Product Change Notification / SYST-06RVUU814

Date:

11-Nov-2020

Product Category:

8-bit Microcontrollers

PCN Type:

Document Change

Notification Subject:

Data Sheet - PIC18(L)F26/27/45/46/47/55/56/57K42 Data Sheet

Affected CPNs:

SYST-06RVUU814_Affected_CPN_11112020.pdf
SYST-06RVUU814_Affected_CPN_11112020.csv

Notification Text:

SYST-06RVUU814

Microchip has released a new Product Documents for the PIC18(L)F26/27/45/46/47/55/56/57K42 Data Sheet of devices. If you are using one of these devices please read the document located at PIC18(L)F26/27/45/46/47/55/56/57K42 Data Sheet.

Notification Status: Final

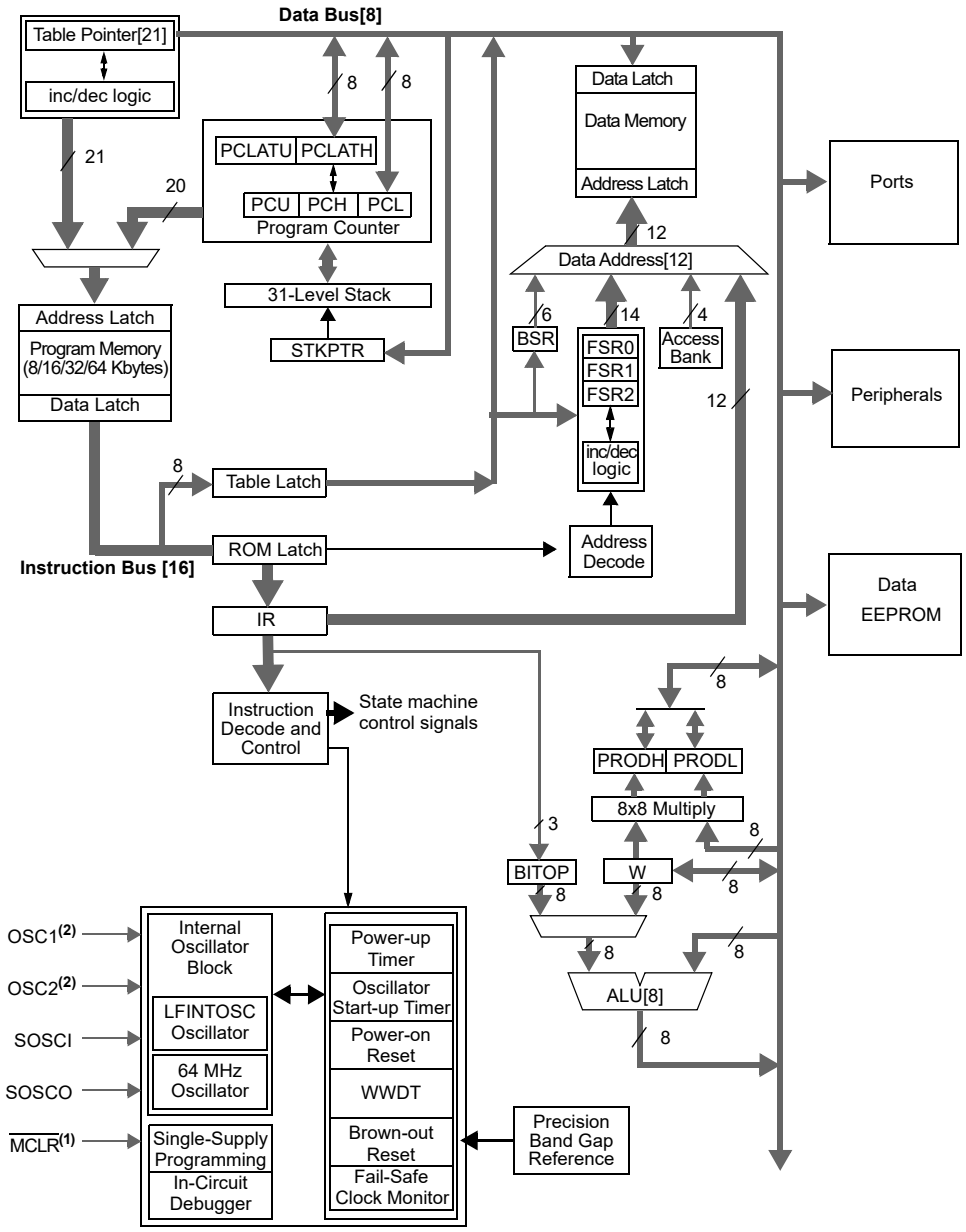
Description of Change:
1) Minor updates to Analog Peripheral and Flexible Oscillator Structure sections in Description chapter.

2) Updated Tables 4-10, 17-2, 36-1, 36-3, 44-15, 44-24 and 44-25.

3) Added Figures 45-90, 45-94, and 45-100 through 45-103 to Characteristics Graphs chapter.

4) Other minor corrections
Impacts to Data Sheet: None
Reason for Change: To Improve Manufacturability
Change Implementation Status: Complete

FIGURE 3-1: PIC18(L)F26/27/45/46/47/55/56/57K42 FAMILY BLOCK DIAGRAM



Note 1: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to Section 7.0, Oscillator Module (with Fail-Safe Clock Monitor) for additional information.

3.1 System Arbitration

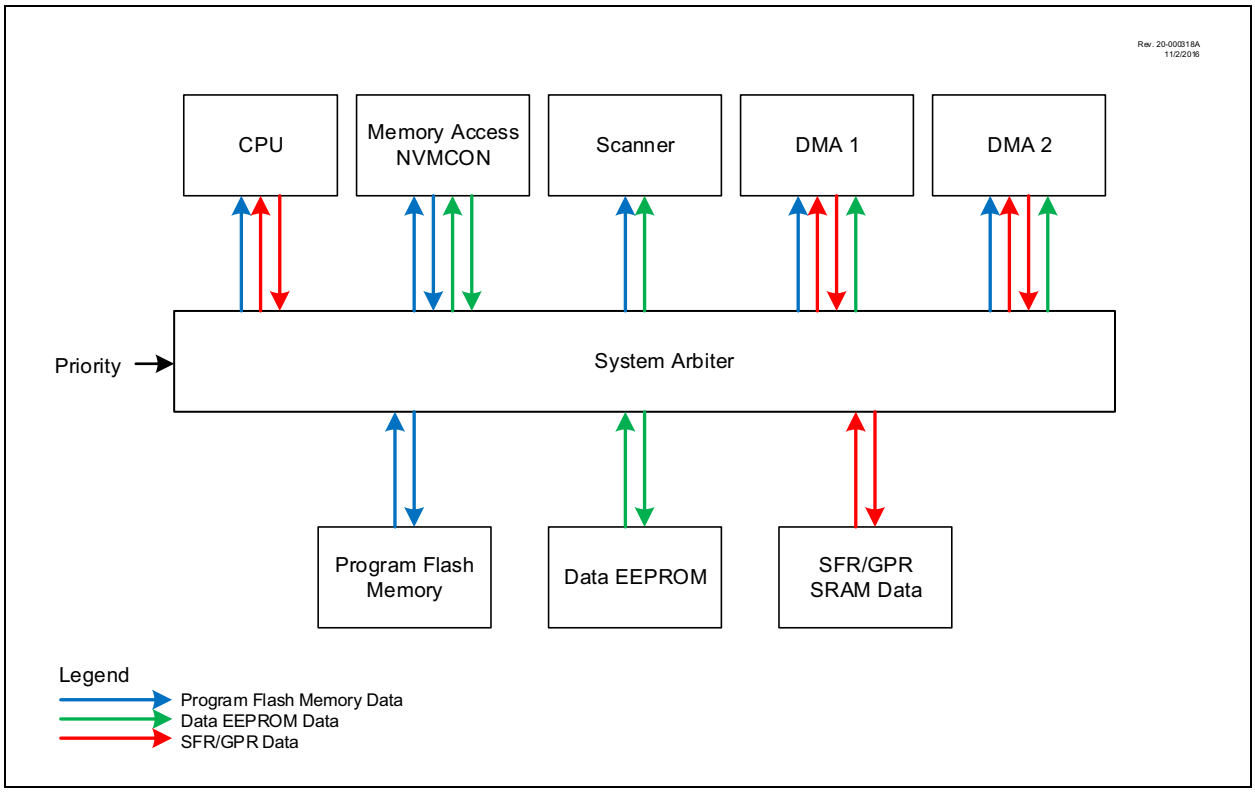
The System Arbiter resolves memory access between the System Level Selections (i.e., Main, Interrupt Service Routine) and Peripheral Selection (i.e., DMA and Scanner) based on user-assigned priorities. Each of the system level and peripheral selections has its own priority selection registers. Memory access priority is resolved using the number written to the corresponding Priority registers, 0 being the highest priority and 4 the lowest. The default priorities are listed in Table 3-1.

In case the user wants to change priorities, ensure each Priority register is written with a unique value from 0 to 4.

TABLE 3-1: DEFAULT PRIORITIES

Selection		Priority register Reset value
System Level	ISR	0
	MAIN	1
Peripheral	DMA1	2
	DMA2	3
	SCANNER	4

FIGURE 3-2: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



3.1.1 PRIORITY LOCK

The System arbiter grants memory access to the peripheral selections (DMAx, Scanner) when the PRLOCKED bit (PRLOCK Register) is set.

Priority selections are locked by setting the PRLOCKED bit of the PRLOCK register. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. Examples of setting and clearing the PRLOCKED bit are shown in Example 3-1 and Example 3-2.

EXAMPLE 3-1: PRIORITY LOCK SEQUENCE

```
; Disable interrupts
BCF INTCON0,GIE

; Bank to PRLOCK register
BANKSEL PRLOCK
MOVLW 55h

; Required sequence, next 4
instructions
MOVWF PRLOCK
MOVLW AAh
MOVWF PRLOCK
; Set PRLOCKED bit to grant memory
access to peripherals
BSF PRLOCK,0

; Enable Interrupts
BSF INTCON0,GIE
```

EXAMPLE 3-2: PRIORITY UNLOCK SEQUENCE

```
; Disable interrupts
BCF INTCON0,GIE

; Bank to PRLOCK register
BANKSEL PRLOCK
MOVLW 55h

; Required sequence, next 4
instructions
MOVWF PRLOCK
MOVLW AAh
MOVWF PRLOCK
; Clear PRLOCKED bit to allow changing
priority settings
BCF PRLOCK,0

; Enable Interrupts
BSF INTCON0,GIE
```

3.2 Memory Access Scheme

The user can assign priorities to both system level and peripheral selections based on which the system arbiter grants memory access. Let us consider the following priority scenarios between ISR, MAIN, and Peripherals.

Note: It is always required that the ISR priority be higher than Main priority.

3.2.1 ISR PRIORITY > MAIN PRIORITY > PERIPHERAL PRIORITY

When the Peripheral Priority (DMAx, Scanner) is lower than ISR and MAIN Priority, and the peripheral requires:

1. Access to the Program Flash Memory, then the peripheral waits for an instruction cycle in which the CPU does not need to access the PFM (such as a branch instruction) and uses that cycle to do its own Program Flash Memory access, unless a PFM Read/Write operation is in progress.
2. Access to the SFR/GPR, then the peripheral waits for an instruction cycle in which the CPU does not need to access the SFR/GPR (such as `MOVLW`, `CALL`, `NOP`) and uses that cycle to do its own SFR/GPR access.
3. Access to the Data EEPROM, then the peripheral has access to Data EEPROM unless a Data EEPROM Read/Write operation is being performed.

This results in the lowest throughput for the peripheral to access the memory, and does so without any impact on execution times.

3.2.2 PERIPHERAL PRIORITY > ISR PRIORITY > MAIN PRIORITY

When the Peripheral Priority (DMAx, Scanner) is higher than ISR and MAIN Priority, the CPU operation is stalled when the peripheral requests memory.

The CPU is held in its current state until the peripheral completes its operation. Since the peripheral requests access to the bus, the peripheral cannot be disabled until it completes its operation.

This results in the highest throughput for the peripheral to access the memory, but has the cost of stalling other execution while it occurs.

3.2.3 ISR PRIORITY > PERIPHERAL PRIORITY > MAIN PRIORITY

In this case, interrupt routines and peripheral operation (DMAx, Scanner) will stall the CPU. Interrupt will preempt peripheral operation. This results in lowest interrupt latency and highest throughput for the peripheral to access the memory.

3.2.4 PERIPHERAL 1 PRIORITY > ISR PRIORITY > MAIN PRIORITY > PERIPHERAL 2 PRIORITY

In this case, the Peripheral 1 will stall the execution of the CPU. However, Peripheral 2 can access the memory in cycles unused by Peripheral 1.

The operation of the System Arbiter is controlled through the following registers:

REGISTER 3-1: ISRPR: INTERRUPT SERVICE ROUTINE PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	ISRPR[2:0]		
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	HS = Hardware set

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **ISRPR[2:0]**: Interrupt Service Routine Priority Selection bits

REGISTER 3-2: MAINPR: MAIN ROUTINE PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-1/1
—	—	—	—	—	MAINPR[2:0]		
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	HS = Hardware set

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **MAINPR[2:0]**: Main Routine Priority Selection bits

REGISTER 3-3: DMA1PR: DMA1 PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-1/1	R/W-0/0
—	—	—	—	—	DMA1PR[2:0]		
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	HS = Hardware set

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **DMA1PR[2:0]**: DMA1 Priority Selection bits

REGISTER 3-4: DMA2PR: DMA2 PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-1/1	R/W-1/1
—	—	—	—	—	DMA2PR[2:0]		
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	HS = Hardware set

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **DMA2PR[2:0]**: DMA2 Priority Selection bits

REGISTER 3-5: SCANPR: SCANNER PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-1/1	R/W-0/0	R/W-0/0
—	—	—	—	—	SCANPR[2:0]		
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	HS = Hardware set

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **SCANPR[2:0]**: Scanner Priority Selection bits

REGISTER 3-6: PRLOCK: PRIORITY LOCK REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	PRLOCKED
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	HS = Hardware set

bit 7-1 **Unimplemented:** Read as ‘0’

bit 0 **PRLOCKED**: PR Register Lock bit(1, 2)
0 = Priority Registers can be modified by write operations; Peripherals do not have access to the memory
1 = Priority Registers are locked and cannot be written; Peripherals have access to the memory

Note 1: The PRLOCKED bit can only be set or cleared after the unlock sequence.

2: If PR1WAY = 1, the PRLOCKED bit cannot be cleared after it has been set. A device Reset will clear the bit and allow one more set.

TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH CPU

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
ISRPR	—	—	—	—	—	ISRPR2	ISRPR1	ISRPR0	30
MAINPR	—	—	—	—	—	MAINPR2	MAINPR1	MAINPR0	30
DMA1PR	—	—	—	—	—	DMA1PR2	DMA1PR1	DMA1PR0	30
DMA2PR	—	—	—	—	—	DMA2PR2	DMA2PR1	DMA2PR0	31
SCANPR	—	—	—	—	—	SCANPR2	SCANPR1	SCANPR0	31
PRLOCK	—	—	—	—	—	—	—	PRLOCKED	31

Legend: — = Unimplemented location, read as ‘0’.

4.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 microcontroller devices:

- Program Flash Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate buses; this allows for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Program Flash Memory and Data EEPROM Memory is provided in **Section 13.0 "Nonvolatile Memory (NVM) Control"**.

4.1 Program Flash Memory Organization

PIC18 microcontrollers implement a 21-bit program counter, which is capable of addressing a 2 Mbyte program memory space. Accessing any unimplemented memory will return all '0's (a NOP instruction).

These devices contain the following:

- PIC18(L)F45/46K42: 32 Kbytes of Flash memory, up to 16,384 single-word instructions
- PIC18(L)F26/46/56K42: 64 Kbytes of Flash memory, up to 32,768 single-word instructions
- PIC18(L)F27/47/57K42: 128 Kbytes of Flash memory, up to 65,536 single-word instructions

The Reset vector for the device is at address 000000h. PIC18(L)F26/27/45/46/47/55/56/57K42 devices feature a vectored interrupt controller with a dedicated interrupt vector table in the program memory, see **Section 9.0 "Interrupt Controller"**.

Note:	For memory information on this family of devices, see Table 4-1 and Table 4-3.

4.2 Memory Access Partition (MAP)

Program Flash memory is partitioned into:

- Application Block
- Boot Block, and
- Storage Area Flash (SAF) Block

4.2.1 APPLICATION BLOCK

Application block is where the user's program resides by default. Default settings of the configuration bits ($\overline{\text{BBEN}}$ = 1 and $\overline{\text{SAFEN}}$ = 1) assign all memory in the program Flash memory area to the application block. The $\overline{\text{WRTAPP}}$ configuration bit is used to protect the application block.

4.2.2 BOOT BLOCK

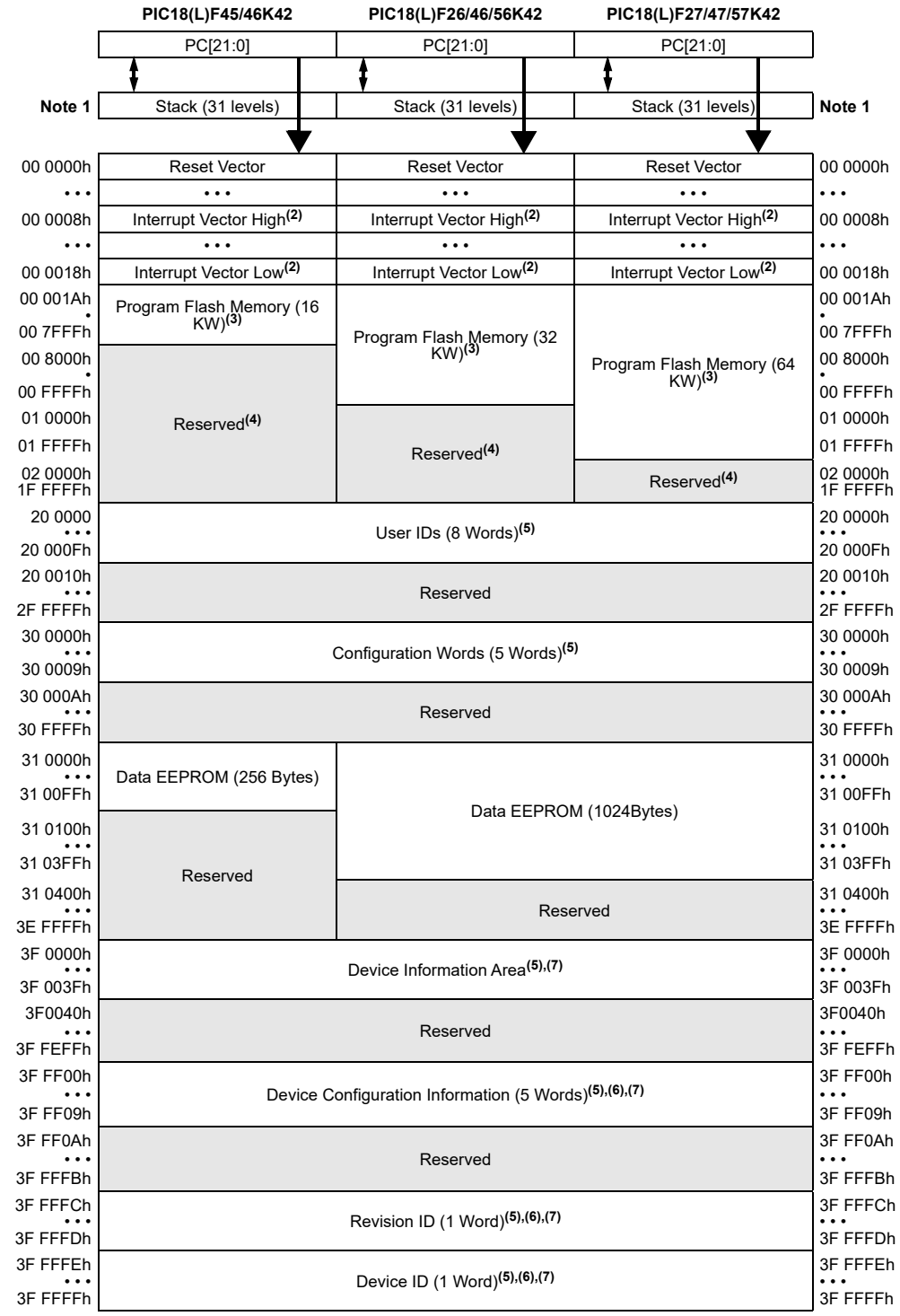
Boot block is an area in program memory that is ideal for storing bootloader code. Code placed in this area can be executed by the CPU. The boot block can be write-protected, independent of the main application block. The Boot Block is enabled by the $\overline{\text{BBEN}}$ bit and size is based on the value of the BBSIZE bits of Configuration word (Register 5-7), see Table 5-1 for boot block sizes. The $\overline{\text{WRTB}}$ Configuration bit is used to write-protect the Boot Block.

4.2.3 STORAGE AREA FLASH

Storage Area Flash (SAF) is the area in program memory that can be used as data storage. SAF is enabled by the $\overline{\text{SAFEN}}$ bit of the Configuration word in Register 5-7. If enabled, the code placed in this area cannot be executed by the CPU. The SAF block is placed at the end of memory and spans 128 Words. The $\overline{\text{WRTSAF}}$ Configuration bit is used to write-protect the Storage Area Flash.

Note:	If write-protected locations are written to, memory is not changed and the WRERR bit defined in Register 13-1 is set.

TABLE 4-1: PROGRAM AND DATA EEPROM MEMORY MAP



Note 1: The stack is a separate SRAM panel, apart from all user memory panels.
2: 00 0008h location is used as the reset default for the IVTBASE register, the vector table can be relocated in the memory by programming the IVTBASE register.
3: Storage area Flash is implemented as the last 128 Words of user Flash.
4: The addresses do not roll over. The region is read as '0'.
5: Not code-protected.
6: Hard-coded in silicon.
7: This region cannot be written by the user and it's not affected by a Bulk Erase.

TABLE 4-2: PROGRAM FLASH MEMORY PARTITION

Region	Address	Partition(3)			
		$\overline{\text{BBEN}}$ = 1 SAFEN = 1	$\overline{\text{BBEN}}$ = 1 SAFEN = 0	$\overline{\text{BBEN}}$ = 0 SAFEN = 1	$\overline{\text{BBEN}}$ = 0 SAFEN = 0
Program Flash Memory	00 0000h • • • Last Boot Block Memory Address	APPLICATION BLOCK	APPLICATION BLOCK	BOOT BLOCK	BOOT BLOCK
	Last Boot Block Memory Address(1) + 1 • • • Last Program Memory Address(2) – 100h			APPLICATION BLOCK	APPLICATION BLOCK
	Last Program Memory Address(2) – FEh(4) • • • Last Program Memory Address(2)		STORAGE AREA FLASH		STORAGE AREA FLASH

Note 1: Last Boot Block Memory Address is based on BBSIZE[2:0], see Table 5-1.
2: For Last Program Memory Address, see Table 4-1.
3: Refer to Register 5-7: Configuration Word 4L for $\overline{\text{BBEN}}$ and $\overline{\text{SAFEN}}$ definitions.
4: Storage area Flash is implemented as the last 128 Words of User Flash.

4.2.4 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bit wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC[15:8] bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC[20:16] bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by any operation that reads PCL. This is useful for computed offsets to the PC (see **Section 4.3.2.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

4.2.5 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack (TOS) Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL, CALLW or RCALL instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

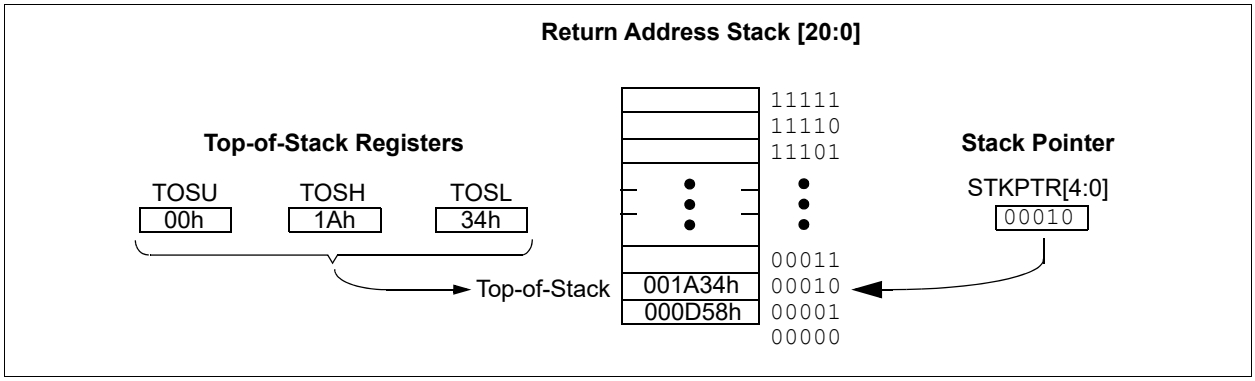
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits in the PCON0 register indicate if the stack has overflowed or underflowed.

4.2.5.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register (Figure 4-1). This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the Global Interrupt Enable (GIE) bits while accessing the stack to prevent inadvertent stack corruption.

FIGURE 4-1: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



4.2.5.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 4-4) contains the Stack Pointer value. The STKOVF (Stack Overflow) Status bit and the STKUNF (Stack Underflow) Status bit can be accessed using the PCON0 register. The value of the Stack Pointer can be 0 through 31. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for stack maintenance. After the PC is pushed onto the stack 32 times (without popping any values off the stack), the STKOVF bit is set. The STKOVF bit is cleared by software or by a POR. The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 5.1 "Configuration Words"** for a description of the device Configuration bits.)

If STVREN is set (default), a Reset will be generated and a Stack Overflow will be indicated by the STKOVF bit when the 32nd push is initiated. This includes `CALL` and `CALLW` instructions, as well as stacking the return address during an interrupt response. The STKOVF bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKOVF bit will be set on the 32nd push and the Stack Pointer will remain at 31 but no Reset will occur. Any additional pushes will overwrite the 31st push but the STKPTR will remain at 31.

Setting STKOVF = `1` in software will change the bit, but will not generate a Reset.

The STKUNF bit is set when a stack pop returns a value of zero. The STKUNF bit is cleared by software or by POR. The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 5.1 "Configuration Words"** for a description of the device Configuration bits).

If STVREN is set (default) and the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC, it will set the STKUNF bit and a Reset will be generated. This condition can be generated by the `RETURN`, `RETLW` and `RETFIE` instructions.

When STVREN = `0`, STKUNF will be set but no Reset will occur.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

4.2.5.3 `PUSH` and `POP` Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, `PUSH` and `POP`, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The `PUSH` instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The `POP` instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

4.3 Register Definitions: Stack Pointer

REGISTER 4-1: TOSU: TOP OF STACK UPPER BYTE

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	TOS[20:16]				
bit 7							bit 0

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

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **TOS[20:16]:** Top of Stack Location bits

REGISTER 4-2: TOSH: TOP OF STACK HIGH BYTE

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TOS[15:8]							
bit 7							bit 0

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

bit 7-0 **TOS[15:8]:** Top of Stack Location bits

REGISTER 4-3: TOSL: TOP OF STACK LOW BYTE

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TOS[7:0]							
bit 7							bit 0

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

bit 7-0 **TOS[7:0]:** Top of Stack Location bits

REGISTER 4-4: STKPTR: STACK POINTER REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	STKPTR[4:0]				
bit 7							bit 0

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

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **STKPTR[4:0]:** Stack Pointer Location bits

4.3.1 FAST REGISTER STACK

There are three levels of fast stack registers available - one for `CALL` type instructions and two for interrupts. A fast register stack is provided for the STATUS, WREG and BSR registers, to provide a “fast return” option for interrupts. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the `RETFIE, FAST` instruction is used to return from the interrupt. Refer to **Section 4.5.6 “Call Shadow Register”** for interrupt call shadow registers.

Example 4-1 shows a source code example that uses the fast register stack during a subroutine call and return.

EXAMPLE 4-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
       •
       •

SUB1   •
       •
   RETURN, FAST    ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

4.3.2 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed `GOTO`
- Table Reads

4.3.2.1 Computed `GOTO`

A computed `GOTO` is accomplished by adding an offset to the program counter. An example is shown in Example 4-2.

A look-up table can be formed with an `ADDWF PCL` instruction and a group of `RETLW nn` instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the `ADDWF PCL` instruction. The next instruction executed will be one of the `RETLW nn` instructions that returns the value '`nn`' to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of two (LSb = `0`).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

EXAMPLE 4-2: COMPUTED `GOTO` USING AN OFFSET VALUE

```
        MOVF    OFFSET, W
        CALL    TABLE
ORG     nn00h
TABLE   ADDWF   PCL
        RETLW   nnh
        RETLW   nnh
        RETLW   nnh
        .
        .
        .
```

4.3.2.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory.

Table read and table write operations are discussed further in **Section 13.1.1 "Table Reads and Table Writes"**.

4.4 PIC18 Instruction Cycle

4.4.1 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four cycles of the oscillator clock. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., `GOTO`), then two cycles are required to complete the instruction (Example 4-3).

A fetch cycle begins with the Program Counter (PC) incrementing followed by the execution cycle.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR). This instruction is then decoded and executed during the next few oscillator clock cycles. Data memory is read (operand read) and written (destination write) during the execution cycle as well.

EXAMPLE 4-3: INSTRUCTION PIPELINE FLOW

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. `MOVLW 55h`	Fetch 1	Execute 1				
2. `MOVWF PORTB`		Fetch 2	Execute 2			
3. `BRA SUB_1`			Fetch 3	Execute 3		
4. `BSF PORTA, BIT3 (Forced NOP)`				Fetch 4	Flush (`NOP`)	
5. `Instruction @ address SUB_1`					Fetch SUB_1	Execute SUB_1

Note: There are some instructions that take multiple cycles to execute. Refer to **Section 41.0 “Instruction Set Summary”** for details.

4.4.2 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as either two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = `0`). To maintain alignment with instruction boundaries, the PC increments in steps of two and the LSb will always read '`0`' (see **Section 4.2.4 "Program Counter"**).

Figure 4-2 shows an example of how instruction words are stored in the program memory.

The `CALL` and `GOTO` instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC[20:1], which accesses the desired byte address in program memory. Instruction #2 in Figure 4-2 shows how the instruction `GOTO 0006h` is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 41.0 "Instruction Set Summary"** provides further details of the instruction set.

4.4.3 MULTI-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: `CALL`, `MOVFF`, `GOTO` and `LFSR` and two three-word instructions: `MOVFFL` and `MOVSFL`. In all cases, the second and the third word of the instruction always has '`1111`' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '`1111`' in the four MSbs of an instruction specifies a special form of `NOP`. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second or third word is executed by itself, a `NOP` is executed instead. This is necessary for cases when the multi-word instruction is preceded by a conditional instruction that changes the PC. Example 4-4 shows how this works.

FIGURE 4-2: INSTRUCTIONS IN PROGRAM MEMORY

Program Memory Byte Locations →

			LSB = 1	LSB = 0	Word Address ↓
					000000h
					000002h
					000004h
					000006h
Instruction 1:	MOVLW	055h	0Fh	55h	000008h
Instruction 2:	GOTO	0006h	EFh	03h	00000Ah
			F0h	00h	00000Ch
Instruction 3:	MOVFF	123h, 456h	C1h	23h	00000Eh
			F4h	56h	000010h
Instruction 4:	MOVFFL	123h, 456h	00h	60h	000012h
			F4h	8Ch	000014h
			F4h	56h	000016h
					000018h
					00001Ah

EXAMPLE 4-4: TWO-WORD INSTRUCTIONS

CASE 1:	
Object Code	**Source Code**
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; Yes, skip this word
1111 0100 0101 0110	; Execute this word as a NOP
0010 0100 0000 0000	ADDWF REG3 ; continue code
CASE 2:	
Object Code	**Source Code**
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; No, execute this word
1111 0100 0101 0110	; 2nd word of instruction
0010 0100 0000 0000	ADDWF REG3 ; continue code

EXAMPLE 4-5: THREE-WORD INSTRUCTIONS

CASE 1:	
Object Code	**Source Code**
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
0000 0000 0110 0000	MOVFFL REG1, REG2 ; Yes, skip this word
1111 0100 1000 1100	; Execute this word as a NOP
1111 0100 0101 0110	; Execute this word as a NOP
0010 0100 0000 0000	ADDWF REG3 ; continue code
CASE 2:	
Object Code	**Source Code**
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
0000 0000 0110 0000	MOVFFL REG1, REG2 ; No, execute this word
1111 0100 1000 1100	; 2nd word of instruction
1111 0100 0101 0110	; 3rd word of instruction
0010 0100 0000 0000	ADDWF REG3 ; continue code

4.5 Data Memory Organization

Data memory in PIC18F26/27/45/46/47/55/56/57K42 devices is implemented as static RAM. Each register in the data memory has a 14-bit address, allowing up to 16384 bytes of data memory. The memory space is divided into 64 banks that contain 256 bytes each. Figure 4-3 shows the data memory organization for the PIC18F26/27/45/46/47/55/56/57K42 devices in this data sheet.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (select SFRs and GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to some SFRs and the lower portion of GPR Bank 0 without using the Bank Select Register (BSR). **Section 4.5.4 "Access Bank"** provides a detailed description of the Access RAM.

4.5.1 BANK SELECT REGISTER (BSR)

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 64 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 14-bit address, or an 8-bit low-order address and a 6-bit Bank Select Register.

This SFR holds the six Most Significant bits of a location address; the instruction itself includes the eight Least Significant bits. Only the six lower bits of the BSR are implemented (BSR[5:0]). The upper two bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the `MOVLB` instruction.

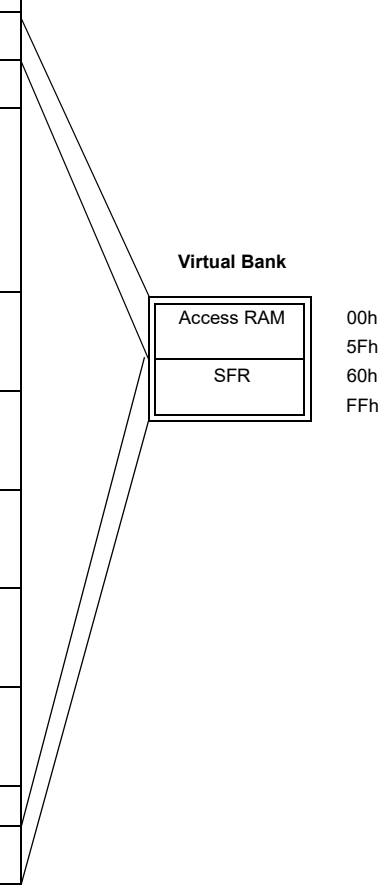
The value of the BSR indicates the bank in data memory; the eight bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 4-3.

Since up to 64 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 3Fh will end up corrupting the program counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory maps in Figure 4-3 indicate which banks are implemented.

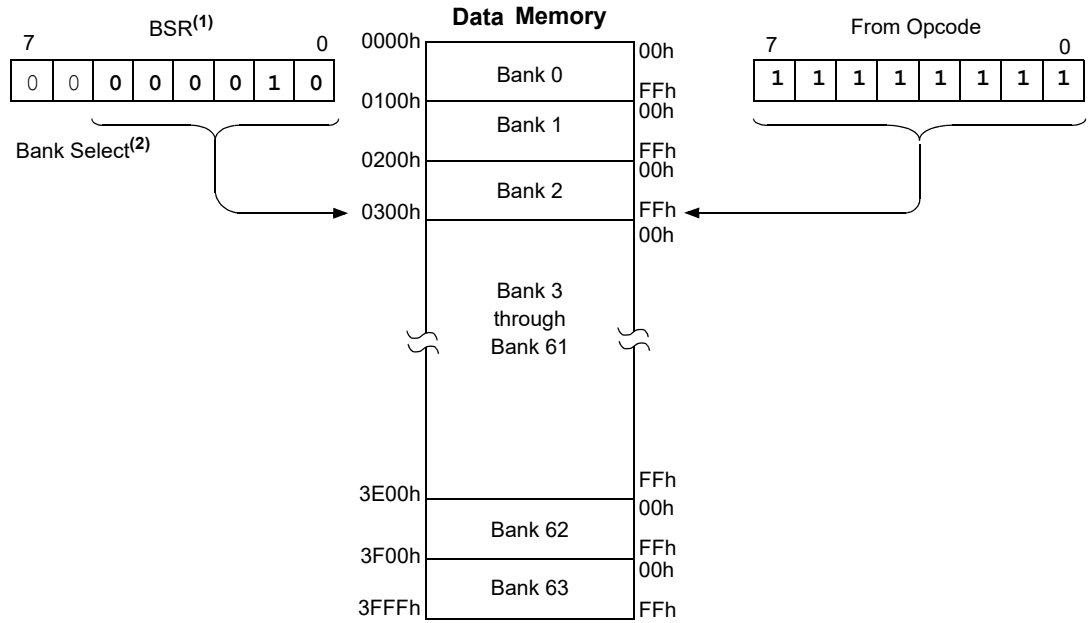
FIGURE 4-4: DATA MEMORY MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

Bank	BSR[5:0]	Address addr[7:0]	PIC18(L)F45K42 PIC18(L)F55K42	PIC18(L)F26K42 PIC18(L)F46K42 PIC18(L)F56K42	PIC18(L)F27K42 PIC18(L)F47K42 PIC18(L)F57K42	Address addr[13:0]
Bank 0	00 0000	00h	Access RAM	Access RAM	Access RAM	0000h 005Fh
		FFh	GPR	GPR	GPR	0060h 00FFh
Bank 1	00 0001	00h FFh	GPR	GPR	GPR	0100h 03FFh
Bank 2	00 0010	00h FFh				
Bank 3	00 0011	00h . . . FFh				
Banks 4 to 7	00 0100 – 00 0111	00h . . . FFh	GPR	GPR	GPR	0400h . . . 07FFh
Banks 8 to 15	00 1000 – 00 1111	00h . . . FFh	Unimplemented	GPR	GPR	0800h . . . 0FFFh
Banks 16 to 31	01 0000 – 01 1111	00h . . . FFh		Unimplemented		1000h . . . 1FFFh
Banks 32 to 55	10 0000 – 11 0111	00h . . . FFh			Unimplemented	2000h . . . 37FFh
Banks 56 to 62	11 1000 – 11 1110	00h . . . FFh	SFR	SFR	SFR	3800h . . . 3EFFh
Bank 63	11 1111	00h . . . FFh	SFR	SFR	SFR	3800h 3EFFh 3F60h 3FFFh



© 2017-2020 Microchip Technology Inc.

FIGURE 4-5: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)



Note 1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR[5:0]) to the registers of the Access Bank.

© 2017-2020 Microchip Technology Inc.

4.5.2 GENERAL PURPOSE REGISTER FILE

General Purpose RAM is available starting Bank 0 of data memory. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

4.5.3 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (3FFFh) and extend downward to occupy Bank 56 through 63 (3800h to 3FFFh). A list of these registers is given in Table 4-3 to Table 4-11. A bitwise summary of these registers can be found in **Section 42.0 "Register Summary"**.

4.5.4 ACCESS BANK

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Bank 63. The lower half is known as the "Access RAM" and is composed of GPRs. This upper half is also where some of the SFRs of the device are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed linearly by an 8-bit address (Figure 4-4).

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

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

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in **Section 4.8.3 "Mapping the Access Bank in Indexed Literal Offset Mode"**.

TABLE 4-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES (DMA ACCESS ONLY)

40FFh	—	40DFh	—	40BFh	—	409Fh	—	407Fh	—	405Fh	—	403Fh	—	401Fh	—
40FEh	—	40DEh	—	40BEh	—	409Eh	—	407Eh	—	405Eh	—	403Eh	—	401Eh	—
40FDh	—	40DDh	T6PR_M2	40BDh	ADRESH_M2	409Dh	—	407Dh	—	405Dh	—	403Dh	—	401Dh	—
40FCh	—	40DCh	PWM5DCH_M2	40BCh	ADRESL_M2	409Ch	—	407Ch	—	405Ch	—	403Ch	—	401Ch	—
40FBh	TMR5H_M1	40DBh	PWM5DCL_M2	40BBh	ADPCH_M2	409Bh	—	407Bh	—	405Bh	—	403Bh	—	401Bh	—
40FAh	TMR5L_M1	40DAh	T6PR_M1	40BAh	ADCLK_M1	409Ah	—	407Ah	—	405Ah	—	403Ah	—	401Ah	—
40F9h	TMR3H_M1	40D9h	CCPR1H_M2	40B9h	ADACT_M1	4099h	—	4079h	—	4059h	—	4039h	—	4019h	—
40F8h	TMR3L_M1	40D8h	CCPR1L_M2	40B8h	ADREF_M1	4098h	—	4078h	—	4058h	—	4038h	—	4018h	—
40F7h	TMR1H_M1	40D7h	T4PR_M4	40B7h	ADCON3_M1	4097h	—	4077h	—	4057h	—	4037h	—	4017h	—
40F6h	TMR1L_M1	40D6h	PWM8DCH_M1	40B6h	ADCON2_M1	4096h	ADRESH_M1	4076h	—	4056h	—	4036h	—	4016h	—
40F5h	—	40D5h	PWM8DCL_M1	40B5h	ADCON1_M1	4095h	ADRESL_M1	4075h	—	4055h	—	4035h	—	4015h	—
40F4h	—	40D4h	T4PR_M3	40B4h	ADCON0_M1	4094h	ADPCH_M1	4074h	—	4054h	—	4034h	—	4014h	—
40F3h	—	40D3h	PWM7DCH_M1	40B3h	ADCAP_M2	4093h	ADCAP_M1	4073h	—	4053h	—	4033h	—	4013h	—
40F2h	—	40D2h	PWM7DCL_M1	40B2h	ADACQH_M2	4092h	ADACQH_M1	4072h	—	4052h	—	4032h	—	4012h	—
40F1h	—	40D1h	T4PR_M2	40B1h	ADACQL_M2	4091h	ADACQL_M1	4071h	—	4051h	—	4031h	—	4011h	—
40F0h	—	40D0h	CCPR4H_M1	40B0h	ADPREVH_M2	4090h	ADPREVH_M1	4070h	—	4050h	—	4030h	—	4010h	—
40EFh	PWM8DCH_M2	40CFh	CCPR4L_M1	40AFh	ADPREVL_M2	408Fh	ADPREVL_M1	406Fh	—	404Fh	—	402Fh	—	400Fh	—
40EEh	PWM8DCL_M2	40CEh	T4PR_M1	40AEh	ADRPT_M2	408Eh	ADRPT_M1	406Eh	—	404Eh	—	402Eh	—	400Eh	—
40EDh	PWM7DCH_M2	40CDh	CCPR3H_M1	40ADh	ADCNT_M2	408Dh	ADCNT_M1	406Dh	—	404Dh	—	402Dh	—	400Dh	—
40ECh	PWM7DCL_M2	40CCh	CCPR3L_M1	40ACh	ADACCU_M2	408Ch	ADACCU_M1	406Ch	—	404Ch	—	402Ch	—	400Ch	—
40EBh	PWM6DCH_M2	40CBh	T2PR_M3	40ABh	ADACCH_M2	408Bh	ADACCH_M1	406Bh	—	404Bh	—	402Bh	—	400Bh	—
40EAh	PWM6DCL_M2	40CAh	PWM6DCH_M1	40AAh	ADACCL_M2	408Ah	ADACCL_M1	406Ah	—	404Ah	—	402Ah	—	400Ah	—
40E9h	PWM5DCH_M3	40C9h	PWM6DCL_M1	40A9h	ADFLTRH_M2	4089h	ADFLTRH_M1	4069h	—	4049h	—	4029h	—	4009h	—
40E8h	PWM5DCL_M3	40C8h	T2PR_M2	40A8h	ADFLTRL_M2	4088h	ADFLTRL_M1	4068h	—	4048h	—	4028h	—	4008h	—
40E7h	CCPR4H_M2	40C7h	PWM5DCH_M1	40A7h	ADSTPTH_M2	4087h	ADSTPTH_M1	4067h	—	4047h	—	4027h	—	4007h	—
40E6h	CCPR4L_M2	40C6h	PWM5DCL_M1	40A6h	ADSTPTL_M2	4086h	ADSTPTL_M1	4066h	—	4046h	—	4026h	—	4006h	—
40E5h	CCPR3H_M2	40C5h	T2PR_M2	40A5h	ADERRH_M2	4085h	ADERRH_M1	4065h	—	4045h	—	4025h	—	4005h	—
40E4h	CCPR3L_M2	40C4h	CCPR2H_M1	40A4h	ADERRL_M2	4084h	ADERRL_M1	4064h	—	4044h	—	4024h	—	4004h	—
40E3h	CCPR2H_M2	40C3h	CCPR2L_M1	40A3h	ADUTHH_M2	4083h	ADUTHH_M1	4063h	IOCEF_M1	4043h	—	4023h	—	4003h	—
40E2h	CCPR2L_M2	40C2h	T2PR_M1	40A2h	ADUTHL_M2	4082h	ADUTHL_M1	4062h	IOCCF_M1	4042h	—	4022h	—	4002h	—
40E1h	CCPR1H_M3	40C1h	CCPR1H_M1	40A1h	ADLTHH_M2	4081h	ADLTHH_M1	4061h	IOCBF_M1	4041h	—	4021h	—	4001h	—
40E0h	CCPR1L_M3	40C0h	CCPR1L_M1	40A0h	ADLTHL_M2	4080h	ADLTHL_M1	4060h	IOCAF_M1	4040h	—	4020h	—	4000h	—

Note 1: Addresses in this table are accessible ONLY through DMA Source and Destination Address Registers. CPU does not have access to these registers.

© 2017-2020 Microchip Technology Inc.

TABLE 4-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 63

3FFFh	TOSU	3FDFh	INDF2	3FBFh	LATF(3)	3F9Fh	T4PR	3F7Fh	CCP1CAP	3F5Fh	CCPTMRS1	3F3Fh	NCO1CLK	3F1Fh	SMT1CON1
3FFEh	TOSH	3FDEh	POSTINC2	3FBEh	LATE(2)	3F9Eh	T4TMR	3F7Eh	CCP1CON	3F5Eh	CCPTMRS0	3F3Eh	NCO1CON	3F1Eh	SMT1CON0
3FFDh	TOSL	3FDDh	POSTDEC2	3FBDh	LATD(2)	3F9Dh	T5CLK	3F7Dh	CCPR1H	3F5Dh	—	3F3Dh	NCO1INCU	3F1Dh	SMT1PRU
3FFCh	STKPTR	3FDCh	PRECIN2	3FBCh	LATC	3F9Ch	T5GATE	3F7Ch	CCPR1L	3F5Ch	—	3F3Ch	NCO1INCH	3F1Ch	SMT1PRH
3FFBh	PCLATU	3FDBh	PLUSW2	3FBBh	LATB	3F9Bh	T5GCON	3F7Bh	CCP2CAP	3F5Bh	—	3F3Bh	NCO1INCL	3F1Bh	SMT1PRL
3FFAh	PCLATH	3FDAh	FSR2H	3FBAh	LATA	3F9Ah	T5CON	3F7Ah	CCP2CON	3F5Ah	CWG1STR	3F3Ah	NCO1ACCU	3F1Ah	SMT1CPWU
3FF9h	PCL	3FD9h	FSR2L	3FB9h	T0CON1	3F99h	TMR5H	3F79h	CCPR2H	3F59h	CWG1AS1	3F39h	NCO1ACCH	3F19h	SMT1CPWH
3FF8h	TBLPRTU	3FD8h	STATUS	3FB8h	T0CON0	3F98h	TMR5L	3F78h	CCPR2L	3F58h	CWG1AS0	3F38h	NCO1ACCL	3F18h	SMT1CPWL
3FF7h	TBLPTRH	3FD7h	IVTBASEU	3FB7h	TMR0H	3F97h	T6RST	3F77h	CCP3CAP	3F57h	CWG1CON1	3F37h	—	3F17h	SMT1CPRU
3FF6h	TBLPTRL	3FD6h	IVTBASEH	3FB6h	TMR0L	3F96h	T6CLK	3F76h	CCP3CON	3F56h	CWG1CON0	3F36h	—	3F16h	SMT1CPRH
3FF5h	TABLAT	3FD5h	IVTBASEL	3FB5h	T1CLK	3F95h	T6HLT	3F75h	CCPR3H	3F55h	CWG1DBF	3F35h	—	3F15h	SMT1CPRL
3FF4h	PRODH	3FD4h	IVTLOCK	3FB4h	T1GATE	3F94h	T6CON	3F74h	CCPR3L	3F54h	CWG1DBR	3F34h	—	3F14h	SMT1TMRU
3FF3h	PRODL	3FD3h	INTCON1	3FB3h	T1GCON	3F93h	T6PR	3F73h	CCP4CAP	3F53h	CWG1ISM	3F33h	—	3F13h	SMT1TMRH
3FF2h	—	3FD2h	INTCON0	3FB2h	T1CON	3F92h	T6TMR	3F72h	CCP4CON	3F52h	CWG1CLK	3F32h	—	3F12h	SMT1TMRL
3FF1h	PCON1	3FD1h	—	3FB1h	TMR1H	3F91h	—	3F71h	CCPR4H	3F51h	CWG2STR	3F31h	—	3F11h	—
3FF0h	PCON0	3FD0h	—	3FB0h	TMR1L	3F90h	—	3F70h	CCPR4L	3F50h	CWG2AS1	3F30h	—	3F10h	—
3FEFh	INDF0	3FCFh	PORTF(3)	3FAFh	T2RST	3F8Fh	—	3F6Fh	—	3F4Fh	CWG2AS0	3F2Fh	—	3F0Fh	—
3FEEh	POSTINC0	3FCEh	PORTE	3FAEh	T2CLK	3F8Eh	—	3F6Eh	PWM5CON	3F4Eh	CWG2CON1	3F2Eh	—	3F0Eh	—
3FEDh	POSTDEC0	3FCDh	PORTD(2)	3FADh	T2HLT	3F8Dh	—	3F6Dh	PWM5DCH	3F4Dh	CWG2CON0	3F2Dh	—	3F0Dh	—
3FECh	PRECIN0	3FCCh	PORTC	3FACh	T2CON	3F8Ch	—	3F6Ch	PWM5DCL	3F4Ch	CWG2DBF	3F2Ch	—	3F0Ch	—
3FEBh	PLUSW0	3FCBh	PORTB	3FABh	T2PR	3F8Bh	—	3F6Bh	—	3F4Bh	CWG2DBR	3F2Bh	—	3F0Bh	—
3FEAh	FSR0H	3FCAh	PORTA	3FAAh	T2TMR	3F8Ah	—	3F6Ah	PWM6CON	3F4Ah	CWG2ISM	3F2Ah	—	3F0Ah	—
3FE9h	FSR0L	3FC9h	—	3FA9h	T3CLK	3F89h	—	3F69h	PWM6DCH	3F49h	CWG2CLK	3F29h	—	3F09h	—
3FE8h	WREG	3FC8h	—	3FA8h	T3GATE	3F88h	—	3F68h	PWM6DCL	3F48h	CWG3STR	3F28h	—	3F08h	—
3FE7h	INDF1	3FC7h	TRISF(3)	3FA7h	T3GCON	3F87h	—	3F67h	—	3F47h	CWG3AS1	3F27h	—	3F07h	—
3FE6h	POSTINC1	3FC6h	TRISE(2)	3FA6h	T3CON	3F86h	—	3F66h	PWM7CON	3F46h	CWG3AS0	3F26h	—	3F06h	—
3FE5h	POSTDEC1	3FC5h	TRISD(2)	3FA5h	TMR3H	3F85h	—	3F65h	PWM7DCH	3F45h	CWG3CON1	3F25h	—	3F05h	—
3FE4h	PRECIN1	3FC4h	TRISC	3FA4h	TMR3L	3F84h	—	3F64h	PWM7DCL	3F44h	CWG3CON0	3F24h	—	3F04h	—
3FE3h	PLUSW1	3FC3h	TRISB	3FA3h	T4RST	3F83h	—	3F63h	—	3F43h	CWG3DBF	3F23h	SMT1WIN	3F03h	—
3FE2h	FSR1H	3FC2h	TRISA	3FA2h	T4CLK	3F82h	—	3F62h	PWM8CON	3F42h	CWG3DBR	3F22h	SMT1SIG	3F02h	—
3FE1h	FSR1L	3FC1h	—	3FA1h	T4HLT	3F81h	—	3F61h	PWM8DCH	3F41h	CWG3ISM	3F21h	SMT1CLK	3F01h	—
3FE0h	BSR	3FC0h	—	3FA0h	T4CON	3F80h	—	3F60h	PWM8DCL	3F40h	CWG3CLK	3F20h	SMT1STAT	3F00h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

© 2017-2020 Microchip Technology Inc.

TABLE 4-5: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 62

3EFFh	ADCLK	3EDFh	ADLTHH	3EBFh	CM1PCH	3E9Fh	—	3E7Fh	—	3E5Fh	—	3E3Fh	—	3E1Fh	—
3EFEh	ADACT	3EDEh	ADLTHL	3EBEh	CM1NCH	3E9Eh	DAC1CON0	3E7Eh	—	3E5Eh	—	3E3Eh	—	3E1Eh	—
3EFDh	ADREF	3EDDh	—	3EBDh	CM1CON1	3E9Dh	—	3E7Dh	—	3E5Dh	—	3E3Dh	—	3E1Dh	—
3EFCh	ADSTAT	3EDCh	—	3EBCh	CM1CON0	3E9Ch	DAC1CON1	3E7Ch	—	3E5Ch	—	3E3Ch	—	3E1Ch	—
3EFBh	ADCON3	3EDBh	—	3EBBh	CM2PCH	3E9Bh	—	3E7Bh	—	3E5Bh	—	3E3Bh	—	3E1Bh	—
3EFAh	ADCON2	3EDAh	—	3EBAh	CM2NCH	3E9Ah	—	3E7Ah	—	3E5Ah	—	3E3Ah	—	3E1Ah	—
3EF9h	ADCON1	3ED9h	—	3EB9h	CM2CON1	3E99h	—	3E79h	—	3E59h	—	3E39h	—	3E19h	—
3EF8h	ADCON0	3ED8h	—	3EB8h	CM2CON0	3E98h	—	3E78h	—	3E58h	—	3E38h	—	3E18h	—
3EF7h	ADPREH	3ED7h	ADCP	3EB7h	—	3E97h	—	3E77h	—	3E57h	—	3E37h	—	3E17h	—
3EF6h	ADPREL	3ED6h	—	3EB6h	—	3E96h	—	3E76h	—	3E56h	—	3E36h	—	3E16h	—
3EF5h	ADCAP	3ED5h	—	3EB5h	—	3E95h	—	3E75h	—	3E55h	—	3E35h	—	3E15h	—
3EF4h	ADACQH	3ED4h	—	3EB4h	—	3E94h	—	3E74h	—	3E54h	—	3E34h	—	3E14h	—
3EF3h	ADACQL	3ED3h	—	3EB3h	—	3E93h	—	3E73h	—	3E53h	—	3E33h	—	3E13h	—
2EF2h	—	3ED2h	—	3EB2h	—	3E92h	—	3E72h	—	3E52h	—	3E32h	—	3E12h	—
3EF1h	ADPCH	3ED1h	—	3EB1h	—	3E91h	—	3E71h	—	3E51h	—	3E31h	—	3E11h	—
3EF0h	ADRESH	3ED0h	—	3EB0h	—	3E90h	—	3E70h	—	3E50h	—	3E30h	—	3E10h	—
3EEFh	ADRESL	3ECFh	—	3EAFh	—	3E8Fh	—	3E6Fh	—	3E4Fh	—	3E2Fh	—	3E0Fh	—
3EEEh	ADPREVH	3ECEh	—	3EAEh	—	3E8Eh	—	3E6Eh	—	3E4Eh	—	3E2Eh	—	3E0Eh	—
3EEDh	ADPREVL	3ECDh	—	3EADh	—	3E8Dh	—	3E6Dh	—	3E4Dh	—	3E2Dh	—	3E0Dh	—
3EECh	ADRPT	3ECCh	—	3EACh	—	3E8Ch	—	3E6Ch	—	3E4Ch	—	3E2Ch	—	3E0Ch	—
3EEBh	ADCNT	3ECBh	—	3EABh	—	3E8Bh	—	3E6Bh	—	3E4Bh	—	3E2Bh	—	3E0Bh	—
3EEAh	ADACCU	3ECAh	HLVDCON1	3EAAh	—	3E8Ah	—	3E6Ah	—	3E4Ah	—	3E2Ah	—	3E0Ah	—
3EE9h	ADACCH	3EC9h	HLVDCON0	3EA9h	—	3E89h	—	3E69h	—	3E49h	—	3E29h	—	3E09h	—
3EE8h	ADACCL	3EC8h	—	3EA8h	—	3E88h	—	3E68h	—	3E48h	—	3E28h	—	3E08h	—
3EE7h	ADFLTRH	3EC7h	—	3EA7h	—	3E87h	—	3E67h	—	3E47h	—	3E27h	—	3E07h	—
3EE6h	ADFLTRL	3EC6h	—	3EA6h	—	3E86h	—	3E66h	—	3E46h	—	3E26h	—	3E06h	—
3EE5h	ADSTPTH	3EC5h	—	3EA5h	—	3E85h	—	3E65h	—	3E45h	—	3E25h	—	3E05h	—
3EE4h	ADSTPTL	3EC4h	—	3EA4h	—	3E84h	—	3E64h	—	3E44h	—	3E24h	—	3E04h	—
3EE3h	ADERRH	3EC3h	ZCDCON	3EA3h	—	3E83h	—	3E63h	—	3E43h	—	3E23h	—	3E03h	—
3EE2h	ADERRL	3EC2h	—	3EA2h	—	3E82h	—	3E62h	—	3E42h	—	3E22h	—	3E02h	—
3EE1h	ADUTHH	3EC1h	FVRCON	3EA1h	—	3E81h	—	3E61h	—	3E41h	—	3E21h	—	3E01h	—
3EE0h	ADUTHL	3EC0h	CMOUT	3EA0h	—	3E80h	—	3E60h	—	3E40h	—	3E20h	—	3E00h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

© 2017-2020 Microchip Technology Inc.

TABLE 4-6: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 61

3DFFh	—	3DDFh	U2FIFO	3DBFh	—	3D9Fh	—	3D7Fh	—	3D5Fh	I2C2CON2	3D3Fh	—	3D1Fh	—
3DFEh	—	3DDEh	U2BRGH	3DBEh	—	3D9Eh	—	3D7Eh	—	3D5Eh	I2C2CON1	3D3Eh	—	3D1Eh	—
3DFDh	—	3DDDh	U2BRGL	3DBDh	—	3D9Dh	—	3D7Dh	—	3D5Dh	I2C2CON0	3D3Dh	—	3D1Dh	—
3DFCh	—	3DDCh	U2CON2	3DBCh	—	3D9Ch	—	3D7Ch	I2C1BTO	3D5Ch	I2C2ADR3	3D3Ch	—	3D1Ch	SPI1CLK
3DFBh	—	3DDBh	U2CON1	3DBBh	—	3D9Bh	—	3D7Bh	I2C1CLK	3D5Bh	I2C2ADR2	3D3Bh	—	3D1Bh	SPI1INTE
3DFAh	U1ERRIE	3DDAh	U2CON0	3DBAh	—	3D9Ah	—	3D7Ah	I2C1PIE	3D5Ah	I2C2ADR1	3D3Ah	—	3D1Ah	SPI1INTF
3DF9h	U1ERRIR	3DD9h	—	3DB9h	—	3D99h	—	3D79h	I2C1PIR	3D59h	I2C2ADR0	3D39h	—	3D19h	SPI1BAUD
3DF8h	U1UIR	3DD8h	U2P3L	3DB8h	—	3D98h	—	3D78h	I2C1STAT1	3D58h	I2C2ADB1	3D38h	—	3D18h	SPI1TWIDTH
3DF7h	U1FIFO	3DD7h	—	3DB7h	—	3D97h	—	3D77h	I2C1STAT0	3D57h	I2C2ADB0	3D37h	—	3D17h	SPI1STATUS
3DF6h	U1BRGH	3DD6h	U2P2L	3DB6h	—	3D96h	—	3D76h	I2C1ERR	3D56h	I2C2CNT	3D36h	—	3D16h	SPI1CON2
3DF5h	U1BRGL	3DD5h	—	3DB5h	—	3D95h	—	3D75h	I2C1CON2	3D55h	I2C2TXB	3D35h	—	3D15h	SPI1CON1
3DF4h	U1CON2	3DD4h	U2P1L	3DB4h	—	3D94h	—	3D74h	I2C1CON1	3D54h	I2C2RXB	3D34h	—	3D14h	SPI1CON0
3DF3h	U1CON1	3DD3h	—	3DB3h	—	3D93h	—	3D73h	I2C1CON0	3D53h	—	3D33h	—	3D13h	SPI1TCNTH
3DF2h	U1CON0	3DD2h	U2TXB	3DB2h	—	3D92h	—	3D72h	I2C1ADR3	3D52h	—	3D32h	—	3D12h	SPI1TCNTL
3DF1h	U1P3H	3DD1h	—	3DB1h	—	3D91h	—	3D71h	I2C1ADR2	3D51h	—	3D31h	—	3D11h	SPI1TXB
3DF0h	U1P3L	3DD0h	U2RXB	3DB0h	—	3D90h	—	3D70h	I2C1ADR1	3D50h	—	3D30h	—	3D10h	SPI1RXB
3DEFh	U1P2H	3DCFh	—	3DAFh	—	3D8Fh	—	3D6Fh	I2C1ADR0	3D4Fh	—	3D2Fh	—	3D0Fh	—
3DEEh	U1P2L	3DCEh	—	3DAEh	—	3D8Eh	—	3D6Eh	I2C1ADB1	3D4Eh	—	3D2Eh	—	3D0Eh	—
3DEDh	U1P1H	3DCDh	—	3DADh	—	3D8Dh	—	3D6Dh	I2C1ADB0	3D4Dh	—	3D2Dh	—	3D0Dh	—
3DECh	U1P1L	3DCCh	—	3DACh	—	3D8Ch	—	3D6Ch	I2C1CNT	3D4Ch	—	3D2Ch	—	3D0Ch	—
3DEBh	U1TXCHK	3DCBh	—	3DABh	—	3D8Bh	—	3D6Bh	I2C1TXB	3D4Bh	—	3D2Bh	—	3D0Bh	—
3DEAh	U1TXB	3DCAh	—	3DAAh	—	3D8Ah	—	3D6Ah	I2C1RXB	3D4Ah	—	3D2Ah	—	3D0Ah	—
3DE9h	U1RXCHK	3DC9h	—	3DA9h	—	3D89h	—	3D69h	—	3D49h	—	3D29h	—	3D09h	—
3DE8h	U1RXB	3DC8h	—	3DA8h	—	3D88h	—	3D68h	—	3D48h	—	3D28h	—	3D08h	—
3DE7h	—	3DC7h	—	3DA7h	—	3D87h	—	3D67h	—	3D47h	—	3D27h	—	3D07h	—
3DE6h	—	3DC6h	—	3DA6h	—	3D86h	—	3D66h	I2C2BTO	3D46h	—	3D26h	—	3D06h	—
3DE5h	—	3DC5h	—	3DA5h	—	3D85h	—	3D65h	I2C2CLK	3D45h	—	3D25h	—	3D05h	—
3DE4h	—	3DC4h	—	3DA4h	—	3D84h	—	3D64h	I2C2PIE	3D44h	—	3D24h	—	3D04h	—
3DE3h	—	3DC3h	—	3DA3h	—	3D83h	—	3D63h	I2C2PIR	3D43h	—	3D23h	—	3D03h	—
3DE2h	U2ERRIE	3DC2h	—	3DA2h	—	3D82h	—	3D62h	I2C2STAT1	3D42h	—	3D22h	—	3D02h	—
3DE1h	U2ERRIR	3DC1h	—	3DA1h	—	3D81h	—	3D61h	I2C2STAT0	3D41h	—	3D21h	—	3D01h	—
3DE0h	U2UIR	3DC0h	—	3DA0h	—	3D80h	—	3D60h	I2C2ERR	3D40h	—	3D20h	—	3D00h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

© 2017-2020 Microchip Technology Inc.

TABLE 4-7: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 60

3CFFh	—	3CDFh	—	3CBFh	—	3C9Fh	—	3C7Fh	—	3C5Fh	CLC4GLS3	3C3Fh	—	3C1Fh	—
3CFEh	MD1CARH	3CDEh	—	3CBEh	—	3C9Eh	—	3C7Eh	CLCDATA0	3C5Eh	CLC4GLS2	3C3Eh	—	3C1Eh	—
3CFDh	MD1CARL	3CDDh	—	3CBDh	—	3C9Dh	—	3C7Dh	CLC1GLS3	3C5Dh	CLC4GLS1	3C3Dh	—	3C1Dh	—
3CFCh	MD1SRC	3CDCh	—	3CBCh	—	3C9Ch	—	3C7Ch	CLC1GLS2	3C5Ch	CLC4GLS0	3C3Ch	—	3C1Ch	—
3CFBh	MD1CON1	3CDBh	—	3CBBh	—	3C9Bh	—	3C7Bh	CLC1GLS1	3C5Bh	CLC4SEL3	3C3Bh	—	3C1Bh	—
3CFAh	MD1CON0	3CDAh	—	3CBAh	—	3C9Ah	—	3C7Ah	CLC1GLS0	3C5Ah	CLC4SEL2	3C3Ah	—	3C1Ah	—
3CF9h	—	3CD9h	—	3CB9h	—	3C99h	—	3C79h	CLC1SEL3	3C59h	CLC4SEL1	3C39h	—	3C19h	—
3CF8h	—	3CD8h	—	3CB8h	—	3C98h	—	3C78h	CLC1SEL2	3C58h	CLC4SEL0	3C38h	—	3C18h	—
3CF7h	—	3CD7h	—	3CB7h	—	3C97h	—	3C77h	CLC1SEL1	3C57h	CLC4POL	3C37h	—	3C17h	—
3CF6h	—	3CD6h	—	3CB6h	—	3C96h	—	3C76h	CLC1SEL0	3C56h	CLC4CON	3C36h	—	3C16h	—
3CF5h	—	3CD5h	—	3CB5h	—	3C95h	—	3C75h	CLC1POL	3C55h	—	3C35h	—	3C15h	—
3CF4h	—	3CD4h	—	3CB4h	—	3C94h	—	3C74h	CLC1CON	3C54h	—	3C34h	—	3C14h	—
3CF3h	—	3CD3h	—	3CB3h	—	3C93h	—	3C73h	CLC2GLS3	3C53h	—	3C33h	—	3C13h	—
3CF2h	—	3CD2h	—	3CB2h	—	3C92h	—	3C72h	CLC2GLS2	3C52h	—	3C32h	—	3C12h	—
3CF1h	—	3CD1h	—	3CB1h	—	3C91h	—	3C71h	CLC2GLS1	3C51h	—	3C31h	—	3C11h	—
3CF0h	—	3CD0h	—	3CB0h	—	3C90h	—	3C70h	CLC2GLS0	3C50h	—	3C30h	—	3C10h	—
3CEFh	—	3CCFh	—	3CAFh	—	3C8Fh	—	3C6Fh	CLC2SEL3	3C4Fh	—	3C2Fh	—	3C0Fh	—
3CEEh	—	3CCEh	—	3CAEh	—	3C8Eh	—	3C6Eh	CLC2SEL2	3C4Eh	—	3C2Eh	—	3C0Eh	—
3CEDh	—	3CCDh	—	3CADh	—	3C8Dh	—	3C6Dh	CLC2SEL1	3C4Dh	—	3C2Dh	—	3C0Dh	—
3CECh	—	3CCCh	—	3CACh	—	3C8Ch	—	3C6Ch	CLC2SEL0	3C4Ch	—	3C2Ch	—	3C0Ch	—
3CEBh	—	3CCBh	—	3CABh	—	3C8Bh	—	3C6Bh	CLC2POL	3C4Bh	—	3C2Bh	—	3C0Bh	—
3CEAh	—	3CCAh	—	3CAAh	—	3C8Ah	—	3C6Ah	CLC2CON	3C4Ah	—	3C2Ah	—	3C0Ah	—
3CE9h	—	3CC9h	—	3CA9h	—	3C89h	—	3C69h	CLC3GLS3	3C49h	—	3C29h	—	3C09h	—
3CE8h	—	3CC8h	—	3CA8h	—	3C88h	—	3C68h	CLC3GLS2	3C48h	—	3C28h	—	3C08h	—
3CE7h	—	3CC7h	—	3CA7h	—	3C87h	—	3C67h	CLC3GLS1	3C47h	—	3C27h	—	3C07h	—
3CE6h	CLKRCLK	3CC6h	—	3CA6h	—	3C86h	—	3C66h	CLC3GLS0	3C46h	—	3C26h	—	3C06h	—
3CE5h	CLKRCON	3CC5h	—	3CA5h	—	3C85h	—	3C65h	CLC3SEL3	3C45h	—	3C25h	—	3C05h	—
3CE4h	—	3CC4h	—	3CA4h	—	3C84h	—	3C64h	CLC3SEL2	3C44h	—	3C24h	—	3C04h	—
3CE3h	—	3CC3h	—	3CA3h	—	3C83h	—	3C63h	CLC3SEL1	3C43h	—	3C23h	—	3C03h	—
3CE2h	—	3CC2h	—	3CA2h	—	3C82h	—	3C62h	CLC3SEL0	3C42h	—	3C22h	—	3C02h	—
3CE1h	—	3CC1h	—	3CA1h	—	3C81h	—	3C61h	CLC3POL	3C41h	—	3C21h	—	3C01h	—
3CE0h	—	3CC0h	—	3CA0h	—	3C80h	—	3C60h	CLC3CON	3C40h	—	3C20h	—	3C00h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

© 2017-2020 Microchip Technology Inc.

TABLE 4-8: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 59

3BFFh	DMA1SIRQ	3BDFh	DMA2SIRQ	3BBFh	—	3B9Fh	—	3B7Fh	—	3B5Fh	—	3B3Fh	—	3B1Fh	—
3BFEh	DMA1AIRQ	3BDEh	DMA2AIRQ	3BBEh	—	3B9Eh	—	3B7Eh	—	3B5Eh	—	3B3Eh	—	3B1Eh	—
3BFDh	DMA1CON1	3BDDh	DMA2CON1	3BBDh	—	3B9Dh	—	3B7Dh	—	3B5Dh	—	3B3Dh	—	3B1Dh	—
3BFCh	DMA1CON0	3BDCh	DMA2CON0	3BBCh	—	3B9Ch	—	3B7Ch	—	3B5Ch	—	3B3Ch	—	3B1Ch	—
3BFBh	DMA1SSAU	3BDBh	DMA2SSAU	3BBBh	—	3B9Bh	—	3B7Bh	—	3B5Bh	—	3B3Bh	—	3B1Bh	—
3BFAh	DMA1SSAH	3BDAh	DMA2SSAH	3BBAh	—	3B9Ah	—	3B7Ah	—	3B5Ah	—	3B3Ah	—	3B1Ah	—
3BF9h	DMA1SSAL	3BD9h	DMA2SSAL	3BB9h	—	3B99h	—	3B79h	—	3B59h	—	3B39h	—	3B19h	—
3BF8h	DMA1SSZH	3BD8h	DMA2SSZH	3BB8h	—	3B98h	—	3B78h	—	3B58h	—	3B38h	—	3B18h	—
3BF7h	DMA1SSZL	3BD7h	DMA2SSZL	3BB7h	—	3B97h	—	3B77h	—	3B57h	—	3B37h	—	3B17h	—
3BF6h	DMA1SPTRU	3BD6h	DMA2SPTRU	3BB6h	—	3B96h	—	3B76h	—	3B56h	—	3B36h	—	3B16h	—
3BF5h	DMA1SPTRH	3BD5h	DMA2SPTRH	3BB5h	—	3B95h	—	3B75h	—	3B55h	—	3B35h	—	3B15h	—
3BF4h	DMA1SPTRL	3BD4h	DMA2SPTRL	3BB4h	—	3B94h	—	3B74h	—	3B54h	—	3B34h	—	3B14h	—
3BF3h	DMA1SCNTH	3BD3h	DMA2SCNTH	3BB3h	—	3B93h	—	3B73h	—	3B53h	—	3B33h	—	3B13h	—
3BF2h	DMA1SCNTL	3BD2h	DMA2SCNTL	3BB2h	—	3B92h	—	3B72h	—	3B52h	—	3B32h	—	3B12h	—
3BF1h	DMA1DSAH	3BD1h	DMA2DSAH	3BB1h	—	3B91h	—	3B71h	—	3B51h	—	3B31h	—	3B11h	—
3BF0h	DMA1DSAL	3BD0h	DMA2DSAL	3BB0h	—	3B90h	—	3B70h	—	3B50h	—	3B30h	—	3B10h	—
3BEFh	DMA1DSZH	3BCFh	DMA2DSZH	3BAFh	—	3B8Fh	—	3B6Fh	—	3B4Fh	—	3B2Fh	—	3B0Fh	—
3BEEh	DMA1DSZL	3BCEh	DMA2DSZL	3BAEh	—	3B8Eh	—	3B6Eh	—	3B4Eh	—	3B2Eh	—	3B0Eh	—
3BEDh	DMA1DPTRH	3BCDh	DMA2DPTRH	3BADh	—	3B8Dh	—	3B6Dh	—	3B4Dh	—	3B2Dh	—	3B0Dh	—
3BECh	DMA1DPTRL	3BCCh	DMA2DPTRL	3BACh	—	3B8Ch	—	3B6Ch	—	3B4Ch	—	3B2Ch	—	3B0Ch	—
3BEBh	DMA1DCNTH	3BCBh	DMA2DCNTH	3BABh	—	3B8Bh	—	3B6Bh	—	3B4Bh	—	3B2Bh	—	3B0Bh	—
3BEAh	DMA1DCNTL	3BCAh	DMA2DCNTL	3BAAh	—	3B8Ah	—	3B6Ah	—	3B4Ah	—	3B2Ah	—	3B0Ah	—
3BE9h	DMA1BUF	3BC9h	DMA2BUF	3BA9h	—	3B89h	—	3B69h	—	3B49h	—	3B29h	—	3B09h	—
3BE8h	—	3BC8h	—	3BA8h	—	3B88h	—	3B68h	—	3B48h	—	3B28h	—	3B08h	—
3BE7h	—	3BC7h	—	3BA7h	—	3B87h	—	3B67h	—	3B47h	—	3B27h	—	3B07h	—
3BE6h	—	3BC6h	—	3BA6h	—	3B86h	—	3B66h	—	3B46h	—	3B26h	—	3B06h	—
3BE5h	—	3BC5h	—	3BA5h	—	3B85h	—	3B65h	—	3B45h	—	3B25h	—	3B05h	—
3BE4h	—	3BC4h	—	3BA4h	—	3B84h	—	3B64h	—	3B44h	—	3B24h	—	3B04h	—
3BE3h	—	3BC3h	—	3BA3h	—	3B83h	—	3B63h	—	3B43h	—	3B23h	—	3B03h	—
3BE2h	—	3BC2h	—	3BA2h	—	3B82h	—	3B62h	—	3B42h	—	3B22h	—	3B02h	—
3BE1h	—	3BC1h	—	3BA1h	—	3B81h	—	3B61h	—	3B41h	—	3B21h	—	3B01h	—
3BE0h	—	3BC0h	—	3BA0h	—	3B80h	—	3B60h	—	3B40h	—	3B20h	—	3B00h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

© 2017-2020 Microchip Technology Inc.

TABLE 4-9: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 58

3AFFh	—	3ADFh	SPI1SDIPPS	3ABFh	PPSLOCK	3A9Fh	—	3A7Fh	—	3A5Fh	—	3A3Fh	—	3A1Fh	RD7PPS(2)
3AFEh	—	3ADEh	SPI1SCKPPS	3ABEh	—(4)	3A9Eh	—	3A7Eh	—	3A5Eh	—	3A3Eh	—	3A1Eh	RD6PPS(2)
3AFDh	—	3ADDh	ADACTPPS	3ABDh	—	3A9Dh	—	3A7Dh	—	3A5Dh	—	3A3Dh	—	3A1Dh	RD5PPS(2)
3AFCh	—	3ADCh	CLCIN3PPS	3ABCh	—	3A9Ch	—	3A7Ch	—	3A5Ch	—	3A3Ch	—	3A1Ch	RD4PPS(2)
3AFBh	—	3ADBh	CLCIN2PPS	3ABBh	—	3A9Bh	—	3A7Bh	RD1I2C(2)	3A5Bh	RB2I2C	3A3Bh	—	3A1Bh	RD3PPS(2)
3AFAh	—	3ADAh	CLCIN1PPS	3ABAh	—	3A9Ah	—	3A7Ah	RD0I2C(2)	3A5Ah	RB1I2C	3A3Ah	—	3A1Ah	RD2PPS(2)
3AF9h	—	3AD9h	CLCIN0PPS	3AB9h	—	3A99h	—(4)	3A79h	—(4)	3A59h	—(4)	3A39h	—	3A19h	RD1PPS(2)
3AF8h	—	3AD8h	MD1SRCPPS	3AB8h	—	3A98h	—(4)	3A78h	—(4)	3A58h	—(4)	3A38h	—	3A18h	RD0PPS(2)
3AF7h	—	3AD7h	MD1CARHPPS	3AB7h	—	3A97h	—	3A77h	—	3A57h	IOCBF	3A37h	—	3A17h	RC7PPS
3AF6h	—	3AD6h	MD1CARLPPS	3AB6h	—	3A96h	—	3A76h	—	3A56h	IOCBN	3A36h	—	3A16h	RC6PPS
3AF5h	—	3AD5h	CWG3INPPS	3AB5h	—	3A95h	—	3A75h	—	3A55h	IOCBP	3A35h	—	3A15h	RC5PPS
3AF4h	—	3AD4h	CWG2INPPS	3AB4h	—	3A94h	INLVLF(3)	3A74h	INLVLD(2)	3A54h	INLVLB	3A34h	—	3A14h	RC4PPS
3AF3h	—	3AD3h	CWG1INPPS	3AB3h	—	3A93h	SLRCONF(3)	3A73h	SLRCOND(2)	3A53h	SLRCONB	3A33h	—	3A13h	RC3PPS
3AF2h	—	3AD2h	SMT1SIGPPS	3AB2h	—	3A92h	ODCONF(3)	3A72h	ODCOND(2)	3A52h	ODCONB	3A32h	—	3A12h	RC2PPS
3AF1h	—	3AD1h	SMT1WINPPS	3AB1h	—	3A91h	WPUF(3)	3A71h	WPUD(2)	3A51h	WPUB	3A31h	—	3A11h	RC1PPS
3AF0h	—	3AD0h	CCP4PPS	3AB0h	—	3A90h	ANSELF(3)	3A70h	ANSELD(2)	3A50h	ANSELB	3A30h	—	3A10h	RC0PPS
3AEFh	—	3ACFh	CCP3PPS	3AAFh	—	3A8Fh	—	3A6Fh	—	3A4Fh	—	3A2Fh	RF7PPS(3)	3A0Fh	RB7PPS
3AEEh	—	3ACEh	CCP2PPS	3AAEh	—	3A8Eh	—	3A6Eh	—	3A4Eh	—	3A2Eh	RF6PPS(3)	3A0Eh	RB6PPS
3AEDh	—	3ACDh	CCP1PPS	3AADh	—	3A8Dh	—	3A6Dh	—	3A4Dh	—	3A2Dh	RF5PPS(3)	3A0Dh	RB5PPS
3AECh	—	3ACCh	T6INPPS	3AACh	—	3A8Ch	—	3A6Ch	—	3A4Ch	—	3A2Ch	RF4PPS(3)	3A0Ch	RB4PPS
3AEBh	—	3ACBh	T4INPPS	3AABh	—	3A8Bh	—	3A6Bh	RC4I2C	3A4Bh	—	3A2Bh	RF3PPS(3)	3A0Bh	RB3PPS
3AEAh	—	3ACAh	T2INPPS	3AAAh	—	3A8Ah	—	3A6Ah	RC3I2C	3A4Ah	—	3A2Ah	RF2PPS(3)	3A0Ah	RB2PPS
3AE9h	U2CTSPPS	3AC9h	T5GPPS	3AA9h	—	3A89h	—(4)	3A69h	—(4)	3A49h	—(4)	3A29h	RF1PPS(3)	3A09h	RB1PPS
3AE8h	U2RXPPS	3AC8h	T5CKIPPS	3AA8h	—	3A88h	—(4)	3A68h	—(4)	3A48h	—(4)	3A28h	RF0PPS(3)	3A08h	RB0PPS
3AE7h	—	3AC7h	T3GPPS	3AA7h	—	3A87h	IOCEF	3A67h	IOCCF	3A47h	IOCAF	3A27h	—	3A07h	RA7PPS
3AE6h	U1CTSPPS	3AC6h	T3CKIPPS	3AA6h	—	3A86h	IOCEN	3A66h	IOCCN	3A46h	IOCAN	3A26h	—	3A06h	RA6PPS
3AE5h	U1RXPPS	3AC5h	T1GPPS	3AA5h	—	3A85h	IOCEP	3A65h	IOCCP	3A45h	IOCAP	3A25h	—	3A05h	RA5PPS
3AE4h	I2C2SDAPPS	3AC4h	T1CKIPPS	3AA4h	—	3A84h	INLVLE	3A64h	INLVLC	3A44h	INLVLA	3A24h	—	3A04h	RA4PPS
3AE3h	I2C2SCLPPS	3AC3h	T0CKIPPS	3AA3h	—	3A83h	SLRCONE(2)	3A63h	SLRCONC	3A43h	SLRCONA	3A23h	—	3A03h	RA3PPS
3AE2h	I2C1SDAPPS	3AC2h	INT2PPS	3AA2h	—	3A82h	ODCONE(2)	3A62h	ODCONC	3A42h	ODCONA	3A22h	RE2PPS(2)	3A02h	RA2PPS
3AE1h	I2C1SCLPPS	3AC1h	INT1PPS	3AA1h	—	3A81h	WPUE	3A61h	WPUC	3A41h	WPUA	3A21h	RE1PPS(2)	3A01h	RA1PPS
3AE0h	SPI1SSPPS	3AC0h	INT0PPS	3AA0h	—	3A80h	ANSELE(2)	3A60h	ANSELC	3A40h	ANSELA	3A20h	RE0PPS(2)	3A00h	RA0PPS

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

4: Reserved, maintain as '0'.

© 2017-2020 Microchip Technology Inc.

