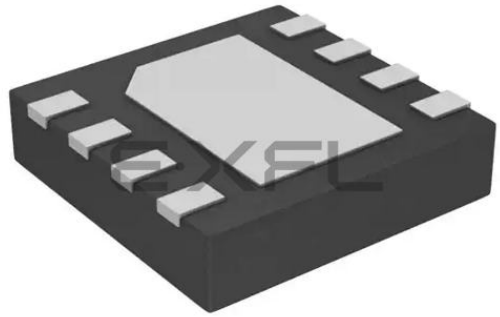


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	-
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	6
Program Memory Size	1.75KB (1K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 4x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	8-UDFN Exposed Pad
Supplier Device Package	8-UDFN (3x3)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic12lf1571-i-rf

3.2.1 READING PROGRAM MEMORY AS DATA

There are two methods of accessing constants in program memory. The first method is to use tables of RETLW instructions. The second method is to set an FSR to point to the program memory.

3.2.1.1 RETLW Instruction

The RETLW instruction can be used to provide access to tables of constants. The recommended way to create such a table is shown in Example 3-1.

EXAMPLE 3-1: RETLW INSTRUCTION

```
constants
    BRW                 ;Add Index in W to
                        ;program counter to
                        ;select data
    RETLW DATA0         ;Index0 data
    RETLW DATA1         ;Index1 data
    RETLW DATA2
    RETLW DATA3


my_function
    ;… LOTS OF CODE…
    MOVLW       DATA_INDEX
    call constants
    ;… THE CONSTANT IS IN W
```

The BRW instruction makes this type of table very simple to implement. If your code must remain portable with previous generations of microcontrollers, then the BRW instruction is not available, so the older table read method must be used.

3.2.1.2 Indirect Read with FSR

The program memory can be accessed as data by setting bit 7 of the FSRnH register and reading the matching INDFn register. The MOVIW instruction will place the lower eight bits of the addressed word in the W register. Writes to the program memory cannot be performed via the INDFn registers. Instructions that access the program memory via the FSR require one extra instruction cycle to complete. Example 3-2 demonstrates accessing the program memory via an FSR.

The HIGH operator will set bit<7> if a label points to a location in program memory.

EXAMPLE 3-2: ACCESSING PROGRAM MEMORY VIA FSR

```
constants
  DW     DATA0         ;First constant
  DW     DATA1         ;Second constant
  DW     DATA2
  DW     DATA3
my_function
  ;… LOTS OF CODE…
  MOVLW  DATA_INDEX
  ADDLW  LOW constants
  MOVWF  FSR1L
  MOVLW  HIGH constants ;MSb is set
                         automatically
  MOVWF  FSR1H
  BTFSC  STATUS,C      ;carry from ADDLW?
  INCF   FSR1H,f       ;yes
  MOVIW  0[FSR1]
;THE PROGRAM MEMORY IS IN W
```

TABLE 3-10: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	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
Bank 4											
20Ch	WPUA	—	—	WPUA<5:0>						--11 1111	--11 1111
20Dh	—	Unimplemented								—	—
20Eh to 21Fh	—	Unimplemented								—	—
Bank 5											
28Ch	ODCONA	—	—	ODA<5:4>		—	ODA<2:0>			--11 -111	--11 -111
28Dh to 29Fh	—	Unimplemented								—	—
Bank 6											
30Ch	SLRCONA	—	—	SLRA<5:4>		—	SLRA<2:0>			--11 -111	--11 -111
30Dh to 31Fh	—	Unimplemented								—	—
Bank 7											
38Ch	INLVLA	—	—	INLVLA<5:0>						--11 1111	--11 1111
38Dh to 390h	—	Unimplemented								—	—
391h	IOCAP	—	—	IOCAP<5:0>						--00 0000	--00 0000
392h	IOCAN	—	—	IOCAN<5:0>						--00 0000	--00 0000
393h	IOCAF	—	—	IOCAF<5:0>						--00 0000	--00 0000
394h to 39Fh	—	Unimplemented								—	—
Bank 8											
40Ch to 41Fh	—	Unimplemented								—	—
Bank 9											
48Ch to 49Fh	—	Unimplemented								—	—

Legend: x = unknown; u = unchanged; q = value depends on condition; — = unimplemented; r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: PIC12F1571/2 only.

2: PIC12(L)F1572 only.

3: Unimplemented, read as '1'.

TABLE 3-10: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	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
Bank 10											
50Ch to 51Fh	—	Unimplemented								—	—
Bank 11											
58Ch to 59Fh	—	Unimplemented								—	—
Bank 12											
60Ch to 61Fh	—	Unimplemented								—	—
Bank 13											
68Ch to 690h	—	Unimplemented								—	—
691h	CWG1DBR	—	—	CWG1DBR<5:0>						--00 0000	--00 0000
692h	CWG1DBF	—	—	CWG1DBF<5:0>						--xx xxxx	--xx xxxx
693h	CWG1CON0	G1EN	G1OEB	G1OEA	G1POLB	G1POLA	—	—	G1CS0	0000 0--0	0000 0--0
694h	CWG1CON1	G1ASDLB<1:0>		G1ASDLA<1:0>		—	G1IS<2:0>			0000 -000	0000 -000
695h	CWG1CON2	G1ASE	G1ARSEN	—	—	—	G1ASDSC1	G1ASDSFLT	—	00-- -00-	00-- -00-
696h to 69Fh	—	Unimplemented								—	—
Banks 14-26											
x0Ch/ x8Ch — x1Fh/ x9Fh	—	Unimplemented								—	—

Legend: x = unknown; u = unchanged; q = value depends on condition; — = unimplemented; r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: PIC12F1571/2 only.

2: PIC12(L)F1572 only.

3: Unimplemented, read as '1'.

