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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	48KB (24K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.25K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.23x24.23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf6585t-i-l

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTF is a bidirectional I/O port.
RF0/AN5	18	28	24			
RF0				I/O	ST	Digital I/O.
AN5				I	Analog	Analog input 5.
RF1/AN6/C2OUT	17	27	23			
RF1				I/O	ST	Digital I/O.
AN6				I	Analog	Analog input 6.
C2OUT				O	ST	Comparator 2 output.
RF2/AN7/C1OUT	16	26	18			
RF2				I/O	ST	Digital I/O.
AN7				I	Analog	Analog input 7.
C1OUT				O	ST	Comparator 1 output.
RF3/AN8/C2IN+	15	25	17			
RF1				I/O	ST	Digital I/O.
AN8				I	Analog	Analog input 8.
C2IN+				I	Analog	Comparator 2 input (+).
RF4/AN9/C2IN-	14	24	16			
RF1				I/O	ST	Digital I/O.
AN9				I	Analog	Analog input 9.
C2IN-				I	Analog	Comparator 2 input (-).
RF5/AN10/C1IN+/CVREF	13	23	15			
RF1						
AN10				I/O	ST	Digital I/O.
C1IN+				I	Analog	Analog input 10.
CVREF				I	Analog	Comparator 1 input (+).
				O	Analog	Comparator VREF output.
RF6/AN11/C1IN-	12	22	14			
RF6				I/O	ST	Digital I/O.
AN11				I	Analog	Analog input 11.
C1IN-				I	Analog	Comparator 1 input (-)
RF7/$\overline{SS}$	11	21	13			
RF7				I/O	ST	Digital I/O.
$\overline{SS}$				I	TTL	SPI slave select input.

Legend:

TTL	= TTL compatible input	CMOS	= CMOS compatible input or output
ST	= Schmitt Trigger input with CMOS levels	Analog	= Analog input
I	= Input	O	= Output
P	= Power	OD	= Open-Drain (no P diode to VDD)

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.

2: Default assignment when CCP2MX is set.

3: External memory interface functions are only available on PIC18F8X8X devices.

4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.

5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.

6: PSP is available in Microcontroller mode only.

7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Ranges Tested:			
Mode	**Freq**	**C1**	**C2**
LP	32.0 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	TBD	TBD
These values are for design guidance only. See notes following this table.			
Crystals Used			
32.0 kHz	Epson C-001R32.768K-A	± 20 PPM	
200 kHz	STD XTL 200.000KHz	± 20 PPM	
1.0 MHz	ECS ECS-10-13-1	± 50 PPM	
4.0 MHz	ECS ECS-40-20-1	± 50 PPM	
8.0 MHz	Epson CA-301 8.000M-C	± 30 PPM	
20.0 MHz	Epson CA-301 20.000M-C	± 30 PPM	

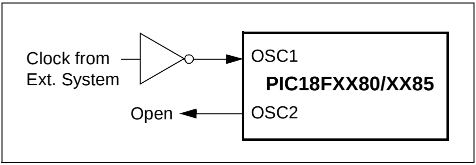
Note 1: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.

2: Rs (see Figure 2-1) may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specifications.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components, or verify oscillator performance.

An external clock source may also be connected to the OSC1 pin in the HS, XT and LP modes, as shown in Figure 2-2.

FIGURE 2-2: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)

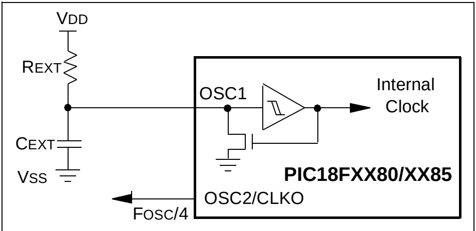


2.3 RC Oscillator

For timing insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit, due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 2-3 shows how the R/C combination is connected.

