOCK SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CS<3:0>			
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **CS<3:0>:** Timer2 Clock Select bits

1111 = Reserved
1110 = LC4_out
1101 = LC3_out
1100 = LC2_out
1011 = LC1_out
1010 = ZCD1_output
1001 = NCO1_out
1000 = CLKR
0111 = Reserved
0110 = MFINTOSC (31.25 kHz)
0101 = MFINTOSC (500 kHz)
0100 = LFINTOSC
0011 = HFINTOSC (32 MHz)
0010 = FOSC
0001 = FOSC/4
0000 = T2CKIPPS

28.1.2 TIMER1 MODE RESOURCE

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

See **Section 26.0 "Timer1 Module with Gate Control"** for more information on configuring Timer1.

28.1.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit of the PIE6 register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIR6 register following any change in Operating mode.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4).

28.1.4 CCP PRESCALER

There are four prescaler settings specified by the CCPxMODE<3:0> bits of the CCPxCON register. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. Example 28-1 demonstrates the code to perform this function.

EXAMPLE 28-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
BANKSEL CCPxCON     ;Set Bank bits to point
                    ;to CCPxCON
CLRF    CCPxCON     ;Turn CCP module off
MOVLW   NEW_CAPT_PS ;Load the W reg with
                    ;the new prescaler
                    ;move value and CCP ON
MOVWF   CCPxCON     ;Load CCPxCON with this
                    ;value
```

28.1.5 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock (FOSC/4), or by an external clock source.

When Timer1 is clocked by FOSC/4, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep when Timer1 is clocked by an external clock source.

28.2 Compare Mode

Compare mode makes use of the 16-bit Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMR1H:TMR1L register pair. When a match occurs, one of the following events can occur:

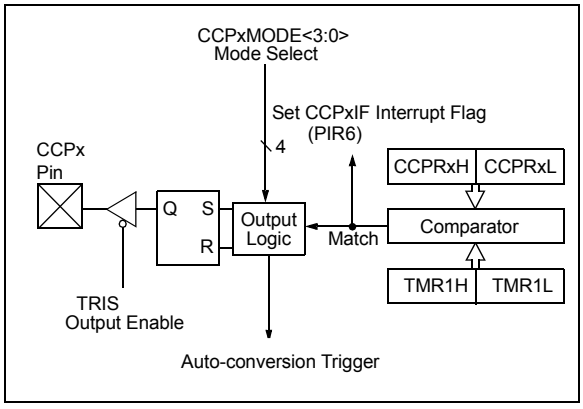
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Generate an Auto-conversion Trigger
- Generate a Software Interrupt

The action on the pin is based on the value of the CCPxMODE<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set, and an ADC conversion can be triggered, if selected.

All Compare modes can generate an interrupt and trigger and ADC conversion.

Figure 28-2 shows a simplified diagram of the compare operation.

FIGURE 28-2: COMPARE MODE OPERATION BLOCK DIAGRAM



REGISTER 30-3: CWG1DBR: CWG1 RISING DEAD-BAND COUNTER REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	DBR<5:0>					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **DBR<5:0>:** Rising Event Dead-Band Value for Counter bits

REGISTER 30-4: CWG1DBF: CWG1 FALLING DEAD-BAND COUNTER REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	DBF<5:0>					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **DBF<5:0>:** Falling Event Dead-Band Value for Counter bits

REGISTER 31-9: CLCxGLS2: GATE 2 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
LCxG3D4T	LCxG3D4N	LCxG3D3T	LCxG3D3N	LCxG3D2T	LCxG3D2N	LCxG3D1T	LCxG3D1N
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7 **LCxG3D4T:** Gate 2 Data 4 True (non-inverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 2