TABLE 3-10: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	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
Bank 27											
D8Ch	—	Unimplemented								—	—
D8Dh	—	Unimplemented								—	—
D8Eh	PWMEN	—	—	—	—	—	PWM3EN_A	PWM2EN_A	PWM1EN_A	---- -000	---- -000
D8Fh	PWMLD	—	—	—	—	—	PWM3LDA_A	PWM2LDA_A	PWM1LDA_A	---- -000	---- -000
D90h	PWMOUT	—	—	—	—	—	PWM3OUT_A	PWM2OUT_A	PWM1OUT_A	---- -000	---- -000
D91h	PWM1PHL	PH<7:0>								xxxx xxxx	uuuu uuuu
D92h	PWM1PHH	PH<15:8>								xxxx xxxx	uuuu uuuu
D93h	PWM1DCL	DC<7:0>								xxxx xxxx	uuuu uuuu
D94h	PWM1DCH	DC<15:8>								xxxx xxxx	uuuu uuuu
D95h	PWM1PRL	PR<7:0>								xxxx xxxx	uuuu uuuu
D96h	PWM1PRH	PR<15:8>								xxxx xxxx	uuuu uuuu
D97h	PWM1OFL	OF<7:0>								xxxx xxxx	uuuu uuuu
D98h	PWM1OFH	OF<15:8>								xxxx xxxx	uuuu uuuu
D99h	PWM1TMRL	TMR<7:0>								xxxx xxxx	uuuu uuuu
D9Ah	PWM1TMRH	TMR<15:8>								xxxx xxxx	uuuu uuuu
D9Bh	PWM1CON	PWM1EN	PWM1OE	PWM1OUT	PWM1POL	PWM1MODE<1:0>		—	—	0000 00--	0000 00--
D9Ch	PWM1INTE	—	—	—	—	PWM1OFIE	PWM1PHIE	PWM1DCIE	PWM1PRIE	---- 000	---- 000
D9Dh	PWM1INTF	—	—	—	—	PWM1OFIF	PWM1PHIF	PWM1DCIF	PWM1PRIF	---- 000	---- 000
D9Eh	PWM1CLKCON	—	PWM1PS<2:0>			—	—	PWM1CS<1:0>		-000 -000	-000 --00
D9Fh	PWM1LDCON	PWM1LDA	PWM1LDT	—	—	—	—	PWM1LDS<1:0>		00-- -000	00-- --00
DA0h	PWM1OFCON	—	PWM1OFM<1:0>		PWM1OFO	—	—	PWM1OFS<1:0>		-000 -000	-000 --00
DA1h	PWM2PHL	PH<7:0>								xxxx xxxx	uuuu uuuu
DA2h	PWM2PHH	PH<15:8>								xxxx xxxx	uuuu uuuu
DA3h	PWM2DCL	DC<7:0>								xxxx xxxx	uuuu uuuu
DA4h	PWM2DCH	DC<15:8>								xxxx xxxx	uuuu uuuu
DA5h	PWM2PRL	PR<7:0>								xxxx xxxx	uuuu uuuu
DA6h	PWM2PRH	PR<15:8>								xxxx xxxx	uuuu uuuu
DA7h	PWM2OFL	OF<7:0>								xxxx xxxx	uuuu uuuu
DA8h	PWM2OFH	OF<15:8>								xxxx xxxx	uuuu uuuu
DA9h	PWM2TMRL	TMR<7:0>								xxxx xxxx	uuuu uuuu
DAAh	PWM2TMRH	TMR<15:8>								xxxx xxxx	uuuu uuuu
DABh	PWM2CON	PWM2EN	PWM2OE	PWM2OUT	PWM2POL	PWM2MODE<1:0>		—	—	0000 00--	0000 00--
DACh	PWM2INTE	—	—	—	—	PWM2OFIE	PWM2PHIE	PWM2DCIE	PWM2PRIE	---- 000	---- 000
DADh	PWM2INTF	—	—	—	—	PWM2OFIF	PWM2PHIF	PWM2DCIF	PWM2PRIF	---- 000	---- 000
DAEh	PWM2CLKCON	—	PWM2PS<2:0>			—	—	PWM2CS<1:0>		-000 -000	-000 --00
DAFh	PWM2LDCON	PWM2LDA	PWM2LDT	—	—	—	—	PWM2LDS<1:0>		00-- -000	00-- --00

Legend: x = unknown; u = unchanged; q = value depends on condition; — = unimplemented; r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: PIC12F1571/2 only.

2: PIC12(L)F1572 only.

3: Unimplemented, read as '1'.

6.3 Register Definitions: BOR Control

REGISTER 6-1: BORCON: BROWN-OUT RESET CONTROL REGISTER

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

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

bit 7 **SBOREN:** Software Brown-out Reset Enable bit

If BOREN<1:0> in Configuration Words = 01:

1 = BOR is enabled

0 = BOR is disabled

If BOREN <1:0> in Configuration Words ≠ 01:

SBOREN is read/write, but has no effect on the BOR.

bit 6 **BORFS:** Brown-out Reset Fast Start bit(1)

If BOREN <1:0> = 10 (Disabled in Sleep) or BOREN<1:0> = 01 (Under software control):

1 = Band gap is forced on always (covers Sleep/wake-up/operating cases)

0 = Band gap operates normally and may turn off

If BOREN<1:0> = 11 (Always On) or BOREN<1:0> = 00 (Always Off):

BORFS is read/write, but has no effect on the BOR.

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

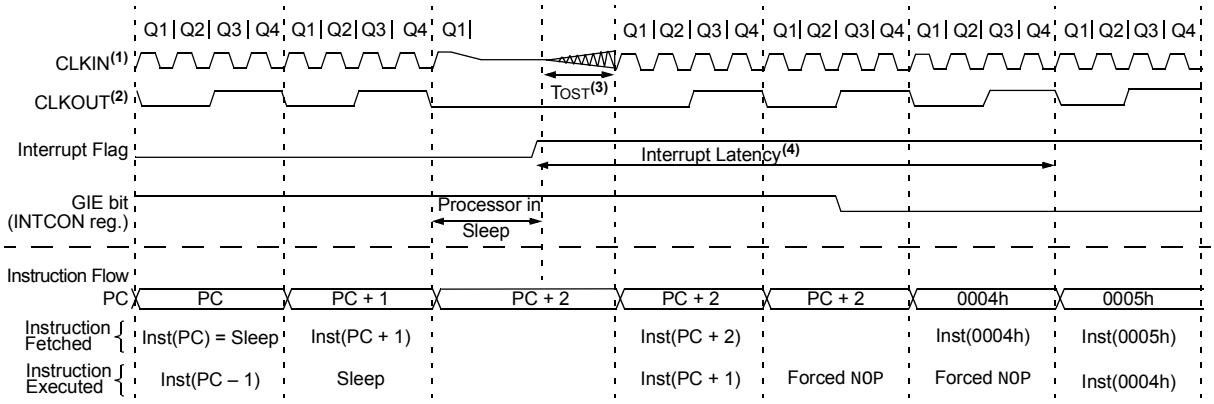
bit 0 **BORRDY:** Brown-out Reset Circuit Ready Status bit

1 = The Brown-out Reset circuit is active

0 = The Brown-out Reset circuit is inactive

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

FIGURE 8-1: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: External Clock. High, Medium, Low mode assumed.
2: CLKOUT is shown here for timing reference.
3: TOST = 1024 TOSC. This delay does not apply to EC, RC and INTOSC Oscillator modes or Two-Speed Start-up (if available).
4: GIE = 1 assumed. In this case, after wake-up, the processor calls the ISR at 0004h. If GIE = 0, execution will continue in-line.

8.2 Low-Power Sleep Mode

This device contains an internal Low Dropout (LDO) voltage regulator, which allows the device I/O pins to operate at voltages up to 5.5V while the internal device logic operates at a lower voltage. The LDO and its associated reference circuitry must remain active when the device is in Sleep mode.