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 2-3: RC OSCILLATOR MODE

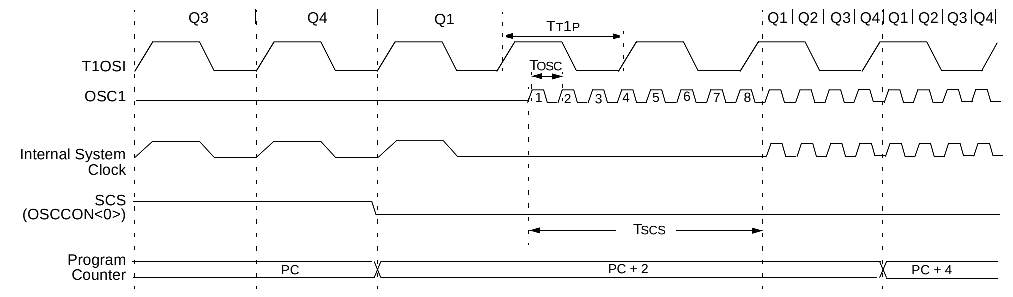


Recommended values: 3 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20pF

The RCIO Oscillator mode 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).

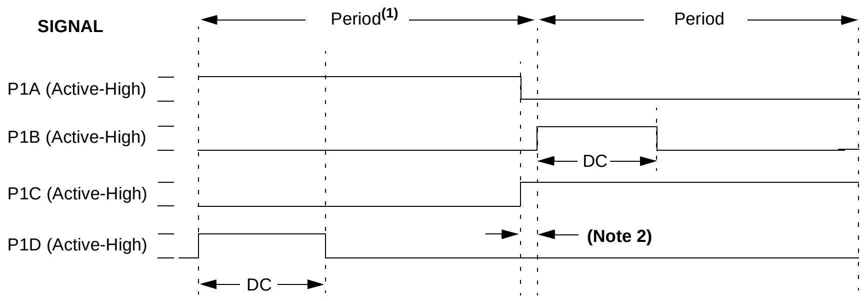
If the main oscillator is configured in the RC, RCIO, EC or ECIO modes, there is no oscillator start-up time-out. Operation will resume after eight cycles of the main oscillator have been counted. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for RC, RCIO, EC and ECIO modes, is shown in Figure 2-12.

FIGURE 2-12: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (RC, EC)



Note: RC Oscillator mode assumed.

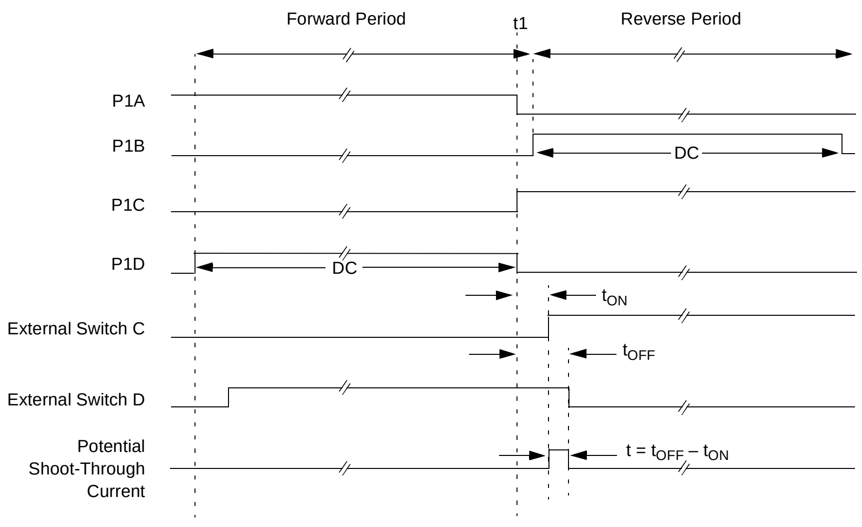
FIGURE 16-9: PWM DIRECTION CHANGE



Note 1: The direction bit in the CCP1 Control register (CCP1CON<7>) is written any time during the PWM cycle.

2: When changing directions, the P1A and P1C signals switch before the end of the current PWM cycle at intervals of 4 TOSC, 16 TOSC or 64 TOSC, depending on the Timer2 prescaler value. The modulated P1B and P1D signals are inactive at this time.

FIGURE 16-10: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



Note 1: All signals are shown as active-high.

2: t_{ON} is the turn on delay of power switch QC and its driver.

3: t_{OFF} is the turn off delay of power switch QD and its driver.

17.4.3.2 Reception

When the $R/\overline{W}$ bit of the address byte is clear and an address match occurs, the $R/\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{ACK}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set or bit SSPOV (SSPCON1<6>) is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR1<3>) must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON2<0> = `1`), RC3/SCK/SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit CKP (SSPCON<4>). See **Section 17.4.4 "Clock Stretching"** for more detail.

