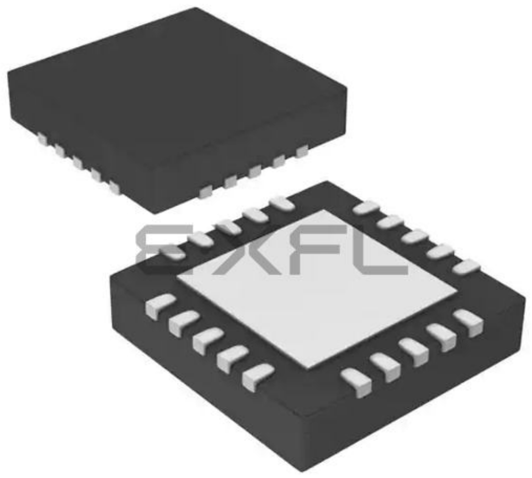
Microchip Technology - PIC18LF14K22T-I/ML Datasheet



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	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	17
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-VFQFN Exposed Pad
Supplier Device Package	20-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf14k22t-i-ml

2.8 Oscillator Start-up Timer

The Primary External Oscillator, when configured for LP, XT or HS modes, incorporates an Oscillator Start-up Timer (OST). The OST ensures that the oscillator starts and provides a stable clock to the oscillator module. The OST times out when 1024 oscillations on OSC1 have occurred. During the OST period, with the system clock set to the Primary External Oscillator, the program counter does not increment suspending program execution. The OST period will occur following:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Wake-up from Sleep
- Oscillator being enabled
- Expiration of Power-up Timer (PWRT)

In order to minimize latency between external oscillator start-up and code execution, the Two-Speed Start-up mode can be selected. See **Section 2.11 “Two-Speed Start-up Mode”** for more information.

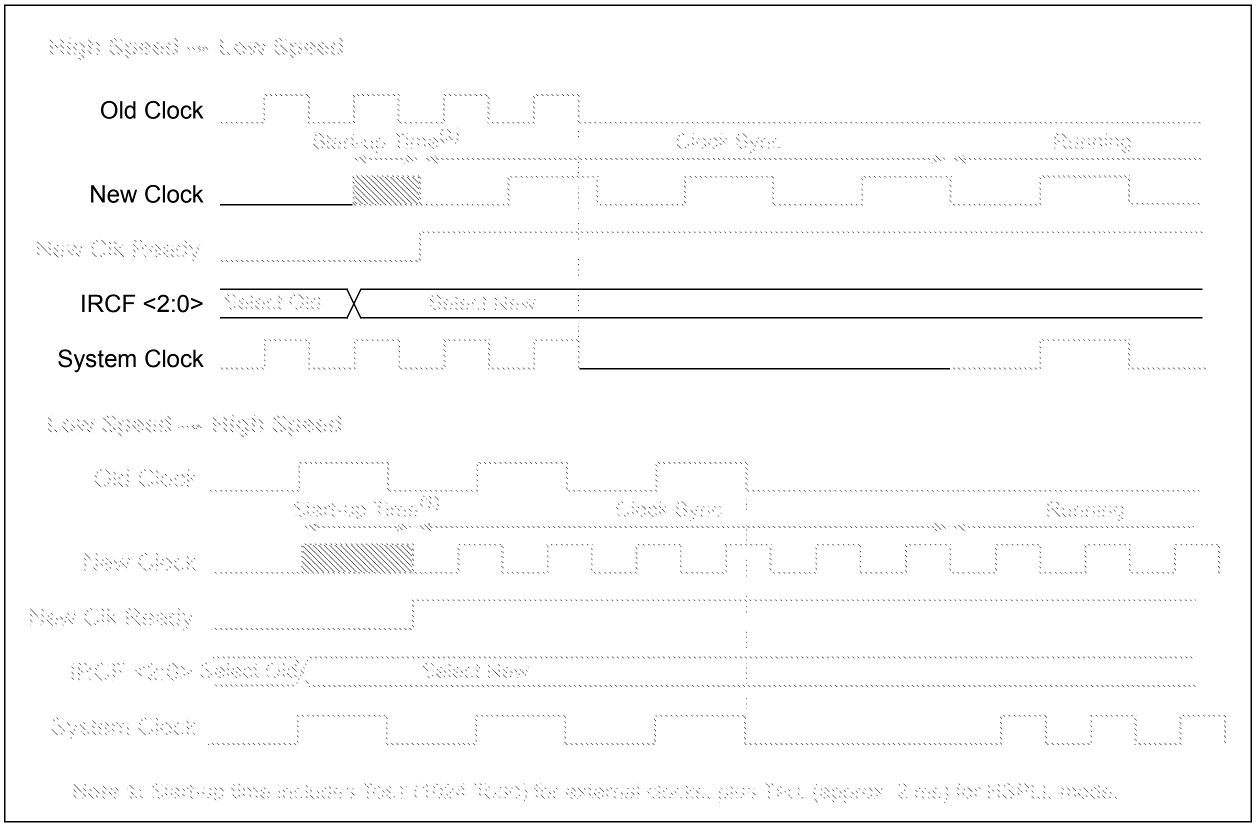
2.9 Clock Switching

The device contains circuitry to prevent clock “glitches” due to a change of the system clock source. To accomplish this, a short pause in the system clock occurs during the clock switch. If the new clock source is not stable (e.g., OST is active), the device will continue to execute from the old clock source until the new clock source becomes stable. The timing of a clock switch is as follows:

1. SCS<1:0> bits of the OSCCON register are modified.
2. The system clock will continue to operate from the old clock until the new clock is ready.
3. Clock switch circuitry waits for two consecutive rising edges of the old clock after the new clock is ready.
4. The system clock is held low, starting at the next falling edge of the old clock.
5. Clock switch circuitry waits for an additional two rising edges of the new clock.
6. On the next falling edge of the new clock, the low hold on the system clock is release and the new clock is switched in as the system clock.
7. Clock switch is complete.

