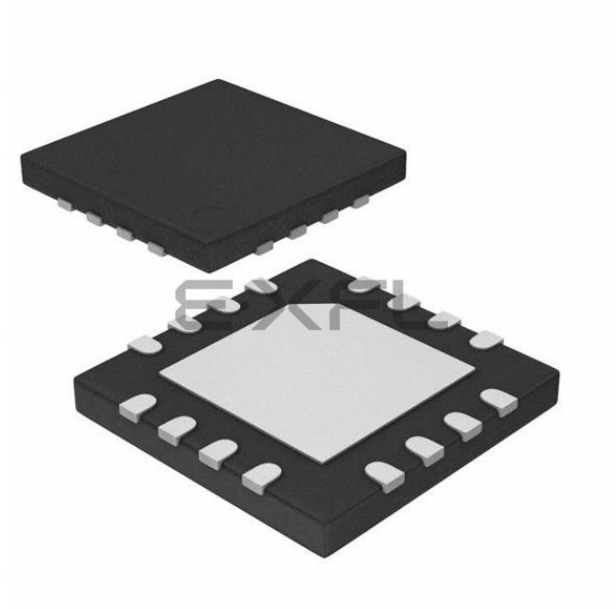


Welcome to E-XFL.COM

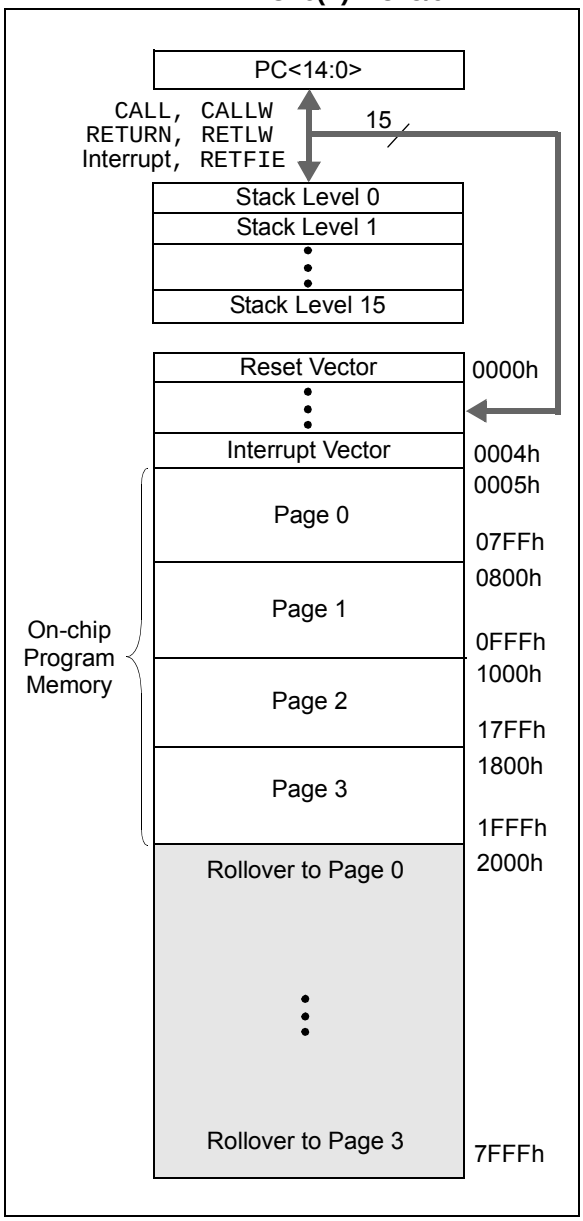
What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	11
Program Memory Size	14KB (8K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	16-UQFN Exposed Pad
Supplier Device Package	16-UQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f1825-i-jq

FIGURE 3-1: PROGRAM MEMORY MAP AND STACK FOR PIC16(L)F1825/9



3.1.1 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.

3.1.1.1 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.

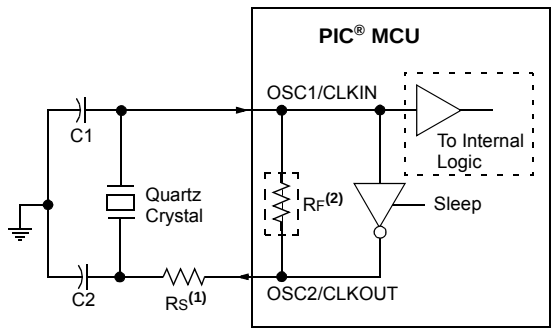
TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	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 6											
300h(1)	INDF0	Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx
301h(1)	INDF1	Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx
302h(1)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	0000 0000
303h(1)	STATUS	—	—	—	$\overline{TO}$	$\overline{PD}$	Z	DC	C	---1 1000	---q quuu
304h(1)	FSR0L	Indirect Data Memory Address 0 Low Pointer								0000 0000	uuuu uuuu
305h(1)	FSR0H	Indirect Data Memory Address 0 High Pointer								0000 0000	0000 0000
306h(1)	FSR1L	Indirect Data Memory Address 1 Low Pointer								0000 0000	uuuu uuuu
307h(1)	FSR1H	Indirect Data Memory Address 1 High Pointer								0000 0000	0000 0000
308h(1)	BSR	—	—	—	BSR<4:0>					---0 0000	---0 0000
309h(1)	WREG	Working Register								0000 0000	uuuu uuuu
30Ah(1)	PCLATH	—	Write Buffer for the upper 7 bits of the Program Counter							-000 0000	-000 0000
30Bh(1)	INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	0000 0000	0000 0000
30Ch	—	Unimplemented								—	—
30Dh	—	Unimplemented								—	—
30Eh	—	Unimplemented								—	—
30Fh	—	Unimplemented								—	—
310h	—	Unimplemented								—	—
311h	CCPR3L	Capture/Compare/PWM Register 3 (LSB)								xxxx xxxx	uuuu uuuu
312h	CCPR3H	Capture/Compare/PWM Register 3 (MSB)								xxxx xxxx	uuuu uuuu
313h	CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	--00 0000	--00 0000
314h	—	Unimplemented								—	—
315h	—	Unimplemented								—	—
316h	—	Unimplemented								—	—
317h	—	Unimplemented								—	—
318h	CCPR4L	Capture/Compare/PWM Register 4 (LSB)								xxxx xxxx	uuuu uuuu
319h	CCPR4H	Capture/Compare/PWM Register 4 (MSB)								xxxx xxxx	uuuu uuuu
31Ah	CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0	--00 0000	--00 0000
31Bh	—	Unimplemented								—	—
31Ch	—	Unimplemented								—	—
31Dh	—	Unimplemented								—	—
31Eh	—	Unimplemented								—	—
31Fh	—	Unimplemented								—	—

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

Note 1: These registers can be addressed from any bank.
2: PIC16(L)F1829 only.
3: PIC16(L)F1825 only.
4: Unimplemented, read as '1'.

FIGURE 5-3: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



Note 1: A series resistor (RS) may be required for quartz crystals with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

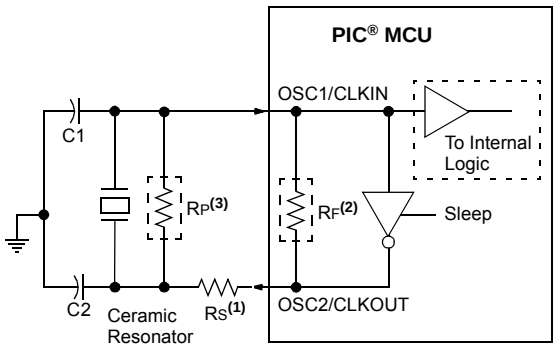
Note 1: Quartz crystal characteristics vary according to type, package and manufacturer. The user should consult the manufacturer data sheets for specifications and recommended application.

2: Always verify oscillator performance over the VDD and temperature range that is expected for the application.