TABLE 4-10: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 57

39FFh	—	39DFh	OSCFRQ	39BFh	—	399Fh	—	397Fh	—	395Fh	WDTU	393Fh	—	391Fh	—
39FEh	—	39DEh	OSCTUNE	39BEh	—	399Eh	—	397Eh	—	395Eh	WDTH	393Eh	—	391Eh	—
39FDh	—	39DDh	OSCEN	39BDh	—	399Dh	—	397Dh	SCANTRIG	395Dh	WDTL	393Dh	—	391Dh	—
39FCh	—	39DCh	OSCSTAT	39BCh	—	399Ch	—	397Ch	SCANCON0	395Ch	WDTCON1	393Ch	—	391Ch	—
39FBh	—	39DBh	OSCCON3	39BBh	—	399Bh	—	397Bh	SCANHADRU	395Bh	WDTCON0	393Bh	—	391Bh	—
39FAh	—	39DAh	OSCCON2	39BAh	—	399Ah	PIE10	397Ah	SCANHADRH	395Ah	—	393Ah	—	391Ah	—
39F9h	—	39D9h	OSCCON1	39B9h	—	3999h	PIE9	3979h	SCANHADRL	3959h	—	3939h	—	3919h	—
39F8h	—	39D8h	CPUDOZE	39B8h	—	3998h	PIE8	3978h	SCANLADRU	3958h	—	3938h	—	3918h	—
39F7h	SCANPR	39D7h	—	39B7h	—	3997h	PIE7	3977h	SCANLADRH	3957h	—	3937h	—	3917h	—
39F6h	—	39D6h	—	39B6h	—	3996h	PIE6	3976h	SCANLADRL	3956h	—	3936h	—	3916h	—
39F5h	—	39D5h	—	39B5h	—	3995h	PIE5	3975h	—	3955h	—	3935h	—	3915h	—
39F4h	DMA2PR	39D4h	—	39B4h	—	3994h	PIE4	3974h	—	3954h	—	3934h	—	3914h	—
39F3h	DMA1PR	39D3h	—	39B3h	—	3993h	PIE3	3973h	—	3953h	—	3933h	—	3913h	—
39F2h	MAINPR	39D2h	—	39B2h	—	3992h	PIE2	3972h	—	3952h	—	3932h	—	3912h	—
39F1h	ISRPR	39D1h	VREGCON(1)	39B1h	—	3991h	PIE1	3971h	—	3951h	—	3931h	—	3911h	—
39F0h	—	39D0h	BORCON	39B0h	—	3990h	PIE0	3970h	—	3950h	—	3930h	—	3910h	—
39EFh	PRLOCK	39CFh	—	39AFh	—	398Fh	—	396Fh	—	394Fh	—	392Fh	—	390Fh	—
39EEh	—	39CEh	—	39AEh	—	398Eh	—	396Eh	—	394Eh	—	392Eh	—	390Eh	—
39EDh	—	39CDh	—	39ADh	—	398Dh	—	396Dh	—	394Dh	—	392Dh	—	390Dh	—
39ECh	—	39CCh	—	39ACh	—	398Ch	—	396Ch	—	394Ch	—	392Ch	—	390Ch	—
39EBh	—	39CBh	—	39ABh	—	398Bh	—	396Bh	—	394Bh	—	392Bh	—	390Bh	—
39EAh	—	39CAh	—	39AAh	PIR10	398Ah	IPR10	396Ah	—	394Ah	—	392Ah	—	390Ah	—
39E9h	—	39C9h	—	39A9h	PIR9	3989h	IPR9	3969h	CRCCON1	3949h	—	3929h	—	3909h	—
39E8h	—	39C8h	—	39A8h	PIR8	3988h	IPR8	3968h	CRCCON0	3948h	—	3928h	—	3908h	—
39E7h	—	39C7h	PMD7	39A7h	PIR7	3987h	IPR7	3967h	CRCXORH	3947h	—	3927h	—	3907h	—
39E6h	NVMCON2	39C6h	PMD6	39A6h	PIR6	3986h	IPR6	3966h	CRCXORL	3946h	—	3926h	—	3906h	—
39E5h	NVMCON1	39C5h	PMD5	39A5h	PIR5	3985h	IPR5	3965h	CRCSHIFTH	3945h	—	3925h	—	3905h	—
39E4h	—	39C4h	PMD4	39A4h	PIR4	3984h	IPR4	3964h	CRCSHIFTL	3944h	—	3924h	—	3904h	—
39E3h	NVMDAT	39C3h	PMD3	39A3h	PIR3	3983h	IPR3	3963h	CRCACCH	3943h	—	3923h	—	3903h	—
39E2h	—	39C2h	PMD2	39A2h	PIR2	3982h	IPR2	3962h	CRCACCL	3942h	—	3922h	—	3902h	—
39E1h	NVMADRH(4)	39C1h	PMD1	39A1h	PIR1	3981h	IPR1	3961h	CRCDATH	3941h	—	3921h	—	3901h	—
39E0h	NVMADRL	39C0h	PMD0	39A0h	PIR0	3980h	IPR0	3960h	CRCDATL	3940h	—	3920h	—	3900h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

4: Unimplemented in PIC18(L)F45/55K42.

© 2017-2020 Microchip Technology Inc.

TABLE 4-11: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 56

38FFh	—	38DFh	—	38BFh	—	389Fh	IVTADU	387Fh	—	385Fh	—	383Fh	—	381Fh	—
38FEh	—	38DEh	—	38BEh	—	389Eh	IVTADH	387Eh	—	385Eh	—	383Eh	—	381Eh	—
38FDh	—	38DDh	—	38BDh	—	389Dh	IVTADL	387Dh	—	385Dh	—	383Dh	—	381Dh	—
38FCh	—	38DCh	—	38BCh	—	389Ch	—	387Ch	—	385Ch	—	383Ch	—	381Ch	—
38FBh	—	38DBh	—	38BBh	—	389Bh	—	387Bh	—	385Bh	—	383Bh	—	381Bh	—
38FAh	—	38DAh	—	38BAh	—	389Ah	—	387Ah	—	385Ah	—	383Ah	—	381Ah	—
38F9h	—	38D9h	—	38B9h	—	3899h	—	3879h	—	3859h	—	3839h	—	3819h	—
38F8h	—	38D8h	—	38B8h	—	3898h	—	3878h	—	3858h	—	3838h	—	3818h	—
38F7h	—	38D7h	—	38B7h	—	3897h	—	3877h	—	3857h	—	3837h	—	3817h	—
38F6h	—	38D6h	—	38B6h	—	3896h	—	3876h	—	3856h	—	3836h	—	3816h	—
38F5h	—	38D5h	—	38B5h	—	3895h	—	3875h	—	3855h	—	3835h	—	3815h	—
38F4h	—	38D4h	—	38B4h	—	3894h	—	3874h	—	3854h	—	3834h	—	3814h	—
38F3h	—	38D3h	—	38B3h	—	3893h	—	3873h	—	3853h	—	3833h	—	3813h	—
38F2h	—	38D2h	—	38B2h	—	3892h	—	3872h	—	3852h	—	3832h	—	3812h	—
38F1h	—	38D1h	—	38B1h	—	3891h	—	3871h	—	3851h	—	3831h	—	3811h	—
38F0h	—	38D0h	—	38B0h	—	3890h	PRODH_SHAD	3870h	—	3850h	—	3830h	—	3810h	—
38EFh	—	38CFh	—	38AFh	—	388Fh	PRODL_SHAD	386Fh	—	384Fh	—	382Fh	—	380Fh	—
38EEh	—	38CEh	—	38AEh	—	388Eh	FSR2H_SHAD	386Eh	—	384Eh	—	382Eh	—	380Eh	—
38EDh	—	38CDh	—	38ADh	—	388Dh	FSR2L_SHAD	386Dh	—	384Dh	—	382Dh	—	380Dh	—
38ECh	—	38CCh	—	38ACh	—	388Ch	FSR1H_SHAD	386Ch	—	384Ch	—	382Ch	—	380Ch	—
38EBh	—	38CBh	—	38ABh	—	388Bh	FSR1L_SHAD	386Bh	—	384Bh	—	382Bh	—	380Bh	—
38EAh	—	38CAh	—	38AAh	—	388Ah	FSR0H_SHAD	386Ah	—	384Ah	—	382Ah	—	380Ah	—
38E9h	—	38C9h	—	38A9h	—	3889h	FSR0L_SHAD	3869h	—	3849h	—	3829h	—	3809h	—
38E8h	—	38C8h	—	38A8h	—	3888h	PCLATU_SHAD	3868h	—	3848h	—	3828h	—	3808h	—
38E7h	—	38C7h	—	38A7h	—	3887h	PCLATH_SHAD	3867h	—	3847h	—	3827h	—	3807h	—
38E6h	—	38C6h	—	38A6h	—	3886h	BSR_SHAD	3866h	—	3846h	—	3826h	—	3806h	—
38E5h	—	38C5h	—	38A5h	—	3885h	WREG_SHAD	3865h	—	3845h	—	3825h	—	3805h	—
38E4h	—	38C4h	—	38A4h	—	3884h	STATUS_SHAD	3864h	—	3844h	—	3824h	—	3804h	—
38E3h	—	38C3h	—	38A3h	—	3883h	SHADCON	3863h	—	3843h	—	3823h	—	3803h	—
38E2h	—	38C2h	—	38A2h	—	3882h	BSR_CSHAD	3862h	—	3842h	—	3822h	—	3802h	—
38E1h	—	38C1h	—	38A1h	—	3881h	WREG_CSHAD	3861h	—	3841h	—	3821h	—	3801h	—
38E0h	—	38C0h	—	38A0h	—	3880h	STATUS_CSHAD	3860h	—	3840h	—	3820h	—	3800h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

© 2017-2020 Microchip Technology Inc.

4.5.5 STATUS REGISTER

The STATUS register, shown in Register 4-2, contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the STATUS register is updated according to the instruction performed. Therefore, the result of an instruction with the STATUS register as its destination may be different than intended. As an example, `CLRF STATUS` will set the Z bit and leave the remaining Status bits unchanged (‘`0uuu u1uu`’).

It is recommended that only `BCF`, `BSF`, `SWAPF`, `MOVFF`, `MOVWF` and `MOVFFL` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register.

For other instructions that do not affect Status bits, see the instruction set summaries in **Section 41.2 “Extended Instruction Set”** and Table 41-3.

Note:	The C and DC bits operate as the borrow and digit borrow bits, respectively, in subtraction.

4.5.6 CALL SHADOW REGISTER

When `CALL` instruction is used, the WREG, BSR and STATUS are automatically saved in hardware and can be accessed using the WREG_CSHAD, BSR_CSHAD and STATUS_CSHAD registers.

Note:	The contents of these registers should be handled correctly to avoid erroneous code execution.

4.6 Register Definitions: Status Registers

REGISTER 4-2: STATUS: STATUS REGISTER

U-0	R-1/q	R-1/q	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	$\overline{TO}$	$\overline{PD}$	N	OV	Z	DC	C
bit 7							bit 0

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

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **$\overline{TO}$:** Time-Out bit
1 = Set at power-up or by execution of CLRWDT or SLEEP instruction
0 = A WDT time-out occurred

bit 5 **$\overline{PD}$:** Power-Down bit
1 = Set at power-up or by execution of CLRWDT instruction
0 = Set by execution of the SLEEP instruction

bit 4 **N:** Negative bit used for signed arithmetic (2’s complement); indicates if the result is negative, (ALU MSb = 1).
1 = The result is negative
0 = The result is positive

bit 3 **OV:** Overflow bit used for signed arithmetic (2’s complement); indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.
1 = Overflow occurred for current signed arithmetic operation
0 = No overflow occurred

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1,2)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note 1: For $\overline{\text{Borrow}}$, the polarity is reversed. A subtraction is executed by adding the two’s complement of the second operand.
2: For Rotate (RRF, RLF) instructions, this bit is loaded with either the high or low-order bit of the Source register.

4.7 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 4.8 "Data Memory and the Extended Instruction Set"** for more information.

While the program memory can be addressed in only one way – through the program counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = `1`). Its operation is discussed in detail in **Section 4.8.1 "Indexed Addressing with Literal Offset"**.

4.7.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include `SLEEP`, `RESET` and `DAW`.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include `ADDLW` and `MOVLW`, which respectively, add or move a literal value to the W register. Other examples include `CALL` and `GOTO`, which include a 20-bit program memory address.

4.7.2 DIRECT ADDRESSING

Direct addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 4.5.2 "General Purpose Register File"**) or a location in the Access Bank (**Section 4.5.4 "Access Bank"**) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 4.5.1 "Bank Select Register (BSR)"**) are used with the address to determine the complete 14-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as `MOVFFL`, include the entire 14-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

4.7.3 INDIRECT ADDRESSING

Indirect addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations which are to be read or written. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for indirect addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 4-6.

EXAMPLE 4-6: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR  FSR0, 100h ;
NEXT    CLRF  POSTINC0   ; Clear INDF
                         ; register then
                         ; inc pointer
        BTFSS FSR0H, 1   ; All done with
                         ; Bank1?
        BRA   NEXT       ; NO, clear next
CONTINUE                 ; YES, continue
```

4.7.3.1 FSR Registers and the INDF Operand

At the core of indirect addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. Each FSR pair holds a 14-bit value, therefore, the two upper bits of the FSRnH register are not used. The 14-bit FSR value can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as "virtual" registers; they are mapped in the SFR space but are not physically implemented. Reading or writing to a particular INDF register actually accesses the data addressed by its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L. Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction's target. The INDF operand is just a convenient way of using the pointer.

Because indirect addressing uses a full 14-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

© 2017-2020 Microchip Technology Inc.

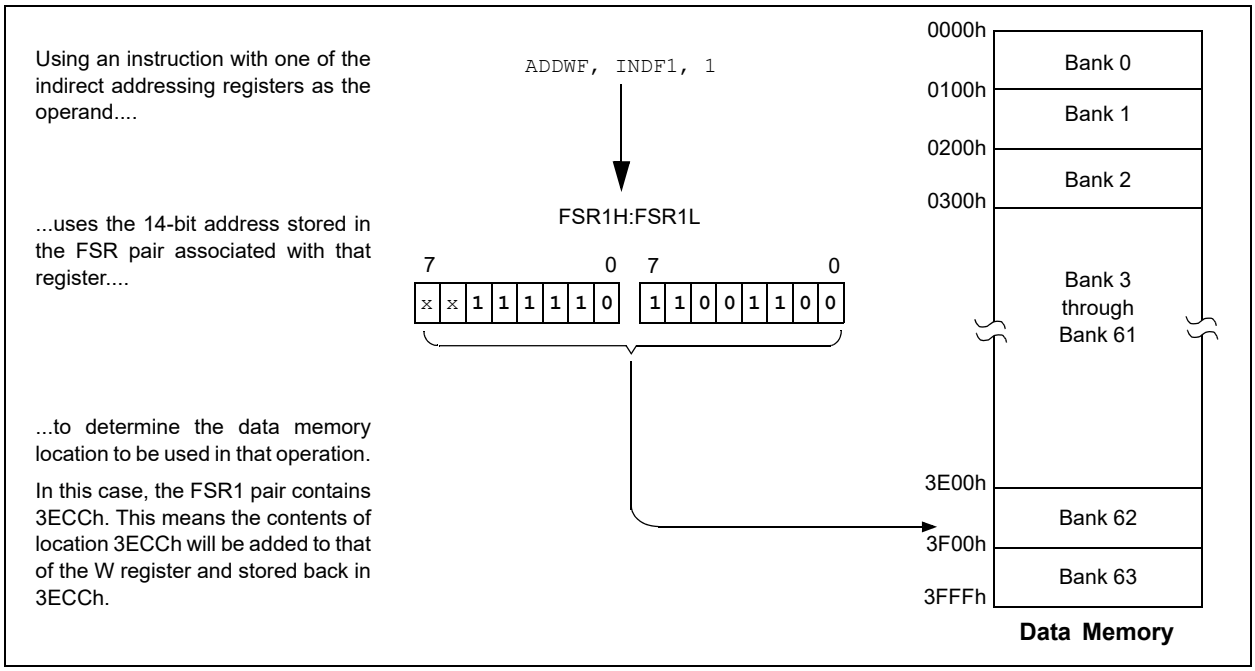
4.7.3.2 FSR Registers, POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers which cannot be directly read or written. Accessing these registers actually accesses the location to which the associated FSR register pair points, and also performs a specific action on the FSR value. They are:

- POSTDEC: accesses the location to which the FSR points, then automatically decrements the FSR by 1 afterwards
- POSTINC: accesses the location to which the FSR points, then automatically increments the FSR by 1 afterwards
- PREINC: automatically increments the FSR by 1, then uses the location to which the FSR points in the operation
- PLUSW: adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the location to which the result points in the operation.

In this context, accessing an INDF register uses the value in the associated FSR register without changing it. Similarly, accessing a PLUSW register gives the FSR value an offset by that in the W register; however, neither W nor the FSR is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR register.

FIGURE 4-6: INDIRECT ADDRESSING



Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, rollovers of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

The PLUSW register can be used to implement a form of indexed addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

4.7.3.3 Operations by FSRs on FSRs

Indirect addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains 3FE7h, the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a `NOP`.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to either the INDF2 or POSTDEC2 register will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses indirect addressing.

Similarly, operations by indirect addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

4.8 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = `1`) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect addressing with FSR0 and FSR1 also remain unchanged.

4.8.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of indirect addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of indexed addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced (‘a’ = `0`) and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in direct addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

4.8.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use direct addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘`1`’), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in Figure 4-7.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 41.2.1 “Extended Instruction Syntax”**.

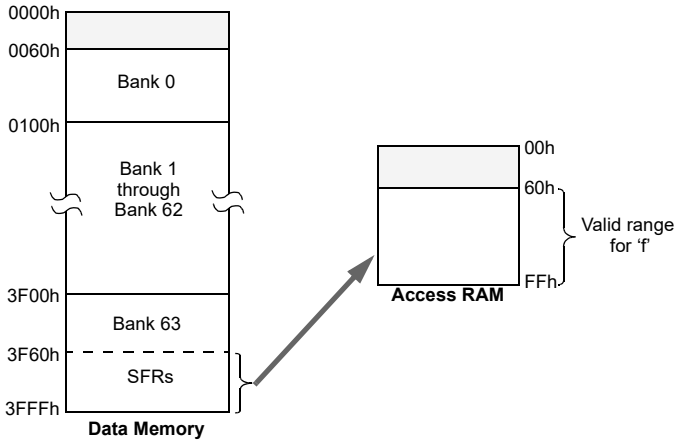
FIGURE 4-7: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

EXAMPLE INSTRUCTION: `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

When 'a' = 0 and f ≥ 60h:

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations 3F60h to 3FFFh (Bank 63) of data memory.

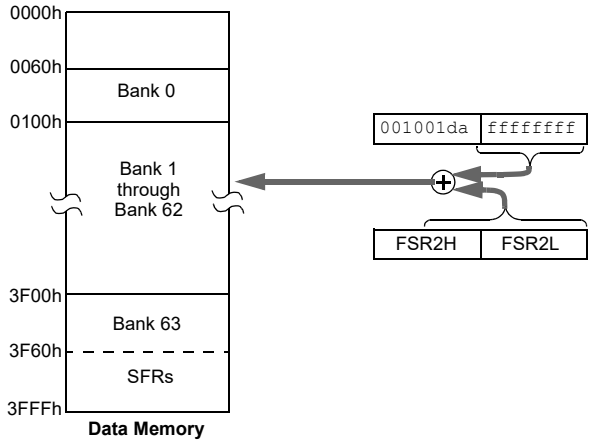
Locations below 60h are not available in this Addressing mode.



When 'a' = 0 and f ≤ 5Fh:

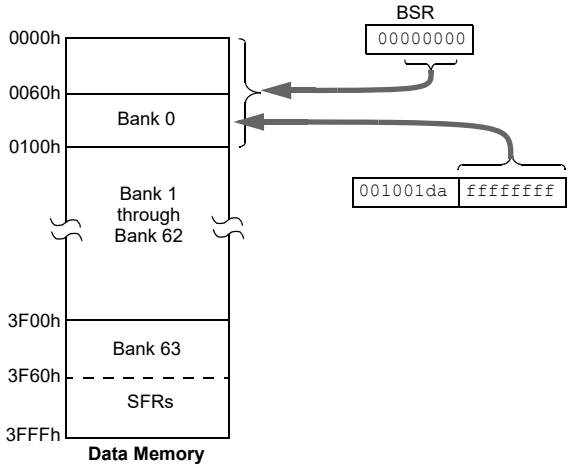
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where 'k' is the same as 'f'.



When 'a' = 1 (all values of f):

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 63 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



4.8.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

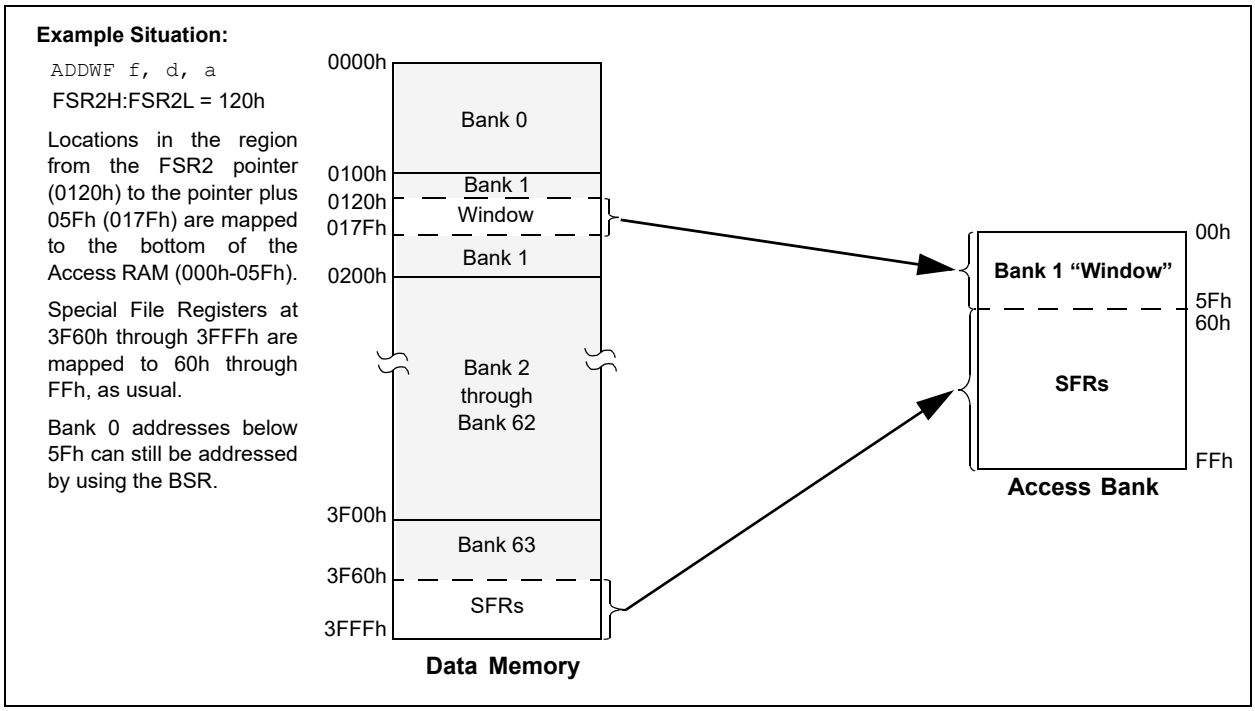
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom section of Bank 0, this mode maps the contents from a user defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 4.5.4 “Access Bank”**). An example of Access Bank remapping in this addressing mode is shown in Figure 4-8.

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is ‘1’) will continue to use direct addressing as before.

4.9 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds eight additional commands to the existing PIC18 instruction set. These instructions are executed as described in **Section 41.2 “Extended Instruction Set”**.

FIGURE 4-8: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING



REGISTER 5-5: CONFIGURATION WORD 3L (30 0004h)

U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	WDTE[1:0]		WDTCPS[4:0]				
bit 7							bit 0

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

bit 7 **Unimplemented:** Read as ‘1’

bit 6-5 **WDTE[1:0]:** WDT Operating Mode bits
00 = WDT is disabled, SWDTEN is ignored
01 = WDT is enabled/disabled by the SWDTEN bit in WDTCON0
10 = WDT is enabled while Sleep = 0, suspended when Sleep = 1; SWDTEN is ignored
11 = WDT is enabled regardless of Sleep; SWDTEN is ignored

bit 4-0 **WDTCPS[4:0]:** WDT Period Select bits

WDTCPS[4:0]	WDTPS at POR				Software Control of WDTPS?
	Value	Divider Ratio		Typical Time-out (FIN = 31 kHz)	
00000	00000	1:32	2^5	1 ms	No
00001	00001	1:64	2^6	2 ms	
00010	00010	1:128	2^7	4 ms	
00011	00011	1:256	2^8	8 ms	
00100	00100	1:512	2^9	16 ms	
00101	00101	1:1024	2^{10}	32 ms	
00110	00110	1:2048	2^{11}	64 ms	
00111	00111	1:4096	2^{12}	128 ms	
01000	01000	1:8192	2^{13}	256 ms	
01001	01001	1:16384	2^{14}	512 ms	
01010	01010	1:32768	2^{15}	1s	
01011	01011	1:65536	2^{16}	2s	
01100	01100	1:131072	2^{17}	4s	
01101	01101	1:262144	2^{18}	8s	
01110	01110	1:524299	2^{19}	16s	
01111	01111	1:1048576	2^{20}	32s	
10000	10000	1:2097152	2^{21}	64s	
10001	10001	1:4194304	2^{22}	128s	
10010	10010	1:8388608	2^{23}	256s	
10011 ... 11110	10011 ... 11110	1:32	2^5	1 ms	No
11111	01011	1:65536	2^{16}	2s	Yes

REGISTER 5-6: CONFIGURATION WORD 3H (30 0005h)

U-1	U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	WDTCCS[2:0]			WDTCWS[2:0]		
bit 7							bit 0

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

bit 7-6 **Unimplemented:** Read as ‘1’

bit 5-3 **WDTCCS[2:0]:** WDT Input Clock Selector bits

If WDTE[1:0] Fuses = 2’b00:

These bits are ignored.

Otherwise:

000 = WDT reference clock is the 31.0 kHz LFINTOSC
001 = WDT reference clock is the 31.25 kHz MFINTOSC
010 = WDT reference clock is SOSC
011 = Reserved (default to LFINTOSC)
•
•
110 = Reserved (default to LFINTOSC)
111 = Software control

bit 2-0 **WDTCWS[2:0]:** WDT Window Select bits

WDTCWS[2:0]	Window at POR			Software Control of Window	Keyed Access Required?
	Value	Window Delay Percent of Time	Window Opening Percent of Time		
000	000	87.5	12.5	No	Yes
001	001	75	25		
010	010	62.5	37.5		
011	011	50	50		
100	100	37.5	62.5		
101	101	25	75		
110	111	n/a	100		
111	111	n/a	100	Yes	No

REGISTER 5-7: CONFIGURATION WORD 4L (30 0006h)

R/W-1	U-1	U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{WRTAPP}}$ (1)	—	—	$\overline{\text{SAFEN}}$ (1)	$\overline{\text{BBEN}}$ (1)	BBSIZE[2:0] (2)		
bit 7							bit 0

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

bit 7 **$\overline{\text{WRTAPP}}$:** Application Block Write Protection bit(1)
1 = Application Block is NOT write-protected
0 = Application Block is write-protected

bit 6-5 **Unimplemented:** Read as ‘1’

bit 4 **$\overline{\text{SAFEN}}$:** Storage Area Flash Enable bit(1)
1 = SAF is disabled
0 = SAF is enabled

bit 3 **$\overline{\text{BBEN}}$:** Boot Block Enable bit(1)
1 = Boot Block disabled
0 = Boot Block enabled

bit 2-0 **BBSIZE[2:0]:** Boot Block Size Selection bits(2)
Refer to Table 5-1.

Note 1: Bits are implemented as sticky bits. Once protection is enabled through ICSP™ or a self-write, it can only be reset through a Bulk Erase.
2: BBSIZE[2:0] bits can only be changed when $\overline{\text{BBEN}}$ = 1. Once $\overline{\text{BBEN}}$ = 0, BBSIZE[2:0] can only be changed through a Bulk Erase.

TABLE 5-1: BOOT BLOCK SIZE BITS

$\overline{\text{BBEN}}$	BBSIZE[2:0]	Boot Block Size (words)	END_ADDRESS_BOOT	Device Size(1)		
				16k	32k	64k
1	xxx	0	—	X	X	X
0	111	512	00 03FFh	X	X	X
0	110	1024	00 07FFh	X	X	X
0	101	2048	00 0FFFh	X	X	X
0	100	4096	00 1FFFh	X	X	X
0	011	8192	00 3FFFh	X	X	X
0	010	16384	00 7FFFh	—	X	X
0	001	32768	00 FFFFh		Note 2	X
0	000	32768	00 FFFFh	—	—	—

Note 1: For each device, the quoted device size specification is listed in Table 4-1.
2: The maximum boot block size is half the user program memory size. All selections higher than the maximum size default to maximum boot block size of half PFM. For example, all settings of BBSIZE = 000 through BBSIZE = 011, default to a boot block size of 8 kW on a 16 kW device.

REGISTER 5-8: CONFIGURATION WORD 4H (30 0007h)

U-1	U-1	R/W-1	U-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	LVP(2)	—	$\overline{\text{WRTSAF}}$ (1,3)	$\overline{\text{WRTD}}$ (1,4)	$\overline{\text{WRTC}}$ (1)	$\overline{\text{WRTB}}$(1,5)
bit 7							bit 0

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

bit 7-6 **Unimplemented:** Read as ‘1’

bit 5 **LVP:** Low-Voltage Programming Enable bit(2)
1 = Low-voltage programming enabled. $\overline{\text{MCLR}}$/VPP pin function is $\overline{\text{MCLR}}$. MCLRE (Register 5-3) is ignored.
0 = HV on $\overline{\text{MCLR}}$/VPP must be used for programming.

bit 4 **Unimplemented:** Read as ‘1’

bit 3 **$\overline{\text{WRTSAF}}$:** Storage Area Flash (SAF) Write Protection bit(1,3)
1 = SAF is NOT write-protected
0 = SAF is write-protected

bit 2 **$\overline{\text{WRTD}}$:** Data EEPROM Write Protection bit(1,4)
1 = Data EEPROM NOT write-protected
0 = Data EEPROM write-protected

bit 1 **$\overline{\text{WRTC}}$:** Configuration Register Write Protection bit(1)
1 = Configuration Register NOT write-protected
0 = Configuration Register write-protected

bit 0 **$\overline{\text{WRTB}}$:** Boot Block Write Protection bit(1,5)
1 = Boot Block NOT write-protected
0 = Boot Block write-protected

Note 1: Bits are implemented as sticky bits. Once protection is enabled through ICSP or a self write, it can only be reset through a Bulk Erase.

2: The LVP bit cannot be written (to zero) while operating from the LVP programming interface. The purpose of this rule is to prevent the user from dropping out of LVP mode while programming from LVP mode, or accidentally eliminating LVP mode from the configuration state.

3: Unimplemented if SAF is not present and only applicable if $\overline{\text{SAFEN}}$ = 0.

4: Unimplemented if data EEPROM is not present.

5: Only applicable if $\overline{\text{BBEN}}$ = 0.

REGISTER 5-9: CONFIGURATION WORD 5L (30 0008h)

U-1	U-1	U-1	U-1	U-1	U-1	U-1	R/W-1
—	—	—	—	—	—	—	$\overline{CP}$
bit 7							bit 0

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

bit 7-1 **Unimplemented:** Read as ‘1’

bit 0 **$\overline{CP}$:** User Program Flash Memory and Data EEPROM Code Protection bit
1 = User Program Flash Memory and Data EEPROM code protection is disabled
0 = User Program Flash Memory and Data EEPROM code protection is enabled

REGISTER 5-10: CONFIGURATION WORD 5H (30 0009h)

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 7							bit 0

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

bit 7-0 **Unimplemented:** Read as ‘1’

TABLE 5-2: SUMMARY OF CONFIGURATION WORDS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
30 0000h	CONFIG1L	—	RSTOSC[2:0]			—	FEXTOSC[2:0]			1111 1111
30 0001h	CONFIG1H	—	—	FCMEN	—	CSWEN	—	PR1WAY	$\overline{CLKOUTEN}$	1111 1111
30 0002h	CONFIG2L	BOREN[1:0]		$\overline{LPBOREN}$	IVT1WAY	MVECEN	PWRTS[1:0]		MCLRE	1111 1111
30 0003h	CONFIG2H	$\overline{XINST}$	—	$\overline{DEBUG}$	STVREN	PPS1WAY	$\overline{ZCD}$	BORV[1:0]		1111 1111
30 0004h	CONFIG3L	—	WDTE[1:0]		WDTCPS[4:0]					1111 1111
30 0005h	CONFIG3H	—	—	WDTCCS[2:0]			WDTCWS[2:0]			1111 1111
30 0006h	CONFIG4L	$\overline{WRTAPP}$	—	—	$\overline{SAFEN}$	$\overline{BBEN}$	BBSIZE[2:0]			1111 1111
30 0007h	CONFIG4H	—	—	LVP	—	$\overline{WRTSAF}$	$\overline{WRTD}$	$\overline{WRTC}$	$\overline{WRTB}$	1111 1111
30 0008h	CONFIG5L	—	—	—	—	—	—	—	$\overline{CP}$	1111 1111
30 0009h	CONFIG5H	—	—	—	—	—	—	—	—	1111 1111

5.3 Code Protection

Code protection allows the device to be protected from external access. Program memory protection and data memory are controlled through the $\overline{CP}$ Configuration bit. Internal access to the program memory is unaffected by code protection setting.

The entire program memory space and Data EEPROM is protected from external reads and writes by the CP bit in Configuration Words. When CP = 0, external reads and writes of memory are inhibited and a read will return all '0's. The CPU can continue to read program memory and data EEPROM, regardless of the protection bit settings. Self-writing the program memory or Data EEPROM is dependent upon the write protection settings.

5.4 User ID

Eight words in the memory space (200000h-20000Fh) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See **Section 13.2 "Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations. For more information on checksum calculation, see the "PIC18(L)F26/27/45/46/47/55/56/57K42 *Memory Programming Specification*" (DS40001886).

7.2 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (ECH, ECM, ECL mode), quartz crystal resonators or ceramic resonators (LP, XT and HS modes).

Internal clock sources are contained within the oscillator module. The internal oscillator block has two internal oscillators that are used to generate internal system clock sources. The High-Frequency Internal Oscillator (HFINTOSC) can produce 1, 2, 4, 8, 12, 16, 32, 48 and 64 MHz clock. The frequency can be controlled through the OSCFRQ register (Register 7-5). The Low-Frequency Internal Oscillator (LFINTOSC) generates a fixed 31 kHz frequency.

A 4x PLL is provided that can be used with an external clock. When used with the HFINTOSC the 4x PLL has input frequency limitations.See **Section 7.2.1.4 "4x PLL"** for more details.

The system clock can be selected between external or internal clock sources via the NOSC bits in the OSCCON1 register. See **Section 7.3 "Clock Switching"** for additional information. The system clock can be made available on the OSC2/CLKOUT pin for any of the modes that do not use the OSC2 pin. The clock out functionality is governed by the $\overline{\text{CLKOUTEN}}$ bit in the CONFIG1H register (Register 5-2). If enabled, the clock out signal is always at a frequency of FOSC/4.

7.2.1 EXTERNAL CLOCK SOURCES

An external clock source can be used as the device system clock by performing one of the following actions:

- Program the RSTOSC[2:0] and FEXTOSC[2:0] bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Write the NOSC[2:0] and NDIV[3:0] bits in the OSCCON1 register to switch the system clock source.

See **Section 7.3 "Clock Switching"** for more information.

7.2.1.1 EC Mode

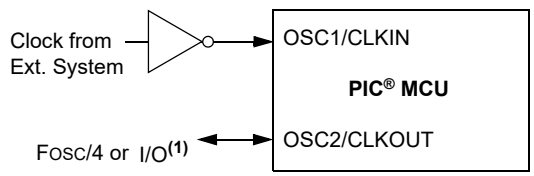
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. Figure 7-2 shows the pin connections for EC mode.

EC mode has three power modes to select from through Configuration Words:

- ECH – High power
- ECM – Medium power
- ECL – Low power

Refer to Table 44-8 for External Clock/Oscillator Timing Requirements. The Oscillator Start-up Timer (OST) is disabled when EC mode is selected. Therefore, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

FIGURE 7-2: EXTERNAL CLOCK (EC) MODE OPERATION



Note 1: Output depends upon $\overline{\text{CLKOUTEN}}$ bit of the Configuration Words (CONFIG1H).

7.2.1.2 LP, XT, HS Modes

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 7-3). The three modes select a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

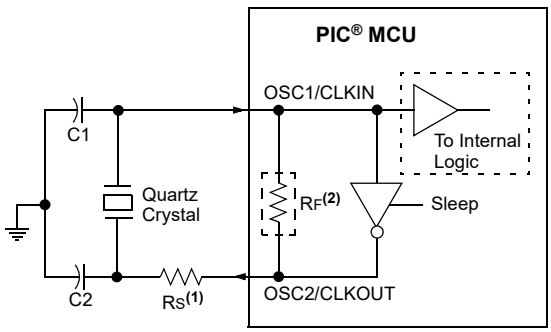
LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals), but can operate up to 100 kHz.

XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive crystals and resonators with a frequency range up to 4 MHz.

HS Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require an operating frequency up to 20 MHz.

Figure 7-3 and Figure 7-4 show typical circuits for quartz crystal and ceramic resonators, respectively.

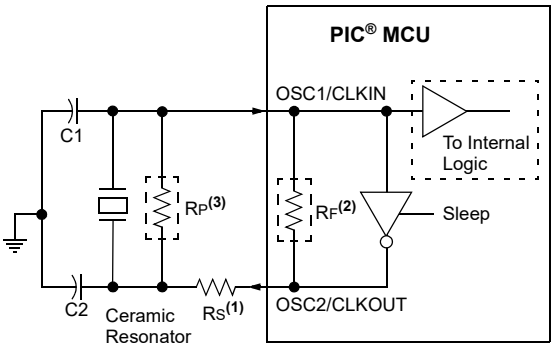
FIGURE 7-3: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



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).

FIGURE 7-4: 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.

7.2.1.3 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), Brown-out Reset (BOR), or a wake-up from Sleep. 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.

7.2.1.4 4x PLL

The oscillator module contains a 4x PLL that can be used with the external clock sources to provide a system clock source. The input frequency for the PLL must fall within specifications. See the PLL Clock Timing Specifications in Table 44-10.

The PLL can be enabled for use by one of two methods:

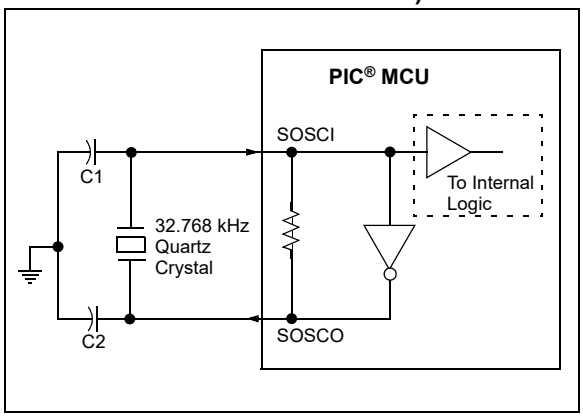
1. Program the RSTOSC bits in the Configuration Word 1 to `010` (enable EXTOSC with 4x PLL).
2. Write the NOSC bits in the OSCCON1 register to `010` (enable EXTOSC with 4x PLL).

7.2.1.5 Secondary Oscillator

The secondary oscillator is a separate oscillator block that can be used as an alternate system clock source. The secondary oscillator is optimized for 32.768 kHz, and can be used with an external crystal oscillator connected to the SOSCI and SOSCO device pins, or an external clock source connected to the SOSCIN pin. The secondary oscillator can be selected during run-time using clock switching. Refer to **Section 7.3 "Clock Switching"** for more information.

Two power modes are available for the secondary oscillator. These modes are selected with the SOSCPWR (OSCCON3[6]). Clearing this bit selects the lower Crystal Gain mode which provides lowest microcontroller power consumption. Setting this bit enables a higher Gain mode to support faster crystal start-up or crystals with higher ESR.

FIGURE 7-5: QUARTZ CRYSTAL OPERATION (SECONDARY OSCILLATOR)



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 PIC® and PIC® Devices*" (DS00826)
- AN849, "*Basic PIC® Oscillator Design*" (DS00849)
- AN943, "*Practical PIC® Oscillator Analysis and Design*" (DS00943)
- AN949, "*Making Your Oscillator Work*" (DS00949)
- TB097, "*Interfacing a Micro Crystal MS1V-T1K 32.768 kHz Tuning Fork Crystal to a PIC16F690/SS*" (DS91097)
- AN1288, "*Design Practices for Low-Power External Oscillators*" (DS01288)

7.2.2 INTERNAL CLOCK SOURCES

The device may be configured to use the internal oscillator block as the system clock by performing one of the following actions:

- Program the RSTOSC[2:0] bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Write the NOSC[2:0] bits in the OSCCON1 register to switch the system clock source to the internal oscillator during run-time. See **Section 7.3 "Clock Switching"** for more information.

In INTOSC mode, OSC1/CLKIN is available for general purpose I/O. OSC2/CLKOUT is available for general purpose I/O or CLKOUT.

The function of the OSC2/CLKOUT pin is determined by the $\overline{\text{CLKOUTEN}}$ bit in Configuration Words.

The internal oscillator block has two independent oscillators that can produce two internal system clock sources.

1. The **HFINTOSC** (High-Frequency Internal Oscillator) is factory-calibrated and operates from 1 to 64 MHz. The frequency of HFINTOSC can be selected through the OSCFRQ Frequency Selection register, and fine-tuning can be done via the OSCTUNE register.
2. The **LFINTOSC** (Low-Frequency Internal Oscillator) is factory-calibrated and operates at 31 kHz.

7.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision digitally-controlled internal clock source that produces a stable clock up to 64 MHz. The HFINTOSC can be enabled through one of the following methods:

- Programming the RSTOSC[2:0] bits in Configuration Word 1 to '110' (FOSC = 1 MHz) or '000' (FOSC = 64 MHz) to set the oscillator upon device Power-up or Reset.
- Write to the NOSC[2:0] bits of the OSCCON1 register during run-time. See **Section 7.3 "Clock Switching"** for more information.

The HFINTOSC frequency can be selected by setting the FRQ[3:0] bits of the OSCFRQ register.

The NDIV[3:0] bits of the OSCCON1 register allow for division of the HFINTOSC output from a range between 1:1 and 1:512.

7.2.2.2 MFINTOSC

The module provides two (500 kHz and 31.25 kHz) constant clock outputs. These clocks are digital divisors of the HFINTOSC clock. Dynamic divider logic is used to provide constant MFINTOSC clock rates for all settings of HFINTOSC.

The MFINTOSC cannot be used to drive the system but it is used to clock certain modules such as the Timers and WWDT.

7.2.2.3 Internal Oscillator Frequency Adjustment

The HFINTOSC is factory-calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register (Register 7-3).

The default value of the OSCTUNE register is 00h. The value is a 6-bit two's complement number. A value of 1Fh will provide an adjustment to the maximum frequency. A value of 20h will provide an adjustment to the minimum frequency.

When the OSCTUNE register is modified, the oscillator frequency will begin shifting to the new frequency. Code execution continues during this shift. There is no indication that the shift has occurred.

OSCTUNE **does not affect** the LFINTOSC frequency. Operation of features that depend on the LFINTOSC clock source frequency, such as the Power-up Timer (PWRT), WWDT, Fail-Safe Clock Monitor (FSCM) and peripherals, are *not* affected by the change in frequency.

7.2.2.4 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a factory-calibrated 31 kHz internal clock source.

The LFINTOSC is the frequency for the Power-up Timer (PWRT), Windowed Watchdog Timer (WWDT) and Fail-Safe Clock Monitor (FSCM). The LFINTOSC can also be used as the system clock, or as a clock or input source to other peripherals.

The LFINTOSC is enabled through one of the following methods:

- Programming the RSTOSC[2:0] bits of Configuration Word 1 to enable LFINTOSC.
- Write to the NOSC[2:0] bits of the OSCCON1 register during run-time. See **Section 7.3, Clock Switching** for more information.

7.2.2.5 ADCRC

The ADCRC is an oscillator dedicated to the ADC^2 module. The ADCRC oscillator can be manually enabled using the ADOEN bit of the OSCEN register. The ADCRC runs at a fixed frequency of 600 kHz. ADCRC is automatically enabled if it is selected as the clock source for the ADC^2 module.

7.2.2.6 Oscillator Status and Manual Enable

The Ready status of each oscillator (including the ADCRC oscillator) is displayed in OSCSTAT (Register 7-4). The oscillators (but not the PLL) may be explicitly enabled through OSCEN (Register 7-7).

7.2.2.7 HFOR and MFOR Bits

The HFOR and MFOR bits indicate that the HFINTOSC and MFINTOSC is ready. These clocks are always valid for use at all times, but only accurate after they are ready.

When a new value is loaded into the OSCFRQ register, the HFOR and MFOR bits will clear, and set again when the oscillator is ready. During pending OSCFRQ changes the MFINTOSC clock will stall at a high or a low state, until the HFINTOSC resumes operation.

7.3 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the New Oscillator Source (NOSC) bits of the OSCCON1 register. The following clock sources can be selected using the following:

- External oscillator
- Internal Oscillator Block (INTOSC)

Note:	The Clock Switch Enable bit in Configuration Word 1 can be used to enable or disable the clock switching capability. When cleared, the NOSC and NDIV bits cannot be changed by user software. When set, writing to NOSC and NDIV is allowed and would switch the clock frequency.

7.3.1 NEW OSCILLATOR SOURCE (NOSC) AND NEW DIVIDER SELECTION REQUEST (NDIV) BITS

The New Oscillator Source (NOSC) and New Divider Selection Request (NDIV) bits of the OSCCON1 register select the system clock source and frequency that are used for the CPU and peripherals.

When new values of NOSC and NDIV are written to OSCCON1, the current oscillator selection will continue to operate while waiting for the new clock source to indicate that it is stable and ready. In some cases, the newly requested source may already be in use, and is ready immediately. In the case of a divider-only change, the new and old sources are the same, so the old source will be ready immediately. The device may enter Sleep while waiting for the switch as described in **Section 7.3.2 "Clock Switch and Sleep"**.

When the new oscillator is ready, the New Oscillator Ready (NOSCR) bit of OSCCON3 and the Clock Switch Interrupt Flag (CSWIF) bit of the respective PIR register are set. If Clock Switch Interrupts are enabled (CSWIE = 1), an interrupt will be generated at that time. The Oscillator Ready (ORDY) bit of OSCCON3 can also be polled to determine when the oscillator is ready in lieu of an interrupt.

Note:	The CSWIF interrupt will not wake the system from Sleep.

If the Clock Switch Hold (CSWHOLD) bit of OSCCON3 is clear, the oscillator switch will occur when the New Oscillator is Ready bit (NOSCR) is set, and the interrupt (if enabled) will be serviced at the new oscillator setting.

If CSWHOLD is set, the oscillator switch is suspended, while execution continues using the current (old) clock source. When the NOSCR bit is set, software should:

- Set CSWHOLD = 0 so the switch can complete, or
- Copy COSC into NOSC to abandon the switch.

If Doze is in effect, the switch occurs on the next clock cycle, whether or not the CPU is operating during that cycle.

Changing the clock post-divider without changing the clock source (i.e., changing FOSC from 1 MHz to 2 MHz) is handled in the same manner as a clock source change, as described previously. The clock source will already be active, so the switch is relatively quick. CSWHOLD must be clear (CSWHOLD = 0) for the switch to complete.

The current COSC and CDIV are indicated in the OSCCON2 register up to the moment when the switch actually occurs, at which time OSCCON2 is updated and ORDY is set. NOSCR is cleared by hardware to indicate that the switch is complete.

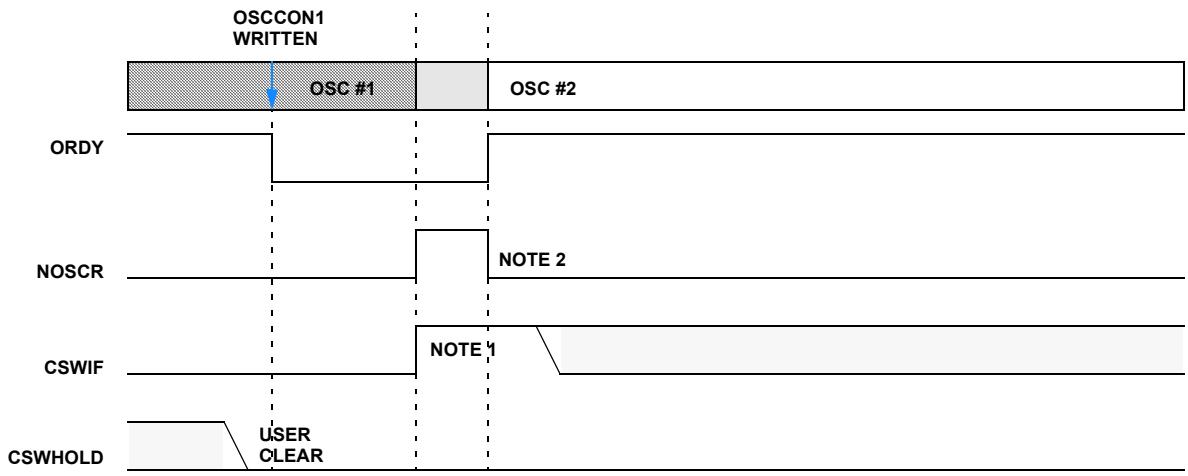
7.3.2 CLOCK SWITCH AND SLEEP

If OSCCON1 is written with a new value and the device is put to Sleep before the switch completes, the switch will not take place and the device will enter Sleep mode.

When the device wakes from Sleep and the CSWHOLD bit is clear, the device will wake with the 'new' clock active, and the Clock Switch Interrupt flag bit (CSWIF) will be set.

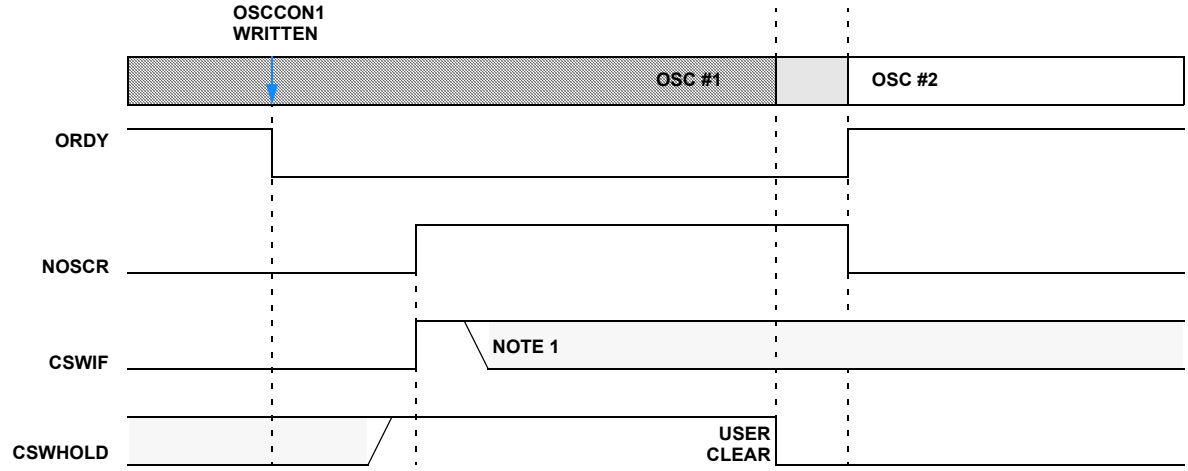
When the device wakes from Sleep and the CSWHOLD bit is set, the device will wake with the 'old' clock active and the new clock will be requested again.

FIGURE 7-6: CLOCK SWITCH (CSWHOLD = 0)



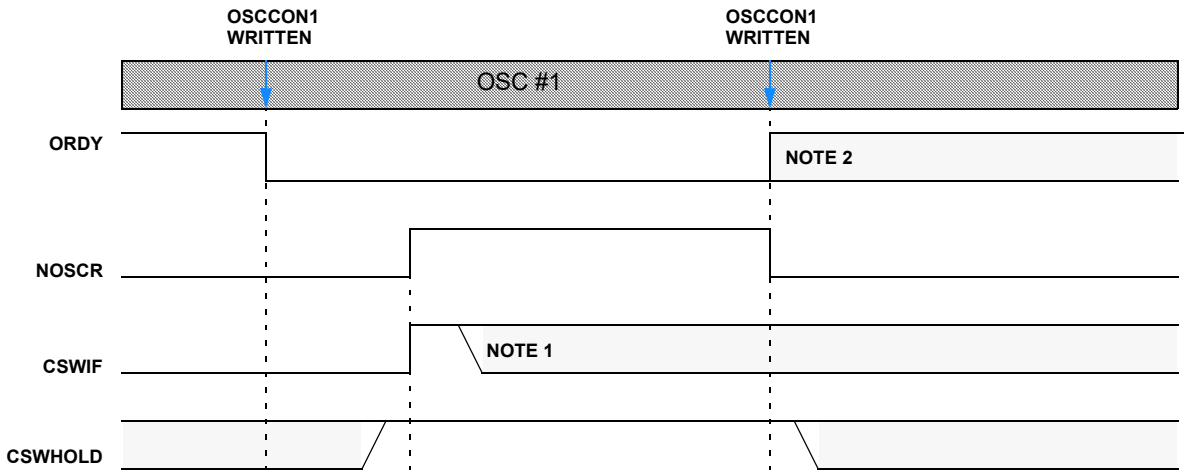
Note 1: CSWIF is asserted coincident with NOSCR; interrupt is serviced at OSC#2 speed.
2: The assertion of NOSCR is hidden from the user because it appears only for the duration of the switch.

FIGURE 7-7: CLOCK SWITCH (CSWHOLD = 1)



Note 1: CSWIF is asserted coincident with NOSCR, and may be cleared before or after clearing CSWHOLD = 0.

FIGURE 7-8: CLOCK SWITCH ABANDONED

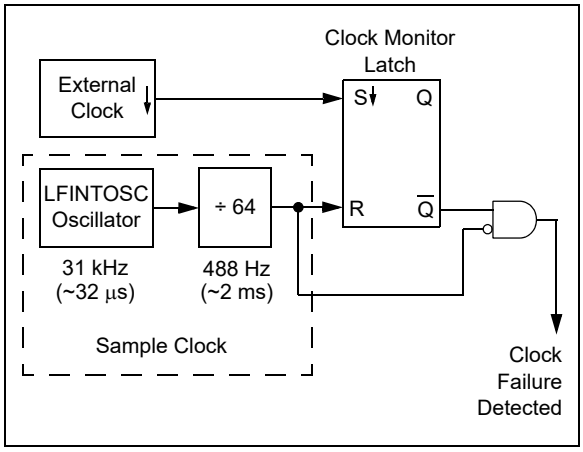


Note 1: CSWIF may be cleared before or after rewriting OSCCON1; CSWIF is not automatically cleared.
2: ORDY = 0 if OSCCON1 does not match OSCCON2; a new switch will begin.

7.4 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM is enabled by setting the FCMEN bit in the Configuration Words. The FSCM is applicable to all external Oscillator modes (LP, XT, HS, ECL/M/H and Secondary Oscillator).

FIGURE 7-9: FSCM BLOCK DIAGRAM



7.4.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 7-9. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the external clock goes low.

7.4.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM overwrites the COSC bits to select HFINTOSC (`3'b110`). The frequency of HFINTOSC would be determined by the previous state of the FRQ bits and the NDIV/CDIV bits. The bit flag OSFIF of the respective PIR register is set. Setting this flag will generate an interrupt if the OSFIE bit of the respective PIR register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation, by writing to the NOSC and NDIV bits of the OSCCON1 register.

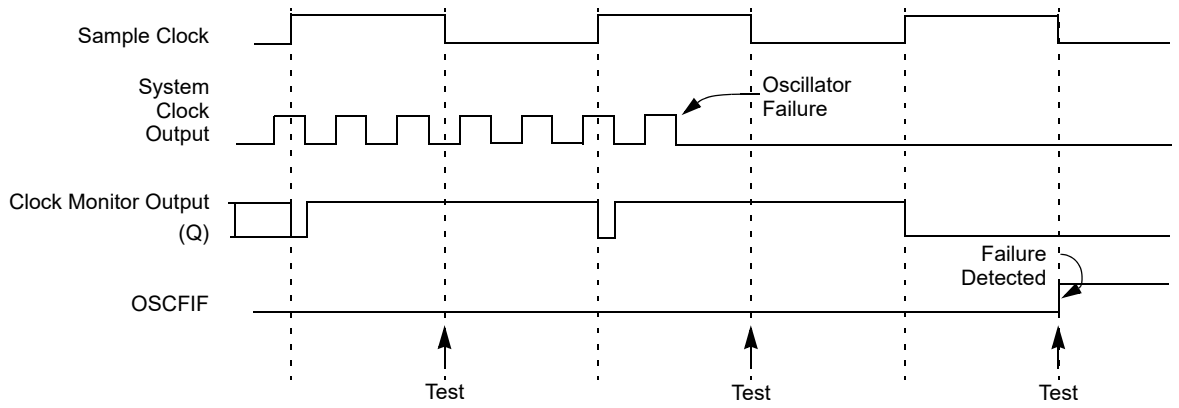
7.4.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared after a Reset, executing a `SLEEP` instruction or changing the NOSC and NDIV bits of the OSCCON1 register. When switching to the external oscillator or PLL, the OST is restarted. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON1. When the OST times out, the Fail-Safe condition is cleared after successfully switching to the external clock source. The OSCFIF bit should be cleared prior to switching to the external clock source. If the Fail-Safe condition still exists, the OSCFIF flag will again become set by hardware.

7.4.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed.

FIGURE 7-10: FSCM TIMING DIAGRAM



Note: The system clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.

TABLE 7-1: NOSC/COSC AND NDIV/CDIV BIT SETTINGS

NOSC[2:0] COSC[2:0]	Clock Source
111	EXTOSC(1)
110	HFINTOSC(2)
101	LFINTOSC
100	SOSC
011	Reserved
010	EXTOSC + 4x PLL(3)
001	Reserved
000	Reserved

NDIV[3:0] CDIV[3:0]	Clock Divider
1111-1010	Reserved
1001	512
1000	256
0111	128
0110	64
0101	32
0100	16
0011	8
0010	4
0001	2
0000	1

Note 1: EXTOSC configured by the FEXTOSC bits of Configuration Word 1 (Register 5-1).
2: HFINTOSC frequency is set with the FRQ bits of the OSCFRQ register (Register 7-5).
3: EXTOSC must meet the PLL specifications (Table 44-10).

7.5 Register Definitions: Oscillator Control

REGISTER 7-1: OSCCON1: OSCILLATOR CONTROL REGISTER 1

U-0	R/W-f/f	R/W-f/f	R/W-f/f	R/W-q/q	R/W-q/q	R/W-q/q	R/W-q/q
—	NOSC[2:0]			NDIV[3:0]			
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	f = determined by Configuration bit setting
		q = Reset value is determined by hardware

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **NOSC[2:0]:** New Oscillator Source Request bits(1,2,3)
The setting requests a source oscillator and PLL combination per Table 7-1.
POR value = RSTOSC (Register 5-1).

bit 3-0 **NDIV[3:0]:** New Divider Selection Request bits(2,3)
The setting determines the new postscaler division ratio per Table 7-1.

Note 1: The default value (f/f) is determined by the RSTOSC Configuration bits. See Table 7-2 below.
2: If NOSC is written with a reserved value (Table 7-1), the operation is ignored and neither NOSC nor NDIV is written.
3: When CSWEN = 0, this register is read-only and cannot be changed from the POR value.

TABLE 7-2: DEFAULT OSCILLATOR SETTINGS

RSTOSC	SFR Reset Values			Initial FOSC Frequency
	NOSC/COSC	CDIV	OSCFRQ	
111	111	1:1	4 MHz	EXTOSC per FEXTOSC
110	110	4:1		FOSC = 1 MHz (4 MHz/4)
101	101	1:1		LFINTOSC
100	100	1:1		SOSC
011	Reserved			
010	010	1:1	4 MHz	EXTOSC + 4xPLL(1)
001	Reserved			
000	110	1:1	64 MHz	FOSC = 64 MHz

Note 1: EXTOSC must meet the PLL specifications (Table 44-10).

REGISTER 7-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

U-0	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f
—	COSC[2:0]			CDIV[3:0]			
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 **Unimplemented:** Read as ‘0’

bit 6-4 **COSC[2:0]:** Current Oscillator Source Select bits (read-only)(1)
Indicates the current source oscillator and PLL combination per Table 7-1.

bit 3-0 **CDIV[3:0]:** Current Divider Select bits (read-only)(1)
Indicates the current postscaler division ratio per Table 7-1.

Note 1: The POR value is the value present when user code execution begins.

REGISTER 7-3: OSCCON3: OSCILLATOR CONTROL REGISTER 3

R/W/HC-0/0	R/W-0/0	U-0	R-0/0	R-0/0	U-0	U-0	U-0
CSWHOLD	SOSCPWR	—	ORDY	NOSCR	—	—	—
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	HC = Bit is cleared by hardware

bit 7 **CSWHOLD:** Clock Switch Hold bit
1 = Clock switch will hold (with interrupt) when the oscillator selected by NOSC is ready
0 = Clock switch may proceed when the oscillator selected by NOSC is ready; NOSCR becomes ‘1’, the switch will occur

bit 6 **SOSCPWR:** Secondary Oscillator Power Mode Select bit
1 = Secondary oscillator operating in High-Power mode
0 = Secondary oscillator operating in Low-Power mode

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **ORDY:** Oscillator Ready bit (read-only)
1 = OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC
0 = A clock switch is in progress

bit 3 **NOSCR:** New Oscillator is Ready bit (read-only)(1)
1 = A clock switch is in progress and the oscillator selected by NOSC indicates a “ready” condition
0 = A clock switch is not in progress, or the NOSC-selected oscillator is not yet ready

bit 2-0 **Unimplemented:** Read as ‘0’

Note 1: If CSWHOLD = 0, the user may not see this bit set because, when the oscillator becomes ready there may be a delay of one instruction clock before this bit is set. The clock switch occurs in the next instruction cycle and this bit is cleared.

REGISTER 7-4: OSCSTAT: OSCILLATOR STATUS REGISTER 1

R-q/q	R-q/q	R-q/q	R-q/q	R-q/q	R-q/q	U-0	R-q/q
EXTOR	HFOR	MFOR	LFOR	SOR	ADOR	—	PLLR
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 = Reset value is determined by hardware

bit 7 **EXTOR:** EXTOSC (external) Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 6 **HFOR:** HFINTOSC Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 5 **MFOR:** MFINTOSC Oscillator Ready
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 4 **LFOR:** LFINTOSC Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 3 **SOR:** Secondary (Timer1) Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 2 **ADOR:** ADC Oscillator Ready bit
1 = The oscillator is ready to be used
0 = The oscillator is not enabled, or is not yet ready to be used

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **PLLR:** PLL is Ready bit
1 = The PLL is ready to be used
0 = The PLL is not enabled, the required input source is not ready, or the PLL is not locked.

REGISTER 7-5: OSCFRQ: HFINTOSC FREQUENCY SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-q/q	R/W-q/q	R/W-q/q	R/W-q/q
—	—	—	—	FRQ[3:0]			
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 = Reset value is determined by hardware

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **FRQ[3:0]:** HFINTOSC Frequency Selection bits(1)

FRQ[3:0]	Nominal Freq (MHz)
1001	Reserved
1010	
1111	
1110	
1101	
1100	
1011	
1000	64
0111	48
0110	32
0101	16
0100	12
0011	8
0010	4
0001	2
0000	1

Note 1: Refer to Table 7-2 for more information.

REGISTER 7-6: OSCTUNE: HFINTOSC TUNING REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	TUN[5:0]					
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 **Unimplemented:** Read as ‘0’

bit 5-0 **TUN[5:0]:** HFINTOSC Frequency Tuning bits

`01 1111` = Maximum frequency

•

•

•

`00 0000` = Center frequency. Oscillator module is running at the calibrated frequency (default value).

•

•

•

`10 0000` = Minimum frequency

REGISTER 7-7: OSCEN: OSCILLATOR MANUAL ENABLE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0
EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN	—	—

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 **EXTOEN:** External Oscillator Manual Request Enable bit
1 = EXTOSC is explicitly enabled, operating as specified by FEXTOSC
0 = EXTOSC could be enabled by requesting peripheral

bit 6 **HFOEN:** HFINTOSC Oscillator Manual Request Enable bit
1 = HFINTOSC is explicitly enabled, operating as specified by OSCFRQ (Register 7-5)
0 = HFINTOSC could be enabled by requesting peripheral

bit 5 **MFOEN:** MFINTOSC (500 kHz/31.25 kHz) Oscillator Manual Request Enable bit (Derived from HFINTOSC)
1 = MFINTOSC is explicitly enabled
0 = MFINTOSC could be enabled by requesting peripheral

bit 4 **LFOEN:** LFINTOSC (31 kHz) Oscillator Manual Request Enable bit
1 = LFINTOSC is explicitly enabled
0 = LFINTOSC could be enabled by requesting peripheral

bit 3 **SOSCEN:** Secondary Oscillator Manual Request Enable bit
1 = Secondary Oscillator is explicitly enabled, operating as specified by SOSCPWR
0 = Secondary Oscillator could be enabled by requesting peripheral

bit 2 **ADOEN:** ADC Oscillator Manual Request Enable bit
1 = ADC oscillator is explicitly enabled
0 = ADC oscillator could be enabled by requesting peripheral

bit 1-0 **Unimplemented:** Read as ‘0’

TABLE 7-3: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
OSCCON1	—	NOSC[2:0]			NDIV[3:0]				104
OSCCON2	—	COSC[2:0]			CDIV[3:0]				105
OSCCON3	CSWHOLD	SOSCPWR	—	ORDY	NOSCR	—	—	—	105
OSCSTAT	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR	—	PLLR	106
OSCTUNE	—	—	TUN[5:0]						108
OSCFRQ	—	—	—	—	FRQ[3:0]				107
OSCEN	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN	—	—	109

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by clock sources.

8.0 REFERENCE CLOCK OUTPUT MODULE

The reference clock output module provides the ability to send a clock signal to the clock reference output pin (CLKR). The reference clock output can also be used as a signal for other peripherals, such as the Data Signal Modulator (DSM), Memory Scanner and Timer module.

The reference clock output module has the following features:

- Selectable clock source using the CLKRCLK register
- Programmable clock divider
- Selectable duty cycle

FIGURE 8-1: CLOCK REFERENCE BLOCK DIAGRAM

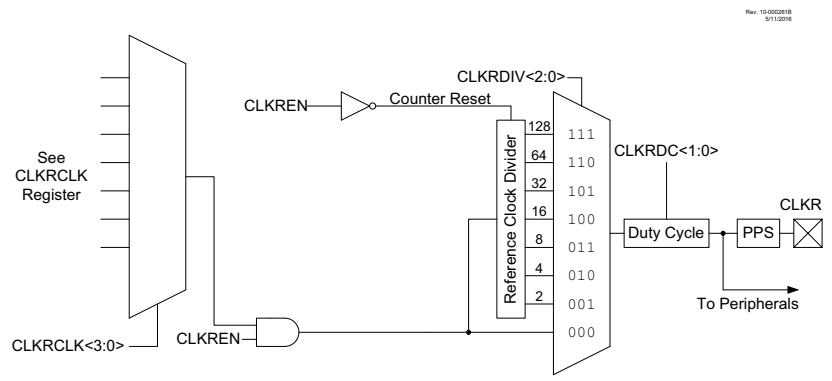
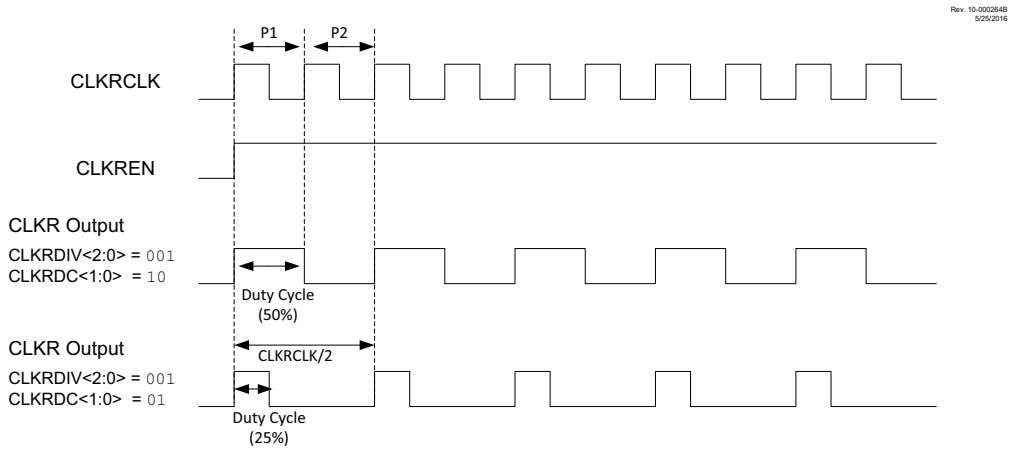


FIGURE 8-2: CLOCK REFERENCE TIMING



8.1 Clock Source

The input to the reference clock output can be selected using the CLKRCLK register.

8.1.1 CLOCK SYNCHRONIZATION

Once the reference clock enable (EN) is set, the module is ensured to be glitch-free at start-up.

When the reference clock output is disabled, the output signal will be disabled immediately.

Clock dividers and clock duty cycles can be changed while the module is enabled, but glitches may occur on the output. To avoid possible glitches, clock dividers and clock duty cycles should be changed only when the CLKREN is clear.

8.2 Programmable Clock Divider

The module takes the clock input and divides it based on the value of the DIV[2:0] bits of the CLKRCON register (Register 8-1).

The following configurations can be made based on the DIV[2:0] bits:

- Base FOSC value
- FOSC divided by 2
- FOSC divided by 4
- FOSC divided by 8
- FOSC divided by 16
- FOSC divided by 32
- FOSC divided by 64
- FOSC divided by 128

The clock divider values can be changed while the module is enabled; however, in order to prevent glitches on the output, the DIV[2:0] bits should only be changed when the module is disabled (EN = 0).

8.3 Selectable Duty Cycle

The DC[1:0] bits of the CLKRCON register can be used to modify the duty cycle of the output clock. A duty cycle of 25%, 50%, or 75% can be selected for all clock rates, with the exception of the undivided base FOSC value.

The duty cycle can be changed while the module is enabled; however, in order to prevent glitches on the output, the DC[1:0] bits should only be changed when the module is disabled (EN = 0).

Note:	The DC1 bit is reset to '1'. This makes the default duty cycle 50% and not 0%.

8.4 Operation in Sleep Mode

The reference clock output module clock is based on the system clock. When the device goes to Sleep, the module outputs will remain in their current state. This will have a direct effect on peripherals using the reference clock output as an input signal. No change should occur in the module from entering or exiting from Sleep.

8.5 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
CLKR	CLKR

REGISTER 8-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER

R/W-0/0	U-0	U-0	R/W-1/1	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	—	DC[1:0]		DIV[2:0]		
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 **EN:** Reference Clock Module Enable bit
1 = Reference clock module enabled
0 = Reference clock module is disabled

bit 6-5 **Unimplemented:** Read as '0'

bit 4-3 **DC[1:0]:** Reference Clock Duty Cycle bits(1)
11 = Clock outputs duty cycle of 75%
10 = Clock outputs duty cycle of 50%
01 = Clock outputs duty cycle of 25%
00 = Clock outputs duty cycle of 0%

bit 2-0 **DIV[2:0]:** Reference Clock Divider bits
111 = Base clock value divided by 128
110 = Base clock value divided by 64
101 = Base clock value divided by 32
100 = Base clock value divided by 16
011 = Base clock value divided by 8
010 = Base clock value divided by 4
001 = Base clock value divided by 2
000 = Base clock value

Note 1: Bits are valid for reference clock divider values of two or larger, the base clock cannot be further divided.

REGISTER 8-2: CLKRCLK: CLOCK REFERENCE CLOCK SELECTION MUX

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CLK[3:0]			
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 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **CLK[3:0]:** CLKR Clock Selection bits

1111 = Reserved
•
•
•
1011 = Reserved
1010 = CLC4 Output
1001 = CLC3 Output
1000 = CLC2 Output
0111 = CLC1 Output
0110 = NCO1 Output
0101 = SOSC
0100 = MFINTOSC (31.25 kHz)
0011 = MFINTOSC (500 kHz)
0010 = LFINTOSC (31 kHz)
0001 = HFINTOSC
0000 = FOSC

TABLE 8-1: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK REFERENCE OUTPUT

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CLKRCON	EN	—	—	DC[1:0]		DIV[2:0]			113
CLKRCLK	—	—	—	—	—	CLK[2:0]			114

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the CLKR module.

9.0 INTERRUPT CONTROLLER

The Vectored Interrupt Controller module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the CPU. This module includes the following major features:

- Interrupt Vector Table (IVT) with a unique vector for each interrupt source
- Fixed and ensured interrupt latency
- Programmable base address for Interrupt Vector Table (IVT) with lock
- Two user-selectable priority levels – High priority and Low priority
- Two levels of context saving
- Interrupt state status bits to indicate the current execution status of the CPU

The Interrupt Controller module assembles all of the interrupt request signals and resolves the interrupts based on both a fixed natural order priority (i.e., determined by the Interrupt Vector Table), and a user-assigned priority (i.e., determined by the IPRx registers), thereby eliminating scanning of interrupt sources.

9.1 Interrupt Control and Status Registers

The devices in this family implement the following registers for the interrupt controller:

- INTCON0, INTCON1 Control Registers
- PIRx – Peripheral Interrupt Status Registers
- PIEx – Peripheral Interrupt Enable Registers
- IPRx – Peripheral Interrupt Priority Registers
- IVTBASE[20:0] Address Registers
- IVTLOCK Register

Global interrupt control functions and external interrupts are controlled from the INTCON0 register. The INTCON1 register contains the status flags for the Interrupt controller.

The PIRx registers contain all of the interrupt request flags. Each source of interrupt has a status bit, which is set by the respective peripherals or an external signal and is cleared via software.

The PIEx registers contain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.

The IPRx registers are used to set the Interrupt Priority Level for each source of interrupt. Each user interrupt source can be assigned to either a high or low priority.

The IVTBASE register is user programmable and is used to determine the start address of the Interrupt Vector Table and the IVTLOCK register is used to prevent any unintended writes to the IVTBASE register.

There are two other configuration bits that control the way the interrupt controller can be configured.

- CONFIG2L[3], MVECEN bit
- CONFIG2L[4], IVT1WAY bit

The MVECEN bit in CONFIG2L determines whether the Vector table is used to determine the interrupt priorities.

- The IVT1WAY determines the number of times the IVTLOCKED bit can be cleared and set after a device Reset. See **Section 9.2.3 “Interrupt Vector Table (IVT) address calculation”** for details.

9.2 Interrupt Vector Table (IVT)

The interrupt controller supports an Interrupt Vector Table (IVT) that contains the vector address location for each interrupt request source.

The Interrupt Vector Table (IVT) resides in program memory, starting at address location determined by the IVTBASE registers; refer to Register 9-36, Register 9-37 and Register 9-38 for details. The IVT contains 68 vectors, one for each source of interrupt. Each interrupt vector location contains the starting address of the associated Interrupt Service Routine (ISR).

The MVECEN bit in Configuration Word 2L controls the availability of the vector table.

9.2.1 INTERRUPT VECTOR TABLE BASE ADDRESS (IVTBASE)

The start address of the vector table is user programmable through the IVTBASE registers. The user must ensure the start address is such that it can encompass the entire vector table inside the program memory.

Each vector address is a 16-bit word (or two address locations on PIC18 devices). So for n interrupt sources, there are 2n address locations necessary to hold the table starting from IVTBASE as the first location. So the staring address of IVTBASE should be chosen such that the address range form IVTBASE to (IVTBASE +2n-1) can be encompassed inside the program flash memory.

For example, the K42 devices have the highest vector number: 81. So IVTBASE should be chosen such that (IVTBASE + 0xA1) is less than the last memory location in program flash memory.

A programmable vector table base address is useful in situations to switch between different sets of vector tables, depending on the application. It can also be used when the application program needs to update the existing vector table (vector address values).

Note:	It is required that the user assign an even address to the IVTBASE register for correct operation.

9.2.2 INTERRUPT VECTOR TABLE CONTENTS

MVECEN = 0

When MVECEN = 0, the address location pointed by the IVTBASE registers has a GOTO instruction for a high priority interrupt. Similarly, the corresponding low priority vector location also has a GOTO instruction, which is executed in case of a low priority interrupt.

MVECEN = 1

When MVECEN = 1, the value in the vector table of each interrupt, points to the address location of the first instruction of the interrupt service routine.

ISR Location = Interrupt Vector Table entry << 2.

9.2.3 INTERRUPT VECTOR TABLE (IVT) ADDRESS CALCULATION

MVECEN = 0

When the MVECEN bit in Configuration Word 2L (Register 5-3) is cleared, the address pointed by IVTBASE registers is used as the high priority interrupt vector address. The low priority interrupt vector address is offset eight instruction words from the address in IVTBASE registers.

For PIC18 devices the IVTBASE registers default to 00 0008h, the high priority interrupt vector address will be 00 0008h and the low priority interrupt vector address will be 00 0018h.

MVECEN = 1

Each interrupt has a unique vector number associated with it as defined in Table 9-2. This vector number is used for calculating the location of the interrupt vector for a particular interrupt source.

Interrupt Vector Address = IVTBASE + (2*Vector Number).

This calculated Interrupt Vector Address value is stored in the IVTAD[20:0] registers when an interrupt is received (Registers 9-39 through 9-41).

User-assigned software priority assigned using the IPRx registers does not affect address calculation and is only used to resolve concurrent interrupts.

If for any reason the address of the ISR could not be fetched from the vector table, it will cause the system to reset and clear the memory execution violation flag (MEMV bit) in PCON1 register (Register 6-3). This occurs due to any one of the following:

- The entry for the interrupt in the vector table lies outside the executable PFM area (SAF area is non-executable when SAFEN = 1).
- ISR pointed by the vector table lies outside the executable PFM area (SAF area is non-executable when SAFEN = 1).

TABLE 9-1: IVT ADDRESS CALCULATION SUMMARY

IVT Address Calculation		Interrupt Priority INTCON0 Register, IPEN bit	
		0	1
Multi-Vector Enable CONFIG 2L register MVECEN bit	0	IVTBASE	High Priority IVTBASE
			Low Priority IVTBASE + 8 words
	1	IVTBASE + 2*(Vector Number)	

9.2.4 ACCESS CONTROL FOR IVTBASE REGISTERS

The Interrupt controller has an IVTLOCKED bit which can be set to avoid inadvertent changes to the IVTBASE registers contents. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes.

To allow writes to IVTBASE registers, the interrupts must be disabled (GIEH = 0) and the IVTLOCKED bit must be cleared. The user must follow the sequence shown in Example 9-1 to clear the IVTLOCKED bit.

EXAMPLE 9-1: IVT UNLOCK SEQUENCE

```
; Disable Interrupts:
        BCF             INTCON0, GIE;
; Bank to IVTLOCK register
        BANKSEL         IVTLOCK;
        MOVLW           55h;

; Required sequence, next 4 instructions
        MOVWF           IVTLOCK;
        MOVLW           AAh;
        MOVWF           IVTLOCK;

; Clear IVTLOCKED bit to enable writes
        BCF             IVTLOCK, IVTLOCKED;

; Enable Interrupts
        BSF             INTCON0, GIE;
```

The user must follow the sequence shown in Example 9-2 to set the IVTLOCKED bit.

EXAMPLE 9-2: IVT LOCK SEQUENCE

```
; Disable Interrupts:
        BCF             INTCON0, GIE;
; Bank to IVTLOCK register
        BANKSEL         IVTLOCK;
        MOVLW           55h;

; Required sequence, next 4 instructions
        MOVWF           IVTLOCK;
        MOVLW           AAh;
        MOVWF           IVTLOCK;

; Set IVTLOCKED bit to enable writes
        BSF             IVTLOCK, IVTLOCKED;

