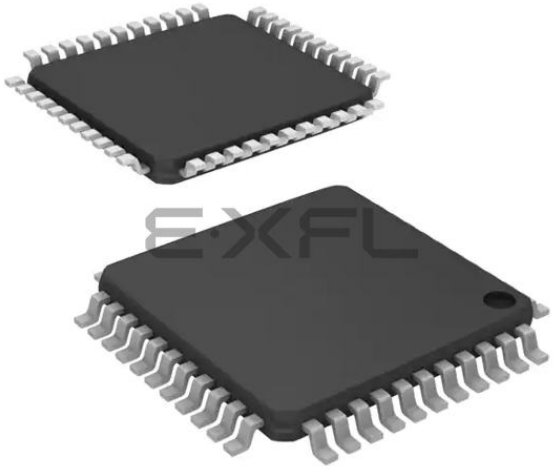


Welcome to E-XFL.COM

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	40MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf4580t-i-pt

3.4 RC Oscillator

For timing insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The actual oscillator frequency is a function of several factors:

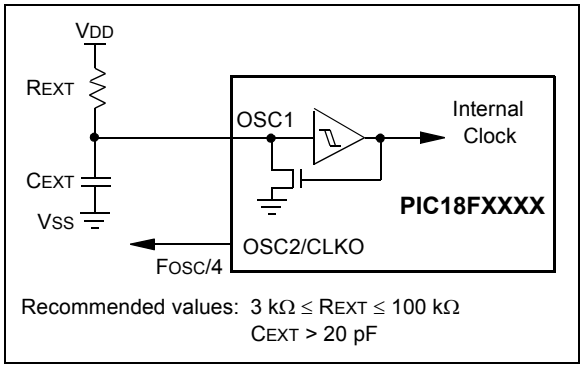
- supply voltage
- values of the external resistor (REXT) and capacitor (CEXT)
- operating temperature

Given the same device, operating voltage and temperature and component values, there will also be unit-to-unit frequency variations. These are due to factors such as:

- normal manufacturing variation
- difference in lead frame capacitance between package types (especially for low CEXT values)
- variations within the tolerance of limits of REXT and CEXT

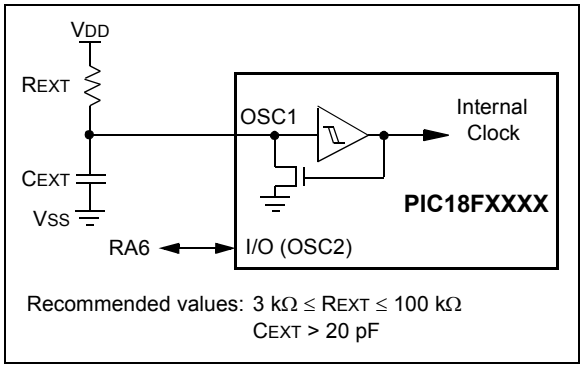
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-5 shows how the R/C combination is connected.

FIGURE 3-5: RC OSCILLATOR MODE



The RCIO Oscillator mode (Figure 3-6) functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

FIGURE 3-6: RCIO OSCILLATOR MODE



3.5 PLL Frequency Multiplier

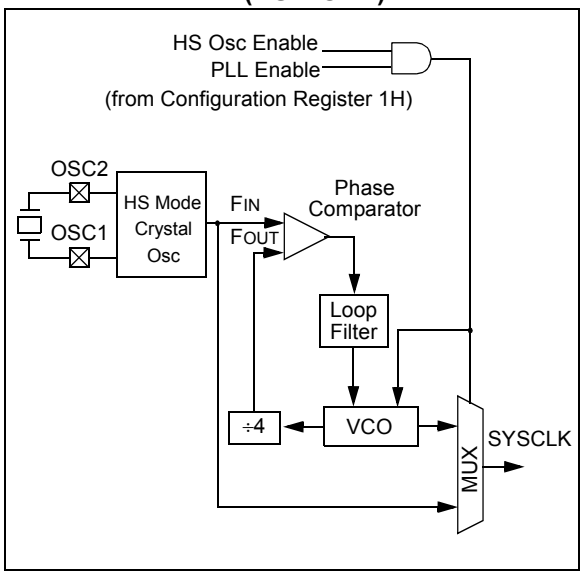
A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower frequency oscillator circuit or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals or users who require higher clock speeds from an internal oscillator.

3.5.1 HSPLL OSCILLATOR MODE

The HSPLL mode makes use of the HS mode oscillator for frequencies up to 10 MHz. A PLL then multiplies the oscillator output frequency by 4 to produce an internal clock frequency up to 40 MHz.

The PLL is only available to the crystal oscillator when the FOSC<3:0> Configuration bits are programmed for HSPLL mode (= 0110).

FIGURE 3-7: PLL BLOCK DIAGRAM (HS MODE)



3.5.2 PLL AND INTOSC

The PLL is also available to the internal oscillator block in selected oscillator modes. In this configuration, the PLL is enabled in software and generates a clock output of up to 32 MHz. The operation of INTOSC with the PLL is described in **Section 3.6.4 "PLL in INTOSC Modes"**.

3.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F2480/2580/4480/4580 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate low-frequency clock source. PIC18F2480/2580/4480/4580 devices offer two alternate clock sources. When an alternate clock source is enabled, the various power-managed operating modes are available.

Essentially, there are three clock sources for these devices:

- Primary oscillators
- Secondary oscillators
- Internal oscillator block

The **primary oscillators** include the external crystal and resonator modes, the external RC modes, the external clock modes and the internal oscillator block. The particular mode is defined by the FOSC<3:0> Configuration bits. The details of these modes are covered earlier in this chapter.

The **secondary oscillators** are those external sources not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power-managed mode.

PIC18F2480/2580/4480/4580 devices offer the Timer1 oscillator as a secondary oscillator. This oscillator, in all power-managed modes, is often the time base for functions such as a Real-Time Clock (RTC).