Refer to Figure 2-5 for more details.

FIGURE 2-5: CLOCK SWITCH TIMING



3.3.2 ACCESS BANK

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 3-5 and Figure 3-6).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the ‘a’ parameter in the instruction). When ‘a’ is equal to ‘`1`’, the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When ‘a’ is ‘`0`’, however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. For 8-bit addresses of 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = `1`). This is discussed in more detail in **Section 3.5.3 “Mapping the Access Bank in Indexed Literal Offset Mode”**.

3.3.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM, which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

3.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top portion of Bank 15 (F60h to FFFh). A list of these registers is given in Table 3-1 and Table 3-2.

The SFRs can be classified into two sets: those associated with the “core” device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and Interrupt registers are described in their respective chapters, while the ALU’s STATUS register is described later in this section. Registers related to the operation of a peripheral feature are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as ‘`0`’s.

5.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD control bit of the EECON1 register and then set control bit, RD. The data is available on the very next instruction cycle; therefore, the EEDATA register can be read by the next instruction. EEDATA will hold this value until another read operation, or until it is written to by the user (during a write operation).

The basic process is shown in Example 5-1.

5.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADR register and the data written to the EEDATA register. The sequence in Example 5-2 must be followed to initiate the write cycle.

The write will not begin if this sequence is not exactly followed (write 55h to EECON2, write 0AAh to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, EECON1, EEADR and EEDATA cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. Both WR and WREN cannot be set with the same instruction.

At the completion of the write cycle, the WR bit is cleared by hardware and the EEPROM Interrupt Flag bit, EEIF, is set. The user may either enable this interrupt or poll this bit. EEIF must be cleared by software.

5.5 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

EXAMPLE 5-1: DATA EEPROM READ

```
        MOVLW   DATA_EE_ADDR   ;
        MOVWF   EEADR          ; Data Memory Address to read
        BCF     EECON1, EEPGD  ; Point to DATA memory
        BCF     EECON1, CFGS   ; Access EEPROM
        BSF     EECON1, RD     ; EEPROM Read
        MOVF    EEDATA, W      ; W = EEDATA
```

EXAMPLE 5-2: DATA EEPROM WRITE

```
                MOVLW   DATA_EE_ADDR_LOW   ;
                MOVWF   EEADR              ; Data Memory Address to write
                MOVLW   DATA_EE_DATA       ;
                MOVWF   EEDATA             ; Data Memory Value to write
                BCF     EECON1, EEPGD      ; Point to DATA memory
                BCF     EECON1, CFGS       ; Access EEPROM
                BSF     EECON1, WREN       ; Enable writes
                BCF     INTCON, GIE        ; Disable Interrupts
                MOVLW   55h                ;
    Required    MOVWF   EECON2             ; Write 55h
    Sequence    MOVLW   0AAh               ;
                MOVWF   EECON2             ; Write 0AAh
                BSF     EECON1, WR         ; Set WR bit to begin write
                BSF     INTCON, GIE        ; Enable Interrupts

                                           ; User code execution
                BCF     EECON1, WREN       ; Disable writes on write complete (EEIF set)
```

REGISTER 7-2: INTCON2: INTERRUPT CONTROL 2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RABPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7	**$\overline{\text{RABPU}}$:** PORTA and PORTB Pull-up Enable bit 1 = PORTA and PORTB pull-ups are disabled 0 = PORTA and PORTB pull-ups are enabled provided that the pin is an input and the corresponding WPUA and WPUB bits are set.
bit 6	**INTEDG0:** External Interrupt 0 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 5	**INTEDG1:** External Interrupt 1 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 4	**INTEDG2:** External Interrupt 2 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 3	**Unimplemented:** Read as ‘0’
bit 2	**TMR0IP:** TMR0 Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	**Unimplemented:** Read as ‘0’
bit 0	**RABIP:** RA and RB Port Change Interrupt Priority bit 1 = High priority 0 = Low priority

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. User software might ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 7-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
OSCFIF	C1IF	C2IF	EEIF	BCLIF	—	TMR3IF	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7	**OSCFIF:** Oscillator Fail Interrupt Flag bit 1 = Device oscillator failed, clock input has changed to HFINTOSC (must be cleared by software) 0 = Device clock operating
bit 6	**C1IF:** Comparator C1 Interrupt Flag bit 1 = Comparator C1 output has changed (must be cleared by software) 0 = Comparator C1 output has not changed
bit 5	**C2IF:** Comparator C2 Interrupt Flag bit 1 = Comparator C2 output has changed (must be cleared by software) 0 = Comparator C2 output has not changed
bit 4	**EEIF:** Data EEPROM/Flash Write Operation Interrupt Flag bit 1 = The write operation is complete (must be cleared by software) 0 = The write operation is not complete or has not been started
bit 3	**BCLIF:** Bus Collision Interrupt Flag bit 1 = A bus collision occurred (must be cleared by software) 0 = No bus collision occurred
bit 2	**Unimplemented:** Read as ‘0’
bit 1	**TMR3IF:** TMR3 Overflow Interrupt Flag bit 1 = TMR3 register overflowed (must be cleared by software) 0 = TMR3 register did not overflow
bit 0	**Unimplemented:** Read as ‘0’

REGISTER 8-1: PORTA: PORTA REGISTER

U-0	U-0	R/W-x	R/W-x	R-x	R/W-x	R/W-x	R/W-x
—	—	RA5	RA4	RA3	RA2	RA1	RA0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7-6	**Unimplemented**: Read as ‘0’
bit 5-0	**RA<5:0>**: PORTA I/O Pin bit(1) 1 = Port pin is > V_{IH} 0 = Port pin is < V_{IL}

Note 1: The RA3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0). Otherwise, RA3 reads as ‘0’. This bit is read-only.