; Enable Interrupts
        BSF             INTCON0, GIE;
```

When the IVT1WAY Configuration bit is set, the IVTLOCKED bit can be cleared and set only once after a device Reset. The unlock operation in Example 9-1 will have no effect after the lock sequence in Example 9-2 is used to set the IVTLOCK. Unlocking is inhibited until a system Reset occurs.

© 2017-2020 Microchip Technology Inc.

9.3 Interrupt Priority

The final priority level for any pending source of interrupt is determined first by the user-assigned priority of that source in the IPRx register, then by the natural order priority within the IVT. The sections below detail the operation of Interrupt priorities.

9.3.1 USER (SOFTWARE) PRIORITY

User-assigned interrupt priority is enabled by setting the IPEN bit in the INTCON0 register (Register 9-1). Each peripheral interrupt source can be assigned a high or low priority level by the user. The user-assignable interrupt priority control bits for each interrupt are located in the IPRx registers (Registers 9-25 through 9-35).

The interrupts are serviced based on predefined interrupt priority scheme defined below.

1. Interrupts set by the user as high-priority interrupt have higher precedence of execution. High-priority interrupts will override a low-priority request when:
   a) A low priority interrupt has been requested or its request is already pending.
   b) A low- and high-priority interrupt are triggered concurrently, i.e., on the same instruction cycle(1).
   c) A low-priority interrupt was requested and the corresponding Interrupt Service Routine is currently executing. In this case, the lower priority interrupt routine will complete executing after the high-priority interrupt has been serviced(2).
2. Interrupts set by the user as a low priority have the lower priority of execution and are preempted by any high-priority interrupt.
3. Interrupts defined with the same software priority cannot preempt or interrupt each other. Concurrent pending interrupts with the same user priority are resolved using the natural order priority. (when MVECEN = ON) or in the order the interrupt flag bits are polled in the ISR (when MVECEN = OFF).

Note 1: When a high priority interrupt preempts a concurrent low priority interrupt, the GIEL bit may be cleared in the high priority Interrupt Service Routine. If the GIEL bit is cleared, the low priority interrupt will NOT be serviced even if it was originally requested. The corresponding interrupt flag needs to be cleared in user code.

2: When a high priority interrupt is requested while a low priority Interrupt Service Routine is executing, the GIEL bit may be cleared in the high priority Interrupt Service Routine. The pending low priority interrupt will resume even if the GIEL bit is cleared.

9.3.2 NATURAL ORDER (HARDWARE) PRIORITY

When more than one interrupt with the same user specified priority level are requested, the priority conflict is resolved by using a method called “Natural Order Priority”. Natural order priority is a fixed priority scheme that is based on the Interrupt Vector Table. Table 9-2 shows the natural order priority and the interrupt vector number assigned for each source.

TABLE 9-2: INTERRUPT VECTOR PRIORITY TABLE

Vector Number	Interrupt Source	Vector Number	Interrupt Source
0	Software Interrupt	42	DMA2SCNT
1	HLVD	43	DMA2DCNT
2	OSF	44	DMA2OR
3	CSW	45	DMA2A
4	NVM	46	I2C2RX
5	SCAN	47	I2C2TX
6	CRC	48	I2C2
7	IOC	49	I2C2E
8	INT0	50	U2RX
9	ZCD	51	U2TX
10	AD	52	U2E
11	ADT	53	U2
12	C1	54	TMR3
13	SMT1	55	TMR3G
14	SMT1PRA	56	TMR4
15	SMT1PWA	57	CCP2
16	DMA1SCNT	58	—
17	DMA1DCNT	59	CWG2
18	DMA1OR	60	CLC2
19	DMA1A	61	INT2
20	SPI1RX	62	—
21	SPI1TX	63	—
22	SPI1	64	—
23	I2C1RX	65	—
24	I2C1TX	66	—
25	I2C1	67	—
26	I2C1E	68	—
27	U1RX	69	—
28	U1TX	70	TMR5
29	U1E	71	TMR5G
30	U1	72	TMR6
31	TMR0	73	CCP3
32	TMR1	74	CWG3
33	TMR1G	75	CLC3
34	TMR2	76	—
35	CCP1	77	—
36	—	78	—
37	NCO	79	—
38	CWG1	80	CCP4
39	CLC1	81	CLC4
40	INT1		
41	C2		

The natural order priority scheme has vector interrupt 0 as the highest priority and vector interrupt 81 as the lowest priority.

For example, when two concurrently occurring interrupt sources that are both designated high priority using the IPRx register will be resolved using the natural order priority (i.e., the interrupt with a lower corresponding vector number will preempt the interrupt with the higher vector number).

The ability for the user to assign every interrupt source to high or low priority levels means that the user program can give an interrupt with a low natural order priority a higher overall priority level.

9.4 Interrupt Operation

All pending interrupts are indicated by the flag bit being equal to a ‘1’ in the PIRx register. All pending interrupts are resolved using the priority scheme explained in **Section 9.3 “Interrupt Priority”**.

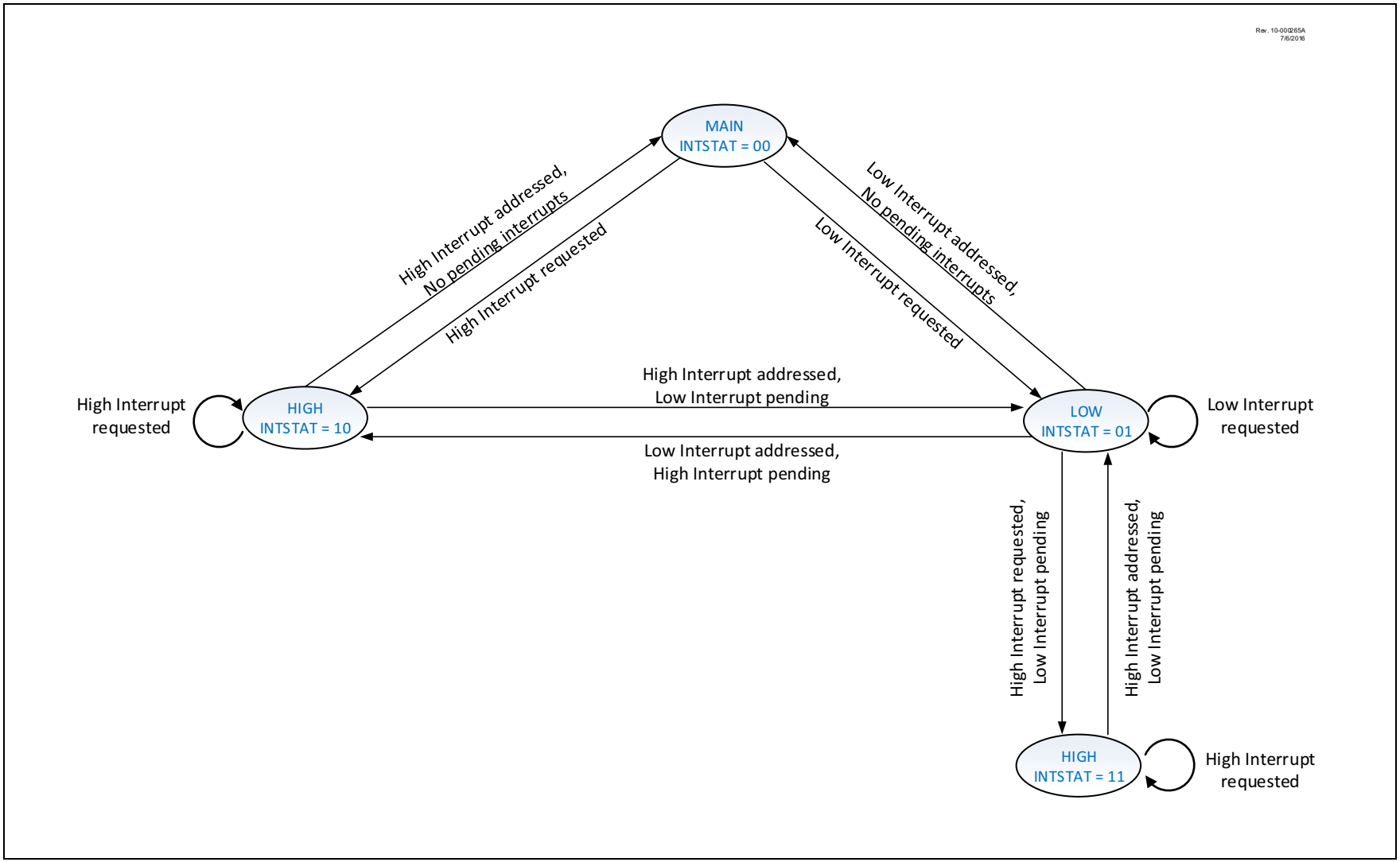
Once the interrupt source to be serviced is resolved, the program execution vectors to the resolved interrupt vector addresses, as explained in **Section 9.2 “Interrupt Vector Table (IVT)”**. The vector number is also stored in the WREG register. Most of the flag bits are required to be cleared by the application software, but in some cases, device hardware clears the interrupt automatically. Some flag bits are read-only in the PIRx registers, these flags are a summary of the source interrupts and the corresponding interrupt flags of the source must be cleared.

A valid interrupt can be either a high or low priority interrupt when in main routine or a high priority interrupt when in low priority Interrupt Service Routine. Depending on order of interrupt requests received and their relative timing, the CPU will be in the state of execution indicated by the STAT bits of the INTCON1 register (Register 9-2).

The State machine shown in Figure 9-1 and the subsequent sections detail the execution of interrupts when received in different orders.

Note: The state of GIEH/L is not changed by the hardware when servicing an interrupt. The internal state machine is used to keep track of execution states. These bits can be manipulated in the user code resulting in transferring execution to the main routine and ignoring existing interrupts.

FIGURE 9-1: VECTORED INTERRUPTS STATE TRANSITION DIAGRAM

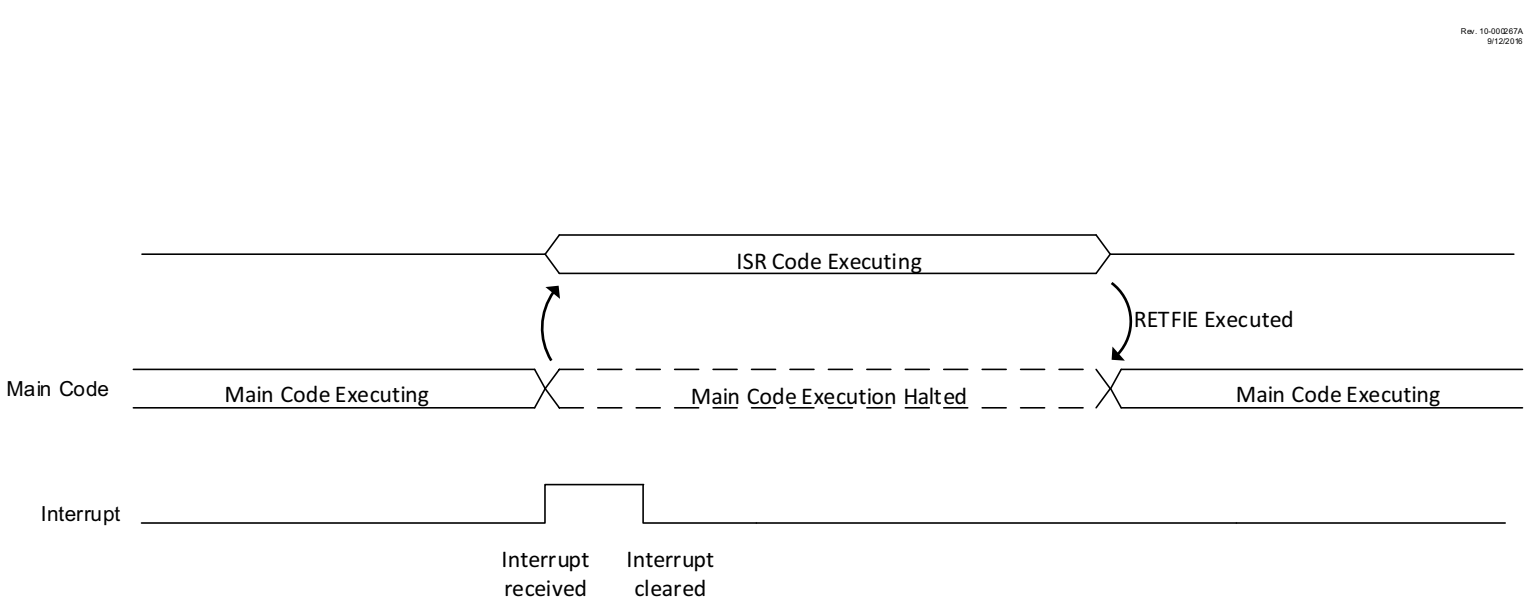


© 2017-2020 Microchip Technology Inc.

9.4.1 SERVING A HIGH OR LOW PRIORITY INTERRUPT WHEN MAIN ROUTINE CODE IS EXECUTING

When a high or low priority interrupt is requested when the main routine code is executing, the main routine execution is halted and the ISR is addressed, see Figure 9-2. Upon a return from the ISR (by executing the RETFIE instruction), the main routine resumes execution.

FIGURE 9-2: INTERRUPT EXECUTION: HIGH/LOW PRIORITY INTERRUPT WHEN EXECUTING MAIN ROUTINE



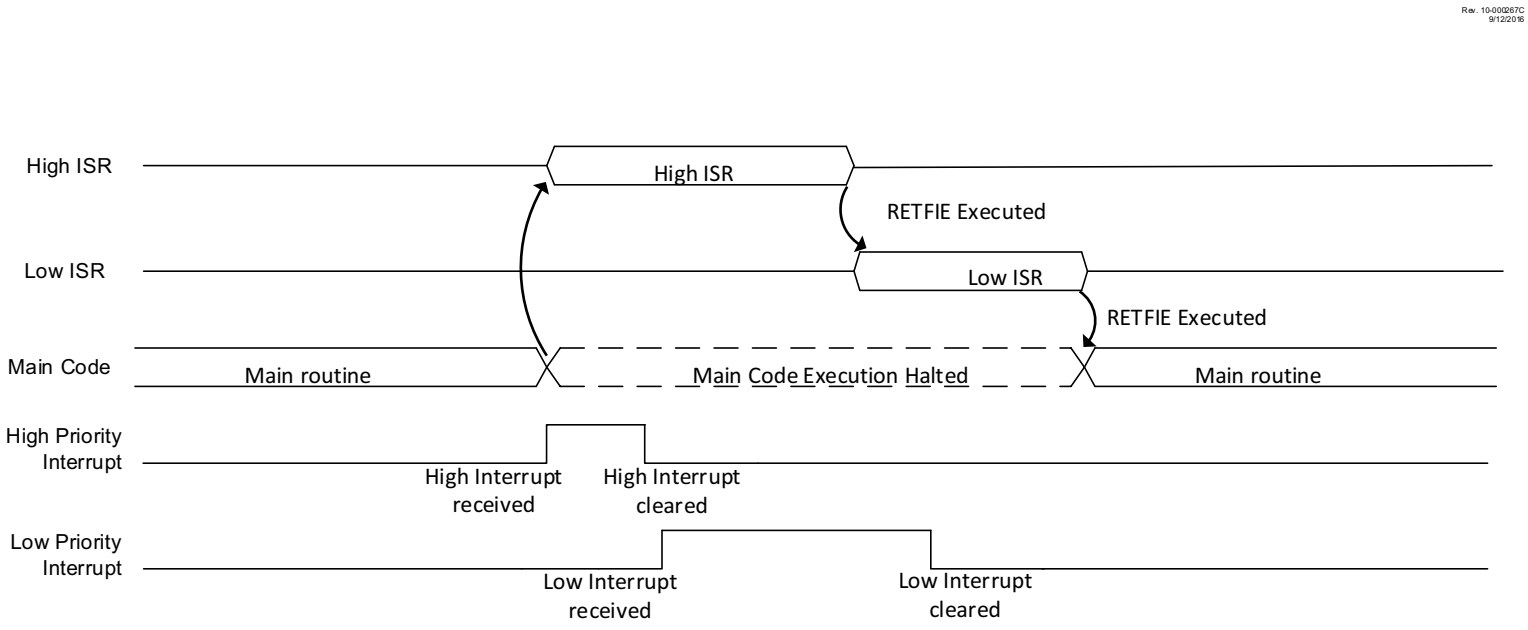
© 2017-2020 Microchip Technology Inc.

9.4.2 SERVING A HIGH PRIORITY INTERRUPT WHILE A LOW PRIORITY INTERRUPT PENDING

A high priority interrupt request will always take precedence over any interrupt of a lower priority. The high priority interrupt is acknowledged first, then the low-priority interrupt is acknowledged. Upon a return from the high priority ISR (by executing the `RETFIE` instruction), the low priority interrupt is serviced, see Figure 9-3.

If any other high priority interrupts are pending and enabled, then they are serviced before servicing the pending low priority interrupt. If no other high priority interrupt requests are active, the low priority interrupt is serviced.

FIGURE 9-3: INTERRUPT EXECUTION: HIGH PRIORITY INTERRUPT WITH A LOW PRIORITY INTERRUPT PENDING



© 2017-2020 Microchip Technology Inc.

9.4.3 PREEMPTING LOW PRIORITY INTERRUPTS

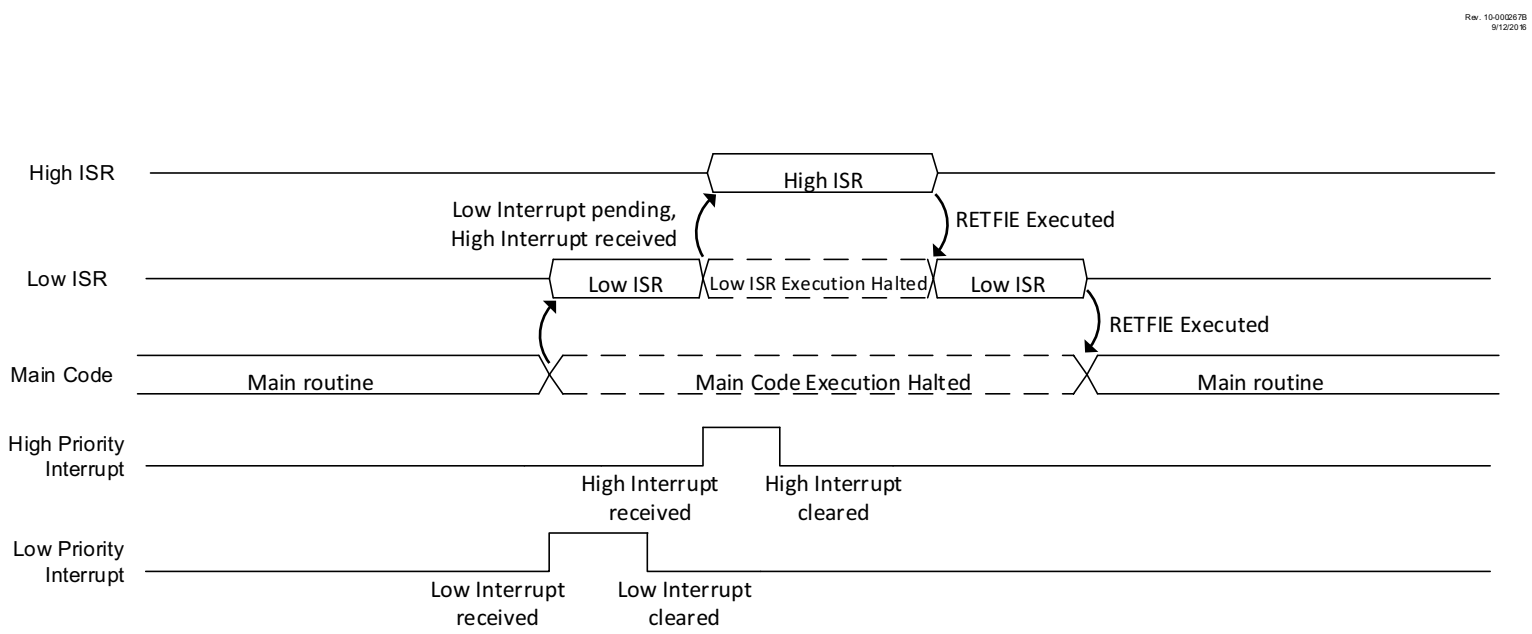
Low-priority interrupts can be preempted by high priority interrupts. While in the low priority ISR, if a high-priority interrupt arrives, the high priority interrupt request is generated and the low priority ISR is suspended, while the high priority ISR is executed, see Figure 9-4.

After the high priority ISR is complete and if any other high priority interrupt requests are not active, the execution returns to the preempted low priority ISR.

Note 1: The high priority interrupt flag must be cleared to avoid recursive interrupts.

2: If a low-priority ISR was already serviced halfway before moving on to a high priority ISR, then the low priority ISR is completely serviced even if user code clears GIEL.

FIGURE 9-4: INTERRUPT EXECUTION: HIGH PRIORITY INTERRUPT PREEMPTING LOW PRIORITY INTERRUPTS

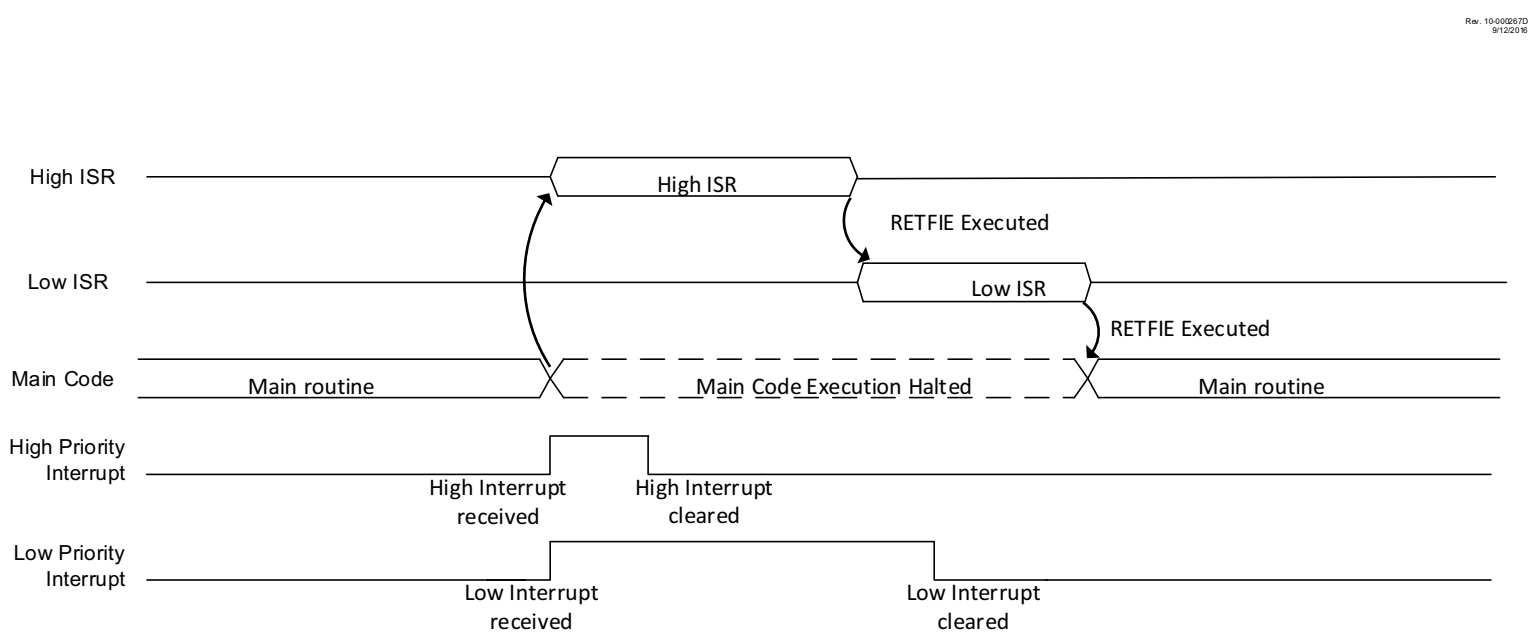


© 2017-2020 Microchip Technology Inc.

9.4.4 SIMULTANEOUS LOW AND HIGH PRIORITY INTERRUPTS

When both high and low interrupts are active in the same instruction cycle (i.e., simultaneous interrupt events), both the high and the low priority requests are generated. The high priority ISR is serviced first before servicing the low priority interrupt see Figure 9-5.

FIGURE 9-5: INTERRUPT EXECUTION: SIMULTANEOUS LOW AND HIGH PRIORITY INTERRUPTS



© 2017-2020 Microchip Technology Inc.

9.5 Context Saving

The Interrupt controller supports a two-level deep context saving (Main routine context and Low ISR context). Refer to state machine shown in Figure 9-6 for details.

The Program Counter (PC) is saved on the dedicated device PC stack. CPU registers saved include STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U.

After WREG has been saved to the context registers, the resolved vector number of the interrupt source to be serviced is copied into WREG. Context save and restore operation is completed by the interrupt controller based on current state of the interrupts and the order in which they were sent to the CPU.

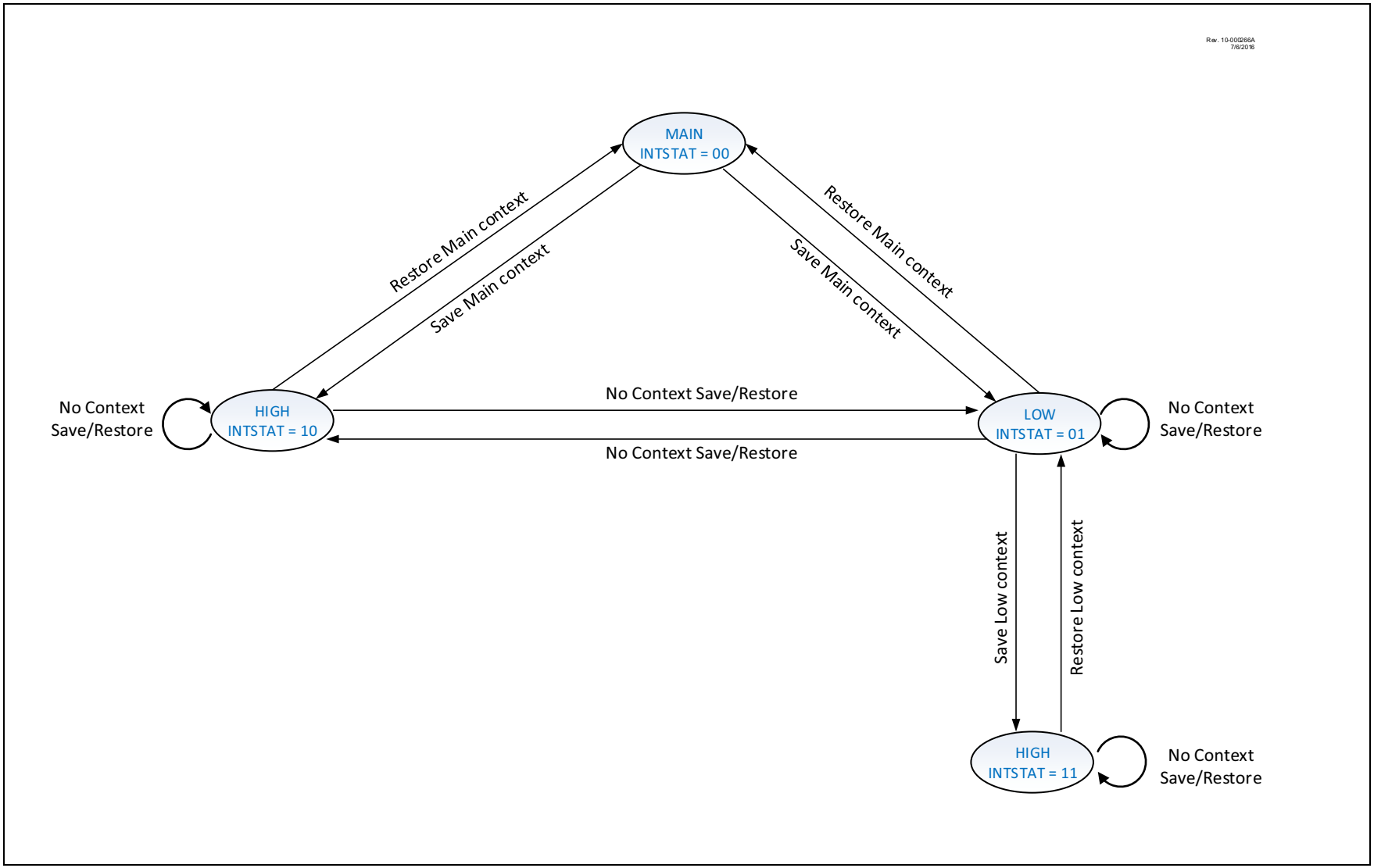
Context save/restore works the same way in both states of MVECEN. When IPEN = 0, there is only one level interrupt active. Hence, only the main context is saved when an interrupt is received.

9.5.1 ACCESSING SHADOW REGISTERS

The Interrupt controller automatically saves the context information in the shadow registers available in Bank 56. Both the saved context values (i.e., main routine and low ISR) can be accessed using the same set of shadow registers. By clearing the SHADLO bit in the SHADCON register (Register 9-43), the CPU register values saved for main routine context can accessed, and by setting the SHADLO bit of the CPU register, values saved for low ISR context can accessed. Low ISR context is automatically restored to the CPU registers upon exiting the high ISR. Similarly, the main context is automatically restored to the CPU registers upon exiting the low ISR.

The Shadow registers in Bank 56 are readable and writable, so if the user desires to modify the context, then the corresponding shadow register should be modified and the value will be restored when exiting the ISR. Depending on the user's application, other registers may also need to be saved.

FIGURE 9-6: CONTEXT SAVE STATE MACHINE DIAGRAM



© 2017-2020 Microchip Technology Inc.

9.6 Returning from Interrupt Service Routine (ISR)

The “Return from Interrupt” instruction (`RETFIE`) is used to mark the end of an ISR.

When `RETFIE 1` instruction is executed, the PC is loaded with the saved PC value from the top of the PC stack. Saved context is also restored with the execution of this instruction. Thus, execution returns to the previous state of operation that existed before the interrupt occurred.

When `RETFIE 0` instruction is executed, the saved context is not restored back to the registers.

9.7 Interrupt Latency

By assigning each interrupt with a vector address/number (MVECEN = `1`), scanning of all interrupts is not necessary to determine the source of the interrupt.

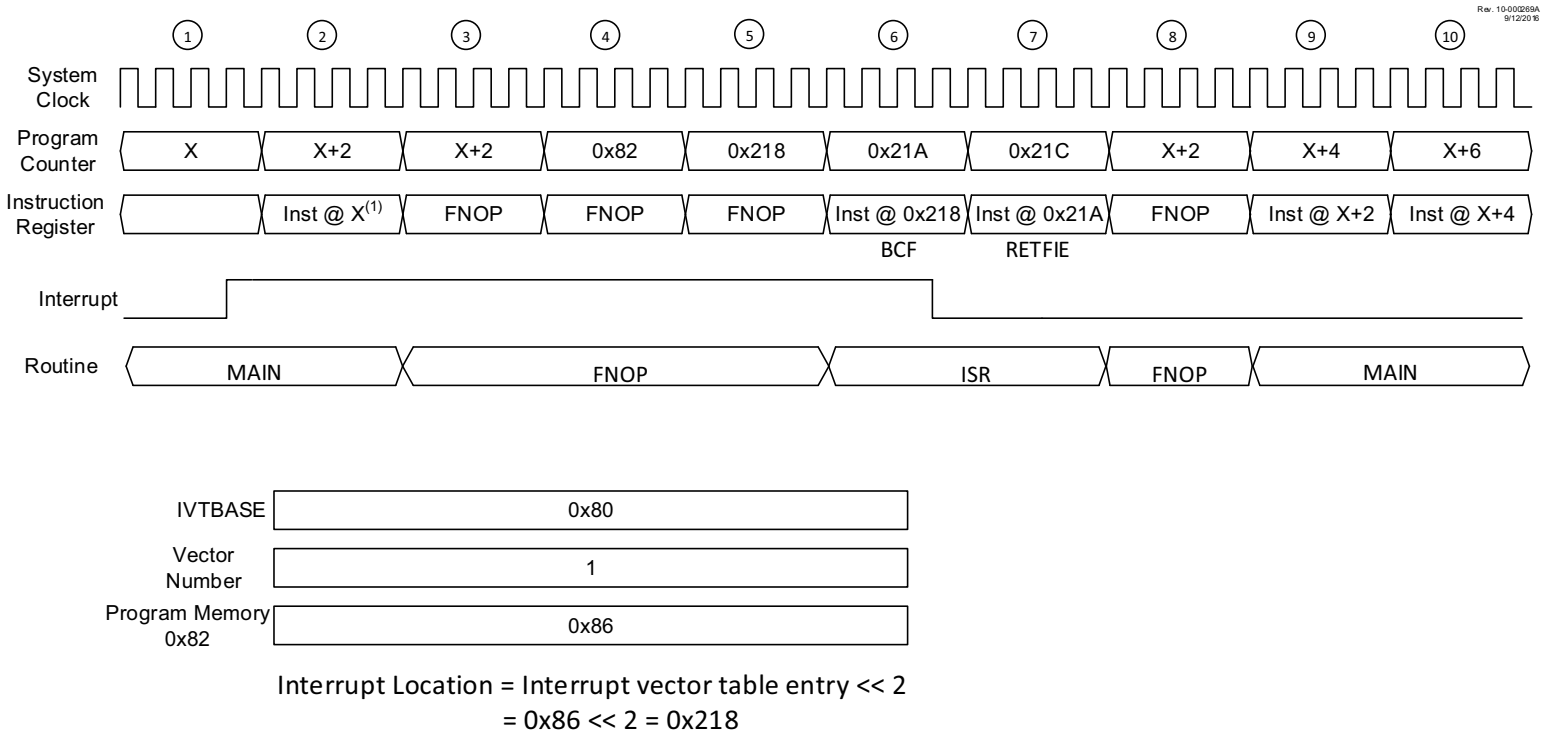
When MVECEN = `1`, Vectored interrupt controller requires three clock cycles to vector to the ISR from main routine, thereby removing dependency of interrupt timing on compiled code.

There is a fixed latency of three instruction cycles between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine. Figure 9-7, Figure 9-8 and Figure 9-9 illustrate the sequence of events when a peripheral interrupt is asserted when the last executed instruction is one-cycle, two-cycle and three-cycle respectively, when MVECEN = `1`.

After the Interrupt Flag Status bit is set, the current instruction completes executing. In the first latency cycle, the contents of the PC, STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U registers are context saved and the IVTBASE+ Vector number is calculated. In the second latency cycle, the PC is loaded with the calculated vector table address for the interrupt source and the starting address of the ISR is fetched. In the third latency cycle, the PC is loaded with the ISR address. All the latency cycles are executed as a `FNOP` instruction.

When MVECEN = `0`, Vectored interrupt controller requires two clock cycles to vector to the ISR from main routine. There is a latency of two instruction cycles plus the software latency between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine.

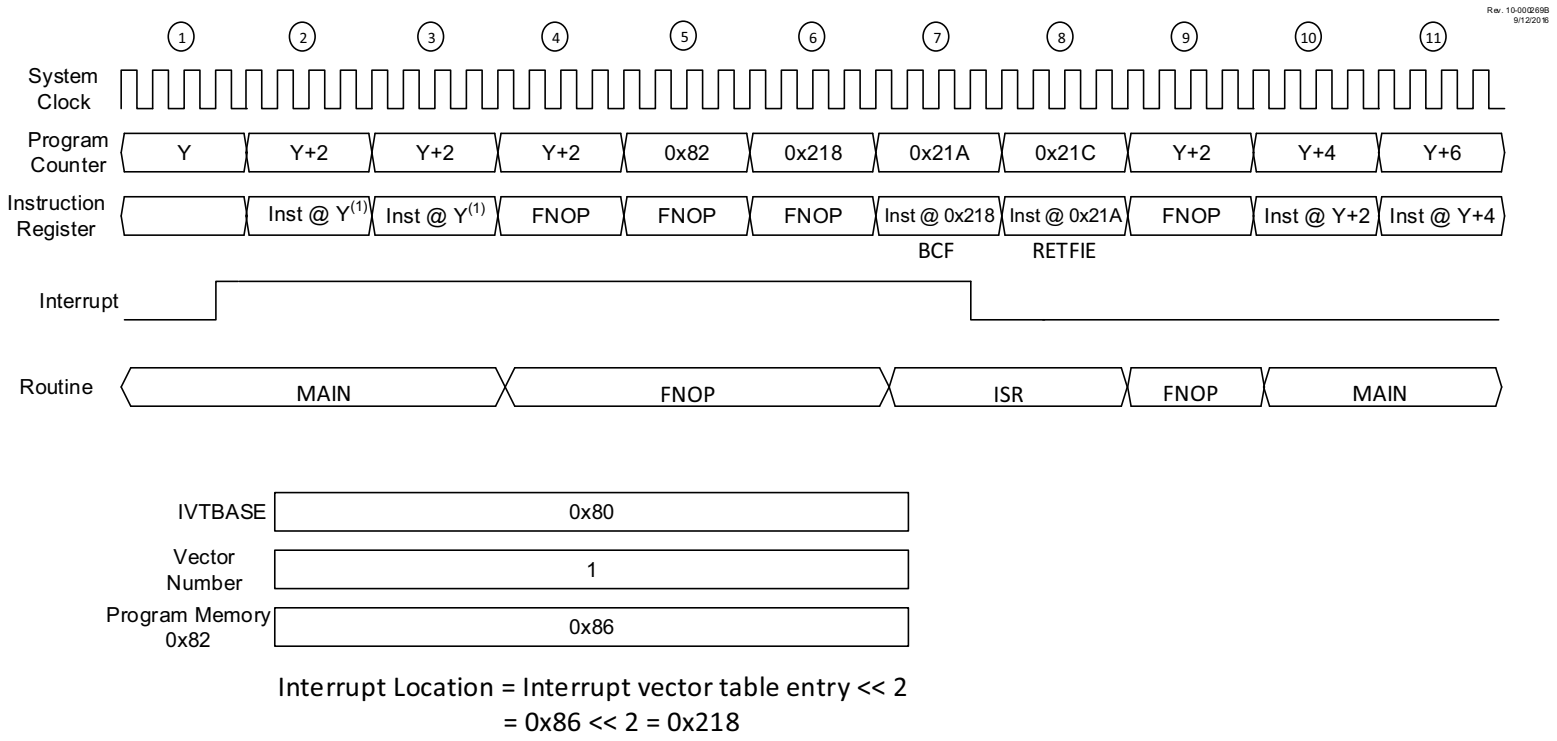
FIGURE 9-7: INTERRUPT TIMING DIAGRAM - ONE CYCLE INSTRUCTION



Note 1: Instruction @ X is a One-cycle Instruction

© 2017-2020 Microchip Technology Inc.

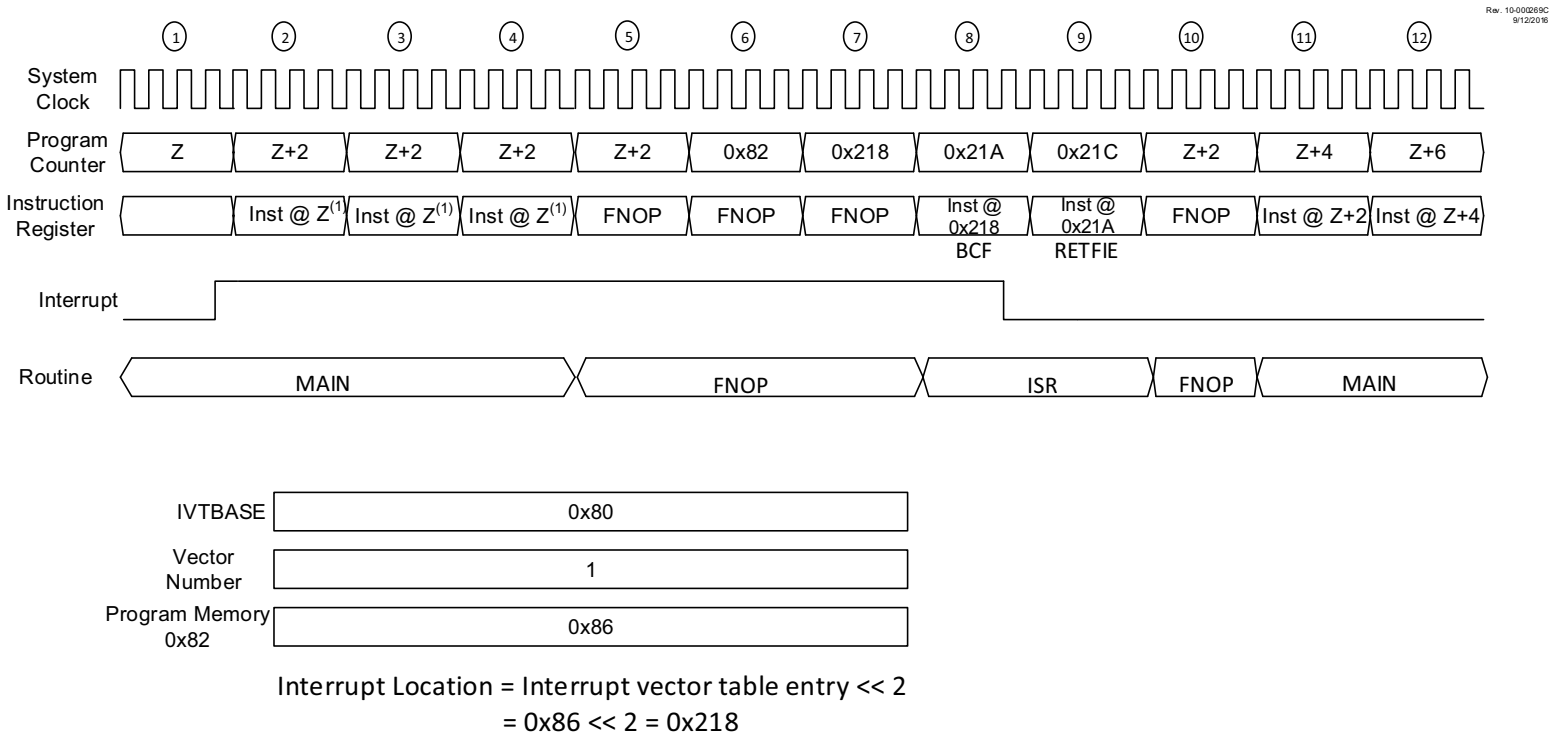
FIGURE 9-8: INTERRUPT TIMING DIAGRAM - TWO WORD INSTRUCTION



Note 1: Instruction @ Y is a Two-cycle instruction.

© 2017-2020 Microchip Technology Inc.

FIGURE 9-9: INTERRUPT TIMING DIAGRAM - THREE CYCLE INSTRUCTION



Note 1: Instruction @ Z is a Three-cycle instruction.

© 2017-2020 Microchip Technology Inc.

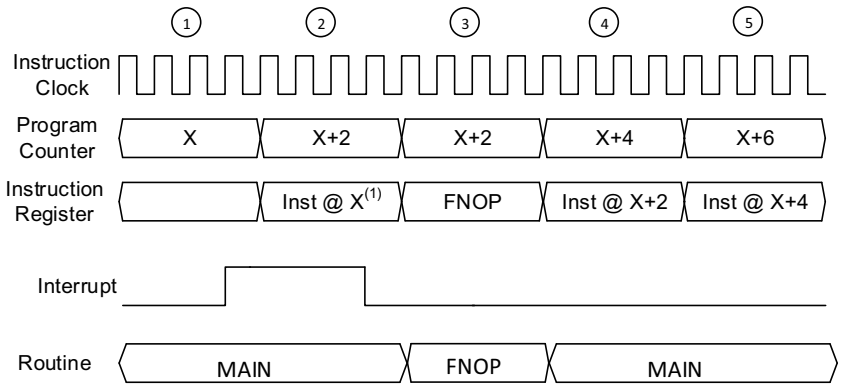
9.7.1 ABORTING INTERRUPTS

If the last instruction before the interrupt controller vectors to the ISR from main routine clears the GIE, PIE or PIR bit associated with the interrupt, the controller executes one force `NOP` cycle before it returns to the main routine.

Figure 9-10 illustrates the sequence of events when a peripheral interrupt is asserted and then cleared on the last executed instruction cycle.

If the GIE, PIE or PIR bit associated with the interrupt is cleared prior to vectoring to the ISR, then the controller continues executing the main routine.

FIGURE 9-10: INTERRUPT TIMING DIAGRAM - ABORTING INTERRUPTS



Note 1: Inst @ X clears the interrupt flag, Example BCF INTCON0, GIE.

© 2017-2020 Microchip Technology Inc.

9.8 Interrupt Setup Procedure

1. When using interrupt priority levels, set the IPEN bit in INTCON0 register and then select the user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPRx Control register.

Note:	At a device Reset, the IPRx registers are initialized, such that all user interrupt sources are assigned to high priority.

2. Clear the Interrupt Flag Status bit associated with the peripheral in the associated PIRx Status register.
3. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate PIEx Control register.
4. If the vector table is used (MVECEN = `1`), then setup the start address for the Interrupt Vector Table using the IVTBASE register. See **Section 9.2.2 "Interrupt Vector Table Contents"**.
5. Once the IVTBASE is written to, set the Interrupt enable bits in INTCON0 register.
6. An example of setting up interrupts and ISRs using assembly and C can be found in Examples 9-3 and 9-4.

9.9 External Interrupt Pins

The PIC18(L)F26/27/45/46/47/55/56/57K42 devices have three external interrupt sources which can be assigned to any pin on different ports based on the PPS settings. Refer **Section 17.0 "Peripheral Pin Select (PPS) Module"** for possible rerouting options. The external interrupt sources are edge-triggered. If the corresponding INTxEDG bit in the INTCON0 register is set (= `1`), the interrupt is triggered by a rising edge. If the bit is clear, the trigger is on the falling edge.

When a valid edge appears on the INTx pin, the corresponding flag bit, INTxF in the PIRx registers, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxE. Flag bit, INTxF, must be cleared by software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1 and INT2) can wake-up the processor from Idle or Sleep modes if bit INTxE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE/GIEH, is set, the processor will branch to the interrupt vector following wake-up. Interrupt priority is determined by the value contained in the interrupt priority bits, INT0IP, INT1IP and INT2IP of the IPRx registers.

9.10 Wake-up from Sleep

The interrupt controller provides a wake-up request to the CPU whenever an interrupt event occurs, if the interrupt event is enabled. This occurs regardless of whether the part is in Run, Idle/Doze or Sleep modes. The status of the GIEH/GIEL bits has no effect on the wake-up request. The wake-up request will be asynchronous to all clocks.

9.11 Interrupt Compatibility

When the MVECEN bit in Configuration Word 2L is cleared (Register 5-3), the Interrupt Vector Table feature is disabled and interrupts are compatible with previous high performance 8-bit PIC18 microcontroller devices. In this mode, the Interrupt Vector Table priority has no effect.

When the IPEN bit is also cleared, the interrupt priority feature is disabled and interrupts are compatible with PIC®16 microcontroller mid-range devices. All interrupts branch to address 0008h since the interrupt priority is disabled.

EXAMPLE 9-3: SETTING UP VECTORED INTERRUPTS USING MPASM

```
; Each ISR routine must have a predetermined origin otherwise there will be
; an assembly error because the address is not determined until link time
; which is too late to do the divide by 4 math on the address.
; Predetermined addresses must be evenly divisible by 4.

ISRCLC2  CODE    0x7E00
; CLC2 interrupt service code here.
   BANKSEL   PIR7
   BCF       PIR7, CLC2IF
   RETFIE    FAST

ISRTMR0  CODE    0x7E40
; Timer0 interrupt service code here.
   BANKSEL   PIR3
   BCF       PIR3, TMR0IF
   RETFIE    FAST

ISRTMR4  CODE    0x7E60
; Timer4 interrupt service code here.
   BANKSEL   PIR7
   BCF       PIR7, TMR4IF
   RETFIE    FAST

IntInit:
   ; Disable all interrupts
   BCF       INTCON0, GIE, ACCESS

   ; Set IVTBASE (optional - default is 0x000008)
   CLRF      IVTBASEU, ACCESS
   MOVLW     0x7F
   MOVWF     IVTBASEH, ACCESS
   CLRF      IVTBASEL, ACCESS

   ; Clear any interrupt flags before enabling interrupts
   BANKSEL   PIR7
   BCF       PIR7, CLC2IF
   BCF       PIR3, TMR0IF
   BCF       PIR7, TMR4IF

   ; Enable interrupts
   BANKSEL   PIE7
   BSF       PIE7, CLC2IE
   BSF       PIE3, TMR0IE
   BSF       PIE7, TMR4IE

   ; Set interrupt priorities if necessary
   BANKSEL   IPR7
   BSF       INTCON0, IPEN_INTCON0, ACCESS   ; Enable interrupt priority
   BCF       IPR7, CLC2IP                    ; Make CLC2 interrupt low priority

   ; Enable interrupts
   BSF     INTCON0, GIEH, ACCESS
   BSF     INTCON0, GIEL, ACCESS

   RETURN 1

; Save TMR0ISR in vector table (IVTBASE+31*2)
ISR1   CODE      0x7F3E
       DW        (0x7E40>>2)             ; (TMR0ISR/4)

; Save TMR4ISR in vector table (IVTBASE+56*2)
ISR2   CODE      0x7F70
       DW        (0x7E60>>2)             ; (TMR4ISR/4)

; Save CLC2ISR in vector table (IVTBASE+60*2)
ISR3   CODE      0x7F78
       DW        (0x7E00>>2)             ; (CLC2ISR/4)
```

EXAMPLE 9-4: SETTING UP VECTORED INTERRUPTS USING XC8

```
// NOTE 1: If IVTBASE is changed from its default value of 0x000008, then the
// "base(...)" argument must be provided in the ISR. Otherwise the vector
// table will be placed at 0x0008 by default regardless of the IVTBASE value.

// NOTE 2: When MVECEN=0 and IPEN=1, a separate argument as "high_priority"
// or "low_priority" can be used to distinguish between the two ISRs.
// If the argument is not provided, the ISR is considered high priority
// by default.

// NOTE 3: Multiple interrupts can be handled by the same ISR if they are
// specified in the "irq(...)" argument. Ex: irq(IRQ_TMR0, IRQ_CCP1)

void __interrupt(irq(IRQ_TMR0), base(0x4008)) TMR0_ISR(void)
{
        PIR3bits.TMR0IF = 0;                  // Clear the interrupt flag
        LATCbits.LC0 ^= 1;                    // ISR code goes here
}

void __interrupt(irq(default), base(0x4008)) DEFAULT_ISR(void)
{
        // Unhandled interrupts go here
}

void INTERRUPT_Initialize (void)
{
        INTCON0bits.GIEH = 1;                 // Enable high priority interrupts
        INTCON0bits.GIEL = 1;                 // Enable low priority interrupts
        INTCON0bits.IPEN = 1;                 // Enable interrupt priority

        PIE3bits.TMR0IE = 1;                  // Enable TMR0 interrupt
        PIE4bits.TMR1IE = 1;                  // Enable TMR1 interrupt

        IPR3bits.TMR0IP = 0;                  // Make TMR0 interrupt low priority

        // Change IVTBASE if required
        IVTBASEU = 0x00;                      // Optional
        IVTBASEH = 0x40;                      // Default is 0x0008
        IVTBASEL = 0x08;
}
```

9.12 Register Definitions: Interrupt Control

REGISTER 9-1: INTCON0: INTERRUPT CONTROL REGISTER 0

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1
GIE/GIEH	GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG
bit 7							bit 0

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

bit 7 **GIE/GIEH:** Global Interrupt Enable bits

If IPEN = 0:

GIE:

1 = Enables all unmasked interrupts
0 = Disables all interrupts

If IPEN = 1:

GIEH:

1 = Enables all unmasked high priority interrupts: bit also needs to be set for enabling low priority interrupts
0 = Disables all interrupts

bit 6 **GIEL:** Global Low Priority Interrupt Enable bit

If IPEN = 0:

Reserved, read as ‘0’

If IPEN = 1:

GIEL:

1 = Enables all unmasked low priority interrupts, GIEH also needs to be set for low priority interrupts
0 = Disables all low priority

bit 5 **IPEN:** Interrupt Priority Enable bit

1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts; all interrupts are treated as high priority interrupts

bit 4-3 **Unimplemented**: Read as ‘0’

bit 2 **INT2EDG:** External Interrupt 2 Edge Select bit

1 = Interrupt on rising edge of INT2 pin
0 = Interrupt on falling edge of INT2 pin

bit 1 **INT1EDG:** External Interrupt 1 Edge Select bit

1 = Interrupt on rising edge of INT1 pin
0 = Interrupt on falling edge of INT1 pin

bit 0 **INT0EDG:** External Interrupt 0 Edge Select bit

1 = Interrupt on rising edge of INT0 pin
0 = Interrupt on falling edge of INT0 pin

REGISTER 9-2: INTCON1: INTERRUPT CONTROL REGISTER 1

R-0/0	R-0/0	U-0	U-0	U-0	U-0	U-0	U-0
STAT[1:0]		—	—	—	—	—	—
bit 7							bit 0

Legend:		
HC = Bit is cleared by hardware		
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 7-6 **STAT[1:0]:** Interrupt State Status bits
- 11 = High priority ISR executing, high priority interrupt was received while a low priority ISR was executing
- 10 = High priority ISR executing, high priority interrupt was received in main routine
- 01 = Low priority ISR executing, low priority interrupt was received in main routine
- 00 = Main routine executing

bit 5-0 **Unimplemented**: Read as ‘0’

REGISTER 9-3: PIR0: PERIPHERAL INTERRUPT REQUEST REGISTER 0

R-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W-0/0
IOCIF(2)	CRCIF	SCANIF	NVMIF	CSWIF(3)	OSFIF	HLVDIF	SWIF
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	HS = Bit is set in hardware

bit 7 **IOCIF:** Interrupt-on-Change Interrupt Flag bit(2)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 6 **CRCIF:** CRC Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5 **SCANIF:** Memory Scanner Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 4 **NVMIF:** NVM Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 3 **CSWIF:** Clock Switch Interrupt Flag bit(3)
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **OSFIF:** Oscillator Fail Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **HLVDIF:** HLVD Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **SWIF:** Software Interrupt Flag bit
1 = Interrupt will trigger (bit is set and cleared by user software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: IOCIF is a read-only bit. To clear the interrupt condition, all bits in the IOCxF registers must be cleared.

3: The CSWIF interrupt will not wake the system from Sleep. The system will sleep until another interrupt causes the wake-up.

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
SMT1PWAIF	SMT1PRAIF	SMT1IF	C1IF	ADTIF	ADIF	ZCDIF	INT0IF(2)
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	HS = Bit is set in hardware

bit 7 **SMT1PWAIF:** SMT1 Pulse-Width Acquisition Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6 **SMT1PRAIF:** SMT1 Period Acquisition Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5 **SMT1IF:** SMT1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 4 **C1IF:** CMP1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 3 **ADTIF:** ADC Threshold Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **ADIF:** ADC Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **ZCDIF:** ZCD Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **INT0IF:** External Interrupt 0 Interrupt Flag bit(2)
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: The external interrupt GPIO pin is selected by the INTxPPS register.

REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REGISTER 2(1)

R-0/0	R-0/0	R-0/0	R-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
I2C1RXIF(2)	SPI1IF(3)	SPI1TXIF(4)	SPI1RXIF(4)	DMA1AIF	DMA1ORIF	DMA1DCNTIF	DMA1SCNTIF
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	HS = Hardware set

bit 7 **I2C1RXIF:** I^2C1 Receive Interrupt Flag bit(2)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 6 **SPI1IF:** SPI1 Interrupt Flag bit(3)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 5 **SPI1TXIF:** SPI1 Transmit Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 4 **SPI1RXIF:** SPI1 Receive Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 3 **DMA1AIF:** DMA1 Abort Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **DMA1ORIF:** DMA1 Overrun Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **DMA1DCNTIF:** DMA1 Destination Count Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **DMA1SCNTIF:** DMA1 Source Count Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: I2CxTXIF and I2CxRXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2CxSTAT1 register must be set.

3: SPIxIF is a read-only bit. To clear the interrupt condition, all bits in the SPIxINTF register must be cleared.

4: SPIxTXIF and SPIxRXIF are read-only bits and cannot be set/cleared by the software.

REGISTER 9-6: PIR3: PERIPHERAL INTERRUPT REGISTER 3(1)

R/W/HS-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
TMR0IF	U1IF(2)	U1EIF(3)	U1TXIF(4)	U1RXIF(4)	I2C1EIF(5)	I2C1IF(6)	I2C1TXIF(7)
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	HS = Bit is set in hardware

bit 7 **TMR0IF:** TMR0 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6 **U1IF:** UART1 Interrupt Flag bit(2)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 5 **U1EIF:** UART1 Framing Error Interrupt Flag bit(3)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 4 **U1TXIF:** UART1 Transmit Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 3 **U1RXIF:** UART1 Receive Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 2 **I2C1EIF:** I^2C1 Error Interrupt Flag bit(5)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 1 **I2C1IF:** I^2C1 Interrupt Flag bit(6)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 0 **I2C1TXIF:** I^2C1 Transmit Interrupt Flag bit(7)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: UxIF is a read-only bit. To clear the interrupt condition, all bits in the UxUIR register must be cleared.

3: UxEIF is a read-only bit. To clear the interrupt condition, all bits in the UxERRIR register must be cleared.

4: UxTXIF and UxRXIF are read-only bits and cannot be set/cleared by the software.

5: I2CxEIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxERR register must be cleared.

6: I2CxIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxPIR register must be cleared.

7: I2CxTXIF and I2CxRXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2CxSTAT1 register must be set.

REGISTER 9-7: PIR4: PERIPHERAL INTERRUPT REGISTER 4(1)

R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
CLC1IF	CWG1IF	NCO1IF	—	CCP1IF	TMR2IF	TMR1GIF	TMR1IF
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	HS = Bit is set in hardware

bit 7 **CLC1IF:** CLC1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6 **CWG1IF:** CWG1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5 **NCO1IF:** NCO1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **CCP1IF:** CCP1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **TMR2IF:** TMR2 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **TMR1GIF:** TMR1 Gate Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **TMR1IF:** TMR1 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 9-8: PIR5: PERIPHERAL INTERRUPT REGISTER 5(1)

R-0/0	R-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
I2C2TXIF(2)	I2C2RXIF(2)	DMA2AIF	DMA2ORIF	DMA2DCNTIF	DMA2SCNTIF	C2IF	INT1IF(3)
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	HS = Bit is set in hardware

bit 7 **I2C2TXIF:** I^2C2 Transmit Interrupt Flag bit(2)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 6 **I2C2RXIF:** I^2C2 Receive Interrupt Flag bit(2)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 5 **DMA2AIF:** DMA2 Abort Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 4 **DMA2ORIF:** DMA2 Overrun Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 3 **DMA2DCNTIF:** DMA2 Destination Count Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **DMA2SCNTIF:** DMA2 Source Count Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **C2IF:** C2 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **INT1IF:** External Interrupt 1 Interrupt Flag bit(3)
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: I2CxTXIF and I2CxRXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2CxSTAT1 register must be set.

3: The external interrupt GPIO pin is selected by the INTxPPS register.

REGISTER 9-9: PIR6: PERIPHERAL INTERRUPT REGISTER 6(1)

R/W/HS-0/0	R/W/HS-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
TMR3GIF	TMR3IF	U2IF(2)	U2EIF(3)	U2TXIF(4)	U2RXIF(4)	I2C2EIF(5)	I2C2IF(6)
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	HS = Bit is set in hardware

bit 7 **TMR3GIF:** TMR3 Gate Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6 **TMR3IF:** TMR3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5 **U2IF:** UART2 Interrupt Flag bit(2)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 4 **U2EIF:** UART2 Framing Error Interrupt Flag bit(3)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 3 **U2TXIF:** UART2 Transmit Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 2 **U2RXIF:** UART2 Receive Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 1 **I2C2EIF:** I^2C2 Error Interrupt Flag bit(5)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 0 **I2C2IF:** I^2C2 Interrupt Flag bit(6)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: UxIF is a read-only bit. To clear the interrupt condition, all bits in the UxUIR register must be cleared.

3: UxEIF is a read-only bit. To clear the interrupt condition, all bits in the UxERRIR register must be cleared.

4: UxTXIF and UxRXIF are read-only bits and cannot be set/cleared by the software.

5: I2CxEIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxERR register must be cleared.

6: I2CxIF is a read-only bit. To clear the interrupt condition, all bits in the I2CxPIR register must be cleared.

REGISTER 9-10: PIR7: PERIPHERAL INTERRUPT REGISTER 7(1)

U-0	U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	U-0	R/W/HS-0/0	R/W/HS-0/0
—	—	INT2IF(2)	CLC2IF	CWG2IF	—	CCP2IF	TMR4IF
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	HS = Bit is set in hardware

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5 **INT2IF:** External Interrupt 2 Interrupt Flag bit(2)
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 4 **CLC2IF:** CLC2 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 3 **CWG2IF:** CWG2 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **CCP2IF:** CCP2 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **TMR4IF:** TMR4 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: The external interrupt GPIO pin is selected by the INTxPPS register.

REGISTER 9-11: PIR8: PERIPHERAL INTERRUPT REGISTER 8(1)

R/W/HS-0/0	R/W/HS-0/0	U-0	U-0	U-0	U-0	U-0	U-0
TMR5GIF	TMR5IF	—	—	—	—	—	—
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	HS = Bit is set in hardware

bit 7 **TMR5GIF:** TMR5 Gate Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6 **TMR5IF:** TMR5 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5-0 **Unimplemented:** Read as ‘0’

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 9-12: PIR9: PERIPHERAL INTERRUPT REGISTER 9(1)

U-0	U-0	U-0	U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
—	—	—	—	CLC3IF	CWG3IF	CCP3IF	TMR6IF
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 7-4 **Unimplemented:** Read as ‘0’

bit 3 **CLC3IF:** CLC3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **CWG3IF:** CWG3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **CCP3IF:** CCP3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **TMR6IF:** TMR6 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 9-13: PIR10: PERIPHERAL INTERRUPT REGISTER 10(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W/HS-0/0	R/W/HS-0/0
—	—	—	—	—	—	CLC4IF	CCP4IF
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	HS = Bit is set in hardware

bit 7-2 **Unimplemented:** Read as ‘0’

bit 1 **CLC4IF:** CLC4 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **CCP4IF:** CCP4 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 9-14: PIE0: PERIPHERAL INTERRUPT ENABLE REGISTER 0

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
IOCIE	CRCIE	SCANIE	NVMIE	CSWIE	OSFIE	HLVDIE	SWIE
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 **IOCIE:** Interrupt-on-Change Enable bit
1 = Enabled
0 = Disabled

bit 6 **CRCIE:** CRC Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **SCANIE:** Memory Scanner Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **NVMIE:** NVM Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **CSWIE:** Clock Switch Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **OSFIE:** Oscillator Fail Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **HLVDIE:** HLVD Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **SWIE:** Software Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-15: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SMT1PWAIE	SMT1PRAIE	SMT1IE	C1IE	ADTIE	ADIE	ZCDIE	INT0IE
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 **SMT1PWAIE:** SMT1 Pulse Width Acquisition Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **SMT1PRAIE:** SMT1 Period Acquisition Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **SMT1IE:** SMT1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **C1IE:** C1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **ADTIE:** ADC Threshold Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **ADIE:** ADC Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **ZCDIE:** ZCD Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **INT0IE:** External Interrupt 0 Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-16: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
I2C1RXIE	SPI1IE	SPI1TXIE	SPI1RXIE	DMA1AIE	DMA1ORIE	DMA1DCNTIE	DMA1SCNTIE
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 **I2C1RXIE:** I^2C1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **SPI1IE:** SPI1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **SPI1TXIE:** SPI1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **SPI1RXIE:** SPI1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **DMA1AIE:** DMA1 Abort Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **DMA1ORIE:** DMA1 Overrun Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **DMA1DCNTIE:** DMA1 Destination Count Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **DMA1SCNTIE:** DMA1 Source Count Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-17: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE
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 **TMR0IE:** TMR0 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **U1IE:** UART1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **U1EIE:** UART1 Framing Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **U1TXIE:** UART1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **U1RXIE:** UART1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **I2C1EIE:** I^2C1 Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **I2C1IE:** I^2C1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **I2C1TXIE:** I^2C1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-18: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
CLC1IE	CWG1IE	NCO1IE	—	CCP1IE	TMR2IE	TMR1GIE	TMR1IE
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 **CLC1IE:** CLC1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **CWG1IE:** CWG1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **NCO1IE:** NCO1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **TMR2IE:** TMR2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR1GIE:** TMR1 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR1IE:** TMR1 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-19: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
I2C2TXIE	I2C2RXIE	DMA2AIE	DMA2ORIE	DMA2DCNTIE	DMA2SCNTIE	C2IE	INT1IE
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 **I2C2TXIE:** I^2C2 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **I2C2RXIE:** I^2C2 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **DMA2AIE:** DMA2 Abort Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **DMA2ORIE:** DMA2 Overrun Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **DMA2DCNTIE:** DMA2 Destination Count Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **DMA2SCNTIE:** DMA2 Source Count Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **C2IE:** C2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **INT1IE:** External Interrupt 1 Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-20: PIE6: PERIPHERAL INTERRUPT ENABLE REGISTER 6

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TMR3GIE	TMR3IE	U2IE	U2EIE	U2TXIE	U2RXIE	I2C2EIE	I2C2IE
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 **TMR3GIE:** TMR3 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **TMR3IE:** TMR3 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **U2IE:** UART2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **U2EIE:** UART2 Framing Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **U2TXIE:** UART2 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **U2RXIE:** UART2 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **I2C2EIE:** I^2C2 Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **I2C2IE:** I^2C2 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-21: PIE7: PERIPHERAL INTERRUPT ENABLE REGISTER 7

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0
—	—	INT2IE	CLC2IE	CWG2IE	—	CCP2IE	TMR4IE
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 **Unimplemented:** Read as ‘0’

bit 5 **INT2IE:** External Interrupt 2 Enable bit
1 = Enabled
0 = Disabled

bit 4 **CLC2IE:** CLC2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **CWG2IE:** CWG2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **CCP2IE:** CCP2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR4IE:** TMR4 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-22: PIE8: PERIPHERAL INTERRUPT ENABLE REGISTER 8

R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	U-0	U-0
TMR5GIE	TMR5IE	—	—	—	—	—	—
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 **TMR5GIE:** TMR5 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **TMR5IE:** TMR5 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5-0 **Unimplemented:** Read as ‘0’

REGISTER 9-23: PIE9: PERIPHERAL INTERRUPT ENABLE REGISTER 9

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CLC3IE	CWG3IE	CCP3IE	TMR6IE
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 7-4 **Unimplemented:** Read as ‘0’

bit 3 **CLC3IE:** CLC3 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **CWG3IE:** CWG3 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **CCP3IE:** CCP3 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR6IE:** TMR6 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-24: PIE10: PERIPHERAL INTERRUPT ENABLE REGISTER 10

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	CLC4IE	CCP4IE
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 7-2 **Unimplemented:** Read as ‘0’

bit 1 **CLC4IE:** CLC4 Interrupt Enable bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **CCP4IE:** CCP4 Interrupt Enable bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

REGISTER 9-25: IPR0: PERIPHERAL INTERRUPT PRIORITY REGISTER 0

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
IOCIP	CRCIP	SCANIP	NVMIP	CSWIP	OSFIP	HLVDIP	SWIP
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 **IOCIP:** Interrupt-on-Change Priority bit
1 = High priority
0 = Low priority

bit 6 **CRCIP:** CRC Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **SCANIP:** Memory Scanner Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **NVMIP:** NVM Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **CSWIP:** Clock Switch Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **OSFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **HLVDIP:** HLVD Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **SWIP:** Software Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-26: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
SMT1PWAIP	SMT1PRAIP	SMT1IP	C1IP	ADTIP	ADIP	ZCDIP	INT0IP
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 **SMT1PWAIP:** SMT1 Pulse Width Acquisition Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **SMT1PRAIP:** SMT1 Period Acquisition Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **SMT1IP:** SMT1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **C1IP:** C1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **ADTIP:** ADC Threshold Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **ADIP:** ADC Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **ZCDIP:** ZCD Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **INT0IP:** External Interrupt 0 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-27: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
I2C1RXIP	SPI1IP	SPI1TXIP	SPI1RXIP	DMA1AIP	DMA1ORIP	DMA1DCNTIP	DMA1SCNTIP
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 **I2C1RXIP:** I^2C1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **SPI1IP:** SPI1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **SPI1TXIP:** I^2C1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **SPI1RXIP:** SPI1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **DMA1AIP:** DMA1 Abort Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **DMA1ORIP:** DMA1 Overrun Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **DMA1DCNTIP:** DMA1 Destination Count Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **DMA1SCNTIP:** DMA1 Source Count Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-28: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
TMR0IP	U1IP	U1EIP	U1TXIP	U1RXIP	I2C1EIP	I2C1IP	I2C1TXIP
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 **TMR0IP:** TMR0 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **U1IP:** UART1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **U1EIP:** UART1 Framing Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **U1TXIP:** UART1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **U1RXIP:** UART1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **I2C1EIP:** I^2C1 Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **I2C1IP:** I^2C1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **I2C1TXIP:** I^2C1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-29: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4

R/W-1/1	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
CLC1IP	CWG1IP	NCO1IP	—	CCP1IP	TMR2IP	TMR1GIP	TMR1IP
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 **CLC1IP:** CLC1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **CWG1IP:** CWG1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **NCO1IP:** NCO1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **TMR2IP:** TMR2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR1GIP:** TMR1 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-30: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
I2C2TXIP	I2C2RXIP	DMA2AIP	DMA2ORIP	DMA2DCNTIP	DMA2SCNTIP	C2IP	INT1IP
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 **I2C2TXIP:** I^2C2 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **I2C2RXIP:** I^2C2 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **DMA2AIP:** DMA2 Abort Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **DMA2ORIP:** DMA2 Overrun Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **DMA2DCNTIP:** DMA2 Destination Count Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **DMA2SCNTIP:** DMA2 Source Count Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **C2IP:** C2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **INT1IP:** External Interrupt 1 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-31: IPR6: PERIPHERAL INTERRUPT PRIORITY REGISTER 6

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
TMR3GIP	TMR3IP	U2IP	U2EIP	U2TXIP	U2RXIP	I2C2EIP	I2C2IP
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 **TMR3GIP:** TMR3 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **TMR3IP:** TMR3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **U2IP:** UART2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **U2EIP:** UART2 Framing Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **U2TXIP:** UART2 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **U2RXIP:** UART2 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **I2C2EIP:** I^2C2 Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **I2C2IP:** I^2C2 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-32: IPR7: PERIPHERAL INTERRUPT PRIORITY REGISTER 7

U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1
—	—	INT2IP	CLC2IP	CWG2IP	—	CCP2IP	TMR4IP
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 **Unimplemented:** Read as ‘0’

bit 5 **INT2IP:** External Interrupt 2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **CLC2IP:** CLC2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **CWG2IP:** CWG2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **CCP2IP:** CRC Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR4IP:** TMR4 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-33: IPR8: PERIPHERAL INTERRUPT PRIORITY REGISTER 8

R/W-1/1	R/W-1/1	U-0	U-0	U-0	U-0	U-0	U-0
TMR5GIP	TMR5IP	—	—	—	—	—	—
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 **TMR5GIP:** TMR5 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **TMR5IP:** TMR5 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5-0 **Unimplemented:** Read as ‘0’

REGISTER 9-34: IPR9: PERIPHERAL INTERRUPT PRIORITY REGISTER 9

U-0	U-0	U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	—	—	CLC3IP	CWG3IP	CCP3IP	TMR6IP
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 7-4 **Unimplemented:** Read as ‘0’

bit 3 **CLC3IP:** CLC3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CWG3IP:** CWG3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **CCP3IP:** CCP3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR6IP:** TMR6 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-35: IPR10: PERIPHERAL INTERRUPT PRIORITY REGISTER 10

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	CLC4IP	CCP4IP
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 7-2 **Unimplemented:** Read as ‘0’

bit 1 **CLC4IP:** CLC4 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP4IP:** CCP4 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-36: IVTBASEU: INTERRUPT VECTOR TABLE BASE ADDRESS UPPER REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	BASE[20:16]				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **BASE[20:16]**: Interrupt Vector Table Base Address bits

REGISTER 9-37: IVTBASEH: INTERRUPT VECTOR TABLE BASE ADDRESS HIGH REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BASE[15:8]							
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 7-0 **BASE[15:8]**: Interrupt Vector Table Base Address bits

REGISTER 9-38: IVTBASEL: INTERRUPT VECTOR TABLE BASE ADDRESS LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-1/1	R/W-0/0	R/W-0/0	R/W-0/0
BASE[7:0]							
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 7-0 **BASE[7:0]**: Interrupt Vector Table Base Address bits

REGISTER 9-39: IVTADU: INTERRUPT VECTOR TABLE ADDRESS UPPER REGISTER

U-0	U-0	U-0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
—	—	—	AD[20:16]				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **AD[20:16]**: Interrupt Vector Table Address bits

REGISTER 9-40: IVTADH: INTERRUPT VECTOR TABLE ADDRESS HIGH REGISTER

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
AD[15:8]							
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 7-0 **AD[15:8]**: Interrupt Vector Table Address bits

REGISTER 9-41: IVTADL: INTERRUPT VECTOR TABLE ADDRESS LOW REGISTER

R-0/0	R-0/0	R-0/0	R-0/0	R-1/1	R-0/0	R-0/0	R-0/0
AD[7:0]							
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 7-0 **AD[7:0]**: Interrupt Vector Table Address bits

REGISTER 9-42: IVTLOCK: INTERRUPT VECTOR TABLE LOCK REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	IVTLOCKED(1,2)
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 7-1 **Unimplemented:** Read as ‘0’

bit 0 **IVTLOCKED**: IVT Registers Lock bits(1,2)
1 = IVTBASE Registers are locked and cannot be written
0 = IVTBASE Registers can be modified by write operations

Note 1: The IVTLOCK bit can only be set or cleared after the unlock sequence in Example 9-1.
2: If IVT1WAY = 1, the IVTLOCK bit cannot be cleared after it has been set. See Register 5-3.

REGISTER 9-43: SHADCON: SHADOW CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	SHADLO
bit 7							bit 0

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

bit 7-1 **Unimplemented:** Read as ‘0’

bit 0 **SHADLO:** Interrupt Shadow Register Access Switch bit
0 = Access Main Context for Interrupt Shadow Registers
1 = Access Low-Priority Interrupt Context for Interrupt Shadow Registers

TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON0	GIE/GIEH	GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	135
INTCON1	STAT[1:0]		—	—	—	—	—	—	136
PIE0	IOCIE	CRCIE	SCANIE	NVMIE	CSWIE	OSFIE	HLVDIE	SWIE	147
PIE1	SMT1PWAIE	SMT1PRAIE	SMT1IE	C1IE	ADTIE	ADIE	ZCDIE	INT0IE	148
PIE2	I2C1RXIE	SPI1IE	SPI1TXIE	SPI1RXIE	DMA1AIE	DMA1ORIE	DMA1DCNTIE	DMA1SCNTIE	149
PIE3	TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE	150
PIE4	CLC1IE	CWG1IE	NCO1IE	—	CCP1IE	TMR2IE	TMR1GIE	TMR1IE	151
PIE5	I2C2TXIE	I2C2RXIE	DMA2AIE	DMA2ORIE	DMA2DCNTIE	DMA2SCNTIE	C2IE	INT1IE	152
PIE6	TMR3GIE	TMR3IE	U2IE	U2EIE	U2TXIE	U2RXIE	I2C2EIE	I2C2IE	153
PIE7	—	—	INT2IE	CLC2IE	CWG2IE	—	CCP2IE	TMR4IE	154
PIE8	TMR5GIE	TMR5IE	—	—	—	—	—	—	155
PIE9	—	—	—	—	CLC3IE	CWG3IE	CCP3IE	TMR6IE	155
PIE10	—	—	—	—	—	—	CLC4IE	CCP4IE	156
PIR0	IOCIF	CRCIF	SCANIF	NVMIF	CSWIF	OSFIF	HLVDIF	SWIF	137
PIR1	SMT1PWAIF	SMT1PRAIF	SMT1IF	C1IF	ADTIF	ADIF	ZCDIF	INT0IF	138
PIR2	I2C1RXIF	SPI1IF	SPI1TXIF	SPI1RXIF	DMA1AIF	DMA1ORIF	DMA1DCNTIF	DMA1SCNTIF	139
PIR3	TMR0IF	U1IF	U1EIF	U1TXIF	U1RXIF	I2C1EIF	I2C1IF	I2C1TXIF	140
PIR4	CLC1IF	CWG1IF	NCO1IF	—	CCP1IF	TMR2IF	TMR1GIF	TMR1IF	141
PIR5	I2C2TXF	I2C2RXF	DMA2AIF	DMA2ORIF	DMA2DCNTIF	DMA2SCNTIF	C2IF	INT1IF	142
PIR6	TMR3GIF	TMR3IF	U2IF	U2EIF	U2TXIF	U2RXIF	I2C2EIF	I2C2IF	143
PIR7	—	—	INT2IF	CLC2IF	CWG2IF	—	CCP2IF	TMR4IF	144
PIR8	TMR5GIF	TMR5IF	—	—	—	—	—	—	145
PIR9	—	—	—	—	CLC3IF	CWG3IF	CCP3IF	TMR6IF	145
PIR10	—	—	—	—	—	—	CLC4IF	CCP4IF	146
IPR0	IOCIP	CRCIP	SCANIP	NVMIP	CSWIP	OSFIP	HLVDIP	SWIP	157
IPR1	SMT1PWAIP	SMT1PRAIP	SMT1IP	C1IP	ADTIP	ADIP	ZCDIP	INT0IP	158
IPR2	I2C1RIP	SPI1IP	SPI1TIP	SPI1RIP	DMA1AIP	DMA1ORIP	DMA1DCNTIP	DMA1SCNTIP	159
IPR3	TMR0IP	U1IP	U1EIP	U1TXIP	U1RXIP	I2C1EIP	I2C1IP	I2C1TXIP	160
IPR4	CLC1IP	CWG1IP	NCO1IP	—	CCP1IP	TMR2IP	TMR1GIP	TMR1IP	161
IPR5	I2C2TXP	I2C2RXP	DMA2AIP	DMA2ORIP	DMA2DCNTIP	DMA2SCNTIP	C2IP	INT1IP	162
IPR6	TMR3GIP	TMR3IP	U2IP	U2EIP	U2TXIP	U2RXIP	I2C2EIP	I2C2IP	163
IPR7	—	—	INT2IP	CLC2IP	CWG2IP	—	CCP2IP	TMR4IP	164
IPR8	TMR5GIP	TMR5IP	—	—	—	—	—	—	164
IPR9	—	—	—	—	CLC3IP	CWG3IP	CCP3IP	TMR6IP	165
IPR10	—	—	—	—	—	—	CLC4IP	CCP4IP	165
IVTBASEU	—	—	—	BASE[20:16]					166
IVTBASEH	BASE[15:8]								166
IVTBASEL	BASE[7:0]								166
IVTADU				AD[20:16]					167
IVTADH	AD[15:8]								167
IVTADL	AD[7:0]								167
IVTLOCK	—	—	—	—	—	—	—	IVTLOCKED	168

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for interrupts.

10.0 POWER-SAVING OPERATION MODES

The purpose of the Power-Down modes is to reduce power consumption. There are three Power-Down modes:

- Doze mode
- Sleep mode
- Idle mode

10.1 Doze Mode

Doze mode saves power by reducing CPU execution and program memory (PFM) access, without affecting peripheral operation.

10.1.1 DOZE OPERATION

When the Doze Enable bit is set (DOZEN = 1), the CPU executes one instruction cycle out of every N cycles as defined by the DOZE[2:0] bits of the CPUDOZE register. FOSC and FOSC/4 clock sources are unaffected in Doze mode and peripherals can continue using these sources.

10.1.2 INTERRUPTS DURING DOZE

When an interrupt occurs during Doze, the system behavior can be configured using the Recover-On-Interrupt bit (ROI) and the Doze-On-Exit bit (DOE). Refer to Table 10-2 for details about system behavior in all the cases for a transition from Main > ISR > Main. For PIC18(L)F26/27/45/46/47/55/56/57 devices, the transition from Main > ISR > Main always happens in Normal operation, regardless of the state of the DOZEN or DOE bits.

TABLE 10-1: SYSTEM BEHAVIOR FOR INTERRUPT DURING DOZE

DOZEN	ROI	Code Flow			
		Main	ISR (1)	Return to Main	
0	0	Normal operation	Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (unchanged)	If DOE = 1 when return from interrupt; Doze operation and DOZEN = 1 (in hardware)	If DOE = 0 when return from interrupt; Normal operation and DOZEN = 0 (in hardware)
0	1	Normal operation	Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (in hardware)		
1	0	Doze operation	Doze operation and DOE = DOZEN (in hardware) DOZEN = 1 (unchanged)		
1	1	Doze operation	Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (in hardware)		

Note 1: User software can change the DOE bit in ISR.

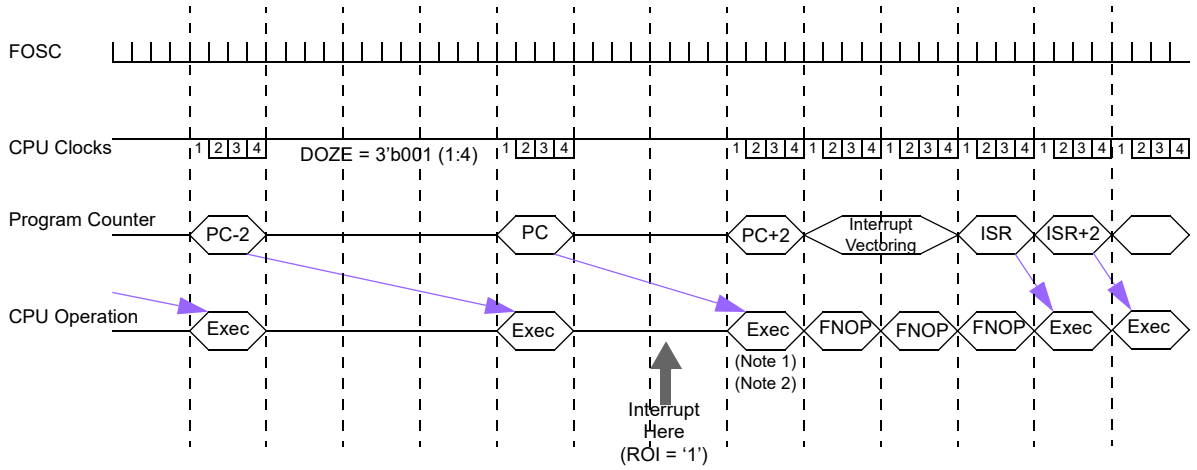
For example, if ROI = 1 and DOZE[2:0] = 001, the instruction cycle ratio is 1:4. The CPU and memory operate for one instruction cycle and stay idle for the next three instruction cycles. The Doze operation is illustrated in Figure 10-1.

EXAMPLE 10-1: DOZE SOFTWARE EXAMPLE

```
//Mainline operation
bool somethingToDo = FALSE:
void main()
{
   initializeSystem();
           // DOZE = 64:1 (for example)
           // ROI = 1;
   GIE = 1; // enable interrupts
   while (1)
   {
       // If ADC completed, process data
       if (somethingToDo)
       {
           doSomething();
           DOZEN = 1; // resume low-power
       }
   }
}