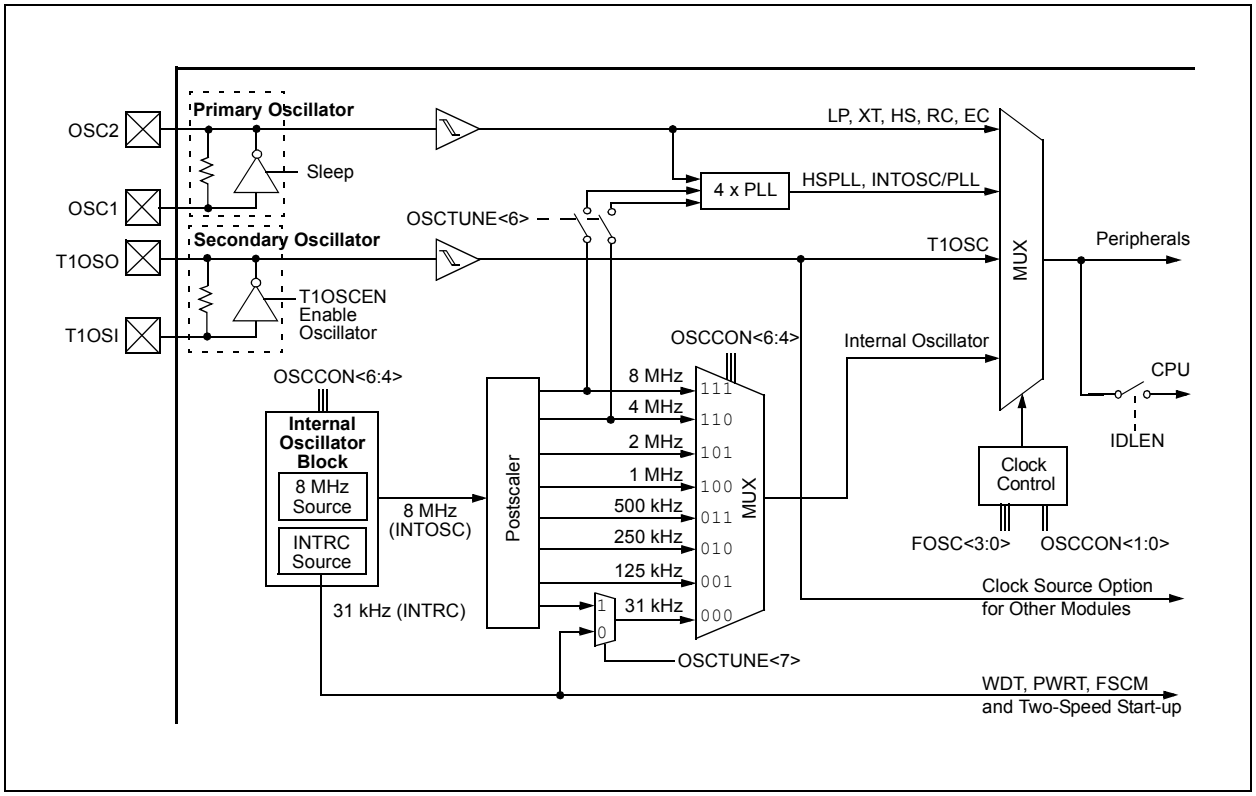
Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T13CKI and RC1/T1OSI pins. Like the LP Oscillator mode circuit, loading capacitors are also connected from each pin to ground.

The Timer1 oscillator is discussed in greater detail in **Section 13.3 "Timer1 Oscillator"**.

In addition to being a primary clock source, the **internal oscillator block** is available as a power-managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor.

The clock sources for the PIC18F2480/2580/4480/4580 devices are shown in Figure 3-8. See **Section 25.0 "Special Features of the CPU"** for Configuration register details.

FIGURE 3-8: PIC18F2480/2580/4480/4580 CLOCK DIAGRAM



4.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

Certain exits from power-managed modes do not invoke the OST at all. There are two cases:

- PRI_IDLE mode where the primary clock source is not stopped; and
- the primary clock source is not any of the LP, XT, HS or HSPLL modes.

In these instances, the primary clock source either does not require an oscillator start-up delay, since it is already running (PRI_IDLE), or normally does not require an oscillator start-up delay (RC, EC and INTIO Oscillator modes). However, a fixed delay of interval, TCSD, following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

TABLE 4-2: EXIT DELAY ON WAKE-UP BY RESET FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)

Clock Source Before Wake-up	Clock Source After Wake-up	Exit Delay	Clock Ready Status bit (OSCCON)
Primary Device Clock (PRI_IDLE mode)	LP, XT, HS	T_{CSD}(2)	OSTS
	HSPLL		
	EC, RC		
	INTRC(1)		—
	INTOSC(3)		IOFS
T1OSC or INTRC(1)	LP, XT, HS	T_{OST}(4)	OSTS
	HSPLL	$T_{OST} + t_{rc}$(4)	
	EC, RC	T_{CSD}(2)	
	INTRC(1)		—
	INTOSC(3)	T_{IOBST}(5)	IOFS
INTOSC(3)	LP, XT, HS	T_{OST}(5)	OSTS
	HSPLL	$T_{OST} + t_{rc}$(4)	
	EC, RC	T_{CSD}(2)	
	INTRC(1)		—
	INTOSC(3)	None	IOFS
None (Sleep mode)	LP, XT, HS	T_{OST}(4)	OSTS
	HSPLL	$T_{OST} + t_{rc}$(4)	
	EC, RC	T_{CSD}(2)	
	INTRC(1)		—
	INTOSC(3)	T_{IOBST}(5)	IOFS

Note 1: In this instance, refers specifically to the 31 kHz INTRC clock source.

2: TCSD (parameter 38) is a required delay when waking from Sleep and all Idle modes and runs concurrently with any other required delays (see **Section 4.4 "Idle Modes"**).

3: Includes both the INTOSC 8 MHz source and postscaler derived frequencies.

4: TOST is the Oscillator Start-up Timer (parameter 32). t_{rc} is the PLL Lock-out Timer (parameter F12); it is also designated as TPLL.

5: Execution continues during TIOBST (parameter 39), the INTOSC stabilization period.

5.4 Brown-out Reset (BOR)

PIC18F2480/2580/4480/4580 devices implement a BOR circuit that provides the user with a number of configuration and power-saving options. The BOR is controlled by the BORV<1:0> and BOREN<1:0> Configuration bits. There are a total of four BOR configurations which are summarized in Table 5-1.

The BOR threshold is set by the BORV<1:0> bits. If BOR is enabled (any values of BOREN<1:0>, except '00'), any drop of VDD below VBOR (parameter D005) for greater than TBOR (parameter 35) will reset the device. A Reset may or may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR.