Low-Power Sleep mode allows the user to optimize the operating current in Sleep. Low-Power Sleep mode can be selected by setting the VREGPM bit of the VREGCON register, which puts the LDO and reference circuitry in a low-power state whenever the device is in Sleep.

8.2.1 SLEEP CURRENT VS. WAKE-UP TIME

In the default operating mode, the LDO and reference circuitry remain in the normal configuration while in Sleep. The device is able to exit Sleep mode quickly since all circuits remain active. In Low-Power Sleep mode, when waking up from Sleep, an extra delay time is required for these circuits to return to the normal configuration and stabilize.

The Low-Power Sleep mode is beneficial for applications that stay in Sleep mode for long periods of time. The normal mode is beneficial for applications that need to wake from Sleep quickly and frequently.

8.2.2 PERIPHERAL USAGE IN SLEEP

Some peripherals that can operate in Sleep mode will not operate properly with the Low-Power Sleep mode selected. The LDO will remain in the normal power mode when those peripherals are enabled. The Low-Power Sleep mode is intended for use with these peripherals:

- Brown-out Reset (BOR)
- Watchdog Timer (WDT)
- External interrupt pin/Interrupt-On-Change pins
- Timer1 (with external clock source)

The Complementary Waveform Generator (CWG) module can utilize the HFINTOSC oscillator as either a clock source or as an input source. Under certain conditions, when the HFINTOSC is selected for use with the CWG module, the HFINTOSC will remain active during Sleep. This will have a direct effect on the Sleep mode current.

Please refer to section **Section 23.10 "Operation During Sleep"** for more information.

Note:	The PIC12LF1571/2 does not have a configurable Low-Power Sleep mode. PIC12LF1571/2 is an unregulated device and is always in the lowest power state when in Sleep with no wake-up time penalty. This device has a lower maximum VDD and I/O voltage than the PIC12F1571/2. See **Section 26.0 "Electrical Specifications"** for more information.

17.0 COMPARATOR MODULE

Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. Comparators are very useful mixed signal building blocks because they provide analog functionality independent of program execution. The analog comparator module includes the following features:

- Independent comparator control
- Programmable input selection
- Comparator output is available internally/externally
- Programmable output polarity
- Interrupt-On-Change
- Wake-up from Sleep
- Programmable speed/power optimization
- PWM shutdown
- Programmable and Fixed Voltage Reference

17.1 Comparator Overview

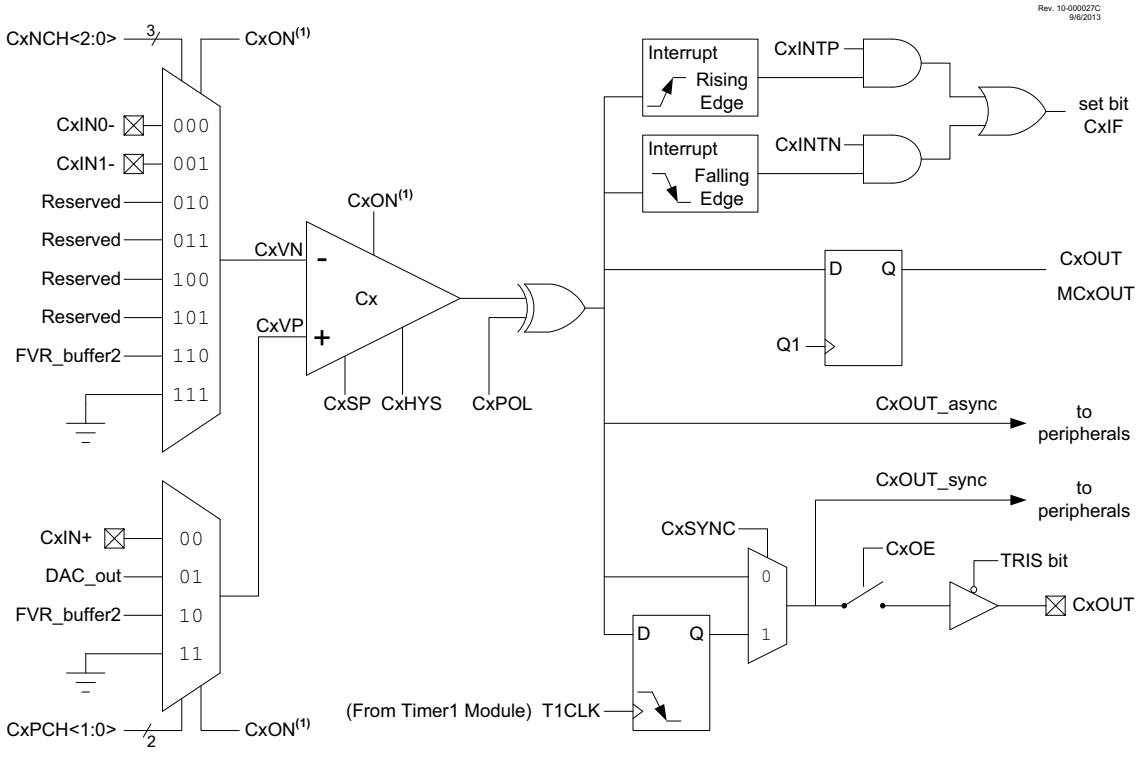
A single comparator is shown in Figure 17-2 along with the relationship between the analog input levels and the digital output. When the analog voltage at VIN+ is less than the analog voltage at VIN-, the output of the comparator is a digital low level. When the analog voltage at VIN+ is greater than the analog voltage at VIN-, the output of the comparator is a digital high level.

The comparators available for this device are listed in Table 17-1.

TABLE 17-1: AVAILABLE COMPARATORS

Device	C1
PIC12(L)F1571	•
PIC12(L)F1572	•

FIGURE 17-1: COMPARATOR MODULE SIMPLIFIED BLOCK DIAGRAM



Note 1: When CxON = 0, all multiplexer inputs are disconnected and the Comparator will produce a '0' at the output.

17.4 Comparator Hysteresis

A selectable amount of separation voltage can be added to the input pins of each comparator to provide a hysteresis function to the overall operation. Hysteresis is enabled by setting the CxHYS bit of the CMxCON0 register.

See **Section 26.0 "Electrical Specifications"** for more information.

17.5 Timer1 Gate Operation

The output resulting from a comparator operation can be used as a source for gate control of Timer1. See **Section 19.5 "Timer1 Gate"** for more information. This feature is useful for timing the duration or interval of an analog event.

It is recommended that the comparator output be synchronized to Timer1. This ensures that Timer1 does not increment while a change in the comparator is occurring.

17.5.1 COMPARATOR OUTPUT SYNCHRONIZATION

The output from the Cx comparator can be synchronized with Timer1 by setting the CxSYNC bit of the CMxCON0 register.