3: For oscillator design assistance, reference the following Microchip Application Notes:

- AN826, "*Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices*" (DS00826)
- AN849, "*Basic PIC® Oscillator Design*" (DS00849)
- AN943, "*Practical PIC® Oscillator Analysis and Design*" (DS00943)
- AN949, "*Making Your Oscillator Work*" (DS00949)

FIGURE 5-4: CERAMIC RESONATOR OPERATION (XT OR HS MODE)



Note 1: A series resistor (RS) may be required for ceramic resonators with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

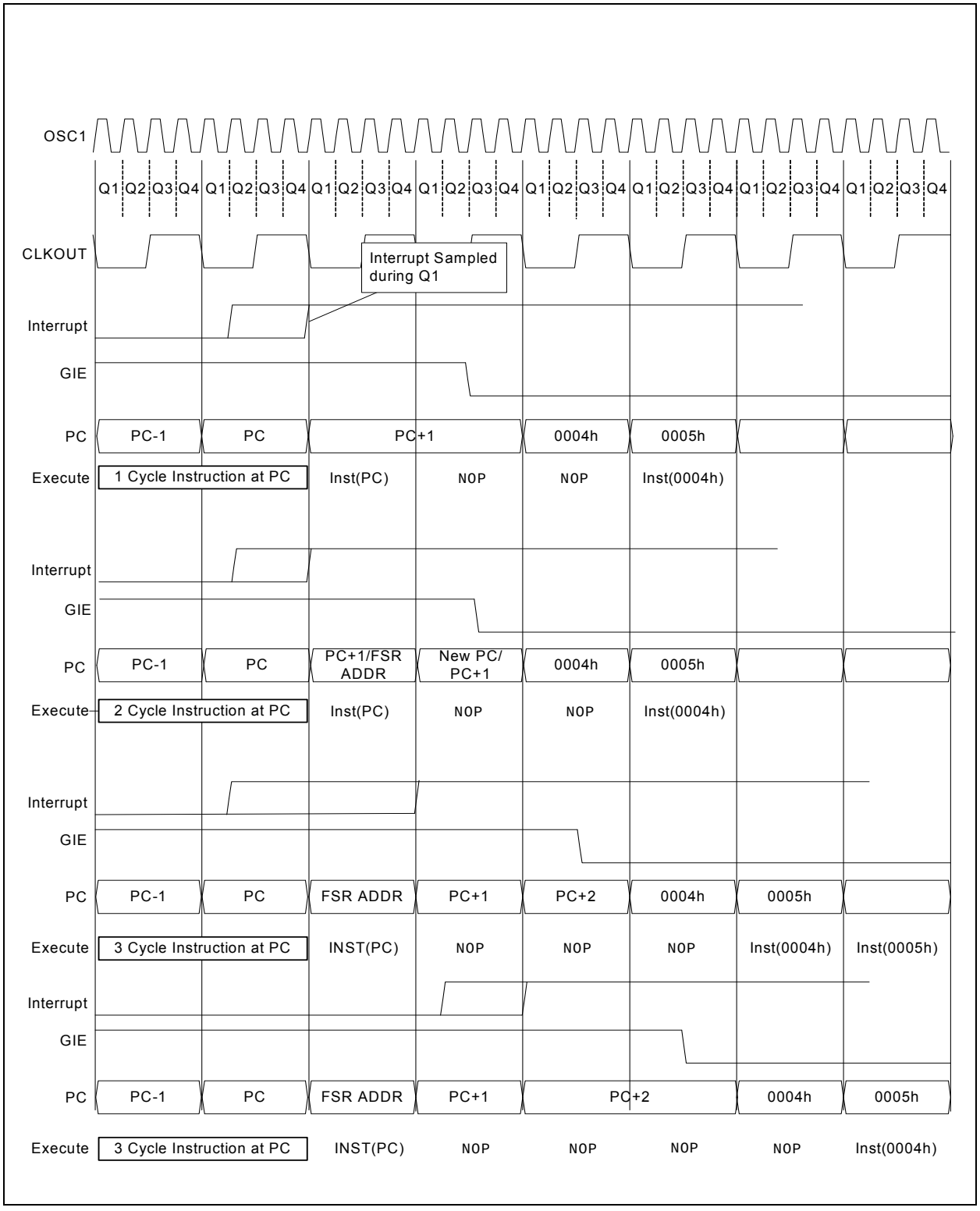
3: An additional parallel feedback resistor (RP) may be required for proper ceramic resonator operation.

5.2.1.3 Oscillator Start-up Timer (OST)

If the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) counts 1024 oscillations from OSC1. This occurs following a Power-on Reset (POR) and when the Power-up Timer (PWRT) has expired (if configured), or a wake-up from Sleep. During this time, the program counter does not increment and program execution is suspended unless either FSCM or Two-Speed Start-up are enabled. In this case, the code will continue to execute at the selected INTOSC frequency while the OST is counting. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the oscillator module.

In order to minimize latency between external oscillator start-up and code execution, the Two-Speed Clock Start-up mode can be selected (see **Section 5.4 "Two-Speed Clock Start-up Mode"**).

FIGURE 8-2: INTERRUPT LATENCY



11.3 Flash Program Memory Overview

It is important to understand the Flash program memory structure for erase and programming operations. Flash program memory is arranged in rows. A row consists of a fixed number of 14-bit program memory words. A row is the minimum block size that can be erased by user software.

Flash program memory may only be written or erased if the destination address is in a segment of memory that is not write-protected, as defined in bits WRT<1:0> of Configuration Word 2.

After a row has been erased, the user can reprogram all or a portion of this row. Data to be written into the program memory row is written to 14-bit wide data write latches. These write latches are not directly accessible to the user, but may be loaded via sequential writes to the EEDATH:EEDATL register pair.

Note:	If the user wants to modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase.

The number of data write latches is not equivalent to the number of row locations. During programming, user software will need to fill the set of write latches and initiate a programming operation multiple times in order to fully reprogram an erased row. For example, a device with a row size of 32 words and eight write latches will need to load the write latches with data and initiate a programming operation four times.

The size of a program memory row and the number of program memory write latches may vary by device. See Table 11-1 for details.

TABLE 11-1: FLASH MEMORY ORGANIZATION BY DEVICE

Device	Erase Block (Row) Size/ Boundary	Number of Write Latches/ Boundary
PIC16(L)F1825 PIC16(L)F1829	32 words, EEADRL<4:0> = 00000	32 words, EEADRL<4:0> = 00000

11.3.1 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

1. Write the Least and Most Significant address bits to the EEADRH:EEADRL register pair.
2. Clear the CFGS bit of the EECON1 register.
3. Set the EEPGD control bit of the EECON1 register.
4. Then, set control bit RD of the EECON1 register.

Once the read control bit is set, the program memory Flash controller will use the second instruction cycle to read the data. This causes the second instruction immediately following the "BSF EECON1,RD" instruction to be ignored. The data is available in the very next cycle, in the EEDATH:EEDATL register pair; therefore, it can be read as two bytes in the following instructions.

EEDATH:EEDATL register pair will hold this value until another read or until it is written to by the user.

Note 1:	The two instructions following a program memory read are required to be NOPs. This prevents the user from executing a two-cycle instruction on the next instruction after the RD bit is set.
2:	Flash program memory can be read regardless of the setting of the $\overline{CP}$ bit.

11.3.2 ERASING FLASH PROGRAM MEMORY

While executing code, program memory can only be erased by rows. To erase a row:

1. Load the EEADRH:EEADRL register pair with the address of new row to be erased.
2. Clear the CFGS bit of the EECON1 register.
3. Set the EEPGD, FREE, and WREN bits of the EECON1 register.
4. Write 55h, then AAh, to EECON2 (Flash programming unlock sequence).
5. Set control bit WR of the EECON1 register to begin the erase operation.
6. Poll the FREE bit in the EECON1 register to determine when the row erase has completed.

See Example 11-4.