// Data interrupt handler
void interrupt()
{
   // DOZEN = 0 because ROI = 1
   if (ADIF)
   {
       somethingToDo = TRUE;
       DOE = 0; // make main() go fast
       ADIF = 0;
   }
   // else check other interrupts...
   if (TMR0IF)
   {
       timerTick++;
       DOE = 1; // make main() go slow
       TMR0IF = 0;
   }
}
```

FIGURE 10-1: DOZE MODE OPERATION EXAMPLE (DOZE[2:0] = 001, 1:4)



Note 1: Multi-cycle instructions are executed to completion before fetching the interrupt vector.
2: If the prefetched instruction clears the interrupt enable or GIEx, ISR vectoring will not occur, but DOZEN is cleared and the CPU will resume execution at full speed.

10.2 Sleep Mode

Sleep mode is entered by executing the SLEEP instruction, while the Idle Enable (IDLEN) bit of the CPUDOZE register is clear (IDLEN = 0).

Upon entering Sleep mode, the following conditions exist:

1. WDT will be cleared but keeps running if enabled for operation during Sleep
2. The $\overline{PD}$ bit of the STATUS register is cleared (Register 4-2)
3. The $\overline{TO}$ bit of the STATUS register is set (Register 4-2)
4. The CPU clock is disabled
5. LFINTOSC, SOSC, HFINTOSC and ADCRC are unaffected and peripherals using them may continue operation in Sleep.
6. I/O ports maintain the status they had before Sleep was executed (driving high, low, or high-impedance)
7. Resets other than WDT are not affected by Sleep mode

Refer to individual chapters for more details on peripheral operation during Sleep.

To minimize current consumption, the following conditions should be considered:

- I/O pins should not be floating
- External circuitry sinking current from I/O pins
- Internal circuitry sourcing current from I/O pins
- Current draw from pins with internal weak pull-ups
- Modules using any oscillator

I/O pins that are high-impedance inputs should be pulled to VDD or VSS externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See **Section 37.0 "5-Bit Digital-to-Analog Converter (DAC) Module"** and **Section 34.0 "Fixed Voltage Reference (FVR)"** for more information on these modules.

10.2.1 WAKE-UP FROM SLEEP

The device can wake up from Sleep through one of the following events:

1. External Reset input on $\overline{\text{MCLR}}$ pin, if enabled
2. BOR Reset, if enabled
3. Low-Power Brown-Out Reset (LPBOR), if enabled
4. POR Reset
5. Windowed Watchdog Timer, if enabled
6. All interrupt sources except clock switch interrupt can wake up the part.

The first five events will cause a device Reset. The last one event is considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to **Section 6.13 "Power Control (PCON0/PCON1) Register"**.

When the `SLEEP` instruction is being executed, the next instruction (PC + 2) is prefetched. For the device to wake-up through an interrupt event, the corresponding Interrupt Enable bit must be enabled. Wake-up will occur regardless of the state of the GIE bit. If the GIE bit is disabled, the device continues execution at the instruction after the `SLEEP` instruction. If the GIE bit is enabled, the device executes the instruction after the `SLEEP` instruction, the device will then call the Interrupt Service Routine. In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

The WDT is cleared when the device wakes-up from Sleep, regardless of the source of wake-up.

Upon a wake from a Sleep event, the core will wait for a combination of three conditions before beginning execution. The conditions are:

- PFM Ready
- COSC-Selected Oscillator Ready
- BOR Ready (unless BOR is disabled)

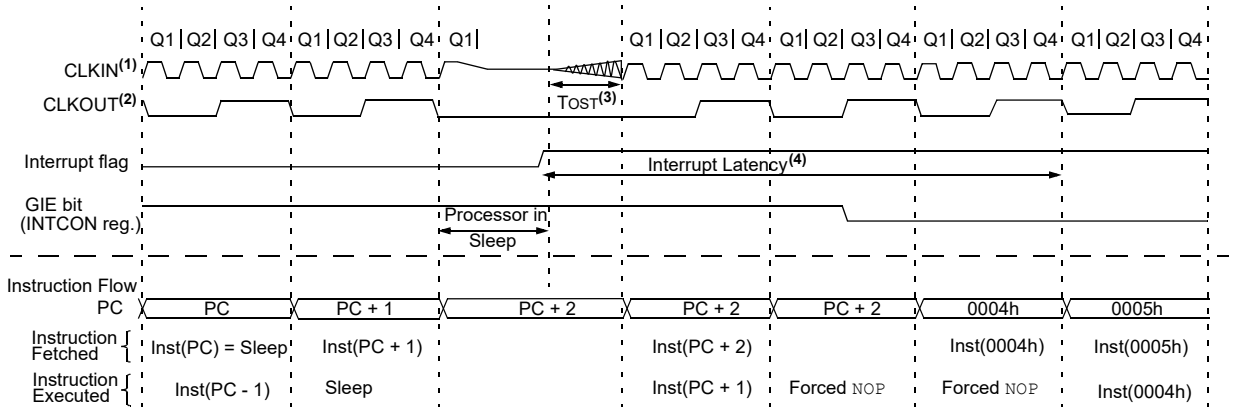
10.2.2 WAKE-UP USING INTERRUPTS

When any interrupt source, with the exception of the clock switch interrupt, has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a `SLEEP` instruction
  - `SLEEP` instruction will execute as a `NOP`
  - WDT and WDT prescaler will not be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will not be set
  - $\overline{\text{PD}}$ bit of the STATUS register will not be cleared
- If the interrupt occurs **during or after** the execution of a `SLEEP` instruction
  - `SLEEP` instruction will be completely executed
  - Device will immediately wake-up from Sleep
  - WDT and WDT prescaler will be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will be set
  - $\overline{\text{PD}}$ bit of the STATUS register will be cleared

Even if the flag bits were checked before executing a `SLEEP` instruction, it may be possible for flag bits to become set before the `SLEEP` instruction completes. To determine whether a `SLEEP` instruction executed, test the $\overline{\text{PD}}$ bit. If the $\overline{\text{PD}}$ bit is set, the `SLEEP` instruction was executed as a `NOP`.

FIGURE 10-2: 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 and INTOSC Oscillator modes.
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.

10.2.3 LOW-POWER SLEEP MODE

The PIC18F26/27/45/46/47/55/56/57K42 device family 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.

The PIC18F26/27/45/46/47/55/56/57K42 devices allow the user to optimize the operating current in Sleep, depending on the application requirements.

Low-Power Sleep mode can be selected by setting the VREGPM bit of the VREGCON register.

10.2.3.1 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.

10.2.3.2 Peripheral Usage in Sleep

Some peripherals that can operate in Sleep mode will not operate properly with the Low-Power Sleep mode selected. The Low-Power Sleep mode is intended for use with these peripherals:

- Brown-out Reset (BOR)
- Windowed Watchdog Timer (WWDT)
- External interrupt pin/Interrupt-On-Change pins
- Peripherals that run off external secondary clock source

It is the responsibility of the end user to determine what is acceptable for their application when setting the VREGPM settings in order to ensure operation in Sleep.

Note:	The PIC18F26/27/45/46/47/55/56/57K42 devices do not have a configurable Low-Power Sleep mode. PIC18F26/27/45/46/47/55/56/57K42 devices are unregulated and are always in the lowest power state when in Sleep, with no wake-up time penalty. These devices have a lower maximum VDD and I/O voltage than the PIC18(L)F26/27/45/46/47/55/56/57K42. See **Section 44.0 "Electrical Specifications"** for more information.

10.2.4 IDLE MODE

When IDLEN is set (IDLEN = 1), the SLEEP instruction will put the device into Idle mode. In Idle mode, the CPU and memory operations are halted, but the peripheral clocks continue to run. This mode is similar to Doze mode, except that in Idle, both the CPU and PFM are shut off.

Note:	If $\overline{\text{CLKOUTEN}}$ is enabled ($\overline{\text{CLKOUTEN}}$ = 0, Configuration Word 1H), the output will continue operating while in Idle.

10.2.4.1 Idle and Interrupts

Idle mode ends when an interrupt occurs (even if GIE = 0), but IDLEN is not changed. The device can re-enter Idle by executing the SLEEP instruction.

If Recover-On-Interrupt is enabled (ROI = 1), the interrupt that brings the device out of Idle also restores full-speed CPU execution when Doze is also enabled.

10.2.4.2 Idle and WWDT

When in Idle, the WWDT Reset is blocked and will instead wake the device. The WWDT wake-up is not an interrupt, therefore ROI does not apply.

Note:	The WDT can bring the device out of Idle, in the same way it brings the device out of Sleep. The DOZEN bit is not affected.

10.3 Peripheral Operation in Power Saving Modes

All selected clock sources and the peripherals running off them are active in both Idle and Doze mode. Only in Sleep mode, both the FOSC and FOSC/4 clocks are unavailable. All the other clock sources are active, if enabled manually or through peripheral clock selection before the part enters Sleep.

10.4 Register Definitions: Voltage Regulator Control

REGISTER 10-1: VREGCON: VOLTAGE REGULATOR CONTROL REGISTER(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-1/1
—	—	—	—	—	—	VREGPM	Reserved
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 7-2 **Unimplemented:** Read as ‘0’

bit 1 **VREGPM:** Voltage Regulator Power Mode Selection bit
- 1 = Low-Power Sleep mode enabled in Sleep(2)
  Draws lowest current in Sleep, slower wake-up
- 0 = Normal Power mode enabled in Sleep(2)
  Draws higher current in Sleep, faster wake-up

bit 0 **Reserved:** Read as ‘1’. Maintain this bit set.

Note 1: Not present in LF parts.
2: See **Section 44.0 “Electrical Specifications”**.

REGISTER 10-2: CPUDOZE: DOZE AND IDLE REGISTER

R/W-0/0	R/W/HC/HS-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
IDLEN	DOZEN	ROI	DOE	—	DOZE[2:0]		
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	HC = Bit is cleared by hardware; HS = Bit is set by hardware

bit 7 **IDLEN:** Idle Enable bit
1 = A SLEEP instruction places the device into Idle mode
0 = A SLEEP instruction places the device into Sleep mode

bit 6 **DOZEN:** Doze Enable bit(1)
1 = Places the device into Doze mode
0 = Places the device into Normal mode

bit 5 **ROI:** Recover-On-Interrupt bit(1)
1 = Entering the Interrupt Service Routine (ISR) makes DOZEN = 0
0 = Entering the Interrupt Service Routine (ISR) does not change DOZEN

bit 4 **DOE:** Doze-On-Exit bit(1)
1 = Exiting the Interrupt Service Routine (ISR) makes DOZEN = 1
0 = Exiting the Interrupt Service Routine (ISR) does not change DOZEN

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **DOZE[2:0]:** Ratio of CPU Instruction Cycles to Peripheral Instruction Cycles
111 =1:256
110 =1:128
101 =1:64
100 =1:32
011 =1:16
010 =1:8
001 =1:4
000 =1:2

Note 1: Refer Table 10-1 for more details.

TABLE 10-2: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
VREGCON(1)	—	—	—	—	—	—	VREGPM	Reserved	176
CPUDOZE	IDLEN	DOZEN	ROI	DOE	—	DOZE[2:0]			177

Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used in Power-Down mode.
Note 1: Not present in LF parts.

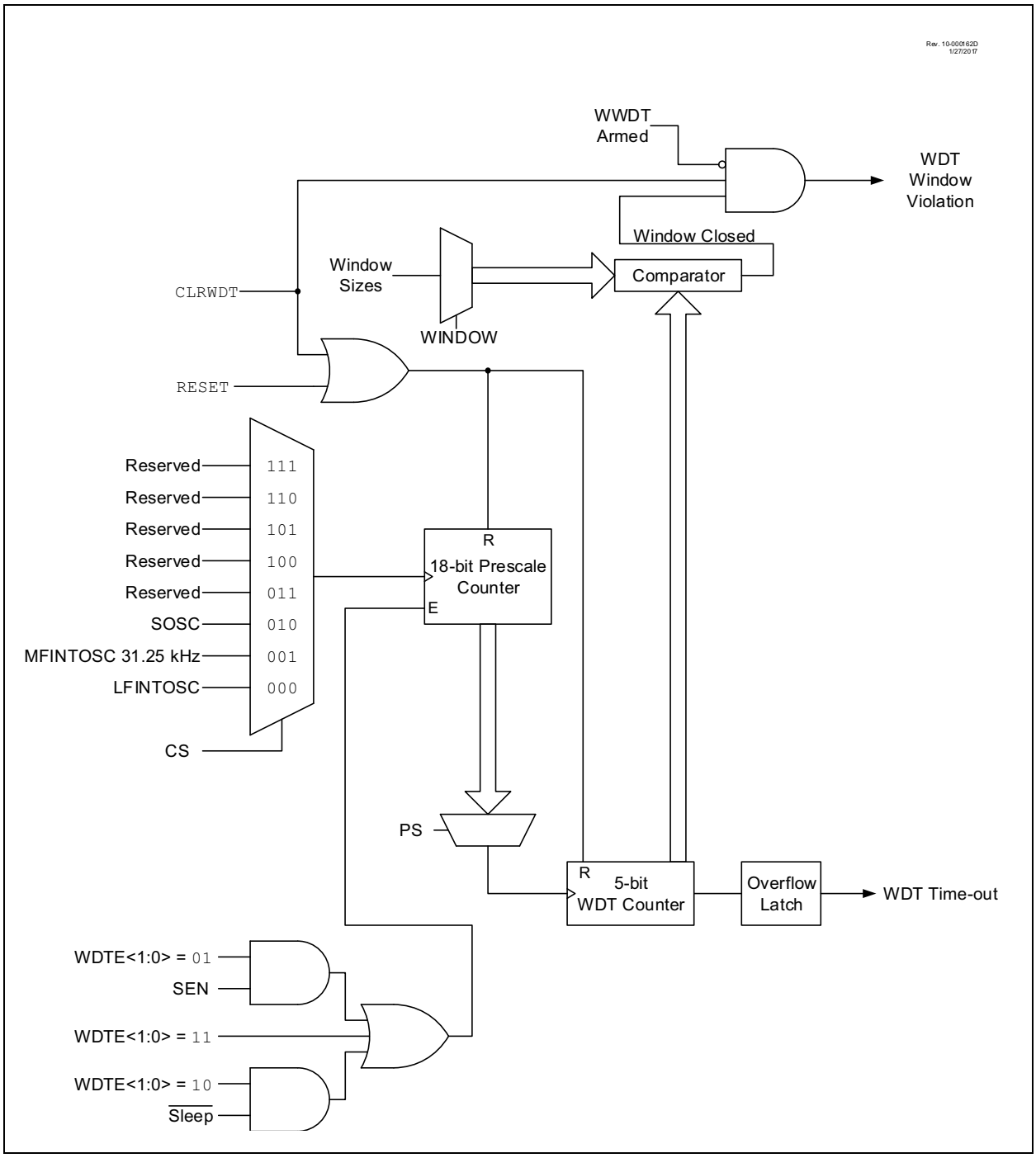
11.0 WINDOWED WATCHDOG TIMER (WWDT)

The Watchdog Timer (WDT) is a system timer that generates a Reset if the firmware does not issue a `CLRWDT` instruction within the time-out period. The Watchdog Timer is typically used to recover the system from unexpected events. The Windowed Watchdog Timer (WWDT) differs in that `CLRWDT` instructions are only accepted when they are performed within a specific window during the time-out period.

The WWDT has the following features:

- Selectable clock source
- Multiple operating modes
  - WWDT is always On
  - WWDT is off when in Sleep
  - WWDT is controlled by software
  - WWDT is always Off
- Configurable time-out period is from 1 ms to 256s (nominal)
- Configurable window size from 12.5% to 100% of the time-out period
- Multiple Reset conditions

FIGURE 11-1: WINDOWED WATCHDOG TIMER BLOCK DIAGRAM



11.1 Independent Clock Source

The WWDT can derive its time base from either the 31 kHz LFINTOSC or 31.25 kHz MFINTOSC internal oscillators, depending on the value of WDTE[1:0] Configuration bits.

If WDTE = `0b1x`, then the clock source will be enabled depending on the WDTCCS[2:0] Configuration bits.

If WDTE = `0b01`, the SEN bit should be set by software to enable WWDT, and the clock source is enabled by the CS bits in the WDTCON1 register.

Time intervals in this chapter are based on a minimum nominal interval of 1 ms. See **Section 44.0 "Electrical Specifications"** for LFINTOSC and MFINTOSC tolerances.

11.2 WWDT Operating Modes

The Windowed Watchdog Timer module has four operating modes controlled by the WDTE[1:0] bits in Configuration Words. See Table 11-1.

11.2.1 WWDT IS ALWAYS ON

When the WDTE bits of Configuration Words are set to '`11`', the WWDT is always on.

WWDT protection is active during Sleep.

11.2.2 WWDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to '`10`', the WWDT is on, except in Sleep.

WWDT protection is not active during Sleep.

11.2.3 WWDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to '`01`', the WWDT is controlled by the SEN bit of the WDTCON0 register.

WWDT protection is unchanged by Sleep. See Table 11-1 for more details.

TABLE 11-1: WWDT OPERATING MODES

WDTE[1:0]	SEN	Device Mode	WWDT Mode
11	X	X	Active
10	X	Awake	Active
		Sleep	Disabled
01	1	X	Active
	0	X	Disabled
00	X	X	Disabled

11.3 Time-out Period

If the WDTCPS[4:0] Configuration bits default to `0b11111`, then the PS bits of the WDTCON0 register set the time-out period from 1 ms to 256 seconds (nominal). If any value other than the default value is assigned to WDTCPS[4:0] Configuration bits, then the timer period will be based on the WDTCPS[4:0] bits in the CONFIG3L register. After a Reset, the default time-out period is 2s.

11.4 Watchdog Window

The Windowed Watchdog Timer has an optional Windowed mode that is controlled by the WDTCWS[2:0] Configuration bits and WINDOW[2:0] bits of the WDTCON1 register. In the Windowed mode, the `CLRWDT` instruction must occur within the allowed window of the WDT period. Any `CLRWDT` instruction that occurs outside of this window will trigger a window violation and will cause a WWDT Reset, similar to a WWDT time out. See Figure 11-2 for an example.

The window size is controlled by the WINDOW[2:0] Configuration bits, or the WINDOW[2:0] bits of WDTCON1, if WDTCWS[2:0] = `111`.

The five Most Significant bits of the WDTTMR register are used to determine whether the window is open, as defined by the WINDOW[2:0] bits of the WDTCON1 register.

In the event of a window violation, a Reset will be generated and the $\overline{\text{WDTWV}}$ bit of the PCON0 register will be cleared. This bit is set by a POR or can be set in firmware.

11.5 Clearing the WWDT

The WWDT is cleared when any of the following conditions occur:

- Any Reset
- Valid `CLRWDT` instruction is executed
- Device enters Sleep
- Exit Sleep by Interrupt
- WWDT is disabled
- Oscillator Start-up Timer (OST) is running
- Any write to the WDTCON0 or WDTCON1 registers

11.5.1 `CLRWDT` CONSIDERATIONS (WINDOWED MODE)

When in Windowed mode, the WWDT must be armed before a `CLRWDT` instruction will clear the timer. This is performed by reading the WDTCON0 register. Executing a `CLRWDT` instruction without performing such an arming action will trigger a window violation regardless of whether the window is open or not.

See Table 11-2 for more information.

11.6 Operation During Sleep

When the device enters Sleep, the WWDT is cleared. If the WWDT is enabled during Sleep, the WWDT resumes counting. When the device exits Sleep, the WWDT is cleared again.

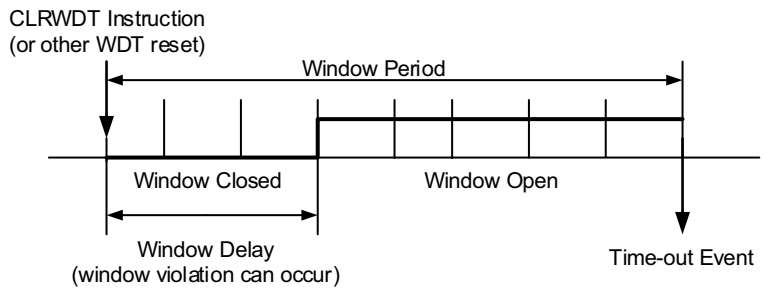
The WWDT remains clear until the Oscillator Start-up Timer (OST) completes, if enabled. See **Section 7.2.1.3 "Oscillator Start-up Timer (OST)"** for more information on the OST.

When a WWDT time-out occurs while the device is in Sleep, no Reset is generated. Instead, the device wakes up and resumes operation. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register are changed to indicate the event. The $\overline{RWDT}$ bit in the PCON0 register can also be used. See **Section 4.0 "Memory Organization"** for more information.

TABLE 11-2: WWDT CLEARING CONDITIONS

Conditions	WWDT
WDTE[1:0] = 00	Cleared
WDTE[1:0] = 01 and SEN = 0	
WDTE[1:0] = 10 and enter Sleep	
CLRWDT Command	
Oscillator Fail Detected	
Exit Sleep + System Clock = SOSC, EXTRC, INTOSC, EXTCLK	
Exit Sleep + System Clock = XT, HS, LP	Cleared until the end of OST
Change INTOSC divider (IRCF bits)	Unaffected

FIGURE 11-2: WINDOW PERIOD AND DELAY



11.7 Register Definitions: Windowed Watchdog Timer Control

REGISTER 11-1: WDTCON0: WATCHDOG TIMER CONTROL REGISTER 0

U-0	U-0	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W-0/0
—	—	PS[4:0]					SEN
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 7-6 **Unimplemented:** Read as ‘0’

bit 5-1 **PS[4:0]:** Watchdog Timer Prescale Select bits(1)

Bit Value = Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)
•
•
•
10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 (2^{23}) (Interval 256s nominal)
10001 = 1:4194304 (2^{22}) (Interval 128s nominal)
10000 = 1:2097152 (2^{21}) (Interval 64s nominal)
01111 = 1:1048576 (2^{20}) (Interval 32s nominal)
01110 = 1:524288 (2^{19}) (Interval 16s nominal)
01101 = 1:262144 (2^{18}) (Interval 8s nominal)
01100 = 1:131072 (2^{17}) (Interval 4s nominal)
01011 = 1:65536 (Interval 2s nominal) (Reset value)
01010 = 1:32768 (Interval 1s nominal)
01001 = 1:16384 (Interval 512 ms nominal)
01000 = 1:8192 (Interval 256 ms nominal)
00111 = 1:4096 (Interval 128 ms nominal)
00110 = 1:2048 (Interval 64 ms nominal)
00101 = 1:1024 (Interval 32 ms nominal)
00100 = 1:512 (Interval 16 ms nominal)
00011 = 1:256 (Interval 8 ms nominal)
00010 = 1:128 (Interval 4 ms nominal)
00001 = 1:64 (Interval 2 ms nominal)
00000 = 1:32 (Interval 1 ms nominal)

bit 0 **SEN:** Software Enable/Disable for Watchdog Timer bit

If WDTE[1:0] = 1x:
This bit is ignored.
If WDTE[1:0] = 01:
1 = WDT is turned on
0 = WDT is turned off
If WDTE[1:0] = 00:
This bit is ignored.

Note 1: Times are approximate. WDT time is based on 31 kHz LFINTOSC.

2: When WDTCPS [4:0] in CONFIG3L = 11111, the Reset value of PS[4:0] is 01011. Otherwise, the Reset value of PS[4:0] is equal to WDTCPS[4:0] in CONFIG3L.

3: When WDTCPS [4:0] in CONFIG3L ≠ 11111, these bits are read-only.

4: When the WWDT is configured to run using the SOSC as a clock source and the device is allowed to undergo a Reset, as triggered by a WDT time-out, the SOSC would also undergo a Reset. That means the SOSC will execute its start-up sequence which requires 1024 SOSC clock counts before it is made available for peripherals to use. So for example, if the WDT is set for a 1 ms time-out and the device is allowed to undergo a WDT Reset, then the actual WDT Reset period will be: WDT_PERIOD = (1/(SOSC_FREQUENCY) * 1024) + 1 ms.

REGISTER 11-2: WDTCON1: WATCHDOG TIMER CONTROL REGISTER 1

U-0	R/W(3)-q/q(1)	R/W(3)-q/q(1)	R/W(3)-q/q(1)	U-0	R/W(4)-q/q(2)	R/W(4)-q/q(2)	R/W(4)-q/q(2)
—	CS[2:0]			—	WINDOW[2:0]		
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 **Unimplemented:** Read as ‘0’

bit 6-4 **CS[2:0]:** Watchdog Timer Clock Select bits

111 = Reserved
•
•
•
011 = Reserved
010 = SOSC
001 = MFINTOSC 31.25 kHz
000 = LFINTOSC 31 kHz

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **WINDOW[2:0]:** Watchdog Timer Window Select bits

WINDOW[2:0]	Window delay Percent of time	Window opening Percent of time
111	N/A	100
110	12.5	87.5
101	25	75
100	37.5	62.5
011	50	50
010	62.5	37.5
001	75	25
000	87.5	12.5

Note 1: If WDTCCS [2:0] in CONFIG3H = 111, the Reset value of CS[2:0] is 000.

2: The Reset value of WINDOW[2:0] is determined by the value of WDTCWS[2:0] in the CONFIG3H register.

3: If WDTCCS[2:0] in CONFIG3H ≠ 111, these bits are read-only.

4: If WDTCWS[2:0] in CONFIG3H ≠ 111, these bits are read-only.

REGISTER 11-3: WDTPSL: WWDT PRESCALE SELECT LOW BYTE REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT[7:0]							
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 7-0 **PSCNT[7:0]:** Prescale Select Low Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT[17:0] includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT[17:0] is intended for debug operations and should not be read during normal operation.

REGISTER 11-4: WDTPSH: WWDT PRESCALE SELECT HIGH BYTE REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT[15:8]							
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 7-0 **PSCNT[15:8]:** Prescale Select High Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT[17:0] includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT[17:0] is intended for debug operations and should not be read during normal operation.

REGISTER 11-5: WDTTMR: WDT TIMER REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
WDTTMR[4:0]					STATE	PSCNT[17:16]	
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 7-3 **WDTTMR[4:0]:** Watchdog Window Value bits

WINDOW	WDT Window State		Open Percent
	Closed	Open	
111	N/A	00000-11111	100
110	00000-00011	00100-11111	87.5
101	00000-00111	01000-11111	75
100	00000-01011	01100-11111	62.5
011	00000-01111	10000-11111	50
010	00000-10011	10100-11111	37.5
001	00000-10111	11000-11111	25
000	00000-11011	11100-11111	12.5

bit 2 **STATE:** WDT Armed Status bit
1 = WDT is armed
0 = WDT is not armed

bit 1-0 **PSCNT[17:16]:** Prescale Select Upper Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT[17:0] includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT[17:0] is intended for debug operations and should not be read during normal operation.

TABLE 11-3: SUMMARY OF REGISTERS ASSOCIATED WITH WINDOWED WATCHDOG TIMER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
WDTCON0	—	—	PS[4:0]					SEN	182
WDTCON1	—	CS[2:0]			—	WINDOW[2:0]			183
WDTPSL	PSCNT[7:0]								184
WDTPSH	PSCNT[15:8]								184
WDTTMR	WDTTMR[4:0]					STATE	PSCNT[17:16]		185

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by Windowed Watchdog Timer.

12.0 8x8 HARDWARE MULTIPLIER

12.1 Introduction

All PIC18 devices include an 8x8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 12-1.

12.2 Operation

Example 12-1 shows the instruction sequence for an 8x8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 12-2 shows the sequence to do an 8x8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 12-1: 8x8 UNSIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W    ;
MULWF   ARG2       ; ARG1 * ARG2 ->
                   ; PRODH:PRODL
```

EXAMPLE 12-2: 8x8 SIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W
MULWF   ARG2       ; ARG1 * ARG2 ->
                   ; PRODH:PRODL
BTFSC   ARG2, SB   ; Test Sign Bit
SUBWF   PRODH, F   ; PRODH = PRODH
                   ;          - ARG1
MOVF    ARG2, W
BTFSC   ARG1, SB   ; Test Sign Bit
SUBWF   PRODH, F   ; PRODH = PRODH
                   ;          - ARG2
```

TABLE 12-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time			
				@ 64 MHz	@ 40 MHz	@ 10 MHz	@ 4 MHz
8x8 unsigned	Without hardware multiply	13	69	4.3 µs	6.9 µs	27.6 µs	69 µs
	Hardware multiply	1	1	62.5 ns	100 ns	400 ns	1 µs
8x8 signed	Without hardware multiply	33	91	5.7 µs	9.1 µs	36.4 µs	91 µs
	Hardware multiply	6	6	375 ns	600 ns	2.4 µs	6 µs
16x16 unsigned	Without hardware multiply	21	242	15.1 µs	24.2 µs	96.8 µs	242 µs
	Hardware multiply	28	28	1.8 µs	2.8 µs	11.2 µs	28 µs
16x16 signed	Without hardware multiply	52	254	15.9 µs	25.4 µs	102.6 µs	254 µs
	Hardware multiply	35	40	2.5 µs	4.0 µs	16.0 µs	40 µs

© 2017-2020 Microchip Technology Inc.

Example 12-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 12-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES[3:0]).

EQUATION 12-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

EXAMPLE 12-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
```

Example 12-4 shows the sequence to do a 16 x 16 signed multiply. Equation 12-2 shows the algorithm used. The 32-bit result is stored in four registers (RES[3:0]). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 12-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H[7] \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H[7] \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

EXAMPLE 12-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1       ; no, check ARG1
    MOVF    ARG1L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG1H, W        ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG1H, 7        ; ARG1H:ARG1L neg?
    BRA     CONT_CODE       ; no, done
    MOVF    ARG2L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG2H, W        ;
    SUBWFB  RES3
;
CONT_CODE
    :
```

13.0 NONVOLATILE MEMORY (NVM) CONTROL

Nonvolatile Memory (NVM) is separated into two types: Program Flash Memory (PFM) and Data EEPROM Memory.

PFM, Data EEPROM, User IDs and Configuration bits can all be accessed using the REG[1:0] bits of the NVMCON1 register.

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump rated to operate over the operating voltage range of the device.

NVM can be protected in two ways, by either code protection or write protection. Code protection ($\overline{CP}$ and $\overline{CPD}$ bits in Configuration Word 5L) disables access, reading and writing to both PFM and Data EEPROM Memory via external device programmers. Code protection does not affect the self-write and erase functionality. Code protection can only be reset by a device programmer performing a Bulk Erase to the device, clearing all nonvolatile memory, Configuration bits and User IDs.

Write protection prohibits self-write and erase to a portion or all of the PFM, as defined by the WRT bits of Configuration Word 4H. Write protection does not affect a device programmer's ability to read, write or erase the device.

TABLE 13-1: NVM ORGANIZATION AND ACCESS INFORMATION

Memory	PC[20:0] ICSP™ Addr[21:0] TBLPTR[21:0] NVMADDR[9:0]	Execution	User Access		
		CPU Execution	REG	TABLAT	NVMDAT
Program Flash Memory (PFM)	00 0000h • • • 01 FFFFh	Read	10	Read/ Write(1)	—(3)
User IDs(2)	20 0000h • • • 20 000Fh	No Access	x1	Read/ Write	—(3)
Reserved	20 0010h 2F FFFFh	No Access	—(3)		
Configuration	30 0000h • • • 30 0009h	No Access	x1	Read/ Write(1)	—(3)
Reserved	30 000Ah 30 FFFFh	No Access	—(3)		
User Data Memory (Data EEPROM)	31 0000h • • • 31 03FFh	No Access	00	—(3)	Read/ Write(1)
Reserved	31 0400h 3E FFFFh	No Access	—(3)		
Device Information Area (DIA)	3F 0000h • • • 3F 003Fh	No Access	x1	Read	—(3)
Reserved	3F 0040h 3F FF09h	No Access	—(3)		
Device Configuration Information (DCI)	3F FF00h • • • 3F FF09h	No Access	x1	Read	—(3)
Reserved	3F FF0Ah 3F FFFBh	No Access	—(3)		
Revision ID/ Device ID	3F FFFCh • • • 3F FFFFh	No Access	x1	Read	—(3)

Note 1: Subject to Memory Write Protection settings.
2: User IDs are eight words ONLY. There is no code protection, table read protection or write protection implemented for this region.
3: Reads as '0', writes clear the WR bit and WRERR bit is set.

13.1 Program Flash Memory

The Program Flash Memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed one byte at a time. A write to program memory or program memory erase is executed on blocks of *n* bytes at a time. Refer to Table 5-4 for write and erase block sizes. A Bulk Erase operation cannot be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a `NOP`.

It is important to understand the PFM memory structure for erase and programming operations. Program memory word size is 16 bits wide. PFM is arranged in rows. A row is the minimum size that can be erased by user software. Refer to Table 5-4 for the row sizes for these devices.

After a row has been erased, all or a portion of this row can be programmed. Data to be written into the program memory row is written to 8-bit wide data write latches by means of 6 address lines. These latches are not directly accessible, but may be loaded via sequential writes to the TABLAT register.

Note:	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. Then, the new data and retained data can be written into the write latches to reprogram the row of PFM. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations.

13.1.1 TABLE READS AND TABLE WRITES

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (`TBLRD`)
- Table Write (`TBLWT`)

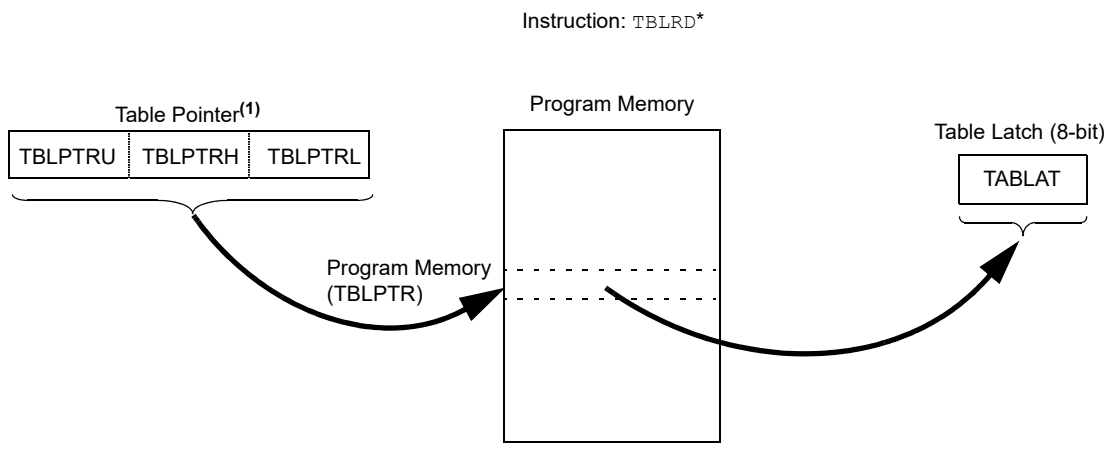
The program memory space is 16 bits wide, while the data RAM space is eight bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. Figure 13-1 shows the operation of a table read.

The table write operation stores one byte of data from the TABLAT register into a write block holding register. The procedure to write the contents of the holding registers into program memory is detailed in **Section 13.1.6 "Writing to Program Flash Memory"**. Figure 13-2 shows the operation of a table write with program memory.

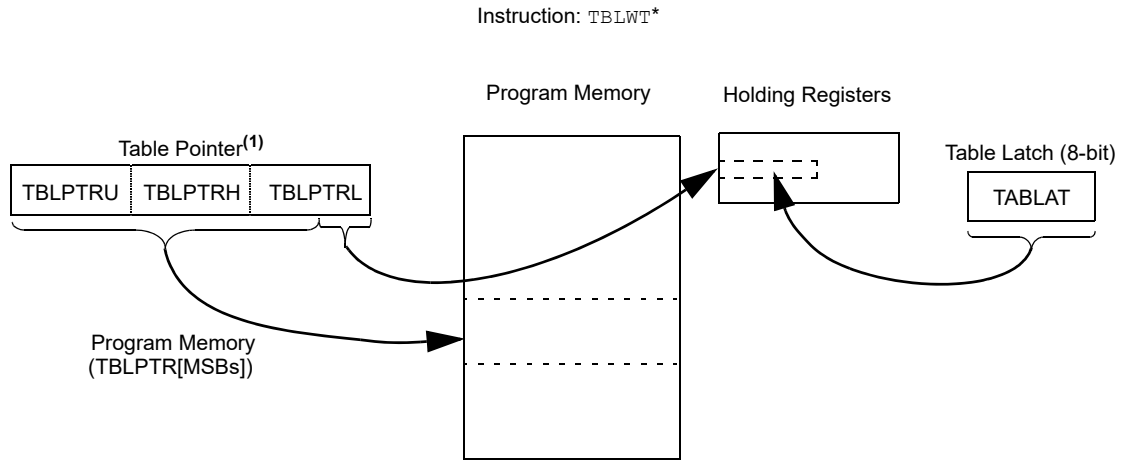
Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word aligned.

FIGURE 13-1: TABLE READ OPERATION



Note 1: Table Pointer register points to a byte in program memory.

FIGURE 13-2: TABLE WRITE OPERATION



Note 1: During table writes the Table Pointer does not point directly to program memory. The LSBs of TBLPRTL actually point to an address within the write block holding registers. The MSBs of the Table Pointer determine where the write block will eventually be written. The process for writing the holding registers to the program memory array is discussed in **Section 13.1.6 "Writing to Program Flash Memory"**.

13.1.2 CONTROL REGISTERS

Several control registers are used in conjunction with the TBLRD and TBLWT instructions. These include the following registers:

- NVMCON1 register
- NVMCON2 register
- TABLAT register
- TBLPTR registers

13.1.2.1 NVMCON1 and NVMCON2 Registers

The NVMCON1 register (Register 13-1) is the control register for memory accesses. The NVMCON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading NVMCON2 will read all '0's.

The REG[1:0] control bits determine if the access will be to Data EEPROM Memory locations. PFM locations or User IDs, Configuration bits, Rev ID and Device ID. When REG[1:0] = 00, any subsequent operations will operate on the Data EEPROM Memory. When REG[1:0] = 10, any subsequent operations will operate on the program memory. When REG[1:0] = x1, any subsequent operations will operate on the Configuration bits, User IDs, Rev ID and Device ID.

The FREE bit allows the program memory erase operation. When the FREE bit is set, an erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled. This bit is applicable only to the PFM and not to data EEPROM.

When set, the WREN bit will allow a program/erase operation. The WREN bit is cleared on power-up.

The WRERR bit is set by hardware when the WR bit is set and is cleared when the internal programming timer expires and the write operation is successfully complete.

The WR control bit initiates erase/write cycle operation when the REG[1:0] bits point to the Data EEPROM Memory location, and it initiates a write operation when the REG[1:0] bits point to the PFM location. The WR bit cannot be cleared by firmware; it can only be set by firmware. Then the WR bit is cleared by hardware at the completion of the write operation.

The NVMIF Interrupt Flag bit is set when the write is complete. The NVMIF flag stays set until cleared by firmware.

13.1.2.2 TABLAT – Table Latch Register

The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The Table Latch register is used to hold 8-bit data during data transfers between program memory and data RAM.

13.1.2.3 TBLPTR – Table Pointer Register

The Table Pointer (TBLPTR) register addresses a byte within the program memory. The TBLPTR is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers join to form a 22-bit wide pointer. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. The 22nd bit allows access to the Device ID, the User ID and the Configuration bits.

The Table Pointer register, TBLPTR, is used by the TBLRD and TBLWT instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations on the TBLPTR affect only the low-order 21 bits.

13.1.2.4 Table Pointer Boundaries

TBLPTR is used in reads, writes and erases of the Program Flash Memory.

When a TBLRD is executed, all 22 bits of the TBLPTR determine which byte is read from program memory directly into the TABLAT register.

When a TBLWT is executed the byte in the TABLAT register is written, not to memory but, to a holding register in preparation for a program memory write. The holding registers constitute a write block which varies depending on the device (see Table 5-4).The 3, 4, or 5 LSbs of the TBLPTRL register determine which specific address within the holding register block is written to. The MSBs of the Table Pointer have no effect during TBLWT operations.

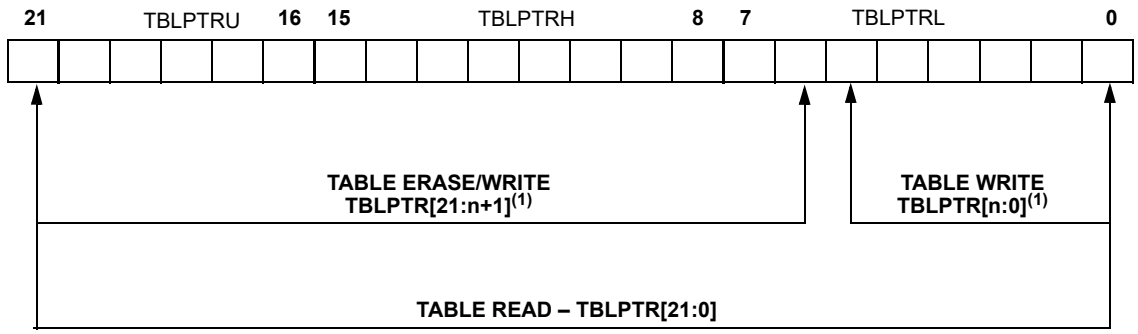
When a program memory write is executed the entire holding register block is written to the memory at the address determined by the MSbs of the TBLPTR. The 3, 4, or 5 LSBs are ignored during memory writes. For more detail, see **Section 13.1.6 "Writing to Program Flash Memory"**.

Figure 13-3 describes the relevant boundaries of TBLPTR based on Program Flash Memory operations.

TABLE 13-2: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

Example	Operation on Table Pointer
`TBLRD*` `TBLWT*`	TBLPTR is not modified
`TBLRD*+` `TBLWT*+`	TBLPTR is incremented after the read/write
`TBLRD*-` `TBLWT*-`	TBLPTR is decremented after the read/write
`TBLRD+*` `TBLWT+*`	TBLPTR is incremented before the read/write

FIGURE 13-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



Note 1: Refer to Table 5-4 for the row size values.

13.1.3 READING THE PROGRAM FLASH MEMORY

The TBLRD instruction retrieves data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

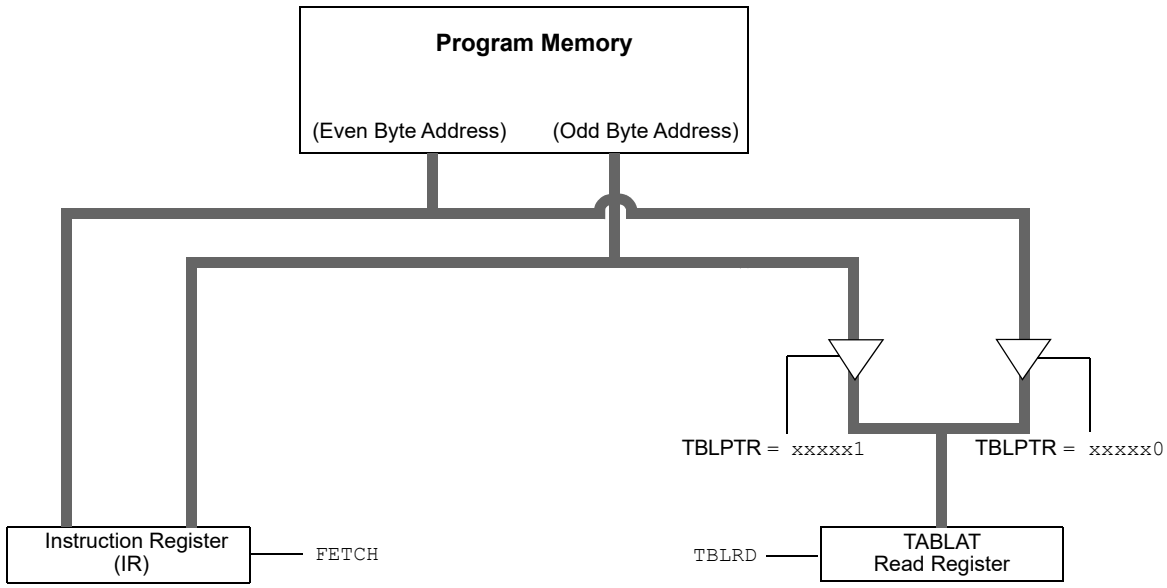
TBLPTR points to a byte address in program space. Executing TBLRD places the byte pointed to into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation.

The CPU operation is suspended during the read, and it resumes immediately after. From the user point of view, TABLAT is valid in the next instruction cycle.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word.

Figure 13-4 shows the interface between the internal program memory and the TABLAT.

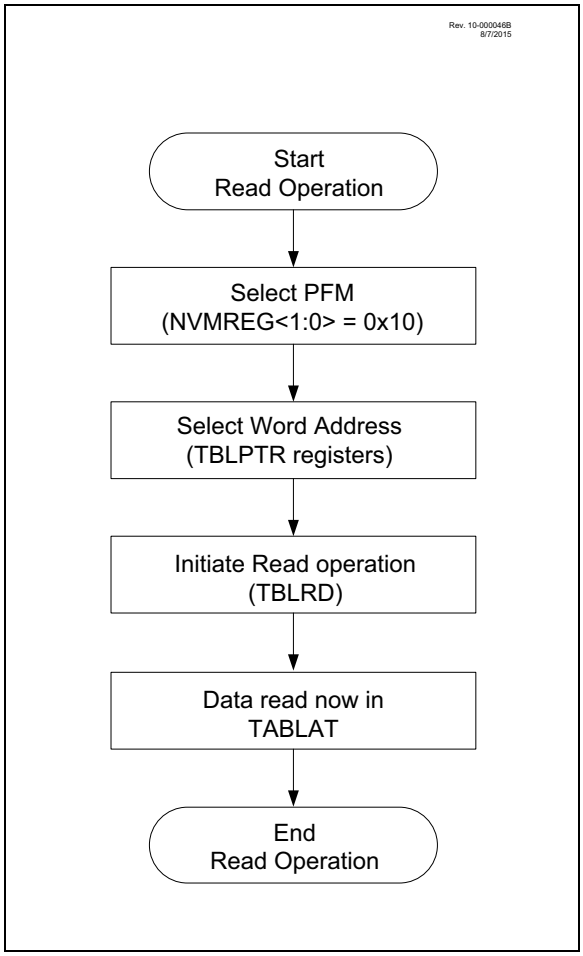
FIGURE 13-4: READS FROM PROGRAM FLASH MEMORY



EXAMPLE 13-1: READING A PROGRAM FLASH MEMORY WORD

```
            BCF       NVMCON1, REG0          ; point to Program Flash Memory
            BSF       NVMCON1, REG1          ; access Program Flash Memory
            MOVLW     CODE_ADDR_UPPER        ; Load TBLPTR with the base
            MOVWF     TBLPTRU                ; address of the word
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
READ_WORD
            TBLRD*+                          ; read into TABLAT and increment
            MOVF      TABLAT, W              ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                          ; read into TABLAT and increment
            MOVFW     TABLAT, W              ; get data
            MOVF      WORD_ODD
```

FIGURE 13-5: PROGRAM FLASH MEMORY READ FLOWCHART



13.1.4 NVM UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the NVM from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- PFM Row Erase
- Write of PFM write latches to PFM memory
- Write of PFM write latches to User IDs
- Write to Data EEPROM Memory
- Write to Configuration Words

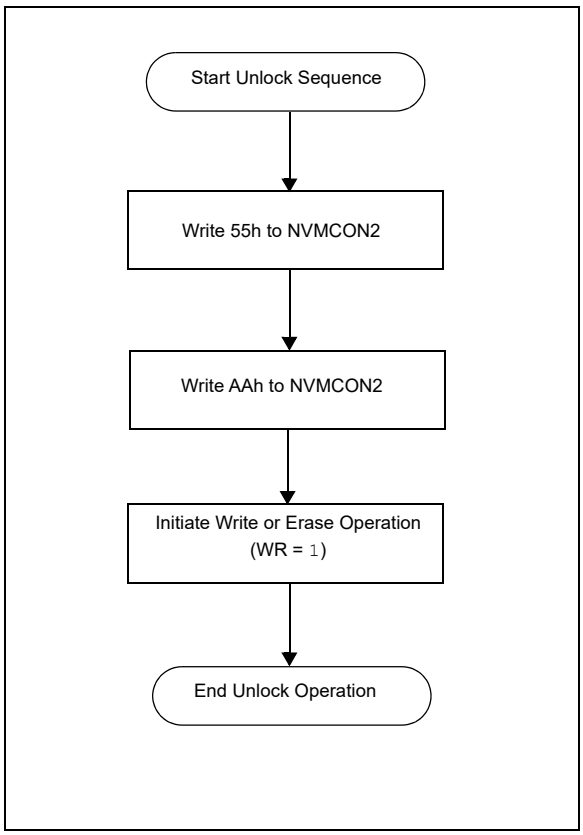
The unlock sequence consists of the following steps and must be completed in order:

- Write 55h to NVMCON2
- Write AAh to NMVCON2
- Set the WR bit of NVMCON1

Once the WR bit is set, the processor will stall internal operations until the operation is complete and then resume with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

FIGURE 13-6: NVM UNLOCK SEQUENCE FLOWCHART



EXAMPLE 13-2: NVM UNLOCK SEQUENCE

```
BCF             INTCON0,GIE          ; Recommended so sequence is not interrupted
BANKSEL         NVMCON1
BSF             NVMCON1,WREN         ; Enable write/erase
MOVLW           55h                  ; Load 55h

MOVWF           NVMCON2              ; Step 1: Load 55h into NVMCON2
MOVLW           AAh                  ; Step 2: Load W with AAh
MOVWF           NVMCON2              ; Step 3: Load AAh into NVMCON2
BSF             NVMCON1,WR           ; Step 4: Set WR bit to begin write/erase

BSF             INTCON0,GIE          ; Re-enable interrupts
```

Note 1: Sequence begins when NVMCON2 is written; steps 1-4 must occur in the cycle-accurate order shown. If the timing of the steps 1 to 4 is corrupted by an interrupt or a debugger Halt, the action will not take place.

2: Opcodes shown are illustrative; any instruction that has the indicated effect may be used.

13.1.5 ERASING PROGRAM FLASH MEMORY

The minimum erase block is 64 words (refer to Table 5-4). Only through the use of an external programmer, or through ICSP™ control, can larger blocks of program memory be bulk erased. Word erase in the program memory array is not supported.

For example, when initiating an erase sequence from a microcontroller with erase row size of 64 words, a block of 64 words (128 bytes) of program memory is erased. The Most Significant 16 bits of the TBLPTR[21:6] point to the block being erased. The TBLPTR[5:0] bits are ignored.

The NVMCON1 register commands the erase operation. The REG[1:0] bits must be set to point to the Program Flash Memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

The NVM unlock sequence described in **Section 13.1.4 “NVM Unlock Sequence”** should be used to guard against accidental writes. This is sometimes referred to as a long write.

A long write is necessary for erasing program memory. Instruction execution is halted during the long write cycle. The long write is terminated by the internal programming timer.

13.1.5.1 Program Flash Memory Erase Sequence

The sequence of events for erasing a block of internal program memory is:

1. REG bits of the NVMCON1 register point to PFM
2. Set the FREE and WREN bits of the NVMCON1 register
3. Perform the unlock sequence as described in **Section 13.1.4 “NVM Unlock Sequence”**

If the PFM address is write-protected, the WR bit will be cleared and the erase operation will not take place, WRERR is signaled in this scenario.

The operation erases the memory row indicated by masking the LSBs of the current TBLPTR.

While erasing PFM, CPU operation is suspended and it resumes when the operation is complete. Upon completion the WR bit is cleared in hardware, the NVMIF is set and an interrupt will occur if the NVMIE bit is also set.

Write latch data is not affected by erase operations and WREN will remain unchanged.

Note 1: If a write or erase operation is terminated by an unexpected event, WRERR bit will set which user can check to decide whether a rewrite of the location(s) is needed.

2: WRERR is set if WR is written to ‘1’ while TBLPTR points to a write-protected address.

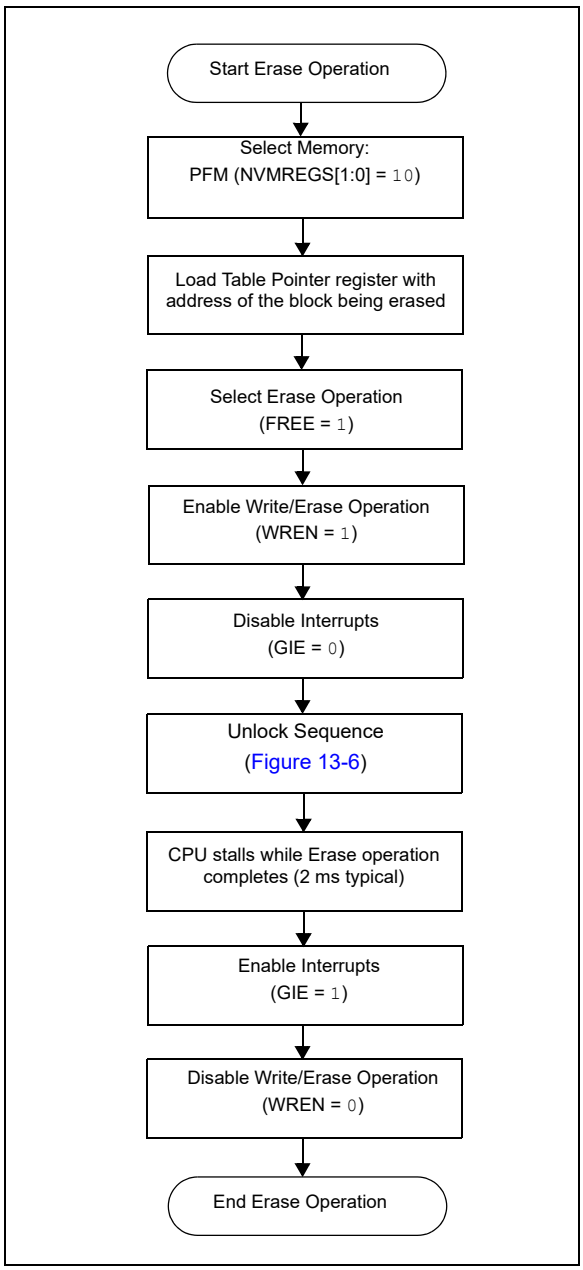
3: WRERR is set if WR is written to ‘1’ while TBLPTR points to an invalid address location (Table 13-1).

EXAMPLE 13-3: ERASING A PROGRAM FLASH MEMORY BLOCK

```
; This sample row erase routine assumes the following:
; 1. A valid address within the erase row is loaded in variables TBLPTR register

           CLRF       NVMCON1            ; Setup PFM Access
           MOVLW      CODE_ADDR_UPPER    ; load TBLPTR with the base
           MOVWF      TBLPTRU            ; address of the memory block
           MOVLW      CODE_ADDR_HIGH
           MOVWF      TBLPTRH
           MOVLW      CODE_ADDR_LOW
           MOVWF      TBLPTRL
ERASE_BLOCK
           BCF        NVMCON1, REG0      ; point to Program Flash Memory
           BSF        NVMCON1, REG1      ; access Program Flash Memory
           BSF        NVMCON1, WREN      ; enable write to memory
           BSF        NVMCON1, FREE      ; enable block Erase operation
           BCF        INTCON0, GIE       ; disable interrupts
           MOVLW      55h                ; next four instructions are required sequence
           MOVWF      NVMCON2            ; write 55h
           MOVLW      AAh
           MOVWF      NVMCON2            ; write AAh
           BSF        NVMCON1, WR        ; start erase (CPU stalls)
           BSF        INTCON0, GIE       ; re-enable interrupts
```

FIGURE 13-7: PFM ROW ERASE FLOWCHART



13.1.6 WRITING TO PROGRAM FLASH MEMORY

The programming write block size is described in Table 5-4. Word or byte programming is not supported. Table writes are used internally to load the holding registers needed to program the memory. There are only as many holding registers as there are bytes in a write block. Refer to Table 5-4 for write latch size.

Since the table latch (TABLAT) is only a single byte, the TBLWT instruction needs to be executed multiple times for each programming operation. The write protection state is ignored for this operation. All of the table write operations will essentially be short writes because only the holding registers are written. NVMIF is not affected while writing to the holding registers.

After all the holding registers have been written, the programming operation of that block of memory is started by configuring the NVMCON1 register for a program memory write and performing the long write sequence.

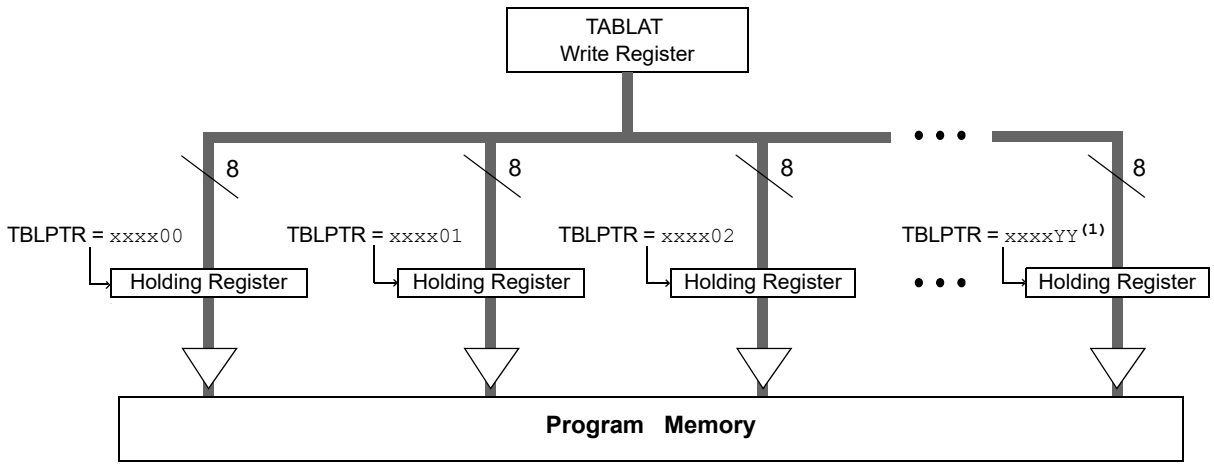
If the PFM address in the TBLPTR is write-protected or if TBLPTR points to an invalid location, the WR bit is cleared without any effect and the WRERR is signaled.

The long write is necessary for programming the program memory. CPU operation is suspended during a long write cycle and resumes when the operation is complete. The long write operation completes in one instruction cycle. When complete, WR is cleared in hardware and NVMIF is set and an interrupt will occur if NVMIE is also set. The latched data is reset to all '1s'. WREN is not changed.

The internal programming timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

> **Note:** The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all holding registers before executing a long write operation.

FIGURE 13-8: TABLE WRITES TO PROGRAM FLASH MEMORY



Note 1: Refer to Table 5-4 for number of holding registers (e.g., YY = 3F for 64 holding registers).

13.1.6.1 Program Flash Memory Write Sequence

The sequence of events for programming an internal program memory location should be:

1. Read appropriate number of bytes into RAM. Refer to Table 5-4 for Write latch size.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the block erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the n-byte block into the holding registers with auto-increment. Refer to Table 5-4 for Write latch size.
7. Set REG[1:0] bits to point to program memory.
8. Clear FREE bit and set WREN bit in NVMCON1 register.
9. Disable interrupts.
10. Execute the unlock sequence (see **Section 13.1.4 “NVM Unlock Sequence”**).
11. WR bit is set in NVMCON1 register.
12. The CPU will stall for the duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Verify the memory (table read).

This procedure will require about 6 ms to update each write block of memory. An example of the required code is given in Example 13-4.

Note:	Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the bytes in the holding registers.

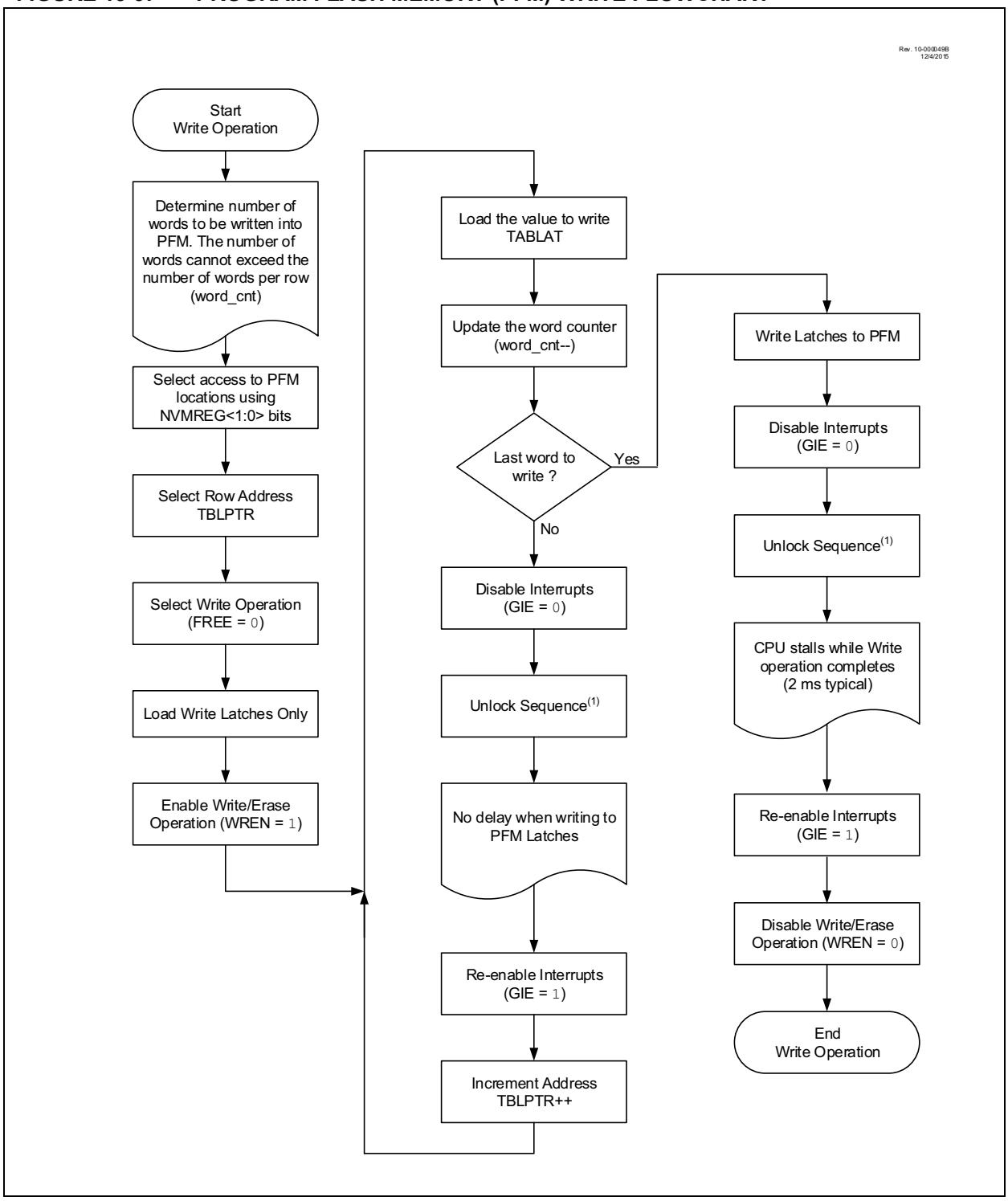
EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY

```
            MOVLW     D'64'                 ; number of bytes in erase block
            MOVWF     COUNTER
            MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
            MOVWF     FSR0H
            MOVLW     BUFFER_ADDR_LOW
            MOVWF     FSR0L
            MOVLW     CODE_ADDR_UPPER       ; Load TBLPTR with the base
            MOVWF     TBLPTRU               ; address of the memory block
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
READ_BLOCK
            TBLRD*+                         ; read into TABLAT, and inc
            MOVF      TABLAT, W             ; get data
            MOVWF     POSTINC0              ; store data
            DECFSZ    COUNTER               ; done?
            BRA       READ_BLOCK            ; repeat
MODIFY_WORD
            MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
            MOVWF     FSR0H
            MOVLW     BUFFER_ADDR_LOW
            MOVWF     FSR0L
            MOVLW     NEW_DATA_LOW          ; update buffer word
            MOVWF     POSTINC0
            MOVLW     NEW_DATA_HIGH
            MOVWF     INDF0
ERASE_BLOCK
            MOVLW     CODE_ADDR_UPPER       ; load TBLPTR with the base
            MOVWF     TBLPTRU               ; address of the memory block
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
            BCF       NVMCON1, REG0         ; point to Program Flash Memory
            BSF       NVMCON1, REG1         ; point to Program Flash Memory
            BSF       NVMCON1, WREN         ; enable write to memory
            BSF       NVMCON1, FREE         ; enable Erase operation
            BCF       INTCON0, GIE          ; disable interrupts
            MOVLW     55h                   ; next four instructions are required sequence
            MOVWF     NVMCON2               ; write 55h
            MOVLW     AAh
            MOVWF     NVMCON2               ; write 0AAh
            BSF       NVMCON1, WR           ; start erase (CPU stall)
            BSF       INTCON0, GIE          ; re-enable interrupts
            TBLRD*-                         ; dummy read decrement
            MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
            MOVWF     FSR0H
            MOVLW     BUFFER_ADDR_LOW
            MOVWF     FSR0L
WRITE_BUFFER_BACK
            MOVLW     BlockSize             ; number of bytes in holding register
            MOVWF     COUNTER
            MOVLW     D'64'/BlockSize       ; number of write blocks in 64 bytes
            MOVWF     COUNTER2
```

EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY (CONTINUED)

```
WRITE_BYTE_TO_HREGS
             MOVF      POSTINC0, W              ; get low byte of buffer data
             MOVWF     TABLAT                   ; present data to table latch
             TBLWT+*                            ; write data, perform a short write
                                                ; to internal TBLWT holding register.
             DECFSZ    COUNTER                  ; loop until holding registers are full
             BRA       WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
             BCF       NVMCON1, REG0            ; point to Program Flash Memory
             BSF       NVMCON1, REG1            ; point to Program Flash Memory
             BSF       NVMCON1, WREN            ; enable write to memory
             BCF       NVMCON1, FREE            ; enable write to memory
             BCF       INTCON0, GIE             ; disable interrupts
             MOVLW     55h                      ; next four instructions are required sequence
             MOVWF     NVMCON2                  ; write 55h
             MOVLW     0AAh
             MOVWF     NVMCON2                  ; write 0AAh
             BSF       NVMCON1, WR              ; start program (CPU stall)
             DCFSZ     COUNTER2                 ; repeat for remaining write blocks
             BRA       WRITE_BYTE_TO_HREGS
             BSF       INTCON0, GIE             ; re-enable interrupts
             BCF       NVMCON1, WREN            ; disable write to memory
```

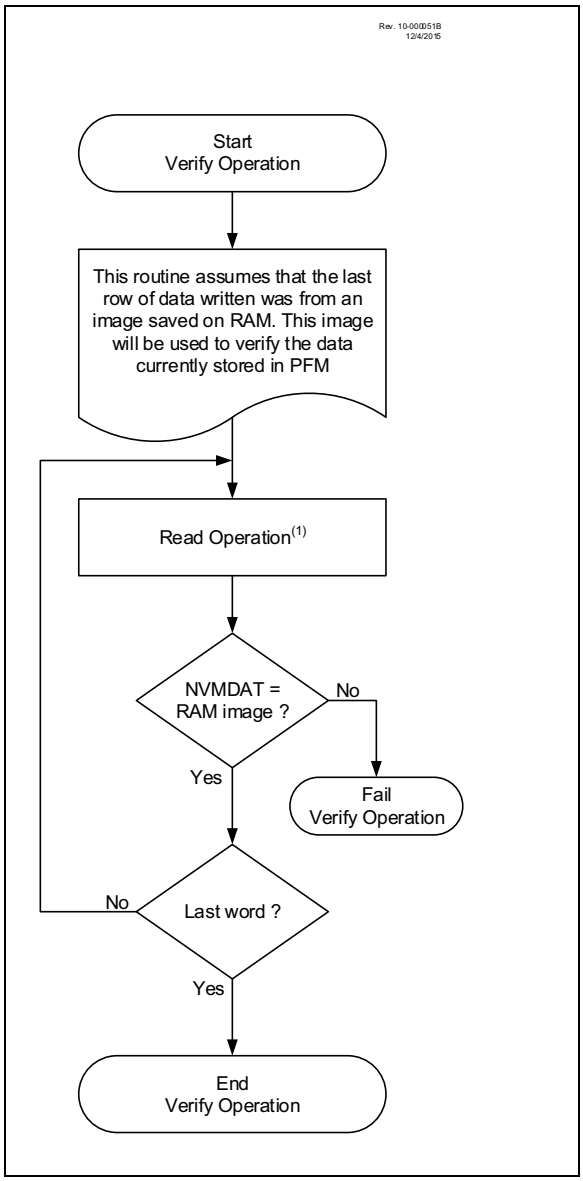
FIGURE 13-9: PROGRAM FLASH MEMORY (PFM) WRITE FLOWCHART



13.1.6.2 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit. Since program memory is stored as a full page, the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

FIGURE 13-10: PROGRAM FLASH MEMORY VERIFY FLOWCHART



13.1.6.3 Unexpected Termination of Write Operation

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. If the write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation, the WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.

13.1.6.4 Protection Against Spurious Writes

A write sequence is valid only when both the following conditions are met, this prevents spurious writes which might lead to data corruption.

1. The WR bit is gated through the WREN bit. It is suggested to have the WREN bit cleared at all times except during memory writes. This prevents memory writes if the WR bit gets set accidentally.
2. The NVM unlock sequence must be performed each time before a write operation.

13.2 Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access

When REG[1:0] = 0b01 or 0b11 in the NVMCON1 register, the Device Information Area, the Device Configuration Area, the User IDs, Device ID/ Revision ID and Configuration Words can be accessed. Different access may exist for reads and writes (see Table 13-1).

13.2.1 Reading Access

The user can read from these blocks by setting the REG bits to 0b01 or 0b11. The user needs to load the address into the TBLPTR registers. Executing a TBLRD after that moves the byte pointed to the TABLAT register. The CPU operation is suspended during the read and resumes after. When read access is initiated on an address outside the parameters listed in Table 13-1, the TABLAT register is cleared, reading back ‘0’s.

13.2.2 Writing Access

The WREN bit in NVMCON1 must be set to enable writes. This prevents accidental writes to the CONFIG words due to errant (unexpected) code execution. The WREN bit should be kept clear at all times, except when updating the CONFIG words. The WREN bit is not cleared by hardware. The WR bit will be inhibited from being set unless the WREN bit is set.

The user needs to load the TBLPTR and TABLAT register with the address and data byte respectively before executing the write command. An unlock sequence needs to be followed for writing to the USER IDs/ DEVICE IDs/CONFIG words (Section 13.1.4, NVM Unlock Sequence). If WRTC = 0 or if TBLPTR points an invalid address location (see Table 13-1), WR bit is cleared without any effect and WRERR is set.

A single CONFIG word byte is written at once and the operation includes an implicit erase cycle for that byte (it is not necessary to set FREE). CPU execution is stalled and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The new CONFIG value takes effect when the CPU resumes operation.

TABLE 13-3: DIA, DCI, USER ID, DEV/REV ID AND CONFIGURATION WORD ACCESS (REG[1:0] = x1)

Address	Function	Read Access	Write Access
20 0000h-20 000Fh	User IDs	Yes	Yes
30 0000h-30 0009h	Configuration Words	Yes	Yes
3F 0000h-3F 003Fh	DIA	Yes	No
3F FF00h-3F FF09h	DCI	Yes	No
3F FFFCh-3F FFFFh	Revision ID/Device ID	Yes	No

13.3 Data EEPROM Memory

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, which is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Five SFRs are used to read and write to the data EEPROM as well as the program memory. They are:

- NVMCON1
- NVMCON2
- NVMDAT
- NVMADRL
- NVMADRH(1)

Note 1: NVMADRH register is not implemented on PIC18(L)F45/55K42.

The data EEPROM allows byte read and write. When interfacing to the data memory block, NVMDAT holds the 8-bit data for read/write and the NVMADRH:NVMADRL register pair holds the address of the EEPROM location being accessed.

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an internal programming timer; it will vary with voltage and temperature as well as from chip-to-chip. Refer to the Data EEPROM Memory parameters in **Section 44.0 "Electrical Specifications"** for limits.

13.3.1 NVMADRL AND NVMADRH REGISTERS

The NVMADRH:NVMADRL registers are used to address the data EEPROM for read and write operations.

13.3.2 NVMCON1 AND NVMCON2 REGISTERS

Access to the data EEPROM is controlled by two registers: NVMCON1 and NVMCON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The NVMCON1 register (Register 13-1) is the control register for data and program memory access. Control bits REG[1:0] determine if the access will be to program, Data EEPROM Memory or the User IDs, Configuration bits, Revision ID and Device ID.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

The WR control bit initiates write operations. The bit can be set but not cleared by software. It is cleared only by hardware at the completion of the write operation.

The NVMIF Interrupt Flag bit of the PIR0 register is set when the write is complete. It must be cleared by software.

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

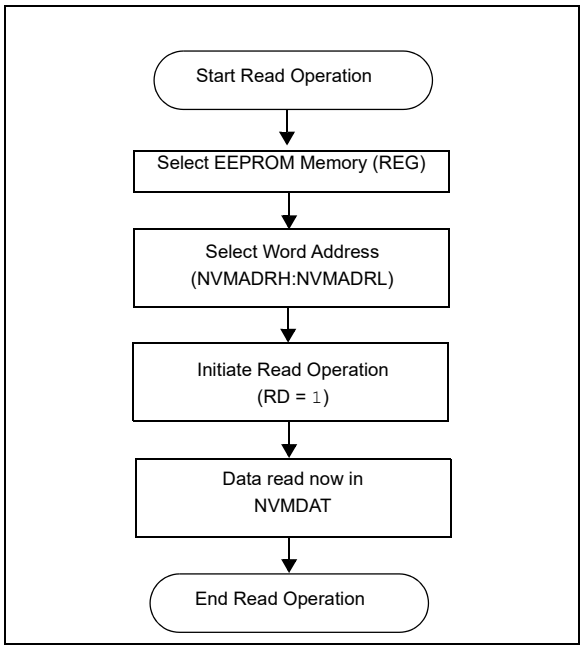
The RD bit cannot be set when accessing program memory (REG[1:0] = 0x10). Program memory is read using table read instructions. See **Section 13.1.1 "Table Reads and Table Writes"** regarding table reads.

13.3.3 READING THE DATA EEPROM MEMORY

To read a data memory location, the user must write the address to the NVMADRL and NVMADRH register pair, clear REG[1:0] control bit in NVMCON1 register to access Data EEPROM locations and then set control bit, RD. The data is available on the very next instruction cycle; therefore, the NVMDAT register can be read by the next instruction. NVMDAT will hold this value until another read operation, or until it is written to by the user (during a write operation).

The basic process is shown in Example 13-5.

FIGURE 13-11: DATA EEPROM READ FLOWCHART



13.3.4 WRITING TO THE DATA EEPROM MEMORY

To write an EEPROM data location, the address must first be written to the NVMADRL and NVMADRH register pair and the data written to the NVMDAT register. The sequence in Example 13-6 must be followed to initiate the write cycle.

The write will not begin if NVM Unlock sequence, described in **Section 13.1.4 "NVM Unlock Sequence"**, is not exactly followed for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in NVMCON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, NVMCON1, NVMADRL, NVMADRH and NVMDAT cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. Both WR and WREN cannot be set with the same instruction.

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. A single Data EEPROM word is written and the operation includes an implicit erase cycle for that word (it is not necessary to set FREE). CPU execution continues in parallel and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The user can either enable this interrupt or poll this bit. NVMIF must be cleared by software.

13.3.5 WRITE VERIFY

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

EXAMPLE 13-5: DATA EEPROM READ

```
; Data Memory Address to read
            CLRF      NVMCON1          ; Setup Data EEPROM Access
            MOVF      EE_ADDRL, W      ;
            MOVWF     NVMADRL          ; Setup Address
            BSF       NVMCON1, RD      ; Issue EE Read
            MOVF      NVMDAT, W        ; W = EE_DATA
```

EXAMPLE 13-6: DATA EEPROM WRITE

```
; Data Memory Address to write
            CLRF      NVMCON1          ; Setup Data EEPROM Access
            MOVF      EE_ADDRL, W      ;
            MOVWF     NVMADRL          ; Setup Address
; Data Memory Value to write
            MOVF      EE_DATA, W       ;
            MOVWF     NVMDAT           ;
; Enable writes
            BSF       NVMCON1, WREN    ;
; Disable interrupts
            BCF       INTCON0, GIE     ;
; Required unlock sequence
            MOVLW     55h              ;
            MOVWF     NVMCON2          ;
            MOVLW     AAh              ;
            MOVWF     NVMCON2          ;
; Set WR bit to begin write
            BSF       NVMCON1, WR      ;
; Enable INT
            BSF       INTCON0, GIE     ;
; Wait for interrupt, write done
            SLEEP                      ;
; Disable writes
            BCF       NVMCON1, WREN    ;
```

13.3.6 OPERATION DURING CODE-PROTECT

Data EEPROM Memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is enabled.

If the Data EEPROM is write-protected or if NVMADR points an invalid address location, the WR bit is cleared without any effect. WRERR is signaled in this scenario.

13.3.7 PROTECTION AGAINST SPURIOUS WRITE

There are conditions when the user may not want to write to the Data EEPROM Memory. To protect against spurious EEPROM writes, various mechanisms have been implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (T_{PWRT}).

The unlock sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

13.3.8 ERASING THE DATA EEPROM MEMORY

Data EEPROM Memory can be erased by writing 0xFF to all locations in the Data EEPROM Memory that needs to be erased.

EXAMPLE 13-7: DATA EEPROM REFRESH ROUTINE

```
        CLRF      NVMADRL              ; Start at address 0
        BCF       NVMCON1, CFGS        ; Set for memory
        BCF       NVMCON1, EEPGD       ; Set for Data EEPROM
        BCF       INTCON0, GIE         ; Disable interrupts
        BSF       NVMCON1, WREN        ; Enable writes
Loop                                   ; Loop to refresh array
        BSF       NVMCON1, RD          ; Read current address
        MOVLW     55h                  ;
        MOVWF     NVMCON2              ; Write 55h
        MOVLW     0AAh                 ;
        MOVWF     NVMCOM2              ; Write 0AAh
        BSF       NVMCON1, WR          ; Set WR bit to begin write
        BTFSC     NVMCON1, WR          ; Wait for write to complete
        BRA       $-2
        INCFSZ    NVMADRL, F           ; Increment address
        BRA       LOOP                 ; Not zero, do it again

        BCF       NVMCON1, WREN        ; Disable writes
        BSF       INTCON0, GIE         ; Enable interrupts
```

13.4 Register Definitions: Nonvolatile Memory

REGISTER 13-1: NVMCON1: NONVOLATILE MEMORY CONTROL 1 REGISTER

R/W-0/0	R/W-0/0	U-0	R/S/HC-0/0	R/W/HS-x/q	R/W-0/0	R/S/HC-0/0	R/S/HC-0/0
REG[1:0]		—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	HC = Bit is cleared by hardware
x = Bit is unknown	-n = Value at POR	S = Bit can be set by software, but not cleared
'0' = Bit is cleared	'1' = Bit is set	U = Unimplemented bit, read as '0'

bit 7-6 **REG[1:0]:** NVM Region Selection bit
`10` =Access PFM Locations
`x1` = Access User IDs, Configuration Bits, DIA, DCI, Rev ID and Device ID
`00` = Access Data EEPROM Memory Locations

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

bit 4 **FREE:** Program Flash Memory Erase Enable bit(1)
`1` = Performs an erase operation on the next WR command
`0` = The next WR command performs a write operation

bit 3 **WRERR:** Write-Reset Error Flag bit(2,3,4)
`1` = A write operation was interrupted by a Reset (hardware set),
or WR was written to 0b1 when an invalid address is accessed (Table 9-1, Table 13-1)
or WR was written to 0b1 when REG[1:0] and address do not point to the same region
or WR was written to 0b1 when a write-protected address is accessed (Table 9-2).
`0` = All write operations have completed normally

bit 2 **WREN:** Program/Erase Enable bit
`1` = Allows program/erase and refresh cycles
`0` = Inhibits programming/erasing and user refresh of NVM

bit 1 **WR:** Write Control bit(5,6,7)
When REG points to a Data EEPROM Memory location:
`1` = Initiates an erase/program cycle at the corresponding Data EEPROM Memory location
When REG points to a PFM location:
`1` = Initiates the PFM write operation with data from the holding registers
`0` = NVM program/erase operation is complete and inactive

bit 0 **RD:** Read Control bit(8)
`1` = Initiates a read at address pointed by REG and NVMADR, and loads data into NVMDAT
`0` = NVM read operation is complete and inactive

Note 1: This can only be used with PFM.
2: This bit is set when WR = `1` and clears when the internal programming timer expires or the write is completed successfully.
3: Bit must be cleared by the user; hardware will not clear this bit.
4: Bit may be written to '`1`' by the user in order to implement test sequences.
5: This bit can only be set by following the unlock sequence of **Section 13.1.4 "NVM Unlock Sequence"**.
6: Operations are self-timed and the WR bit is cleared by hardware when complete.
7: Once a write operation is initiated, setting this bit to zero will have no effect.
8: The bit can only be set in software. The bit is cleared by hardware when the operation is complete.

REGISTER 13-2: NVMCON2: NONVOLATILE MEMORY CONTROL 2 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMCON2[7:0]							
bit 7							bit 0

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

bit 7-0 **NVMCON2[7:0]:**
Refer to **Section 13.1.4 “NVM Unlock Sequence”**.

Note 1: This register always reads zeros, regardless of data written.

Register 13-3: NVMADRL: Data EEPROM Memory Address Low

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ADR[7:0]							
bit 7							bit 0

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

bit 7-0 **ADR[7:0]:** EEPROM Read Address bits

REGISTER 13-4: NVMADRH: DATA EEPROM MEMORY ADDRESS HIGH(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	ADR[9:8]	
bit 7							bit 0

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

bit 7-2 **Unimplemented:** Read as ‘0’
bit 1-0 **ADR[9:8]:** EEPROM Read Address bits

Note 1: The NVMADRH register is not implemented on PIC18(L)F45/55K42.

REGISTER 13-5: NVMDAT: DATA EEPROM MEMORY DATA

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
DAT[7:0]							
bit 7							bit 0

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

bit 7-0 **DAT[7:0]:** The value of the data memory word returned from NVMADR after a Read command, or the data written by a Write command.

TABLE 13-4: SUMMARY OF REGISTERS ASSOCIATED WITH NONVOLATILE MEMORY CONTROL

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
NVMCON1	REG[1:0]		—	FREE	WRERR	WREN	WR	RD	210
NVMCON2	Unlock Pattern								211
NVMADRL	NVMADR[7:0]								211
NVMADRH (1)	—	—	—	—	—	—	NVMADR[9:8]		211
NVMDAT	NVMDAT[7:0]								212

Legend: — = unimplemented, read as ‘0’. Shaded bits are not used during EEPROM access.

*Page provides register information.

Note 1: The NVMADRH register is not implemented on PIC18(L)F45/55K42.

14.0 CYCLIC REDUNDANCY CHECK (CRC) MODULE WITH MEMORY SCANNER

The Cyclic Redundancy Check (CRC) module provides a software-configurable hardware-implemented CRC checksum generator. This module includes the following features:

- Any standard CRC up to 16 bits can be used
- Configurable Polynomial
- Any seed value up to 16 bits can be used
- Standard and reversed bit order available
- Augmented zeros can be added automatically or by the user
- Memory scanner for fast CRC calculations on program/Data EEPROM memory user data
- Software loadable data registers for communication CRC's

14.1 CRC Module Overview

The CRC module provides a means for calculating a check value of program/Data EEPROM memory. The CRC module is coupled with a memory scanner for faster CRC calculations. The memory scanner can automatically provide data to the CRC module. The CRC module can also be operated by directly writing data to SFRs, without using a scanner.

14.2 CRC Functional Overview

The CRC module can be used to detect bit errors in the program memory using the built-in memory scanner or through user input RAM memory. The CRC module can accept up to a 16-bit polynomial with up to a 16-bit seed value. A CRC calculated check value (or checksum) will then be generated into the CRCACC[15:0] registers for user storage. The CRC module uses an XOR shift register implementation to perform the polynomial division required for the CRC calculation.

EXAMPLE 14-1: CRC EXAMPLE

Rev. 10-000206A
1/8/2014

CRC-16-ANSI

$x^{16} + x^{15} + x^{2} + 1$ (17 bits)

Standard 16-bit representation = 0x8005

```
CRCXORH = 0b10000000
CRCXORL = 0b0000010-
```
(1)

Data Sequence:
0x55, 0x66, 0x77, 0x88

```
DLEN = 0b0111
PLEN = 0b1111
```

Data entered into the CRC:
SHIFTM = 0:

```
01010101 01100110 01110111 10001000
```

SHIFTM = 1:

```
10101010 01100110 11101110 00010001
```

Check Value (ACCM = 1):

SHIFTM = 0: 0x32D6