Once enabled, the comparator output is latched on the falling edge of the Timer1 source clock. If a prescaler is used with Timer1, the comparator output is latched after the prescaling function. To prevent a race condition, the comparator output is latched on the falling edge of the Timer1 clock source and Timer1 increments on the rising edge of its clock source. See the Comparator Block Diagram (Figure 17-2) and the Timer1 Block Diagram (Figure 19-1) for more information.

17.6 Comparator Interrupt

An interrupt can be generated upon a change in the output value of the comparator for each comparator, a rising edge detector and a falling edge detector are present.

When either edge detector is triggered and its associated enable bit is set (CxINTP and/or CxINTN bits of the CMxCON1 register), the corresponding interrupt flag bit (CxIF bit of the PIR2 register) will be set.

To enable the interrupt, you must set the following bits:

- CxON and CxPOL bits of the CMxCON0 register
- CxIE bit of the PIE2 register
- CxINTP bit of the CMxCON1 register (for a rising edge detection)
- CxINTN bit of the CMxCON1 register (for a falling edge detection)
- PEIE and GIE bits of the INTCON register

The associated interrupt flag bit, CxIF bit of the PIR2 register, must be cleared in software. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

Note:	Although a comparator is disabled, an interrupt can be generated by changing the output polarity with the CxPOL bit of the CMxCON0 register, or by switching the comparator on or off with the CxON bit of the CMxCON0 register.

17.7 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in **Section 26.0 "Electrical Specifications"** for more details.

19.6 Timer1 Interrupt

The Timer1 register pair (TMR1H:TMR1L) increments to FFFFh and rolls over to 0000h. When Timer1 rolls over, the Timer1 interrupt flag bit of the PIR1 register is set. To enable the interrupt on rollover, you must set these bits:

- TMR1ON bit of the T1CON register
- TMR1IE bit of the PIE1 register
- PEIE bit of the INTCON register
- GIE bit of the INTCON register

The interrupt is cleared by clearing the TMR1IF bit in the Interrupt Service Routine.

Note:	The TMR1H:TMR1L register pair and the TMR1IF bit should be cleared before enabling interrupts.

19.7 Timer1 Operation During Sleep

Timer1 can only operate during Sleep when set up in Asynchronous Counter mode. In this mode, an external crystal or clock source can be used to increment the counter. To set up the timer to wake the device:

- TMR1ON bit of the T1CON register must be set
- TMR1IE bit of the PIE1 register must be set
- PEIE bit of the INTCON register must be set
- $\overline{\text{T1SYNC}}$ bit of the T1CON register must be set
- TMR1CS bits of the T1CON register must be configured

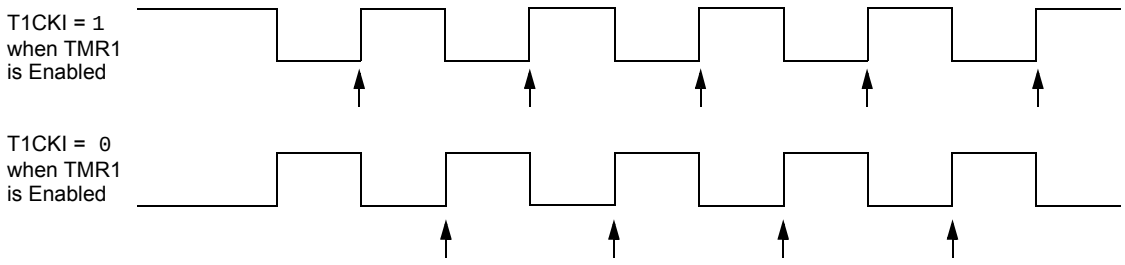
The device will wake-up on an overflow and execute the next instructions. If the GIE bit of the INTCON register is set, the device will call the Interrupt Service Routine.

Timer1 oscillator will continue to operate in Sleep regardless of the $\overline{\text{T1SYNC}}$ bit setting.

19.7.1 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the Alternate Pin Function register, APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see **Section 11.1 "Alternate Pin Function"** for more information.

FIGURE 19-2: TIMER1 INCREMENTING EDGE



Note 1: Arrows indicate counter increments.

2: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.

21.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer, independent of device program execution. The EUSART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or half-duplex synchronous system. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers. Half-Duplex Synchronous mode is intended for communications with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs or other microcontrollers. These devices typically do not have internal clocks for baud rate generation and require the external clock signal provided by a master synchronous device.

The EUSART module includes the following capabilities:

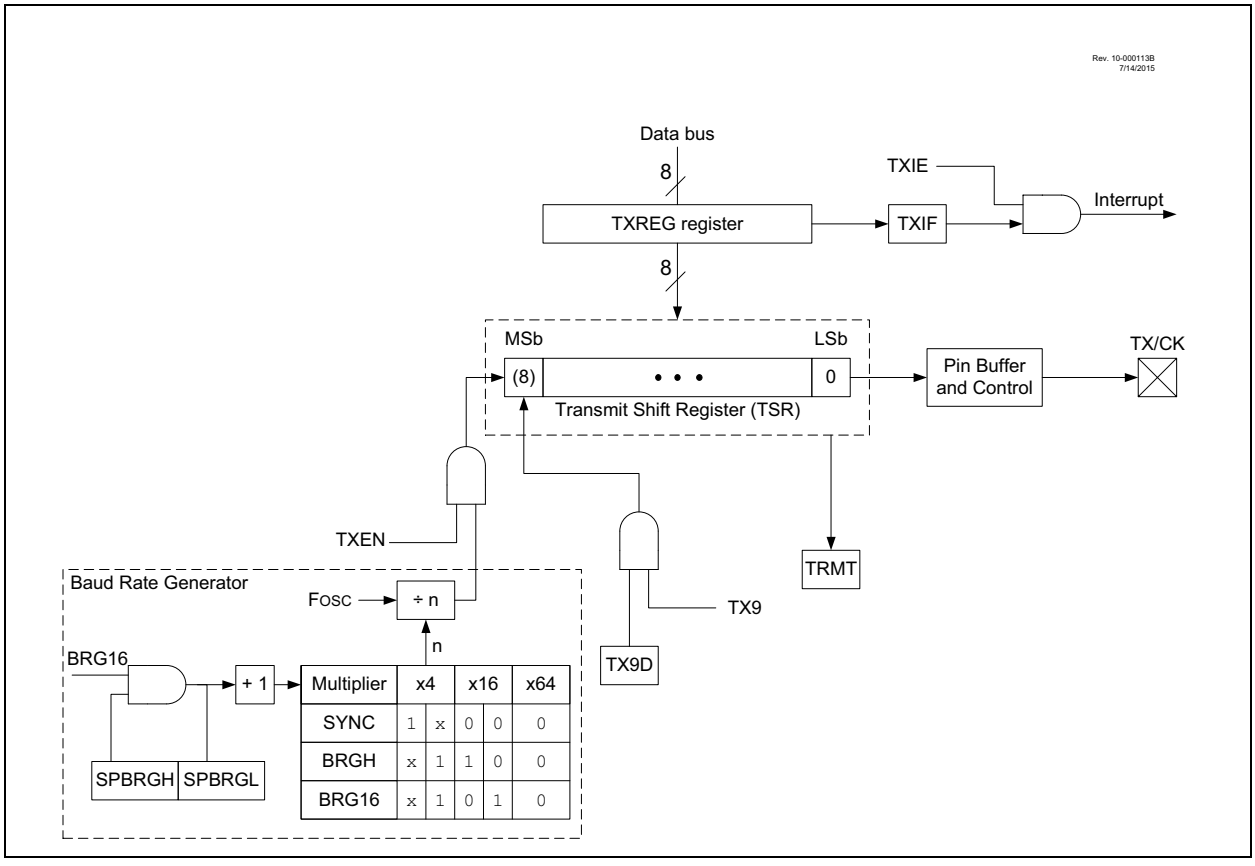
- Full-duplex asynchronous transmit and receive
- Two-character input buffer
- One-character output buffer
- Programmable 8-bit or 9-bit character length
- Address detection in 9-bit mode
- Input buffer overrun error detection
- Received character framing error detection
- Half-duplex synchronous master
- Half-duplex synchronous slave
- Programmable clock polarity in synchronous modes
- Sleep operation