17.4.3.3 Transmission

When the $R/\overline{W}$ bit of the incoming address byte is set and an address match occurs, the $R/\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low, regardless of SEN (see **Section 17.4.4 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register which also loads the SSPSR register. Then pin RC3/SCK/SCL should be enabled by setting bit CKP (SSPCON1<4>). The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 17-9).

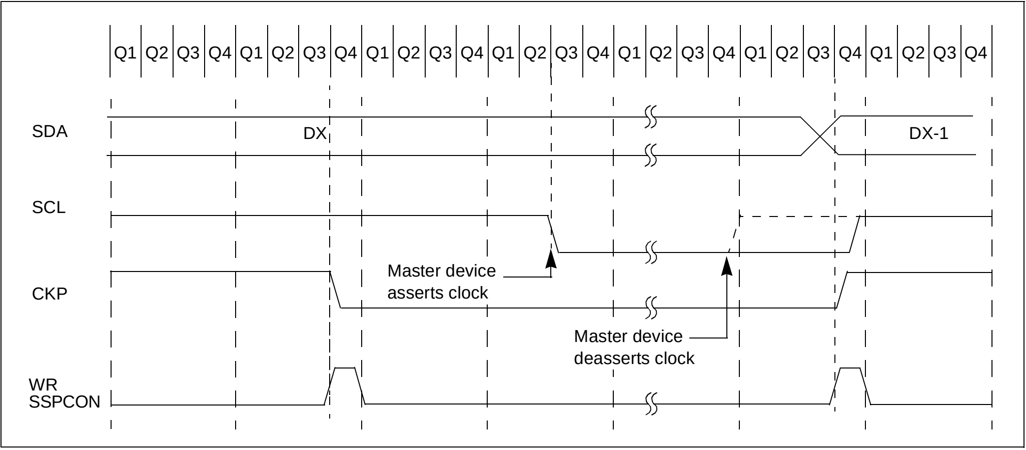
The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{ACK}$), then the data transfer is complete. In this case, when the $\overline{ACK}$ is latched by the slave, the slave logic is reset (resets SSPSTAT register) and the slave monitors for another occurrence of the Start bit. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPBUF register. Again, pin RC3/SCK/SCL must be enabled by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

17.4.4.5 Clock Synchronization and the CKP bit

When the CKP bit is cleared, the SCL output is forced to '0'. However, setting the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I^{2}C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I^{2}C bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 17-12).

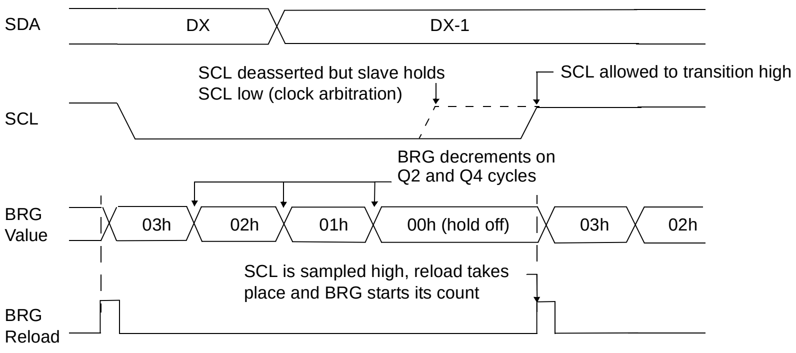
FIGURE 17-12: CLOCK SYNCHRONIZATION TIMING


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
SDA
DX
DX-1
SCL
CKP
Master device asserts clock
Master device deasserts clock
WR SSPCON

17.4.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 17-18).

FIGURE 17-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION


SDA
DX
DX-1
SCL deasserted but slave holds SCL low (clock arbitration)
SCL allowed to transition high
SCL
BRG decrements on Q2 and Q4 cycles
BRG Value
03h
02h
01h
00h (hold off)
03h
02h
SCL is sampled high, reload takes place and BRG starts its count
BRG Reload

17.4.17.3 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<6:0> 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 17-31). 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 17-32).