REGISTER 8-2: TRISA: PORTA TRI-STATE REGISTER

U-0	U-0	R/W-1	R/W-1	U-1	R/W-1	R/W-1	R/W-1
—	—	TRISA5	TRISA4	—	TRISA2	TRISA1	TRISA0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7-6	**Unimplemented**: Read as ‘0’
bit 5-4	**TRISA<5:4>:** PORTA Tri-State Control bit(1) 1 = PORTA pin configured as an input (tri-stated) 0 = PORTA pin configured as an output
bit 3	**Unimplemented**: Read as ‘1’
bit 2-0	**TRISA<2:0>:** PORTA Tri-State Control bit(1) 1 = PORTA pin configured as an input (tri-stated) 0 = PORTA pin configured as an output

Note 1: TRISA<5:4> always reads ‘1’ in XT, HS and LP Oscillator modes.

EXAMPLE 10-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```
RTCinit
           MOVLW    80h               ; Preload TMR1 register pair
           MOVWF    TMR1H             ; for 1 second overflow
           CLRF     TMR1L
           MOVLW    b'00001111'       ; Configure for external clock,
           MOVWF    T1CON             ; Asynchronous operation, external oscillator
           CLRF     secs              ; Initialize timekeeping registers
           CLRF     mins              ;
           MOVLW    .12
           MOVWF    hours
           BSF      PIE1, TMR1IE      ; Enable Timer1 interrupt
           RETURN
RTCisr
           BSF      TMR1H, 7          ; Preload for 1 sec overflow
           BCF      PIR1, TMR1IF      ; Clear interrupt flag
           INCF     secs, F           ; Increment seconds
           MOVLW    .59               ; 60 seconds elapsed?
           CPFSGT   secs
           RETURN                     ; No, done
           CLRF     secs              ; Clear seconds
           INCF     mins, F           ; Increment minutes
           MOVLW    .59               ; 60 minutes elapsed?
           CPFSGT   mins
           RETURN                     ; No, done
           CLRF     mins              ; clear minutes
           INCF     hours, F          ; Increment hours
           MOVLW    .23               ; 24 hours elapsed?
           CPFSGT   hours
           RETURN                     ; No, done
           CLRF     hours             ; Reset hours
           RETURN                     ; Done
```

TABLE 10-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	245
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	248
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	248
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	248
TMR1H	Timer1 Register, High Byte								246
TMR1L	Timer1 Register, Low Byte								246
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	248
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	246

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: Unimplemented, read as'1'.

13.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

PIC18(L)F1XK22 devices have one ECCP (Capture/Compare/PWM) module. The module contains a 16-bit register which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

CCP1 is implemented as a standard CCP module with enhanced PWM capabilities. These include:

- Provision for two or four output channels
- Output steering
- Programmable polarity