If the Power-up Timer is enabled, it will be invoked after VDD rises above VBOR; it then will keep the chip in Reset for an additional time delay, TPWRT (parameter 33). If VDD drops below VBOR while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above VBOR, the Power-up Timer will execute the additional time delay.

BOR and the Power-on Timer (PWRT) are independently configured. Enabling a Brown-out Reset does not automatically enable the PWRT.

5.4.1 SOFTWARE ENABLED BOR

When BOREN<1:0> = 01, the BOR can be enabled or disabled by the user in software. This is done with the control bit, SBOREN (RCON<6>). Setting SBOREN enables the BOR to function as previously described. Clearing SBOREN disables the BOR entirely. The SBOREN bit operates only in this mode; otherwise it is read as '0'.

Placing the BOR under software control gives the user the additional flexibility of tailoring the application to its environment without having to reprogram the device to change BOR configuration. It also allows the user to tailor device power consumption in software by eliminating the incremental current that the BOR consumes. While the BOR current is typically very small, it may have some impact in low-power applications.

Note:	Even when BOR is under software control, the Brown-out Reset voltage level is still set by the BORV<1:0> Configuration bits. It cannot be changed in software.

5.4.2 DETECTING BOR

When Brown-out Reset is enabled, the $\overline{\text{BOR}}$ bit always resets to '0' on any Brown-out Reset or Power-on Reset event. This makes it difficult to determine if a Brown-out Reset event has occurred just by reading the state of $\overline{\text{BOR}}$ alone. A more reliable method is to simultaneously check the state of both $\overline{\text{POR}}$ and $\overline{\text{BOR}}$. This assumes that the $\overline{\text{POR}}$ bit is reset to '1' in software immediately after any Power-on Reset event. IF $\overline{\text{BOR}}$ is '0' while $\overline{\text{POR}}$ is '1', it can be reliably assumed that a Brown-out Reset event has occurred.

5.4.3 DISABLING BOR IN SLEEP MODE

When BOREN<1:0> = 10, the BOR remains under hardware control and operates as previously described. Whenever the device enters Sleep mode, however, the BOR is automatically disabled. When the device returns to any other operating mode, BOR is automatically re-enabled.

This mode allows for applications to recover from brown-out situations, while actively executing code, when the device requires BOR protection the most. At the same time, it saves additional power in Sleep mode by eliminating the small incremental BOR current.

TABLE 5-1: BOR CONFIGURATIONS

BOR Configuration		Status of SBOREN (RCON<6>)	BOR Operation
BOREN1	BOREN0		
0	0	Unavailable	BOR disabled; must be enabled by reprogramming the Configuration bits.
0	1	Available	BOR enabled in software; operation controlled by SBOREN.
1	0	Unavailable	BOR enabled in hardware in Run and Idle modes, disabled during Sleep mode.
1	1	Unavailable	BOR enabled in hardware; must be disabled by reprogramming the Configuration bits.

TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
RXF15EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	65, 309
RXF15SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	65, 308
RXF15SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	65, 309
RXF14EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	65, 309
RXF14EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	65, 309
RXF14SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	65, 308
RXF14SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	65, 309
RXF13EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF13EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF13SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF13SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309
RXF12EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF12EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF12SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF12SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309
RXF11EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF11EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF11SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF11SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309
RXF10EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF10EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF10SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF10SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309
RXF9EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF9EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF9SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF9SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309
RXF8EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF8EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF8SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF8SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309
RXF7EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF7EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF7SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF7SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309
RXF6EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	66, 309
RXF6EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	66, 309
RXF6SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	66, 308
RXF6SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	66, 309

Legend: x = unknown, u = unchanged, – = unimplemented, q = value depends on condition

Note 1: Bit 21 of the PC is only available in Test mode and Serial Programming modes.

2: The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

3: These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

4: The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

5: The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

6: RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

7: CAN bits have multiple functions depending on the selected mode of the CAN module.

8: This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

9: These registers are available on PIC18F4X80 devices only.