After the "BSF EECON1,WR" instruction, the processor requires two cycles to set up the erase operation. The user must place two NOP instructions after the WR bit is set. The processor will halt internal operations for the typical 2 ms erase time. This is not Sleep mode as the clocks and peripherals will continue to run. After the erase cycle, the processor will resume operation with the third instruction after the EECON1 write instruction.

11.3.3 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the starting address of the word(s) to be programmed.
2. Load the write latches with data.
3. Initiate a programming operation.
4. Repeat steps 1 through 3 until all data is written.

Before writing to program memory, the word(s) to be written must be erased or previously unwritten. Program memory can only be erased one row at a time. No automatic erase occurs upon the initiation of the write.

Program memory can be written one or more words at a time. The maximum number of words written at one time is equal to the number of write latches. See Figure 11-2 for more details. The write latches are aligned to the address boundary defined by EEADRL as shown in Table 11-1. Write operations do not cross these boundaries. At the completion of a program memory write operation, the write latches are reset to contain 0x3FFF.

The following steps should be completed to load the write latches and program a block of program memory. These steps are divided into two parts. First, all write latches are loaded with data except for the last program memory location. Then, the last write latch is loaded and the programming sequence is initiated. A special unlock sequence is required to load a write latch with data or initiate a Flash programming operation. This unlock sequence should not be interrupted.

1. Set the EEPGD and WREN bits of the EECON1 register.
2. Clear the CFGS bit of the EECON1 register.
3. Set the LWLO bit of the EECON1 register. When the LWLO bit of the EECON1 register is '1', the write sequence will only load the write latches and will not initiate the write to Flash program memory.
4. Load the EEADRH:EEADRL register pair with the address of the location to be written.
5. Load the EEDATH:EEDATL register pair with the program memory data to be written.
6. Write 55h, then AAh, to EECON2, then set the WR bit of the EECON1 register (Flash programming unlock sequence). The write latch is now loaded.
7. Increment the EEADRH:EEADRL register pair to point to the next location.
8. Repeat steps 5 through 7 until all but the last write latch has been loaded.
9. Clear the LWLO bit of the EECON1 register. When the LWLO bit of the EECON1 register is '0', the write sequence will initiate the write to Flash program memory.
10. Load the EEDATH:EEDATL register pair with the program memory data to be written.
11. Write 55h, then AAh, to EECON2, then set the WR bit of the EECON1 register (Flash programming unlock sequence). The entire latch block is now written to Flash program memory.

It is not necessary to load the entire write latch block with user program data. However, the entire write latch block will be written to program memory.

An example of the complete write sequence for eight words is shown in Example 11-5. The initial address is loaded into the EEADRH:EEADRL register pair; the eight words of data are loaded using indirect addressing.

13.0 INTERRUPT-ON-CHANGE

The PORTA pins can be configured to operate as Interrupt-on-Change (IOC) pins. On the PIC16(L)F1829 devices, the PORTB pins can also be configured to operate as IOC pins. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual port pin, or combination of port pins, can be configured to generate an interrupt. The interrupt-on-change module has the following features:

- Interrupt-on-change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

Figure 13-1 is a block diagram of the IOC module.

13.1 Enabling the Module

To allow individual port pins to generate an interrupt, the IOCIE bit of the INTCON register must be set. If the IOCIE bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

13.2 Individual Pin Configuration

For each port pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated bit of the IOCxP register is set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting both associated bits of the IOCxP and IOCxN registers, respectively.

13.3 Interrupt Flags

The IOCAFx and IOCBFx bits located in the IOCAF and IOCBF registers, respectively, are status flags that correspond to the interrupt-on-change pins of the associated port. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit of the INTCON register reflects the status of all IOCAFx and IOCBFx bits.

13.4 Clearing Interrupt Flags

The individual status flags, (IOCAFx and IOCBFx bits), can be cleared by resetting them to zero. If another edge is detected during this clearing operation, the associated status flag will be set at the end of the sequence, regardless of the value actually being written.

In order to ensure that no detected edge is lost while clearing flags, only AND operations masking out known changed bits should be performed. The following sequence is an example of what should be performed.

EXAMPLE 13-1: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)

```
MOVLW  0xff
XORWF  IOCAF, W
ANDWF  IOCAF, F
```

13.5 Operation in Sleep

The interrupt-on-change interrupt sequence will wake the device from Sleep mode, if the IOCIE bit is set.

If an edge is detected while in Sleep mode, the IOCxF register will be updated prior to the first instruction executed out of Sleep.

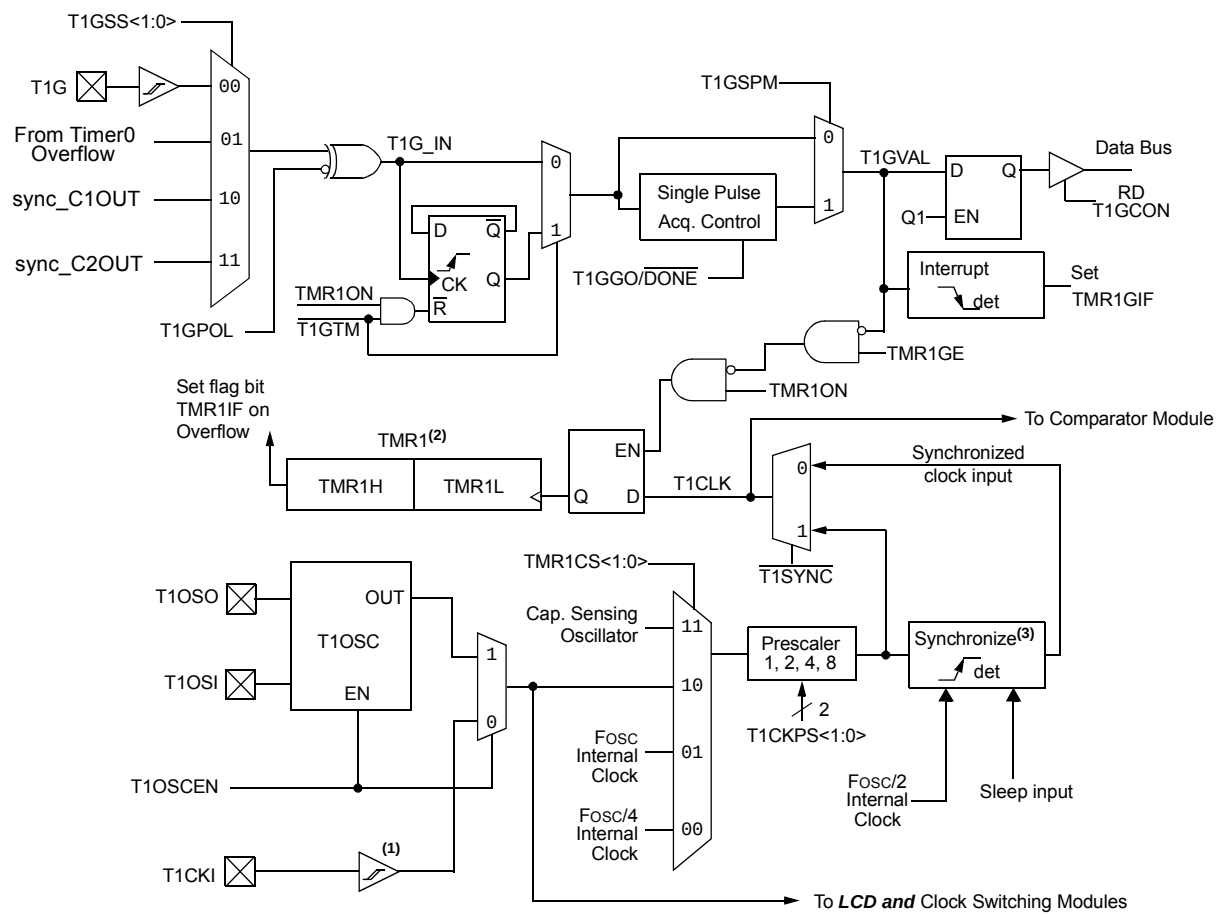
21.0 TIMER1 MODULE WITH GATE CONTROL

The Timer1 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMR1H:TMR1L)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- Time base for the Capture/Compare function
- Special Event Trigger (with CCP/ECCP)
- Selectable Gate Source Polarity
- Gate Toggle Mode
- Gate Single-pulse Mode
- Gate Value Status
- Gate Event Interrupt

Figure 21-1 is a block diagram of the Timer1 module.