FIGURE 17-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)

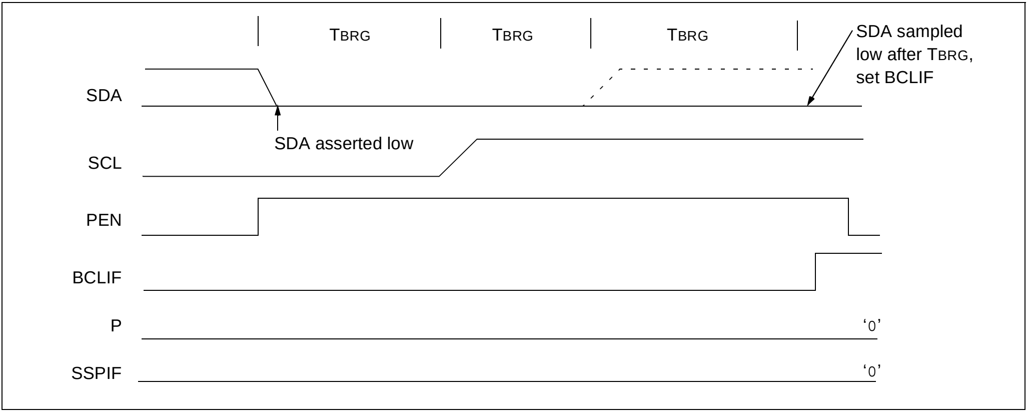
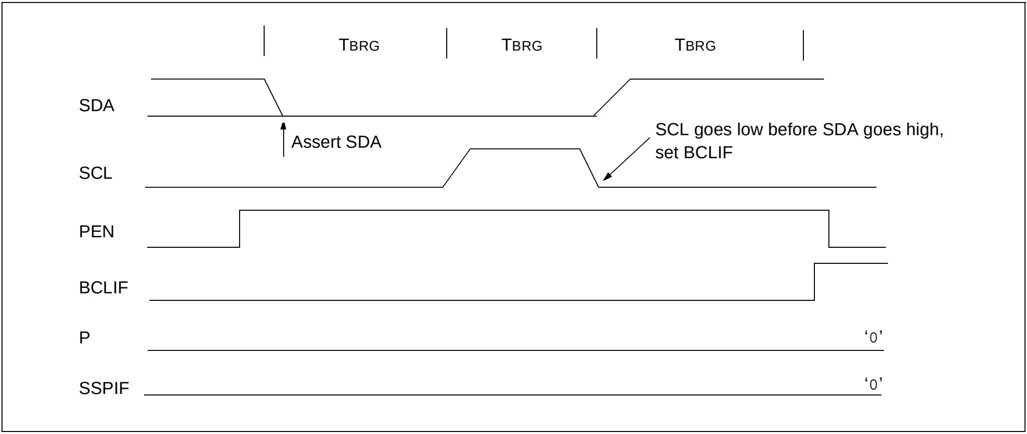


FIGURE 17-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)



18.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on sync break reception and 12-bit break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN bus) systems.

The USART can be configured in the following modes:

- Asynchronous (full-duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit break character transmission
- Synchronous – Master (half-duplex) with selectable clock polarity
- Synchronous – Slave (half-duplex) with selectable clock polarity

In order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter:

- SPEN (RCSTA<7>) bit must be set (= `1`),
- TRISC<6> bit must be set (= `1`), and
- TRISC<7> bit must be set (= `1`).

Note:	The USART control will automatically reconfigure the pin from input to output as needed.

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

These are detailed on the following pages in Register 18-1, Register 18-2 and Register 18-3, respectively.

18.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the USART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RX/DT line while the USART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCON<1>). Once set, the typical receive sequence on RX/DT is disabled and the USART remains in an Idle state monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a sync break or a wake-up signal character for the LIN protocol.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 18-7) and asynchronously, if the device is in Sleep mode (Figure 18-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line following the wake-up event. At this point, the USART module is in Idle mode and returns to normal operation. This signals to the user that the sync break event is over.

18.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false end-of-character and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bytes) for standard RS-232 devices or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., XT or HS mode). The sync break (or wake-up signal) character must be of sufficient length and be followed by a sufficient interval to allow enough time for the selected oscillator to start and provide proper initialization of the USART.