- Programmable dead-band control
- Automatic shutdown and restart

The enhanced features are discussed in detail in **Section 13.4 "PWM (Enhanced Mode)"**.

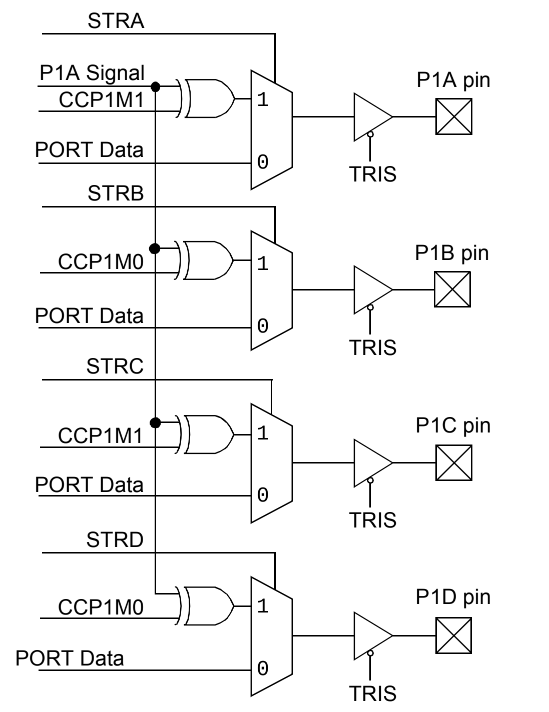
REGISTER 13-1: CCP1CON: ENHANCED CAPTURE/COMPARE/PWM CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6	**P1M<1:0>:** Enhanced PWM Output Configuration bits If CCP1M<3:2> = 00, 01, 10: xx = P1A assigned as Capture/Compare input/output; P1B, P1C, P1D assigned as port pins If CCP1M<3:2> = 11: 00 = Single output: P1A, P1B, P1C and P1D controlled by steering (See **Section 13.4.7 "Pulse Steering Mode"**). 01 = Full-bridge output forward: P1D modulated; P1A active; P1B, P1C inactive 10 = Half-bridge output: P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins 11 = Full-bridge output reverse: P1B modulated; P1C active; P1A, P1D inactive
bit 5-4	**DC1B<1:0>**: PWM Duty Cycle bit 1 and bit 0 Capture mode: Unused. Compare mode: Unused. PWM mode: These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.
bit 3-0	**CCP1M<3:0>**: Enhanced CCP Mode Select bits 0000 = Capture/Compare/PWM off (resets ECCP module) 0001 = Reserved 0010 = Compare mode, toggle output on match 0011 = Reserved 0100 = Capture mode, every falling edge 0101 = Capture mode, every rising edge 0110 = Capture mode, every 4th rising edge 0111 = Capture mode, every 16th rising edge 1000 = Compare mode, initialize CCP1 pin low, set output on compare match (set CCP1IF) 1001 = Compare mode, initialize CCP1 pin high, clear output on compare match (set CCP1IF) 1010 = Compare mode, generate software interrupt only, CCP1 pin reverts to I/O state 1011 = Compare mode, trigger special event (ECCP resets TMR1 or TMR3, start A/D conversion, sets CC1IF bit) 1100 = PWM mode; P1A, P1C active-high; P1B, P1D active-high 1101 = PWM mode; P1A, P1C active-high; P1B, P1D active-low 1110 = PWM mode; P1A, P1C active-low; P1B, P1D active-high 1111 = PWM mode; P1A, P1C active-low; P1B, P1D active-low

FIGURE 13-16: SIMPLIFIED STEERING BLOCK DIAGRAM



Note 1: Port outputs are configured as shown when the CCP1CON register bits P1M<1:0> = `00` and CCP1M<3:2> = 11.

2: Single PWM output requires setting at least one of the STRx bits.

14.2.8 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in Full Power mode; in the case of the Sleep mode, all clocks are halted.

In all Idle modes, a clock is provided to the peripherals. That clock could be from the primary clock source, the secondary clock (Timer1 oscillator at 32.768 kHz) or the INTOSC source. See **Section 18.0 "Power-Managed Modes"** for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

When MSSP interrupts are enabled, after the master completes sending data, an MSSP interrupt will wake the controller:

- From Sleep, in Slave mode
- From Idle, in Slave or Master mode

If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in any Power-Managed mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

14.2.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

14.2.10 BUS MODE COMPATIBILITY

Table 14-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