FIGURE 21-1: TIMER1 BLOCK DIAGRAM



Note 1: ST Buffer is high speed type when using T1CKI.
2: Timer1 register increments on rising edge.
3: Synchronize does not operate while in Sleep.

21.11 Timer1 Control Register

The Timer1 Control register (T1CON), shown in Register 21-1, is used to control Timer1 and select the various features of the Timer1 module.

REGISTER 21-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	U-0	R/W-0/u
TMR1CS<1:0>		T1CKPS<1:0>		T1OSCEN	$\overline{\text{T1SYNC}}$	—	TMR1ON
bit 7							bit 0

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

bit 7-6 **TMR1CS<1:0>:** Timer1 Clock Source Select bits
- 11 = Timer1 clock source is Capacitive Sensing Oscillator (CAPOSC)
- 10 = Timer1 clock source is pin or oscillator:
  - If T1OSCEN = 0:
  - External clock from T1CKI pin (on the rising edge)
  - If T1OSCEN = 1:
  - Crystal oscillator on T1OSI/T1OSO pins
- 01 = Timer1 clock source is system clock (FOSC)
- 00 = Timer1 clock source is instruction clock (FOSC/4)

bit 5-4 **T1CKPS<1:0>:** Timer1 Input Clock Prescale Select bits
- 11 = 1:8 Prescale value
- 10 = 1:4 Prescale value
- 01 = 1:2 Prescale value
- 00 = 1:1 Prescale value

bit 3 **T1OSCEN:** LP Oscillator Enable Control bit
- 1 = Dedicated Timer1 oscillator circuit enabled
- 0 = Dedicated Timer1 oscillator circuit disabled

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Control bit

TMR1CS<1:0> = 1X:
- 1 = Do not synchronize external clock input
- 0 = Synchronize external clock input with system clock (FOSC)

TMR1CS<1:0> = 0X:
This bit is ignored.

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

bit 0 **TMR1ON:** Timer1 On bit
- 1 = Enables Timer1
- 0 = Stops Timer1
  - Clears Timer1 gate flip-flop

25.5.3 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSPx module configured as an I^2C Slave in 10-bit Addressing mode.

Figure 25-20 is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish I^2C communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with the R/$\overline{W}$ bit clear; UA bit of the SSPxSTAT register is set.
4. Slave sends $\overline{ACK}$ and SSPxIF is set.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. Slave loads low address into SSPxADD, releasing SCLx.
8. Master sends matching low address byte to the slave; UA bit is set.

Note: Updates to the SSPxADD register are not allowed until after the $\overline{ACK}$ sequence.

9. Slave sends $\overline{ACK}$ and SSPxIF is set.

Note: If the low address does not match, SSPxIF and UA are still set so that the slave software can set SSPxADD back to the high address. BF is not set because there is no match. CKP is unaffected.

10. Slave clears SSPxIF.
11. Slave reads the received matching address from SSPxBUF clearing BF.
12. Slave loads high address into SSPxADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{ACK}$ on the ninth SCLx pulse; SSPxIF is set.
14. If SEN bit of SSPxCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPxIF.
16. Slave reads the received byte from SSPxBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCLx.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

25.5.4 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCLx line is held low are the same. Figure 25-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 25-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

FIGURE 25-34: BUS COLLISION DURING START CONDITION (SCLX = 0)

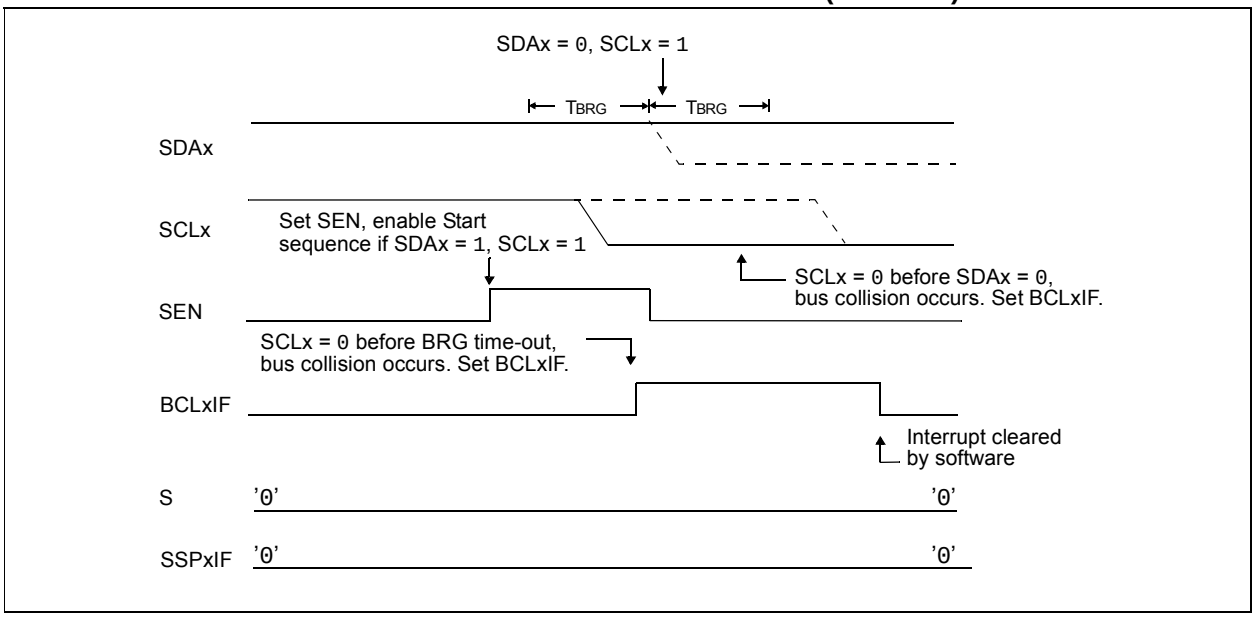
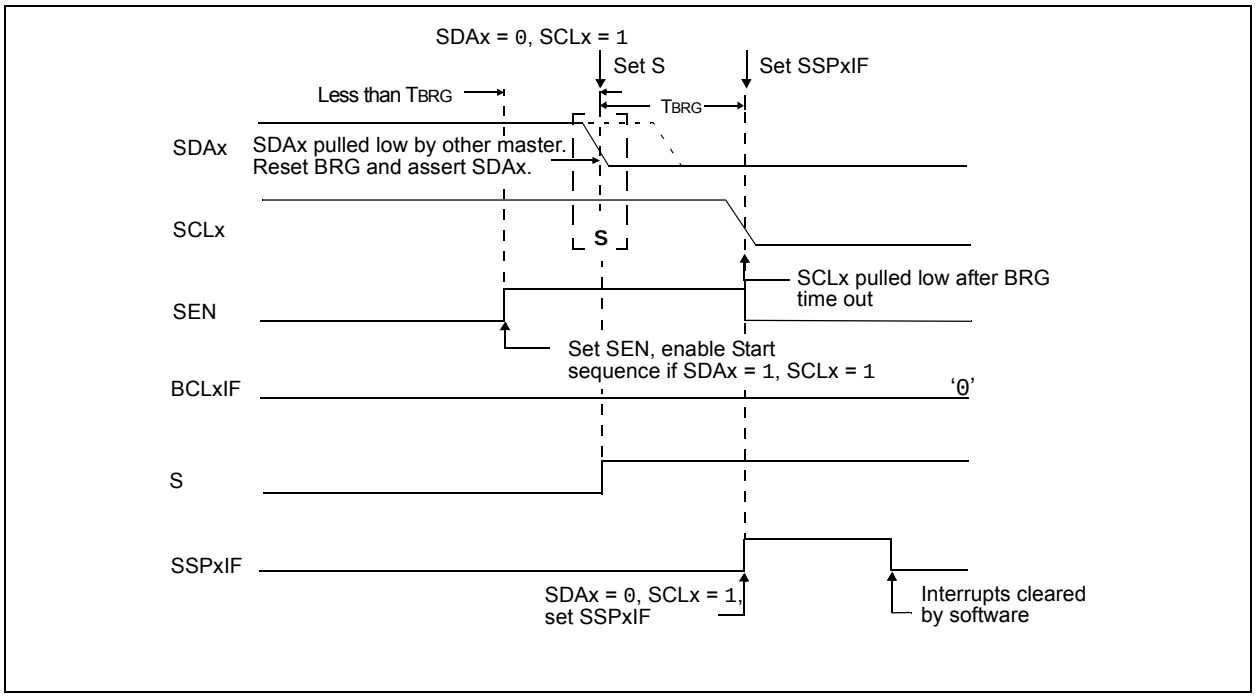