0 = CLCIN3 (true) is not gated into CLCx Gate 2

bit 6 **LCxG3D4N:** Gate 2 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 2
0 = CLCIN3 (inverted) is not gated into CLCx Gate 2

bit 5 **LCxG3D3T:** Gate 2 Data 3 True (non-inverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 2
0 = CLCIN2 (true) is not gated into CLCx Gate 2

bit 4 **LCxG3D3N:** Gate 2 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 2
0 = CLCIN2 (inverted) is not gated into CLCx Gate 2

bit 3 **LCxG3D2T:** Gate 2 Data 2 True (non-inverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 2
0 = CLCIN1 (true) is not gated into CLCx Gate 2

bit 2 **LCxG3D2N:** Gate 2 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 2
0 = CLCIN1 (inverted) is not gated into CLCx Gate 2

bit 1 **LCxG3D1T:** Gate 2 Data 1 True (non-inverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 2
0 = CLCIN0 (true) is not gated into CLCx Gate 2

bit 0 **LCxG3D1N:** Gate 2 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 2
0 = CLCIN0 (inverted) is not gated into CLCx Gate 2

32.5.2.2 7-bit Reception with AHEN and DHEN

Slave device reception with AHEN and DHEN set operate the same as without these options with extra interrupts and clock stretching added after the eighth falling edge of SCL. These additional interrupts allows time for the slave software to decide whether it wants to $\overline{ACK}$ the receive address or data byte.

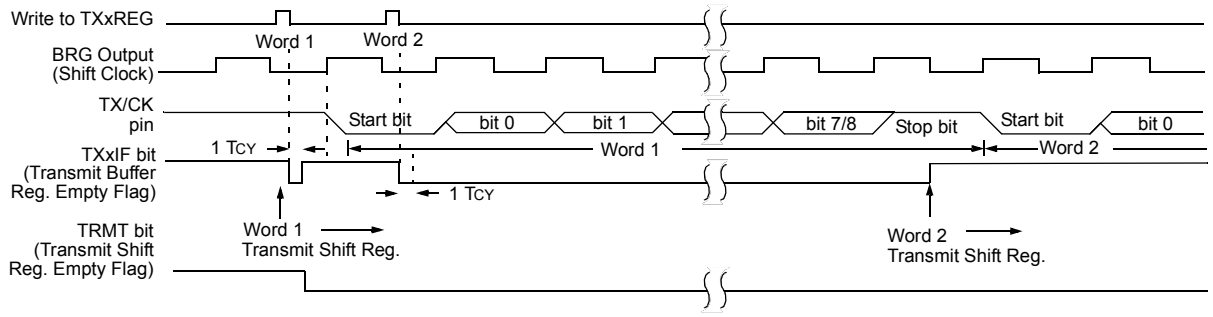
This list describes the steps that need to be taken by slave software to use these options for I^2C communication. Figure 32-16 displays a module using both address and data holding. Figure 32-17 includes the operation with the SEN bit of the SSP1CON2 register set.

1. S bit of SSP1STAT is set; SSP1IF is set if interrupt on Start detect is enabled.
2. Matching address with R/$\overline{W}$ bit clear is clocked in. SSP1IF is set and CKP cleared after the eighth falling edge of SCL.
3. Slave clears the SSP1IF.
4. Slave can look at the ACKTIM bit of the SSP1CON3 register to determine if the SSP1IF was after or before the $\overline{ACK}$.
5. Slave reads the address value from SSP1BUF, clearing the BF flag.
6. Slave sets $\overline{ACK}$ value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSP1IF is set after an $\overline{ACK}$, not after a NACK.
9. If SEN = 1 the slave hardware will stretch the clock after the $\overline{ACK}$.
10. Slave clears SSP1IF.

Note:	SSP1IF is still set after the ninth falling edge of SCL even if there is no clock stretching and BF has been cleared. Only if NACK is sent to master is SSP1IF not set

11. SSP1IF set and CKP cleared after eighth falling edge of SCL for a received data byte.
12. Slave looks at ACKTIM bit of SSP1CON3 to determine the source of the interrupt.
13. Slave reads the received data from SSP1BUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an $\overline{ACK}$ = 1, or the master sending a Stop condition. If a Stop is sent and Interrupt on Stop Detect is disabled, the slave will only know by polling the P bit of the SSP1STAT register.

FIGURE 33-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)



Note: This timing diagram shows two consecutive transmissions.