TABLE 14-1: SPI BUS MODES

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

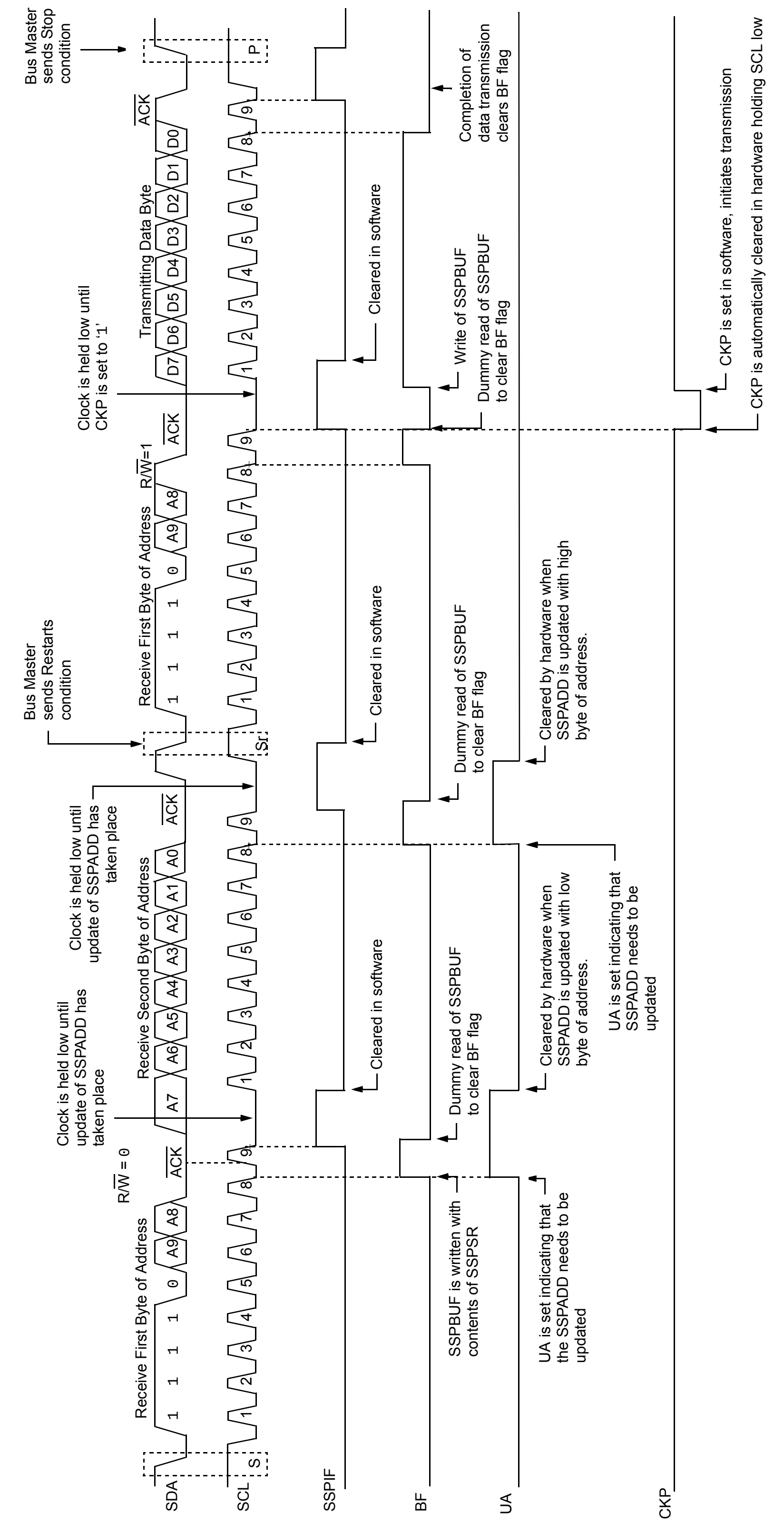
There is also an SMP bit which controls when the data is sampled.

TABLE 14-2: REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	245
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	248
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	248
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	248
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	248
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	248
SSPBUF	SSP Receive Buffer/Transmit Register								246
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	246
SSPSTAT	SMP	CKE	$D/\overline{A}$	P	S	$R/\overline{W}$	UA	BF	246

Legend: Shaded cells are not used by the MSSP in SPI mode.

FIGURE 14-11: I²C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)


SDA
SCL
SSPIF
BF
UA
CKP
S
Sr
P
Receive First Byte of Address
1 1 1 1 0 A9 A8
R/W = 0
ACK
Receive Second Byte of Address
A7 A6 A5 A4 A3 A2 A1 A0
R/W=1
Transmitting Data Byte
D7 D6 D5 D4 D3 D2 D1 D0
1 2 3 4 5 6 7 8 9
Clock is held low until update of SSPADD has taken place
Bus Master sends Restarts condition
Clock is held low until CKP is set to '1'
Bus Master sends Stop condition
Cleared in software
SSPBUF is written with contents of SSPSR
Dummy read of SSPBUF to clear BF flag
Write of SSPBUF
Completion of data transmission clears BF flag
UA is set indicating that the SSPADD needs to be updated
Cleared by hardware when SSPADD is updated with low byte of address.
UA is set indicating that SSPADD needs to be updated
Cleared by hardware when SSPADD is updated with high byte of address.
CKP is set in software, initiates transmission
CKP is automatically cleared in hardware holding SCL low

14.3.4 CLOCK STRETCHING

Both 7-bit and 10-bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit of the SSPCON2 register allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

14.3.4.1 Clock Stretching for 7-bit Slave Receive Mode (SEN = 1)

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{\text{ACK}}$ sequence if the BF bit is set, the CKP bit of the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another data transfer sequence. This will prevent buffer overruns from occurring (see Figure 14-13).

Note 1: If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set by software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

14.3.4.2 Clock Stretching for 10-bit Slave Receive Mode (SEN = 1)

In 10-bit Slave Receive mode during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{\text{W}}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

14.3.4.3 Clock Stretching for 7-bit Slave Transmit Mode

7-bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another data transfer sequence (see Figure 14-9).

Note 1: If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set by software regardless of the state of the BF bit.

14.3.4.4 Clock Stretching for 10-bit Slave Transmit Mode