FIGURE 25-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



REGISTER 26-3: BAUDCON: BAUD RATE CONTROL REGISTER

R-0/0	R-1/1	U-0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0
ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

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

bit 7 **ABDOVF:** Auto-Baud Detect Overflow bit
Asynchronous mode:
1 = Auto-baud timer overflowed
0 = Auto-baud timer did not overflow
Synchronous mode:
Don't care

bit 6 **RCIDL**: Receive Idle Flag bit
Asynchronous mode:
1 = Receiver is Idle
0 = Start bit has been received and the receiver is receiving
Synchronous mode:
Don't care

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

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
1 = Transmit inverted data to the TX/CK pin
0 = Transmit non-inverted data to the TX/CK pin
Synchronous mode:
1 = Data is clocked on rising edge of the clock
0 = Data is clocked on falling edge of the clock

bit 3 **BRG16:** 16-bit Baud Rate Generator bit
1 = 16-bit Baud Rate Generator is used
0 = 8-bit Baud Rate Generator is used

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

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = Receiver is waiting for a falling edge. No character will be received, byte RCIF will be set. WUE will automatically clear after RCIF is set.
0 = Receiver is operating normally
Synchronous mode:
Don't care

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)
0 = Auto-Baud Detect mode is disabled
Synchronous mode:
Don't care

FIGURE 26-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION

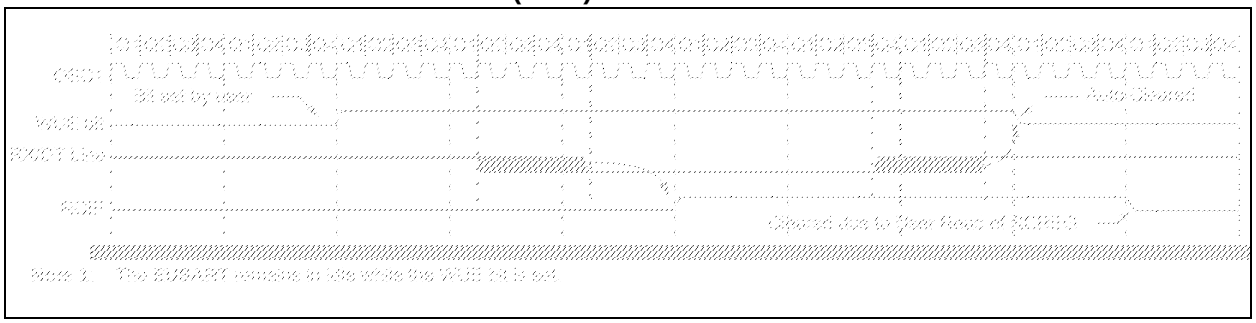
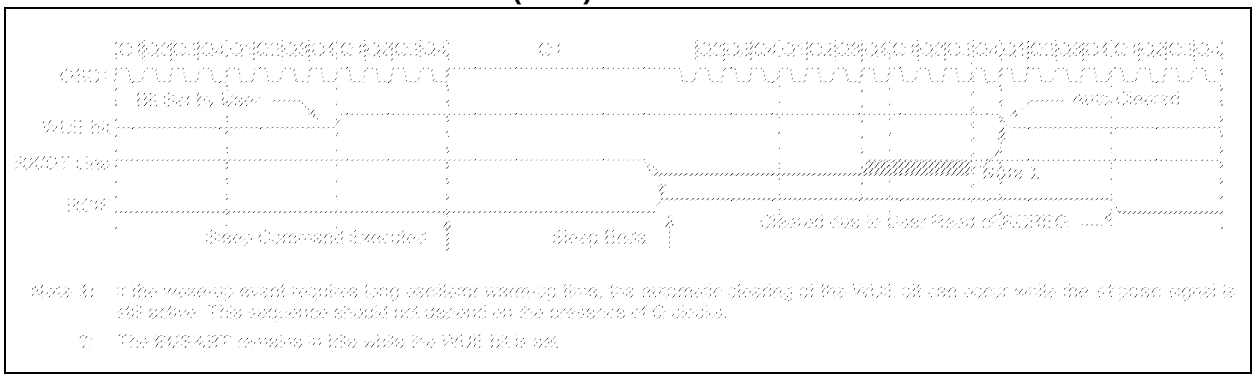


FIGURE 26-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



27.5 Timer Resources

To measure the change in frequency of the capacitive sensing oscillator, a fixed time base is required. For the period of the fixed time base, the capacitive sensing oscillator is used to clock either Timer0 or Timer1. The frequency of the capacitive sensing oscillator is equal to the number of counts in the timer divided by the period of the fixed time base.

27.6 Fixed Time Base

To measure the frequency of the capacitive sensing oscillator, a fixed time base is required. Any timer resource or software loop can be used to establish the fixed time base. It is up to the end user to determine the method in which the fixed time base is generated.

Note:	The fixed time base can not be generated by the timer resource that the capacitive sensing oscillator is clocking.

27.6.1 TIMER0

To select Timer0 as the timer resource for the CPS module:

- Set the T0XCS bit of the CPSCON0 register.
- Clear the TMR0CS bit of the OPTION_REG register.

When Timer0 is chosen as the timer resource, the capacitive sensing oscillator will be the clock source for Timer0. Refer to **Section 20.0 “Timer0 Module”** for additional information.

27.6.2 TIMER1

To select Timer1 as the timer resource for the CPS module, set the TMR1CS<1:0> of the T1CON register to ‘11’. When Timer1 is chosen as the timer resource, the capacitive sensing oscillator will be the clock source for Timer1. Because the Timer1 module has a gate control, developing a time base for the frequency measurement can be simplified by using the Timer0 overflow flag.

It is recommend that the Timer0 overflow flag, in conjunction with the Toggle mode of the Timer1 Gate, be used to develop the fixed time base required by the software portion of the CPS module. Refer to **Section 21.12 “Timer1 Gate Control Register”** for additional information.

TABLE 27-2: TIMER1 ENABLE FUNCTION

TMR1ON	TMR1GE	Timer1 Operation
0	0	Off
0	1	Off
1	0	On
1	1	Count Enabled by input

27.7 Software Control

The software portion of the CPS module is required to determine the change in frequency of the capacitive sensing oscillator. This is accomplished by the following:

- Setting a fixed time base to acquire counts on Timer0 or Timer1.
- Establishing the nominal frequency for the capacitive sensing oscillator.
- Establishing the reduced frequency for the capacitive sensing oscillator due to an additional capacitive load.
- Set the frequency threshold.