33.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 33-2. The data is received on the RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all eight or nine bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the EUSART receiver. The FIFO and RSR registers are not directly accessible by software. Access to the received data is via the RC1REG register.

33.1.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:

- CREN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the CREN bit of the RC1STA register enables the receiver circuitry of the EUSART. Clearing the SYNC bit of the TX1STA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RC1STA register enables the EUSART. The programmer must set the corresponding TRIS bit to configure the RX/DT I/O pin as an input.

Note:	If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

33.1.2.2 Receiving Data

The receiver data recovery circuit initiates character reception on the falling edge of the first bit. The first bit, also known as the Start bit, is always a zero. The data recovery circuit counts one-half bit time to the center of the Start bit and verifies that the bit is still a zero. If it is not a zero then the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit. If the Start bit zero verification succeeds then the data recovery circuit counts a full bit time to the center of the next bit. The bit is then sampled by a majority detect circuit and the resulting '0' or '1' is shifted into the RSR. This repeats until all data bits have been sampled and shifted into the RSR. One final bit time is measured and the level sampled. This is the Stop bit, which is always a '1'. If the data recovery circuit samples a '0' in the Stop bit position then a framing error is set for this character, otherwise the framing error is cleared for this character. See **Section 33.1.2.4 "Receive Framing Error"** for more information on framing errors.

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the EUSART receive FIFO and the RX1IF interrupt flag bit of the PIR3 register is set. The top character in the FIFO is transferred out of the FIFO by reading the RC1REG register.

Note:	If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See **Section 33.1.2.5 "Receive Overrun Error"** for more information on overrun errors.

FIGURE 33-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION

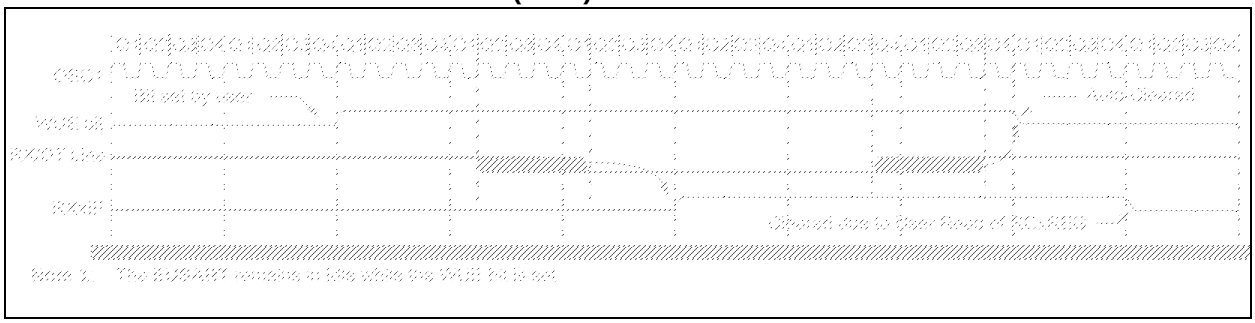
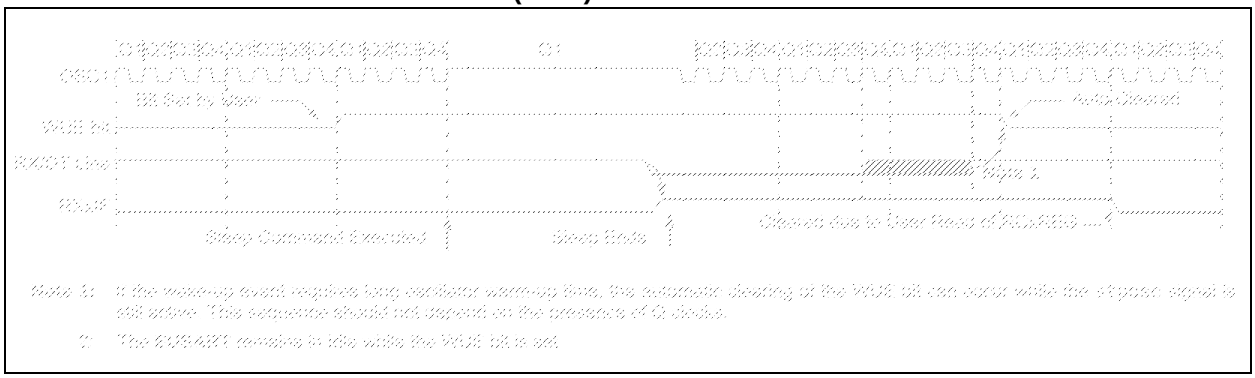


FIGURE 33-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



33.3.4 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. A Break character consists of a Start bit, followed by 12 '0' bits and a Stop bit.

To send a Break character, set the SENDB and TXEN bits of the TX1STA register. The Break character transmission is then initiated by a write to the TX1REG. The value of data written to TX1REG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit of the TX1STA register indicates when the transmit operation is active or idle, just as it does during normal transmission. See Figure 33-9 for the timing of the Break character sequence.

33.3.4.1 Break and Sync Transmit Sequence

The following sequence will start a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to enable the Break sequence.