FIGURE 11-3: PARALLEL SLAVE PORT WRITE WAVEFORMS

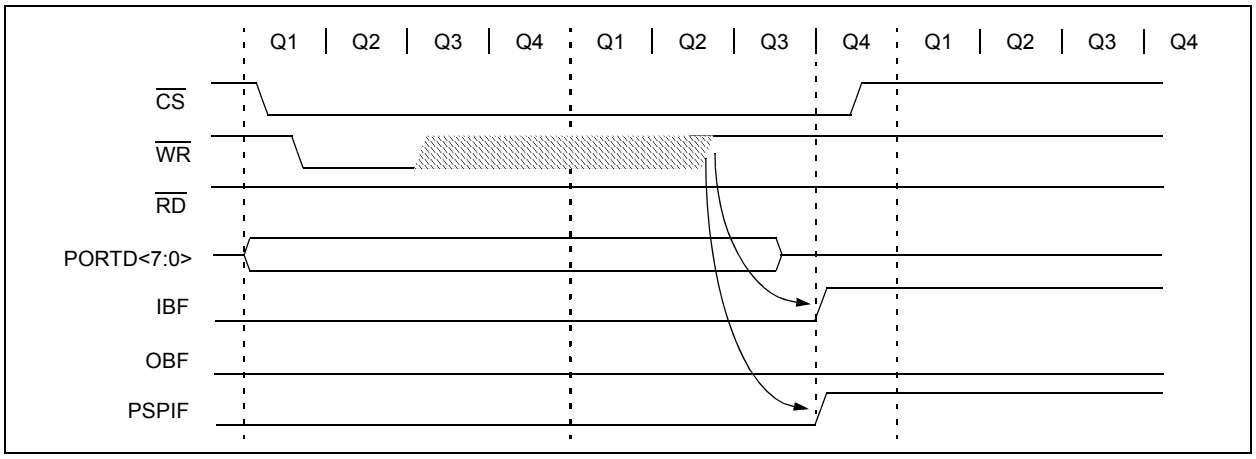


FIGURE 11-4: PARALLEL SLAVE PORT READ WAVEFORMS

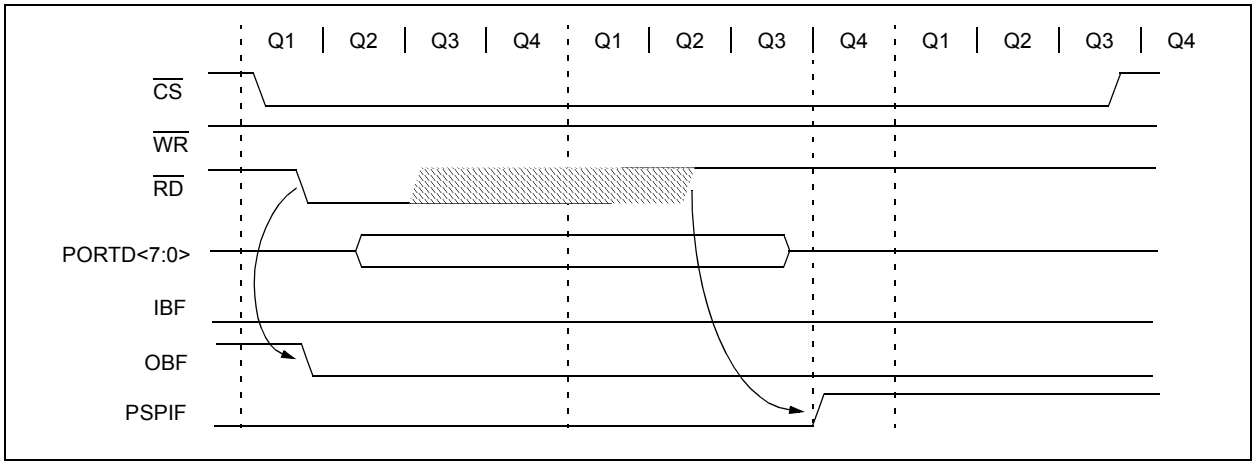


TABLE 11-11: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
PORTD(1)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	58
LATD(1)	LATD Output Latch Register								58
TRISD(1)	PORTD Data Direction Register								58
PORTE(1)	—	—	—	—	RE3	RE2	RE1	RE0	58
LATE(1)	—	—	—	—	—	LATE Output Latch Register			58
TRISE(1)	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	58
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	55
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	58
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	58
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	58
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	56
CMCON(1)	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	57

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.

Note 1: These registers are available on PIC18F4X80 devices only.

14.0 TIMER2 MODULE

The Timer2 module timer incorporates the following features:

- 8-Bit Timer and Period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscaler (1:1 through 1:16)
- Interrupt on TMR2 to PR2 match
- Optional use as the shift clock for the MSSP module

The module is controlled through the T2CON register (Register 14-1), which enables or disables the timer and configures the prescaler and postscaler. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in Figure 14-1.

14.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 2-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options; these are selected by the prescaler control bits, T2CKPS<1:0> (T2CON<1:0>). The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscaler (see **Section 14.2 "Timer2 Interrupt"**).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

REGISTER 14-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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 **Unimplemented:** Read as '0'

bit 6-3 **T2OUTPS<3:0>**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS<1:0>**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

REGISTER 18-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV(1)	SSPEN(2)	CKP	SSPM3(3)	SSPM2(3)	SSPM1(3)	SSPM0(3)
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 **WCOL:** Write Collision Detect bit (Transmit mode only)
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)
SPI Slave mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(2)
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins(2)
0 = Disables serial port and configures these pins as I/O port pins(2)

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits(3)
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0011 = SPI Master mode, clock = TMR2 output/2
0010 = SPI Master mode, clock = FOSC/64
0001 = SPI Master mode, clock = FOSC/16
0000 = SPI Master mode, clock = FOSC/4

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
2: When enabled, these pins must be properly configured as input or output.
3: Bit combinations not specifically listed here are either reserved or implemented in I^2C™ mode only.

18.4.2 OPERATION

The MSSP module functions are enabled by setting the MSSP Enable bit, SSPEN (SSPCON<5>).

The SSPCON1 register allows control of the I^2C operation. Four mode selection bits (SSPCON<3:0>) allow one of the following I^2C modes to be selected:

- I^2C Master mode, clock = (FOSC/4) x (SSPADD + 1)
- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I^2C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I^2C Firmware Controlled Master mode, slave is Idle

Selection of any I^2C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRISC bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

18.4.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The I^2C Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{ACK}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{ACK}$ pulse:

- The Buffer Full bit, BF (SSPSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPCON<6>), was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit, SSPIF (PIR1<3>), is set. The BF bit is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I^2C specification, as well as the requirement of the MSSP module, are shown in timing parameter 100 and parameter 101.