```
CRCACCH = 0b00110010
CRCACCL = 0b11010110
```

SHIFTM = 1: 0x6BA2

```
CRCACCH = 0b01101011
CRCACCL = 0b10100010
```

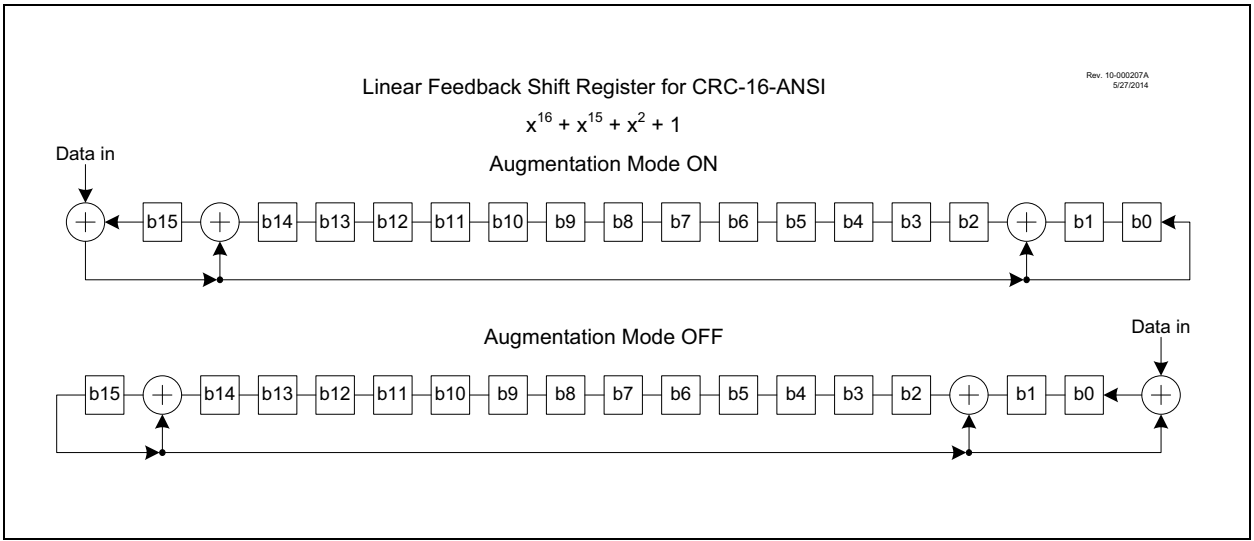
Note 1: Bit 0 is unimplemented. The LSb of any CRC polynomial is always ‘1’ and will always be treated as a ‘1’ by the CRC for calculating the CRC check value. This bit will be read in software as a ‘0’.

14.3 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the PLEN[3:0] bits. For an n-bit accumulator, PLEN = `n-1` and the corresponding polynomial is n+1 bits. Therefore the accumulator can be any size up to 16 bits with a corresponding polynomial up to 17 bits. The MSb and LSb of the polynomial are always '`1`' which is forced by hardware. All polynomial bits between the MSb and LSb are specified by the CRCXOR registers. For example, when using CRC-16-ANSI, the polynomial is defined as $X^{16}+X^{15}+X^{2}+1$.

The X^{16} and X^{0} = `1` terms are the MSb and LSb controlled by hardware. The X^{15} and X^{2} terms are specified by setting the corresponding CRCXOR[15:0] bits with the value of '`0x8004`'. The actual value is '`0x8005`' because the hardware sets the LSb to 1. However, the LSb of the CRCXORL register is unimplemented and always reads as '`0`'. Refer to Example 14-1.

EXAMPLE 14-2: CRC LFSR EXAMPLE



14.4 CRC Data Sources

Data can be input to the CRC module in two ways:

- User data using the CRCDAT registers (CRCDATH and CRCDATL)
- Program memory using the Program Memory Scanner

To set the number of bits of data, up to 16 bits, the DLEN bits of CRCCON1 must be set accordingly. Only data bits in CRCDAT registers up to DLEN will be used, other data bits in CRCDAT registers will be ignored.

Data is moved into the CRCSHIFT as an intermediate to calculate the check value located in the CRCACC registers.

The SHIFTM bit is used to determine the bit order of the data being shifted into the accumulator. If SHIFTM is not set, the data will be shifted in MSb first (Big Endian). The value of DLEN will determine the MSb. If SHIFTM bit is set, the data will be shifted into the accumulator in reversed order, LSb first (Little Endian).

The CRC module can be seeded with an initial value by setting the CRCACC[15:0] registers to the appropriate value before beginning the CRC.

14.4.1 CRC FROM USER DATA

To use the CRC module on data input from the user, the user must write the data to the CRCDAT registers. The data from the CRCDAT registers will be latched into the shift registers on any write to the CRCDATL register.

14.4.2 CRC FROM FLASH

To use the CRC module on data located in Program memory, the user can initialize the Program Memory Scanner as defined in **Section 14.8, Scanner Module Overview**.

14.5 CRC Check Value

The CRC check value will be located in the CRCACC registers after the CRC calculation has finished. The check value will depend on two mode settings of the CRCCON0 register: ACCM and SHIFTM. When the ACCM bit is set, the CRC module augments the data with a number of zeros equal to the length of the polynomial to align the final check value. When the ACCM bit is not set, the CRC will stop at the end of the data. A number of zeros equal to the length of the polynomial can then be entered into CRCDAT to find the same check value as augmented mode. Alternatively, the expected check value can be entered at this point to make the final result equal '0'.

When the CRC check value is computed with the SHIFTM bit set, selecting LSb first, and the ACCM bit is also set, then the final value in the CRCACC registers will be reversed such that the LSb will be in the MSb position and vice versa. This is the expected check value in bit reversed form. If you are creating a check value to be appended to a data stream, then a bit reversal must be performed on the final value to achieve the correct checksum. You can use the CRC to do this reversal by the following method:

- Save the CRCACC value in user RAM space
- Clear the CRCACC registers
- Clear the CRCXOR registers
- Write the saved CRCACC value to the CRCDAT input.

The properly oriented check value will be in the CRCACC registers as the result.

14.6 CRC Interrupt

The CRC will generate an interrupt when the BUSY bit transitions from 1 to 0. The CRCIF Interrupt Flag is set every time the BUSY bit transitions, regardless of whether or not the CRC interrupt is enabled. The CRCIF bit can only be cleared in software.

14.7 Configuring the CRC

The following steps illustrate how to properly configure the CRC.

1. Determine if the automatic program memory scan will be used with the scanner or manual calculation through the SFR interface and perform the actions specified in **Section 14.4 "CRC Data Sources"**, depending on which decision was made.
2. If desired, seed a starting CRC value into the CRCACCH/L registers.
3. Program the CRCXORH/L registers with the desired generator polynomial.
4. Program the DLEN[3:0] bits of the CRCCON1 register with the length of the data word - 1 (refer to Example 14-1). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the PLEN[3:0] bits of the CRCCON1 register with the length of the polynomial -2 (refer to Example 14-1).
6. Determine whether shifting in trailing zeros is desired and set the ACCM bit of the CRCCON0 register appropriately.
7. Likewise, determine whether the MSb or LSb should be shifted first and write the SHIFTM bit of the CRCCON0 register appropriately.
8. Write the GO bit of the CRCCON0 register to begin the shifting process.

9a. If manual SFR entry is used, monitor the FULL bit of the CRCCON0 register. When FULL = 0, another word of data can be written to the CRCDATH/L registers, keeping in mind that CRCDATH should be written first if the data has more than eight bits, as the shifter will begin upon the CRCDATL register being written.

9b. If the scanner is used, the scanner will automatically load words into the CRCDATH/L registers as needed, as long as the GO bit is set.

10a. If manual entry is used, monitor the CRCIF (and BUSY bit to determine when the completed CRC calculation can be read from CRCACCH/L registers.

10b. If using the memory scanner, monitor the SCANIF (or the GO bit) for the scanner to finish pushing information into the CRCDAT registers. After the scanner is completed, monitor the BUSY bit to determine that the CRC has been completed and the check value can be read from the CRCACC registers. If both the interrupt flags are set and the BUSY and GO bits are cleared, the completed CRC calculation can be read from the CRCACCH/L registers.

14.8 Scanner Module Overview

The Scanner allows segments of the Program Flash Memory or Data EEPROM, to be read out (scanned) to the CRC Peripheral. The Scanner module interacts with the CRC module and supplies it data one word at a time. Data is fetched from the address range defined by SCANLADR registers up to the SCANHADR registers.

The Scanner begins operation when the SGO bit is set (SCANCON0 Register) and ends when either SGO is cleared by the user or when SCANLADR increments past SCANHADR. The SGO bit is also cleared by clearing the EN bit (CRCCON0 register).

14.9 Configuring the Scanner

The scanner module may be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory or Data EEPROM addresses. In order to set up the scanner to work with the CRC, perform the following steps:

1. Set up the CRC module (See **Section 14.7 "Configuring the CRC"**) and enable the Scanner module by setting the EN bit in the SCANCON0 register.
2. Choose which memory region the Scanner module should operate on and set the MREG bit of the SCANCON0 register appropriately.
3. If trigger is used for scanner operation, set the TRIGEN bit of the SCANCON0 register and select the trigger source using SCANTRIG register. Select the trigger source using SCANTRIG register and then set the TRIGEN bit of the SCANCON0 register. See Table 14-1 for Scanner Operation.
4. If Burst mode of operation is desired, set the BURSTMD bit (SCANCON0 register). See Table 14-1 for Scanner Operation.
5. Set the SCANLADRL/H/U and SCANHADRL/H/U registers with the beginning and ending locations in memory that are to be scanned.
6. Select the priority level for the Scanner module (See **Section 3.1 "System Arbitration"**) and lock the priorities (See **Section 3.1.1 "Priority Lock"**).
7. Both CRCEN and CRCGO bits must be enabled to use the scanner. Setting the SGO bit will start the scanner operation.

14.10 Scanner Interrupt

The scanner will trigger an interrupt when the SCANLADR increments past SCANHADR. The SCANIF bit can only be cleared in software.

14.11 Scanning Modes

The interaction of the scanner with the system operation is controlled by the priority selection in the System Arbiter (see **Section 3.2 "Memory Access Scheme"**). Additionally, BURSTMD and TRIGEN also determine the operation of the Scanner.

14.11.1 TRIGEN = 0, BURSTMD = 0

In this case, the memory access request is granted to the scanner if no other higher priority source is requesting access.

All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

14.11.2 TRIGEN = 1, BURSTMD = 0

In this case, the memory access request is generated when the CRC module is ready to accept.

The memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

The memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

14.11.3 TRIGEN = x, BURSTMD = 1

In this case, the memory access is always requested by the scanner.

The memory access request is granted to the scanner if no other higher priority source is requesting access. The memory access cycles will not be granted to lower priority sources than the scanner until it completes operation i.e. SGO = 0 (SCANCON0 register)

Note:	If TRIGEN = 1 and BURSTMD = 1, the user should ensure that the trigger source is active for the Scanner operation to complete.

14.12 Register Definitions: CRC and Scanner Control

Long bit name prefixes for the CRC and Scanner peripherals are shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
CRC	CRC

REGISTER 14-1: CRCCON0: CRC CONTROL REGISTER 0

R/W-0/0	R/W-0/0	R-0	R/W-0/0	U-0	U-0	R/W-0/0	R-0
EN	GO	BUSY	ACCM	—	—	SHIFTM	FULL
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 **EN:** CRC Enable bit
1 = CRC module is enabled
0 = CRC is disabled

bit 6 **GO:** CRC Go bit
1 = Start CRC serial shifter
0 = CRC serial shifter turned off

bit 5 **BUSY:** CRC Busy bit
1 = Shifting in progress or pending
0 = All valid bits in shifter have been shifted into accumulator

bit 4 **ACCM:** Accumulator Mode bit
1 = Data is concatenated with zeros
0 = Data is not concatenated with zeros

bit 3-2 **Unimplemented:** Read as '0'

bit 1 **SHIFTM:** Shift Mode bit
1 = Shift right (LSb)
0 = Shift left (MSb)

bit 0 **FULL:** Data Path Full Indicator bit
1 = CRCDATH/L registers are full
0 = CRCDATH/L registers have shifted their data into the shifter

REGISTER 14-2: CRCCON1: CRC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
DLEN[3:0]				PLEN[3:0]			
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 7-4 **DLEN[3:0]:** Data Length bits
Denotes the length of the data word -1 (See Example 14-1)

bit 3-0 **PLEN[3:0]:** Polynomial Length bits
Denotes the length of the polynomial -1 (See Example 14-1)

REGISTER 14-3: CRCDATH: CRC DATA HIGH BYTE REGISTER

R/W-xx	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
DATA[15:8]							
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 7-0 **DATA[15:8]**: CRC Input/Output Data bits

REGISTER 14-4: CRCDATL: CRC DATA LOW BYTE REGISTER

R/W-xx	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
DATA[7:0]							
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 7-0 **DATA[7:0]**: CRC Input/Output Data bits
Writing to this register fills the shifter.

REGISTER 14-5: CRCACCH: CRC ACCUMULATOR HIGH BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ACC[15:8]							
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 7-0 **ACC[15:8]**: CRC Accumulator Register bits

REGISTER 14-6: CRCACCL: CRC ACCUMULATOR LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ACC[7:0]							
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 7-0 **ACC[7:0]**: CRC Accumulator Register bits

REGISTER 14-7: CRCSHIFTH: CRC SHIFT HIGH BYTE REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SHIFT[15:8]							
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 7-0 **SHIFT[15:8]**: CRC Shifter Register bits
Reading from this register reads the CRC Shifter.

REGISTER 14-8: CRCSHIFTL: CRC SHIFT LOW BYTE REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SHIFT[7:0]							
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 7-0 **SHIFT[7:0]**: CRC Shifter Register bits
Reading from this register reads the CRC Shifter.

REGISTER 14-9: CRCXORH: CRC XOR HIGH BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
X[15:8]							
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 7-0 **X[15:8]**: XOR of Polynomial Term X^n Enable bits

REGISTER 14-10: CRCXORL: CRC XOR LOW BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	U-1
X[7:1]							—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit
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 7-1 **X[7:1]**: XOR of Polynomial Term X^n Enable bits

bit 0 **Unimplemented:** Read as ‘1’

REGISTER 14-11: SCANCON0: SCANNER ACCESS CONTROL REGISTER 0

R/W-0/0	R/W-0/0	R/W/HC-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R-0/0
EN	TRIGEN	SGO	—	—	MREG	BURSTMD	BUSY
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	HC = Bit is cleared by hardware

bit 7 **EN:** Scanner Enable bit(1)
1 = Scanner is enabled
0 = Scanner is disabled

bit 6 **TRIGEN:** Scanner Trigger Enable bit(2)
1 = Scanner trigger is enabled
0 = Scanner trigger is disabled
Refer Table 14-1.

bit 5 **SGO:** Scanner GO bit(3, 4)
1 = When the CRC is ready, the Memory region set by the MREG bit will be accessed and data is passed to the CRC peripheral.
0 = Scanner operations will not occur

bit 4-3 **Unimplemented:** Read as ‘0’

bit 2 **MREG:** Scanner Memory Region Select bit(2)
1 = Scanner address points to Data EEPROM
0 = Scanner address points to Program Flash Memory

bit 1 **BURSTMD:** Scanner Burst Mode bit
1 = Memory access request to the CPU Arbiter is always true
0 = Memory access request to the CPU Arbiter is dependent on the CRC request and Trigger
Refer Table 14-1.

bit 0 **BUSY:** Scanner Busy Indicator bit
1 = Scanner cycle is in process
0 = Scanner cycle is compete (or never started)

Note 1: Setting EN = 1 (SCANCON0 register) does not affect any other register content.
2: Scanner trigger selection can be set using the SCANTRIG register.
3: This bit can be cleared in software. It is cleared in hardware when LADR>HADR (and a data cycle is not occurring) or when CRCGO = 0 (CRCCON0 register).
4: CRCEN and CRCGO bits (CRCCON0 register) must be set before setting the SGO bit.

TABLE 14-1: SCANNER OPERATING MODES(1)

TRIGEN	BURSTMD	Scanner Operation
0	0	Memory access is requested when the CRC module is ready to accept data; the request is granted if no other higher priority source request is pending.
1	0	Memory access is requested when the CRC module is ready to accept data and trigger selection is true; the request is granted if no other higher priority source request is pending.
x	1	Memory access is always requested, the request is granted if no other higher priority source request is pending.

Note 1: See **Section 3.1 "System Arbitration"** for Priority selection and **Section 3.2 "Memory Access Scheme"** for Memory Access Scheme.

REGISTER 14-12: SCANLADRU: SCAN LOW ADDRESS UPPER BYTE REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	LADR[21:16](1,2)					
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 **Unimplemented:** Read as '0'

bit 5-0 **LADR[21:16]:** Scan Start/Current Address bits(1,2)
Upper bits of the current address to be fetched from, value increments on each fetch of memory.

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).

2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-13: SCANLADRH: SCAN LOW ADDRESS HIGH BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LADR[15:8](1, 2)							
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 7-0 **LADR[15:8]**: Scan Start/Current Address bits(1, 2)
Most Significant bits of the current address to be fetched from, value increments on each fetch of memory.

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).

2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-14: SCANLADRL: SCAN LOW ADDRESS LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LADR[7:0](1, 2)							
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 7-0 **LADR[7:0]**: Scan Start/Current Address bits(1, 2)
Least Significant bits of the current address to be fetched from, value increments on each fetch of memory

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-15: SCANHADRU: SCAN HIGH ADDRESS UPPER BYTE REGISTER

U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	HADR[21:16]					
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 **Unimplemented:** Read as ‘0’
bit 5-0 **HADR[21:16]:** Scan End Address bits(1, 2)
Upper bits of the address at the end of the designated scan

Note 1: Registers SCANHADRU/H/L form a 22-bit value but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-16: SCANHADRH: SCAN HIGH ADDRESS HIGH BYTE REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
HADR[15:8](1, 2)							
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 7-0 **HADR[15:8]**: Scan End Address bits(1, 2)
Most Significant bits of the address at the end of the designated scan

Note 1: Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-17: SCANHADRL: SCAN HIGH ADDRESS LOW BYTE REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
HADR[7:0](1, 2)							
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 7-0 **HADR[7:0]**: Scan End Address bits(1, 2)
Least Significant bits of the address at the end of the designated scan

Note 1: Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-18: SCANTRIG: SCAN TRIGGER SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	TSEL[3:0]			
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 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **TSEL[3:0]**: Scanner Data Trigger Input Selection bits

1111 = Reserved
•
•
•

1010 = Reserved
1001 = SMT1_output
1000 = TMR6_postscaled
0111 = TMR5_output
0110 = TMR4_postscaled
0101 = TMR3_output
0100 = TMR2_postscaled
0011 = TMR1_output
0010 = TMR0_output
0001 = CLKREF_output
0000 = LFINTOSC

TABLE 14-2: SUMMARY OF REGISTERS ASSOCIATED WITH CRC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CRCACCH	ACC[15:8]								219
CRCACCL	ACC[7:0]								220
CRCCON0	EN	GO	BUSY	ACCM	—	—	SHIFTM	FULL	218
CRCCON1	DLEN[3:0]				PLEN[3:0]				218
CRCDATH	DATA[15:8]								219
CRCDATL	DATA[7:0]								219
CRCSHIFTH	SHIFT[15:8]								220
CRCSHIFTL	SHIFT[7:0]								220
CRCXORH	X[15:8]								221
CRCXORL	X[7:1]							—	221
SCANCON0	EN	TRIGEN	SGO	—	—	MREG	BURSTMD	BUSY	222
SCANHADRU	—	—	HADR[21:16]						224
SCANHADRH	HADR[15:8]								225
SCANHADRL	HADR[7:0]								225
SCANLADRU	—	—	LADR[21:16]						223
SCANLADRH	LADR[15:8]								223
SCANLADRL	LADR[7:0]								224
SCANTRIG	—	—	—	—	TSEL[3:0]				226

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for the CRC module.

15.0 DIRECT MEMORY ACCESS (DMA)

15.1 Introduction

The Direct Memory Access (DMA) module is designed to service data transfers between different memory regions directly without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.

PIC18(L)F26/27/45/46/47/55/56/57K42 family has two DMA modules which can be independently programmed to transfer data between different memory locations, move different data sizes, and use a wide range of hardware triggers to initiate transfers. The two DMA registers can even be programmed to work together, in order to carry out more complex data transfers without CPU overhead.

Key features of the DMA module include:

- Support access to the following memory regions:
  - GPR and SFR space (R/W)
  - Program Flash Memory (R only)
  - Data EEPROM Memory (R only)
- Programmable priority between the DMA and CPU Operations. Refer to **Section 3.1 "System Arbitration"** for details.
- Programmable Source and Destination address modes
  - Fixed address
  - Post-increment address
  - Post-decrement address
- Programmable Source and Destination sizes
- Source and destination pointer register, dynamically updated and reloadable
- Source and destination count register, dynamically updated and reloadable
- Programmable auto-stop based on Source or Destination counter
- Software triggered transfers
- Multiple user selectable sources for hardware triggered transfers
- Multiple user selectable sources for aborting DMA transfers

15.2 DMA Registers

The operation of the DMA module has the following registers:

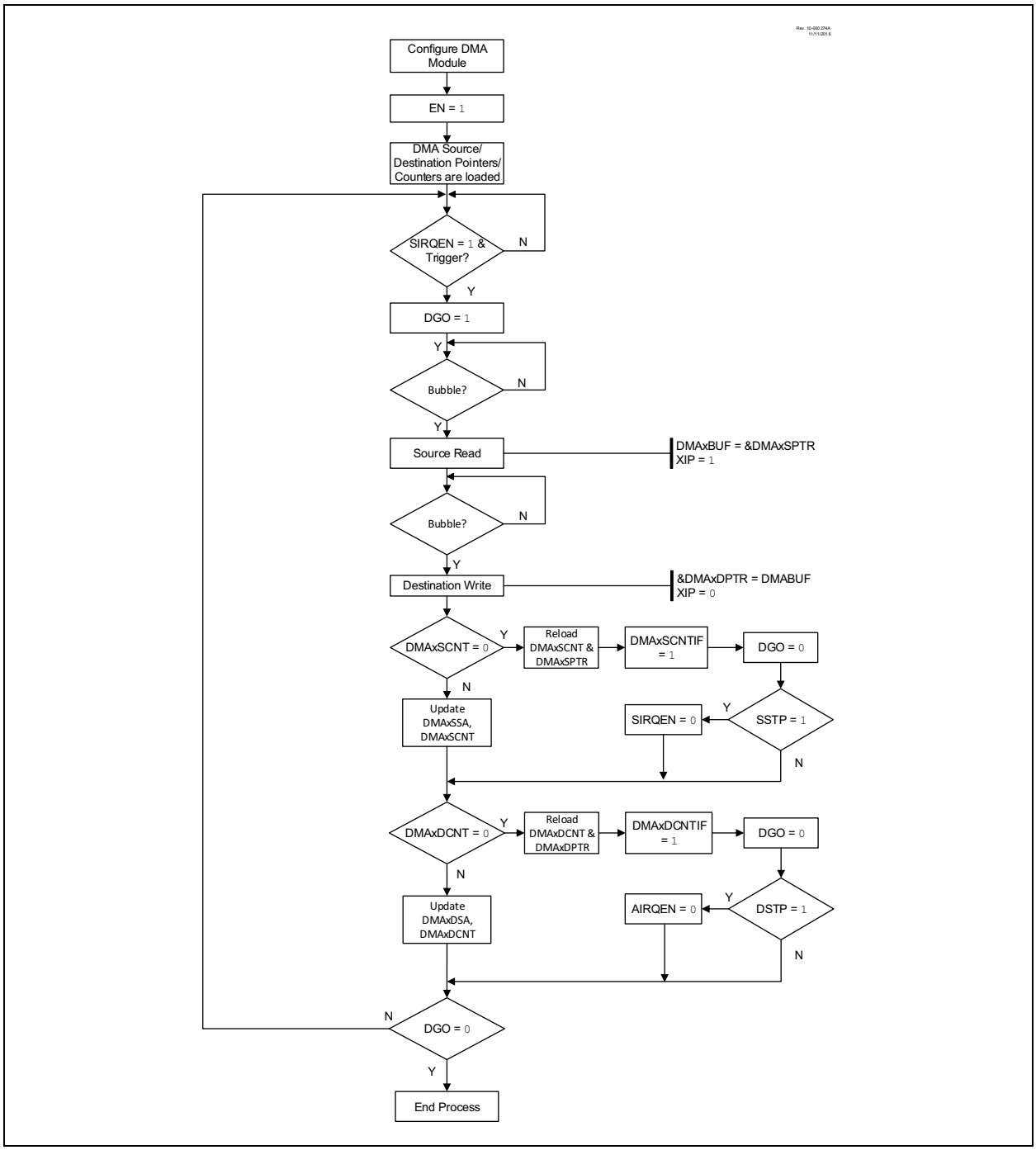
- Control registers (DMAxCON0, DMAxCON1)
- Data buffer register (DMAxBUF)
- Source Start Address Register (DMAxSSAU:H:L)
- Source Pointer Register (DMAxSPTRU:H:L)
- Source Message Size Register (DMAxSSZH:L)
- Source Count Register (DMAxSCNTH:L)
- Destination Start Address Register (DMAxDSAH:L)
- Destination Pointer Register (DMAxDPTRH:L)
- Destination Message Size Register (DMAxDSZH:L)
- Destination Count Register (DMAxDCNTH:L)
- Start Interrupt Request Source Register (DMAxSIRQ)
- Abort Interrupt Request Source Register (DMAxAIRQ)

These registers are detailed in **Section 15.13 "Register definitions: DMA"**.

15.3 DMA Organization

The DMA module on the K42 family of devices is designed to move data by using the existing Instruction Bus[16] and Data Bus[8] without the need for any dual-porting of memory or peripheral systems (Figure 15-1). The DMA accesses the required bus when it has been granted to by the System Arbiter.

FIGURE 15-1: DMA FUNCTIONAL BLOCK DIAGRAM



Depending on the priority of the DMA with respect to CPU execution (Refer to **Section 3.2 "Memory Access Scheme"** for more information), the DMA Controller can move data through two methods:

- Stalling the CPU execution until it has completed its transfers (DMA has higher priority over the CPU in this mode of operation)
- Utilizing unused CPU cycles for DMA transfers (CPU has higher priority over the DMA in this mode of operation). Unused CPU cycles are referred to as bubbles which are instruction cycles available for use by the DMA to perform read and write operations. In this way, the effective bandwidth for handling data is increased; at the same time, DMA operations can proceed without causing a processor stall.

15.4 DMA Interface

The DMA module transfers data from the source to the destination one byte at a time, this smallest data movement is called a DMA data transaction. A DMA Message refers to one or more DMA data transactions.

Each DMA data transaction consists of two separate actions:

- Reading the Source Address Memory and storing the value in the DMA Buffer register
- Writing the contents of the DMA Buffer register to the Destination Address Memory

Note:	DMA data movement is a two-cycle operation.

The XIP bit (DMAxCON0 register) is a status bit to indicate whether or not the data in the DMAxBUF register has been written to the destination address. If the bit is set then data is waiting to be written to the destination. If clear, it means that either data has been written to the destination or that no source read has occurred.

The DMA has read access to PFM, Data EEPROM, and SFR/GPR space, and write access to SFR/GPR space. Based on these memory access capabilities, the DMA can support the following memory transactions:

TABLE 15-1: DMA MEMORY ACCESS

Read Source	Write Destination
Program Flash Memory	GPR
Program Flash Memory	SFR
Data EE	GPR
Data EE	SFR
GPR	GPR
SFR	GPR
GPR	SFR
SFR	SFR

Note:	Even though the DMA module has access to all memory and peripherals that are also available to the CPU, it is recommended that the DMA does not access any register that is part of the System arbitration. The DMA, as a system arbitration client should not be read or written by itself or by another DMA instantiation.

The following sections discuss the various control interfaces required for DMA data transfers.

15.4.1 DMA ADDRESSING

The start addresses for the source read and destination write operations are set using the DMAxSSA [21:0] and DMAxDSA [15:0] registers, respectively.

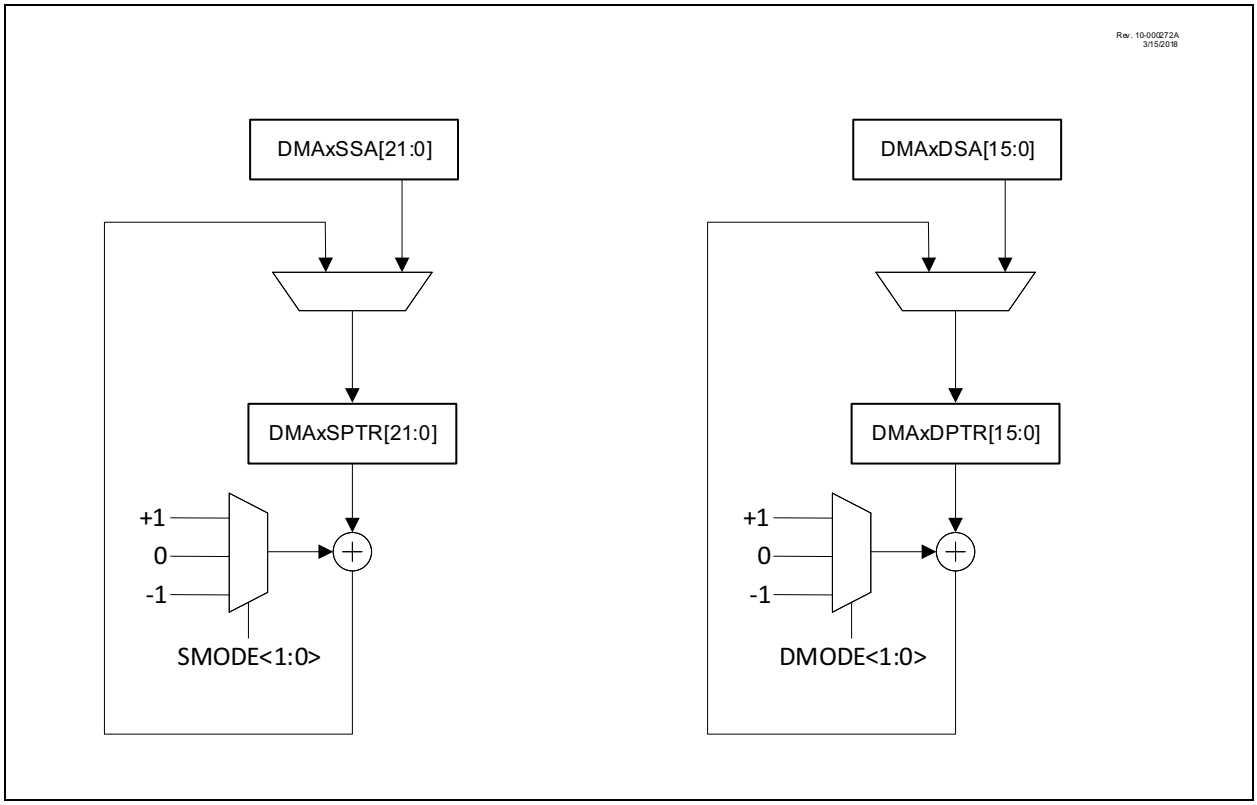
When the DMA Message transfers are in progress, the DMAxSPTR [21:0] and DMAxDPTR [15:0] registers contain the current address pointers for each source read and destination write operation, these registers are modified after each transaction based on the Address mode selection bits.

The SMODE and DMODE bits in the DMAxCON1 control register determine the address modes of operation by controlling how the DMAxSPTR [21:0] and DMAxDPTR [15:0] bits are updated after every DMA data transaction combination (Figure 15-2).

Each address can be separately configured to:

- Remain unchanged
- Increment by 1
- Decrement by 1

FIGURE 15-2: DMA POINTERS BLOCK DIAGRAM



The DMA can initiate data transfers from the PFM, Data EEPROM or SFR/GPR Space. The SMR[1:0] bits in the DMAxCON1 register are used to select the type of memory being pointed to by the Source Address Pointer. The SMR[1.0] bits are required because the PFM and SFR/GPR spaces have overlapping addresses that do not allow the specified address to uniquely define the memory location to be accessed.

Note 1: For proper memory read access to occur, the combination of address and space selection must be valid.

2: The destination does not have space selection bits because it can only write to the SFR/GPR space.

15.4.2 DMA MESSAGE SIZE/COUNTERS

A transaction is the transfer of one byte. A message consists of one or more transactions. A complete DMA process consists of one or more messages. The size registers determine how many transactions are in a message. The DMAxSSZ registers determine the source size and DMAxDSZ registers determine the destination size.

When a DMA transfer is initiated, the size registers are copied to corresponding counter registers that control the duration of the message. The DMAxSCNT registers count the source transactions and the DMAxDCNT registers count the destination transactions. Both are simultaneously decremented by one after each transaction.

A message is started by setting the DGO bit of the DMAxCON0 register and terminates when the smaller of the two counters reaches zero.

When either counter reaches zero the DGO bit is cleared and the counter and pointer registers are immediately reloaded with the corresponding size and address data. If the other counter did not reach zero then the next message will continue with the count and address corresponding to that register.

When the source and destination size registers are not equal, then the ratio of the largest to the smallest size determines how many messages are in the DMA process. For example, when the destination size is 6 and the source size is 2, then each message will consist of two transactions and the complete DMA process will consist of three messages. When the larger size is not an even integer of the smaller size, then the last message in the process will terminate early when the larger count reaches zero. In that case, the larger counter will reset and the smaller counter will have a remainder skewing any subsequent messages by that amount.

Note:	Reading the DMAxSCNT or DMAxDCNT registers will never return zero. When either register is decremented from '1' it is immediately reloaded from the corresponding size register.

FIGURE 15-3: DMA COUNTERS BLOCK DIAGRAM

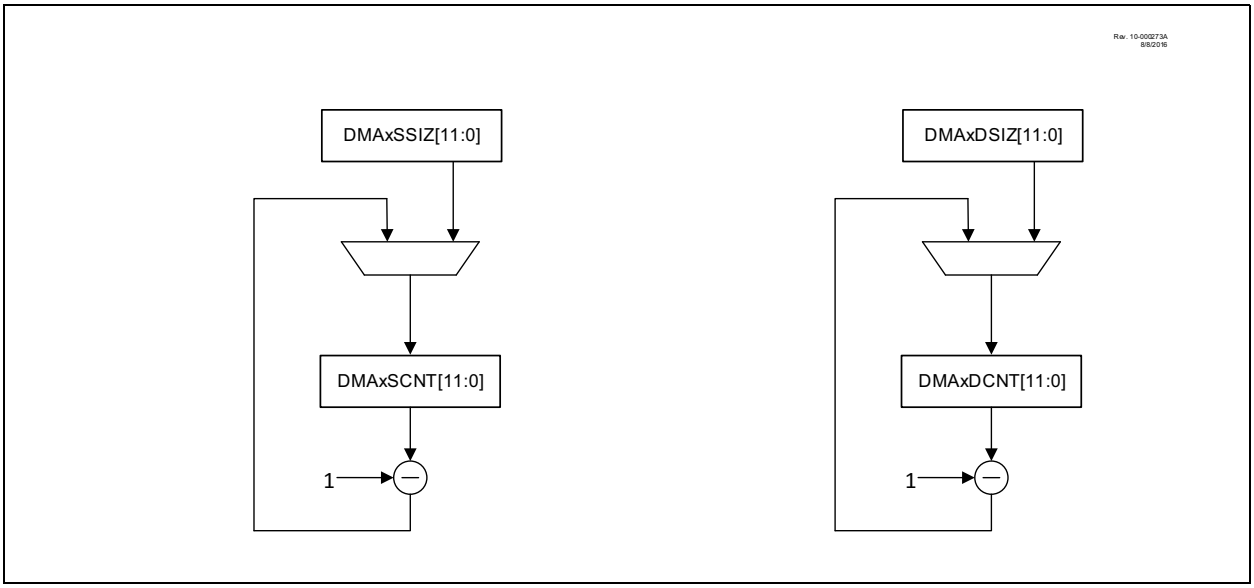


Table 15-2 has a few examples of configuring DMA Message sizes.

TABLE 15-2: EXAMPLE MESSAGE SIZE TABLE

Operation	Example	SCNT	DCNT	Comments
Read from single SFR location to RAM	U1RXB	1	N	N equals the number of bytes desired in the destination buffer. N >= 1.
Write to single SFR location from RAM	U1TXB	N	1	N equals the number of bytes desired in the source buffer. N >= 1.
Read from multiple SFR location	ADRES[H:L]	2	2*N	N equals the number of ADC results to be stored in memory. N>= 1
	TMR1[H:L]	2	2*N	N equals the number of TMR1 Acquisition results to be stored in memory. N>= 1
	SMT1CPR[U:H:L]	3	3*N	N equals the number of Capture Pulse Width measurements to be stored in memory. N>= 1
Write to Multiple SFR registers	PWMDC[H:L]	2*N	2	N equals the number of PWM duty cycle values to be loaded from a memory table. N>= 1
	All ADC registers	N*31	31	Using the DMA to transfer a complete ADC context from RAM to the ADC registers.N>= 1

15.5 DMA Message Transfers

Once the Enable bit is set to start DMA message transfers, the Source/Destination pointer and counter registers are initialized to the conditions shown in Table 15-3.

TABLE 15-3: DMA INITIAL CONDITIONS

Register	Value loaded
DMAxSPTR[21:0]	DMAxSSA[21:0]
DMAxSCNT[11:0]	DMAxSSZ[11:0]
DMAxDPTR[15:0]	DMAxDSA[15:0]
DMAxDCNT[11:0]	DMAxDSZ[11:0]

During the DMA Operation after each transaction, Table 15-4 and Table 15-5 indicate how the Source/Destination pointer and counter registers are modified.

TABLE 15-4: DMA SOURCE POINTER/COUNTER DURING OPERATION

Register	Modified Source Counter/Pointer Value
DMAxSCNT[11:0] != 1	DMAxSCNT = DMAxSCNT -1
	SMODE = 00: DMAxSPTR = DMAxSPTR
	SMODE = 01: DMAxSPTR = DMAxSPTR + 1
	SMODE = 10: DMAxSPTR = DMAxSPTR - 1
DMAxSCNT[11:0] == 1	DMAxSCNT = DMAxSSZ
	DMAxSPTR = DMAxSSA

TABLE 15-5: DMA DESTINATION POINTER/COUNTER DURING OPERATION

Register	Modified Destination Counter/Pointer Value
DMAxDCNT[11:0]!= 1	DMAxDCNT = DMAxDCNT -1
	DMODE = 00: DMAxDPTR = DMAxDPTR
	DMODE = 01: DMAxDPTR = DMAxDPTR + 1
	DMODE = 10: DMAxDPTR = DMAxDPTR - 1
DMAxDCNT[11:0] == 1	DMAxDCNT = DMAxDSZ
	DMAxDPTR = DMAxDSA

The following sections discuss how to initiate and terminate DMA transfers.

15.5.1 STARTING DMA MESSAGE TRANSFERS

The DMA can initiate data transactions by either of the following two conditions:

1. User software control
2. Hardware trigger, SIRQ

15.5.1.1 User Software Control

Software starts or stops DMA transaction by setting/clearing the DGO bit. The DGO bit is also used to indicate whether a DMA hardware trigger has been received and a message is in progress.

> **Note 1:** Software start can only occur if the EN bit (DMAxCON0) is set.
>
> **2:** If the CPU writes to the DGO bit while it is already set, there is no effect on the system, the DMA will continue to operate normally.

15.5.1.2 Hardware Trigger, SIRQ

A Hardware trigger is an interrupt request from another module sent to the DMA with the purpose of starting a DMA message. The DMA start trigger source is user selectable using the DMAxSIRQ register.

The SIRQEN bit (DMAxCON0 register) is used to enable sampling of external interrupt triggers by which a DMA transfer can be started. When set, the DMA will sample the selected Interrupt source and when cleared, the DMA will ignore the selected Interrupt source. Clearing SIRQEN does not stop a DMA transaction currently in progress, it only stops more hardware request signals from being received.

15.5.2 STOPPING DMA MESSAGE TRANSFERS

The DMA controller can stop data transactions by either of the following two conditions:

1. Clearing the DGO bit
2. Hardware trigger, AIRQ
3. Source Count reload
4. Destination Count reload
5. Clearing the Enable bit

15.5.2.1 User Software Control

If the user clears the DGO bit, the message will be stopped and the DMA will remain in the current configuration.

For example, if the user clears the DGO bit after source data has been read but before it is written to the destination, then the data in DMAxBUF will not reach its destination.

This is also referred to as a soft-stop as the operation can resume if desired by setting DGO bit again.

15.5.2.2 Hardware Trigger, AIRQ

The AIRQEN bit (DMAxCON0 register) is used to enable sampling of external interrupt triggers by which a DMA transaction can be aborted.

Once an Abort interrupt request has been received, the DMA will perform a soft-stop by clearing the DGO bit as well as clearing the SIRQEN bit so overruns do not occur. The AIRQEN bit is also cleared to prevent additional abort signals from triggering false aborts.

If desired, the DGO bit can be set again and the DMA will resume operation from where it left off after the soft-stop had occurred as none of the DMA state information is changed in the event of an abort.

15.5.2.3 Source Count Reload

A DMA message is considered to be complete when the Source count register is decremented from 1 and then reloaded (i.e., once the last byte from either the source read or destination write has occurred). When the SSTP bit is set (DMAxCON1 register) and the source count register is reloaded, then further message transfer is stopped.

15.5.2.4 Destination Count Reload

A DMA message is considered to be complete when the Destination count register is decremented from 1 and then reloaded (i.e., once the last byte from either the source read or destination write has occurred). When the DSTP bit is set (DMAxCON1) and the destination count register is reloaded then further message transfer is stopped.

Note:	Reading the DMAxSCNT or DMAxDCNT registers will never return zero. When either register is decremented from '1' it is immediately reloaded from the corresponding size register.

15.5.2.5 Clearing the Enable bit

If the User clears the EN bit, the message will be stopped and the DMA will return to its default configuration. This is also referred to as a hard-stop as the DMA cannot resume operation from where it was stopped.

Note:	After the DMA message transfer is stopped, it requires an extra instruction cycle before the Stop condition takes effect. Thus, after the Stop condition has occurred, a Source read or a Destination write can occur depending on the Source or Destination Bus availability.

15.5.3 DISABLE DMA MESSAGES TRANSFERS UPON COMPLETION

Once the DMA message is complete it may be desirable to disable the trigger source to prevent overrun or under run of data. This can be done by either of the following methods:

1. Clearing the SIRQEN bit
2. Setting the SSTP bit
3. Setting the DSTP bit

15.5.3.1 Clearing the SIRQEN bit

Clearing the SIRQEN bit (DMAxCON1 register) stops the sampling of external start interrupt triggers, hence preventing further DMA Message transfers.

An example would be a communications peripheral with a level-triggered interrupt. The peripheral will continue to request data (because its buffer is empty) even though there is no more data to be moved. Disabling the SIRQEN bit prevents the DMA from processing these requests.

15.5.3.2 Source/Destination Stop

The SSTP and DSTP bits (DMAxCON0 register) determine whether or not to disable the hardware triggers (SIRQEN = 0) once a DMA message has completed.

When the SSTP bit is set and the DMAxSCNT = 0, then the SIRQEN bit will be cleared. Similarly, when the DSTP bit is set and the DMAxDCNT = 0, the SIRQEN bit will be cleared.

Note:	The SSTP and DSTP bits are independent functions and do not depend on each other. It is possible for a message to be stopped by either counter at message end or both counters at message end.

15.6 Types of Hardware Triggers

The DMA has two different trigger inputs namely the Source trigger and the abort trigger. Each of these trigger sources is user configurable using the DMAxSIRQ and DMAxAIRQ registers.

Based on the source selected for each trigger, there are two types of requests that can be sent to the DMA.

- Edge triggers
- Level triggers

15.6.1 EDGE TRIGGER REQUESTS

Edge triggers are generated by the signal that sets the corresponding interrupt flag. The DMA responds to this event but leaves the interrupt flag set. An Edge request occurs only once when a given module interrupt requirements are true.

15.6.2 LEVEL TRIGGER REQUESTS

A level request is asserted as long as the condition that causes the interrupt is true. For example, the RXIF interrupt is asserted as long as the UART receive buffer has unread data. The RXIF cannot be cleared except by emptying the receive buffer.

15.7 Types of Data Transfers

Based on the memory access capabilities of the DMA (See Table 15-1), the following sections discuss the different types of data movement between the Source and Destination Memory regions.

- N: 1

This type of transfer is common when sending predefined data packets (such as strings) through a single interface point (such as communications modules transmit registers).

- N: N

This type of transfer is useful for moving information out of the Program Flash or Data EEPROM to SRAM for manipulation by the CPU or other peripherals.

- 1: N

This type of transfer is common when bridging two different modules data streams together (communications bridge).

- 1: N

This type of transfer is useful for moving information from a single data source into a memory buffer (communications receive registers).

15.8 DMA Interrupts

Each DMA has its own set of four interrupt flags, used to indicate a range of conditions during data transfers. The interrupt flag bits can be accessed using the corresponding PIR registers (Refer to the Interrupt Section).

15.8.1 DMA SOURCE COUNT INTERRUPT

The DMAxSCNTIF source count interrupt flag is set every time the DMAxSCNT[11:0] reaches zero and is reloaded to its starting value.

15.8.2 DMA DESTINATION COUNT INTERRUPT

The DMAxDCNTIF destination count interrupt flag is set every time the DMAxDCNT[11:0] reaches zero and is reloaded to its starting value.

The DMA Source Count zero and Destination Count zero interrupts are used in conjunction to determine when to signal the CPU when the DMA Messages are completed.

15.8.3 ABORT INTERRUPT

The DMAxAIF abort interrupt flag is used to signal that the DMA has halted activity due to an abort signal from one of the abort sources. This is used to indicate that the transaction has been halted for some reason.

© 2017-2020 Microchip Technology Inc.

15.8.4 OVERRUN INTERRUPT

When the DMA receives a trigger to start a new message before the current message is completed, then the DMAxORIF Overrun interrupt flag is set.

This condition indicates that the DMA is being requested before its current transaction is finished. This implies that the active DMA may not be able to keep up with the demands from the peripheral module being serviced, which may result in data loss.

The DMAxORIF flag being set does not cause the current DMA transfer to terminate.

The Overrun interrupt is only available for trigger sources that are edge based and not available for sources that are level-based. Therefore a level-based interrupt source does not trigger a DMA overrun error due to the potential latency issues in the system.

An example of an interrupt that could use the overrun interrupt would be a timer overflow (or period match) interrupt. This event only happens every time the timer rolls over and is not dependent on any other system conditions.

An example of an interrupt that does not allow the overrun interrupt would be the UARTTX buffer. The UART will continue to assert the interrupt until the DMA is able to process the MSG. Due to latency issues, the DMA may not be able to service an empty buffer immediately, but the UART continues to assert its transmit interrupt until it is serviced. If overrun was allowed in this case, the overrun would occur almost immediately as the module samples the interrupt sources every instruction cycle.

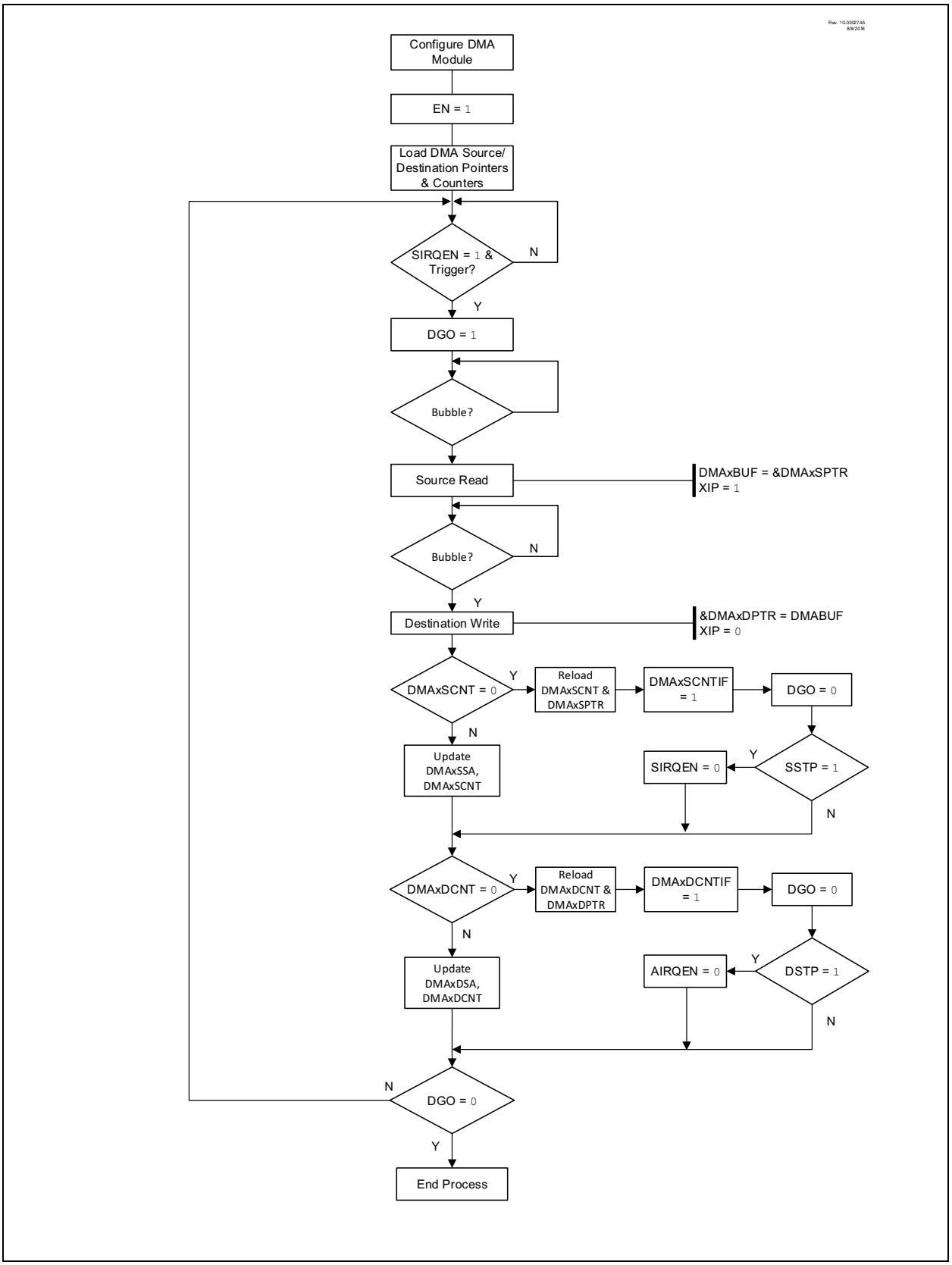
15.9 DMA Setup and Operation

The following steps illustrate how to configure the DMA for data transfer:

1. Program the appropriate Source and Destination addresses for the transaction into the DMAxSSA and DMAxDSA registers
2. Select the source memory region that is being addressed by DMAxSSA register, using the SMR[1:0] bits.
3. Program the SMODE and DMODE bits to select the addressing mode.
4. Program the Source size DMAxSSZ and Destination size DMAxDSZ registers with the number of bytes to be transferred. It is recommended for proper operation that the size registers be a multiple of each other.
5. If the user desires to disable data transfers once the message has completed, then the SSTP and DSTP bits in DMAxCON1 register need to be set. (see **Section 15.5.3.2 “Source/Destination Stop”**).
6. If using hardware triggers for data transfer, setup the hardware trigger interrupt sources for the starting and aborting DMA transfers (DMAxSIRQ and DMAxAIRQ), and set the corresponding interrupt request enable bits (SIRQEN and AIRQEN).
7. Select the priority level for the DMA (see **Section 3.1 “System Arbitration”**) and lock the priorities (see **Section 3.1.1 “Priority Lock”**)
8. Enable the DMA (DMAxCON1bits. EN = `1`)
9. If using software control for data transfer, set the DGO bit, else this bit will be set by the hardware trigger.

Once the DMA is set up, the following flow chart describes the sequence of operation when the DMA uses hardware triggers and utilizes the unused CPU cycles (bubble) for DMA transfers.

FIGURE 15-4: DMA OPERATION WITH HARDWARE TRIGGER

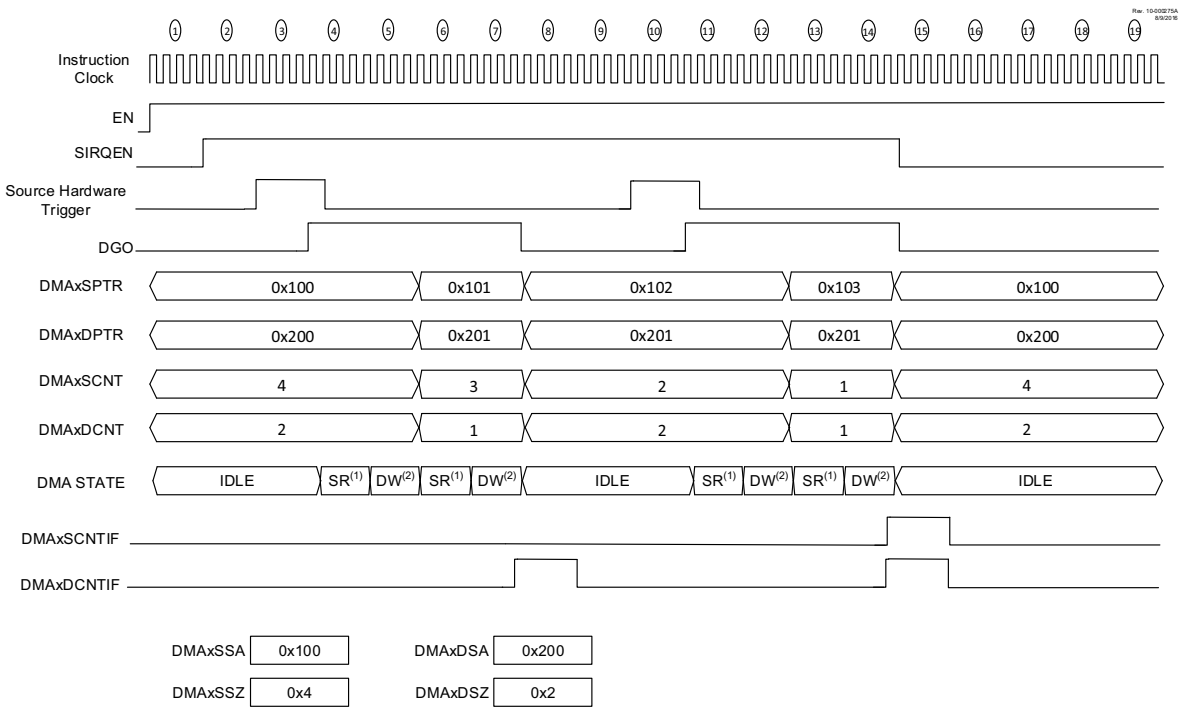


The following sections describe with visual reference the sequence of events for different configurations of the DMA module

15.9.1 SOURCE STOP

When the Source Stop bit is set (SSTP = 1) and the DMAxSCNT register reloads, the DMA clears the SIRQEN bit to stop receiving new start interrupt request signals and sets the DMAxSCNTIF flag.

FIGURE 15-5: GPR-GPR TRANSACTIONS WITH HARDWARE TRIGGERS, SSTP = 1



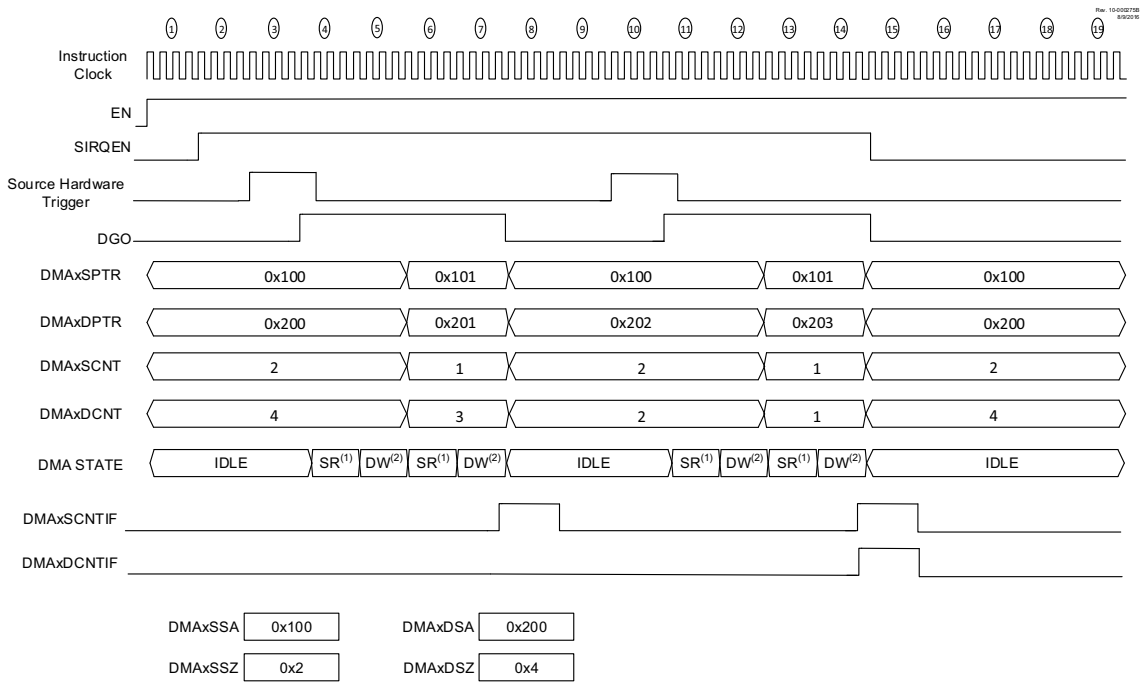
Note 1: SR - Source Read

2: DW - Destination Write

15.9.2 DESTINATION STOP

When the Destination Stop bit is set (DSTP = 1) and the DMAxDCNT register reloads, the DMA clears the SIRQEN bit to stop receiving new start interrupt request signals and sets the DMAxDCNTIF flag.

FIGURE 15-6: GPR-GPR TRANSACTIONS WITH HARDWARE TRIGGERS, DSTP = 1



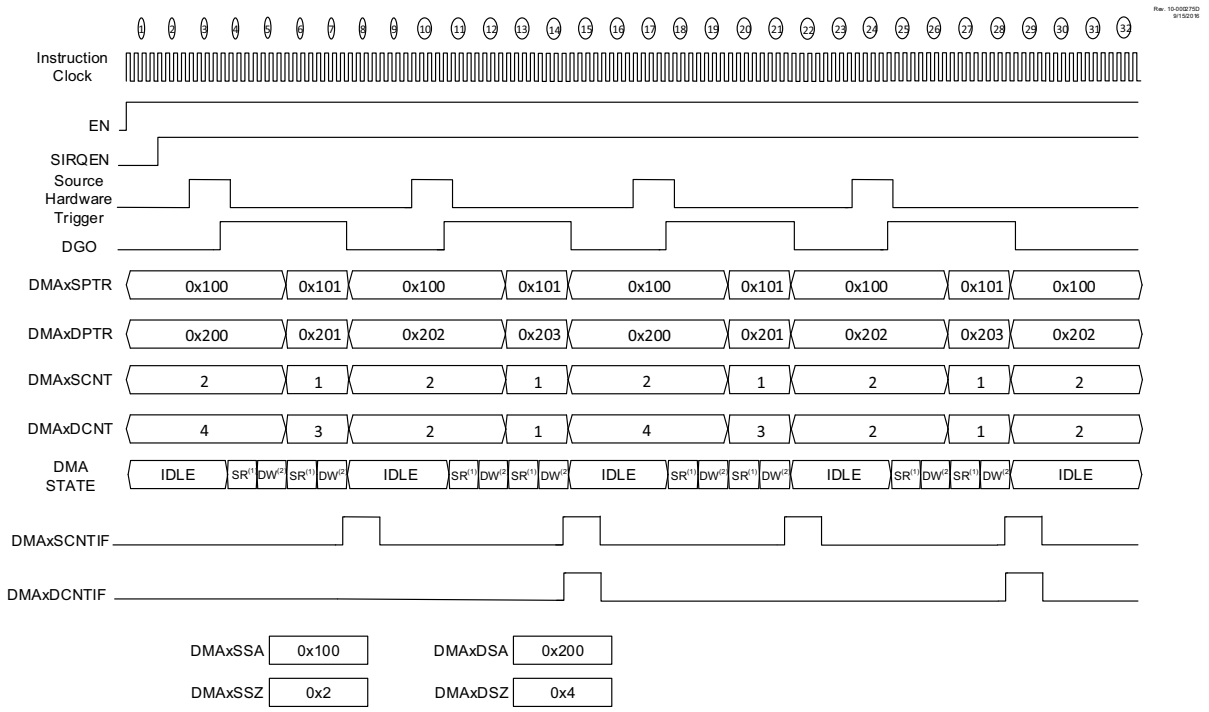
Note 1: SR - Source Read

2: DW - Destination Write

15.9.3 CONTINUOUS TRANSFER

When the Source or the Destination stop bit is cleared (SSTP, DSTP = 0), the transactions continue unless cleared by the user. The DMAxSCNTIF and DMAxDCNTIF flags are set whenever the respective counter registers are reloaded.

FIGURE 15-7: GPR-GPR TRANSACTIONS WITH HARDWARE TRIGGERS, SSTP, DSTP = 0



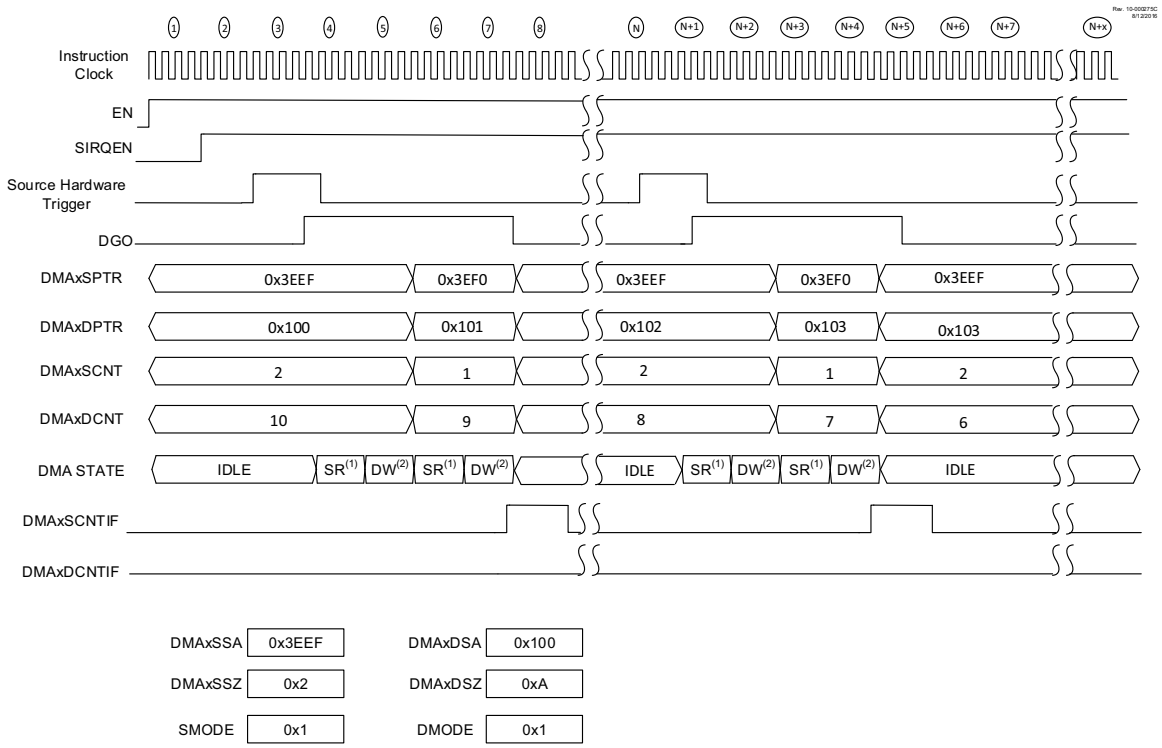
Note 1: SR - Source Read

2: DW - Destination Write

15.9.4 TRANSFER FROM SFR TO GPR

The following visual reference describes the sequence of events when copying ADC results to a GPR location. The ADC Interrupt Flag can be chosen as the Source Hardware trigger, the Source address can be set to point to the ADC Result registers at 3EEF, the Destination address can be set to point to any GPR location of our choice (Example 0x100).

FIGURE 15-8: SFR SPACE TO GPR SPACE TRANSFER



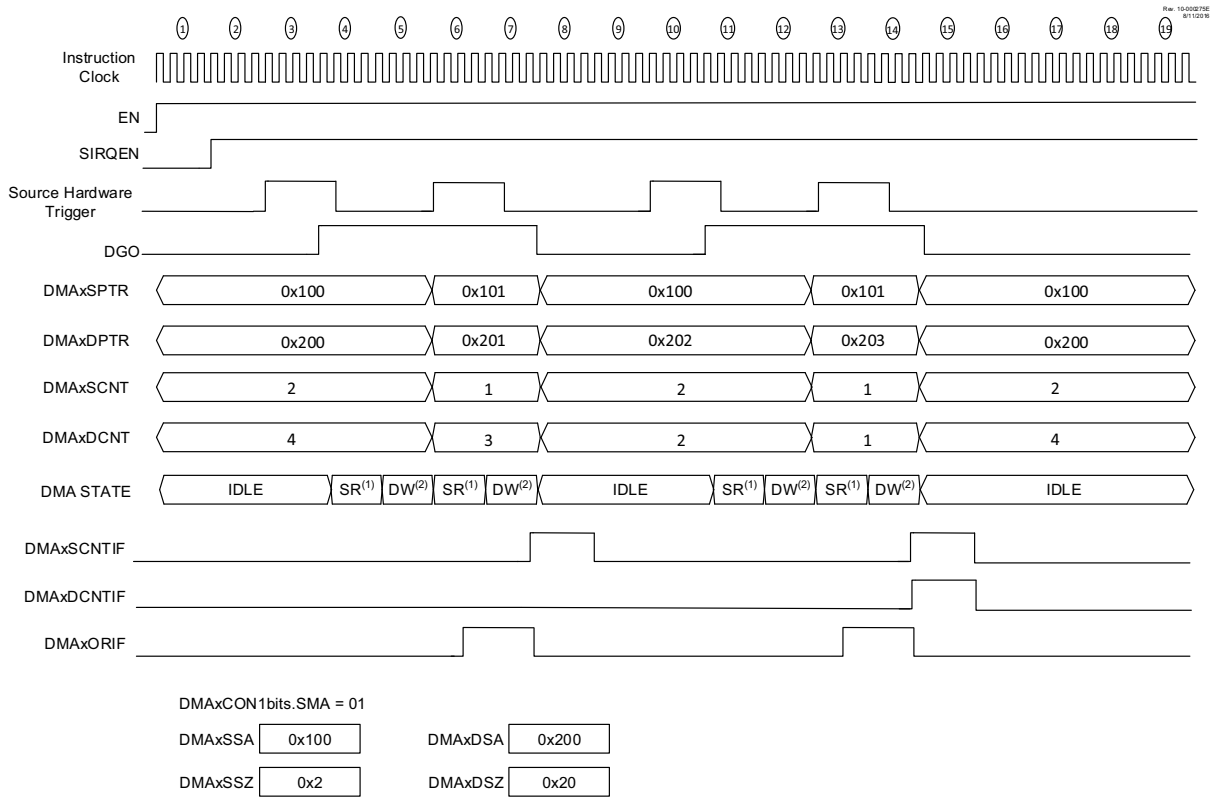
Note 1: SR - Source Read

2: DW - Destination Write

15.9.5 OVERRUN INTERRUPT

The Overrun Interrupt flag is set if the DMA receives a trigger to start a new message before the current message is completed.

FIGURE 15-9: OVERRUN INTERRUPT



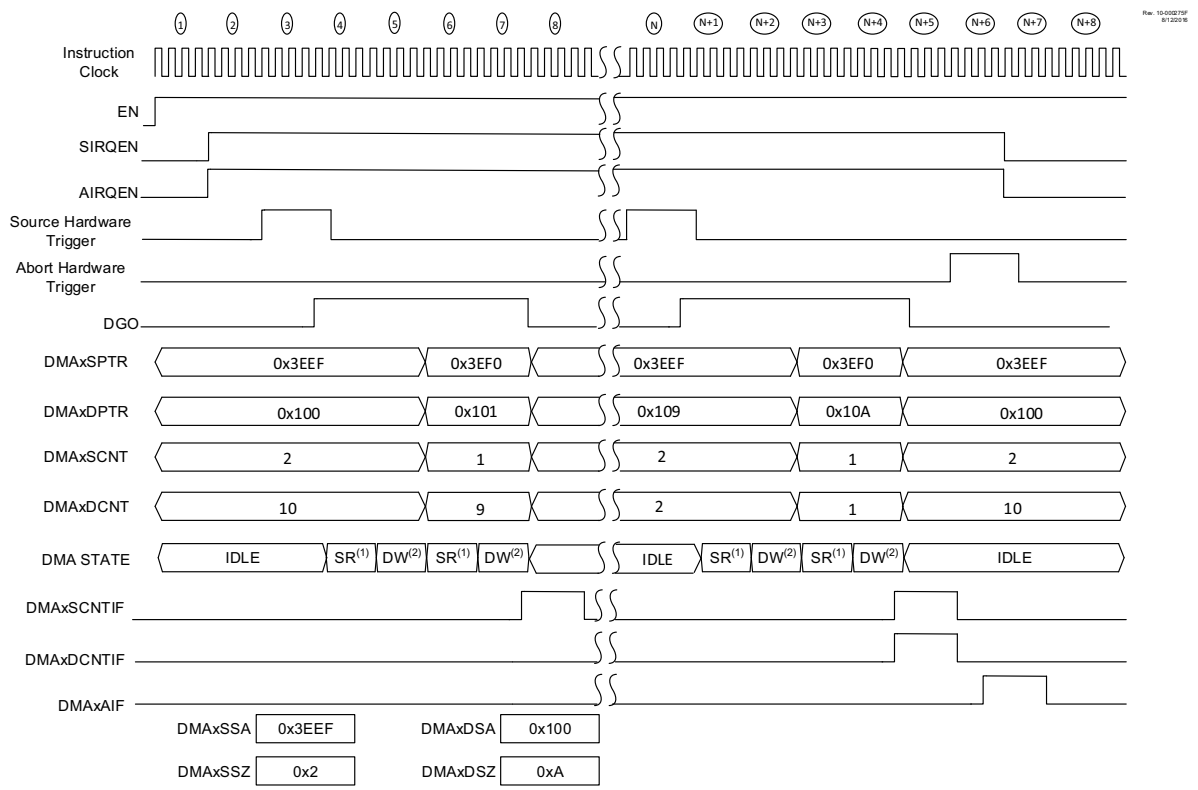
Note 1: SR - Source Read

2: DW - Destination Write

15.9.6 ABORT TRIGGER, MESSAGE COMPLETE

The AIRQEN needs to be set in order for the DMA to sample Abort Interrupt sources. When an abort interrupt is received the SIRQEN bit is cleared and the AIRQEN bit is cleared to avoid receiving further abort triggers.

FIGURE 15-10: ABORT AT THE END OF MESSAGE



Note 1: SR - Source Read

2: DW - Destination Write

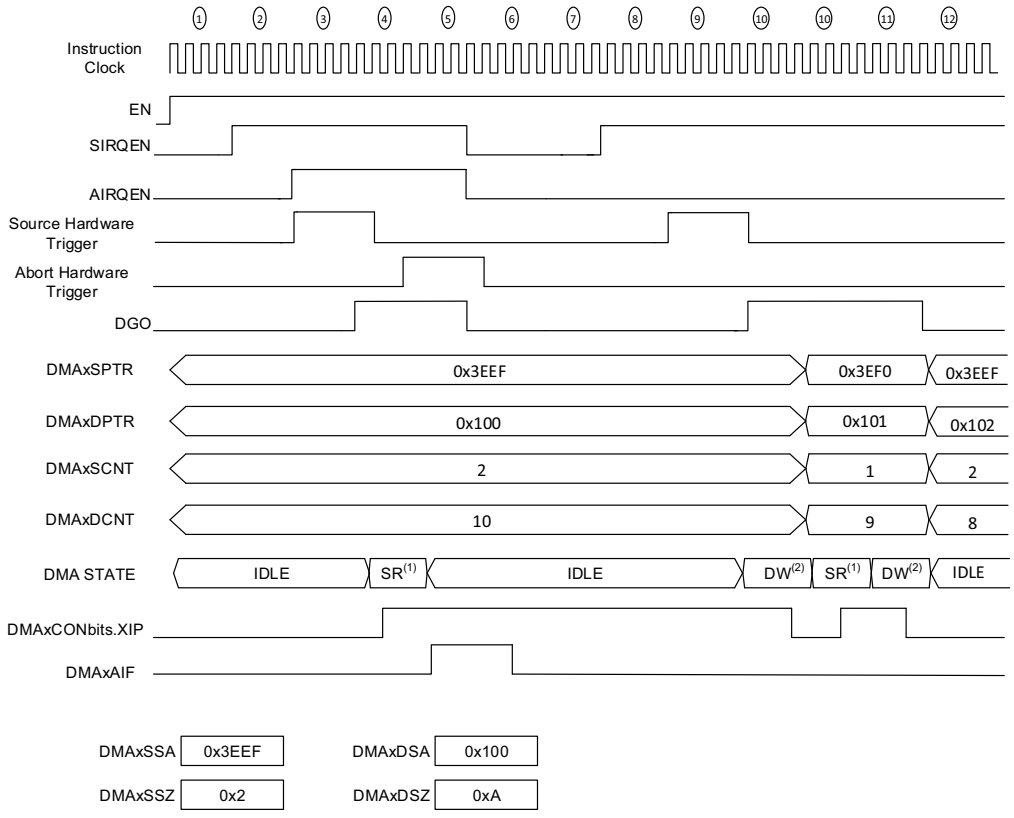
15.9.7 ABORT TRIGGER, MESSAGE IN PROGRESS

When an abort interrupt request is received in a DMA transaction, the DMA will perform a soft-stop by clearing the DGO (i.e., if the DMA was reading the source register, it will complete the read operation and then clear the DGO bit).

The SIREQEN bit is cleared to prevent any overrun and the AIRQEN bit is cleared to prevent any false aborts.

When the DGO bit is set again the DMA will resume operation from where it left off after the soft-stop.

FIGURE 15-11: ABORT DURING MESSAGE TRANSFER



Note 1: SR - Source Read

2: DW - Destination Write

The following table contains some of the cases in which the DMA module can be configured to.

TABLE 15-6: EXAMPLE DMA USE CASE TABLE

Source Module	Source Register(s)	Destination Module	Destination Register(s)	DCHxSIRQ	Comment
Signal Measurement Timer (SMT)	SMTxCPW[U:H:L]	GPR	GPR[x,y,z]	SMTxPWAIF	Store Captured Pulse-width values
	SMTxCPR[U:H:L]			SMTxPRAIF	Store Captured Period values
GPR/SFR/Program Flash/Data EEPROM	MEMORY[x,y]	TMR0	TMR0[H:L]	TMR0IF	Use as a Timer0 reload for custom 16-bit value
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	TMR0	PR0	ANY	Update TMR0 frequency based on a specific trigger
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y]	TMR1	TMR1[H:L]	TMR1IF	Use as a Timer1 reload for custom 16-bit value
TMR1	TMR1[H:L]	GPR	GPR[x,y]	TMR1GIF	Use TMR1 Gate interrupt flag to read data out of TMR1 register
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	TMR2	PR2	TMR2IF	
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y,z]	TMR2 CCP or PWM	PR2 CCPR[H:L] or PWMDC[H:L]	ANY	Frequency generator with 50% duty cycle look up table
CCP	CCPR[H:L]	GPR	GPR[x,y]	CCPxIF	Move data from CCP 16b Capture
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y]	CCP	CCPR[H:L]	ANY	Load Compare value or PWM values into the CCP
GPR/SFR/ Program Flash/Data EEPROM	MEMORY [x,y,z,u,v,w]	CCPx CCPy CCPz	CCPxR[H:L] CCPyR[H:L] CCPzR[H:L]	ANY	Update multiple PWM values at the same time e.g. 3-phase motor control
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y,z]	NCO	NCOxINC[U:H:L]	ANY	Frequency Generator look-up table
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	DAC	DACxCON0	ANY	Update DAC values
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	OSCTUNE	OSCTUNE	ANY	Automated Frequency dithering

© 2017-2020 Microchip Technology Inc.

15.10 Reset

The DMA registers are set to the default state on any Reset. The registers are also reset to the default state when the enable bit is cleared (DMA1CON1bits.EN=0).

15.11 Power Saving Mode Operation

The DMA utilizes system clocks and it is treated as a peripheral when it comes to power saving operations. Like other peripherals, the DMA also uses Peripheral Module Disable bits to further tailor its operation in low-power states.

15.11.1 SLEEP MODE

When the device enters Sleep mode, the system clock to the module is shut down, therefore no DMA operation is supported in Sleep. Once the system clock is disabled, the requisite read and write clocks are also disabled, without which the DMA cannot perform any of its tasks.

Any transfers that may be in progress are resumed on exiting from Sleep mode. Register contents are not affected by the device entering or leaving Sleep mode. It is recommended that DMA transactions be allowed to finish before entering Sleep mode.

15.11.2 IDLE MODE

In Idle mode, all of the system clocks (including the read and write clocks) are still operating but the CPU is not using them to save power.

Therefore, every instruction cycle is available to the system arbiter and if the bubble is granted to the DMA, it may be utilized to move data.

15.11.3 DOZE MODE

Similar to the Idle mode, the CPU does not utilize all of the available instruction cycles slots that are available to it in order to save power. It only executes instructions based on its settings from the Doze settings.

Therefore, every instruction not used by the CPU is available for system arbitration and may be utilized by the DMA if granted by the arbiter.

15.11.4 PERIPHERAL MODULE DISABLE

The Peripheral Module Disable (PMD) registers provide a method to disable DMA by gating all clock sources supplied to it. The respective DMAxMD bit needs to be set in order to disable the DMA.

15.12 DMA Register Interfaces

The DMA can transfer data to any GPR or SFR location. For better user accessibility, some of the more commonly used SFR spaces have their Mirror registers placed in a separate data memory location (0x4000-0x40FF). These Mirror registers can be only accessed through the DMA Source and Destination Address registers. Refer to Table 4-3 for details about these mirror registers.

EXAMPLE 15-1: SETUP DMA1 TO MOVE DATA FROM PROGRAM FLASH MEMORY TO UART1 TRANSMIT BUFFER USING HARDWARE TRIGGERS

```
//This code example illustrates using DMA1 to transfer
//10 bytes of data from 0x1000 in PFM to U1TXB 0x3DEA

void main() {
    //System Initialize
    initializeSystem();

    //Setup UART1
    initializeUART1();

    //Setup DMA1
    //DMA1CON1 - DPTR remains, Source Memory Region PFM, SPTR increments, SSTP
    DMA1CON1 = 0x0B;

    //Source registers
    //Source size
    DMA1SSZH = 0x00;
    DMA1SSZL = 0x0A;

    //Source start address, 0x1000
    DMA1SSAU = 0x00;
    DMA1SSAH = 0x10;
    DMA1SSAL = 0x00;

    //Destination registers
    //Destination size
    DMA1DSZH = 0x00;
    DMA1DSZL = 0x01;

    //Destination start address, 0x3DEA
    DMA1DSAH = 0x3D;
    DMA1DSAL = 0xEA;

    //Start trigger source U1TX
    DMA1SIRQ = 0x1C;

    //Set PRLOCKED bit to grant memory access to DMA
    INTCON0bits.GIE = 0;
    PRLOCK = 0x55;
    PRLOCK = 0xAA;
    PRLOCKbits.PRLOCKED = 1;
    INTCON0bits.GIE = 1;