The EUSART module implements the following additional features, making it ideally suited for use in Local Interconnect Network (LIN) bus systems:

- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- 13-bit Break character transmit

Block diagrams of the EUSART transmitter and receiver are shown in Figure 21-1 and Figure 21-2.

FIGURE 21-1: EUSART TRANSMIT BLOCK DIAGRAM



21.1 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard Non-Return-to-Zero (NRZ) format. NRZ is implemented with two levels: a VOH mark state which represents a '1' data bit, and a VOL space state which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port Idles in the mark state. Each character transmission consists of one Start bit, followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is eight bits. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 8-bit/16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See Table 21-5 for examples of baud rate configurations.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

21.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 21-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the TXREG register.

21.1.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

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

Setting the TXEN bit of the TXSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART and automatically configures the TX/CK I/O pin as an output. If the TX/CK pin is shared with an analog peripheral, the analog I/O function must be disabled by clearing the corresponding ANSELx bit.

Note:	The TXIF transmitter interrupt flag is set when the TXEN enable bit is set.

21.1.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXREG until the Stop bit of the previous character has been transmitted. The pending character in the TXREG is then transferred to the TSR in one TCY immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXREG.

21.1.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the SCKP bit of the BAUDCON register. The default state of this bit is '0' which selects high true transmit Idle and data bits. Setting the SCKP bit to '1' will invert the transmit data resulting in low true Idle and data bits. The SCKP bit controls transmit data polarity in Asynchronous mode only. In Synchronous mode, the SCKP bit has a different function. See **Section 21.5.1.2 "Clock Polarity"**.

21.1.1.4 Transmit Interrupt Flag

The TXIF interrupt flag bit of the PIR1 register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXREG. In other words, the TXIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXREG. The TXIF flag bit is not cleared immediately upon writing TXREG. TXIF becomes valid in the second instruction cycle following the write execution. Polling TXIF immediately following the TXREG write will return invalid results. The TXIF bit is read-only, it cannot be set or cleared by software.

The TXIF interrupt can be enabled by setting the TXIE interrupt enable bit of the PIE1 register. However, the TXIF flag bit will be set whenever the TXREG is empty, regardless of the state of the TXIE enable bit.

To use interrupts when transmitting data, set the TXIE bit only when there is more data to send. Clear the TXIE interrupt enable bit upon writing the last character of the transmission to the TXREG.

21.1.2.8 Asynchronous Reception Setup

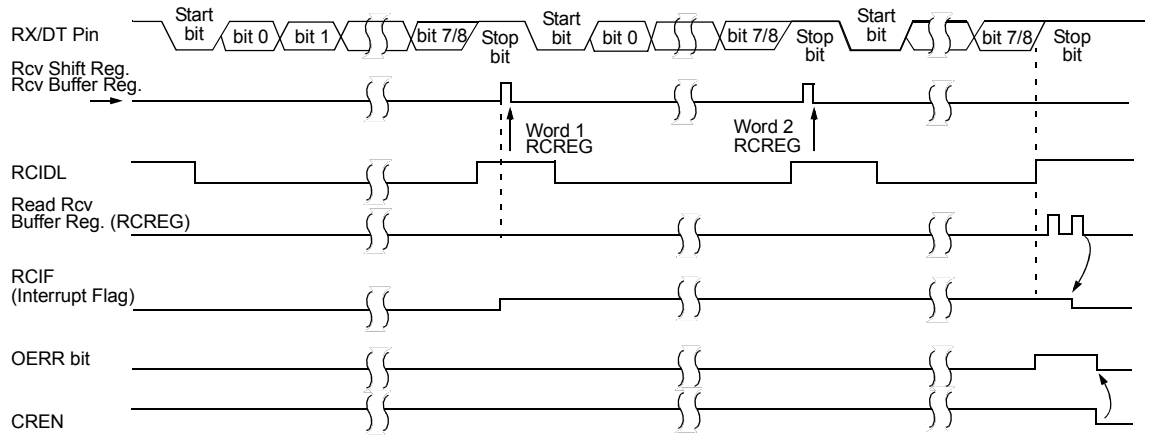
1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 21.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSELx bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register, and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Enable reception by setting the CREN bit.
7. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
8. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received eight Least Significant data bits from the receive buffer by reading the RCREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

21.1.2.9 9-Bit Address Detection Mode Setup

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH/SPBRGL register pair, and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 21.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSELx bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register, and the GIE and PEIE bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Enable reception by setting the CREN bit.
8. The RCIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
9. Read the RCSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCREG register. Software determines if this is the device's address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

FIGURE 21-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

21.4 EUSART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is an 8-bit or 16-bit timer that is dedicated to the support of both the asynchronous and synchronous EUSART operation. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit of the BAUDCON register selects 16-bit mode.

The SPBRGH/SPBRGL register pair determines the period of the free-running baud rate timer. In Asynchronous mode, the multiplier of the baud rate period is determined by both the BRGH bit of the TXSTA register and the BRG16 bit of the BAUDCON register. In Synchronous mode, the BRGH bit is ignored.

Table 21-3 contains the formulas for determining the baud rate. Example 21-1 provides a sample calculation for determining the baud rate and baud rate error.

Typical baud rates and error values for various Asynchronous modes have been computed for your convenience and are shown in Table 21-3. It may be advantageous to use the high baud rate (BRGH = 1) or the 16-bit BRG (BRG16 = 1) to reduce the baud rate error. The 16-bit BRG mode is used to achieve slow baud rates for fast oscillator frequencies.