18.4.3.1 Addressing

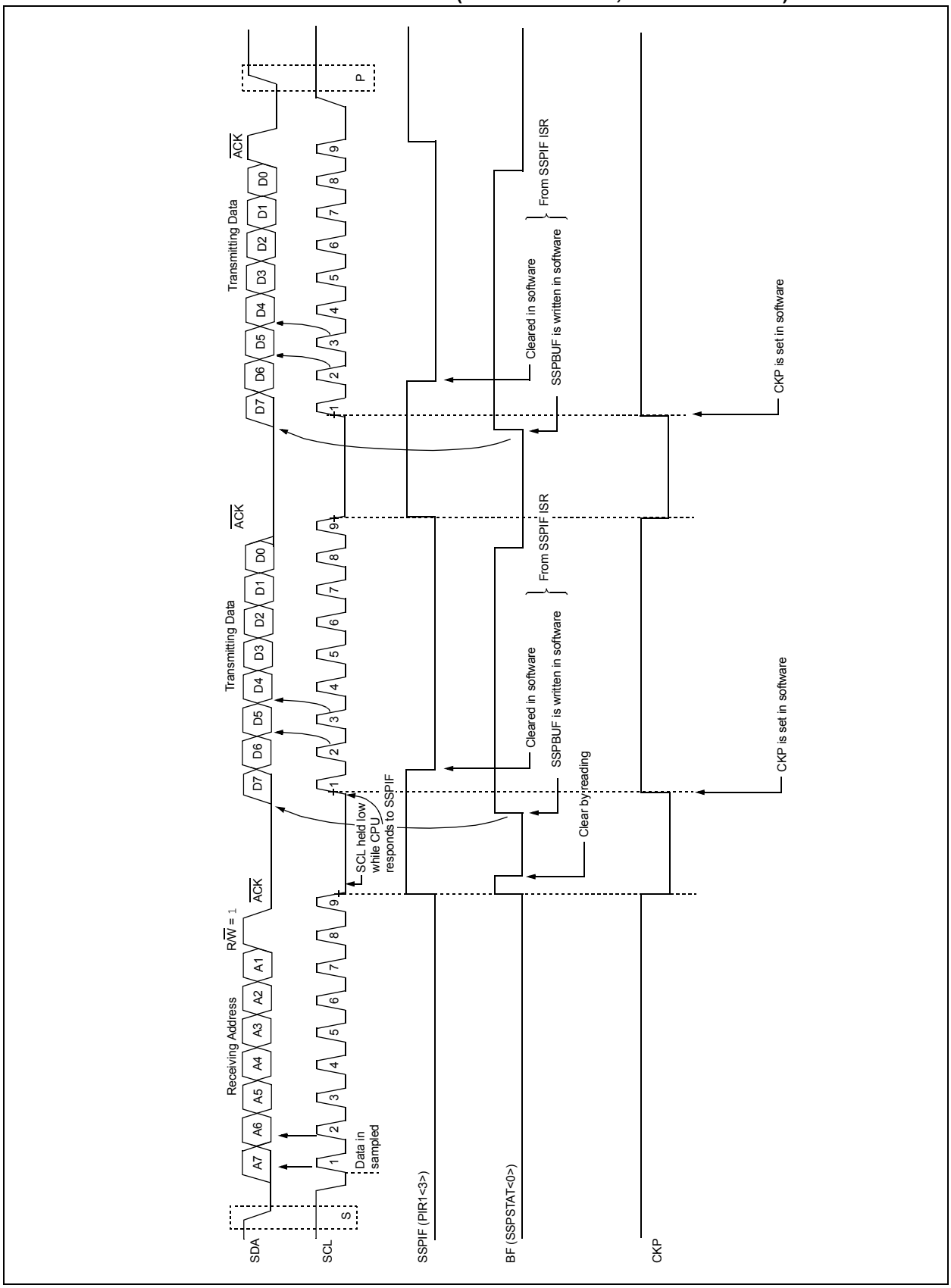
Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8-bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register, SSPSR<7:1>, is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit, BF, is set.
3. An $\overline{ACK}$ pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF (PIR1<3>), is set (interrupt is generated, if enabled) on the falling edge of the ninth SCL pulse.

In 10-Bit Addressing mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit, R/$\overline{W}$ (SSPSTAT<2>), must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '11110 A9 A8 0', where 'A9' and 'A8' are the two MSbs of the address. The sequence of events for 10-bit addressing is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits, SSPIF, BF and UA (SSPSTAT<1>), are set).
2. Update the SSPADD register with second (low) byte of address (clears bit, UA, and releases the SCL line).
3. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.
4. Receive second (low) byte of address (bits, SSPIF, BF and UA, are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit, UA.
6. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits, SSPIF and BF, are set).
9. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.

FIGURE 18-9: I²C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)



18.4.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from a low level to a high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', see Figure 18-29). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high-to-low before the BRG times out, and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 18-30.

If, at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

FIGURE 18-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

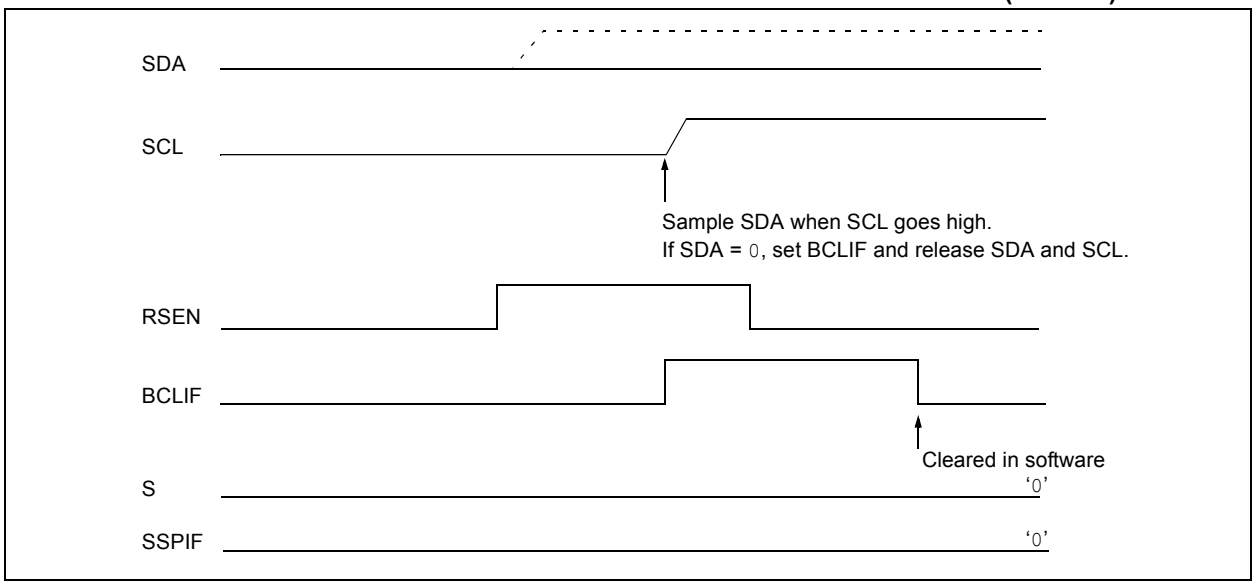
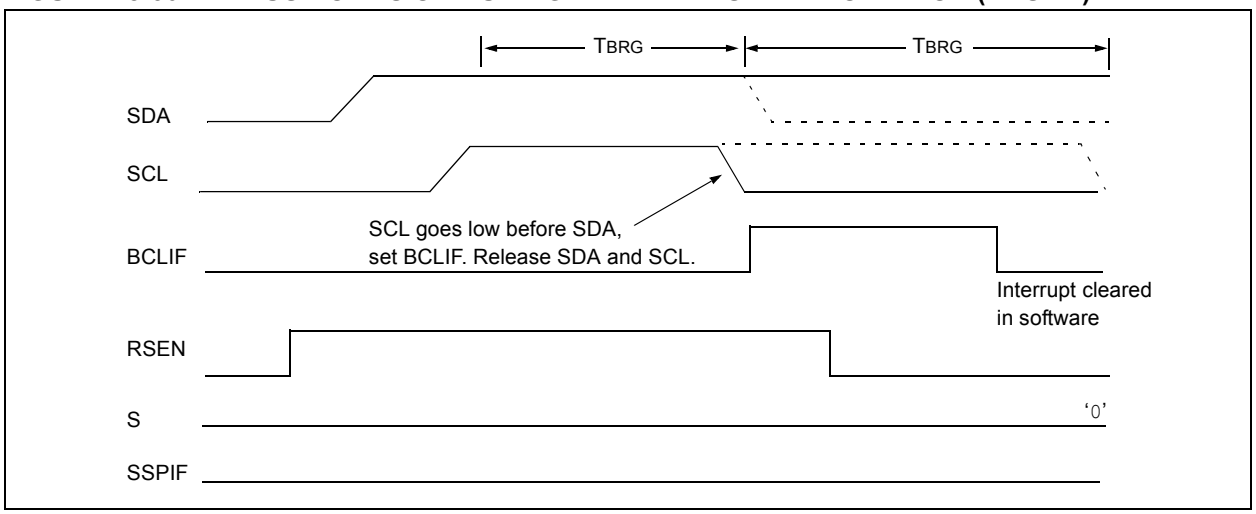