    //Enable & Start DMA transfer
    DMA1CON0 = 0xC0;

    while (1) {
        doSomething();
    }
}
```

15.13 Register definitions: DMA

Long bit name prefixes for the DMA peripherals are shown in Table 15-7. Refer to **Section 1.3 "Register and Bit naming conventions"** for more information.

TABLE 15-7: REGISTER AND BIT NAMING

Peripheral	Bit Name Prefix
DMA 1	DMA1
DMA 2	DMA2

REGISTER 15-1: DMAxCON0: DMAx CONTROL REGISTER 0

R/W-0/0	R/W/HC-0/0	R/W/HS/HC-0/0	U-0	U-0	R/W/HC-0/0	U-0	R/HS/HC-0/0
EN	SIRQEN	DGO	—	—	AIRQEN	—	XIP
bit 7							bit 0

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

bit 7 **EN:** DMA Module Enable bit
- 1 = Enables module
- 0 = Disables module

bit 6 **SIRQEN:** Start of Transfer Interrupt Request Enable bits
- 1 = Hardware triggers are allowed to start DMA transfers
- 0 = Hardware triggers are not allowed to start DMA transfers

bit 5 **DGO:** DMA transaction bit
- 1 = DMA transaction is in progress
- 0 = DMA transaction is not in progress

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **AIRQEN:** Abort of Transfer Interrupt Request Enable bits
- 1 = Hardware triggers are allowed to abort DMA transfers
- 0 = Hardware triggers are not allowed to abort DMA transfers

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

bit 0 **XIP:** Transfer in Progress Status bit
- 1 = The DMAxBUF register currently holds contents from a read operation and has not transferred data to the destination.
- 0 = The DMAxBUF register is empty or has successfully transferred data to the destination address

REGISTER 15-2: DMAxCON1: DMAx CONTROL REGISTER1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
DMODE[1:0]		DSTP	SMR[1:0]		SMODE[1:0]		SSTP
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

bit 7-6 **DMODE[1:0]:** Destination Address Mode Selection bits
- 11 = Reserved, Do not use
- 10 = DMAxDPTR[15:0] is decremented after each transfer completion
- 01 = DMAxDPTR[15:0] is incremented after each transfer completion
- 00 = DMAxDPTR[15:0] remains unchanged after each transfer completion

bit 5 **DSTP:** Destination Counter Reload Stop bit
- 1 = SIRQEN bit is cleared when Destination Counter reloads
- 0 = SIRQEN bit is not cleared when Destination Counter reloads

bit 4-3 **SMR[1:0]:** Source Memory Region Select bits
- 1x = DMAxSSA[21:0] points to Data EEPROM
- 01 = DMAxSSA[21:0] points to Program Flash Memory
- 00 = DMAxSSA[21:0] points to SFR/GPR Data Space

bit 2-1 **SMODE[1:0]:** Source Address Mode Selection bits
- 11 = Reserved, Do not use
- 10 = DMAxSPTR[21:0] is decremented after each transfer completion
- 01 = DMAxSPTR[21:0] is incremented after each transfer completion
- 00 = DMAxSPTR[21:0] remains unchanged after each transfer completion

bit 0 **SSTP:** Source Counter Reload Stop bit
- 1 = SIRQEN bit is cleared when Source Counter reloads
- 0 = SIRQEN bit is not cleared when Source Counter reloads

REGISTER 15-3: DMAxBUF: DMAx DATA BUFFER REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
BUF7	BUF6	BUF5	BUF4	BUF3	BUF2	BUF1	BUF0
bit 7							bit 0

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

bit 7-0 **BUF[7:0]:** DMA Internal Data Buffer bits
DMABUF[7:0]
These bits reflect the content of the internal data buffer the DMA peripheral uses to hold the data being moved from the source to destination.

REGISTER 15-4: DMAxSSAL: DMAx SOURCE START ADDRESS LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SSA[7:0]							
bit 7							bit 0

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

bit 7-0 **SSA[7:0]:** Source Start Address bits

REGISTER 15-5: DMAxSSAH: DMAx SOURCE START ADDRESS HIGH REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SSA[15:8]							
bit 7							bit 0

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

bit 7-0 **SSA[15:8]:** Source Start Address bits

REGISTER 15-6: DMAxSSAU: DMAx SOURCE START ADDRESS UPPER REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	SSA[21:16]					
bit 7							bit 0

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

bit 7-0 **SSA[21:16]:** Source Start Address bits

REGISTER 15-7: DMAxSPTRL: DMAx SOURCE POINTER LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SPTR[7:0]							
bit 7							bit 0

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

bit 15-0 **SPTR[7:0]:** Current Source Address Pointer

REGISTER 15-8: DMAxSPTRH: DMAx SOURCE POINTER HIGH REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SPTR[15:8]							
bit 7							bit 0

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

bit 5-0 **SPTR[15:8]:** Current Source Address Pointer

REGISTER 15-9: DMAxSPTRU: DMAx SOURCE POINTER UPPER REGISTER

U-0	U-0	R-0	R-0	R-0	R-0	R-0	R-0
—	—	SPTR[21:16]					
bit 7							bit 0

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

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **SPTR[21:16]:** Current Source Address Pointer

REGISTER 15-10: DMAxSSZL: DMAx SOURCE SIZE LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SSZ[7:0]							
bit 7							bit 0

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

bit 7-0 **SSZ[7:0]:** Source Message Size bits

REGISTER 15-11: DMAxSSZH: DMAx SOURCE SIZE HIGH REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	SSZ[11:8]			
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **SSZ[11:8]:** Source Message Size bits

REGISTER 15-12: DMAxSCNTL: DMAx SOURCE COUNT LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SCNT[7:0]							
bit 7							bit 0

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

bit 7-0 **SCNT[7:0]:** Current Source Byte Count

REGISTER 15-13: DMAxSCNTH: DMAx SOURCE COUNT HIGH REGISTER

U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	SCNT[11:8]			
bit 7							bit 0

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

bit 7-4 **Unimplemented**: Read as ‘0’

bit 3-0 **SCNT[11:8]:** Current Source Byte Count

REGISTER 15-14: DMAxDSAL: DMAx DESTINATION START ADDRESS LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
DSA[7:0]							
bit 7							bit 0

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

bit 7-0 **DSA[7:0]:** Destination Start Address bits

REGISTER 15-15: DMAxDSAH: DMAx DESTINATION START ADDRESS HIGH REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
DSA[15:8]							
bit 7							bit 0

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

bit 7-0 **DSA[15:8]:** Destination Start Address bits

REGISTER 15-16: DMAxDPTRL: DMAx DESTINATION POINTER LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DPTR[7:0]							
bit 7							bit 0

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

bit 7-0 **DPTR[7:0]:** Current Destination Address Pointer

REGISTER 15-17: DMAxDPTRH: DMAx DESTINATION POINTER HIGH REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DPTR[15:8]							
bit 7							bit 0

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

bit 7-0 **DPTR[15:8]:** Current Destination Address Pointer

REGISTER 15-18: DMAxDSZL: DMAx DESTINATION SIZE LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
DSZ[7:0]							
bit 7							bit 0

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

bit 7-0 **DSZ[7:0]:** Destination Message Size bits

REGISTER 15-19: DMAxDSZH: DMAx DESTINATION SIZE HIGH REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	DSZ[11:8]			
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **DSZ[11:8]:** Destination Message Size bits

REGISTER 15-20: DMAxDCNTL: DMAx DESTINATION COUNT LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DCNT[7:0]							
bit 7							bit 0

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

bit 7-0 **DCNT[7:0]:** Current Destination Byte Count

REGISTER 15-21: DMAxDCNTH: DMAx DESTINATION COUNT HIGH REGISTER

U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	DCNT[11:8]			
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **DCNT[11:8]:** Current Destination Byte Count

REGISTER 15-22: DMAxSIRQ: DMAx START INTERRUPT REQUEST SOURCE SELECTION REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	SIRQ[6:0]						
bit 7							bit 0

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

bit 7 **Unimplemented**: Read as ‘0’

bit 6-0 **SIRQ[6:0]:** DMAx Start Interrupt Request Source Selection bits
Please refer to Table 15-2 for more information.

REGISTER 15-23: DMAxAIRQ: DMAx ABORT INTERRUPT REQUEST SOURCE SELECTION REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	AIRQ[6:0]						
bit 7							bit 0

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

bit 7 **Unimplemented**: Read as ‘0’

bit 6-0 **AIRQ[6:0]:** DMAx Interrupt Request Source Selection bits
Please refer to Table 15-2 for more information.

TABLE 15-2: DMAxSIRQ AND DMAxAIRQ TRIGGER SOURCES

DMAxSIRQ DMAxAIRQ	Trigger Source(2)	Level Triggered(1)
0x00	Reserved	
0x01	HLVDIF	No
0x02	OSFIF	No
0x03	CSWIF	No
0x04	NVMIF	No
0x05	SCANIF	No
0x06	CRCIF	No
0x07	IOCIF	Yes
0x08	INT0IF	No
0x09	ZCDIF	No
0x0A	ADIF	No
0x0B	ADTIF	No
0x0C	CMP1IF	No
0x0D	SMT1IF	No
0x0E	SMT1PRAIF	No
0x0F	SMT1PWAIF	No
0x10	DMA1SCNTIF	No
0x11	DMA1DCNTIF	No
0x12	DMA1ORIF	No
0x13	DMA1AIF	No
0x14	SPI1RXIF	Yes
0x15	SPI1TXIF	Yes
0x16	SPI1IF	Yes
0x17	I2C1RXIF	Yes
0x18	I2C1TXIF	Yes
0x19	I2C1IF	Yes
0x1A	I2C1EIF	Yes
0x1B	U1RXIF	Yes
0x1C	U1TXIF	Yes
0x1D	U1EIF	Yes
0x1E	U1IF	No
0x1F	TMR0IF	No
0x20	TMR1IF	No
0x21	TMR1GIF	No
0x22	TMR2IF	No
0x23	CCP1IF	No
0x24	Reserved	
0x25	NCOIF	No
0x26	CWG1IF	No
0x27	CLC1IF	No
0x28	INT1IF	No
0x29	CMP2IF	No
0x2A	DMA2SCNTIF	No
0x2B	DMA2DCNTIF	No
0x2C	DMA2ORIF	No
0x2D	DMA2AIF	No
0x2E	I2C2RXIF	Yes
0x2F	I2C2TXIF	Yes
0x30	I2C2IF	Yes
0x31	I2C2EIF	Yes
0x32	U2RXIF	Yes
0x33	U2TXIF	Yes
0x34	U2EIF	Yes
0x35	U2IF	No
0x36	TMR3IF	No
0x37	TMR3GIF	No
0x38	TMR4IF	No
0x39	CCP2IF	No
0x3A	Reserved	
0x3B	CWG2IF	No
0x3C	CLC2IF	No
0x3D	INT2IF	No
0x3E	Reserved	
0x3F	Reserved	
0x40	Reserved	
0x41	Reserved	
0x42	Reserved	
0x43	Reserved	
0x44	Reserved	
0x45	Reserved	
0x46	TMR5IF	No
0x47	TMR5GIF	No
0x48	TMR6IF	No
0x49	CCP3IF	No
0x4A	CWG3IF	No
0x4B	CLC3IF	No
0x4C	Reserved	
0x4D	Reserved	
0x4E	Reserved	
0x4F	Reserved	
0x50	CCP4IF	No
0x51	CLC4IF	No
0x52 – 0xFF	Reserved	

Note 1: All trigger sources that are not Level-triggered are Edge-triggered.

2: The event that sets the flag is the interrupt trigger, not the flag itself. The flag remains set.

TABLE 15-3: SUMMARY OF REGISTERS ASSOCIATED WITH DMA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
DMAxCON0	EN	SIRQEN	DGO	—	—	AIRQEN	—	XIP	249
DMAxCON1	DMODE[1:0]		DSTP	SMR[1:0]		SMODE[1:0]		SSTP	250
DMAxBUF	DBUF7	DBUF6	DBUF5	DBUF4	DBUF3	DBUF2	DBUF1	DBUF0	251
DMAxSSAL	SSA[7:0]								251
DMAxSSAH	SSA[15:8]								251
DMAxSSAU	—	—	SSA[21:16]						252
DMAxSPTRL	SPTR[7:0]								252
DMAxSPTRH	SPTR[15:8]								252
DMAxSPTRU	—	—	SPTR[21:16]						253
DMAxSSZL	SSZ[7:0]								253
DMAxSSZH	—	—	—	—	SSZ[11:8]				253
DMAxSCNTL	SCNT[7:0]								254
DMAxSCNTH	—	—	—	—	SCNT[11:8]				254
DMAxDSAL	DSA[7:0]								254
DMAxDSAH	DSA[15:8]								255
DMAxDPTRL	DPTR[7:0]								255
DMAxDPTRH	DPTR[15:8]								255
DMAxDSZL	DSZ[7:0]								256
DMAxDSZH	—	—	—	—	DSZ[11:8]				256
DMAxDCNTL	DCNT[7:0]								256
DMAxDCNTH	—	—	—	—	DCNT[11:8]				257
DMAxSIRQ	—	SIRQ[6:0]							257
DMAxAIRQ	—	AIRQ[6:0]							257

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by DMA.

16.0 I/O PORTS

The PIC18(L)F26/27/45/46/47/55/56/57K42 devices have six I/O ports, allocated as shown in Table 16-1.

TABLE 16-1: PORT ALLOCATION TABLE FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

Device	PORTA	PORTB	PORTC	PORTD	PORTE	PORTF
PIC18(L)F26K42	•	•	•		•(1)	
PIC18(L)F27K42	•	•	•		•(1)	
PIC18(L)F45K42	•	•	•	•	•(2)	
PIC18(L)F46K42	•	•	•	•	•(2)	
PIC18(L)F47K42	•	•	•	•	•(2)	
PIC18(L)F55K42	•	•	•	•	•(2)	•
PIC18(L)F56K42	•	•	•	•	•(2)	•
PIC18(L)F57K42	•	•	•	•	•(2)	•

Note 1: Pin RE3 only.
2: Pins RE0, RE1, RE2 and RE3 only.

Each port has ten registers to control the operation. These registers are:

- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)
- TRISx registers (data direction)
- ANSELx registers (analog select)
- WPUx registers (weak pull-up)
- INLVLx (input level control)
- SLRCONx registers (slew rate control)
- ODCONx registers (open-drain control)

Most port pins share functions with device peripherals, both analog and digital. In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output; however, the pin can still be read.

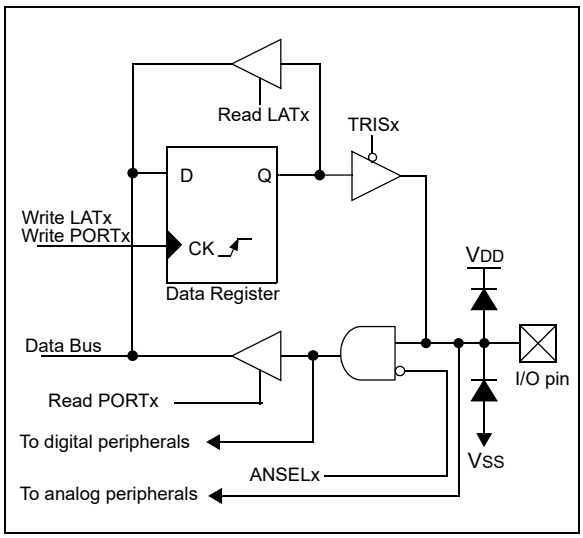
The Data Latch (LATx registers) is useful for read-modify-write operations on the value that the I/O pins are driving.

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSELx bit is set, the digital input buffer associated with that bit is disabled.

Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 16-1.

FIGURE 16-1: GENERIC I/O PORT OPERATION



16.1 I/O Priorities

Each pin defaults to the PORT data latch after Reset. Other functions are selected with the peripheral pin select logic. See **Section 17.0 "Peripheral Pin Select (PPS) Module"** for more information.

Analog input functions, such as ADC and comparator inputs, are not shown in the peripheral pin select lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx register. Digital output functions may continue to control the pin when it is in Analog mode.

Analog outputs, when enabled, take priority over digital outputs and force the digital output driver into a high-impedance state.

The pin function priorities are as follows:

1. Configuration bits
2. Analog outputs (disable the input buffers)
3. Analog inputs
4. Port inputs and outputs from PPS

16.2 PORTx Registers

In this section, the generic names such as PORTx, LATx, TRISx, etc. can be associated with PORTA, PORTB, and PORTC. The functionality of PORTE is different compared to other ports and is explained in a separate section.

16.2.1 DATA REGISTER

PORTx is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISx (Register 16-2). Setting a TRISx bit ('1') will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISx bit ('0') will make the corresponding PORTx pin an output (i.e., it enables output driver and puts the contents of the output latch on the selected pin). Example 16-1 shows how to initialize PORTx.

Reading the PORTx register (Register 16-1) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATx).

The PORT data latch LATx (Register 16-3) holds the output port data and contains the latest value of a LATx or PORTx write.

EXAMPLE 16-1: INITIALIZING PORTA

```
; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL  PORTA        ;
CLRF     PORTA        ;Init PORTA
BANKSEL  LATA         ;Data Latch
CLRF     LATA         ;
BANKSEL  ANSELA       ;
CLRF     ANSELA       ;digital I/O
BANKSEL  TRISA        ;
MOVLW    B'11111000'  ;Set RA[7:3] as inputs
MOVWF    TRISA        ;and set RA[2:0] as
                      ;outputs
```

16.2.2 DIRECTION CONTROL

The TRISx register (Register 16-2) controls the PORTx pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISx register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

16.2.3 ANALOG CONTROL

The ANSELx register (Register 16-4) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELx bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELx bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

Note:	The ANSELx bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.

16.2.4 OPEN-DRAIN CONTROL

The ODCONx register (Register 16-6) controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When an ODCONx bit is set, the corresponding port output becomes an open-drain driver capable of sinking current only. When an ODCONx bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

Note:	It is necessary to set open-drain control when using the pin for I^2C.

16.2.5 SLEW RATE CONTROL

The SLRCONx register (Register 16-7) controls the slew rate option for each port pin. Slew rate for each port pin can be controlled independently. When an SLRCONx bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONx bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

16.2.6 INPUT THRESHOLD CONTROL

The INLVLx register (Register 16-8) controls the input voltage threshold for each of the available PORTx input pins. A selection between the Schmitt Trigger CMOS or the TTL compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTx register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See Table 44-5 for more information on threshold levels.

Note:	Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

16.2.7 WEAK PULL-UP CONTROL

The WPUx register (Register 16-5) controls the individual weak pull-ups for each port pin.

16.2.8 EDGE SELECTABLE INTERRUPT-ON-CHANGE

An interrupt can be generated by detecting a signal at the port pin that has either a rising edge or a falling edge. Any individual pin can be configured to generate an interrupt. The interrupt-on-change module is present on all the pins. For further details about the IOC module refer to **Section 18.0 "Interrupt-on-Change"**.

16.2.9 I^2C PAD CONTROL

For the PIC18(L)F26/27/45/46/47/55/56/57K42 devices, the I^2C specific pads are available on RB1, RB2, RC3, RC4, RD0**(1)** and RD1**(1)** pins. The I^2C characteristics of each of these pins is controlled by the RxyI2C registers (see Register 16-9). These characteristics include enabling I^2C specific slew rate (over standard GPIO slew rate), selecting internal pull-ups for I^2C pins, and selecting appropriate input threshold as per I^2C/SMBus specifications.

Note 1:	RD0 and RD1 I^2C pads are not available in PIC18(L)F26K42 parts.
2:	Any peripheral using the I^2C pins read the I^2C ST inputs when enabled via RxyI2C.

.

16.3 PORTE Registers

Depending on the device, PORTE is implemented in two different ways.

16.3.1 PORTE ON 40/44/48-PIN DEVICES

For PIC18(L)F45/46/47/55/56/57K42 devices, PORTE is a 4-bit wide port. Three pins (RE0, RE1 and RE2) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as '0's. The corresponding data direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., disable the output driver).
Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). TRISE controls the direction of the REx pins, even when they are being used as analog pins. The user must make sure to keep the pins configured as inputs when using them as analog inputs. RE[2:0] bits have other registers associated with them (i.e., ANSELE, WPUE, INLVLE, SLRCONE and ODCONE). The functionality is similar to the other ports. The Data Latch register (LATE) is also memory-mapped. Read-modify-write operations on the LATE register read and write the latched output value for PORTE.

Note:	On a Power-on Reset, RE[2:0] are configured as analog inputs.

The fourth pin of PORTE (MCLR/VPP/RE3) is an input-only pin. Its operation is controlled by the MCLRE Configuration bit. When selected as a port pin, (MCLRE = 0), it functions as a digital input-only pin; as such, it does not have TRIS or LAT bits associated with its operation. Otherwise, it functions as the device's Master Clear input. In either configuration, RE3 also functions as the programming voltage input during programming. RE3 in PORTE register is a read-only bit and will read '1' when MCLRE = 1 (i.e., Master Clear enabled).

Note:	On a Power-on Reset, RE3 is enabled as a digital input only if Master Clear functionality is disabled.

EXAMPLE 16-2: INITIALIZING PORTE

```
CLRF    PORTE       ;Initialize PORTE by
                    ;clearing output
                    ;data latches
CLRF    LATE        ;Alternate method
                    ;to clear output
                    ;data latches
CLRF    ANSELE      ;Configure analog pins
                    ;for digital only
MOVLW   05h         ;Value used to
                    ;initialize data
                    ;direction
MOVWF   TRISE       ;Set RE[0] as input
                    ;RE[1] as output
                    ;RE[2] as input
```

16.3.2 PORTE ON 28-PIN DEVICES

For PIC18(L)F26/27K42 devices, PORTE is only available when Master Clear functionality is disabled (MCLRE = 0). In this case, PORTE is a single bit, input-only port comprised of RE3 only. The pin operates as previously described. RE3 in PORTE register is a read-only bit and will read '1' when MCLRE = 1 (i.e., Master Clear enabled).

16.3.3 RE3 WEAK PULL-UP

The port RE3 pin has an individually controlled weak internal pull-up. When set, the WPUE3 bit enables the RE3 pin pull-up. When the RE3 port pin is configured as $\overline{\text{MCLR}}$, (CONFIG2L, MCLRE = 1 and CONFIG4H, LVP = 0), or configured for Low-Voltage Programming, (MCLRE = x and LVP = 1), the pull-up is always enabled and the WPUE3 bit has no effect.

16.3.4 INTERRUPT-ON-CHANGE

The interrupt-on-change feature is available only on the RE3 pin of PORTE for all devices. If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and interrupt-on-change on RE3 is not available. For further details refer to **Section 18.0 "Interrupt-on-Change"**.

16.4 Register Definitions: Port Control

REGISTER 16-1: PORTx: PORTx REGISTER(1)

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
Rx7	Rx6	Rx5	Rx4	Rx3	Rx2	Rx1	Rx0
bit 7							bit 0

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

bit 7-0 **Rx[7:0]:** Rx7:Rx0 Port I/O Value bits
1 = Port pin is ≥ V_{IH}
0 = Port pin is ≤ V_{IL}

Note 1: Writes to PORTx are actually written to the corresponding LATx register. Reads from PORTx register return actual I/O pin values.

TABLE 16-2: PORT REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
PORTB	RB7(1)	RB6(1)	RB5	RB4	RB3	RB2	RB1	RB0
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
PORTD(3)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
PORTE	—	—	—	—	RE3(2)	RE2(3)	RE1(3)	RE0(3)
PORTF(4)	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0

Note 1: Bits RB6 and RB7 read ‘1’ while in Debug mode.
2: Bit PORTE3 is read-only, and will read ‘1’ when MCLRE = 1 (Master Clear enabled).
3: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F26/27/45/46/47K42.

REGISTER 16-2: TRISx: TRI-STATE CONTROL REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
TRISx7	TRISx6	TRISx5	TRISx4	TRISx3	TRISx2	TRISx1	TRISx0
bit 7							bit 0

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

bit 7-0 **TRISx[7:0]:** TRISx Port I/O Tri-state Control bits
`1` = Port output driver is disabled
`0` = Port output driver is enabled

TABLE 16-3: TRIS REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
TRISB	TRISB7(1)	TRISB6(1)	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
TRISD(2)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
TRISE(2)	—	—	—	—	—	TRISE2	TRISE1	TRISE0
TRISF(3)	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0

Note 1: Bits RB6 and RB7 read ‘`1`’ while in Debug mode.
2: Unimplemented in PIC18(L)F26/27K42.
3: Unimplemented in PIC18(L)F26/45/46/47K42.

REGISTER 16-3: LATx: LATx REGISTER(1)

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
LATx7	LATx6	LATx5	LATx4	LATx3	LATx2	LATx1	LATx0
bit 7							bit 0

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

bit 7-0 **LATx[7:0]:** Rx7:Rx0 Output Latch Value bits

Note 1: Writes to LATx are equivalent with writes to the corresponding PORTx register. Reads from LATx register return register values, not I/O pin values.

TABLE 16-4: LAT REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LATA	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
LATD(1)	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0
LATE(1)	—	—	—	—	—	LATE2	LATE1	LATE0
LATF(2)	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0

Note 1: Unimplemented in PIC18(L)F26/27K42.
2: Unimplemented in PIC18(L)F26/45/46/47K42.

REGISTER 16-4: ANSELx: ANALOG SELECT REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
ANSELx7	ANSELx6	ANSELx5	ANSELx4	ANSELx3	ANSELx2	ANSELx1	ANSELx0
bit 7							bit 0

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

bit 7-0 **ANSELx[7:0]**: Analog Select on Pins Rx[7:0]
1 = Digital Input buffers are disabled.
0 = ST and TTL input devices are enabled

TABLE 16-5: ANALOG SELECT PORT REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ANSELA	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
ANSELB	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
ANSELC	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
ANSELD(1)	ANSELD7	ANSELD6	ANSELD5	ANSELD4	ANSELD3	ANSELD2	ANSELD1	ANSELD0
ANSELE(1)	—	—	—	—	—	ANSELE2	ANSELE1	ANSELE0
ANSELF(2)	ANSELF7	ANSELF6	ANSELF5	ANSELF4	ANSELF3	ANSELF2	ANSELF1	ANSELF0

Note 1: Unimplemented in PIC18(L)F26/27K42.
2: Unimplemented in PIC18(L)F26/45/46/47K42.

REGISTER 16-5: WPUx: WEAK PULL-UP REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
WPUx7	WPUx6	WPUx5	WPUx4	WPUx3	WPUx2	WPUx1	WPUx0
bit 7							bit 0

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

bit 7-0 **WPUx[7:0]:** Weak Pull-up PORTx Control bits
1 = Weak Pull-up enabled
0 = Weak Pull-up disabled

TABLE 16-6: WEAK PULL-UP PORT REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
WPUD(2)	WPUD7	WPUD6	WPUD5	WPUD4	WPUD3	WPUD2	WPUD1	WPUD0
WPUE	—	—	—	—	WPUE3(1)	WPUE2(2)	WPUE1(2)	WPUE0(2)
WPUF(3)	WPUF7	WPUF6	WPUF5	WPUF4	WPUF3	WPUF2	WPUF1	WPUF0

Note 1: If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.
2: Unimplemented in PIC18(L)F26/27K42.
3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

REGISTER 16-6: ODCONx: OPEN-DRAIN CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ODCx7	ODCx6	ODCx5	ODCx4	ODCx3	ODCx2	ODCx1	ODCx0
bit 7							bit 0

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

bit 7-0 **ODCx[7:0]**: Open-Drain Configuration on Pins Rx[7:0]
`1` = Output drives only low-going signals (sink current only)
`0` = Output drives both high-going and low-going signals (source and sink current)

TABLE 16-7: OPEN-DRAIN CONTROL REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ODCONA	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0
ODCONB	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0
ODCONC	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
ODCOND(1)	ODCD7	ODCD6	ODCD5	ODCD4	ODCD3	ODCD2	ODCD1	ODCD0
ODCONE(1)	—	—	—	—	—	ODCE2	ODCE1	ODCE0
ODCONF(2)	ODCF7	ODCF6	ODCF5	ODCF4	ODCF3	ODCF2	ODCF1	ODCF0

Note 1: Unimplemented in PIC18(L)F26/27K42.
2: Unimplemented in PIC18(L)F26/27/45/46/47K42.

REGISTER 16-7: SLRCONx: SLEW RATE CONTROL REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
SLRx7	SLRx6	SLRx5	SLRx4	SLRx3	SLRx2	SLRx1	SLRx0
bit 7							bit 0

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

bit 7-0 **SLRx[7:0]**: Slew Rate Control on Pins Rx[7:0], respectively
`1` = Port pin slew rate is limited
`0` = Port pin slews at maximum rate

TABLE 16-8: SLEW RATE CONTROL REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SLRCONA	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0
SLRCONB	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0
SLRCONC	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
SLRCOND(1)	SLRD7	SLRD6	SLRD5	SLRD4	SLRD3	SLRD2	SLRD1	SLRD0
SLRCONE(1)	—	—	—	—	—	SLRE2	SLRE1	SLRE0
SLRCONF(2)	SLRF7	SLRF6	SLRF5	SLRF4	SLRF3	SLRF2	SLRF1	SLRF0

Note 1: Unimplemented in PIC18(L)F26/27K42.
2: Unimplemented in PIC18(L)F26/27/45/46/47K42.

REGISTER 16-8: INLVLx: INPUT LEVEL CONTROL REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
INLVLx7	INLVLx6	INLVLx5	INLVLx4	INLVLx3	INLVLx2	INLVLx1	INLVLx0
bit 7							bit 0

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

bit 7-0 **INLVLx[7:0]**: Input Level Select on Pins Rx[7:0], respectively
`1` = ST input used for port reads and interrupt-on-change
`0` = TTL input used for port reads and interrupt-on-change

TABLE 16-9: INPUT LEVEL PORT REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INLVLA	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
INLVLB	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2(1)	INLVLB1(1)	INLVLB0
INLVLC	INLVLC7	INLVLC6	INLVLC5	INLVLC4(1)	INLVLC3(1)	INLVLC2	INLVLC1	INLVLC0
INLVLD(2)	INLVLD7	INLVLD6	INLVLD5	INLVLD4	INLVLD3	INLVLD2	INLVLD1(1)	INLVLD0(1)
INLVLE	—	—	—	—	INLVLE3	INLVLE2(2)	INLVLE1(2)	INLVLE0(2)
INLVLF(3)	INLVLF7	INLVLF6	INLVLF5	INLVLF4	INLVLF3	INLVLF2	INLVLF1	INLVLF0

Note 1: Any peripheral using the I^2C pins read the I^2C ST inputs when enabled via RxyI2C.
2: Unimplemented in PIC18(L)F26/27K42.
3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

REGISTER 16-9: RxyI2C: I²C PAD Rxy CONTROL REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	SLEW	PU[1:0]		—	—	TH[1:0]	
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	HS = Hardware set

bit 7 **Unimplemented**: Read as ‘0’

bit 6 **SLEW**: I²C specific slew rate limiting is enabled
1 = I²C specific slew rate limiting is enabled. Standard pad slew limiting is disabled. The SLRxy bit is ignored.
0 = Standard GPIO Slew Rate; enabled/disabled via SLRxy bit.

bit 5-4 **PU[1:0]**: I²C Pull-up Selection bits
11 = Reserved
10 = 10x current of standard weak pull-up
01 = 2x current of standard weak pull-up
00 = Standard GPIO weak pull-up, enabled via WPUxy bit

bit 3-2 **Unimplemented**: Read as ‘0’

bit 1-0 **TH[1:0]**: I²C Input Threshold Selection bits
11 = SMBus 3.0 (1.35 V) input threshold
10 = SMBus 2.0 (2.1 V) input threshold
01 = I²C specific input thresholds
00 = Standard GPIO Input pull-up, enabled via INLVLxy registers

TABLE 16-10: I2C PAD CONTROL REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RB1I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]	
RB2I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]	
RC3I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]	
RC4I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]	
RD0I2C(1)	—	SLEW	PU[1:0]		—	—	TH[1:0]	
RD1I2C(1)	—	SLEW	PU[1:0]		—	—	TH[1:0]	

Note 1: Unimplemented in PIC18(L)F26/27K42.

TABLE 16-11: SUMMARY OF REGISTERS ASSOCIATED WITH I/O

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	264
PORTB	RB7(1)	RB6(1)	RB5	RB4	RB3	RB2	RB1	RB0	264
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	264
PORTD(6)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	264
PORTE	—	—	—	—	RE3(2)	RE2(6)	RE1(6)	RE0(6)	264
PORTF(7)	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	264
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	265
TRISB	TRISB7(3)	TRISB6(3)	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	265
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	265
TRISD(6)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	265
TRISE(6)	—	—	—	—	—	TRISE2	TRISE1	TRISE0	265
TRISF(7)	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	265
LATA	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	266
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	266
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	266
LATD(6)	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	266
LATE(6)	—	—	—	—	—	LATE2	LATE1	LATE0	266
LATF(7)	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	266
ANSELA	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0	267
ANSELB	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0	267
ANSELC	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0	267
ANSELD(6)	ANSELD7	ANSELD6	ANSELD5	ANSELD4	ANSELD3	ANSELD2	ANSELD1	ANSELD0	267
ANSELE(6)	—	—	—	—	—	ANSELE2	ANSELE1	ANSELE0	267
ANSELF(7)	ANSELF7	ANSELF6	ANSELF5	ANSELF4	ANSELF3	ANSELF2	ANSELF1	ANSELF0	267
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	268
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	268
WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0	268
WPUD(6)	WPUD7	WPUD6	WPUD5	WPUD4	WPUD3	WPUD2	WPUD1	WPUD0	268
WPUE	—	—	—	—	WPUE3(4)	WPUE2(6)	WPUE1(6)	WPUE0(6)	268
WPUF(6)	WPUF7	WPUF6	WPUF5	WPUF4	WPUF3	WPUF2	WPUF1	WPUF0	268
ODCONA	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0	269
ODCONB	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0	269
ODCONC	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0	269
ODCOND(6)	ODCD7	ODCD6	ODCD5	ODCD4	ODCD3	ODCD2	ODCD1	ODCD0	269
ODCONE(6)	—	—	—	—	—	ODCE2	ODCE1	ODCE0	269
ODCONF(7)	ODCF7	ODCF6	ODCF5	ODCF4	ODCF3	ODCF2	ODCF1	ODCF0	269
SLRCONA	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0	270
SLRCONB	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0	270
SLRCONC	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0	270
SLRCOND(6)	SLRD7	SLRD6	SLRD5	SLRD4	SLRD3	SLRD2	SLRD1	SLRD0	270
SLRCONE(6)	—	—	—	—	—	SLRE2	SLRE1	SLRE0	270
SLRCONF(7)	SLRF7	SLRF6	SLRF5	SLRF4	SLRF3	SLRF2	SLRF1	SLRF0	270
INLVLA	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0	271
INLVLB	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2(5)	INLVLB1(5)	INLVLB0	271

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by I/O Ports.

Note 1: Bits RB6 and RB7 read '1' while in Debug mode.
2: Bit PORTE3 is read-only, and will read '1' when MCLRE = 1 (Master Clear enabled).
3: Bits RB6 and RB7 read '1' while in Debug mode.
4: If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.
5: Any peripheral using the I^2C pins read the I^2C ST inputs when enabled via RxyI2C.
6: Unimplemented in PIC18(L)F26/27K42.
7: Unimplemented in PIC18(L)F26/27/45/46/47K42 parts.

TABLE 16-11: SUMMARY OF REGISTERS ASSOCIATED WITH I/O (CONTINUED)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INLVLC	INLVLC7	INLVLC6	INLVLC5	INLVLC4(5)	INLVLC3(5)	INLVLC2	INLVLC1	INLVLC0	271
INLVLD(6)	INLVLD7	INLVLD6	INLVLD5	INLVLD4	INLVLD3	INLVLD2	INLVLD1(5)	INLVLD0(5)	271
INLVLF(7)	INLVLF7	INLVLF6	INLVLF5	INLVLF4	INLVLF3	INLVLF2	INLVLF1	INLVLF0	271
INLVLE	—	—	—	—	INLVLE3	—	—	—	271
RB1I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]		272
RB2I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]		272
RC3I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]		272
RC4I2C	—	SLEW	PU[1:0]		—	—	TH[1:0]		272
RD0I2C(6)	—	SLEW	PU[1:0]		—	—	TH[1:0]		272
RD1I2C(6)	—	SLEW	PU[1:0]		—	—	TH[1:0]		272

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by I/O Ports.

Note 1: Bits RB6 and RB7 read '1' while in Debug mode.
2: Bit PORTE3 is read-only, and will read '1' when MCLRE = 1 (Master Clear enabled).
3: Bits RB6 and RB7 read '1' while in Debug mode.
4: If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.
5: Any peripheral using the I^2C pins read the I^2C ST inputs when enabled via RxyI2C.
6: Unimplemented in PIC18(L)F26/27K42.
7: Unimplemented in PIC18(L)F26/27/45/46/47K42 parts.

17.0 PERIPHERAL PIN SELECT (PPS) MODULE

The Peripheral Pin Select (PPS) module connects peripheral inputs and outputs to the device I/O pins. Only digital signals are included in the selections. All analog inputs and outputs remain fixed to their assigned pins. Input and output selections are independent as shown in the simplified block diagram Figure 17-1.

The peripheral input is selected with the peripheral xxxPPS register (Register 17-1), and the peripheral output is selected with the PORT RxyPPS register (Register 17-2). For example, to select PORTC[7] as the UART1 RX input, set U1RXPPS to `0b1 0111`, and to select PORTC[6] as the UART1 TX output set RC6PPS to `0b01 0011`.

17.1 PPS Inputs

Each peripheral has a PPS register with which the inputs to the peripheral are selected. Inputs include the device pins.

Multiple peripherals can operate from the same source simultaneously. Port reads always return the pin level regardless of peripheral PPS selection. If a pin also has analog functions associated, the ANSEL bit for that pin must be cleared to enable the digital input buffer.

Although every peripheral has its own PPS input selection register, the selections are identical for every peripheral as shown in Register 17-1.

Note:	The notation “xxx” in the register name is a place holder for the peripheral identifier. For example, INT0PPS.

17.2 PPS Outputs

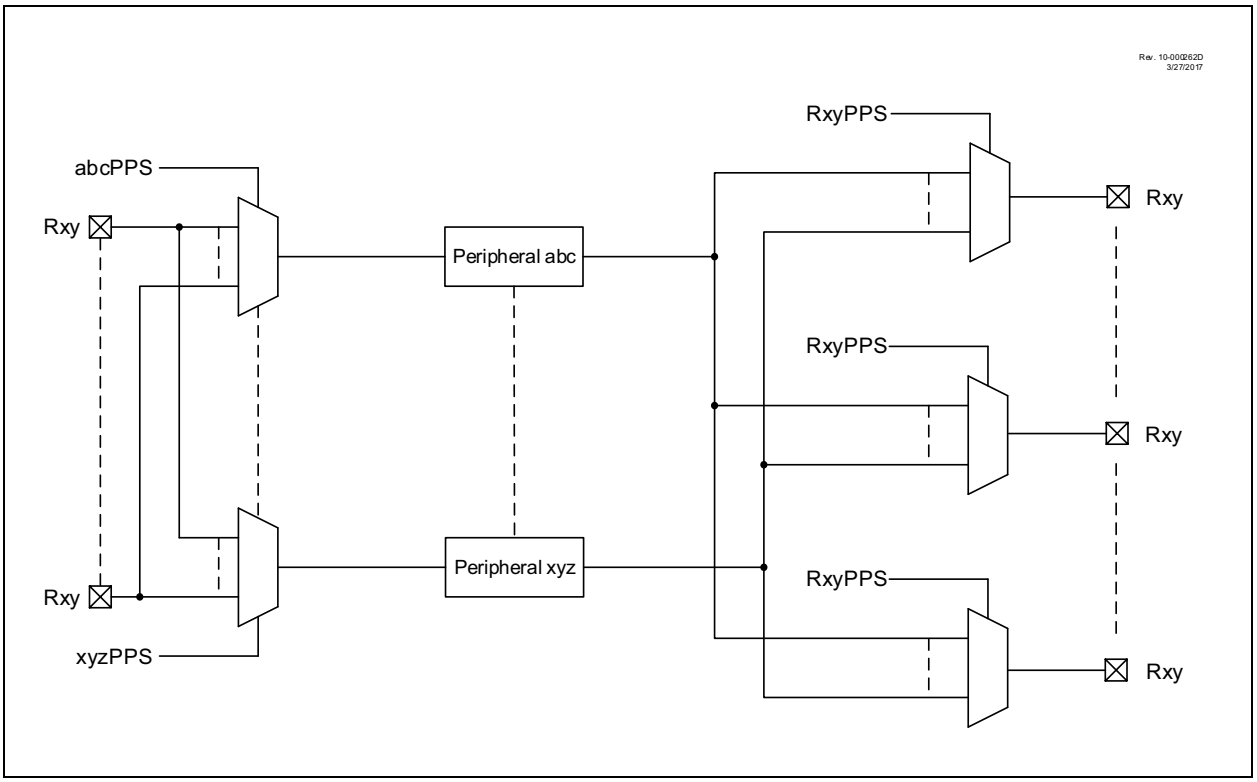
Each I/O pin has a PPS register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that control the pin output driver as part of the peripheral operation will override the TRIS control as needed. These peripherals include:

- UART I^2C

Although every pin has its own PPS peripheral selection register, the selections are identical for every pin as shown in Register 17-2.

Note:	The notation “Rxy” is a place holder for the pin identifier. For example, RA0PPS.

FIGURE 17-1: SIMPLIFIED PPS BLOCK DIAGRAM



17.3 Bidirectional Pins

PPS selections for peripherals with bidirectional signals on a single pin must be made so that the PPS input and PPS output select the same pin. Peripherals that have bidirectional signals include:

- I^2C

Note: Refer to Table 17-1 for pins that are I^2C compatible. Clock and data signals can be routed to any pin, however pins without I^2C compatibility will operate at standard TTL/ST logic levels as selected by the INVLV register.

17.4 PPS Lock

The PPS includes a mode in which all input and output selections can be locked to prevent inadvertent changes. PPS selections are locked by setting the PPSLOCKED bit of the PPSLOCK register. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. Examples of setting and clearing the PPSLOCKED bit are shown in Example 17-1.

EXAMPLE 17-1: PPS LOCK SEQUENCE

```
; Disable interrupts:
    BCF      INTCON0,GIE

; Bank to PPSLOCK register
    BANKSEL  PPSLOCK
    MOVLB    PPSLOCK
    MOVLW    55h

; Required sequence, next 4 instructions
    MOVWF    PPSLOCK
    MOVLW    AAh
    MOVWF    PPSLOCK

; Set PPSLOCKED bit to disable writes
; Only a BSF instruction will work
    BSF      PPSLOCK,0

; Enable Interrupts
    BSF      INTCON0,GIE
```

EXAMPLE 17-2: PPS UNLOCK SEQUENCE

```
; Disable interrupts:
    BCF      INTCON0,GIE

; Bank to PPSLOCK register
    BANKSEL  PPSLOCK
    MOVLB    PPSLOCK
    MOVLW    55h

; Required sequence, next 4 instructions
    MOVWF    PPSLOCK
    MOVLW    AAh
    MOVWF    PPSLOCK

; Clear PPSLOCKED bit to enable writes
; Only a BCF instruction will work
    BCF      PPSLOCK,0