Writing a new value to the SPBRGH/SPBRGL register pair causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RCIDL bit to make sure that the receive operation is Idle before changing the system clock.

EXAMPLE 21-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$Desired\ Baud\ Rate = \frac{FOSC}{64([SPBRGH:SPBRGL] + 1)}$$

Solving for SPBRGH:SPBRGL:

$$X = \frac{\frac{FOSC}{Desired\ Baud\ Rate}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

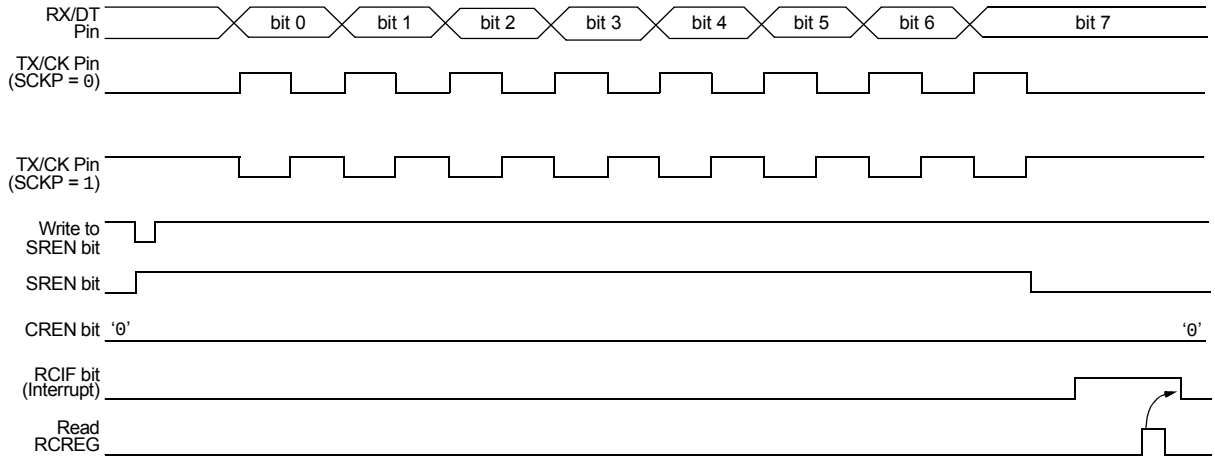
$$Calculated\ Baud\ Rate = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$Error = \frac{Calc.\ Baud\ Rate - Desired\ Baud\ Rate}{Desired\ Baud\ Rate}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

FIGURE 21-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.

TABLE 21-8: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

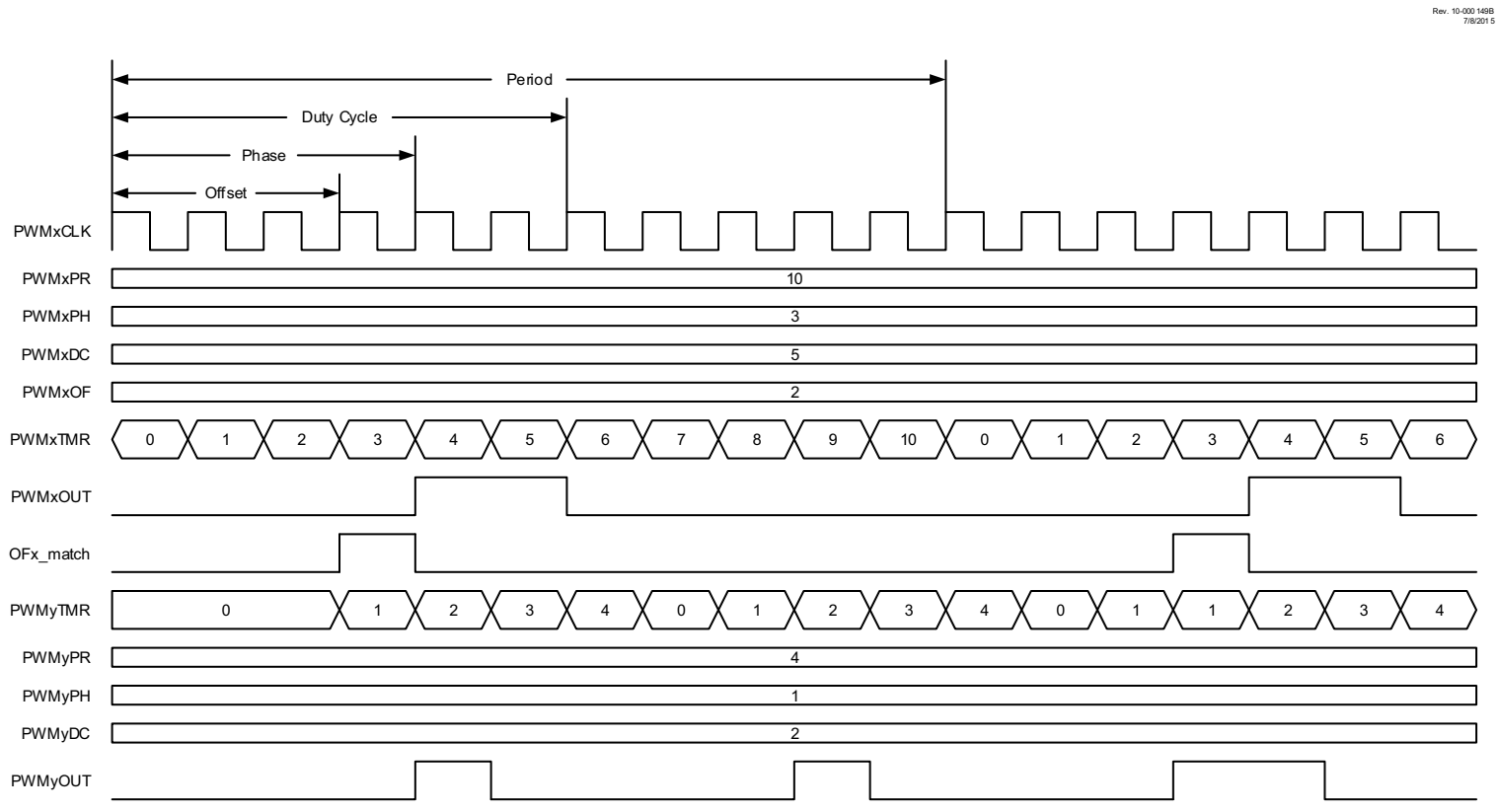
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	186
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	74
PIE1	TMR1GIE	ADIE	RCIE(1)	TXIE(1)	—	—	TMR2IE	TMR1IE	75
PIR1	TMR1GIF	ADIF	RCIF(1)	TXIF(1)	—	—	TMR2IF	TMR1IF	78
RCREG	EUSART Receive Data Register								180*
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	185
SPBRGL	BRG<7:0>								187*
SPBRGH	BRG<15:8>								187*
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	184

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous master reception.