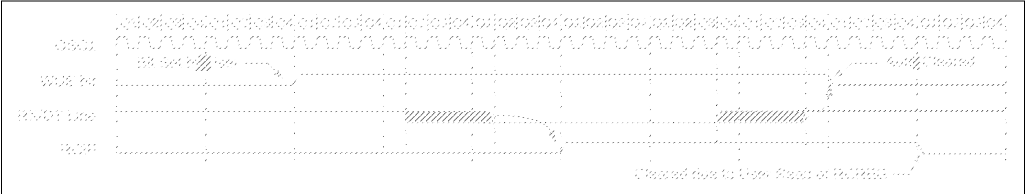
18.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the USART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

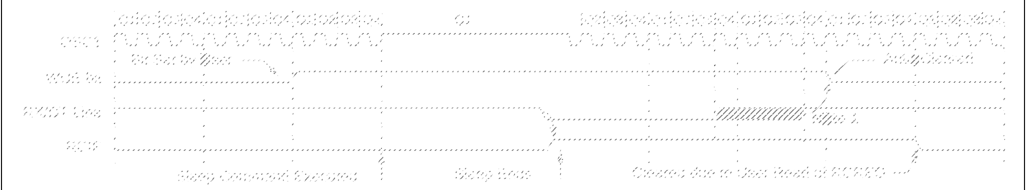
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

FIGURE 18-7: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION



Note: The USART remains in Idle while the WUE bit is set.

FIGURE 18-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



Note 1: If the wake-up event requires long oscillator warm-up time, the auto-clear of the WUE bit can occur while the stposc signal is still active. This sequence should not depend on the presence of Q clocks.

2: The USART remains in Idle while the WUE bit is set.

18.4 USART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit CSRC (TXSTA<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the RC6/TX/CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

18.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical except in the case of the Sleep mode.

If two words are written to the TXREG and then the `SLEEP` instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.

b) The second word will remain in TXREG register.

c) Flag bit TXIF will not be set.

d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit TXIF will now be set.

e) If enable bit TXIE is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a synchronous slave transmission:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 18-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-1 0-00	-1-1 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

REGISTER 23-2: CANSTAT: CAN STATUS REGISTER

	R-1	R-0	R-0	R-0	R-0	R-0	R-0	U-0
Mode 0	OPMODE2[1]	OPMODE1[1]	OPMODE0[1]	—	ICODE2	ICODE1	ICODE0	—

	R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0
Mode 1, 2	OPMODE2[1]	OPMODE1[1]	OPMODE0[1]	EICODE4	EICODE3	EICODE2	EICODE1	EICODE0
	bit 7							bit 0

bit 7-5 **OPMODE2:OPMODE0:** Operation Mode Status bits[1]

`111` = Reserved
`110` = Reserved
`101` = Reserved
`100` = Configuration mode
`011` = Listen Only mode
`010` = Loopback mode
`001` = Disable/Sleep mode
`000` = Normal mode

bit 4 Mode 0:

Unimplemented: Read as '`0`'

bit 3-1 **ICODE2:ICODE0**: Interrupt Code bits in Mode 0

When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. By copying ICODE2:ICODE0 to WIN2:WIN0, it is possible to select the correct buffer to map into the Access Bank area. See Example 23-2 for a code example.

	ICODE2:ICODE0 Value
No interrupt	`000`
Error interrupt	`001`
TXB2 interrupt	`010`
TXB1 interrupt	`011`
TXB0 interrupt	`100`
RXB1 interrupt	`101`
RXB0 interrupt	`110`
Wake-up interrupt	`111`