27.7.1 NOMINAL FREQUENCY (NO CAPACITIVE LOAD)

To determine the nominal frequency of the capacitive sensing oscillator:

- Remove any extra capacitive load on the selected CPSx pin.
- At the start of the fixed time base, clear the timer resource.
- At the end of the fixed time base save the value in the timer resource.

The value of the timer resource is the number of oscillations of the capacitive sensing oscillator for the given time base. The frequency of the capacitive sensing oscillator is equal to the number of counts on in the timer divided by the period of the fixed time base.

27.7.2 REDUCED FREQUENCY (ADDITIONAL CAPACITIVE LOAD)

The extra capacitive load will cause the frequency of the capacitive sensing oscillator to decrease. To determine the reduced frequency of the capacitive sensing oscillator:

- Add a typical capacitive load on the selected CPSx pin.
- Use the same fixed time base as the nominal frequency measurement.
- At the start of the fixed time base, clear the timer resource.
- At the end of the fixed time base save the value in the timer resource.

The value of the timer resource is the number of oscillations of the capacitive sensing oscillator with an additional capacitive load. The frequency of the capacitive sensing oscillator is equal to the number of counts on in the timer divided by the period of the fixed time base. This frequency should be less than the value obtained during the nominal frequency measurement.

TABLE 30-19: DC CHARACTERISTICS FOR PIC16F1825/9-H (High Temp.)

PIC16F1825/9			Standard Operating Conditions: (unless otherwise stated) Operating Temperature: -40°C ≤ TA ≤ +150°C for High Temperature				
Param No.	**Sym.**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Condition**
D001	VDD	Supply Voltage	2.5	—	5.5	V	FOSC ≤ 32 MHz **(Note 2)**
D002*	VDR	RAM Data Retention Voltage(1)	2.1	—	5.5	V	Device in Sleep mode
D003	VADFVR	Fixed Voltage Reference Voltage for ADC	-10	—	8	%	1.024V, VDD ≥ 2.5V 2.048V, VDD ≥ 2.5V 4.096V, VDD ≥ 4.75V
D003A	VCDAFVR	Fixed Voltage Reference Voltage for ADC	-13	—	9	%	1.024V, VDD ≥ 2.5V 2.048V, VDD ≥ 2.5V 4.096V, VDD ≥ 4.75V

* 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: This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

2: PLL required for 32 MHz operation.

31.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

The graphs and tables provided in this section are for **design guidance** and are **not tested**.

In some graphs or tables, the data presented are **outside specified operating range** (i.e., outside specified V_{DD} range). This is for **information only** and devices are ensured to operate properly only within the specified range.

Note:	The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "MAXIMUM", "Max.", "MINIMUM" or "Min." represents (mean + 3σ) or (mean - 3σ) respectively, where σ is a standard deviation, over each temperature range.