* Page provides register information.

Note 1: PIC12(L)F1572 only.

FIGURE 22-11: CONTINUOUS SLAVE RUN MODE WITH IMMEDIATE RESET AND SYNC START TIMING DIAGRAM



Note: Master= PWMx, Slave=PWMy

TABLE 26-3: POWER-DOWN CURRENTS (I_{PD})(1,2) (CONTINUED)

PIC12LF1571/2		Operating Conditions (unless otherwise stated) Low-Power Sleep Mode						
PIC12F1571/2		Low-Power Sleep Mode, VREGPM = 1						
Param. No.	Device Characteristics	Min.	Typ†	Max. +85°C	Max. +125°C	Units	Conditions	
							V_{DD}	Note
D027		—	4	7	9	µA	1.8	Comparator, CxSP = 0
		—	4.2	8	10	µA	3.0	
D027		—	13	20	21	µA	2.3	Comparator, CxSP = 0
		—	14	23	25	µA	3.0	
		—	16	24	26	µA	5.0	
D028A		—	20	35	36	µA	1.8	Comparator, Normal Power, CxSP = 1 **(Note 1)**
		—	21	36	38	µA	3.0	
D028A		—	28	47	48	µA	2.3	Comparator, Normal Power, CxSP = 1, VREGPM = 1 **(Note 1)**
		—	29	51	52	µA	3.0	
		—	31	52	53	µA	5.0	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, +25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: The peripheral Δ current can be determined by subtracting the base I_{PD} current from this limit. Max. values should be used when calculating total current consumption.

2: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to V_{SS}.

3: ADC clock source is FRC.