FIGURE 18-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



REGISTER 24-9: TXBnEIDL: TRANSMIT BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
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 7-0 **EID<7:0>:** Extended Identifier bits (not used when transmitting standard identifier message)

REGISTER 24-10: TXBnDm: TRANSMIT BUFFER n DATA FIELD BYTE m REGISTERS [0 ≤ n ≤ 2, 0 ≤ m ≤ 7]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
TXBnDm7	TXBnDm6	TXBnDm5	TXBnDm4	TXBnDm3	TXBnDm2	TXBnDm1	TXBnDm0
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 7-0 **TXBnDm<7:0>:** Transmit Buffer n Data Field Byte m bits (where 0 ≤ n < 3 and 0 ≤ m < 8)

Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

EXAMPLE 24-3: TRANSMITTING A CAN MESSAGE USING BANKED METHOD

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. And since we want banked method, we need to make sure
; that correct bank is selected.
BANKSEL  TXB0CON                    ; One BANKSEL in beginning will make sure that we are
                                    ; in correct bank for rest of the buffer access.
; Now load transmit data into TXB0 buffer.
MOVLW    MY_DATA_BYTE1              ; Load first data byte into buffer
MOVWF    TXB0D0                     ; Compiler will automatically set "BANKED" bit
; Load rest of data bytes - up to 8 bytes into TXB0 buffer.
...
; Load message identifier
MOVLW    60H                        ; Load SID2:SID0, EXIDE = 0
MOVWF    TXB0SIDL
MOVLW    24H                        ; Load SID10:SID3
MOVWF    TXB0SIDH
; No need to load TXB0EIDL:TXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW    B'00001000'                ; Normal priority; Request transmission
MOVWF    TXB0CON

; If required, wait for message to get transmitted
BTFSC    TXB0CON, TXREQ             ; Is it transmitted?
BRA      $-2                        ; No. Continue to wait...

; Message is transmitted.
```

EXAMPLE 24-4: TRANSMITTING A CAN MESSAGE USING WIN BITS

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. Use WIN bits to map it to RXB0 area.
MOVF    CANCON, W                   ; WIN bits are in lower 4 bits only. Read CANCON
                                    ; register to preserve all other bits. If operation
                                    ; mode is already known, there is no need to preserve
                                    ; other bits.
ANDLW   B'11110000'                 ; Clear WIN bits.
IORLW   B'00001000'                 ; Select Transmit Buffer 0
MOVWF   CANCON                      ; Apply the changes.
; Now TXB0 is mapped in place of RXB0. All future access to RXB0 registers will actually
; yield TXB0 register values.

; Load transmit data into TXB0 buffer.
MOVLW   MY_DATA_BYTE1               ; Load first data byte into buffer
MOVWF   RXB0D0                      ; Access TXB0D0 via RXB0D0 address.
; Load rest of the data bytes - up to 8 bytes into "TXB0" buffer using RXB0 registers.
...
; Load message identifier
MOVLW   60H                         ; Load SID2:SID0, EXIDE = 0
MOVWF   RXB0SIDL
MOVLW   24H                         ; Load SID10:SID3
MOVWF   RXB0SIDH
; No need to load RXB0EIDL:RXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW   B'00001000'                 ; Normal priority; Request transmission
MOVWF   RXB0CON

; If required, wait for message to get transmitted
BTFSC   RXB0CON, TXREQ              ; Is it transmitted?
BRA     $-2                         ; No. Continue to wait...

; Message is transmitted.
; If required, reset the WIN bits to default state.
```

REGISTER 24-16: RXBnSIDL: RECEIVE BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXID	—	EID17	EID16
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 7-5 **SID<2:0>:** Standard Identifier bits (if EXID = 0)
Extended Identifier bits, EID<20:18> (if EXID = 1).

bit 4 **SRR:** Substitute Remote Request bit
This bit is always '1' when EXID = 1 or equal to the value of RXRTRRO (RBXnCON<3>) when EXID = 0.

bit 3 **EXID:** Extended Identifier bit
1 = Received message is an extended data frame, SID<10:0> are EID<28:18 >
0 = Received message is a standard data frame

bit 2 **Unimplemented:** Read as '0'

bit 1-0 **EID<17:16>:** Extended Identifier bits

REGISTER 24-17: RXBnEIDH: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
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 7-0 **EID<15:8>:** Extended Identifier bits

REGISTER 24-18: RXBnEIDL: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
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 7-0 **EID<7:0>:** Extended Identifier bits

TABLE 24-1: CAN CONTROLLER REGISTER MAP (CONTINUED)

Address[1]	Name
D7Fh	—[4]
D7Eh	—[4]
D7Dh	—[4]
D7Ch	—[4]
D7Bh	RXF11EIDL
D7Ah	RXF11EIDH
D79h	RXF11SIDL
D78h	RXF11SIDH
D77h	RXF10EIDL
D76h	RXF10EIDH
D75h	RXF10SIDL
D74h	RXF10SIDH
D73h	RXF9EIDL
D72h	RXF9EIDH
D71h	RXF9SIDL
D70h	RXF9SIDH
D6Fh	—[4]
D6Eh	—[4]
D6Dh	—[4]
D6Ch	—[4]
D6Bh	RXF8EIDL
D6Ah	RXF8EIDH
D69h	RXF8SIDL
D68h	RXF8SIDH
D67h	RXF7EIDL
D66h	RXF7EIDH
D65h	RXF7SIDL
D64h	RXF7SIDH
D63h	RXF6EIDL
D62h	RXF6EIDH
D61h	RXF6SIDL
D60h	RXF6SIDH

Note 1: Shaded registers are available in Access Bank low area while the rest are available in Bank 15.