FIGURE 31-3: IDD TYPICAL, XT AND EXTRC OSCILLATOR, PIC16LF1825/9 ONLY

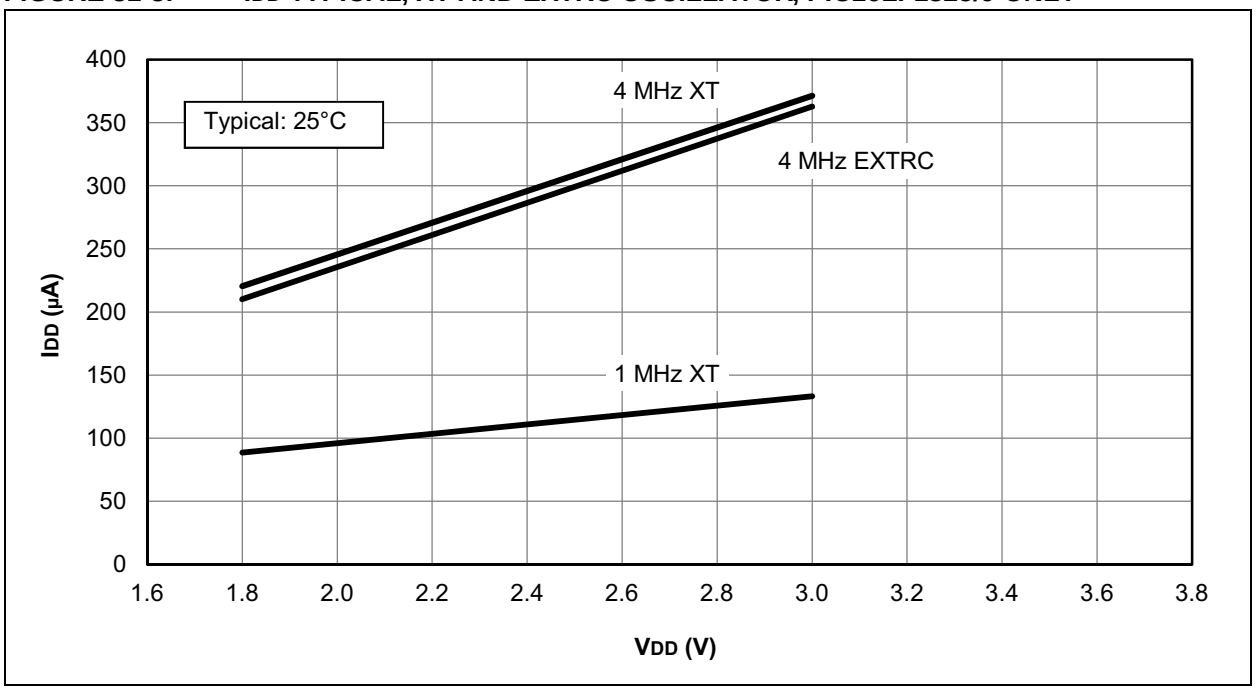


FIGURE 31-4: IDD MAXIMUM, XT AND EXTRC OSCILLATOR, PIC16LF1825/9 ONLY

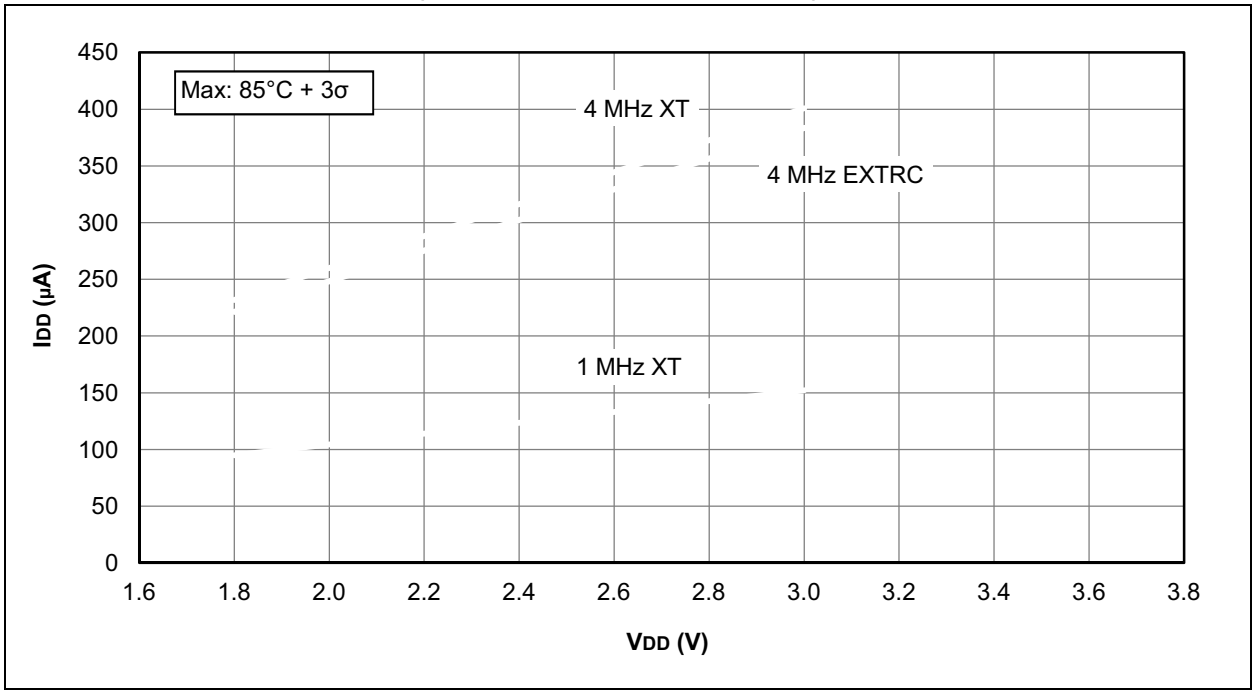


FIGURE 31-17: IDD TYPICAL, HFINTOSC MODE, PIC16F1825/9 ONLY

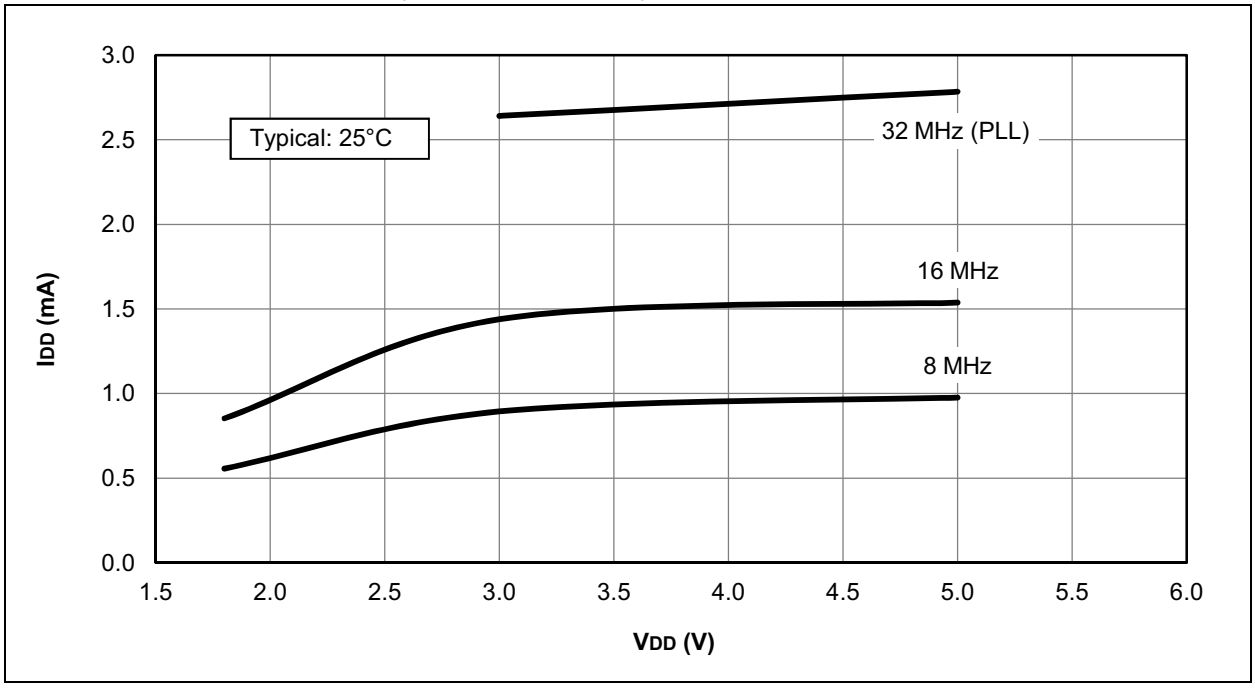


FIGURE 31-18: IDD MAXIMUM, HFINTOSC MODE, PIC16F1825/9 ONLY

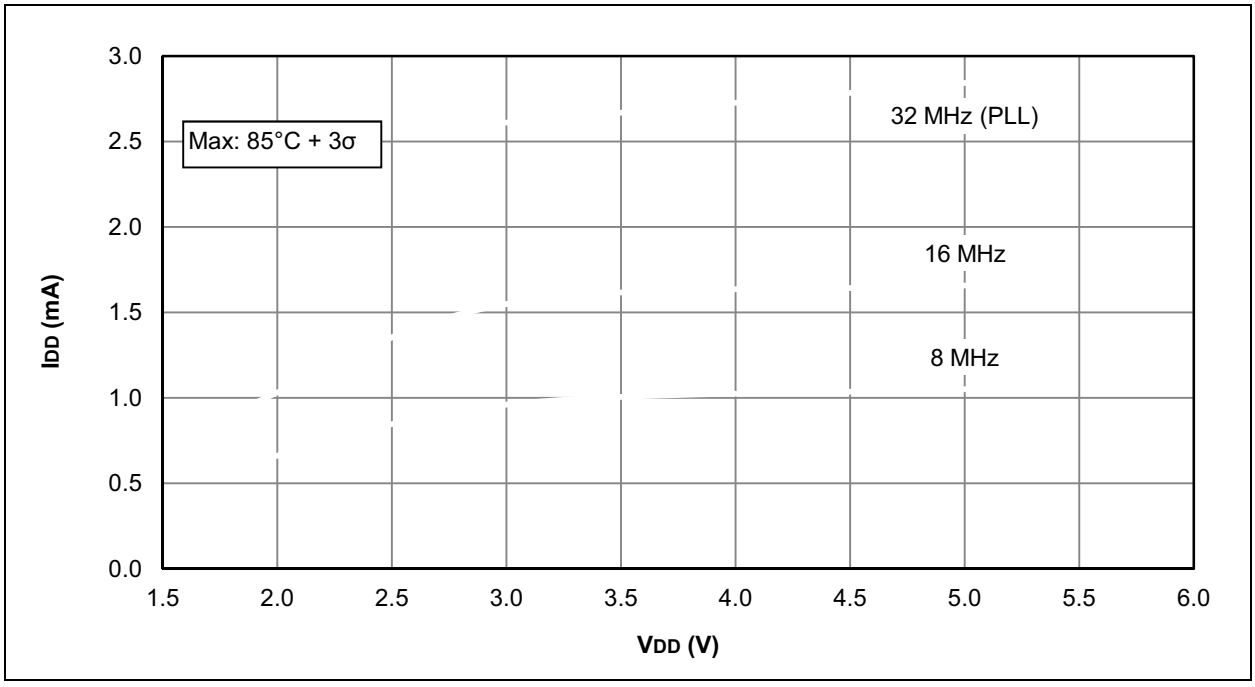