In 10-bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high-order bits of the 10-bit address and the R/$\overline{\text{W}}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is automatic with the hardware clearing CKP, as in 7-bit Slave Transmit mode (see Figure 14-11).

14.3.10 I^{2}C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification parameter SP106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification parameter SP107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{\text{ACK}}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{\text{ACK}}$ is written into the ACKDT bit on the falling edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 14-21).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{\text{W}}$ bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{\text{ACK}}$ bit is loaded into the ACKSTAT Status bit of the SSPCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

14.3.10.1 BF Status Flag

In Transmit mode, the BF bit of the SSPSTAT register is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

14.3.10.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

WCOL must be cleared by software before the next transmission.

14.3.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPCON2 register is cleared when the slave has sent an Acknowledge ($\overline{\text{ACK}}$ = `0`) and is set when the slave does not Acknowledge ($\overline{\text{ACK}}$ = `1`). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

14.3.11 I^{2}C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN bit of the SSPCON2 register.

Note:	The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPCON2 register.

14.3.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

14.3.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

14.3.11.3 WCOL Status Flag

If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 14-27: BUS COLLISION DURING START CONDITION (SCL = 0)

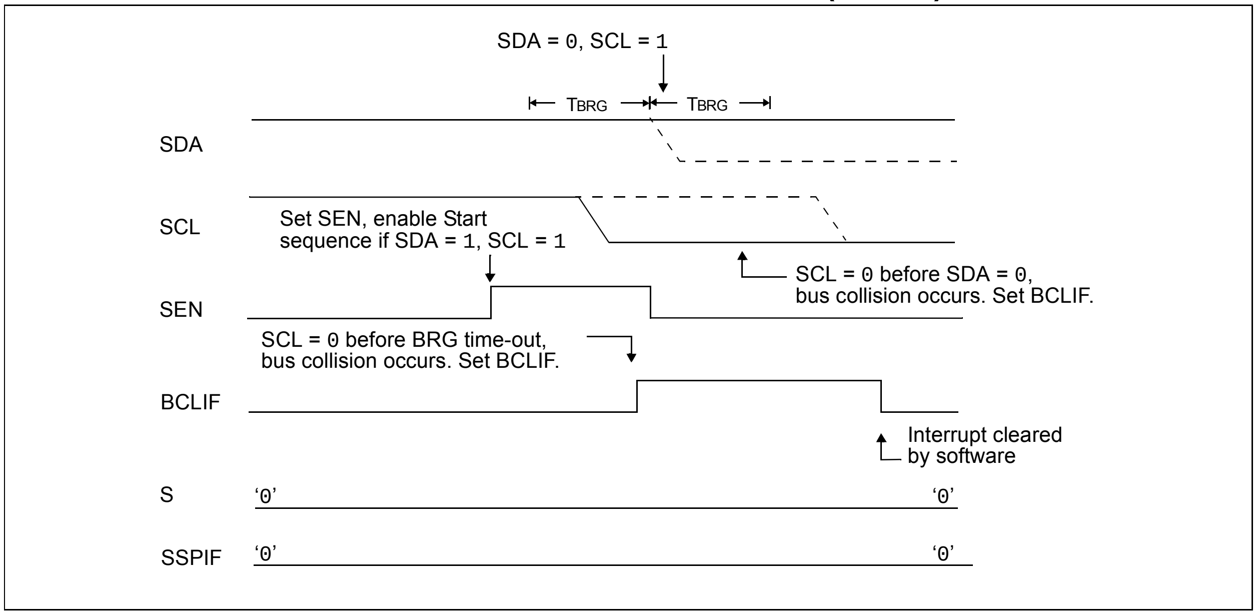
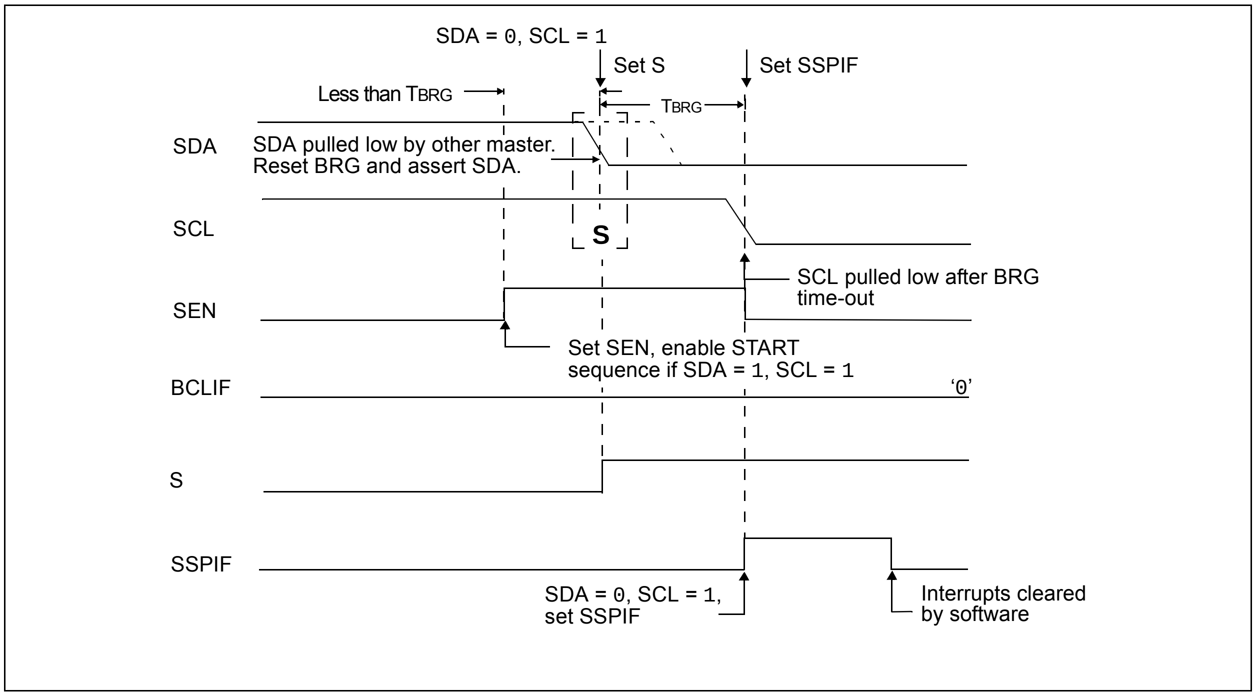

SDA = 0, SCL = 1
TBRG
TBRG
SDA
SCL
Set SEN, enable Start sequence if SDA = 1, SCL = 1
SCL = 0 before SDA = 0, bus collision occurs. Set BCLIF.
SEN
SCL = 0 before BRG time-out, bus collision occurs. Set BCLIF.
BCLIF
Interrupt cleared by software
S
'0'
'0'
SSPIF
'0'
'0'

FIGURE 14-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION


SDA = 0, SCL = 1
Set S
Set SSPIF
Less than TBRG
TBRG
SDA
SDA pulled low by other master. Reset BRG and assert SDA.
SCL
S
SCL pulled low after BRG time-out
SEN
Set SEN, enable START sequence if SDA = 1, SCL = 1
BCLIF
'0'
S
SSPIF
SDA = 0, SCL = 1, set SSPIF
Interrupts cleared by software

14.3.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 14-31). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 14-32).

FIGURE 14-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)

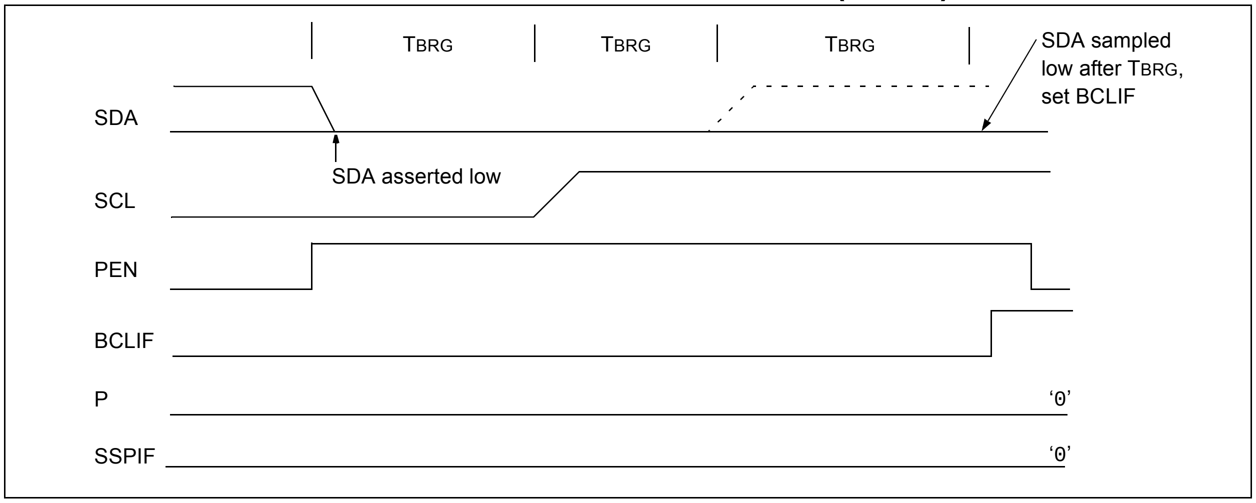
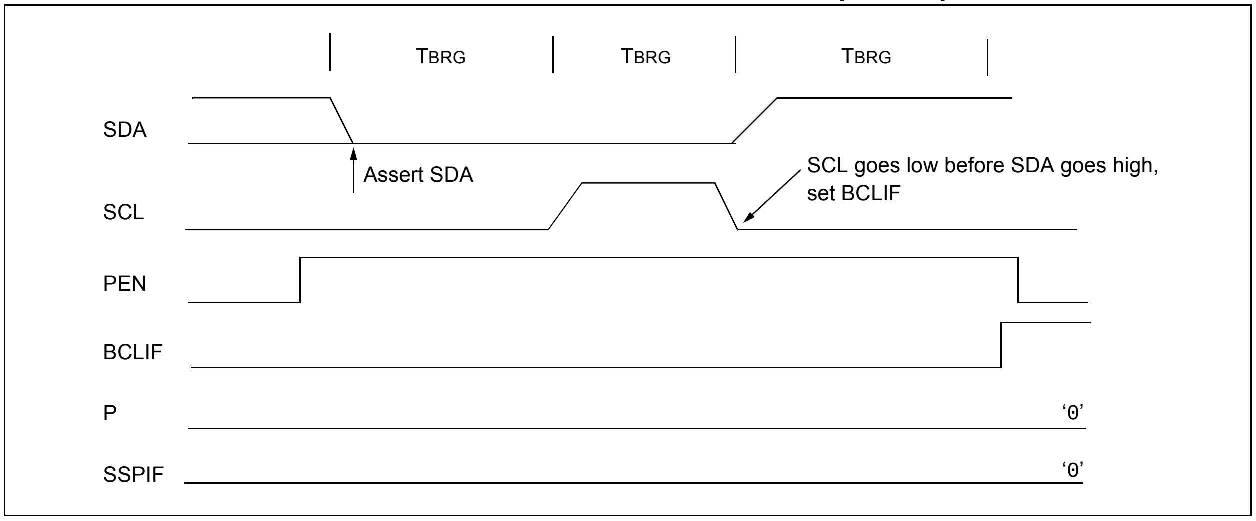


FIGURE 14-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)



16.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESL and ADRESH).

The ADC voltage reference is software selectable to either VDD, or a voltage applied to the external reference pins.