2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

3: These registers are not CAN registers.

4: Unimplemented registers are read as '0'.

24.3 CAN Modes of Operation

The PIC18F2480/2580/4480/4580 has six main modes of operation:

- Configuration mode
- Disable/Sleep mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode

All modes, except Error Recognition, are requested by setting the REQOP bits (CANCON<7:5>). Error Recognition mode is requested through the RXM bits of the Receive Buffer register(s). Entry into a mode is Acknowledged by monitoring the OPMODE bits.

When changing modes, the mode will not actually change until all pending message transmissions are complete. Because of this, the user must verify that the device has actually changed into the requested mode before further operations are executed.

24.3.1 CONFIGURATION MODE

The CAN module has to be initialized before the activation. This is only possible if the module is in the Configuration mode. The Configuration mode is requested by setting the REQOP2 bit. Only when the status bit, OPMODE2, has a high level can the initialization be performed. Afterwards, the Configuration registers, the acceptance mask registers and the acceptance filter registers can be written. The module is activated by setting the REQOP control bits to zero.

The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is online. The CAN module will not be allowed to enter the Configuration mode while a transmission or reception is taking place. The Configuration mode serves as a lock to protect the following registers:

- Configuration Registers
- Functional Mode Selection Registers
- Bit Timing Registers
- Identifier Acceptance Filter Registers
- Identifier Acceptance Mask Registers
- Filter and Mask Control Registers
- Mask Selection Registers

In the Configuration mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to Configuration registers that are access restricted in other modes. I/O pins will revert to normal I/O functions.

24.3.2 DISABLE/SLEEP MODE

In Disable/Sleep mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity; however, any pending interrupts will remain and the error counters will retain their value.

If the REQOP<2:0> bits are set to '001', the module will enter the module Disable/Sleep mode. This mode is similar to disabling other peripheral modules by turning off the module enables. This causes the module internal clock to stop unless the module is active (i.e., receiving or transmitting a message). If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an Idle bus, then accept the module Disable/Sleep command. OPMODE<2:0> = 001 indicates whether the module successfully went into the module Disable/Sleep mode.

The WAKIF interrupt is the only module interrupt that is still active in the Disable/Sleep mode. If the WAKDIS is cleared and WAKIE is set, the processor will receive an interrupt whenever the module detects recessive to dominant transition. On wake-up, the module will automatically be set to the previous mode of operation. For example, if the module was switched from Normal to Disable/Sleep mode on bus activity wake-up, the module will automatically enter into Normal mode and the first message that caused the module to wake-up is lost. The module will not generate any error frame. Firmware logic must detect this condition and make sure that retransmission is requested. If the processor receives a wake-up interrupt while it is sleeping, more than one message may get lost. The actual number of messages lost would depend on the processor oscillator start-up time and incoming message bit rate.

The TXCAN pin will stay in the recessive state while the module is in Disable/Sleep mode.

24.3.3 NORMAL MODE

This is the standard operating mode of the PIC18F2480/2580/4480/4580 devices. In this mode, the device actively monitors all bus messages and generates Acknowledge bits, error frames, etc. This is also the only mode in which the PIC18F2480/2580/4480/4580 devices will transmit messages over the CAN bus.

24.4.3 MODE 2 – ENHANCED FIFO MODE

In Mode 2, two or more receive buffers are used to form the receive FIFO (first in, first out) buffer. There is no one-to-one relationship between the receive buffer and acceptance filter registers. Any filter that is enabled and linked to any FIFO receive buffer can generate acceptance and cause FIFO to be updated.

FIFO length is user-programmable, from 2-8 buffers deep. FIFO length is determined by the very first programmable buffer that is configured as a transmit buffer. For example, if Buffer 2 (B2) is programmed as a transmit buffer, FIFO consists of RXB0, RXB1, B0 and B1 – creating a FIFO length of 4. If all programmable buffers are configured as receive buffers, FIFO will have the maximum length of 8.

The following is the list of resources available in Mode 2:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX; receive buffers form FIFO: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC, useful for DeviceNet protocol

24.5 CAN Message Buffers

24.5.1 DEDICATED TRANSMIT BUFFERS

The PIC18F2480/2580/4480/4580 devices implement three dedicated transmit buffers – TXB0, TXB1 and TXB2. Each of these buffers occupies 14 bytes of SRAM and are mapped into the SFR memory map. These are the only transmit buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each transmit buffer contains one Control register (TXBnCON), four Identifier registers (TXBnSIDL, TXBnSIDH, TXBnEIDL, TXBnEIDH), one Data Length Count register (TXBnDLC) and eight Data Byte registers (TXBnDm).

24.5.2 DEDICATED RECEIVE BUFFERS

The PIC18F2480/2580/4480/4580 devices implement two dedicated receive buffers: RXB0 and RXB1. Each of these buffers occupies 14 bytes of SRAM and are mapped into SFR memory map. These are the only receive buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each receive buffer contains one Control register (RXBnCON), four Identifier registers (RXBnSIDL, RXBnSIDH, RXBnEIDL, RXBnEIDH), one Data Length Count register (RXBnDLC) and eight Data Byte registers (RXBnDm).

There is also a separate Message Assembly Buffer (MAB) which acts as an additional receive buffer. MAB is always committed to receiving the next message from the bus and is not directly accessible to user firmware. The MAB assembles all incoming messages one by one. A message is transferred to appropriate receive buffers only if the corresponding acceptance filter criteria is met.