FIGURE 31-52: WDT TIME-OUT PERIOD

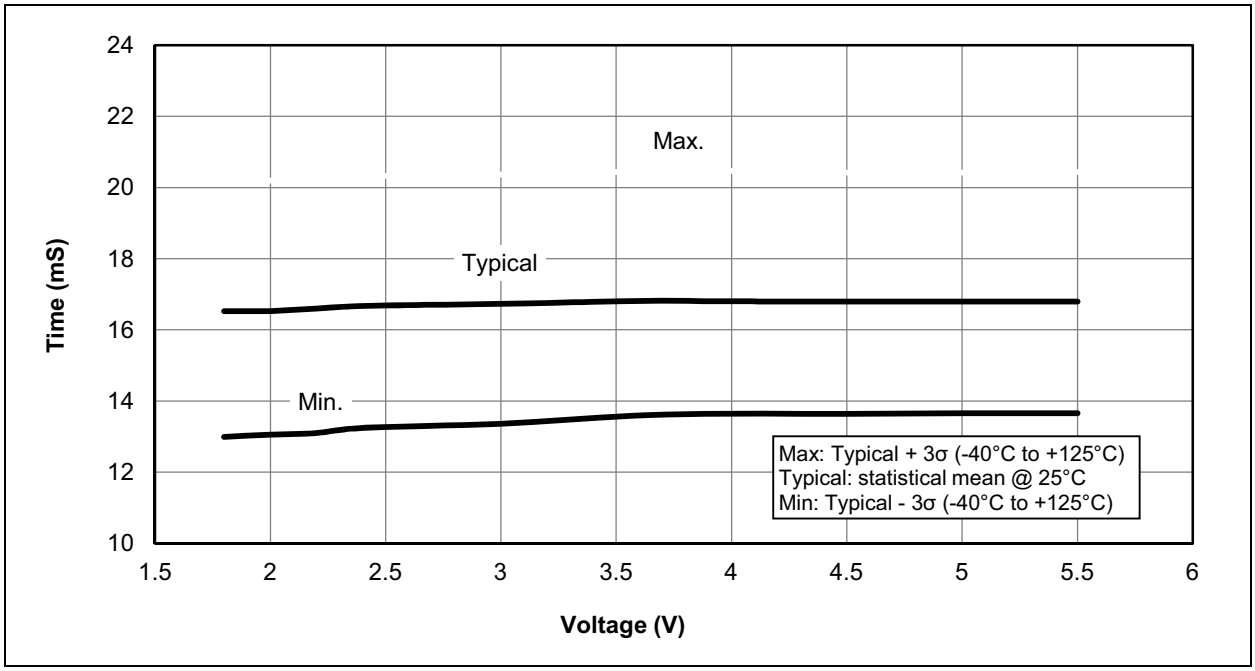
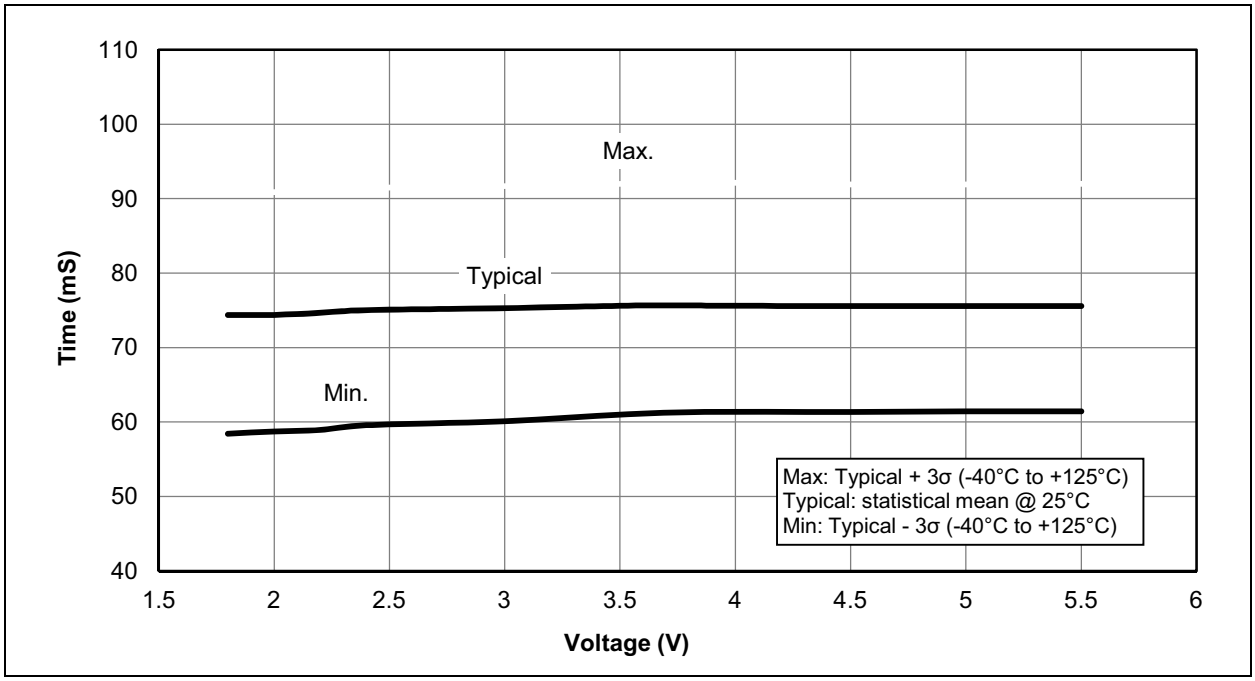
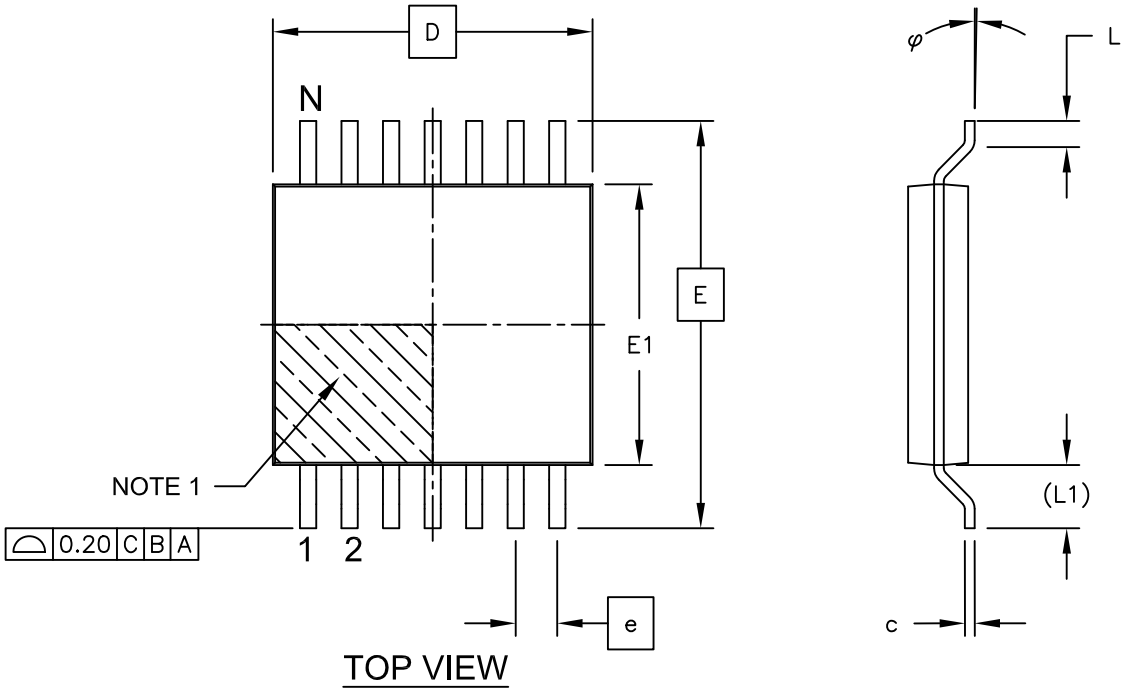


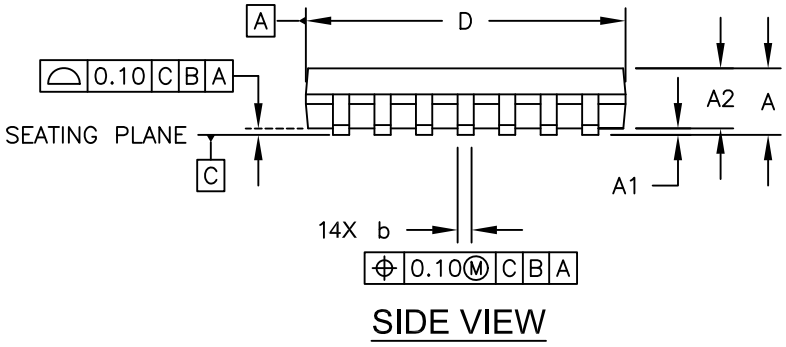
FIGURE 31-53: PWRT PERIOD



14-Lead Plastic Thin Shrink Small Outline (ST) - 4.4 mm Body [TSSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging





Microchip Technology Drawing C04-087C Sheet 1 of 2