3. Load the TX1REG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TX1REG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware and the Sync character is then transmitted.

When the TX1REG becomes empty, as indicated by the TX1IF, the next data byte can be written to TX1REG.

36.3 Instruction Descriptions

ADDFSR — Add Literal to FSRn

Syntax:	[*label*] ADDFSR FSRn, k
Operands:	-32 ≤ k ≤ 31 n ∈ [0, 1]
Operation:	FSR(n) + k → FSR(n)
Status Affected:	None
Description:	The signed 6-bit literal ‘k’ is added to the contents of the FSRnH:FSRnL register pair. FSRn is limited to the range 0000h-FFFFh. Moving beyond these bounds will cause the FSR to wrap-around.

ADDLW — Add literal and W

Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the 8-bit literal ‘k’ and the result is placed in the W register.

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

ADDWFC — ADD W and CARRY bit to f

Syntax:	[*label*] ADDWFC f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) + (C) → dest
Status Affected:	C, DC, Z
Description:	Add W, the Carry flag and data memory location ‘f’. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed in data memory location ‘f’.

ANDLW — AND literal with W

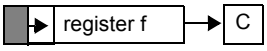
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND’ed with the 8-bit literal ‘k’. The result is placed in the W register.

ANDWF — AND W with f

Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

ASRF — Arithmetic Right Shift

Syntax:	[*label*] ASRF f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<7>)→ dest<7> (f<7:1>) → dest<6:0>, (f<0>) → C,
Status Affected:	C, Z
Description:	The contents of register ‘f’ are shifted one bit to the right through the Carry flag. The MSb remains unchanged. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’.



BCF **Bit Clear f**

Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BRA **Relative Branch**

Syntax:	[*label*] BRA label [*label*] BRA $+k
Operands:	-256 ≤ label - PC + 1 ≤ 255 -256 ≤ k ≤ 255
Operation:	(PC) + 1 + k → PC
Status Affected:	None
Description:	Add the signed 9-bit literal 'k' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + k. This instruction is a 2-cycle instruction. This branch has a limited range.

BRW **Relative Branch with W**

Syntax:	[*label*] BRW
Operands:	None
Operation:	(PC) + (W) → PC
Status Affected:	None
Description:	Add the contents of W (unsigned) to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + (W). This instruction is a 2-cycle instruction.

BSF **Bit Set f**

Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSC **Bit Test f, Skip if Clear**

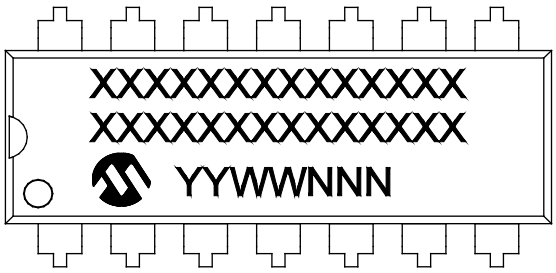
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2-cycle instruction.

BTFSS **Bit Test f, Skip if Set**

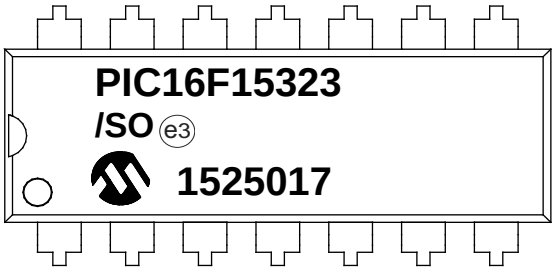
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.

40.1 Package Marking Information

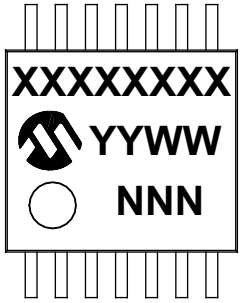
14-Lead PDIP (300 mil)



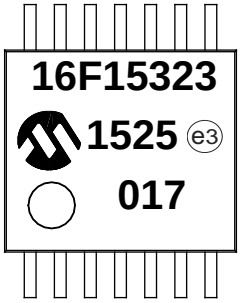
Example



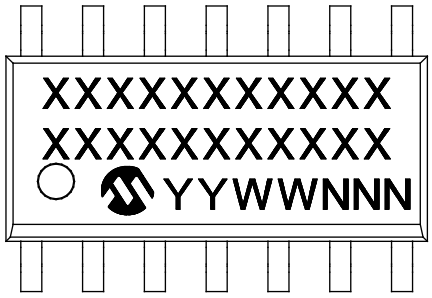
14-Lead TSSOP (4.4 mm)



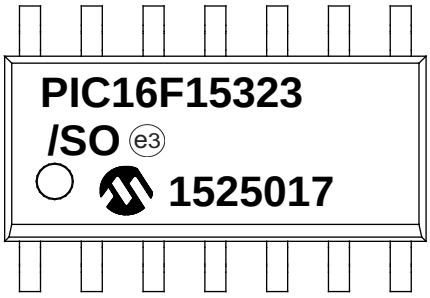
Example



14-Lead SOIC (3.90 mm)



Example



Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
		Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.