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

EXAMPLE 23-3: 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.
```

REGISTER 23-48: MSEL0: MASK SELECT REGISTER 0(1)

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0
bit 7							bit 0

bit 7-6 **FIL3_1:FIL3_0:** Filter 3 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 5-4 **FIL2_1:FIL2_0:** Filter 2 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 3-2 **FIL1_1:FIL1_0:** Filter 1 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 1-0 **FIL0_1:FIL0_0:** Filter 0 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

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

25.1 Instruction Set

ADDLW — ADD literal to W

Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → W
Status Affected:	N, OV, C, DC, Z
Encoding:	`0000` `1111` `kkkk` `kkkk`
Description:	The contents of W are added to the 8-bit literal ‘k’ and 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: `ADDLW 0x15`

Before Instruction

W = 0x10

After Instruction

W = 0x25

ADDWF — ADD W to f

Syntax:	[*label*] ADDWF f [,d [,a] f [,d [,a]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(W) + (f) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0010` `01da` `ffff` `ffff`
Description:	Add W to register ‘f’. If ‘d’ is ‘`0`’, the result is stored in W. If ‘d’ is ‘`1`’, the result is stored back in register ‘d’ (default). If ‘a’ is ‘`0`’, the Access Bank will be selected. If ‘a’ is ‘`1`’, the BSR is used.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ADDWF REG, 0, 0`

Before Instruction

W = 0x17
REG = 0xC2

After Instruction

W = 0xD9
REG = 0xC2

COMF — Complement f

Syntax:	[*label*] COMF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(\overline{f}) \rightarrow$ dest
Status Affected:	N, Z
Encoding:	0001 11da ffff ffff
Description:	The contents of register 'f' are complemented. 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
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	=	0x13

After Instruction

REG	=	0x13
W	=	0xEC

CPFSEQ — Compare f with W, skip if f = W

Syntax:	[*label*] 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
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)

DECFSZ	**Decrement f, skip if 0**
Syntax:	[*label*] DECFSZ f [,d [,a]]
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – 1 → dest, skip if result = `0`
Status Affected:	None
Encoding:	`0010` `11da` `ffff` `ffff`
Description:	The contents of register 'f' are decremented. If 'd' is '`0`', the result is placed in W. If 'd' is '`1`', the result is placed back in register 'f' (default). If the result is '`0`', the next instruction which is already fetched is discarded and a `NOP` is executed instead, making it a two-cycle instruction. If 'a' is '`0`', the Access Bank will be selected, overriding the BSR value. If 'a' = `1`, then the bank will be selected as per the BSR value (default).
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	Write to destination

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        DECFSZ   CNT, 1, 1
            GOTO     LOOP
CONTINUE
```

Before Instruction

PC	=	Address `(HERE)`

After Instruction

CNT	=	CNT - 1
If CNT	=	0;
PC	=	Address `(CONTINUE)`
If CNT	≠	0;
PC	=	Address `(HERE+2)`

DCFSNZ	**Decrement f, skip if not 0**
Syntax:	[*label*] DCFSNZ f [,d [,a]]
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – 1 → dest, skip if result ≠ `0`
Status Affected:	None
Encoding:	`0100` `11da` `ffff` `ffff`
Description:	The contents of register 'f' are decremented. If 'd' is '`0`', the result is placed in W. If 'd' is '`1`', the result is placed back in register 'f' (default). If the result is not '`0`', the next instruction which is already fetched is discarded and a `NOP` is executed instead, making it a two-cycle instruction. If 'a' is '`0`', the Access Bank will be selected, overriding the BSR value. If 'a' = `1`, then the bank will be selected as per the BSR value (default).
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	Write to destination