; Enable Interrupts
    BSF      INTCON0,GIE
```

17.5 PPS One-way Lock

When the PPS1WAY Configuration bit is set, the PPSLOCKED bit can only be cleared and set one time after a device Reset. This allows for clearing the PPSLOCKED bit so that the input and output selections can be made during initialization. When the PPSLOCKED bit is set after all selections have been made, it will remain set and cannot be cleared until after the next device Reset event.

17.6 Operation During Sleep

PPS input and output selections are unaffected by Sleep.

17.7 Effects of a Reset

A device Power-on-Reset (POR) clears all PPS input and output selections to their default values. All other Resets leave the selections unchanged. Default input selections are shown in pin allocation Table 1. The PPS one-way lock is also removed.

17.8 Register Definitions: PPS Input Selection

REGISTER 17-1: xxxPPS: PERIPHERAL xxx INPUT SELECTION

U-0	U-0	R/W-m/u(1,3)	R/W-m/u(1)	R/W-m/u(1)	R/W-m/u(1)	R/W-m/u(1)	R/W-m/u(1)
—	—	xxxPPS[5:0]					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	-n/n = Value at POR and BOR/Value at all other Resets
u = Bit is unchanged	x = Bit is unknown	q = value depends on peripheral
‘1’ = Bit is set	U = Unimplemented bit, read as ‘0’	m = value depends on default location for that input
‘0’ = Bit is cleared		

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-3 **xxxPPS[5:3]:** Peripheral xxx Input PORTx Pin Selection bits
See Table 17-1 for the list of available ports and default pin locations.
101 = PORTF(2)
100 = PORTE(3)
011 = PORTD(3)
010 = PORTC
001 = PORTB
000 = PORTA

bit 2-0 **xxxPPS[2:0]:** Peripheral xxx Input PORTx Pin Selection bits
111 = Peripheral input is from PORTx Pin 7 (Rx7)
110 = Peripheral input is from PORTx Pin 6 (Rx6)
101 = Peripheral input is from PORTx Pin 5 (Rx5)
100 = Peripheral input is from PORTx Pin 4 (Rx4)
011 = Peripheral input is from PORTx Pin 3 (Rx3)
010 = Peripheral input is from PORTx Pin 2 (Rx2)
001 = Peripheral input is from PORTx Pin 1 (Rx1)
000 = Peripheral input is from PORTx Pin 0 (Rx0)

Note 1: The Reset value ‘m’ of this register is determined by device default locations for that input.
2: Reserved on PIC18LF26/27/45/46/47K42 parts.
3: Reserved on PIC18LF26K42 parts.

TABLE 17-1: PPS INPUT REGISTER DETAILS

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	Input Available from Selected PORTx													
				PIC18(L)F26/27K42			PIC18(L)F45/46/47K42					PIC18(L)F55/56/57K42					
Interrupt 0	INT0PPS	RB0	0b0 1000	A	B	—	A	B	—	—	—	A	B	—	—	—	—
Interrupt 1	INT1PPS	RB1	0b0 1001	A	B	—	A	B	—	—	—	—	B	—	D	—	—
Interrupt 2	INT2PPS	RB2	0b0 1010	A	B	—	A	B	—	—	—	—	B	—	—	—	F
Timer0 Clock	T0CKIPPS	RA4	0b0 0100	A	B	—	A	B	—	—	—	A	—	—	—	—	F
Timer1 Clock	T1CKIPPS	RC0	0b1 0000	A	—	C	A	—	C	—	—	—	—	C	—	E	—
Timer1 Gate	T1GPPS	RB5	0b0 1101	—	B	C	—	B	C	—	—	—	B	C	—	—	—
Timer3 Clock	T3CKIPPS	RC0	0b1 0000	—	B	C	—	B	C	—	—	-	—	C	—	E	—
Timer3 Gate	T3GPPS	RC0	0b1 0000	A	—	C	A	—	C	—	—	A	—	C	—	—	—
Timer5 Clock	T5CKIPPS	RC2	0b1 0010	A	—	C	A	—	C	—	—	—	—	C	—	E	—
Timer5 Gate	T5GPPS	RB4	0b0 1100	—	B	C	—	B	—	D	—	—	B	—	D	—	—
Timer2 Clock	T2INPPS	RC3	0b1 0011	A	—	C	A	—	C	—	—	A	—	C	—	—	—
Timer4 Clock	T4INPPS	RC5	0b1 0101	—	B	C	—	B	C	—	—	—	B	C	—	—	—
Timer6 Clock	T6INPPS	RB7	0b0 1111	—	B	C	—	B	—	D	—	—	B	—	D	—	—
CCP1	CCP1PPS	RC2	0b1 0010	—	B	C	—	B	C	—	—	—	—	C	—	—	F
CCP2	CCP2PPS	RC1	0b1 0001	—	B	C	—	B	C	—	—	—	—	C	—	—	F
CCP3	CCP3PPS	RB5	0b0 1101	—	B	C	—	B	—	D	—	—	B	—	D	—	—
CCP4	CCP4PPS	RB0	0b0 1000	—	B	C	—	B	—	D	—	—	B	—	D	—	—
SMT1 Window	SMT1WINPPS	RC0	0b1 0000	—	B	C	—	B	C	—	—	—	—	C	—	—	F
SMT1 Signal	SMT1SIGPPS	RC1	0b1 0001	—	B	C	—	B	C	—	—	—	—	C	—	—	F
CWG1	CWG1PPS	RB0	0b0 1000	—	B	C	—	B	—	D	—	—	B	—	D	—	—
CWG2	CWG2PPS	RB1	0b0 1001	—	B	C	—	B	—	D	—	—	B	—	D	—	—
CWG3	CWG3PPS	RB2	0b0 1010	—	B	C	—	B	—	D	—	—	B	—	D	—	—
DSM1 Carrier Low	MD1CARLPPS	RA3	0b0 0011	A	—	C	A	—	—	D	—	A	—	—	D	—	—
DSM1 Carrier High	MD1CARHPPS	RA4	0b0 0100	A	—	C	A	—	—	D	—	A	—	—	D	—	—
DSM1 Source	MD1SRCPPS	RA5	0b0 0101	A	—	C	A	—	—	D	—	A	—	—	D	—	—
CLCx Input 1	CLCIN0PPS	RA0	0b0 0000	A	—	C	A	—	C	—	—	A	—	C	—	—	—
CLCx Input 2	CLCIN1PPS	RA1	0b0 0001	A	—	C	A	—	C	—	—	A	—	C	—	—	—
CLCx Input 3	CLCIN2PPS	RB6	0b0 1110	—	B	C	—	B	—	D	—	—	B	—	D	—	—
CLCx Input 4	CLCIN3PPS	RB7	0b0 1111	—	B	C	—	B	—	D	—	—	B	—	D	—	—

© 2017-2020 Microchip Technology Inc.

TABLE 17-1: PPS INPUT REGISTER DETAILS

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	Input Available from Selected PORTx													
				PIC18(L)F26/27K42			PIC18(L)F45/46/47K42					PIC18(L)F55/56/57K42					
ADC Conversion Trigger	ADACTPPS	RB4	0b0 1100	—	B	C	—	B	—	D	—	—	B	—	D	—	—
SPI1 Clock	SPI1SCKPPS	RC3	0b1 0011	—	B	C	—	B	C	—	—	—	B	C	—	—	—
SPI1 Data	SPI1SDIPPS	RC4	0b1 0100	—	B	C	—	B	C	—	—	—	B	C	—	—	—
SPI1 Slave Select	SPI1SSPPS	RA5	0b0 0101	A	—	C	A	—	—	D	—	A	—	—	D	—	—
I^2C1 Clock	I2C1SCLPPS	RC3	0b1 0011	—	B	C	—	B	C	—	—	—	B	C	—	—	—
I^2C1 Data	I2C1SDAPPS	RC4	0b1 0100	—	B	C	—	B	C	—	—	—	B	C	—	—	—
I^2C2 Clock	I2C2SCLPPS	RB1	0b0 1001	—	B	C	—	B	—	D	—	—	B	—	D	—	—
I^2C2 Data	I2C2SDAPPS	RB2	0b0 1010	—	B	C	—	B	—	D	—	—	B	—	D	—	—
UART1 Receive	U1RXPPS	RC7	0b1 0111	—	B	C	—	B	C	—	—	—	—	C	—	—	F
UART1 Clear To Send	U1CTSPPS	RC6	0b1 0110	—	B	C	—	B	C	—	—	—	—	C	—	—	F
UART2 Receive	U2RXPPS	RB7	0b0 1111	—	B	C	—	B	—	D	—	—	B	—	D	—	—
UART2 Clear To Send	U2CTSPPS	RB6	0b0 1110	—	B	C	—	B	—	D	—	—	B	—	D	—	—

© 2017-2020 Microchip Technology Inc.

REGISTER 17-2: RxyPPS: PIN Rxy OUTPUT SOURCE SELECTION REGISTER

U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	RxyPPS[5:0]					
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 **Unimplemented:** Read as ‘0’

bit 5-0 **RxyPPS[5:0]:** Pin Rxy Output Source Selection bits
See Table 17-2 for the list of available ports.

TABLE 17-2: PPS OUTPUT REGISTER DETAILS

RxyPPS[5:0]	Pin Rxy Output Source	Device Configuration													
		PIC18(L)F26K42			PIC18(L)F45/46/47K42					PIC18(L)F55/56/57K42					
0b11 1111 – 0b11 0011	Reserved														
0b11 0010	ADGRDB	A	—	C	A	—	C	—	—	A	—	—	—	—	F
0b11 0001	ADGRDA	A	—	C	A	—	C	—	—	A	—	—	—	—	F
0b11 0000	CWG3D	A	—	C	A	—	—	D	—	A	—	—	D	—	—
0b10 1111	CWG3C	A	—	C	A	—	—	D	—	A	—	—	D	—	—
0b10 1110	CWG3B	A	—	C	A	—	—	—	E	A	—	—	—	E	—
0b10 1101	CWG3A	—	B	C	—	B	C	—	—	—	B	C	—	—	—
0b10 1100	CWG2D	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b10 1011	CWG2C	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b10 1010	CWG2B	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b10 1001	CWG2A	—	B	C	—	B	C	—	—	—	B	C	—	—	—
0b10 1000	DSM1	A	—	C	A	—	—	D	—	A	—	—	D	-	—
0b10 0111	CLKR	—	B	C	—	B	C	—	—	—	B	—	—	E	—
0b10 0110	NCO1	A	—	C	A	—	—	D	—	A	—	—	D	—	—
0b10 0101	TMR0	—	B	C	—	B	C	—	—	—	—	C	—	—	F
0b10 0100	I^2C2 (SDA)	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b10 0011	I^2C2 (SCL)	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b10 0010	I^2C1 (SDA)	—	B	C	—	B	C	—	—	—	B	C	—	—	—
0b10 0001	I^2C1 (SCL)	—	B	C	—	B	C	—	—	—	B	C	—	—	—
0b10 0000	SPI1 (SS)	A	—	C	A	—	—	D	—	A	—	—	D	—	—
0b01 1111	SPI1 (SDO)	—	B	C	—	B	C	—	—	—	B	C	—	—	—
0b01 1110	SPI1 (SCK)	—	B	C	—	B	C	—	—	—	B	C	—	—	—
0b01 1101	C2OUT	A	—	C	A	—	—	—	E	A	—	—	—	E	—
0b01 1100	C1OUT	A	—	C	A	—	—	D	—	A	—	—	D	—	—
0b01 1011 – 0b01 1001	Reserved														
0b01 1000	UART2 ($\overline{RTS}$)	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b01 0111	UART2 (TXDE)	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b01 0110	UART2 (TX)	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b01 0101	UART1 ($\overline{RTS}$)	—	B	C	—	B	C	—	—	—	—	C	—	—	F
0b01 0100	UART1 (TXDE)	—	B	C	—	B	C	—	—	—	—	C	—	—	F
0b01 0011	UART1 (TX)	—	B	C	—	B	C	—	—	—	—	C	—	—	F

© 2017-2020 Microchip Technology Inc.

TABLE 17-2: PPS OUTPUT REGISTER DETAILS

RxyPPS[5:0]	Pin Rxy Output Source	Device Configuration													
		PIC18(L)F26K42			PIC18(L)F45/46/47K42					PIC18(L)F55/56/57K42					
0b01 0010 – 0b01 0001	Reserved														
0b01 0000	PWM8	A	—	C	A	—	—	D	—	A	—	—	D	—	—
0b00 1111	PWM7	A	—	C	A	—	C	—	—	—	—	C	—	—	F
0b00 1110	PWM6	A	—	C	A	—	—	D	—	A	—	—	D	—	—
0b00 1101	PWM5	A	—	C	A	—	C	—	—	A	—	—	—	—	F
0b00 1100	CCP4	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b00 1011	CCP3	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b00 1010	CCP2	—	B	C	—	B	C	—	—	—	—	C	—	—	F
0b00 1001	CCP1	—	B	C	—	B	C	—	—	—	—	C	—	—	F
0b00 1000	CWG1D	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b00 0111	CWG1C	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b00 0110	CWG1B	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b00 0101	CWG1A	—	B	C	—	B	C	—	—	—	B	C	—	—	—
0b00 0100	CLC4OUT	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b00 0011	CLC3OUT	—	B	C	—	B	—	D	—	—	B	—	D	—	—
0b00 0010	CLC2OUT	A	—	C	A	—	C	—	—	A	—	—	—	—	F
0b00 0001	CLC1OUT	A	—	C	A	—	C	—	—	A	—	—	—	—	F
0b00 0000	LATxy	A	B	C	A	B	C	D	E	A	B	C	D	E	F

© 2017-2020 Microchip Technology Inc.

REGISTER 17-3: PPSLOCK: PPS LOCK REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	PPSLOCKED
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 7-1 **Unimplemented:** Read as ‘0’

bit 0 **PPSLOCKED:** PPS Locked bit
1 = PPS is locked.
0 = PPS is not locked. PPS selections can be changed.

TABLE 17-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE PPS MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
PPSLOCK	—	—	—	—	—	—	—	PPSLOCKED	284
INT0PPS	—	—	INT0PPS[5:0]						278
INT1PPS	—	—	INT1PPS[5:0]						278
INT2PPS	—	—	INT2PPS[5:0]						278
T0CKIPPS	—	—	T0CKPPS[5:0]						278
T1CKIPPS	—	—	T1CKPPS[5:0]						278
T1GPPS	—	—	T1GPPS[5:0]						278
T3CKIPPS	—	—	T3CKIPPS[5:0]						278
T3GPPS	—	—	T3GPPS[5:0]						278
T5CKIPPS	—	—	T5CKPPS[5:0]						278
T5GPPS	—	—	T5GPPS[5:0]						278
T2INPPS	—	—	T2INPPS[5:0]						278
T4INPPS	—	—	T4INPPS[5:0]						278
T6INPPS	—	—	T6INPPS[5:0]						278
CCP1PPS	—	—	CCP1PPS[5:0]						278
CCP2PPS	—	—	CCP2PPS[5:0]						278
CCP3PPS	—	—	CCP3PPS[5:0]						278
CCP4PPS	—	—	CCP4PPS[5:0]						278
SMT1WINPPS	—	—	SMT1WINPPS[5:0]						278
SMT1SIGPPS	—	—	SMT1SIGPPS[5:0]						278
CWG1PPS	—	—	CWG1PPS[5:0]						278
CWG2PPS	—	—	CWG2PPS[5:0]						278
CWG3PPS	—	—	CWG3PPS[5:0]						278
MD1CARLPPS	—	—	MD1CARLPPS[5:0]						278
MD1CARHPPS	—	—	MD1CARHPPS[5:0]						278
MD1SRCPPS	—	—	MD1SRCPPS[5:0]						278
CLCIN0PPS	—	—	CLCIN0PPS[5:0]						278
CLCIN1PPS	—	—	CLCIN1PPS[5:0]						278
CLCIN2PPS	—	—	CLCIN2PPS[5:0]						278
CLCIN3PPS	—	—	CLCIN3PPS[5:0]						278
ADACTPPS	—	—	ADACTPPS[5:0]						278
SPI1SCKPPS	—	—	SPI1SCKPPS[5:0]						278
SPI1SDIPPS	—	—	SPI1SDIPPS[5:0]						278
SPI1SSPPS	—	—	SPI1SSPPS[5:0]						278
I2C1SCLPPS	—	—	I2C1SCLPPS[5:0]						278
I2C1SDAPPS	—	—	I2C1SDAPPS[5:0]						278
I2C2SCLPPS	—	—	I2C2SCLPPS[5:0]						278
I2C2SDAPPS	—	—	I2C2SDAPPS[5:0]						278
U1RXPPS	—	—	U1RXPPS[5:0]						278
U1CTSPPS	—	—	U1CTSPPS[5:0]						278
U2RXPPS	—	—	U2RXPPS[5:0]						278
U2CTSPPS	—	—	U2CTSPPS[5:0]						278
RxyPPS	—	—	RxyPPS[5:0]						281

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the PPS module.

18.0 INTERRUPT-ON-CHANGE

PORTA, PORTB, PORTC and pin RE3 of PORTE can be configured to operate as Interrupt-on-Change (IOC) pins on PIC18(L)F26/27/45/46/47/55/56/57K42 family devices. 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 18-1 is a block diagram of the IOC module.

18.1 Enabling the Module

To allow individual port pins to generate an interrupt, the IOCIE bit of the PIEx 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.

18.2 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.

18.3 Interrupt Flags

The IOCAFx, IOCBFx, IOCCFx and IOCEF3 bits located in the IOCAF, IOCBF, IOCCF and IOCEF 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 PIR0 register reflects the status of all IOCAFx, IOCBFx, IOCCFx and IOCEF3 bits.

18.4 Clearing Interrupt Flags

The individual status flags, (IOCAFx, IOCBFx, IOCCFx and IOCEF3 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 18-1: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)

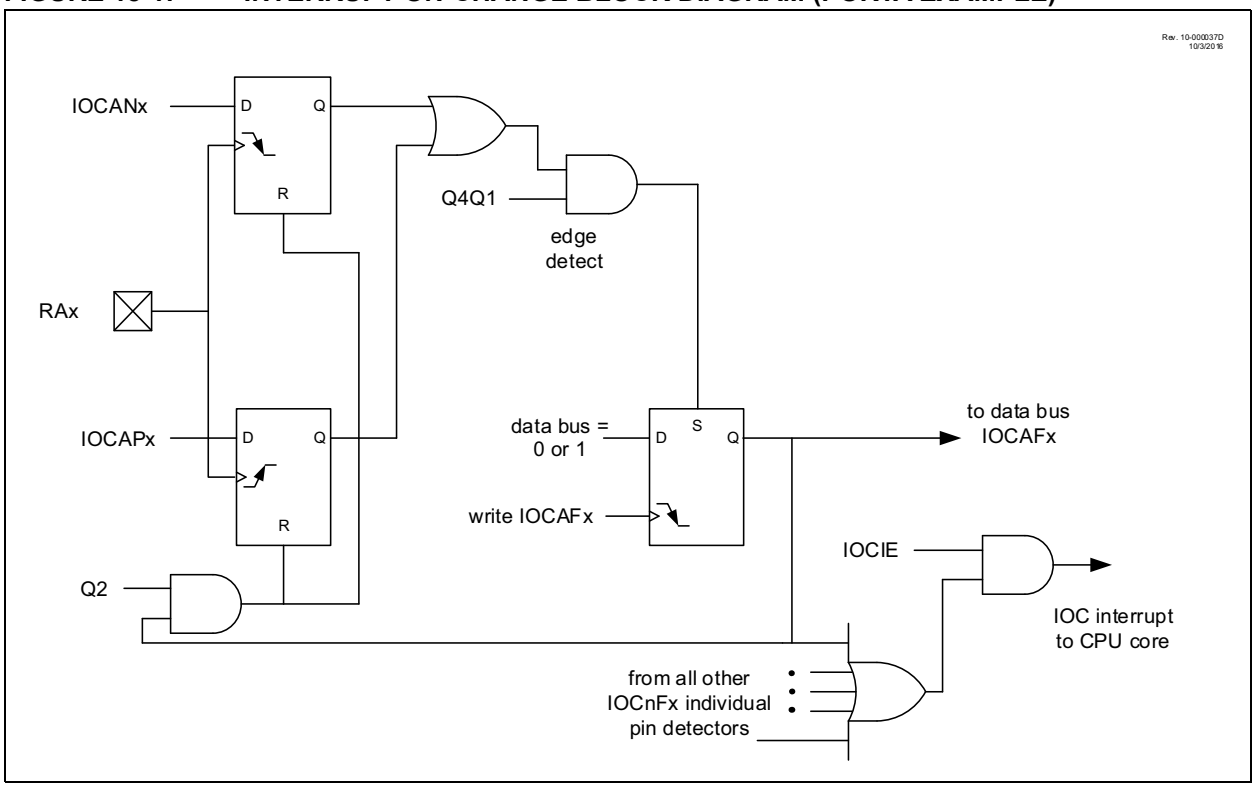
```
MOVLW   0xff
XORWF   IOCAF, W
ANDWF   IOCAF, F
```

18.5 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.

FIGURE 18-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)



18.6 Register Definitions: Interrupt-on-Change Control

REGISTER 18-1: IOCxP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER EXAMPLE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
IOCxP7	IOCxP6	IOCxP5	IOCxP4	IOCxP3	IOCxP2	IOCxP1	IOCxP0
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 7-0 **IOCxP[7:0]:** Interrupt-on-Change Positive Edge Enable bits
1 = Interrupt-on-Change enabled on the IOCx pin for a positive-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin.

REGISTER 18-2: IOCxN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER EXAMPLE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
IOCxN7	IOCxN6	IOCxN5	IOCxN4	IOCxN3	IOCxN2	IOCxN1	IOCxN0
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 7-0 **IOCxN[7:0]:** Interrupt-on-Change Negative Edge Enable bits
1 = Interrupt-on-Change enabled on the IOCx pin for a negative-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin

REGISTER 18-3: IOCxF: INTERRUPT-ON-CHANGE FLAG REGISTER EXAMPLE

R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
IOCxF7	IOCxF6	IOCxF5	IOCxF4	IOCxF3	IOCxF2	IOCxF1	IOCxF0
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	HS - Bit is set in hardware

bit 7-0 **IOCxF[7:0]:** Interrupt-on-Change Flag bits
1 = A enabled change was detected on the associated pin. Set when IOCP[n] = 1 and a positive edge was detected on the IOCn pin, or when IOCN[n] = 1 and a negative edge was detected on the IOCn pin
0 = No change was detected, or the user cleared the detected change

TABLE 18-1: IOC REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IOCAP	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
IOCAN	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
IOCAF	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
IOCBP	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
IOCBN	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
IOCBF	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
IOCCP	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
IOCCN	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
IOCCF	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
IOCEP	—	—	—	—	IOCEP3(1)	—	—	—
IOCEN	—	—	—	—	IOCEN3(1)	—	—	—
IOCEF	—	—	—	—	IOCEF3(1)	—	—	—

Note 1: If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and IOC on RE3 is not available.

TABLE 18-2: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
IOCxF	IOCxF7	IOCxF6	IOCxF5	IOCxF4	IOCxF3	IOCxF2	IOCxF1	IOCxF0	288
IOCxN	IOCxN7	IOCxN6	IOCxN5	IOCxN4	IOCxN3	IOCxN2	IOCxN1	IOCxN0	288
IOCxP	IOCxP7	IOCxP6	IOCxP5	IOCxP4	IOCxP3	IOCxP2	IOCxP1	IOCxP0	288

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

19.0 PERIPHERAL MODULE DISABLE (PMD)

Sleep, Idle and Doze modes allow users to substantially reduce power consumption by slowing or stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume some amount of power. There may be cases where the application needs what these modes do not provide: the ability to allocate limited power resources to the CPU while eliminating power consumption from the peripherals.

The PIC18F26/27/45/46/47/55/56/57K42 microcontrollers address this requirement by allowing peripheral modules to be selectively enabled or disabled, placing them into the lowest possible power mode.

All modules are ON by default following any Reset.

19.1 Disabling a Module

Disabling a module has the following effects:

- All clock and control inputs to the module are suspended; there are no logic transitions, and the module will not function.
- The module is held in Reset.
- Any SFR becomes “unimplemented”
  - Writing is disabled
  - Reading returns 00h
- I/O functionality is prioritized as per **Section 16.1, I/O Priorities**
- All associated Input Selection registers are also disabled

19.2 Enabling a Module

When the PMD register bit is cleared, the module is re-enabled and will be in its Reset state (Power-on Reset). SFR data will reflect the POR Reset values.

Depending on the module, it may take up to one full instruction cycle for the module to become active. There should be no interaction with the module (e.g., writing to registers) for at least one instruction after it has been re-enabled.

19.3 Effects of a Reset

Following any Reset, each control bit is set to ‘0’, enabling all modules.

19.4 System Clock Disable

Setting SYSCMD (PMD0, Register 19-1) disables the system clock (FOSC) distribution network to the peripherals. Not all peripherals make use of SYSCLK, so not all peripherals are affected. Refer to the specific peripheral description to see if it will be affected by this bit.

19.5 Register Definitions: Peripheral Module Disable

REGISTER 19-1: PMD0: PMD CONTROL REGISTER 0

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
7							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 **SYSCMD:** Disable Peripheral System Clock Network bit(1)
See description in **Section 19.4 “System Clock Disable”**.
1 = System clock network disabled (FOSC)
0 = System clock network enabled

bit 6 **FVRMD:** Disable Fixed Voltage Reference bit
1 = FVR module disabled
0 = FVR module enabled

bit 5 **HLVDMD:** Disable High/Low-Voltage Detect bit
1 = HLVD module disabled
0 = HLVD module enabled

bit 4 **CRCMD:** Disable CRC Engine bit
1 = CRC module disabled
0 = CRC module enabled

bit 3 **SCANMD:** Disable NVM Memory Scanner bit(2)
1 = NVM Memory Scan module disabled
0 = NVM Memory Scan module enabled

bit 2 **NVMMD:** NVM Module Disable bit(3)
1 = All Memory reading and writing is disabled; NVMCON registers cannot be written
0 = NVM module enabled

bit 1 **CLKRMD:** Disable Clock Reference bit
1 = CLKR module disabled
0 = CLKR module enabled

bit 0 **IOCMD:** Disable Interrupt-on-Change bit, All Ports
1 = IOC module(s) disabled
0 = IOC module(s) enabled

Note 1: Clearing the SYSCMD bit disables the system clock (FOSC) to peripherals, however peripherals clocked by FOSC/4 are not affected.
2: Subject to SCANE bit in CONFIG4H.
3: When enabling NVM, a delay of up to 1 µs may be required before accessing data.

REGISTER 19-2: PMD1: PMD CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
NCO1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
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 **NCO1MD:** Disable NCO1 Module bit
1 = NCO1 module disabled
0 = NCO1 module enabled

bit 6 **TMR6MD:** Disable Timer TMR6 bit
1 = TMR6 module disabled
0 = TMR6 module enabled

bit 5 **TMR5MD:** Disable Timer TMR5 bit
1 = TMR5 module disabled
0 = TMR5 module enabled

bit 4 **TMR4MD:** Disable Timer TMR4 bit
1 = TMR4 module disabled
0 = TMR4 module enabled

bit 3 **TMR3MD:** Disable Timer TMR3 bit
1 = TMR3 module disabled
0 = TMR3 module enabled

bit 2 **TMR2MD:** Disable Timer TMR2 bit
1 = TMR2 module disabled
0 = TMR2 module enabled

bit 1 **TMR1MD:** Disable Timer TMR1 bit
1 = TMR1 module disabled
0 = TMR1 module enabled

bit 0 **TMR0MD:** Disable Timer TMR0 bit
1 = TMR0 module disabled
0 = TMR0 module enabled

REGISTER 19-3: PMD2: PMD CONTROL REGISTER 2

U-0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	DACMD	ADCMD	—	—	CMP2MD	CMP1MD	ZCDMD(1)
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 **Unimplemented:** Read as ‘0’

bit 6 **DACMD:** Disable DAC bit
1 = DAC module disabled
0 = DAC module enabled

bit 5 **ADCMD:** Disable ADCC bit
1 = ADCC module disabled
0 = ADCC module enabled

bit 4-3 **Unimplemented:** Read as ‘0’

bit 2 **CMP2MD:** Disable Comparator CMP2 bit
1 = CMP2 module disabled
0 = CMP2 module enabled

bit 1 **CMP1MD:** Disable Comparator CMP1 bit
1 = CMP1 module disabled
0 = CMP1 module enabled

bit 0 **ZCDMD:** Disable Zero-Cross Detect module bit(1)
1 = ZCD module disabled
0 = ZCD module enabled

Note 1: Subject to $\overline{ZCD}$ bit in CONFIG2H.

REGISTER 19-4: PMD3: PMD CONTROL REGISTER 3

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
PWM8MD	PWM7MD	PWM6MD	PWM5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD
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 **PWM8MD:** Disable Pulse-Width Modulator PWM8 bit
1 = PWM8 module disabled
0 = PWM8 module enabled

bit 6 **PWM7MD:** Disable Pulse-Width Modulator PWM7 bit
1 = PWM7 module disabled
0 = PWM7 module enabled

bit 5 **PWM6MD:** Disable Pulse-Width Modulator PWM6 bit
1 = PWM6 module disabled
0 = PWM6 module enabled

bit 4 **PWM5MD:** Disable Pulse-Width Modulator PWM5 bit
1 = PWM5 module disabled
0 = PWM5 module enabled

bit 3 **CCP4MD:** Disable Capture/Compare/PWM CCP4 bit
1 = CCP4 module disabled
0 = CCP4 module enabled

bit 2 **CCP3MD:** Disable Capture/Compare/PWM CCP3 bit
1 = CCP3 module disabled
0 = CCP3 module enabled

bit 1 **CCP2MD:** Disable Capture/Compare/PWM CCP2 bit
1 = CCP2 module disabled
0 = CCP2 module enabled

bit 0 **CCP1MD:** Disable Capture/Compare/PWM CCP1 bit
1 = CCP1 module disabled
0 = CCP1 module enabled

© 2017-2020 Microchip Technology Inc.

REGISTER 19-5: PMD4: PMD CONTROL REGISTER 4

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	U-0
CWG3MD	CWG2MD	CWG1MD	—	—	—	—	—
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 **CWG3MD:** Disable CWG3 Module bit
1 = CWG3 module disabled
0 = CWG3 module enabled

bit 6 **CWG2MD:** Disable CWG2 Module bit
1 = CWG2 module disabled
0 = CWG2 module enabled

bit 5 **CWG1MD:** Disable CWG1 Module bit
1 = CWG1 module disabled
0 = CWG1 module enabled

bit 4-0 **Unimplemented:** Read as ‘0’

REGISTER 19-6: PMD5: PMD CONTROL REGISTER 5

U-0	U-0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	U2MD	U1MD	—	SPI1MD	I2C2MD	I2C1MD
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 7-6 **Unimplemented:** Read as ‘0’

bit 5 **U2MD:** Disable UART2 bit
1 = UART2 module disabled
0 = UART2 module enabled

bit 4 **U1MD:** Disable UART1 bit
1 = UART1 module disabled
0 = UART1 module enabled

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **SPI1MD:** Disable SPI1 Module bit
1 = SPI1 module disabled
0 = SPI1 module enabled

bit 1 **I2C2MD:** Disable I^2C2 Module bit
1 = I^2C2 module disabled
0 = I^2C2 module enabled

bit 0 **I2C1MD:** Disable I^2C1 Module bit
1 = I^2C1 module disabled
0 = I^2C1 module enabled

REGISTER 19-7: PMD6: PMD CONTROL REGISTER 6

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	SMT1MD	CLC4MD	CLC3MD	CLC2MD	CLC1MD	DSMMD
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 7-6 **Unimplemented:** Read as ‘0’

bit 5 **SMT1MD:** Disable SMT1 Module bit
1 = SMT1 module disabled
0 = SMT1 module enabled

bit 4 **CLC1MD:** Disable CLC4 Module bit
1 = CLC4 module disabled
0 = CLC4 module enabled

bit 3 **CLC3MD:** Disable CLC3 Module bit
1 = CLC3 module disabled
0 = CLC3 module enabled

bit 2 **CLC2MD:** Disable CLC2 Module bit
1 = CLC2 module disabled
0 = CLC2 module enabled

bit 1 **CLC1MD:** Disable CLC1 Module bit
1 = CLC1 module disabled
0 = CLC1 module enabled

bit 0 **DSMMD:** Disable Data Signal Modulator bit
1 = DSM module disabled
0 = DSM module enabled

REGISTER 19-8: PMD7: PMD CONTROL REGISTER 7

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	DMA2MD	DMA1MD
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 7-2 **Unimplemented:** Read as ‘0’

bit 1 **DMA2MD:** Disable DMA2 Module bit
1 = DMA2 module disabled
0 = DMA2 module enabled

bit 0 **DMA1MD:** Disable DMA1 Module bit
1 = DMA1 module disabled
0 = DMA1 module enabled

TABLE 19-1: SUMMARY OF REGISTERS ASSOCIATED WITH PERIPHERAL MODULE DISABLE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PMD0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	291
PMD1	NCO1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD	292
PMD2	—	DACMD	ADCMD	—	—	CMP2MD	CMP1MD	ZCDMD	293
PMD3	PWM8MD	PWM7MD	PWM6MD	PWM5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	294
PMD4	CWG3MD	CWG2MD	CWG1MD	—	—	—	—	—	295
PMD5	—	—	U2MD	U1MD	—	SPI1MD	I2C2MD	I2C1MD	296
PMD6	—	—	SMT1MD	CLC4MD	CLC3MD	CLC2MD	CLC1MD	DSMMD	296
PMD7	—	—	—	—	—	—	DMA2MD	DMA1MD	298

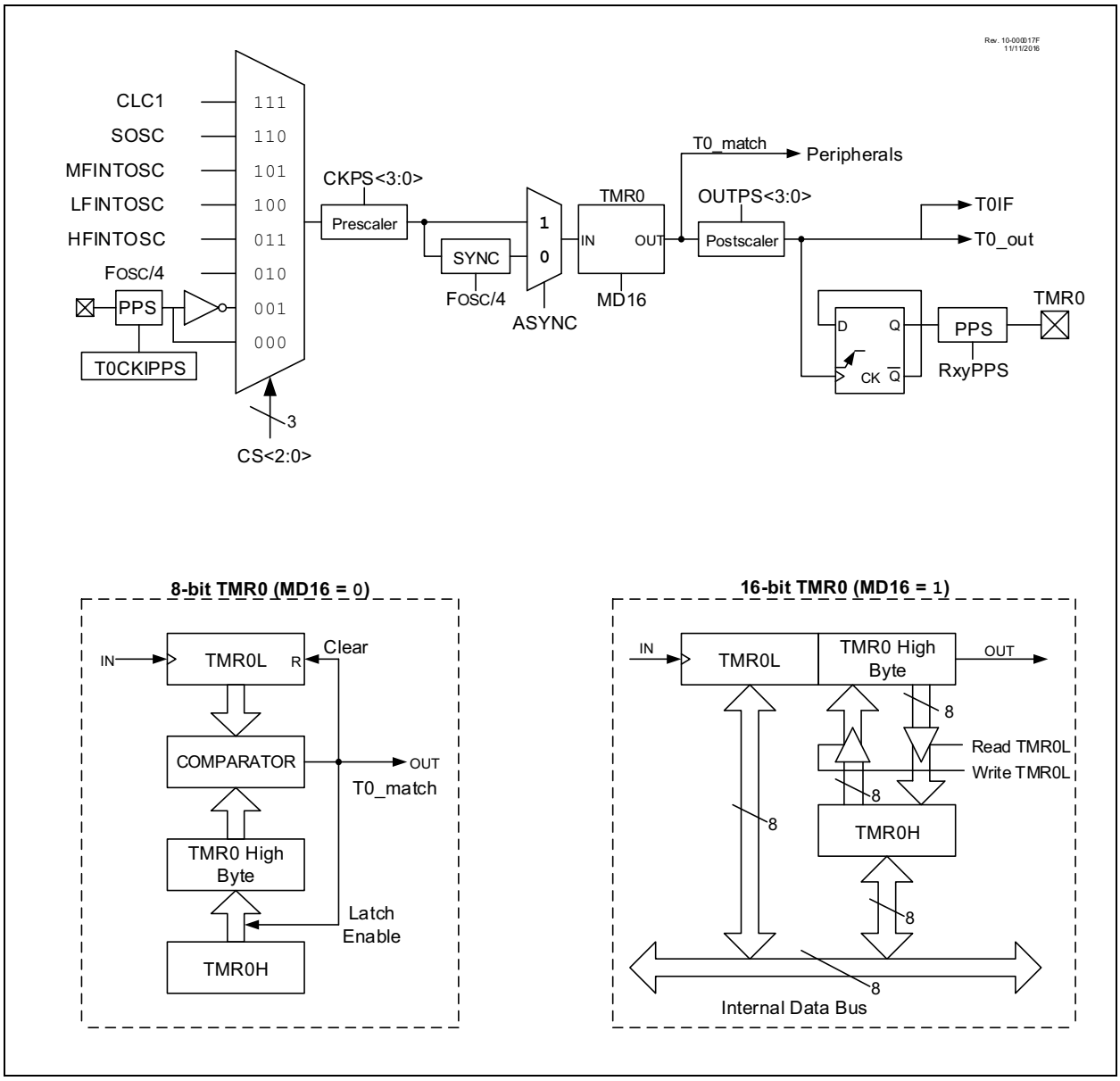
Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used by peripheral module disable.

20.0 TIMER0 MODULE

Timer0 module is an 8/16-bit timer/counter with the following features:

- 16-bit timer/counter
- 8-bit timer/counter with programmable period
- Synchronous or asynchronous operation
- Selectable clock sources
- Programmable prescaler
- Programmable postscaler
- Operation during Sleep mode
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals

FIGURE 20-1: BLOCK DIAGRAM OF TIMER0



20.1 Timer0 Operation

Timer0 can operate as either an 8-bit timer/counter or a 16-bit timer/counter. The mode is selected with the MD16 bit of the T0CON register.

20.1.1 16-BIT MODE

The register pair TMR0H:TMR0L increments on the rising edge of the clock source. A 15-bit prescaler on the clock input gives several prescale options (see prescaler control bits, CKPS[3:0] in the T0CON1 register).

20.1.1.1 Timer0 Reads and Writes in 16-Bit Mode

In 16-bit mode, in order to avoid rollover between reading high and low registers, the TMR0H register is a buffered copy of the actual high byte of Timer0, which is neither directly readable, nor writable (see Figure 20-1). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte was valid, due to a rollover between successive reads of the high and low byte.

Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

20.1.2 8-BIT MODE

In 8-bit mode, the value of TMR0L is compared to that of the Period buffer, a copy of TMR0H, on each clock cycle. When the two values match, the following events happen:

- TMR0_out goes high for one prescaled clock period
- TMR0L is reset
- The contents of TMR0H are copied to the period buffer

In 8-bit mode, the TMR0L and TMR0H registers are both directly readable and writable. The TMR0L register is cleared on any device Reset, while the TMR0H register initializes at FFh.

Both the prescaler and postscaler counters are cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset – Power-on Reset (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or
- Brown-out Reset (BOR)

20.1.3 COUNTER MODE

In Counter mode, the prescaler is normally disabled by setting the CKPS bits of the T0CON1 register to ‘0000’. Each rising edge of the clock input (or the output of the prescaler if the prescaler is used) increments the counter by ‘1’.

20.1.4 TIMER MODE

In Timer mode, the Timer0 module will increment every instruction cycle as long as there is a valid clock signal and the CKPS bits of the T0CON1 register (Register 20-2) are set to ‘0000’. When a prescaler is added, the timer will increment at the rate based on the prescaler value.

20.1.5 ASYNCHRONOUS MODE

When the ASYNC bit of the T0CON1 register is set (ASYNC = ‘1’), the counter increments with each rising edge of the input source (or output of the prescaler, if used). Asynchronous mode allows the counter to continue operation during Sleep mode provided that the clock also continues to operate during Sleep.

20.1.6 SYNCHRONOUS MODE

When the ASYNC bit of the T0CON1 register is clear (ASYNC = ‘0’), the counter clock is synchronized to the system clock (FOSC/4). When operating in Synchronous mode, the counter clock frequency cannot exceed FOSC/4.

20.2 Clock Source Selection

The CS[2:0] bits of the T0CON1 register are used to select the clock source for Timer0. Register 20-2 displays the clock source selections.

20.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected, Timer0 operates as a timer and will increment on multiples of the clock source, as determined by the Timer0 prescaler.

20.2.2 EXTERNAL CLOCK SOURCE

When an external clock source is selected, Timer0 can operate as either a timer or a counter. Timer0 will increment on multiples of the rising edge of the external clock source, as determined by the Timer0 prescaler.

© 2017-2020 Microchip Technology Inc.

20.3 Programmable Prescaler

A software programmable prescaler is available for exclusive use with Timer0. There are 16 prescaler options for Timer0 ranging in powers of two from 1:1 to 1:32768. The prescaler values are selected using the CKPS[3:0] bits of the T0CON1 register.

The prescaler is not directly readable or writable. Clearing the prescaler register can be done by writing to the TMR0L register or to the T0CON0/T0CON1 register or by any Reset.

20.4 Programmable Postscaler

A software programmable postscaler (output divider) is available for exclusive use with Timer0. There are 16 postscaler options for Timer0 ranging from 1:1 to 1:16. The postscaler values are selected using the OUTPS bits of the T0CON0 register.

The postscaler is not directly readable or writable. Clearing the postscaler register can be done by writing to the TMR0L register or to the T0CON0/T0CON1 register or by any Reset.

20.5 Operation During Sleep

When operating synchronously, Timer0 will halt. When operating asynchronously, Timer0 will continue to increment and wake the device from Sleep (if Timer0 interrupts are enabled) provided that the input clock source is active.

20.6 Timer0 Interrupts

The Timer0 interrupt flag bit (TMR0IF) is set when either of the following conditions occur:

- 8-bit TMR0L matches the TMR0H value
- 16-bit TMR0 rolls over from ‘FFFFh’

When the postscaler bits (OUTPS) are set to 1:1 operation (no division), the T0IF flag bit will be set with every TMR0 match or rollover. In general, the TMR0IF flag bit will be set every OUTPS +1 matches or rollovers.

If Timer0 interrupts are enabled (TMR0IE bit of the PIE3 register = ‘1’), the CPU will be interrupted and the device may wake from Sleep (see **Section 20.2 “Clock Source Selection”** for more details).

20.7 Timer0 Output

The Timer0 output can be routed to any I/O pin via the RxyPPS output selection register (see **Section 17.0 “Peripheral Pin Select (PPS) Module”** for additional information). The Timer0 output can also be used by other peripherals, such as the auto-conversion trigger of the Analog-to-Digital Converter. Finally, the Timer0 output can be monitored through software via the Timer0 output bit (OUT) of the T0CON0 register (Register 20-1).

TMR0_out will be a pulse of one postscaled clock period when a match occurs between TMR0L and PR0 (Period register for TMR0) in 8-bit mode, or when TMR0 rolls over in 16-bit mode. The Timer0 output is a 50% duty cycle that toggles on each TMR0_out rising clock edge.

20.8 Register Definitions: Timer0 Control

REGISTER 20-1: T0CON0: TIMER0 CONTROL REGISTER 0

R/W-0/0	U-0	R-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	OUT	MD16	OUTPS[3:0]			
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 **EN:** TMR0 Enable bit
`1` = The module is enabled and operating
`0` = The module is disabled and in the lowest power mode

bit 6 **Unimplemented:** Read as ‘`0`’

bit 5 **OUT:** TMR0 Output bit (read-only)
TMR0 output bit

bit 4 **MD16:** TMR0 Operating as 16-Bit Timer Select bit
`1` = TMR0 is a 16-bit timer
`0` = TMR0 is an 8-bit timer

bit 3-0 **OUTPS[3:0]:** TMR0 Output Postscaler (Divider) Select bits
`1111` = 1:16 Postscaler
`1110` = 1:15 Postscaler
`1101` = 1:14 Postscaler
`1100` = 1:13 Postscaler
`1011` = 1:12 Postscaler
`1010` = 1:11 Postscaler
`1001` = 1:10 Postscaler
`1000` = 1:9 Postscaler
`0111` = 1:8 Postscaler
`0110` = 1:7 Postscaler
`0101` = 1:6 Postscaler
`0100` = 1:5 Postscaler
`0011` = 1:4 Postscaler
`0010` = 1:3 Postscaler
`0001` = 1:2 Postscaler
`0000` = 1:1 Postscaler

REGISTER 20-2: T0CON1: TIMER0 CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
CS[2:0]			ASYNC	CKPS[3:0]			
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 7-5 **CS[2:0]:** Timer0 Clock Source Select bits
111 = CLC1
110 = SOSC
101 = MFINTOSC (500 kHz)
100 = LFINTOSC
011 = HFINTOSC
010 = FOSC/4
001 = Pin selected by T0CKIPPS (Inverted)
000 = Pin selected by T0CKIPPS (Non-inverted)

bit 4 **ASYNC:** TMR0 Input Asynchronization Enable bit
1 = The input to the TMR0 counter is not synchronized to system clocks
0 = The input to the TMR0 counter is synchronized to FOSC/4

bit 3-0 **CKPS[3:0]:** Prescaler Rate Select bit
1111 = 1:32768
1110 = 1:16384
1101 = 1:8192
1100 = 1:4096
1011 = 1:2048
1010 = 1:1024
1001 = 1:512
1000 = 1:256
0111 = 1:128
0110 = 1:64
0101 = 1:32
0100 = 1:16
0011 = 1:8
0010 = 1:4
0001 = 1:2
0000 = 1:1

REGISTER 20-3: TMR0L: TIMER0 COUNT REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TMR0L[7:0]							
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 7-0 **TMR0L[7:0]:** TMR0 Counter bits [7:0]

REGISTER 20-4: TMR0H: TIMER0 PERIOD REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
TMR0H[15:8]							
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 7-0 When MD16 = 0
PR0[7:0]:TMR0 Period Register Bits [7:0]
When MD16 = 1
TMR0H[15:8]: TMR0 Counter bits [15:8]

TABLE 20-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
T0CON0	EN	—	OUT	MD16	OUTPS[3:0]				302
T0CON1	CS[2:0]			ASYNC	CKPS[3:0]				303
TMR0L	TMR0L[7:0]								304
TMR0H	TMR0H[15:8]								304

Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used by Timer0.

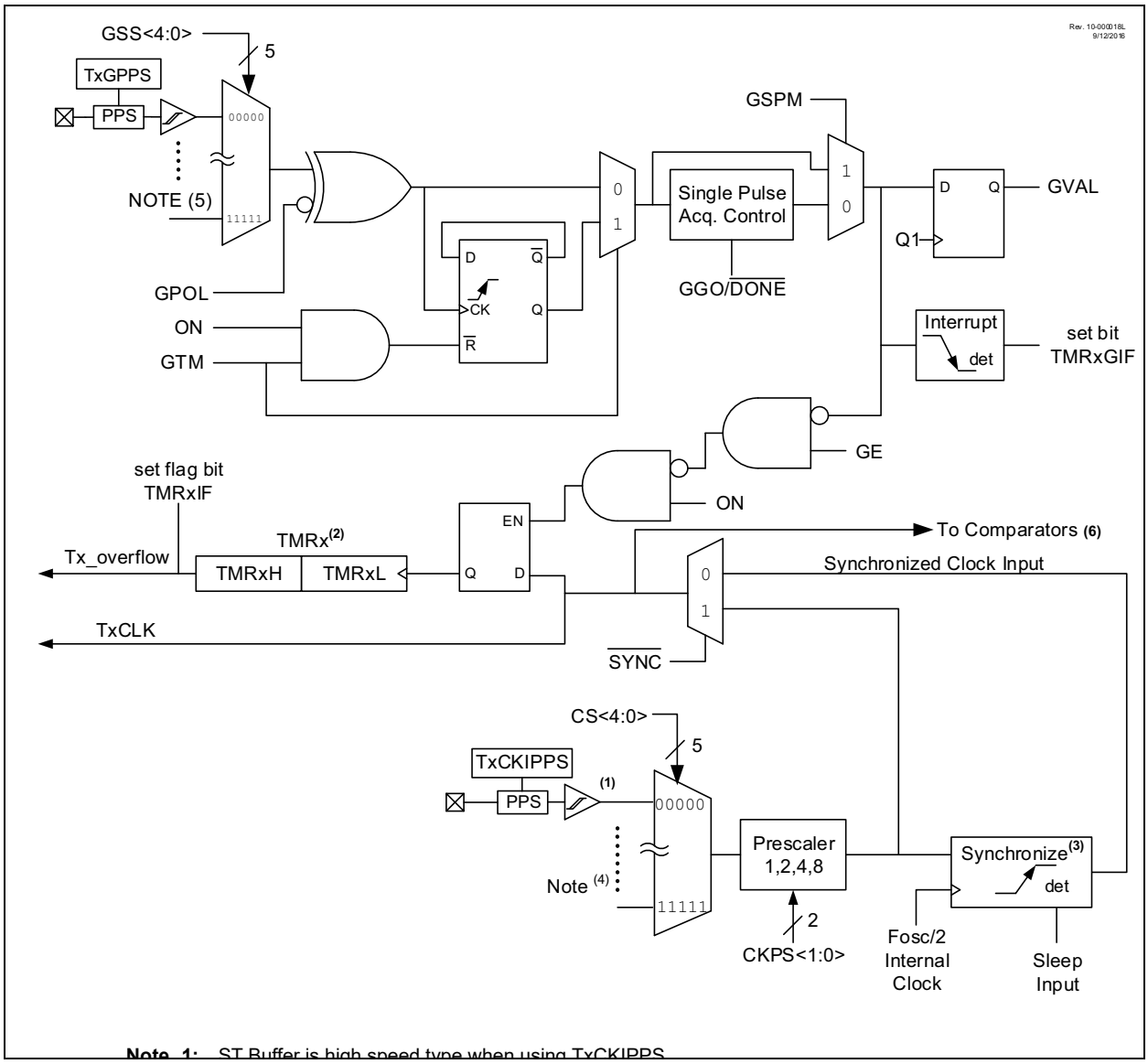
21.0 TIMER1/3/5 MODULE WITH GATE CONTROL

Timer1/3/5 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated Secondary 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1/3/5 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- 16-Bit Read/Write Operation
- Time base for the Capture/Compare function with the CCP modules
- Special Event Trigger (with CCP)
- 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 Timer1/3/5 module.

FIGURE 21-1: TIMER1/3/5 BLOCK DIAGRAM



Note 1: ST Buffer is high speed type when using TxCKIPPS.

21.1 Timer1/3/5 Operation

The Timer1/3/5 module is a 16-bit incrementing counter which is accessed through the TMRxH:TMRxL register pair. Writes to TMRxH or TMRxL directly update the counter.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.

Timer1/3/5 is enabled by configuring the ON and GE bits in the TxCON and TxGCON registers, respectively. Table 21-1 displays the Timer1/3/5 enable selections.

TABLE 21-1: TIMER1/3/5 ENABLE SELECTIONS

ON	GE	Timer1/3/5 Operation
1	1	Count Enabled
1	0	Always On
0	1	Off
0	0	Off

21.2 Clock Source Selection

The CS[4:0] bits of the TMRxCLK register (Register 21-3) are used to select the clock source for Timer1/3/5. The TxCLK register allows the selection of several possible synchronous and asynchronous clock sources. Register 21-3 displays the clock source selections.

21.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected the TMRxH:TMRxL register pair will increment on multiples of FOSC as determined by the Timer1/3/5 prescaler.

When the FOSC internal clock source is selected, the Timer1/3/5 register value will increment by four counts every instruction clock cycle. Due to this condition, a 2 LSB error in resolution will occur when reading the Timer1/3/5 value. To utilize the full resolution of Timer1/3/5, an asynchronous input signal must be used to gate the Timer1/3/5 clock input.

The following asynchronous sources may be used at the Timer1/3/5 gate:

- Asynchronous event on the TxGPPS pin
- TMR0OUT
- TMR1/3/5OUT (excluding the TMR for which it is being used)
- TMR 2/4/6OUT (postscaled)
- CMP1/2OUT
- SMT1_match
- NCO1OUT
- PWM3/4 OUT
- CCP1/2/3/4 OUT
- CLC1/2/3/4 OUT
- ZCDOUT

Note: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1/3/5 enabled after POR
- Write to TMRxH or TMRxL
- Timer1/3/5 is disabled
- Timer1/3/5 is disabled (TMRxON = 0) when TxCKI is high then Timer1/3/5 is enabled (TMRxON = 1) when TxCKI is low.

21.2.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1/3/5 module may work as a timer or a counter.

When enabled to count, Timer1/3/5 is incremented on the rising edge of the external clock input of the TxCKIPPS pin. This external clock source can be synchronized to the microcontroller system clock or it can run asynchronously.

When used as a timer with a clock oscillator, an external 32.768 kHz crystal can be used in conjunction with the dedicated secondary internal oscillator circuit.

21.3 Timer1/3/5 Prescaler

Timer1/3/5 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The CKPS bits of the TxCON register control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRxH or TMRxL.

21.4 Timer1/3/5 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{SYNC}}$ of the TxCON register is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If external clock source is selected then the timer will continue to run during Sleep and can generate an interrupt on overflow, which will wake up the processor. However, special precautions in software are needed to read/write the timer (see **Section 21.4.1 "Reading and Writing Timer1/3/5 in Asynchronous Counter Mode"**).

Note:	When switching from synchronous to asynchronous operation, it is possible to skip an increment. When switching from asynchronous to synchronous operation, it is possible to produce an additional increment.

21.4.1 READING AND WRITING TIMER1/3/5 IN ASYNCHRONOUS COUNTER MODE

Reading TMRxH or TMRxL while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads. For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the TMRxH:TMRxL register pair.

21.5 Timer1/3/5 16-Bit Read/Write Mode

Timer1/3/5 can be configured to read and write all 16 bits of data, to and from, the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the RD16 bit of the TxCON register.

To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-Bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1/3/5 value from a single instance in time. Reference the block diagram in Figure 21-2 for more details.

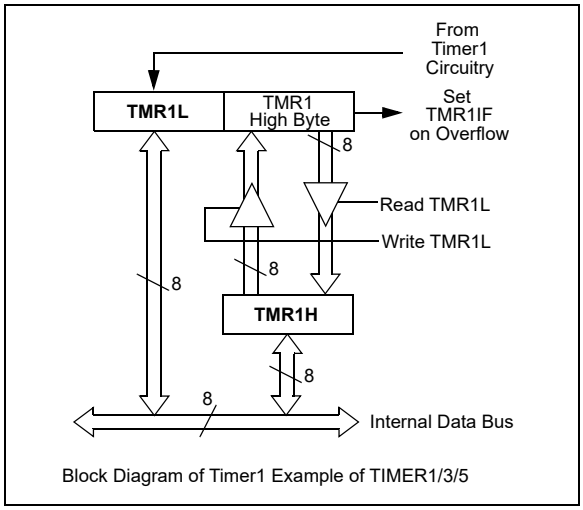
In contrast, when not in 16-Bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRxL:TMRxH register pair at the same time.

Any requests to write to the TMRxH directly does not clear the Timer1/3/5 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.

© 2017-2020 Microchip Technology Inc.

FIGURE 21-2: TIMER1/3/5 16-BIT READ/ WRITE MODE BLOCK DIAGRAM



21.6 Timer1/3/5 Gate

Timer1/3/5 can be configured to count freely or the count can be enabled and disabled using Timer1/3/5 gate circuitry. This is also referred to as Timer1/3/5 gate enable.

Timer1/3/5 gate can also be driven by multiple selectable sources.

21.6.1 TIMER1/3/5 GATE ENABLE

The Timer1/3/5 Gate Enable mode is enabled by setting the TMRxGE bit of the TxGCON register. The polarity of the Timer1/3/5 Gate Enable mode is configured using the TxGPOL bit of the TxGCON register.

When Timer1/3/5 Gate Enable mode is enabled, Timer1/3/5 will increment on the rising edge of the Timer1/3/5 clock source. When Timer1/3/5 Gate signal is inactive, the timer will not increment and hold the current count. See Figure 21-4 for timing details.

TABLE 21-2: TIMER1/3/5 GATE ENABLE SELECTIONS

TMRxCLK	TxGPOL	TxG	Timer1/3/5 Operation
↑	1	1	Counts
↑	1	0	Holds Count
↑	0	1	Holds Count
↑	0	0	Counts

21.6.2 TIMER1/3/5 GATE SOURCE SELECTION

The gate source for Timer1/3/5 can be selected using the GSS[4:0] bits of the TMRxGATE register (Register 21-4). The polarity selection for the gate source is controlled by the TxGPOL bit of the TxGCON register (Register 21-2).

Any of the above mentioned signals can be used to trigger the gate. The output of the CMPx can be synchronized to the Timer1/3/5 clock or left asynchronous. For more information see **Section 38.3.1 "Comparator Output Synchronization"**.

21.6.3 TIMER1/3/5 GATE TOGGLE MODE

When Timer1/3/5 Gate Toggle mode is enabled, it is possible to measure the duration between every rising and falling edge of the gate signal.

The Timer1/3/5 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 21-5 for timing details.

Timer1/3/5 Gate Toggle mode is enabled by setting the GTM bit of the TxGCON register. When the GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note:	Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

21.6.4 TIMER1/3/5 GATE SINGLE-PULSE MODE

When Timer1/3/5 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1/3/5 Gate Single-Pulse mode is first enabled by setting the GSPM bit in the TxGCON register. Next, the GGO/$\overline{\text{DONE}}$ bit in the TxGCON register must be set. The Timer1/3/5 will be fully enabled on the next incrementing edge of the gate signal. On the next trailing edge of the pulse, the GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1/3/5 until the GGO/$\overline{\text{DONE}}$ bit is once again set in software.

Clearing the TxGSPM bit of the TxGCON register will also clear the GGO/$\overline{\text{DONE}}$ bit. See Figure 21-6 for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the period on the Timer1/3/5 gate source to be measured. See Figure 21-7 for timing details.

21.6.5 TIMER1/3/5 GATE VALUE STATUS

When Timer1/3/5 Gate Value Status is utilized, it is possible to read the most current level of the gate signal. The value is stored in the GVAL bit in the TxGCON register. The GVAL bit is valid even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

21.6.6 TIMER1/3/5 GATE EVENT INTERRUPT

When Timer1/3/5 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of GVAL occurs, the TMRxGIF flag bit in the respective PIR register will be set. If the TMRxGIE bit in the respective PIE register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

For more information on selecting high or low priority status for the Timer1/3/5 Gate Event Interrupt see **Section 9.0 "Interrupt Controller"**.

21.7 Timer1/3/5 Interrupt

The Timer1/3/5 register pair (TMRxH:TMRxL) increments to FFFFh and rolls over to 0000h. When Timer1/3/5 rolls over, the Timer1/3/5 interrupt flag bit of the respective PIR register is set. To enable the interrupt-on-rollover, you must set these bits:

- ON bit of the TxCON register
- TMRxIE bits of the respective PIE register
- GIE/GIEH bit of the INTCON0 register

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

For more information on selecting high or low priority status for the Timer1/3/5 Overflow Interrupt, see **Section 9.0 "Interrupt Controller"**.

Note:	The TMRxH:TMRxL register pair and the TMRxIF bit should be cleared before enabling interrupts.

21.8 Timer1/3/5 Operation During Sleep

Timer1/3/5 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:

- ON bit of the TxCON register must be set
- TMRxIE bit of the respective PIE register must be set
- $\overline{\text{SYNC}}$ bit of the TxCON register must be set
- Configure the TMRxCLK register for using secondary oscillator as the clock source
- Enable the SOSCEN bit of the OSCEN register (Register 7-7)

The device will wake-up on an overflow and execute the next instruction. If the GIE/GIEH bit of the INTCON0 register is set, the device will call the Interrupt Service Routine.

The secondary oscillator will continue to operate in Sleep regardless of the $\overline{\text{SYNC}}$ bit setting.

21.9 CCP Capture/Compare Time Base

The CCP modules use the TMRxH:TMRxL register pair as the time base when operating in Capture or Compare mode.

In Capture mode, the value in the TMRxH:TMRxL register pair is copied into the CCPRxH:CCPRxL register pair on a configured event.

In Compare mode, an event is triggered when the value in the CCPRxH:CCPRxL register pair matches the value in the TMRxH:TMRxL register pair. This event can be a Special Event Trigger.

For more information, see **Section 23.0 "Capture/Compare/PWM Module"**.

21.10 CCP Special Event Trigger

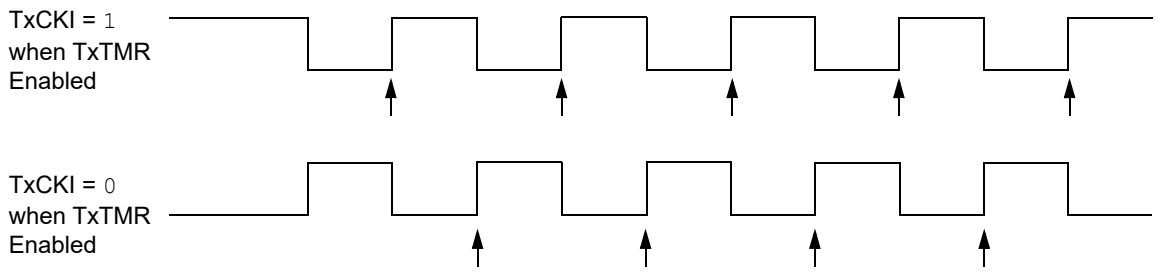
When any of the CCP's are configured to trigger a special event, the trigger will clear the TMRxH:TMRxL register pair. This special event does not cause a Timer1/3/5 interrupt. The CCP module may still be configured to generate a CCP interrupt.

In this mode of operation, the CCPRxH:CCPRxL register pair becomes the period register for Timer1/3/5.

Timer1/3/5 should be synchronized and FOSC/4 should be selected as the clock source in order to utilize the Special Event Trigger. Asynchronous operation of Timer1/3/5 can cause a Special Event Trigger to be missed.

In the event that a write to TMRxH or TMRxL coincides with a Special Event Trigger from the CCP, the write will take precedence.

FIGURE 21-3: TIMER1/3/5 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.

FIGURE 21-4: TIMER1/3/5 GATE ENABLE MODE

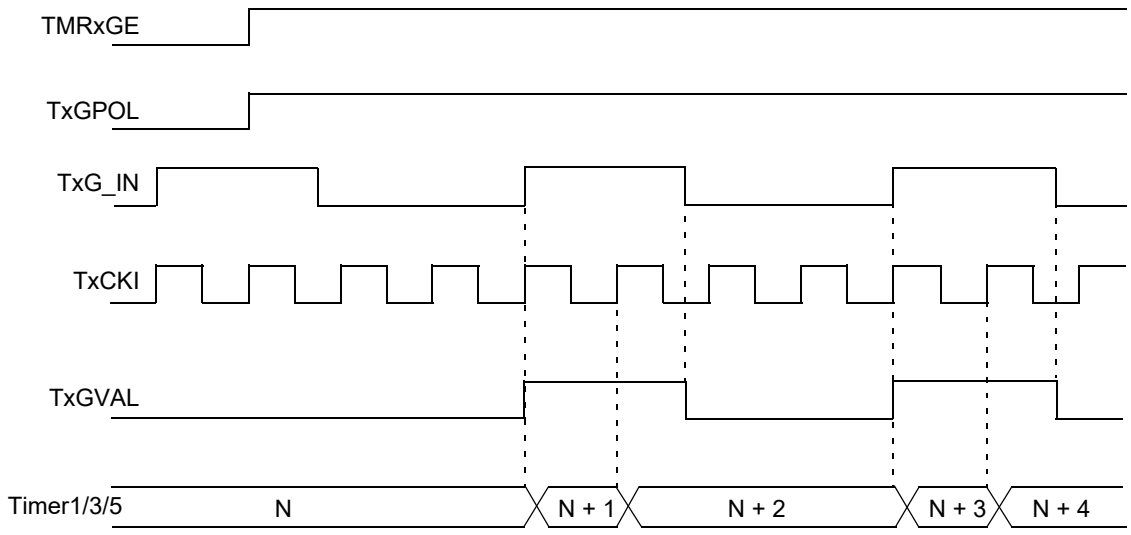


FIGURE 21-5: TIMER1/3/5 GATE TOGGLE MODE

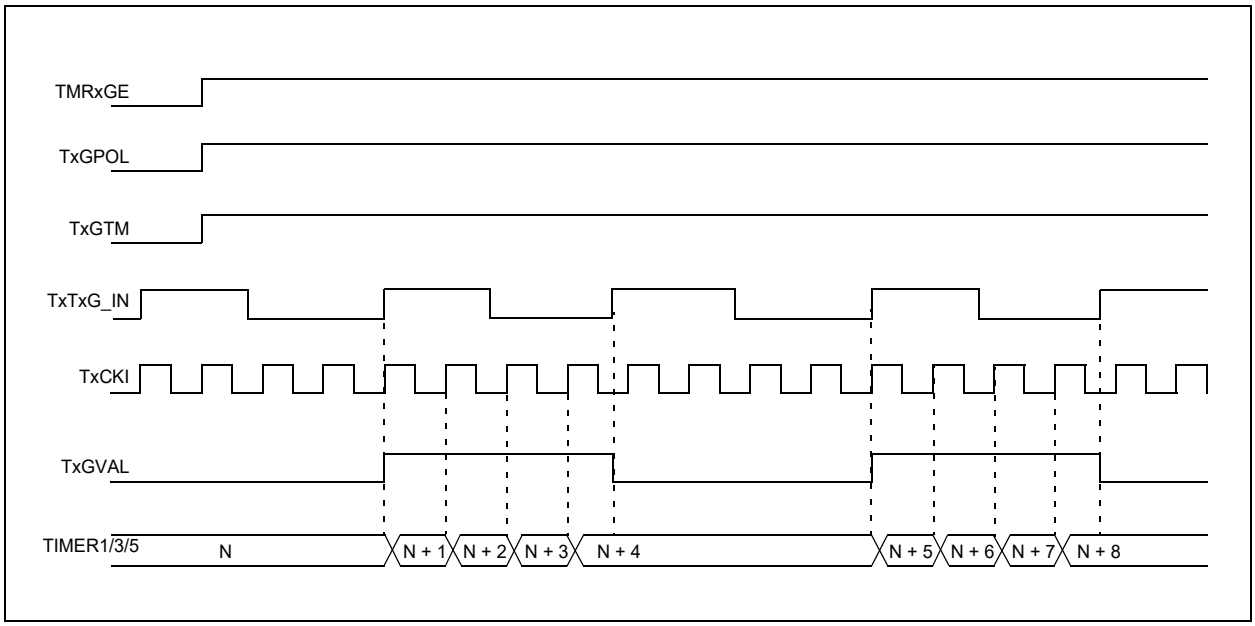


FIGURE 21-6: TIMER1/3/5 GATE SINGLE-PULSE MODE

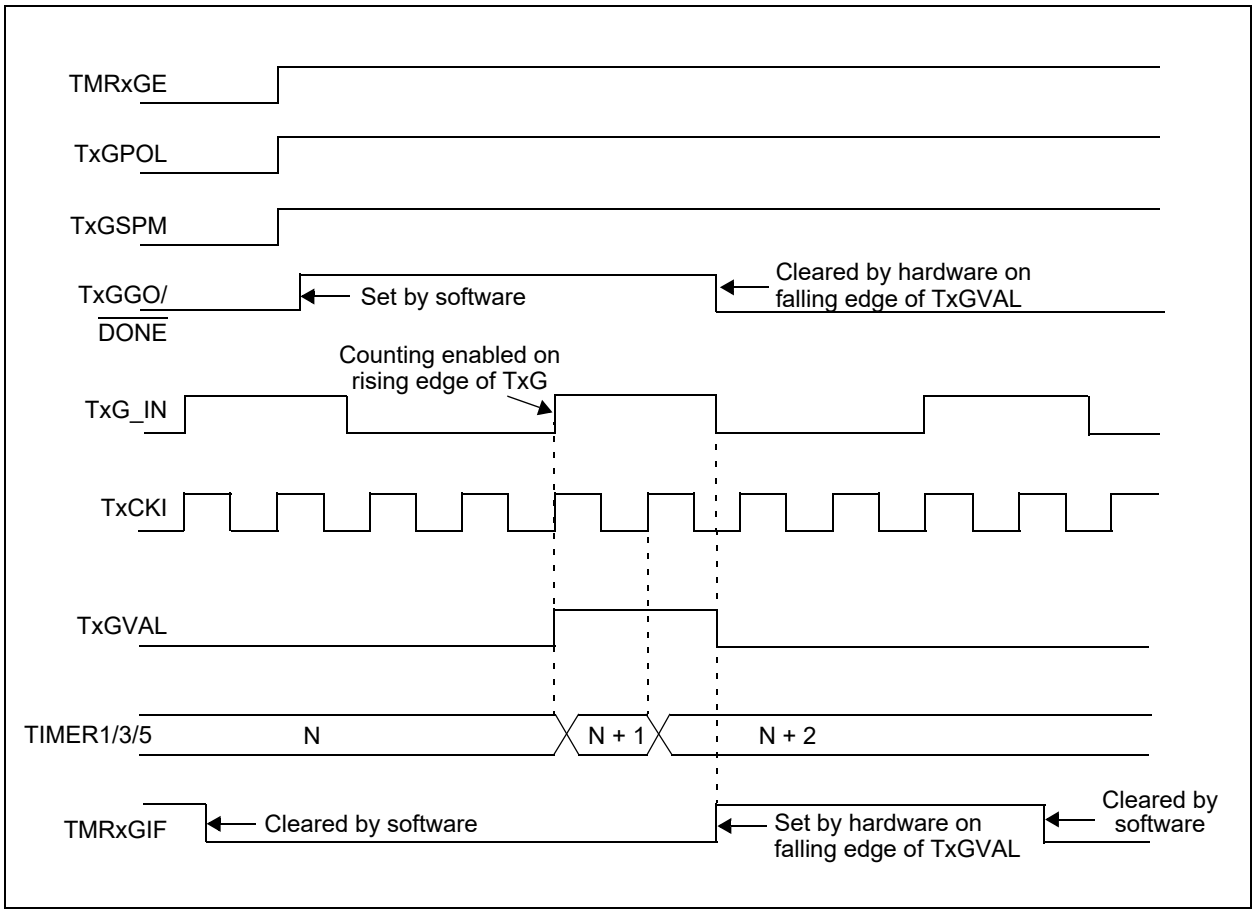
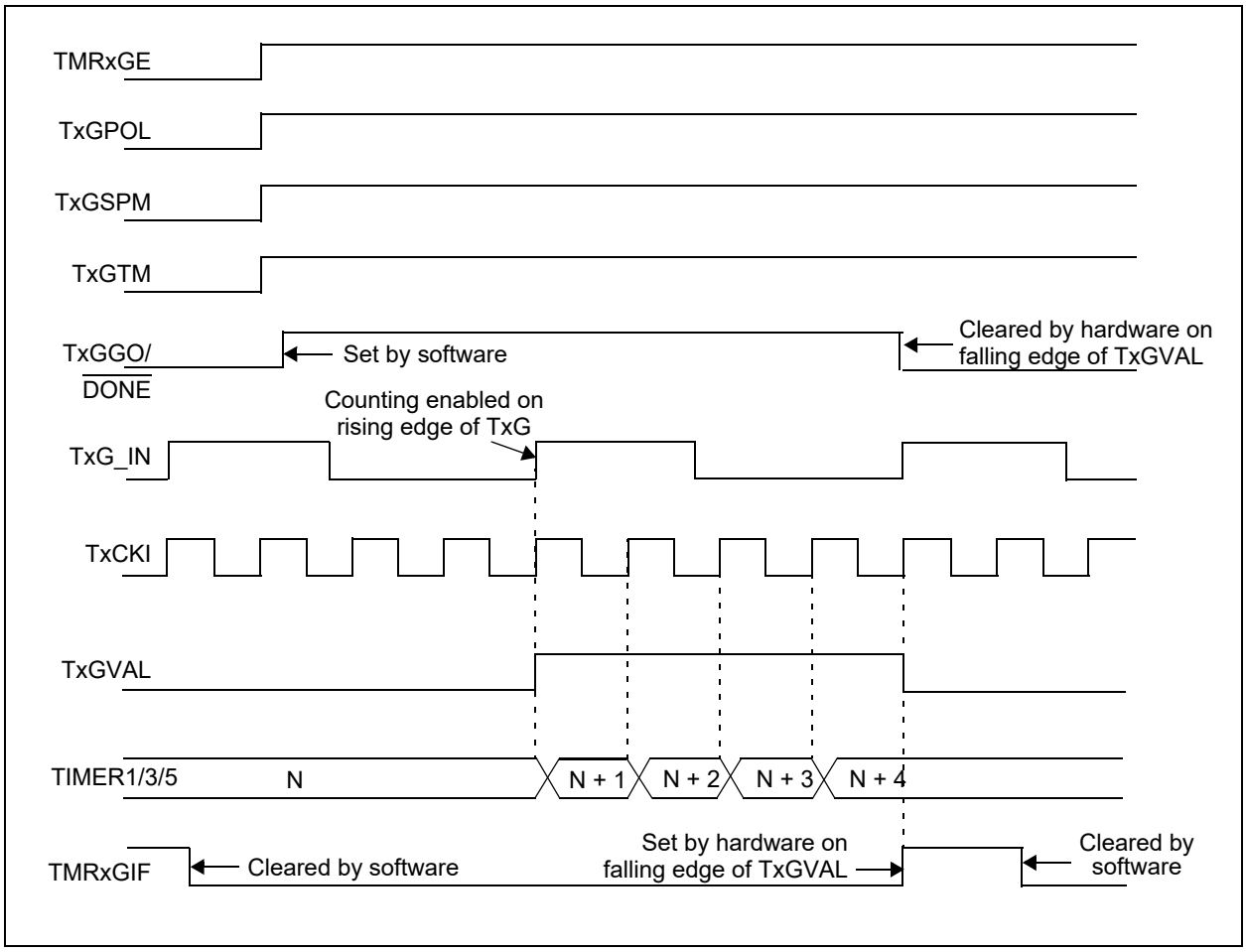


FIGURE 21-7: TIMER1/3/5 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE



21.11 Peripheral Module Disable

When a peripheral module is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bits for Timer1 (TMR1MD), Timer3 (TMR3MD) and Timer5 (TMR5MD) are in the respective PMD registers. See **Section 19.0 "Peripheral Module Disable (PMD)"** for more information.

21.12 Register Definitions: Timer1/3/5

Long bit name prefixes for the Timer1/3/5 are shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
Timer1	T1
Timer3	T3
Timer5	T5

REGISTER 21-1: TXCON: TIMERx CONTROL REGISTER

U-0	U-0	R/W-0/u	R/W-0/u	U-0	R/W-0/u	R/W-0/0	R/W-0/u
—	—	CKPS[1:0]		—	$\overline{\text{SYNC}}$	RD16	ON
bit 7							bit 0

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

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CKPS[1:0]:** Timerx 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 **Unimplemented:** Read as '0'

bit 2 **$\overline{\text{SYNC}}$:** Timerx External Clock Input Synchronization Control bit

TMRxCLK = FOSC/4 or FOSC:

This bit is ignored. Timer1 uses the incoming clock as is.

Else:
- 1 = Do not synchronize external clock input
- 0 = Synchronize external clock input with system clock

bit 1 **RD16:** 16-Bit Read/Write Mode Enable bit
- 1 = Enables register read/write of Timerx in one 16-bit operation
- 0 = Enables register read/write of Timerx in two 8-bit operation

bit 0 **ON:** Timerx On bit
- 1 = Enables Timerx
- 0 = Disables Timerx

REGISTER 21-2: TxGCON: TIMERx GATE CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R-x	U-0	U-0
GE	GPOL	GTM	GSPM	GGO/$\overline{\text{DONE}}$	GVAL	—	—
bit 7							bit 0

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

bit 7 **GE:** Timerx Gate Enable bit
If TMRxON = 1:
1 = Timerx counting is controlled by the Timerx gate function
0 = Timerx is always counting
If TMRxON = 0:
This bit is ignored

bit 6 **GPOL:** Timerx Gate Polarity bit
1 = Timerx gate is active-high (Timerx counts when gate is high)
0 = Timerx gate is active-low (Timerx counts when gate is low)

bit 5 **GTM:** Timerx Gate Toggle Mode bit
1 = Timerx Gate Toggle mode is enabled
0 = Timerx Gate Toggle mode is disabled and Toggle flip-flop is cleared
Timerx Gate Flip Flop Toggles on every rising edge

bit 4 **GSPM:** Timerx Gate Single Pulse Mode bit
1 = Timerx Gate Single Pulse mode is enabled and is controlling Timerx gate)
0 = Timerx Gate Single Pulse mode is disabled

bit 3 **GGO/$\overline{\text{DONE}}$:** Timerx Gate Single Pulse Acquisition Status bit
1 = Timerx Gate Single Pulse Acquisition is ready, waiting for an edge
0 = Timerx Gate Single Pulse Acquisition has completed or has not been started.
This bit is automatically cleared when TxGSPM is cleared.

bit 2 **GVAL:** Timerx Gate Current State bit
Indicates the current state of the Timerx gate that could be provided to TMRxH:TMRxL
Unaffected by Timerx Gate Enable (TMRxGE)

bit 1-0 **Unimplemented:** Read as ‘0’

REGISTER 21-3: TxCLK: TIMERx CLOCK REGISTER

U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	CS[4:0]				
bit 7							bit 0

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

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **CS[4:0]:** Timerx Clock Source Selection bits

CS	Timer1	Timer3	Timer5
	Clock Source	Clock Source	Clock Source
11111-10001	Reserved	Reserved	Reserved
10000	CLC4	CLC4	CLC4
01111	CLC3	CLC3	CLC3
01110	CLC2	CLC2	CLC2
01101	CLC1	CLC1	CLC1
01100	TMR5 overflow	TMR5 overflow	Reserved
01011	TMR3 overflow	Reserved	TMR3 overflow
01010	Reserved	TMR1 overflow	TMR1 overflow
01001	TMR0 overflow	TMR0 overflow	TMR0 overflow
01000	CLKREF	CLKREF	CLKREF
00111	SOSC	SOSC	SOSC
00110	MFINTOSC (32 kHz)	MFINTOSC (32 kHz)	MFINTOSC (32 kHz)
00101	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)
00100	LFINTOSC	LFINTOSC	LFINTOSC
00011	HFINTOSC	HFINTOSC	HFINTOSC
00010	Fosc	Fosc	Fosc
00001	Fosc/4	Fosc/4	Fosc/4
00000	T1CKIPPS	T3CKIPPS	T5CKIPPS

REGISTER 21-4: TxGATE: TIMERx GATE ISM REGISTER

U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	GSS[4:0]				
bit 7							bit 0

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

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **GSS[4:0]:** Timerx Gate Source Selection bits

GSS	Timer1	Timer3	Timer5
	Gate Source	Gate Source	Gate Source
11111-11011	Reserved	Reserved	Reserved
11010	CLC4_out	CLC4_out	CLC4_out
11001	CLC3_out	CLC3_out	CLC3_out
11000	CLC2_out	CLC2_out	CLC2_out
10111	CLC1_out	CLC1_out	CLC1_out
10110	ZCDOUT	ZCDOUT	ZCDOUT
10101	CMP2OUT	CMP2OUT	CMP2OUT
10100	CMP1OUT	CMP1OUT	CMP1OUT
10011	NCO1OUT	NCO1OUT	NCO1OUT
10010-10001	Reserved	Reserved	Reserved
10000	PWM8OUT	PWM8OUT	PWM8OUT
01111	PWM7OUT	PWM7OUT	PWM7OUT
01110	PWM6OUT	PWM6OUT	PWM6OUT
01101	PWM5OUT	PWM5OUT	PWM5OUT
01100	CCP4OUT	CCP4OUT	CCP4OUT
01011	CCP3OUT	CCP3OUT	CCP3OUT
01010	CCP2OUT	CCP2OUT	CCP2OUT
01001	CCP1OUT	CCP1OUT	CCP1OUT
01000	SMT1_match	SMT1_match	SMT1_match
00111	TMR6OUT (postscaled)	TMR6OUT (postscaled)	TMR6OUT (postscaled)
00110	TMR5 overflow	TMR5 overflow	Reserved
00101	TMR4OUT (postscaled)	TMR4OUT (postscaled)	TMR4OUT (postscaled)
00100	TMR3 overflow	Reserved	TMR3 overflow
00011	TMR2OUT (postscaled)	TMR2OUT (postscaled)	TMR2OUT (postscaled)
00010	Reserved	TMR1 overflow	TMR1 overflow
00001	TMR0 overflow	TMR0 overflow	TMR0 overflow
00000	Pin selected by T1GPPS	Pin selected by T3GPPS	Pin selected by T5GPPS

© 2017-2020 Microchip Technology Inc.

REGISTER 21-5: TMRxL: TIMERx LOW BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
TMRxL[7:0]							
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 7-0 **TMRxL[7:0]:** Timerx Low Byte bits

REGISTER 21-6: TMRxH: TIMERx HIGH BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
TMRxH[7:0]							
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 7-0 **TMRxH[7:0]:** Timerx High Byte bits

TABLE 21-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1/3/5 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TxCON	—	—	CKPS[1:0]		—	$\overline{\text{SYNC}}$	RD16	ON	314
TxGCON	GE	GPOL	GTM	GSPM	GO/$\overline{\text{DONE}}$	GVAL	—	—	315
TxCLK	—	—	—	CS[4:0]					316
TxGATE	—	—	—	GSS[4:0]					317
TMRxL	Least Significant Byte of the 16-bit TMR3 Register								318
TMRxH	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								318

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used by TIMER1/3/5.

22.0 TIMER2/4/6 MODULE

The Timer2/4/6 modules are 8-bit timers that can operate as free-running period counters or in conjunction with external signals that control start, run, freeze, and reset operation in One-Shot and Monostable modes of operation. Sophisticated waveform control such as pulse density modulation are possible by combining the operation of these timers with other internal peripherals such as the comparators and CCP modules. Features of the timer include:

- 8-bit timer register
- 8-bit period register
- Selectable external hardware timer resets
- Programmable prescaler (1:1 to 1:128)
- Programmable postscaler (1:1 to 1:16)
- Selectable synchronous/asynchronous operation
- Alternate clock sources
- Interrupt on period
- Three modes of operation:
  - Free Running Period
  - One-Shot
  - Monostable

See Figure 22-1 for a block diagram of Timer2. See Figure 22-2 for the clock source block diagram.

> **Note:** Three identical Timer2 modules are implemented on this device. The timers are named Timer2, Timer4, and Timer6. All references to Timer2 apply as well to Timer4 and Timer6. All references to T2PR apply as well to T4PR and T6PR.

FIGURE 22-1: TIMER2 BLOCK DIAGRAM

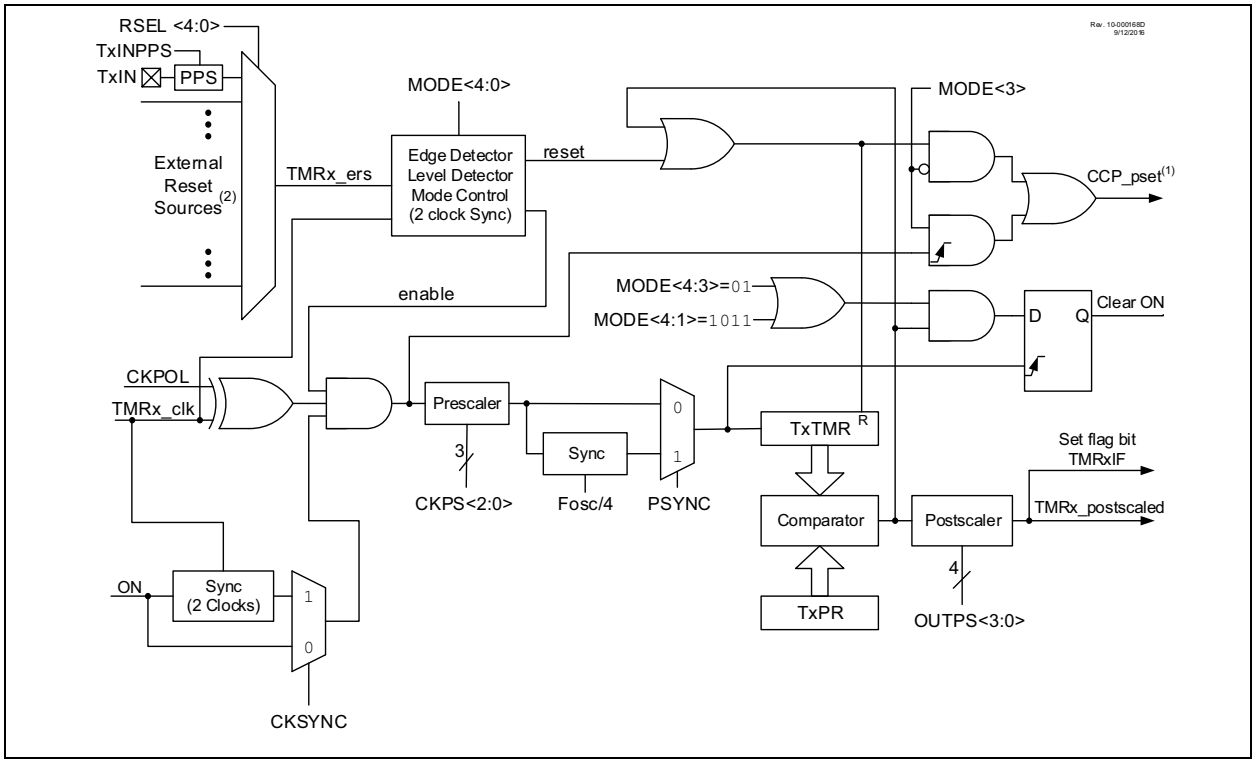
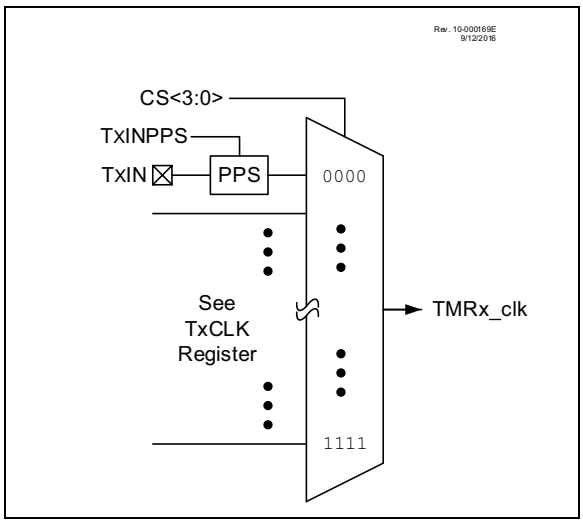


FIGURE 22-2: TIMER2 CLOCK SOURCE BLOCK DIAGRAM



22.1 Timer2 Operation

Timer2 operates in three major modes:

- Free Running Period
- One-Shot
- Monostable

Within each mode there are several options for starting, stopping, and reset. Table 22-1 lists the options.

In all modes the T2TMR count register is incremented on the rising edge of the clock signal from the programmable prescaler. When T2TMR equals T2PR then a high level is output to the postscaler counter. T2TMR is cleared on the next clock input.

An external signal from hardware can also be configured to gate the timer operation or force a T2TMR count Reset. In gate modes, the counter stops when the gate is disabled and resumes when the gate is enabled. In Reset modes the T2TMR count is reset on either the level or edge from the external source.

The T2TMR and T2PR registers are both directly readable and writable. The T2TMR register is cleared and the T2PR register initializes to FFh on any device Reset. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the T2TMR register
- a write to the TxCON register
- any device Reset
- External Reset Source event that resets the timer.

Note:	T2TMR is not cleared when TxCON is written.

22.1.1 FREE RUNNING PERIOD MODE

The value of T2TMR is compared to that of the Period register, T2PR, on each clock cycle. When the two values match, the comparator resets the value of T2TMR to 00h on the next cycle and increments the output postscaler counter. When the postscaler count equals the value in the OUTPS bits of the TxCON register, then a one clock period wide pulse occurs on the T2TMR_postscaled output, and the postscaler count is cleared.

22.1.2 ONE-SHOT MODE

The One-Shot mode is identical to the Free Running Period mode except that the ON bit is cleared and the timer is stopped when T2TMR matches T2PR and will not restart until the T2ON bit is cycled off and on. Postscaler OUTPS values other than 0 are meaningless in this mode because the timer is stopped at the first period event and the postscaler is reset when the timer is restarted.

22.1.3 MONOSTABLE MODE

Monostable modes are similar to One-Shot modes except that the ON bit is not cleared and the timer can be restarted by an external Reset event.

22.2 Timer2 Output

The Timer2 module's primary output is T2TMR_postscaled, which pulses for a single T2TMR_clk period when the postscaler counter matches the value in the OUTPS bits of the TxCON register. The T2PR postscaler is incremented each time the T2TMR value matches the T2PR value. This signal can be selected as an input to several other input modules.

Timer2 is also used by the CCP module for pulse generation in PWM mode. Both the actual T2TMR value as well as other internal signals are sent to the CCP module to properly clock both the period and pulse width of the PWM signal. See **Section 23.0 "Capture/Compare/PWM Module"** for more details on setting up Timer2 for use with the CCP, as well as the timing diagrams in **Section 22.5 "Operation Examples"** for examples of how the varying Timer2 modes affect CCP PWM output.

22.3 External Reset Sources

In addition to the clock source, the Timer2 also takes in an external Reset source. This external Reset source is selected for Timer2, Timer4, and Timer6 with the T2RST, T4RST, and T6RST registers, respectively. This source can control starting and stopping of the timer, as well as resetting the timer, depending on which mode the timer is in. The mode of the timer is controlled by the MODE bits of the T2HLT register. Edge Triggered modes require six Timer clock periods between external triggers. Level Triggered modes require the triggering level to be at least three Timer clock periods long. External triggers are ignored while in Debug Freeze mode.

TABLE 22-1: TIMER2 OPERATING MODES

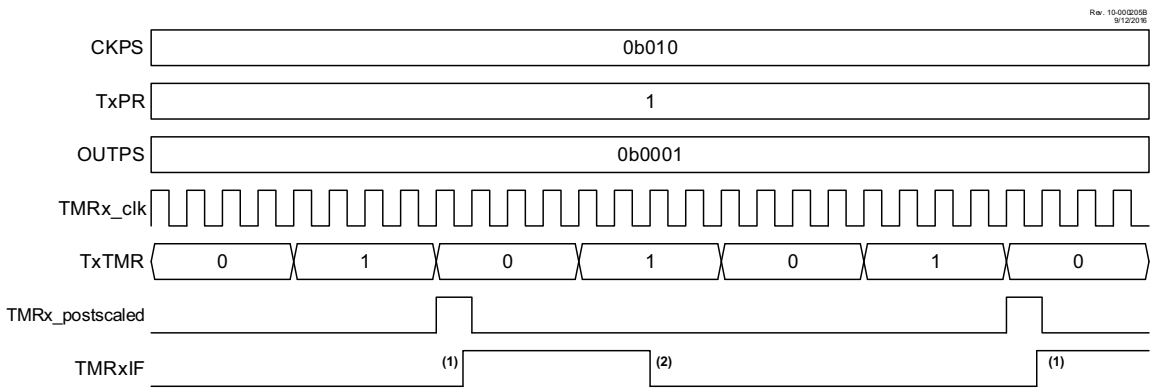
Mode	MODE[4:0]		Output Operation	Operation	Timer Control		
	[4:3]	[2:0]			Start	Reset	Stop
Free Running Period	00	000	Period Pulse	Software gate (Figure 22-4)	ON = 1	—	ON = 0
		001		Hardware gate, active-high (Figure 22-5)	ON = 1 & TMRx_ers = 1	—	ON = 0 or TMRx_ers = 0
		010		Hardware gate, active-low	ON = 1 & TMRx_ers = 0	—	ON = 0 or TMRx_ers = 1
		011	Period Pulse with Hardware Reset	Rising or Falling Edge Reset	ON = 1	TMRx_ers ↕	ON = 0
		100		Rising Edge Reset (Figure 22-6)		TMRx_ers ↑	
		101		Falling Edge Reset		TMRx_ers ↓	
		110		Low Level Reset		TMRx_ers = 0	ON = 0 or TMRx_ers = 0
		111		High Level Reset (Figure 22-7)		TMRx_ers = 1	ON = 0 or TMRx_ers = 1
One-shot	01	000	One-Shot	Software Start (Figure 22-8)	ON = 1	—	ON = 0 or Next clock after TMRx = PRx **(Note 2)**
		001	Edge Triggered Start (Note 1)	Rising Edge Start (Figure 22-9)	ON = 1 & TMRx_ers ↑	—	