The ADC can generate an interrupt upon completion of a conversion. This interrupt can be used to wake-up the device from Sleep.

Figure 16-1 shows the block diagram of the ADC.

FIGURE 16-1: ADC BLOCK DIAGRAM

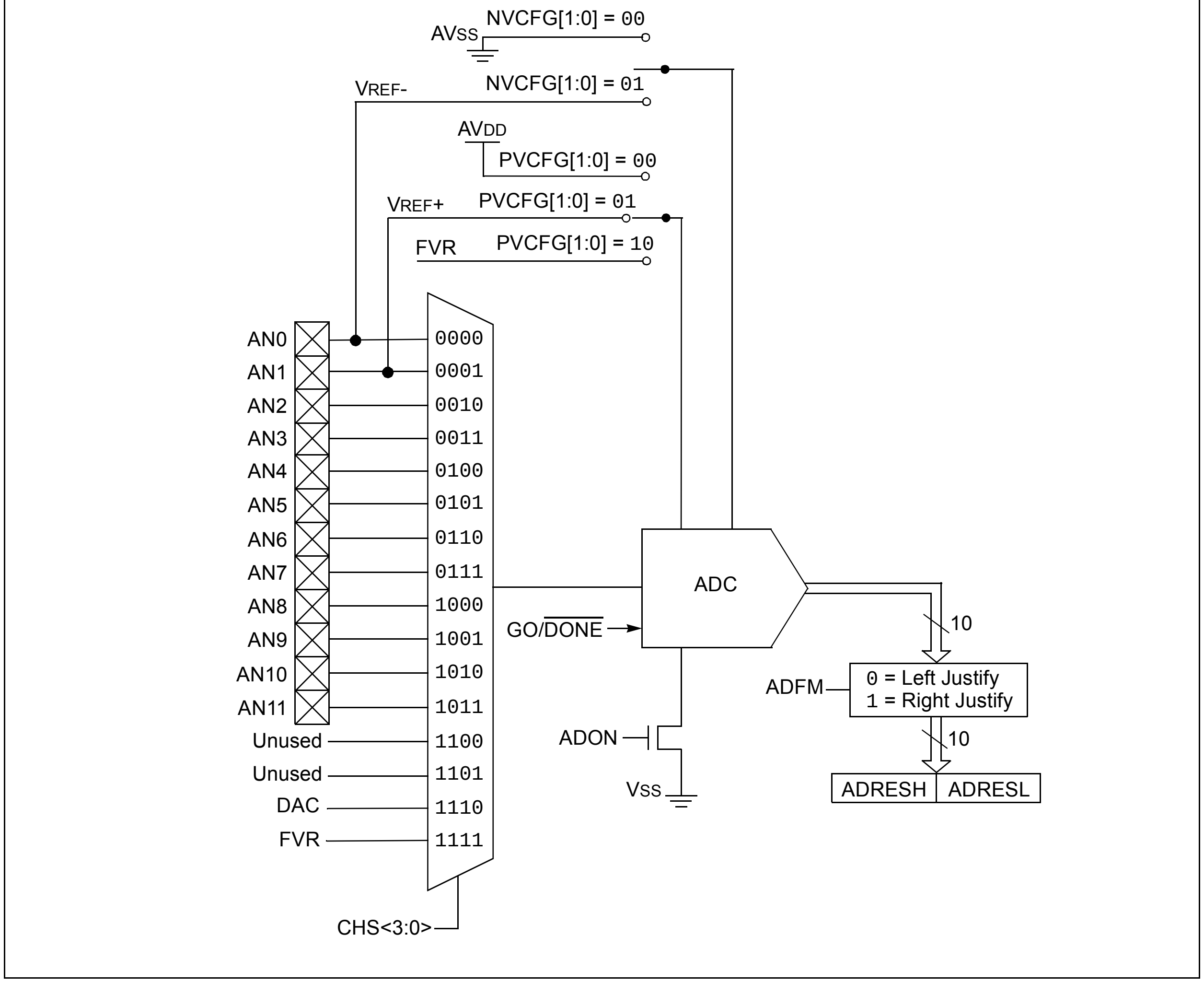


TABLE 17-2: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	248
CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH1	C1CH0	248
CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH1	C2CH0	248
CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	248
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	245
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	—	TMR3IP	—	248
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	248
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	—	TMR3IE	—	248
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	—	TMR3IF	—	248
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	248
VREFCON0	FVR1EN	FVR1ST	FVR1S<1:0>		—	—	—	—	247
VREFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS<1:0>		—	D1NSS	247
TRISA	—	—	TRISA5	TRISA4	—[(1)]	TRISA2	TRISA1	TRISA0	248
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	248

Legend: — = unimplemented, read as ‘0’. Shaded cells are unused by the comparator module.

Note 1: Unimplemented, read as ‘1’.

23.2 Watchdog Timer (WDT)

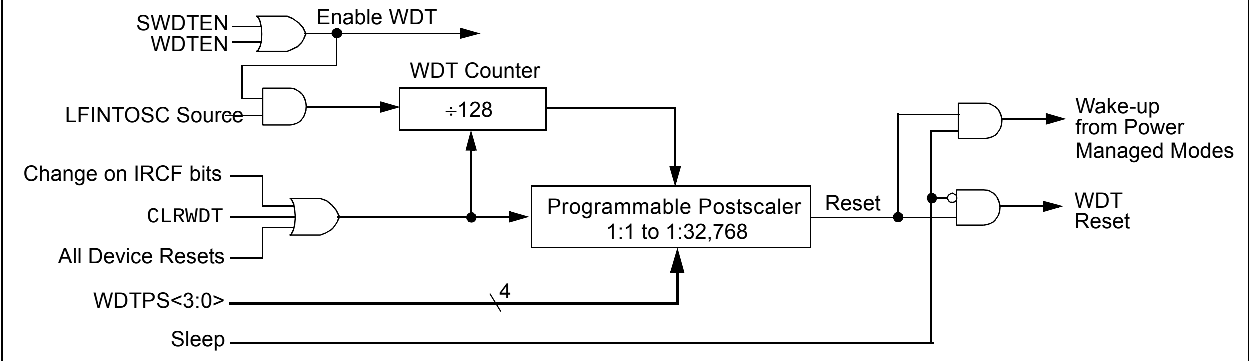
For PIC18(L)F1XK22 devices, the WDT is driven by the LFINTOSC source. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the LFINTOSC oscillator.

The 4-millisecond period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by bits in Configuration register 2H. Available periods range from 4 ms to 131.072 seconds (2.18 minutes). The WDT and postscaler are cleared when any of the following events occur: a `SLEEP` or `CLRWDT` instruction is executed, the IRCF bits of the OSCCON register are changed or a clock failure has occurred.

Note 1: The `CLRWDT` and `SLEEP` instructions clear the WDT and postscaler counts when executed.

2: Changing the setting of the IRCF bits of the OSCCON register clears the WDT and postscaler counts.

FIGURE 23-1: WDT BLOCK DIAGRAM



ADDWFC — ADD W and CARRY bit to f

Syntax:	ADDWFC f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(W) + (f) + (C) → dest
Status Affected:	N,OV, C, DC, Z
Encoding:	`0010` `00da` `ffff` `ffff`
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'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWFC    REG, 0, 1`

Before Instruction

CARRY bit	=	1
REG	=	02h
W	=	4Dh

After Instruction

CARRY bit	=	0
REG	=	02h
W	=	50h

ANDLW — AND literal with W

Syntax:	ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. k → W
Status Affected:	N, Z
Encoding:	`0000` `1011` `kkkk` `kkkk`
Description:	The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `ANDLW    05Fh`

Before Instruction

W = A3h

After Instruction

W = 03h

SUBWFB — Subtract W from f with Borrow

Syntax:	SUBWFB f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – (W) – ($\overline{C}$) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0101` `10da` `ffff` `ffff`
Description:	Subtract W and the CARRY flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: `SUBWFB REG, 1, 0`