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     DCFSNZ  TEMP, 1, 0
ZERO     :
NZERO    :
```

Before Instruction

TEMP	=	?

After Instruction

TEMP	=	TEMP - 1,
If TEMP	=	0;
PC	=	Address `(ZERO)`
If TEMP	≠	0;
PC	=	Address `(NZERO)`

26.9 MPLAB ICE 2000 High-Performance Universal In-Circuit Emulator

The MPLAB ICE 2000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers. Software control of the MPLAB ICE 2000 in-circuit emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE 2000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

26.10 MPLAB ICE 4000 High-Performance Universal In-Circuit Emulator

The MPLAB ICE 4000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for high-end PIC microcontrollers. Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICD 4000 is a premium emulator system, providing the features of MPLAB ICE 2000, but with increased emulation memory and high-speed performance for dsPIC30F and PIC18XXXX devices. Its advanced emulator features include complex triggering and timing, up to 2 Mb of emulation memory and the ability to view variables in real-time.

The MPLAB ICE 4000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

26.11 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PIC MCUs and can be used to develop for these and other PIC microcontrollers. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost effective in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single-stepping and watching variables, CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real-time. MPLAB ICD 2 also serves as a development programmer for selected PIC devices.

26.12 PRO MATE II Universal Device Programmer

The PRO MATE II is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} for maximum reliability. It features an LCD display for instructions and error messages and a modular detachable socket assembly to support various package types. In Stand-Alone mode, the PRO MATE II device programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode.

26.13 MPLAB PM3 Device Programmer

The MPLAB PM3 is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 device programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. MPLAB PM3 connects to the host PC via an RS-232 or USB cable. MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an SD/MMC card for file storage and secure data applications.

27.3 DC Characteristics: PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial)

DC CHARACTERISTICS			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial; -40°C ≤ TA ≤ +125°C for extended			
Param No.	**Symbol**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
	V_{IL}	**Input Low Voltage**				
		I/O ports:				
D030		with TTL buffer	V_{SS}	0.15 V_{DD}	V	V_{DD} < 4.5V
D030A			—	0.8	V	4.5V ≤ V_{DD} ≤ 5.5V
D031		with Schmitt Trigger buffer RC3 and RC4	V_{SS} V_{SS}	0.2 V_{DD} 0.3 V_{DD}	V V	
D032		$\overline{\text{MCLR}}$	V_{SS}	0.2 V_{DD}	V	
D032A		OSC1 (in XT, HS and LP modes) and T1OSI	V_{SS}	0.3 V_{DD}	V	
D033		OSC1 (in RC and EC mode)(1)	V_{SS}	0.2 V_{DD}	V	
	V_{IH}	**Input High Voltage**				
		I/O ports:				
D040		with TTL buffer	0.25 V_{DD} + 0.8V	V_{DD}	V	V_{DD} < 4.5V
D040A			2.0	V_{DD}	V	4.5V ≤ V_{DD} ≤ 5.5V
D041		with Schmitt Trigger buffer RC3 and RC4	0.8 V_{DD} 0.7 V_{DD}	V_{DD} V_{DD}	V V	
D042		$\overline{\text{MCLR}}$, OSC1 (EC mode)	0.8 V_{DD}	V_{DD}	V	
D042A		OSC1 (in XT, HS and LP modes) and T1OSI	0.7 V_{DD}	V_{DD}	V	
D043		OSC1 (RC mode)(1)	0.9 V_{DD}	V_{DD}	V	
	I_{IL}	**Input Leakage Current(2,3)**				
D060		I/O ports	—	±1	μA	V_{SS} ≤ V_{PIN} ≤ V_{DD}, Pin at high-impedance
D061		$\overline{\text{MCLR}}$	—	±5	μA	Vss ≤ V_{PIN} ≤ V_{DD}
D063		OSC1	—	±5	μA	Vss ≤ V_{PIN} ≤ V_{DD}
	I_{PU}	**Weak Pull-up Current**				
D070	I_{PURB}	PORTB weak pull-up current	50	400	μA	V_{DD} = 5V, V_{PIN} = V_{SS}

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC device be driven with an external clock while in RC mode.

2: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

4: Parameter is characterized but not tested.

TABLE 27-1: COMPARATOR SPECIFICATIONS

Operating Conditions: 3.0V < V_{DD} < 5.5V, -40°C < T_A < +125°C, unless otherwise stated							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D300	VIOFF	Input Offset Voltage	—	± 5.0	± 10	mV	
D301	VICM	Input Common Mode Voltage	0	—	VDD – 1.5	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
300 300A	TRESP	Response Time(1)	—	150	400 600	ns ns	PIC18FXX8X PIC18LFXX8X
301	TMC2OV	Comparator Mode Change to Output Valid	—	—	10	µs	

Note 1: Response time measured with one comparator input at (VDD – 1.5)/2 while the other input transitions from VSS to VDD.

TABLE 27-2: VOLTAGE REFERENCE SPECIFICATIONS

Operating Conditions: 3.0V < V_{DD} < 5.5V, -40°C < T_A < +125°C, unless otherwise stated							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D310	VRES	Resolution	VDD/24	—	VDD/32	LSb	
D311	VRAA	Absolute Accuracy	— —	— —	1/4 1/2	LSb LSb	Low Range (VRR = 1) High Range (VRR = 0)
D312	VRUR	Unit Resistor Value (R)	—	2k	—	Ω	
310	TSET	Settling Time(1)	—	—	10	µs	

Note 1: Settling time measured while VRR = 1 and VR<3:0> transitions from 0000 to 1111.