24.5.3 PROGRAMMABLE TRANSMIT/ RECEIVE BUFFERS

The ECAN module implements six new buffers: B0-B5. These buffers are individually programmable as either transmit or receive buffers. These buffers are available only in Mode 1 and 2. As with dedicated transmit and receive buffers, each of these programmable buffers occupies 14 bytes of SRAM and are mapped into SFR memory map.

Each buffer contains one Control register (BnCON), four Identifier registers (BnSIDL, BnSIDH, BnEIDL, BnEIDH), one Data Length Count register (BnDLC) and eight Data Byte registers (BnDm). Each of these registers contains two sets of control bits. Depending on whether the buffer is configured as transmit or receive, one would use the corresponding control bit set. By default, all buffers are configured as receive buffers. Each buffer can be individually configured as a transmit or receive buffer by setting the corresponding TXENn bit in the BSEL0 register.

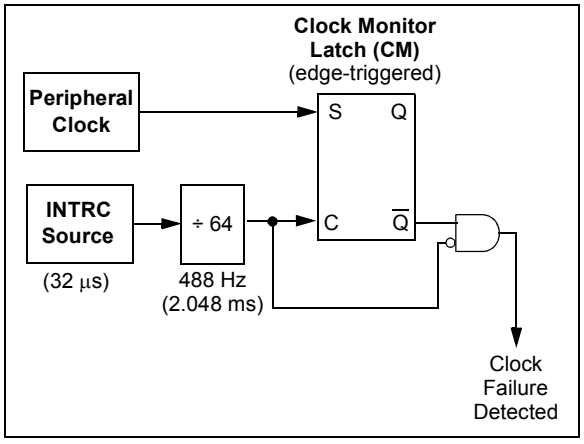
When configured as transmit buffers, user firmware may access transmit buffers in any order similar to accessing dedicated transmit buffers. In receive configuration with Mode 1 enabled, user firmware may also access receive buffers in any order required. But in Mode 2, all receive buffers are combined to form a single FIFO. Actual FIFO length is programmable by user firmware. Access to FIFO must be done through the FIFO Pointer bits (FP<4:0>) in the CANCON register. It must be noted that there is no hardware protection against out of order FIFO reads.

25.4 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the microcontroller to continue operation in the event of an external oscillator failure by automatically switching the device clock to the internal oscillator block. The FSCM function is enabled by setting the FCMEN Configuration bit.

When FSCM is enabled, the INTRC oscillator runs at all times to monitor clocks to peripherals and provide a backup clock in the event of a clock failure. Clock monitoring (shown in Figure 25-3) is accomplished by creating a sample clock signal, which is the INTRC output divided by 64. This allows ample time between FSCM sample clocks for a peripheral clock edge to occur. The peripheral device clock and the sample clock are presented as inputs to the Clock Monitor (CM) latch. The CM is set on the falling edge of the device clock source, but cleared on the rising edge of the sample clock.

FIGURE 25-3: FSCM BLOCK DIAGRAM



Clock failure is tested for on the falling edge of the sample clock. If a sample clock falling edge occurs while CM is still set, a clock failure has been detected (Figure 25-4). This causes the following:

- the FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>);
- the device clock source is switched to the internal oscillator block (OSCCON is not updated to show the current clock source – this is the Fail-Safe condition); and
- the WDT is reset.

During switchover, the postscaler frequency from the internal oscillator block may not be sufficiently stable for timing-sensitive applications. In these cases, it may be desirable to select another clock configuration and enter an alternate power-managed mode. This can be done to attempt a partial recovery or execute a controlled shutdown. See **Section 4.1.4 "Multiple Sleep Commands"** and **Section 25.3.1 "Special Considerations for Using Two-Speed Start-up"** for more details.

To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IRCF<2:0>, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF<2:0> bits prior to entering Sleep mode.

The FSCM will detect failures of the primary or secondary clock sources only. If the internal oscillator block fails, no failure would be detected, nor would any action be possible.

25.4.1 FSCM AND THE WATCHDOG TIMER

Both the FSCM and the WDT are clocked by the INTRC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTRC oscillator when the FSCM is enabled.

As already noted, the clock source is switched to the INTOSC clock when a clock failure is detected. Depending on the frequency selected by the IRCF<2:0> bits, this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed allows a WDT time-out to occur and a subsequent device Reset. For this reason, Fail-Safe Clock events also reset the WDT and postscaler, allowing it to start timing from when execution speed was changed and decreasing the likelihood of an erroneous time-out.

25.4.2 EXITING FAIL-SAFE OPERATION

The Fail-Safe condition is terminated by either a device Reset or by entering a power-managed mode. On Reset, the controller starts the primary clock source specified in Configuration Register 1H (with any required start-up delays that are required for the oscillator mode, such as OST or PLL timer). The INTOSC multiplexer provides the device clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock source is then switched to the primary clock (indicated by the OSTS bit in the OSCCON register becoming set). The Fail-Safe Clock Monitor then resumes monitoring the peripheral clock.

The primary clock source may never become ready during start-up. In this case, operation is clocked by the INTOSC multiplexer. The OSCCON register will remain in its Reset state until a power-managed mode is entered.

COMF **Complement f**

Syntax: COMF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ dest

Status Affected: N, Z

Encoding:

0001	11da	ffff	ffff

Description: The contents of register 'f' are complemented. If 'd' is '1', the result is stored in W. If 'd' is '0', the result is stored back in register 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "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: `COMF REG, 0, 0`

Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

CPFSEQ **Compare f with W, Skip if f = W**

Syntax: CPFSEQ f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '0', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

26.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F2480/2580/4480/4580 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers or use them for indexed addressing. Two of the instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and de-allocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 26-3. Detailed descriptions are provided in **Section 26.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 26-1 apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

26.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets ("[ ]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 26.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }").

TABLE 26-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
ADDFSR	f, k	Add Literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK	k	Add Literal to FSR2 and Return	2	1110	1000	11kk	kkkk	None
CALLW		Call Subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word	2	1110	1011	0zzz	zzzz	None
		f_d (destination) 2nd word		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	Move z_s (source) to 1st word	2	1110	1011	1zzz	zzzz	None
		z_d (destination) 2nd word		1111	xxxx	xzzz	zzzz	
PUSHL	k	Store Literal at FSR2, Decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR	f, k	Subtract Literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK	k	Subtract Literal from FSR2 and Return	2	1110	1001	11kk	kkkk	None