```
Before Instruction
    REG   =  19h    (0001 1001)
    W     =  0Dh    (0000 1101)
    C     =  1
After Instruction
    REG   =  0Ch    (0000 1011)
    W     =  0Dh    (0000 1101)
    C     =  1
    Z     =  0
    N     =  0      ; result is positive
```

Example 2: `SUBWFB REG, 0, 0`

```
Before Instruction
    REG   =  1Bh    (0001 1011)
    W     =  1Ah    (0001 1010)
    C     =  0
After Instruction
    REG   =  1Bh    (0001 1011)
    W     =  00h
    C     =  1
    Z     =  1      ; result is zero
    N     =  0
```

Example 3: `SUBWFB REG, 1, 0`

```
Before Instruction
    REG   =  03h    (0000 0011)
    W     =  0Eh    (0000 1101)
    C     =  1
After Instruction
    REG   =  F5h    (1111 0100)
                    ; [2's comp]
    W     =  0Eh    (0000 1101)
    C     =  0
    Z     =  0
    N     =  1      ; result is negative
```

SWAPF — Swap f

Syntax:	SWAPF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>
Status Affected:	None
Encoding:	`0011` `10da` `ffff` `ffff`
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `SWAPF REG, 1, 0`

```
Before Instruction
    REG   =  53h
After Instruction
    REG   =  35h
```