FIGURE 27-5: IDD TYPICAL, EC OSCILLATOR, MEDIUM POWER MODE, PIC12LF1571/2 ONLY

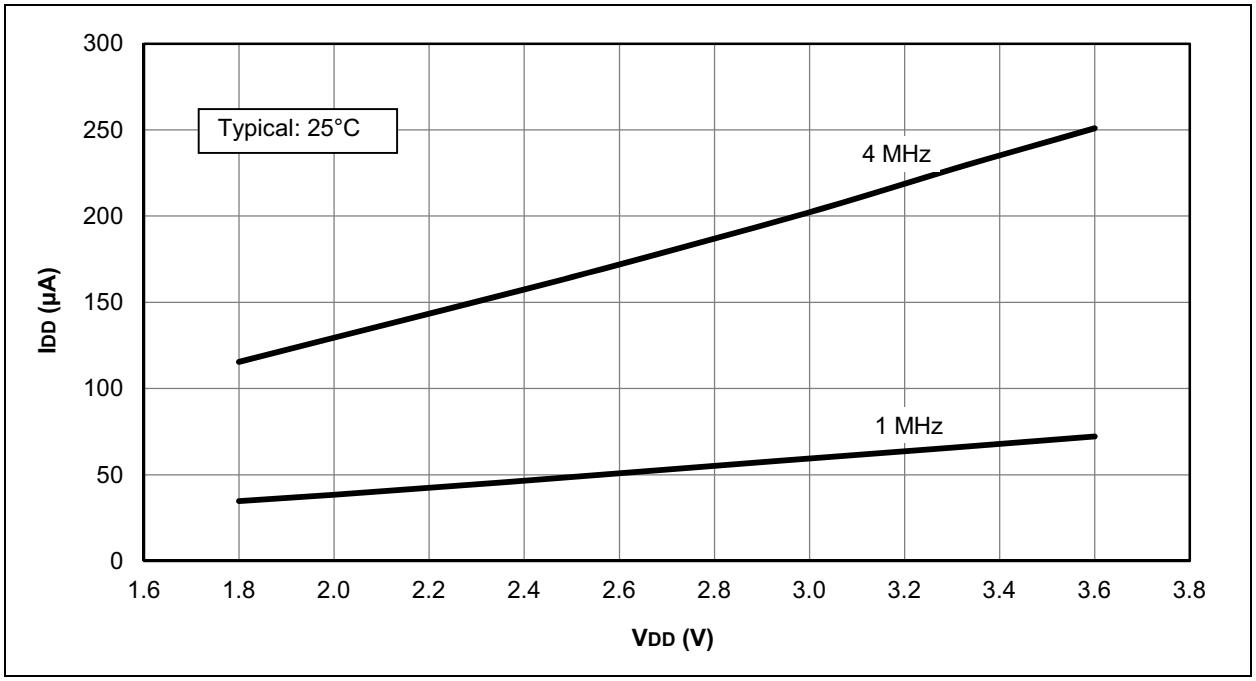


FIGURE 27-6: IDD MAXIMUM, EC OSCILLATOR, MEDIUM POWER MODE, PIC12LF1571/2 ONLY

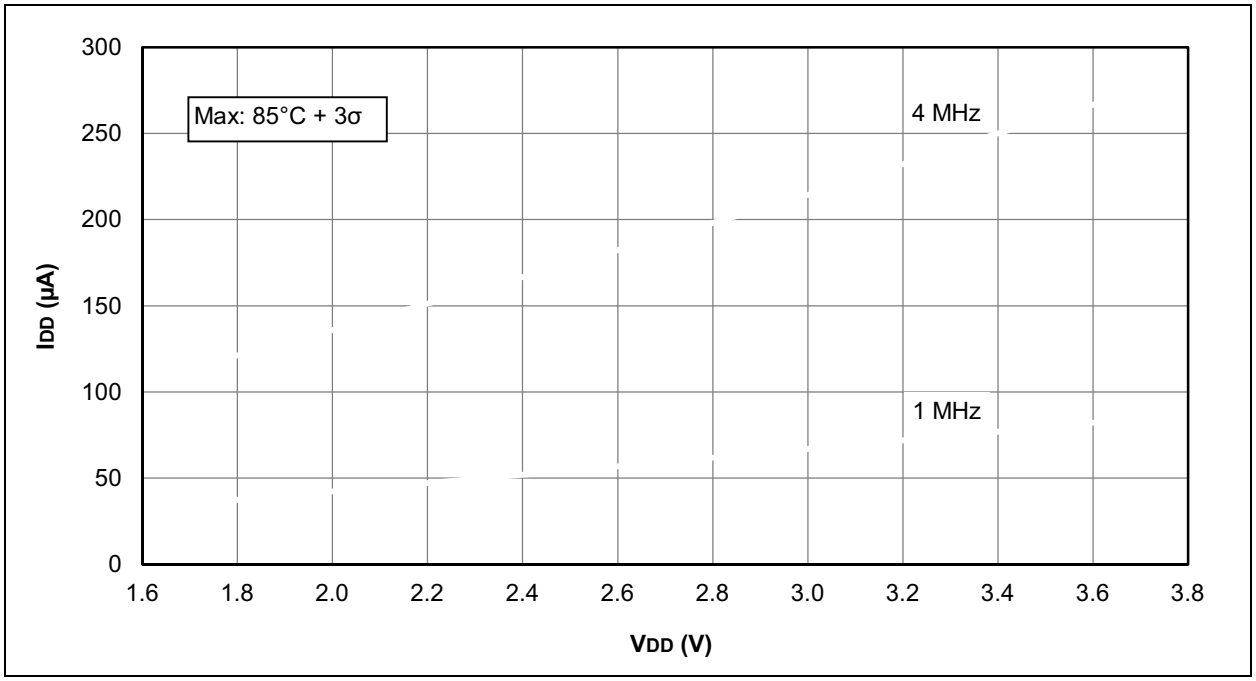


FIGURE 27-25: IPD, FIXED VOLTAGE REFERENCE (FVR), PIC12LF1571/2 ONLY

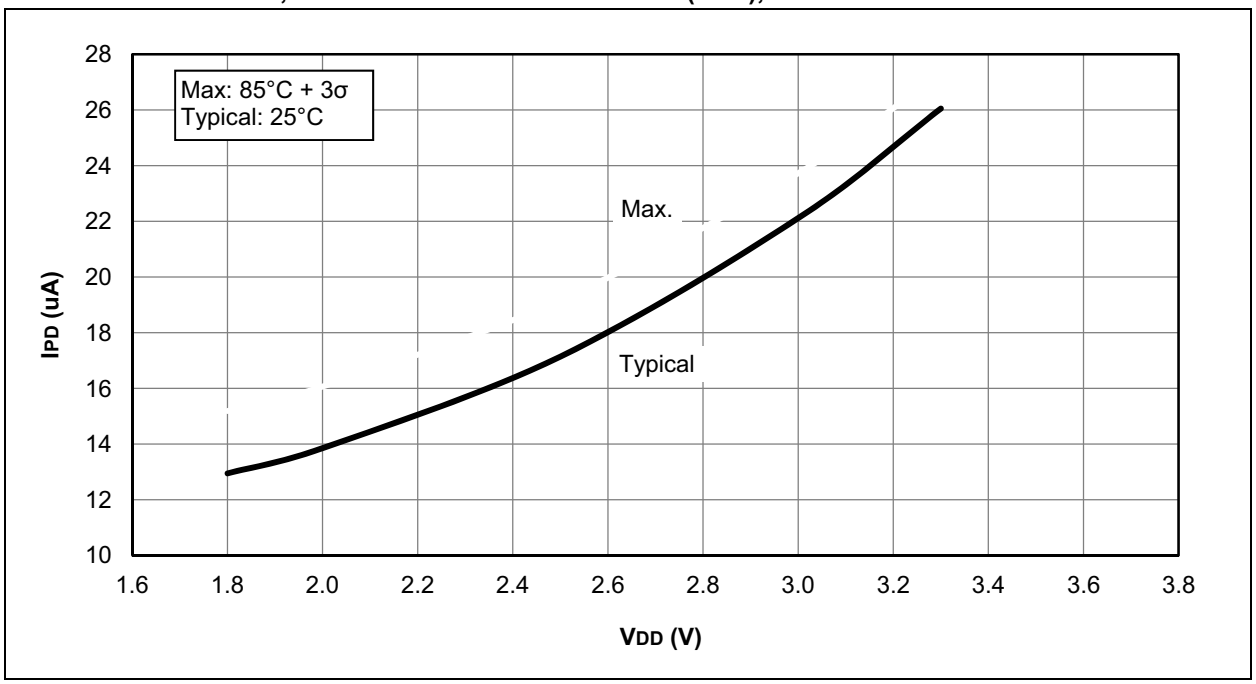
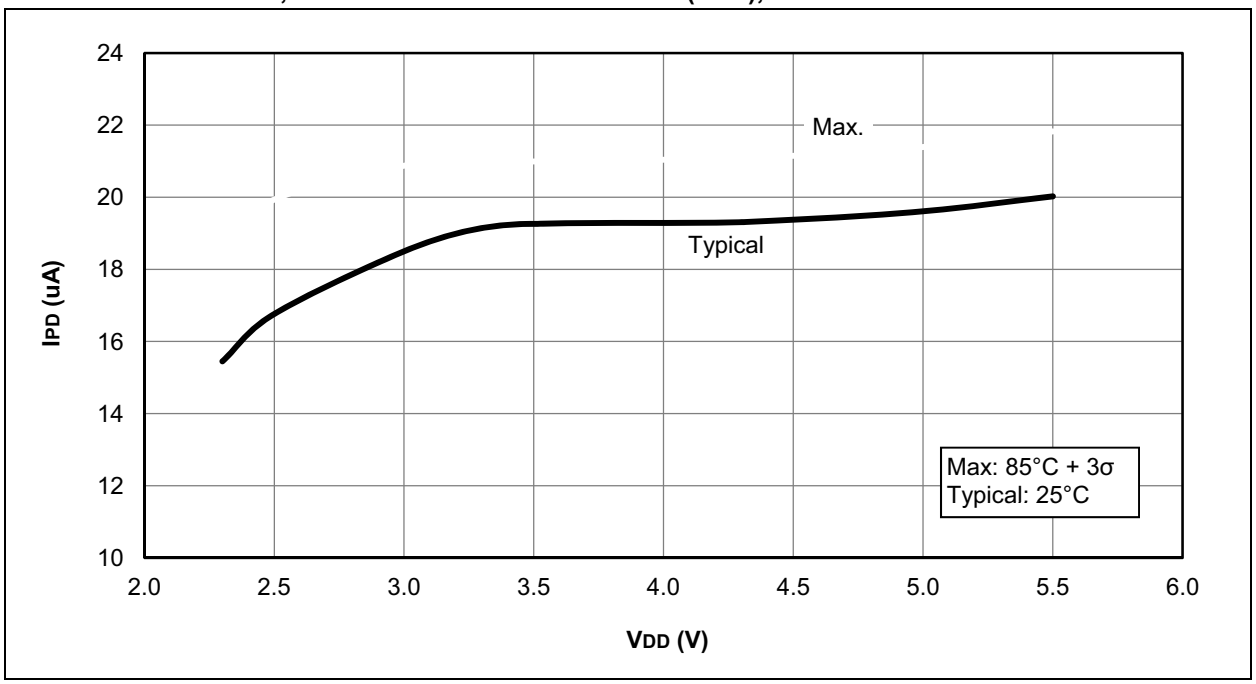


FIGURE 27-26: IPD, FIXED VOLTAGE REFERENCE (FVR), PIC12F1571/2 ONLY



28.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

28.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

28.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

28.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

28.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer 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. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-63277-715-7

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.