		010		Falling Edge Start	ON = 1 & TMRx_ers ↓	—	
		011		Any Edge Start	ON = 1 & TMRx_ers ↕	—	
		100	Edge Triggered Start and Hardware Reset (Note 1)	Rising Edge Start & Rising Edge Reset (Figure 22-10)	ON = 1 & TMRx_ers ↑	TMRx_ers ↑	
		101		Falling Edge Start & Falling Edge Reset	ON = 1 & TMRx_ers ↓	TMRx_ers ↓	
		110		Rising Edge Start & Low Level Reset (Figure 22-11)	ON = 1 & TMRx_ers ↑	TMRx_ers = 0	
		111		Falling Edge Start & High Level Reset	ON = 1 & TMRx_ers ↓	TMRx_ers = 1	
Monostable	10	000	Reserved				
		001	Edge Triggered Start (Note 1)	Rising Edge Start (Figure 22-12)	ON = 1 & TMRx_ers ↑	—	ON=0 or Next clock after TxTMR = TxPR **(Note 3)**
		010		Falling Edge Start	ON = 1 & TMRx_ers ↓	—	
		011		Any Edge Start	ON = 1 & TMRx_ers ↕	—	
Reserved		100	Reserved				
Reserved		101	Reserved				
One-shot		110	Level Triggered Start and Hardware Reset	High Level Start & Low Level Reset (Figure 22-13)	ON = 1 & TMRx_ers = 1	TMRx_ers = 0	ON = 0 or Held in Reset **(Note 2)**
		111		Low Level Start & High Level Reset	ON = 1 & TMRx_ers = 0	TMRx_ers = 1	
Reserved	11	xxx	Reserved				

Note 1: If ON = 0 then an edge is required to restart the timer after ON = 1.
2: When TxTMR = TxPR then the next clock clears ON and stops TxTMR at 00h.
3: When TxTMR = TxPR then the next clock stops TxTMR at 00h but does not clear ON.

22.4 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches one of 16 postscale options (from 1:1 through 1:16), which is selected with the postscaler control bits, OUTPS of the T2CON register. The interrupt is enabled by setting the T2TMR Interrupt Enable bit, TMR2IE, of the respective PIE register. The interrupt timing is illustrated in Figure 22-3.

FIGURE 22-3: TIMER2 PRESCALER, POSTSCALER, AND INTERRUPT TIMING DIAGRAM



Note 1: Setting the interrupt flag is synchronized with the instruction clock. Synchronization may take as many as 2 instruction cycles
2: Cleared by software.

22.5 Operation Examples

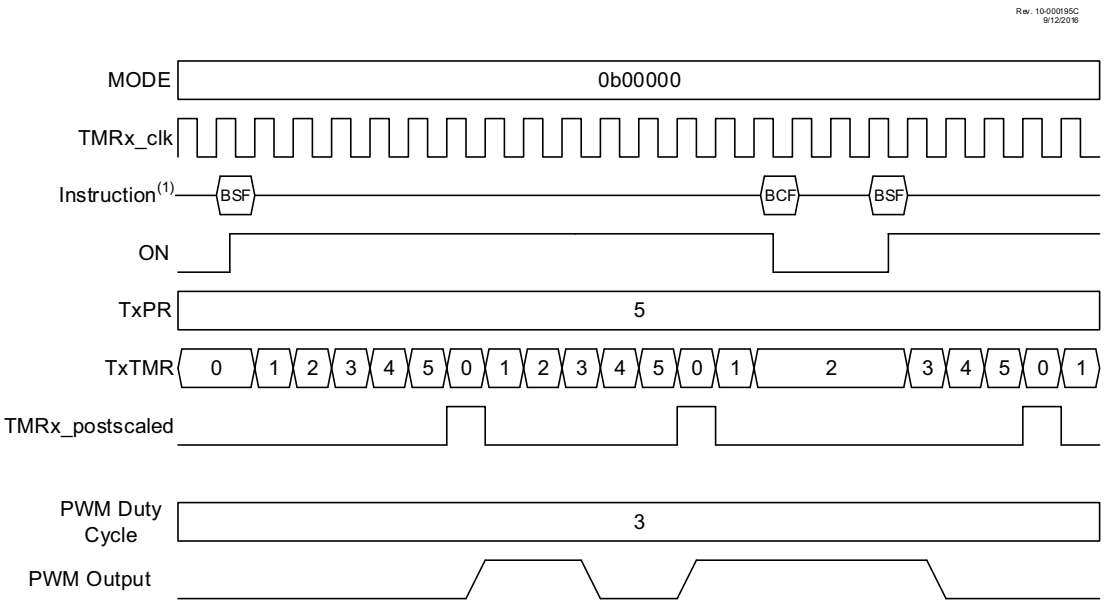
Unless otherwise specified, the following notes apply to the following timing diagrams:

- Both the prescaler and postscaler are set to 1:1 (both the CKPS and OUTPS bits in the T2CON register are cleared).
- The diagrams illustrate any clock except FOSC/4 and show clock-sync delays of at least two full cycles for both ON and T2TMR_ers. When using FOSC/4, the clock-sync delay is at least one instruction period for T2TMR_ers; ON applies in the next instruction period.
- ON and T2TMR_ers are somewhat generalized, and clock-sync delays may produce results that are slightly different than illustrated.
- The PWM Duty Cycle and PWM output are illustrated assuming that the timer is used for the PWM function of the CCP module as described in **Section 23.0 “Capture/Compare/PWM Module”** and **Section 24.0 “Pulse-Width Modulation (PWM)”**. The signals are not a part of the T2TMR module.

22.5.1 SOFTWARE GATE MODE

The timer increments with each clock input when ON = 1 and does not increment when ON = 0. When the T2TMR count equals the T2PR period count the timer resets on the next clock and continues counting from 0. Operation with the ON bit software controlled is illustrated in Figure 22-4. With T2PR = 5, the counter advances until T2TMR = 5, and goes to zero with the next clock.

FIGURE 22-4: SOFTWARE GATE MODE TIMING DIAGRAM



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

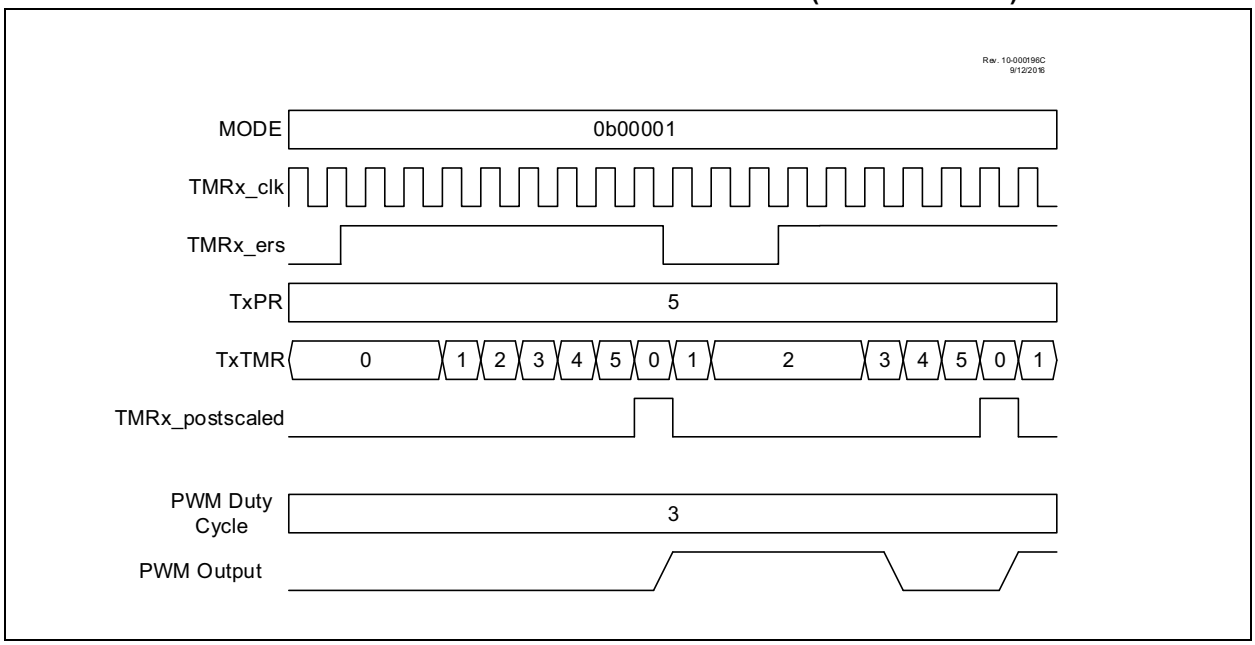
22.5.2 HARDWARE GATE MODE

The Hardware Gate modes operate the same as the Software Gate mode except the T2TMR_ers external signal can also gate the timer. When used with the CCP the gating extends the PWM period. If the timer is stopped when the PWM output is high, then the duty cycle is also extended.

When MODE[4:0] = `00001`, then the timer is stopped when the external signal is high. When MODE[4:0] = `00010`, then the timer is stopped when the external signal is low.

Figure 22-5 illustrates the Hardware Gating mode for MODE[4:0] = `00001` in which a high input level starts the counter.

FIGURE 22-5: HARDWARE GATE MODE TIMING DIAGRAM (MODE = `00001`)



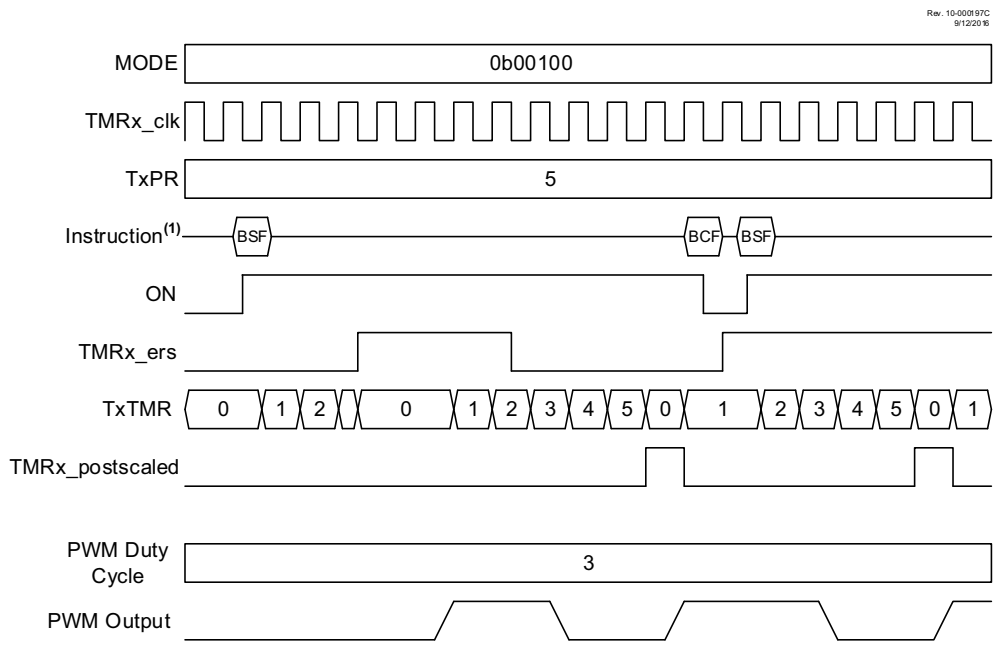
22.5.3 EDGE-TRIGGERED HARDWARE LIMIT MODE

In Hardware Limit mode the timer can be reset by the TMRx_ers external signal before the timer reaches the period count. Three types of Resets are possible:

- Reset on rising or falling edge (MODE[4:0] = `00011`)
- Reset on rising edge (MODE[4:0] = `0010`)
- Reset on falling edge (MODE[4:0] = `00101`)

When the timer is used in conjunction with the CCP in PWM mode then an early Reset shortens the period and restarts the PWM pulse after a two clock delay. Refer to Figure 22-6.

FIGURE 22-6: EDGE TRIGGERED HARDWARE LIMIT MODE TIMING DIAGRAM (MODE=`00100`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

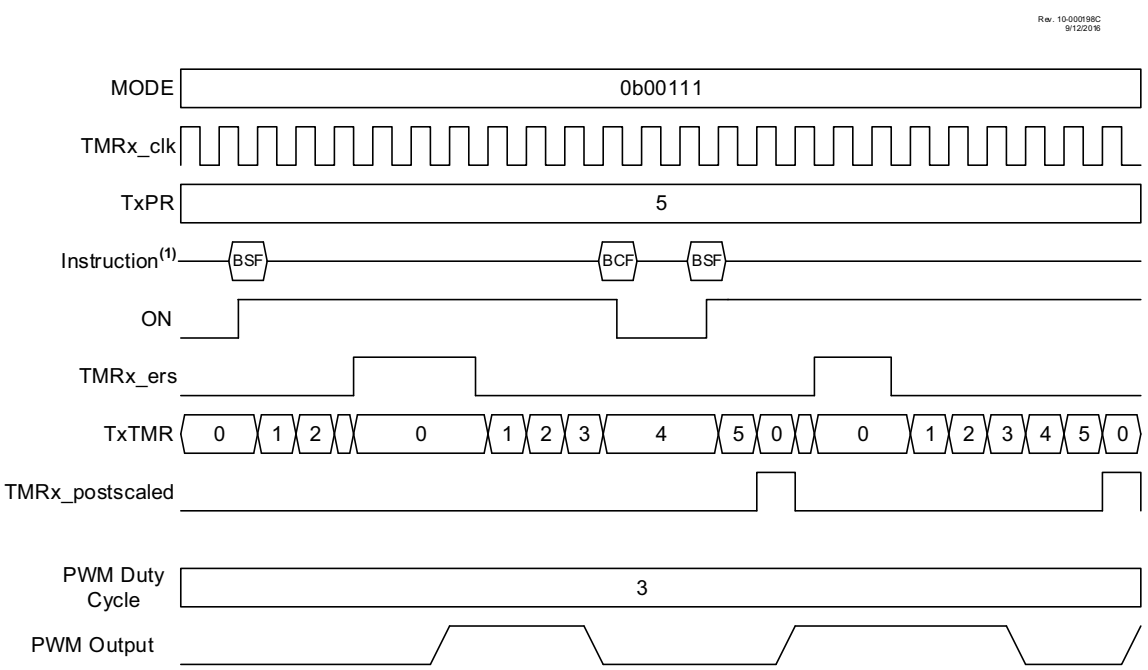
22.5.4 LEVEL-TRIGGERED HARDWARE LIMIT MODE

In the level triggered Hardware Limit Timer modes the counter is reset by high or low levels of the external signal TMR2_ers, as shown in Figure 22-7. Selecting MODE[4:0] = `00110` will cause the timer to reset on a low level external signal. Selecting MODE[4:0] = `00111` will cause the timer to reset on a high level external signal. In the example, the counter is reset while TMR2_ers = `1`. ON is controlled by BSF and BCF instructions. When ON=`0` the external signal is ignored.

When the CCP uses the timer as the PWM time base then the PWM output will be set high when the timer starts counting and then set low only when the timer count matches the CCPRx value. The timer is reset when either the timer count matches the T2PR value or two clock periods after the external Reset signal goes true and stays true.

The timer starts counting, and the PWM output is set high, on either the clock following the T2PR match or two clocks after the external Reset signal relinquishes the Reset. The PWM output will remain high until the timer counts up to match the CCPRx pulse width value. If the external Reset signal goes true while the PWM output is high then the PWM output will remain high until the Reset signal is released allowing the timer to count up to match the CCPRx value.

FIGURE 22-7: LEVEL TRIGGERED HARDWARE LIMIT MODE TIMING DIAGRAM (MODE = `00111`)



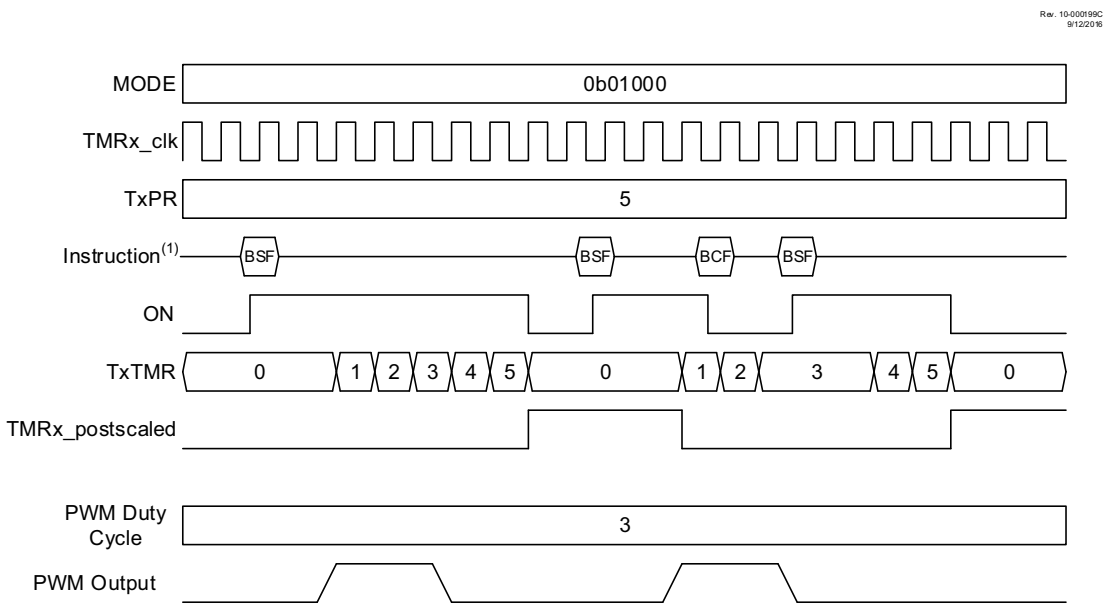
Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

22.5.5 SOFTWARE START ONE-SHOT MODE

In One-Shot mode, the timer resets and the ON bit is cleared when the timer value matches the T2PR period value. The ON bit must be set by software to start another timer cycle. Setting MODE[4:0] = 01000 selects One-Shot mode which is illustrated in Figure 22-8. In the example, ON is controlled by BSF and BCF instructions. In the first case, a BSF instruction sets ON and the counter runs to completion and clears ON. In the second case, a BSF instruction starts the cycle, BCF/BSF instructions turn the counter off and on during the cycle, and then it runs to completion.

When One-Shot mode is used in conjunction with the CCP PWM operation, the PWM pulse drive starts concurrent with setting the ON bit. Clearing the ON bit while the PWM drive is active will extend the PWM drive. The PWM drive will terminate when the timer value matches the CCPRx pulse width value. The PWM drive will remain off until software sets the ON bit to start another cycle. If software clears the ON bit after the CCPRx match but before the T2PR match then the PWM drive will be extended by the length of time the ON bit remains cleared. Another timing cycle can only be initiated by setting the ON bit after it has been cleared by a T2PR period count match.

FIGURE 22-8: SOFTWARE START ONE-SHOT MODE TIMING DIAGRAM (MODE = 01000)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

22.5.6 EDGE-TRIGGERED ONE-SHOT MODE

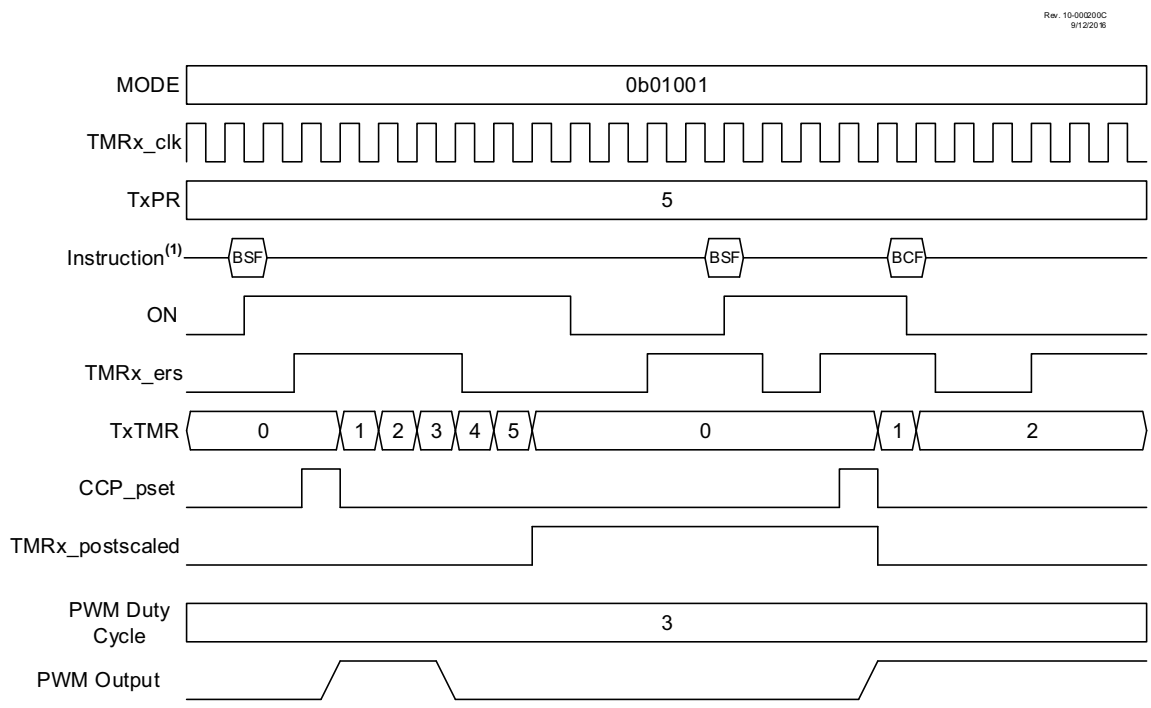
The Edge-Triggered One-Shot modes start the timer on an edge from the external signal input, after the ON bit is set, and clear the ON bit when the timer matches the T2PR period value. The following edges will start the timer:

- Rising edge (MODE[4:0] = `01001`)
- Falling edge (MODE[4:0] = `01010`)
- Rising or Falling edge (MODE[4:0] = `01011`)

If the timer is halted by clearing the ON bit then another TMRx_ers edge is required after the ON bit is set to resume counting. Figure 22-9 illustrates operation in the rising edge One-Shot mode.

When Edge-Triggered One-Shot mode is used in conjunction with the CCP then the edge-trigger will activate the PWM drive and the PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated when the timer halts at the T2PR period count match.

FIGURE 22-9: EDGE TRIGGERED ONE-SHOT MODE TIMING DIAGRAM (MODE = `01001`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

22.5.7 EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE

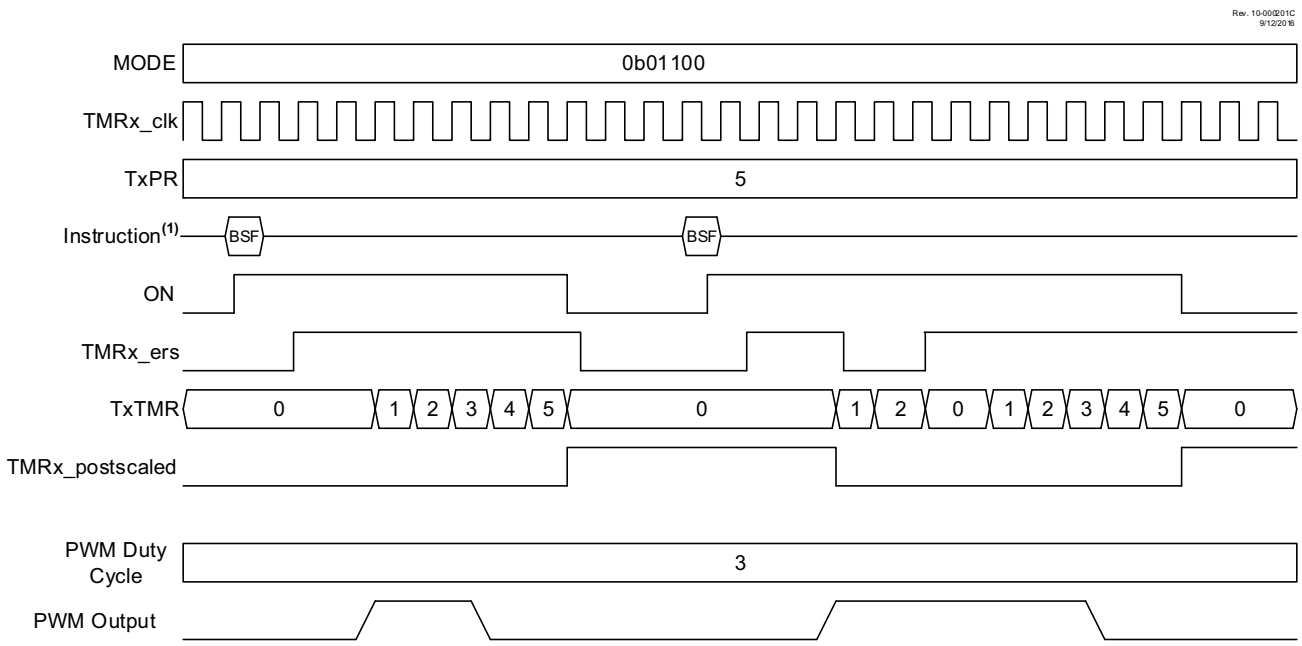
In Edge-Triggered Hardware Limit One-Shot modes, the timer starts on the first external signal edge after the ON bit is set and resets on all subsequent edges. Only the first edge after the ON bit is set is needed to start the timer. The counter will resume counting automatically two clocks after all subsequent external Reset edges. Edge triggers are as follows:

- Rising edge Start and Reset (MODE[4:0] = `01100`)
- Falling edge Start and Reset (MODE[4:0] = `01101`)

The timer resets and clears the ON bit when the timer value matches the T2PR period value. External signal edges will have no effect until after software sets the ON bit. Figure 22-10 illustrates the rising edge hardware limit one-shot operation.

When this mode is used in conjunction with the CCP then the first starting edge trigger, and all subsequent Reset edges, will activate the PWM drive. The PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated until the timer halts at the T2PR period match unless an external signal edge resets the timer before the match occurs.

FIGURE 22-10: EDGE TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = `01100`))



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

© 2017-2020 Microchip Technology Inc.

22.5.8 LEVEL RESET, EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODES

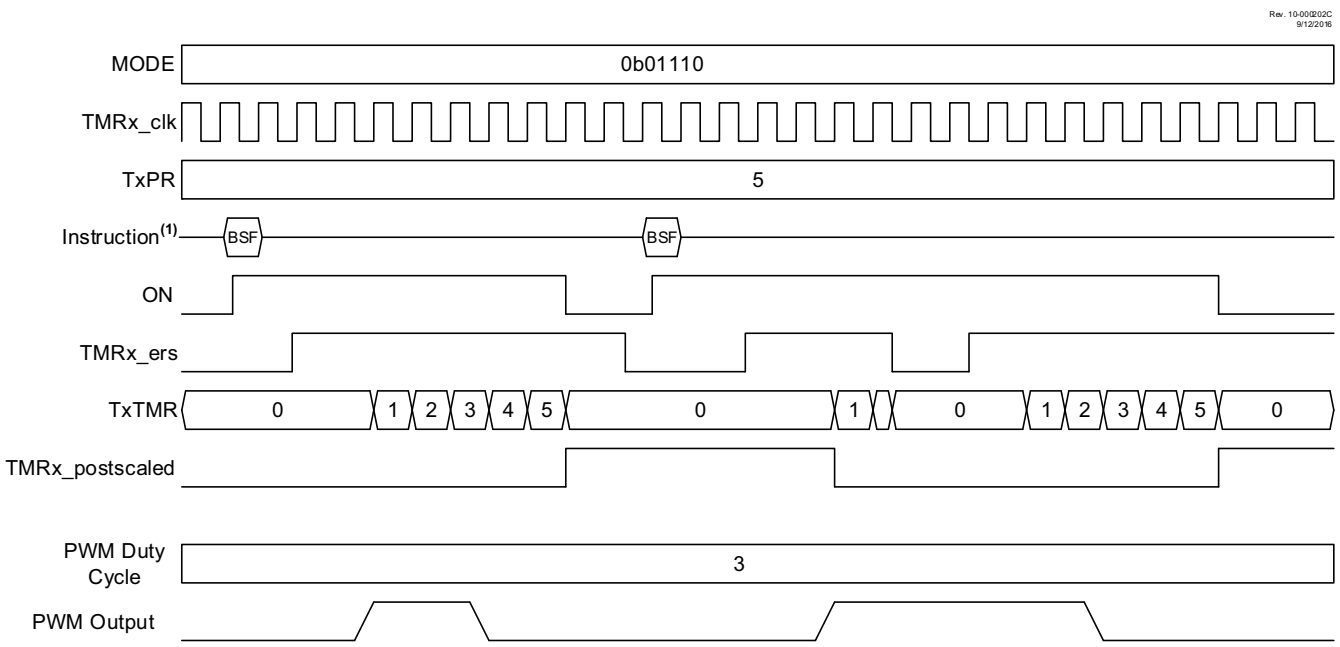
In Level Triggered One-Shot mode, the timer count is reset on the external signal level and starts counting on the rising/falling edge of the transition from reset level to the active level while the ON bit is set. Reset levels are selected as follows:

- Low reset level (MODE[4:0] = `01110`)
- High reset level (MODE[4:0] = `01111`)

When the timer count matches the T2PR period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a T2PR match or by software control a new external signal edge is required after the ON bit is set to start the counter.

When Level Triggered Reset One-Shot mode is used in conjunction with the CCP PWM operation, the PWM drive goes active with the external signal edge that starts the timer. The PWM drive goes inactive when the timer count equals the CCPRx pulse-width count. The PWM drive does not go active when the timer count clears at the T2PR period count match.

FIGURE 22-11: LOW LEVEL RESET, EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = `01110`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

© 2017-2020 Microchip Technology Inc.

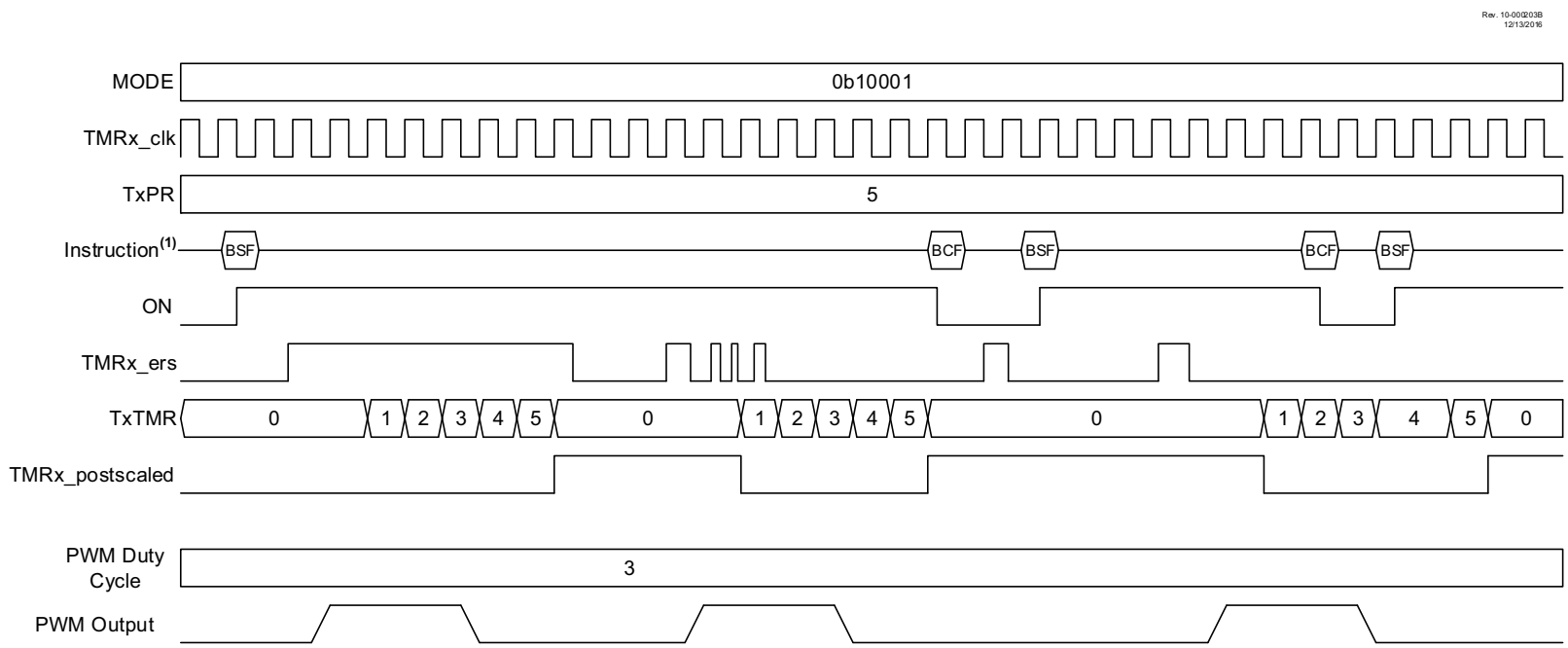
22.5.9 EDGE-TRIGGERED MONOSTABLE MODES

The Edge-Triggered Monostable modes start the timer on an edge from the external Reset signal input, after the ON bit is set, and stop incrementing the timer when the timer matches the T2PR period value. The following edges will start the timer:

- Rising edge (MODE[4:0] = `10001`)
- Falling edge (MODE[4:0] = `10010`)
- Rising or Falling edge (MODE[4:0] = `10011`)

When an Edge-Triggered Monostable mode is used in conjunction with the CCP PWM operation the PWM drive goes active with the external Reset signal edge that starts the timer, but will not go active when the timer matches the T2PR value. While the timer is incrementing, additional edges on the external Reset signal will not affect the CCP PWM.

FIGURE 22-12: RISING EDGE-TRIGGERED MONOSTABLE MODE TIMING DIAGRAM (MODE = `10001`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

© 2017-2020 Microchip Technology Inc.

22.5.10 LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODES

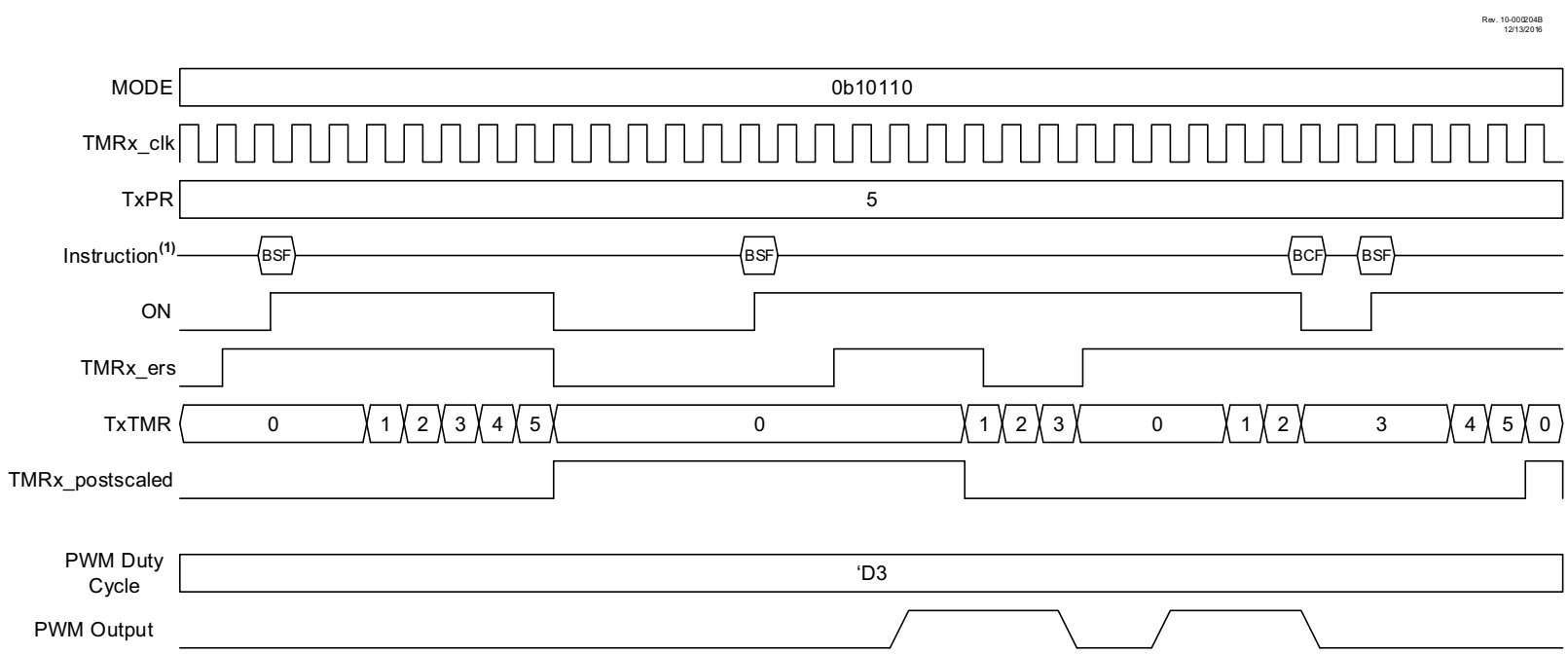
The Level Triggered Hardware Limit One-Shot modes hold the timer in Reset on an external Reset level and start counting when both the ON bit is set and the external signal is not at the Reset level. If one of either the external signal is not in reset or the ON bit is set then the other signal being set/made active will start the timer. Reset levels are selected as follows:

- Low reset level (MODE[4:0] = `10110`)
- High reset level (MODE[4:0] = `10111`)

When the timer count matches the T2PR period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a T2PR match or by software control, the timer will stay in Reset until both the ON bit is set and the external signal is not at the Reset level.

When Level Triggered Hardware Limit One-Shot modes are used in conjunction with the CCP PWM operation, the PWM drive goes active with either the external signal edge or the setting of the ON bit, whichever of the two starts the timer.

FIGURE 22-13: LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = `10110`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

© 2017-2020 Microchip Technology Inc.

22.6 Timer2 Operation During Sleep

When PSYNC = 1, Timer2 cannot be operated while the processor is in Sleep mode. The contents of the T2TMR and T2PR registers will remain unchanged while processor is in Sleep mode.

When PSYNC = 0, Timer2 will operate in Sleep as long as the clock source selected is also still running. Selecting the LFINTOSC, MFINTOSC, or HFINTOSC oscillator as the timer clock source will keep the selected oscillator running during Sleep.

22.7 Register Definitions: Timer2/4/6 Control

Long bit name prefixes for the Timer2/4/6 peripherals are shown in Table 22-2. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

TABLE 22-2: OPERATING MODES

Peripheral	Bit Name Prefix
Timer2	T2
Timer4	T4
Timer6	T6

REGISTER 22-1: TxCLK: TIMERx CLOCK SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CS[3:0]			
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 7-4 **Unimplemented:** Read as '0'

bit 3-0 **CS[3:0]:** Timerx Clock Selection bits

CS[3:0]	T2TMR	TMR4	TMR6
	Clock Source	Clock Source	Clock Source
1111	Reserved	Reserved	Reserved
1110	CLC4_out	CLC4_out	CLC4_out
1101	CLC3_out	CLC3_out	CLC3_out
1100	CLC2_out	CLC2_out	CLC2_out
1011	CLC1_out	CLC1_out	CLC1_out
1010	ZCD_OUT	ZCD_OUT	ZCD_OUT
1001	NCO1OUT	NCO1OUT	NCO1OUT
1000	CLKREF_OUT	CLKREF_OUT	CLKREF_OUT
0111	SOSC	SOSC	SOSC
0110	MFINTOSC (32 kHz)	MFINTOSC (32 kHz)	MFINTOSC (32 kHz)
0101	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)
0100	LFINTOSC	LFINTOSC	LFINTOSC
0011	HFINTOSC	HFINTOSC	HFINTOSC
0010	Fosc	Fosc	Fosc
0001	Fosc/4	Fosc/4	Fosc/4
0000	Pin selected by T2INPPS	Pin selected by T4INPPS	Pin selected by T6INPPS

REGISTER 22-2: TxRST: TIMER2 EXTERNAL RESET SIGNAL SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	RSEL[4:0]				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **RSEL[4:0]:** Timer2 External Reset Signal Source Selection bits

RSEL[4:0]	T2TMR	TMR4	TMR6
	Reset Source	Reset Source	Reset Source
11111-11001	Reserved	Reserved	Reserved
11000	UART2_tx_edge	UART2_tx_edge	UART2_tx_edge
10111	UART2_rx_edge	UART2_rx_edge	UART2_rx_edge
10110	UART1_tx_edge	UART1_tx_edge	UART1_tx_edge
10101	UART1_rx_edge	UART1_rx_edge	UART1_rx_edge
10100	CLC4_out	CLC4_out	CLC4_out
10011	CLC3_out	CLC3_out	CLC3_out
10010	CLC2_out	CLC2_out	CLC2_out
10001	CLC1_out	CLC1_out	CLC1_out
10000	ZCD_OUT	ZCD_OUT	ZCD_OUT
01111	CMP2OUT	CMP2OUT	CMP2OUT
01110	CMP1OUT	CMP1OUT	CMP1OUT
01101-01100	Reserved	Reserved	Reserved
01011	PWM8OUT	PWM8OUT	PWM8OUT
01010	PWM7OUT	PWM7OUT	PWM7OUT
01001	PWM6OUT	PWM6OUT	PWM6OUT
01000	PWM5OUT	PWM5OUT	PWM5OUT
00111	CCP4OUT	CCP4OUT	CCP4OUT
00110	CCP3OUT	CCP3OUT	CCP3OUT
00101	CCP2OUT	CCP2OUT	CCP2OUT
00100	CCP1OUT	CCP1OUT	CCP1OUT
00011	TMR6 postscaled	TMR6 postscaled	Reserved
00010	TMR4 postscaled	Reserved	TMR4 postscaled
00001	Reserved	T2TMR postscaled	T2TMR postscaled
00000	Pin selected by T2INPPS	Pin selected by T4INPPS	Pin selected by T6INPPS

REGISTER 22-3: TxTMR: TIMERx COUNTER REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TMRx[7:0]							
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 7-0 **TMRx[7:0]:** Timerx Counter bits

REGISTER 22-4: TxPR: TIMERx PERIOD REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
PRx[7:0]							
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 7-0 **PRx[7:0]:** Timerx Period Register bits

REGISTER 22-5: TxCON: TIMERx CONTROL REGISTER

R/W/HC-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ON	CKPS[2:0]			OUTPS[3:0]			
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	HC = Bit is cleared by hardware

bit 7 **ON:** Timerx On bit(1)
`1` = Timerx is On
`0` = Timerx is Off: all counters and state machines are reset

bit 6-4 **CKPS[2:0]:** Timerx-type Clock Prescale Select bits
`111` = 1:128 Prescaler
`110` = 1:64 Prescaler
`101` = 1:32 Prescaler
`100` = 1:16 Prescaler
`011` = 1:8 Prescaler
`010` = 1:4 Prescaler
`001` = 1:2 Prescaler
`000` = 1:1 Prescaler

bit 3-0 **OUTPS[3:0]:** Timerx Output Postscaler Select bits
`1111` = 1:16 Postscaler
`1110` = 1:15 Postscaler
`1101` = 1:14 Postscaler
`1100` = 1:13 Postscaler
`1011` = 1:12 Postscaler
`1010` = 1:11 Postscaler
`1001` = 1:10 Postscaler
`1000` = 1:9 Postscaler
`0111` = 1:8 Postscaler
`0110` = 1:7 Postscaler
`0101` = 1:6 Postscaler
`0100` = 1:5 Postscaler
`0011` = 1:4 Postscaler
`0010` = 1:3 Postscaler
`0001` = 1:2 Postscaler
`0000` = 1:1 Postscaler

Note 1: In certain modes, the ON bit will be auto-cleared by hardware. See **Section 22.1.2 “One-Shot Mode”**.

REGISTER 22-6: TxHLT: TIMERx HARDWARE LIMIT CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
PSYNC	CKPOL	CKSYNC	MODE[4:0]				
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 **PSYNC:** Timerx Prescaler Synchronization Enable bit[1, 2]
1 = TxTMR Prescaler Output is synchronized to Fosc/4
0 = TxTMR Prescaler Output is not synchronized to Fosc/4

bit 6 **CKPOL:** Timerx Clock Polarity Selection bit[3]
1 = Falling edge of input clock clocks timer/prescaler
0 = Rising edge of input clock clocks timer/prescaler

bit 5 **CKSYNC:** Timerx Clock Synchronization Enable bit[4, 5]
1 = ON register bit is synchronized to T2TMR_clk input
0 = ON register bit is not synchronized to T2TMR_clk input

bit 4-0 **MODE[4:0]:** Timerx Control Mode Selection bits[6, 7]
See Table 22-1 for all operating modes.

Note 1: Setting this bit ensures that reading TxTMR will return a valid data value.
2: When this bit is ‘1’, Timer2 cannot operate in Sleep mode.
3: CKPOL should not be changed while ON = 1.
4: Setting this bit ensures glitch-free operation when the ON is enabled or disabled.
5: When this bit is set then the timer operation will be delayed by two TxTMR input clocks after the ON bit is set.
6: Unless otherwise indicated, all modes start upon ON = 1 and stop upon ON = 0 (stops occur without affecting the value of TxTMR).
7: When TxTMR = TxPR, the next clock clears TxTMR, regardless of the operating mode.

TABLE 22-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
TxPR	Timer2 Module Period Register								321*
TxTMR	Holding Register for the 8-bit T2TMR Register								321*
TxCON	ON	CKPS[2:0]			OUTPS[3:0]				339
TxCLK	—	—	—	—	—	CS[2:0]			336
TxRST	—	—	—	—	RSEL[3:0]				337
TxHLT	$\overline{\text{PSYNC}}$	CPOL	$\overline{\text{CSYNC}}$	MODE[4:0]					340

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for Timer2 module.
* Page provides register information.

23.0 CAPTURE/COMPARE/PWM MODULE

The Capture/Compare/PWM module is a peripheral that allows the user to time and control different events, and to generate Pulse-Width Modulation (PWM) signals. In Capture mode, the peripheral allows the timing of the duration of an event. The Compare mode allows the user to trigger an external event when a predetermined amount of time has expired. The PWM mode can generate pulse-width modulated signals of varying frequency and duty cycle.

This family of devices contains four standard Capture/Compare/PWM modules (CCP1, CCP2, CCP3 and CCP4). Each individual CCP module can select the timer source that controls the module. Each module has an independent timer selection which can be accessed using the CxTSEL bits in the CCPTMRS register (Register 23-2). The default timer selection is TMR1 when using Capture/Compare mode and TMR2 when using PWM mode in the CCPx module.

Please note that the Capture/Compare mode operation is described with respect to TMR1 and the PWM mode operation is described with respect to TMR2 in the following sections.

The Capture and Compare functions are identical for all CCP modules.

Note 1: In devices with more than one CCP module, it is very important to pay close attention to the register names used. A number placed after the module acronym is used to distinguish between separate modules. For example, the CCP1CON and CCP2CON control the same operational aspects of two completely different CCP modules.

2: Throughout this section, generic references to a CCP module in any of its operating modes may be interpreted as being equally applicable to CCPx module. Register names, module signals, I/O pins, and bit names may use the generic designator ‘x’ to indicate the use of a numeral to distinguish a particular module, when required.

23.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (CCPxCON), a capture input selection register (CCPxCAP) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte).

23.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers 1 through 6 that vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in Table 23-1.

TABLE 23-1: CCP MODE – TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1, Timer3 or Timer5
Compare	
PWM	Timer2, Timer4 or Timer6

The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the CCPTMRS register (see Register 23-2) All of the modules may be active at once and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time.

23.1.2 OPEN-DRAIN OUTPUT OPTION

When operating in Output mode (the Compare or PWM modes), the drivers for the CCPx pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor and allows the output to communicate with external circuits without the need for additional level shifters.

Note: The voltage on the pin should not exceed the maximum recommended voltage level for that pin.

23.2 Capture Mode

Capture mode makes use of the 16-bit Timer1 resource. When an event occurs on the capture source, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMRxH:TMRxL register pair, respectively. An event is defined as one of the following and is configured by the MODE[3:0] bits of the CCPxCON register:

- Every falling edge of CCPx input
- Every rising edge of CCPx input
- Every 4th rising edge of CCPx input
- Every 16th rising edge of CCPx input
- Every edge of CCPx input (rising or falling)

When a capture is made, the Interrupt Request Flag bit CCPxIF of the respective PIR register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH:CCPRxL register pair is read, the old captured value is overwritten by the new captured value.

Note:	If an event occurs during a 2-byte read, the high and low-byte data will be from different events. It is recommended while reading the CCPRxH:CCPRxL register pair to either disable the module or read the register pair twice for data integrity.

Figure 23-1 shows a simplified diagram of the capture operation.

23.2.1 CAPTURE SOURCES

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

Note:	If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

The capture source is selected by configuring the CTS[2:0] bits of the CCPxCAP register. Refer to CCPxCAP register (Register 23-3) for a list of sources that can be selected.

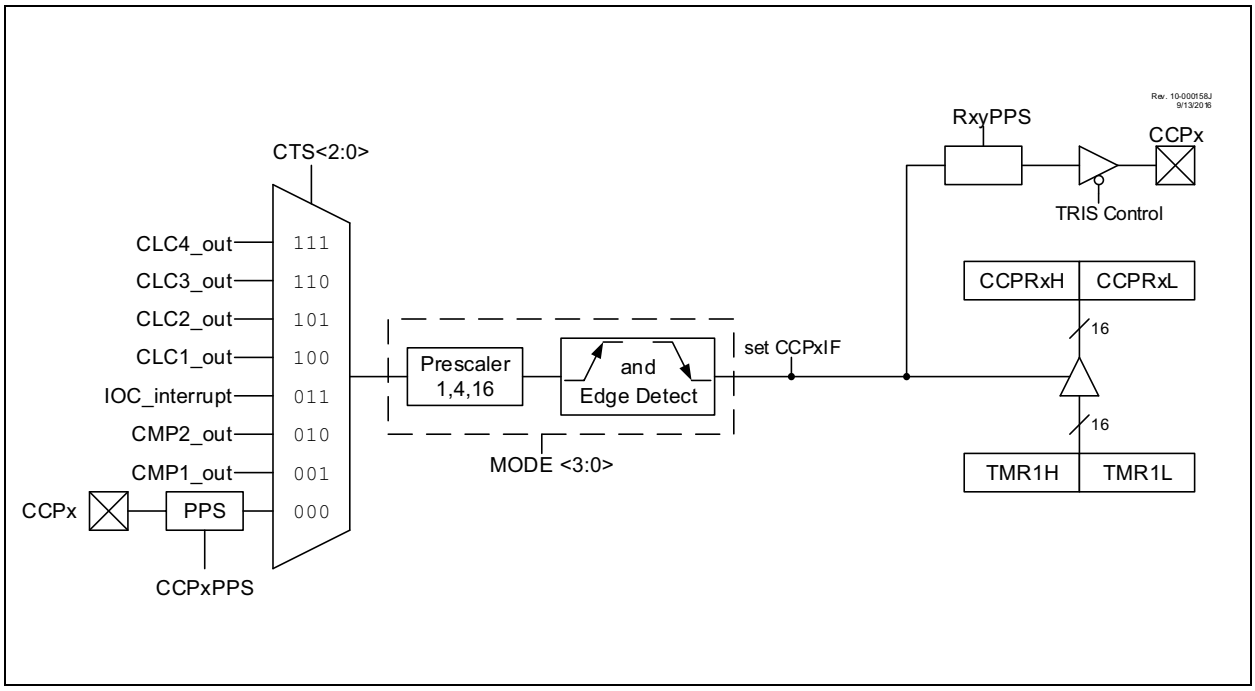
23.2.2 TIMER1 MODE RESOURCE

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

- See **Section 21.0 "Timer1/3/5 Module with Gate Control"** for more information on configuring Timer1.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

FIGURE 23-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



23.2.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE Interrupt Priority bit of the respective PIE register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the respective PIR register following any change in Operating mode.

23.2.4 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock (FOSC/4), or by an external clock source.

When Timer1 is clocked by FOSC/4, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep as long as the clock source for Timer1 is active in Sleep.

23.3 Compare Mode

Compare mode makes use of the 16-bit Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMRxH:TMRxL register pair. When a match occurs, one of the following events can occur:

- Toggle the CCPx output, clear TMRx
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Pulse output(1)
- Pulse output, clear TMRx

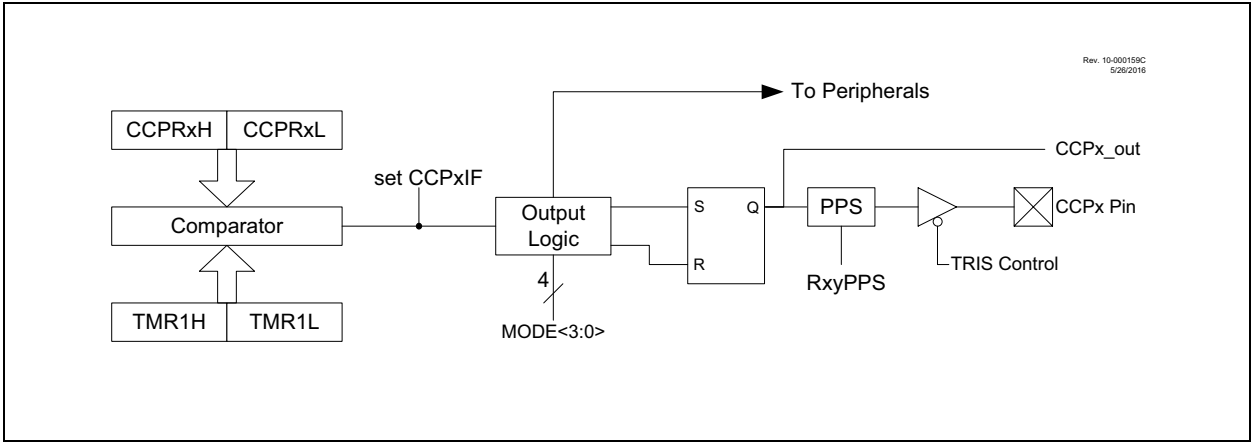
Note 1: The pulse output goes high at the rising edge of the timer clock where the CCP match occurs and lasts until the rising edge of the next timer clock. The pulse output also goes low if the timer is written to before the second clock edge occurs.

The action on the pin is based on the value of the MODE[3:0] control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set, and an ADC conversion can be triggered, if selected.

All Compare modes can generate an interrupt and trigger an ADC conversion. When MODE = `0b0001` or `0b1011`, the CCP resets the TMR register pair.

Figure 23-2 shows a simplified diagram of the compare operation.

FIGURE 23-2: COMPARE MODE OPERATION BLOCK DIAGRAM



23.3.1 CCPx PIN CONFIGURATION

The software must configure the CCPx pin as an output by clearing the associated TRIS bit and defining the appropriate output pin through the RxyPPS registers. See **Section 17.0 "Peripheral Pin Select (PPS) Module"** for more details.

Note:	Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

23.3.2 TIMER1 MODE RESOURCE

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See **Section 21.0 "Timer1/3/5 Module with Gate Control"** for more information on configuring Timer1.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, TImer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

23.3.3 AUTO-CONVERSION TRIGGER

All CCPx modes set the CCP interrupt flag (CCPxIF). When this flag is set and a match occurs, an auto-conversion trigger can take place if the CCP module is selected as the conversion trigger source.

Refer to **Section 36.2.5 "Auto-Conversion Trigger"** for more information.

Note:	Removing the match condition by changing the contents of the CCPRxH and CCPRxL register pair, between the clock edge that generates the Auto-conversion Trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring

23.3.4 COMPARE DURING SLEEP

Since FOSC is shut down during Sleep mode, the Compare mode will not function properly during Sleep, unless the timer is running. The device will wake on interrupt (if enabled).

23.4 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully ON and fully OFF states. The PWM signal resembles a square wave where the high portion of the signal is considered the ON state and the low portion of the signal is considered the OFF state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse-width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 23-3 shows a typical waveform of the PWM signal.

23.4.1 STANDARD PWM OPERATION

The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- T2PR registers
- T2CON registers
- CCPRxL and CCPRxH registers
- CCPxCON registers

It is required to have FOSC/4 as the clock input to TMR2/4/6 for correct PWM operation. Figure 23-4 shows a simplified block diagram of PWM operation.

Note:	The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

FIGURE 23-3: CCP PWM OUTPUT SIGNAL

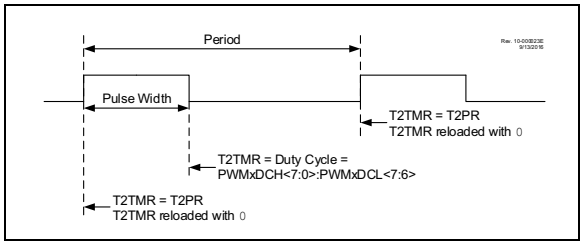
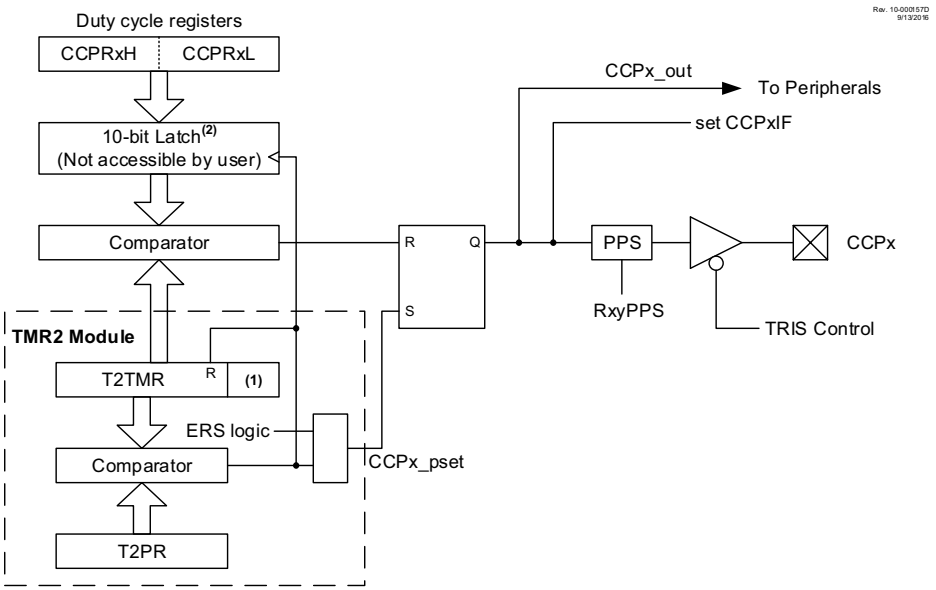


FIGURE 23-4: SIMPLIFIED PWM BLOCK DIAGRAM



Notes: 1. 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time-base.
2. The alignment of the 10 bits from the CCPR register is determined by the CCPxFMT bit.

23.4.2 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for standard PWM operation:

1. Use the desired output pin RxyPPS control to select CCPx as the source and disable the CCPx pin output driver by setting the associated TRIS bit.
2. Load the T2PR register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register, and the CCPRxH register with the PWM duty cycle value and configure the FMT bit of the CCPxCON register to set the proper register alignment.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the respective PIR register. See Note below.
   - Select the timer clock source to be as FOSC/4 using the T2CLK register. This is required for correct operation of the PWM module.
   - Configure the CKPS bits of the T2CON register with the Timer prescale value.
   - Enable the Timer by setting the ON bit of the T2CON register.
6. Enable PWM output pin:
   - Wait until the Timer overflows and the TMR2IF bit of the PIR4 register is set. See Note below.
   - Enable the CCPx pin output driver by clearing the associated TRIS bit.

Note:	In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

23.4.3 TIMER2 TIMER RESOURCE

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

23.4.4 PWM PERIOD

The PWM period is specified by the T2PR register of Timer2. The PWM period can be calculated using the formula of Equation 23-1.

EQUATION 23-1: PWM PERIOD

$$PWM\ Period = [(T2PR) + 1] \bullet 4 \bullet TOSC \bullet (TMR2\ Prescale\ Value)$$

Note 1: TOSC = 1/FOSC

When T2TMR is equal to T2PR, the following three events occur on the next increment cycle:

- T2TMR is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is transferred from the CCPRxL/H register pair into a 10-bit buffer.

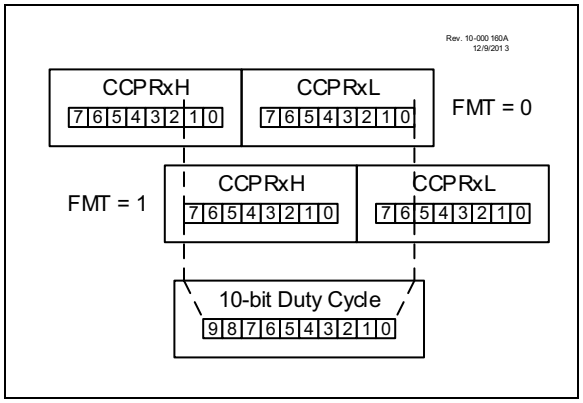
Note:	The Timer postscaler (see **Section 22.3 "External Reset Sources"**) is not used in the determination of the PWM frequency.

23.4.5 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the CCPRxH:CCPRxL register pair. The alignment of the 10-bit value is determined by the FMT bit of the CCPxCON register (see Figure 23-5). The CCPRxH:CCPRxL register pair can be written to at any time; however the duty cycle value is not latched into the 10-bit buffer until after a match between T2PR and T2TMR.

Equation 23-2 is used to calculate the PWM pulse width. Equation 23-3 is used to calculate the PWM duty cycle ratio.

FIGURE 23-5: PWM 10-BIT ALIGNMENT



EQUATION 23-2: PULSE WIDTH

$$Pulse\ Width = (CCPRxH{:}CCPRxL\ register\ pair) \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

EQUATION 23-3: DUTY CYCLE RATIO

$$Duty\ Cycle\ Ratio = \frac{(CCPRxH{:}CCPRxL\ register\ pair)}{4(T2PR + 1)}$$

CCPRxH:CCPRxL register pair are used to double buffer the PWM duty cycle. This double buffering provides glitchless PWM operation.

The 8-bit timer T2TMR register is concatenated with either the 2-bit internal system clock (FOSC), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRxH:CCPRxL register pair, then the CCPx pin is cleared (see Figure 23-4).

23.4.6 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when T2PR is 255. The resolution is a function of the T2PR register value as shown by Equation 23-4.

EQUATION 23-4: PWM RESOLUTION

$$Resolution = \frac{\log[4(T2PR + 1)]}{\log(2)}\ bits$$

Note: If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

TABLE 23-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 20 MHz)

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 23-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 8 MHz)

PWM Frequency	1.22 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

23.4.7 OPERATION IN SLEEP MODE

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from its previous state.

23.4.8 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See **Section 7.0 “Oscillator Module (with Fail-Safe Clock Monitor)”** for additional details.

23.4.9 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

23.5 Register Definitions: CCP Control

Long bit name prefixes for the CCP peripherals are shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
CCP1	CCP1
CCP2	CCP2
CCP3	CCP3
CCP4	CCP4

REGISTER 23-1: CCPxCON: CCPx CONTROL REGISTER

R/W-0/0	U-0	R-x	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	OUT	FMT	MODE[3:0]			
bit 7							bit 0

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

bit 7 **EN:** CCP Module Enable bit
1 = CCP is enabled
0 = CCP is disabled

bit 6 **Unimplemented:** Read as '0'

bit 5 **OUT:** CCPx Output Data bit (read-only)

bit 4 **FMT:** CCPW (pulse-width) Alignment bit
MODE = Capture mode:
Unused
MODE = Compare mode:
Unused
MODE = PWM mode:
1 = Left-aligned format
0 = Right-aligned format

bit 3-0 **MODE[3:0]:** CCPx Mode Select bits

MODE	Operating Mode	Operation	Set CCPxIF
11xx	PWM	PWM operation	Yes
1011	Compare	Pulse output; clear TMR1(2)	Yes
1010		Pulse output	Yes
1001		Clear output(1)	Yes
1000		Set output(1)	Yes
0111	Capture	Every 16th rising edge of CCPx input	Yes
0110		Every 4th rising edge of CCPx input	Yes
0101		Every rising edge of CCPx input	Yes
0100		Every falling edge of CCPx input	Yes
0011		Every edge of CCPx input	Yes
0010	Compare	Toggle output	Yes
0001		Toggle output; clear TMR1(2)	Yes
0000	Disabled		—

Note 1: The set and clear operations of the Compare mode are reset by setting MODE = 4'b0000 or EN = 0.
2: When MODE = 0001 or 1011, then the timer associated with the CCP module is cleared. TMR1 is the default selection for the CCP module, so it is used for indication purpose only.

REGISTER 23-2: CCPTMRS0: CCP TIMERS CONTROL REGISTER 0

R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1
C4TSEL[1:0]		C3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	
bit 7							bit 0

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

bit 7-6 **C4TSEL[1:0]:** CCP4 Timer Selection bits
`11` = CCP4 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
`10` = CCP4 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
`01` = CCP4 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
`00` = Reserved

bit 5-4 **C3TSEL[1:0]:** CCP3 Timer Selection bits
`11` = CCP3 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
`10` = CCP3 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
`01` = CCP3 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
`00` = Reserved

bit 3-2 **C2TSEL[1:0]:** CCP2 Timer Selection bits
`11` = CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
`10` = CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
`01` = CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
`00` = Reserved

bit 1-0 **C1TSEL[1:0]:** CCP1 Timer Selection bits
`11` = CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
`10` = CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
`01` = CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
`00` = Reserved

REGISTER 23-3: CCPxCAP: CAPTURE INPUT SELECTION MULTIPLEXER REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/x	R/W-0/x	R/W-0/x
—	—	—	—	—	CTS[2:0]		
bit 7							bit 0

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

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **CTS[2:0]:** Capture Trigger Input Selection bits

CTS[1:0]	Connection			
	CCP1	CCP2	CCP3	CCP4
111	CLC4_out			
110	CLC3_out			
101	CLC2_out			
100	CLC1_out			
011	IOC_Interrupt			
010	CMP2_output			
001	CMP1_output			
000	Pin selected by CCP1PPS	Pin selected by CCP2PPS	Pin selected by CCP3PPS	Pin selected by CCP4PPS

REGISTER 23-4: CCPRxL: CCPx REGISTER LOW BYTE

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
RL[7:0]							
bit 7							bit 0

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

bit 7-0 MODE = Capture Mode:
RL[7:0]: LSB of captured TMR1 value
MODE = Compare Mode:
RL[7:0]: LSB compared to TMR1 value
MODE = PWM Mode && FMT = 0:
RL[7:0]: CCPW[7:0] – Pulse-Width LS 8 bits
MODE = PWM Mode && FMT = 1:
RL[7:6]: CCPW[1:0] – Pulse-Width LS 2 bits
RL[5:0]: Not used

REGISTER 23-5: CCPRxH: CCPx REGISTER HIGH BYTE

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
RH[7:0]							
bit 7							bit 0

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

bit 7-0 MODE = Capture Mode:
RH[7:0]: MSB of captured TMR1 value
MODE = Compare Mode:
RH[7:0]: MSB compared to TMR1 value
MODE = PWM Mode && FMT = 0:
RH[7:2]: Not used
RH[1:0]: CCPW[9:8] – Pulse-Width MS 2 bits
MODE = PWM Mode && FMT = 1:
RH[7:0]: CCPW[9:2] – Pulse-Width MS 8 bits

TABLE 23-4: SUMMARY OF REGISTERS ASSOCIATED WITH CCPx

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CCPxCON	EN	—	OUT	FMT	MODE[3:0]				351
CCPxCAP	—	—	—	—	—	—	CTS[1:0]		353
CCPRxL	CCPRx[7:0]								353
CCPRxH	CCPRx[15:8]								354
CCPTMRS0	C4TSEL[1:0]		C3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]		352

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used by the CCP module.

24.0 PULSE-WIDTH MODULATION (PWM)

The PWM module generates a pulse-width modulated signal determined by the duty cycle, period, and resolution that are configured by the following registers:

- TxPR
- TxCON
- PWMxDCH
- PWMxDCL
- PWMxCON

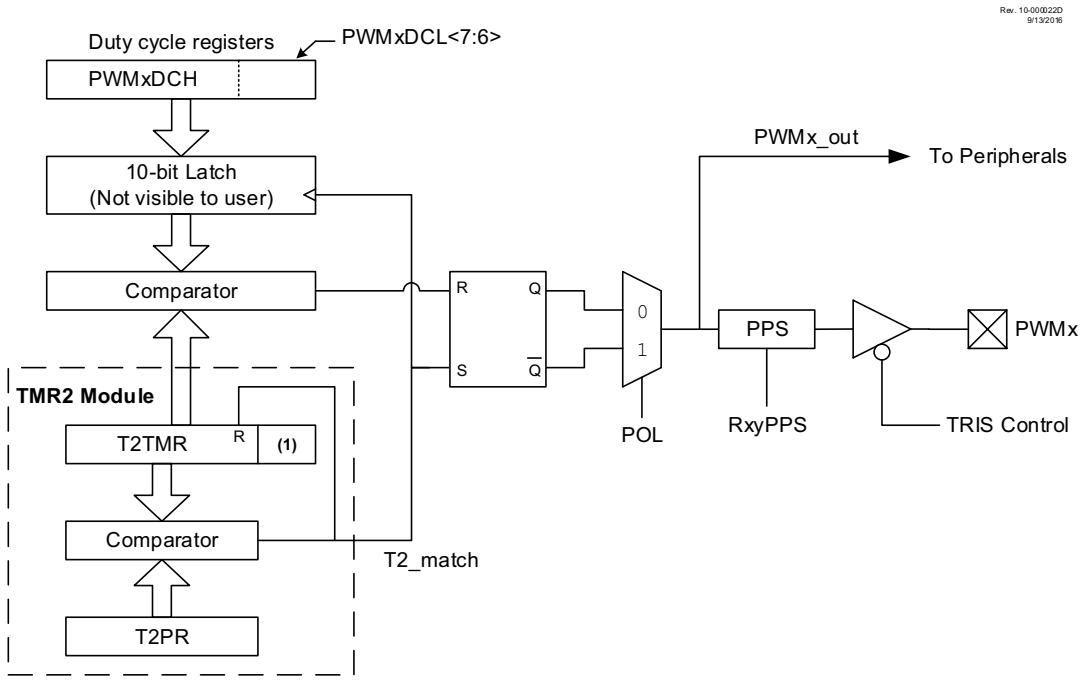
Note: The corresponding TRIS bit must be cleared to enable the PWM output on the PWMx pin.

Each PWM module can select the timer source that controls the module. Each module has an independent timer selection which can be accessed using the CCPTMRS1 register (Register 23-2). Please note that the PWM mode operation is described with respect to T2TMR in the following sections.

Figure 24-1 shows a simplified block diagram of PWM operation.

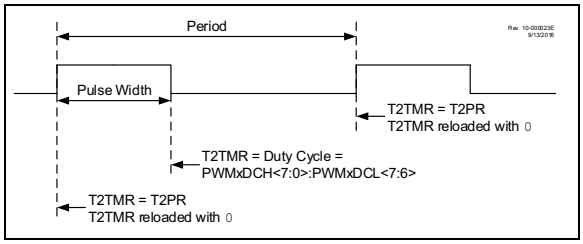
Figure 24-2 shows a typical waveform of the PWM signal.

FIGURE 24-1: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time-base.

FIGURE 24-2: PWM OUTPUT



For a step-by-step procedure on how to set up this module for PWM operation, refer to **Section 24.1.9 "Setup for PWM Operation using PWMx Pins"**.

24.1 PWMx Pin Configuration

All PWM outputs are multiplexed with the PORT data latch. The user must configure the pins as outputs by clearing the associated TRIS bits.

24.1.1 FUNDAMENTAL OPERATION

The PWM module produces a 10-bit resolution output. The PWM timer can be selected using the PxTSEL bits in the CCPTMRS1 register. The default selection for PWMx is T2TMR. Please note that the PWM module operation in the following sections is described with respect to T2TMR. Timer2 and T2PR set the period of the PWM. The PWMxDCL and PWMxDCH registers configure the duty cycle. The period is common to all PWM modules, whereas the duty cycle is independently controlled.

> **Note:** The Timer2 postscaler is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

All PWM outputs associated with Timer2 are set when T2TMR is cleared. Each PWMx is cleared when T2TMR is equal to the value specified in the corresponding PWMxDCH (8 MSb) and PWMxDCL[7:6] (2 LSb) registers. When the value is greater than or equal to T2PR, the PWM output is never cleared (100% duty cycle).

> **Note:** The PWMxDCH and PWMxDCL registers are double buffered. The buffers are updated when Timer2 matches T2PR. Care should be taken to update both registers before the timer match occurs.

24.1.2 PWM OUTPUT POLARITY

The output polarity is inverted by setting the PWMxPOL bit of the PWMxCON register.

24.1.3 PWM PERIOD

The PWM period is specified by the T2PR register of Timer2. The PWM period can be calculated using the formula of Equation 24-1. It is required to have FOSC/4 as clock input to Timer2/4/6 for correct PWM operation.

EQUATION 24-1: PWM PERIOD

$$PWM\ Period = [(T2PR) + 1] \bullet 4 \bullet TOSC \bullet (TMR2\ Prescale\ Value)$$

Note: TOSC = 1/FOSC

When T2TMR is equal to T2PR, the following three events occur on the next increment cycle:

- T2TMR is cleared
- The PWM output is active. (Exception: When the PWM duty cycle = 0%, the PWM output will remain inactive.)
- The PWMxDCH and PWMxDCL register values are latched into the buffers.

> **Note:** The Timer2 postscaler has no effect on the PWM operation.

24.1.4 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the PWMxDCH and PWMxDCL register pair. The PWMxDCH register contains the eight MSbs and the PWMxDCL[7:6], the two LSbs. The PWMxDCH and PWMxDCL registers can be written to at any time.

Equation 24-2 is used to calculate the PWM pulse width.

Equation 24-3 is used to calculate the PWM duty cycle ratio.

EQUATION 24-2: PULSE WIDTH

$$Pulse\ Width = (PWMxDCH{:}PWMxDCL<7{:}6>) \bullet TOSC \bullet (TMR2\ Prescale\ Value)$$

Note: TOSC = 1/FOSC

EQUATION 24-3: DUTY CYCLE RATIO

$$Duty\ Cycle\ Ratio = \frac{(PWMxDCH{:}PWMxDCL<7{:}6>)}{4(T2PR + 1)}$$

The 8-bit timer T2TMR register is concatenated with the two Least Significant bits of 1/FOSC, adjusted by the Timer2 prescaler to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

24.1.5 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when T2PR is 255. The resolution is a function of the T2PR register value as shown by Equation 24-4.

EQUATION 24-4: PWM RESOLUTION

$$Resolution = \frac{log[4(T2PR + 1)]}{log(2)} \ bits$$

Note: If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

TABLE 24-1: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 20 MHz)

PWM Frequency	0.31 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescale	64	4	1	1	1	1
T2PR Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 24-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 8 MHz)

PWM Frequency	0.31 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
Timer Prescale	64	4	1	1	1	1
T2PR Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

24.1.6 OPERATION IN SLEEP MODE

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the PWMx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from its previous state.

24.1.7 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency (FOSC). Any changes in the system clock frequency will result in changes to the PWM frequency. Refer to **Section 7.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for additional details.

24.1.8 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the PWM registers to their Reset states.

24.1.9 SETUP FOR PWM OPERATION USING PWMx PINS

The following steps should be taken when configuring the module for PWM operation using the PWMx pins:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the T2PR register with the PWM period value.
4. Load the PWMxDCH register and bits [7:6] of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the respective PIR register. See Note 1 below.
   - Select the timer clock source to be as FOSC/4 using the TxCLK register. This is required for correct operation of the PWM module.
   - Configure the CKPS bits of the T2CON register with the Timer2 prescale value.
   - Enable Timer2 by setting the ON bit of the T2CON register.
6. Enable PWM output pin and wait until Timer2 overflows, TMR2IF bit of the respective PIR register is set. See note below.
7. Enable the PWMx pin output driver(s) by clearing the associated TRIS bit(s) and setting the desired pin PPS control bits.
8. Configure the PWM module by loading the PWMxCON register with the appropriate values.

Note 1: In order to send a complete duty cycle and period on the first PWM output, the above steps must be followed in the order given. If it is not critical to start with a complete PWM signal, then move Step 8 to replace Step 4.

2: For operation with other peripherals only, disable PWMx pin outputs.

24.1.10 SETUP FOR PWM OPERATION TO OTHER DEVICE PERIPHERALS

The following steps should be taken when configuring the module for PWM operation to be used by other device peripherals:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the T2PR register with the PWM period value.
4. Load the PWMxDCH register and bits [7:6] of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the respective PIR register. See Note 1 below.
   - Select the timer clock source to be as FOSC/4 using the TxCLK register. This is required for correct operation of the PWM module.
   - Configure the CKPS bits of the T2CON register with the Timer2 prescale value.
   - Enable Timer2 by setting the ON bit of the T2CON register.
6. Enable PWM output pin:
   - Wait until Timer2 overflows, TMR2IF bit of the respective PIR register is set. See Note 1 below.
7. Configure the PWM module by loading the PWMxCON register with the appropriate values.

Note 1: In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

24.2 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
PWM5	PWM5
PWM6	PWM6
PWM7	PWM7
PWM8	PWM8

REGISTER 24-1: PWMxCON: PWM CONTROL REGISTER

R/W-0/0	U-0	R-0/0	R/W-0/0	U-0	U-0	U-0	U-0
EN	—	OUT	POL	—	—	—	—
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 **EN:** PWM Module Enable bit
1 = PWM module is enabled
0 = PWM module is disabled

bit 6 **Unimplemented:** Read as '0'

bit 5 **OUT:** PWM Module Output Level When Bit is Read

bit 4 **POL:** PWM Output Polarity Select bit
1 = PWM output is inverted
0 = PWM output is normal

bit 3-0 **Unimplemented:** Read as '0'

REGISTER 24-2: CCPTMRS1: CCP TIMERS CONTROL REGISTER 1

R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1
P8TSEL[1:0]		P7TSEL[1:0]		P6TSEL[1:0]		P5TSEL[1:0]	
bit 7							bit 0

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

bit 7-6 **P8TSEL[1:0]:** PWM8 Timer Selection bits
11 = PWM8 based on TMR6
10 = PWM8 based on TMR4
01 = PWM8 based on TMR2
00 = Reserved

bit 5-4 **P7TSEL[1:0]:** PWM7 Timer Selection bits
11 = PWM7 based on TMR6
10 = PWM7 based on TMR4
01 = PWM7 based on TMR2
00 = Reserved

bit 3-2 **P6TSEL[1:0]:** PWM6 Timer Selection bits
11 = PWM6 based on TMR6
10 = PWM6 based on TMR4
01 = PWM6 based on TMR2
00 = Reserved

bit 1-0 **P5TSEL[1:0]:** PWM5 Timer Selection bits
11 = PWM5 based on TMR6
10 = PWM5 based on TMR4
01 = PWM5 based on TMR2
00 = Reserved

REGISTER 24-3: PWMxDCH: PWM DUTY CYCLE HIGH BITS

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
DC[9:2]							
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 7-0 **DC[9:2]:** PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle. The two LSbs are found in PWMxDCL Register.

REGISTER 24-4: PWMxDCL: PWM DUTY CYCLE LOW BITS

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
DC[1:0]		—	—	—	—	—	—
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 **DC[1:0]:** PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle. The MSbs are found in PWMxDCH Register.

bit 5-0 **Unimplemented:** Read as ‘0’

TABLE 24-3: SUMMARY OF REGISTERS ASSOCIATED WITH PWM

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PWMxCON	EN	—	OUT	POL	—	—	—	—	359
PWMxDCH	DC[9:2]								361
PWMxDCL	DC[1:0]		—	—	—	—	—	—	361
CCPTMRS1	P8TSEL[1:0]		P7TSEL[1:0]		P6TSEL[1:0]		P5TSEL[1:0]		360

Legend: – = Unimplemented locations, read as ‘0’, u = unchanged, x = unknown. Shaded cells are not used by the PWM.

25.0 SIGNAL MEASUREMENT TIMER (SMT)

The SMT is a 24-bit counter with advanced clock and gating logic, which can be configured for measuring a variety of digital signal parameters such as pulse width, frequency and duty cycle, and the time difference between edges on two signals. The device has only one SMT module implemented.

Features of the SMT include:

- 24-bit timer/counter
  - Three 8-bit registers (SMT1L/H/U)
  - Readable and writable
  - Optional 16-bit operating mode
- Two 24-bit measurement capture registers
- One 24-bit period match register
- Multi-mode operation, including relative timing measurement
- Interrupt on period match
- Multiple clock, gate and signal sources
- Interrupt on acquisition complete
- Ability to read current input values

FIGURE 25-1: SMT BLOCK DIAGRAM

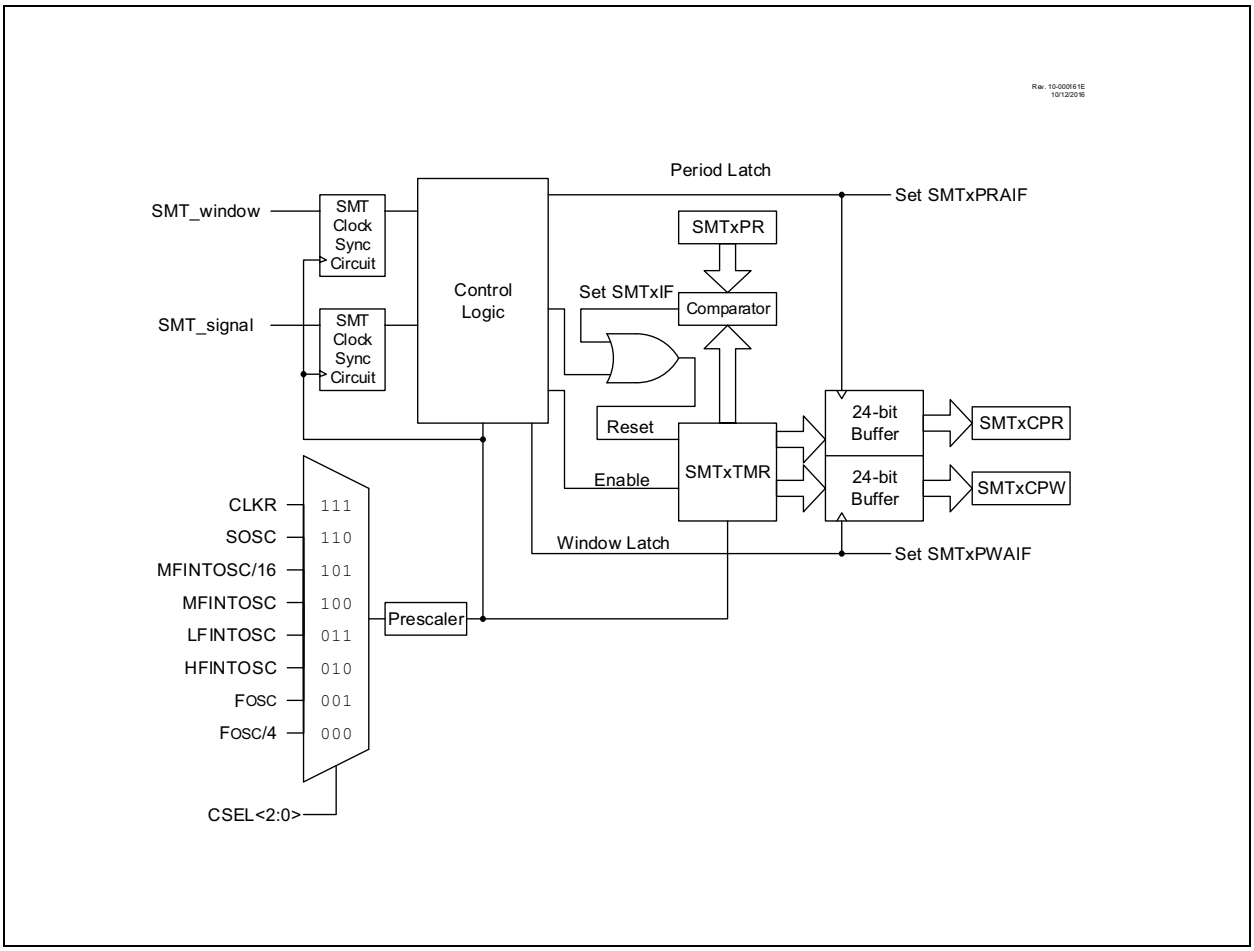
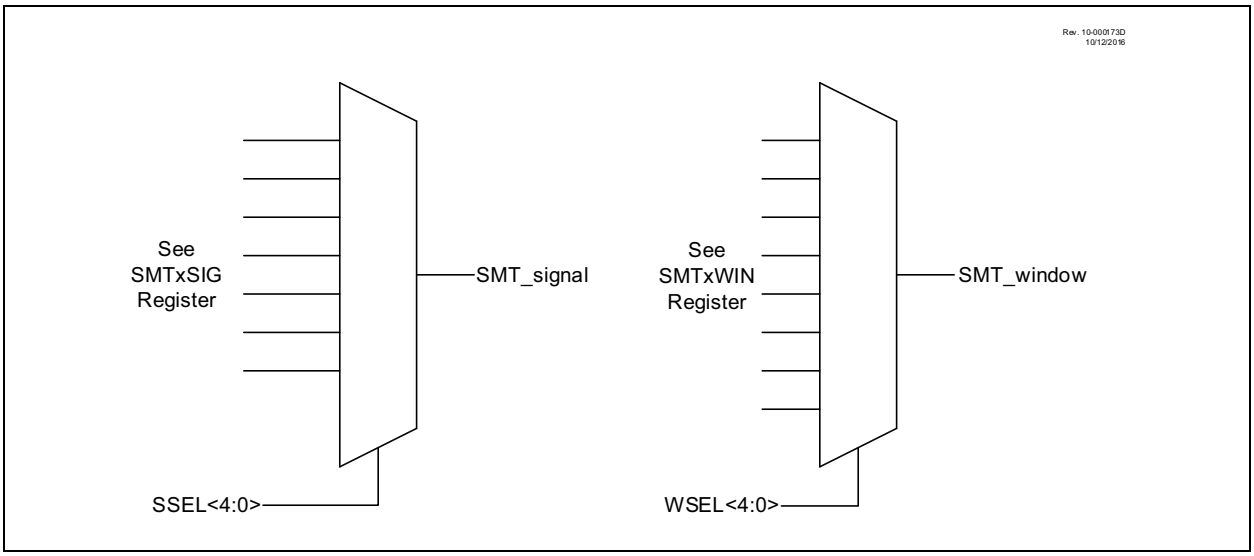


FIGURE 25-2: SMT SIGNAL AND WINDOW BLOCK DIAGRAM



25.1 SMT Operation

The core of the module is the 24-bit counter, SMT1TMR combined with a complex data acquisition front-end. Depending on the mode of operation selected, the SMT can perform a variety of measurements summarized in Table 25-1.

25.1.1 CLOCK SOURCES

Clock sources available to the SMT include:

- FOSC
- FOSC/4
- HFINTOSC 16 MHz
- LFINTOSC
- MFINTOSC 31.25 kHz

The SMT clock source is selected by configuring the CSEL[2:0] bits in the SMT1CLK register. The clock source can also be prescaled using the PS[1:0] bits of the SMT1CON0 register. The prescaled clock source is used to clock both the counter and any synchronization logic used by the module.

25.1.2 PERIOD MATCH INTERRUPT

Similar to other timers, the SMT triggers an interrupt when SMT1TMR rolls over to '0'. This happens when SMT1TMR = SMT1PR, regardless of mode. Hence, in any mode that relies on an external signal or a window to reset the timer, proper operation requires that SMT1PR be set to a period larger than that of the expected signal or window.

25.2 Basic Timer Function Registers

The SMT1TMR time base and the SMT1CPW/SMT1PR/SMT1CPR buffer registers serve several functions and can be manually updated using software.

25.2.1 TIME BASE

The SMT1TMR is the 24-bit counter that is the center of the SMT. It is used as the basic counter/timer for measurement in each of the modes of the SMT. It can be reset to a value of 24'h00_0000 by setting the RST bit of the SMT1STAT register. It can be written to and read from software, but it is not guarded for atomic access, therefore reads and writes to the SMT1TMR should only be made when the GO = 0, or the software should have other measures to ensure integrity of SMT1TMR reads/writes.

25.2.2 PULSE-WIDTH LATCH REGISTERS

The SMT1CPW registers are the 24-bit SMT pulse-width latch. They are used to latch in the value of the SMT1TMR when triggered by various signals, which are determined by the mode the SMT is currently in. The SMT1CPW registers can also be updated with the current value of the SMT1TMR value by setting the CPWUP bit of the SMT1STAT register.

25.2.3 PERIOD LATCH REGISTERS

The SMT1CPR registers are the 24-bit SMT period latch. They are used to latch in other values of the SMT1TMR when triggered by various other signals, which are determined by the mode the SMT is currently in.

The SMT1CPR registers can also be updated with the current value of the SMT1TMR value by setting the CPRUP bit in the SMT1STAT register.

25.3 Halt Operation

The counter can be prevented from rolling-over using the STP bit in the SMT1CON0 register. When halting is enabled, the period match interrupt persists until the SMT1TMR is reset (either by a manual Reset, **Section 25.2.1 "Time Base"**) or by clearing the GO bit of the SMT1CON1 register and writing the SMT1TMR values in software.

25.4 Polarity Control

The three input signals for the SMT have polarity control to determine whether or not they are active-high/positive edge or active-low/negative edge signals.

The following bits apply to Polarity Control:

- WSEL bit (Window Polarity)
- SSEL bit (Signal Polarity)
- CSEL bit (Clock Polarity)

These bits are located in the SMT1CON0 register.

25.5 Status Information

The SMT provides input status information for the user without requiring the need to deal with the polarity of the incoming signals.

25.5.1 WINDOW STATUS

Window status is determined by the WS bit of the SMT1STAT register. This bit is only used in Windowed Measure, Gated Counter and Gated Window Measure modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

25.5.2 SIGNAL STATUS

Signal status is determined by the AS bit of the SMT1STAT register. This bit is used in all modes except Window Measure, Time of Flight and Capture modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

25.5.3 GO STATUS

Timer run status is determined by the TS bit of the SMT1STAT register, and will be delayed in time by synchronizer delays in non-Counter modes.

© 2017-2020 Microchip Technology Inc.

25.6 Modes of Operation

The modes of operation are summarized in Table 25-1. The following sections provide detailed descriptions, examples of how the modes can be used. Note that all waveforms assume WPOL/SPOL/CPOL = 0. When WPOL/SPOL/CPOL = 1, all SMTSIGx, SMTWINx and SMT clock signals will have a polarity opposite to that indicated. For all modes, the REPEAT bit controls whether the acquisition is repeated or single. When REPEAT = 0 (Single Acquisition mode), the timer will stop incrementing and the GO bit will be reset upon the completion of an acquisition. Otherwise, the timer will continue and allow for continued acquisitions to overwrite the previous ones until the timer is stopped in software.

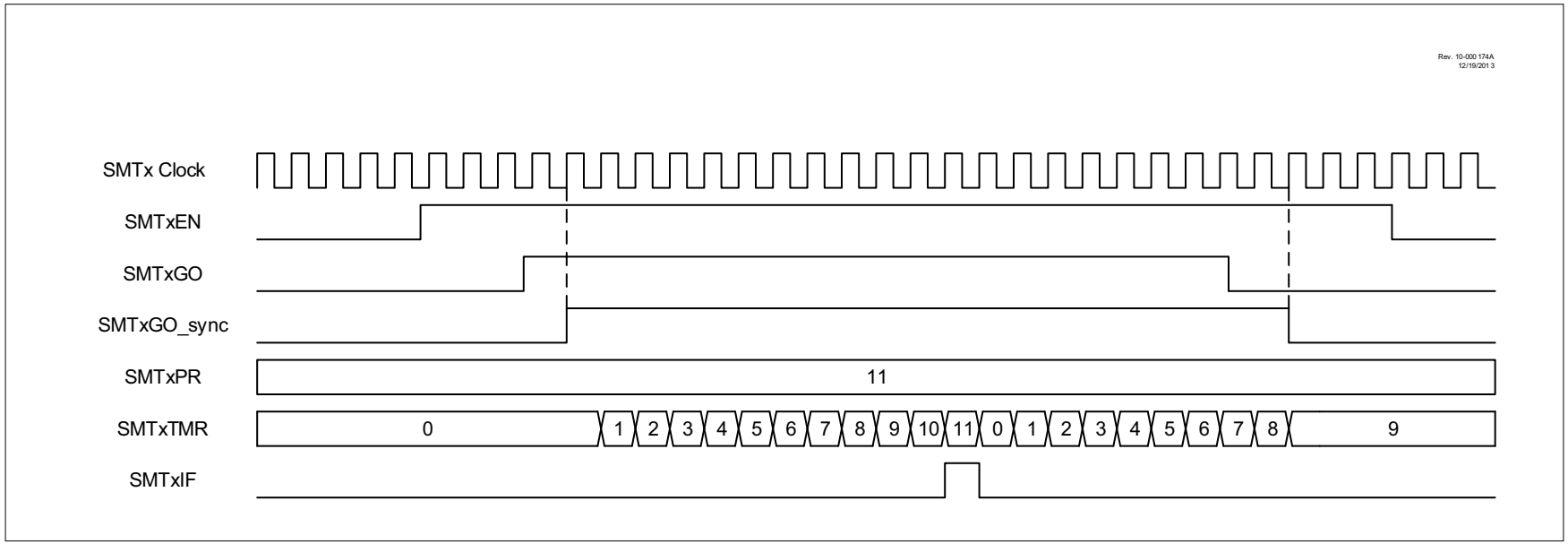
25.6.1 TIMER MODE

Timer mode is the simplest mode of operation where the SMT1TMR is used as a 16/24-bit timer. No data acquisition takes place in this mode. The timer increments as long as the GO bit has been set by software. No SMT window or SMT signal events affect the GO bit. Everything is synchronized to the SMT clock source. When the timer experiences a period match (SMT1TMR = SMT1PR), SMT1TMR is reset and the period match interrupt trips. See Figure 25-3.

TABLE 25-1: MODES OF OPERATION

MODE	Mode of Operation	Synchronous Operation	Reference
0000	Timer	Yes	**Section 25.6.1 “Timer Mode”**
0001	Gated Timer	Yes	**Section 25.6.2 “Gated Timer Mode”**
0010	Period and Duty Cycle Acquisition	Yes	**Section 25.6.3 “Period and Duty Cycle Mode”**
0011	High and Low Time Measurement	Yes	**Section 25.6.4 “High and Low Measure Mode”**
0100	Windowed Measurement	Yes	**Section 25.6.5 “Windowed Measure Mode”**
0101	Gated Windowed Measurement	Yes	**Section 25.6.6 “Gated Windowed Measure Mode”**
0110	Time of Flight	Yes	**Section 25.6.7 “Time of Flight Measure Mode”**
0111	Capture	Yes	**Section 25.6.8 “Capture Mode”**
1000	Counter	No	**Section 25.6.9 “Counter Mode”**
1001	Gated Counter	No	**Section 25.6.10 “Gated Counter Mode”**
1010	Windowed Counter	No	**Section 25.6.11 “Windowed Counter Mode”**
1011-1111	Reserved	—	—

FIGURE 25-3: TIMER MODE TIMING DIAGRAM

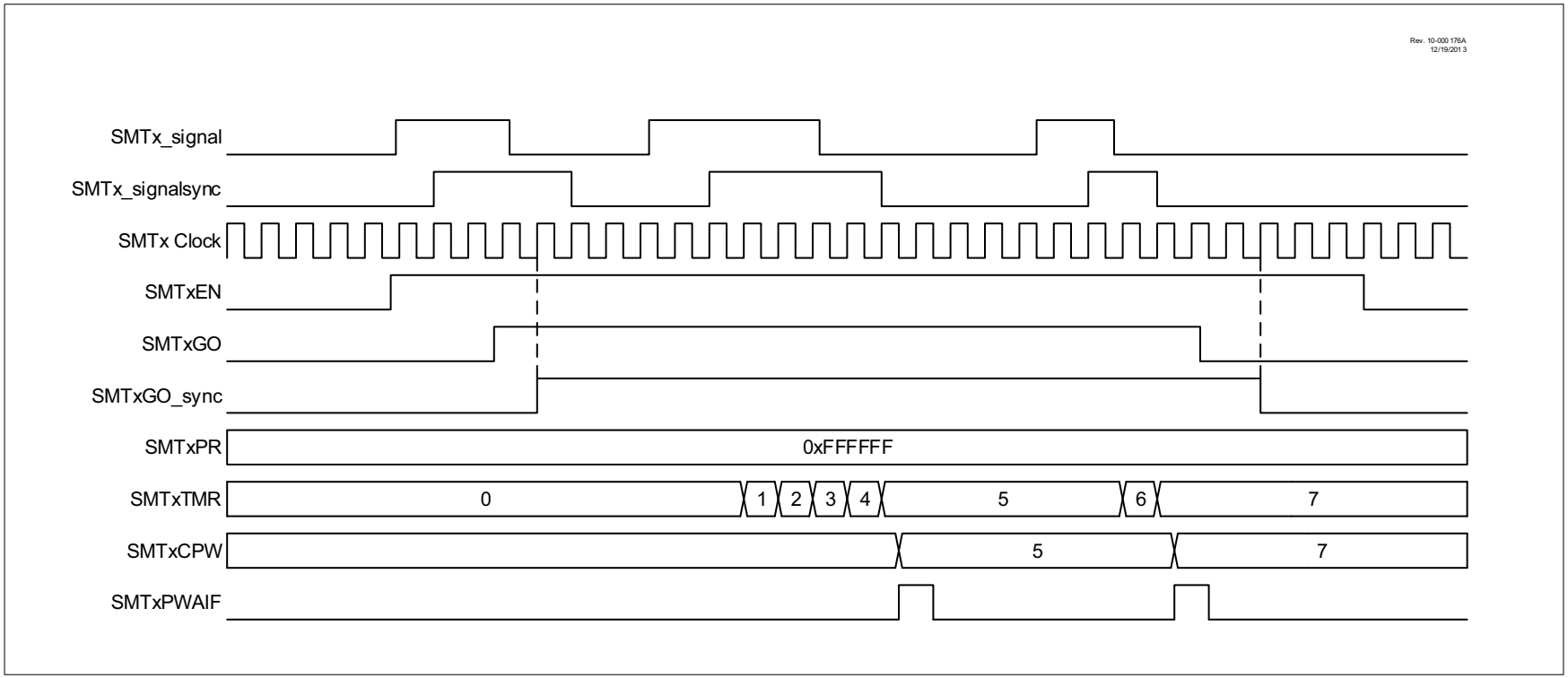


© 2017-2020 Microchip Technology Inc.

25.6.2 GATED TIMER MODE

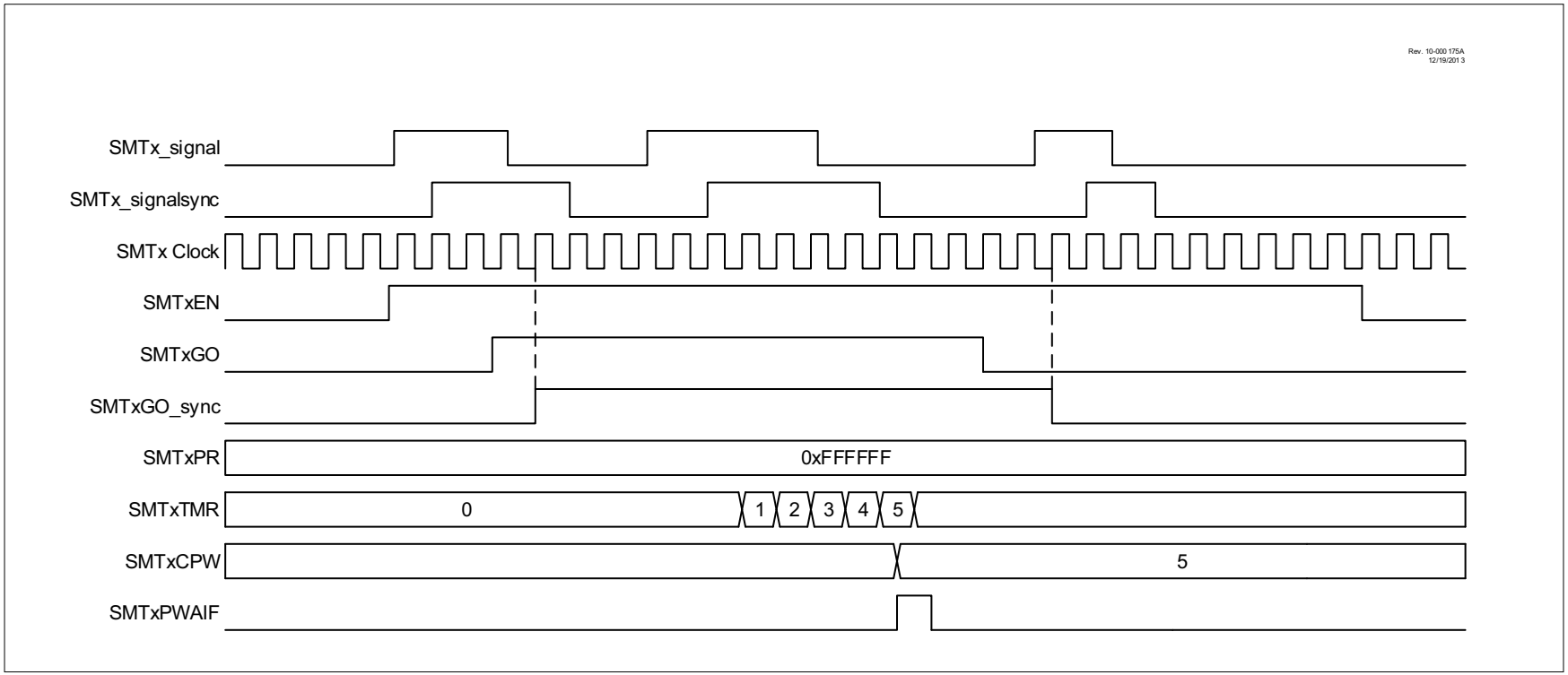
Gated Timer mode uses the SMTSIGx input to control whether or not the SMT1TMR will increment. Upon a falling edge of the external signal, the SMT1CPW register will update to the current value of the SMT1TMR. Example waveforms for both repeated and single acquisitions are provided in Figure 25-4 and Figure 25-5.

FIGURE 25-4: GATED TIMER MODE REPEAT ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-5: GATED TIMER MODE SINGLE ACQUISITION TIMING DIAGRAM

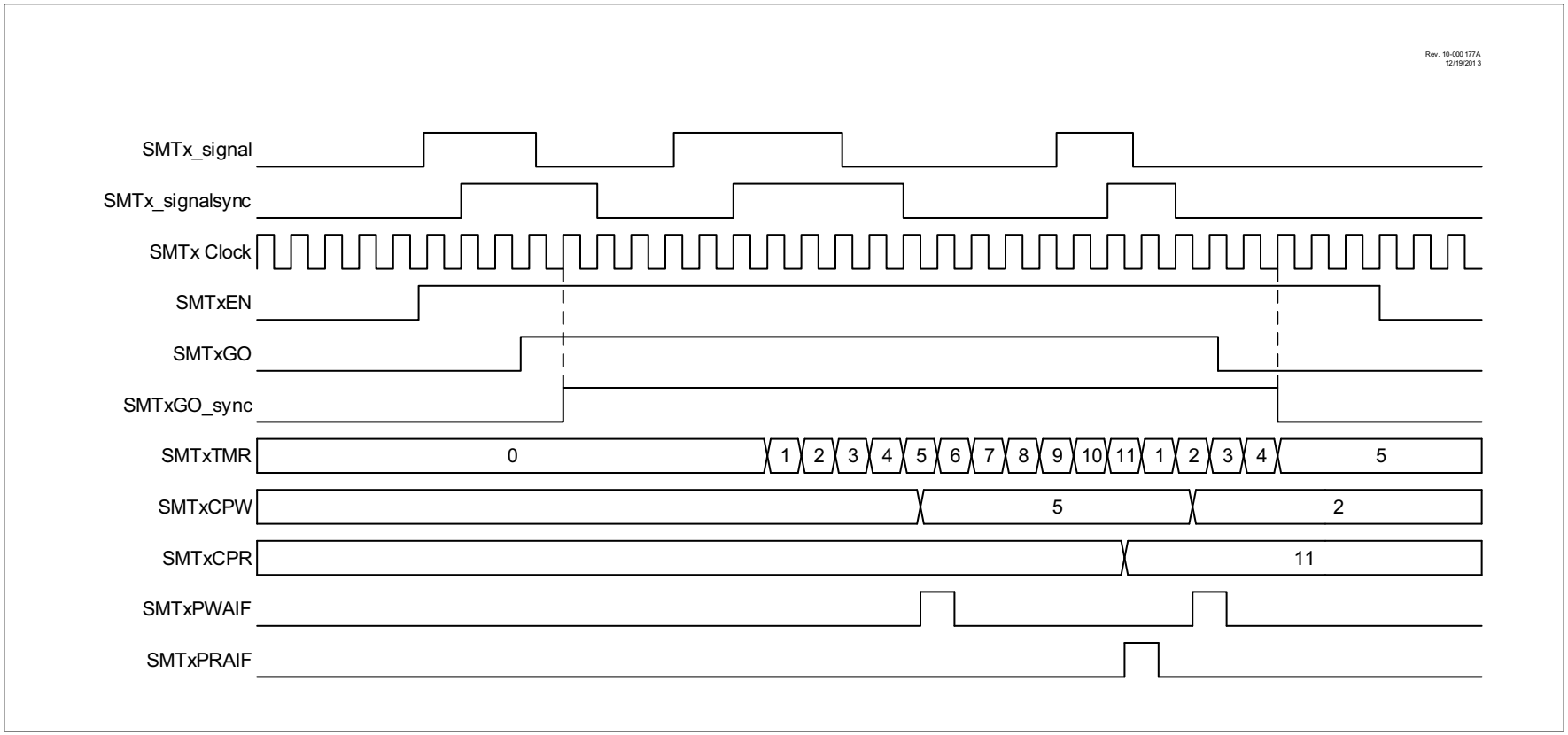


© 2017-2020 Microchip Technology Inc.

25.6.3 PERIOD AND DUTY CYCLE MODE

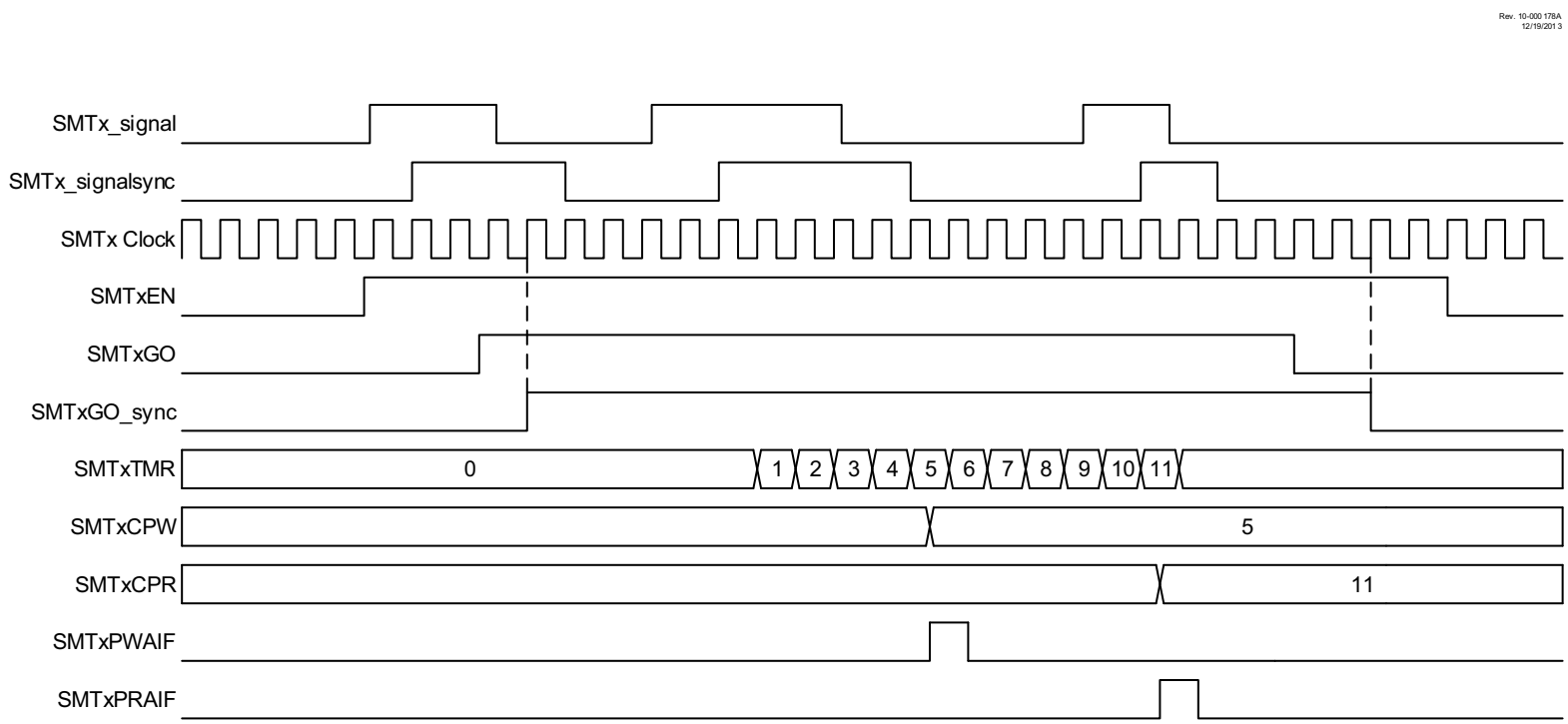
In Duty Cycle mode, either the duty cycle or period (depending on polarity) of the SMT1_signal can be acquired relative to the SMT clock. The CPW register is updated on a falling edge of the signal, and the CPR register is updated on a rising edge of the signal, along with the SMT1TMR resetting to 0x0001. In addition, the GO bit is reset on a rising edge when the SMT is in Single Acquisition mode. See Figure 25-6 and Figure 25-7.

FIGURE 25-6: PERIOD AND DUTY-CYCLE REPEAT ACQUISITION MODE TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-7: PERIOD AND DUTY-CYCLE SINGLE ACQUISITION TIMING DIAGRAM

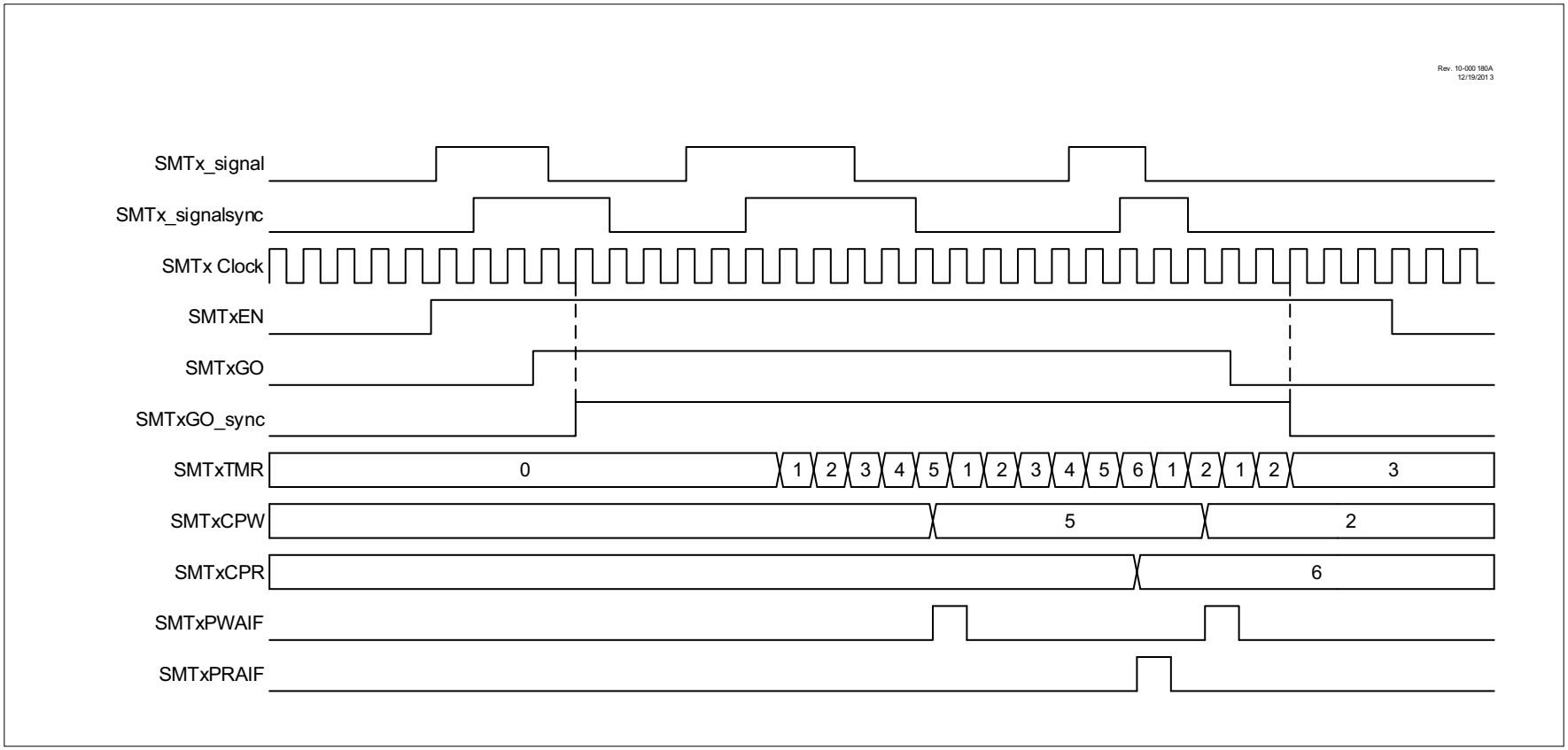


© 2017-2020 Microchip Technology Inc.

25.6.4 HIGH AND LOW MEASURE MODE

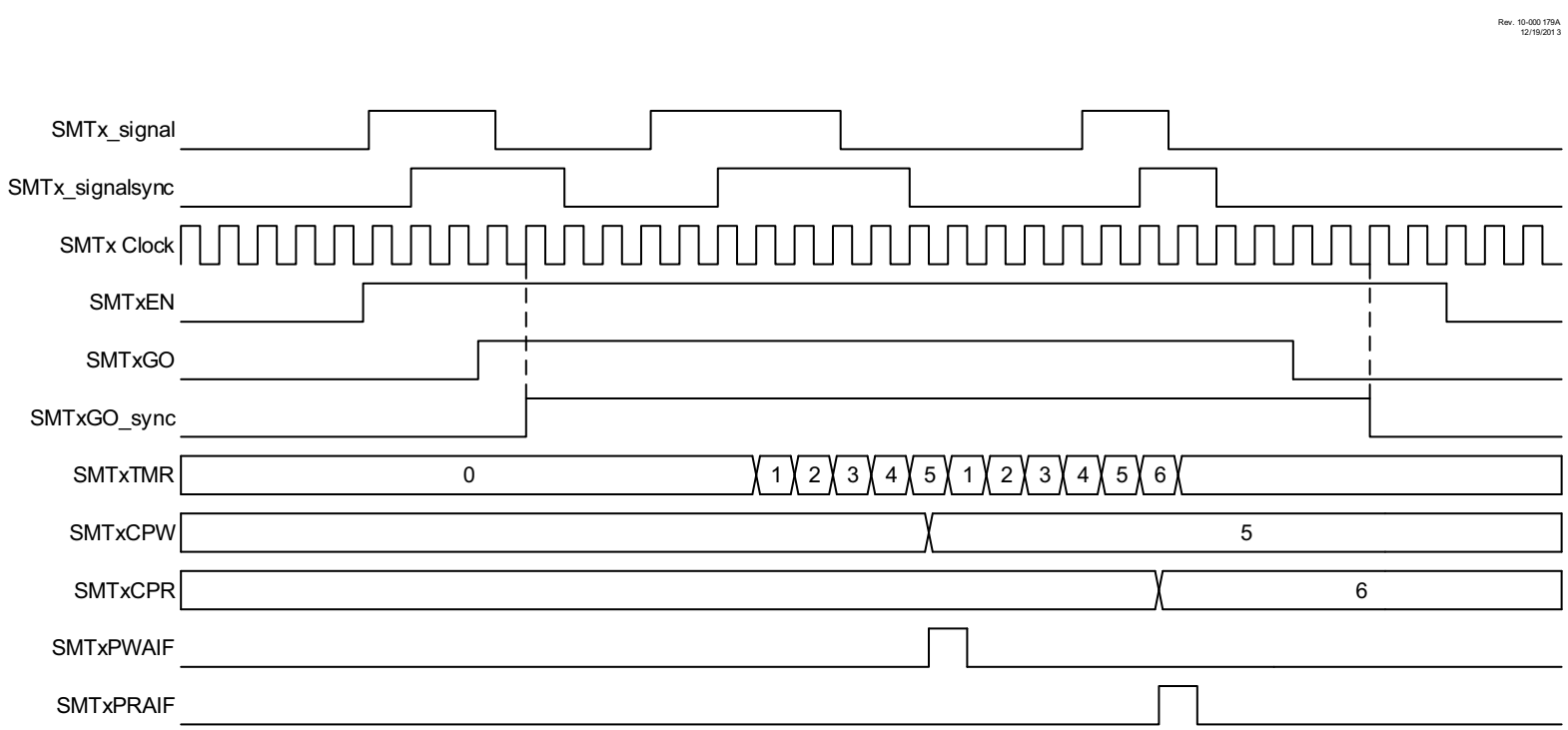
This mode measures the high and low pulse time of the SMTSIGx relative to the SMT clock. It begins incrementing the SMT1TMR on a rising edge on the SMTSIGx input, then updates the SMT1CPW register with the value and resets the SMT1TMR on a falling edge, starting to increment again. Upon observing another rising edge, it updates the SMT1CPR register with its current value and once again resets the SMT1TMR value and begins incrementing again. See Figure 25-8 and Figure 25-9.

FIGURE 25-8: HIGH AND LOW MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-9: HIGH AND LOW MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM

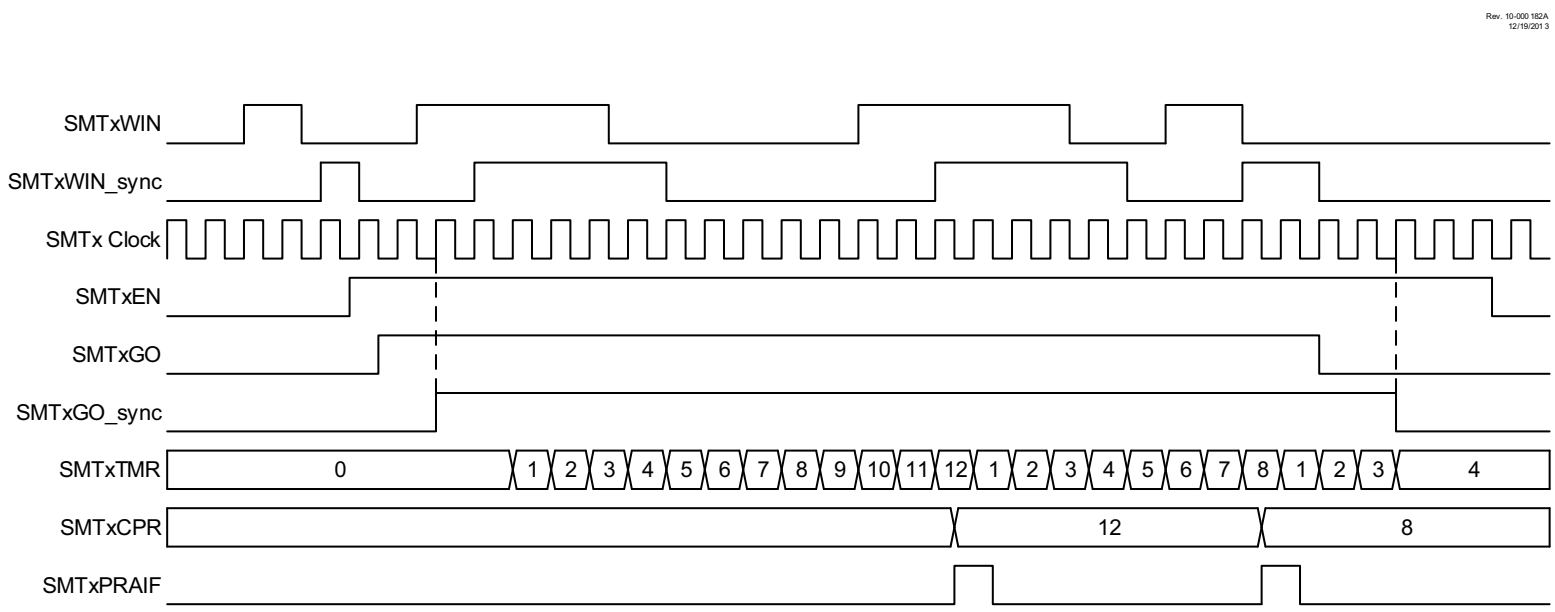


© 2017-2020 Microchip Technology Inc.

25.6.5 WINDOWED MEASURE MODE

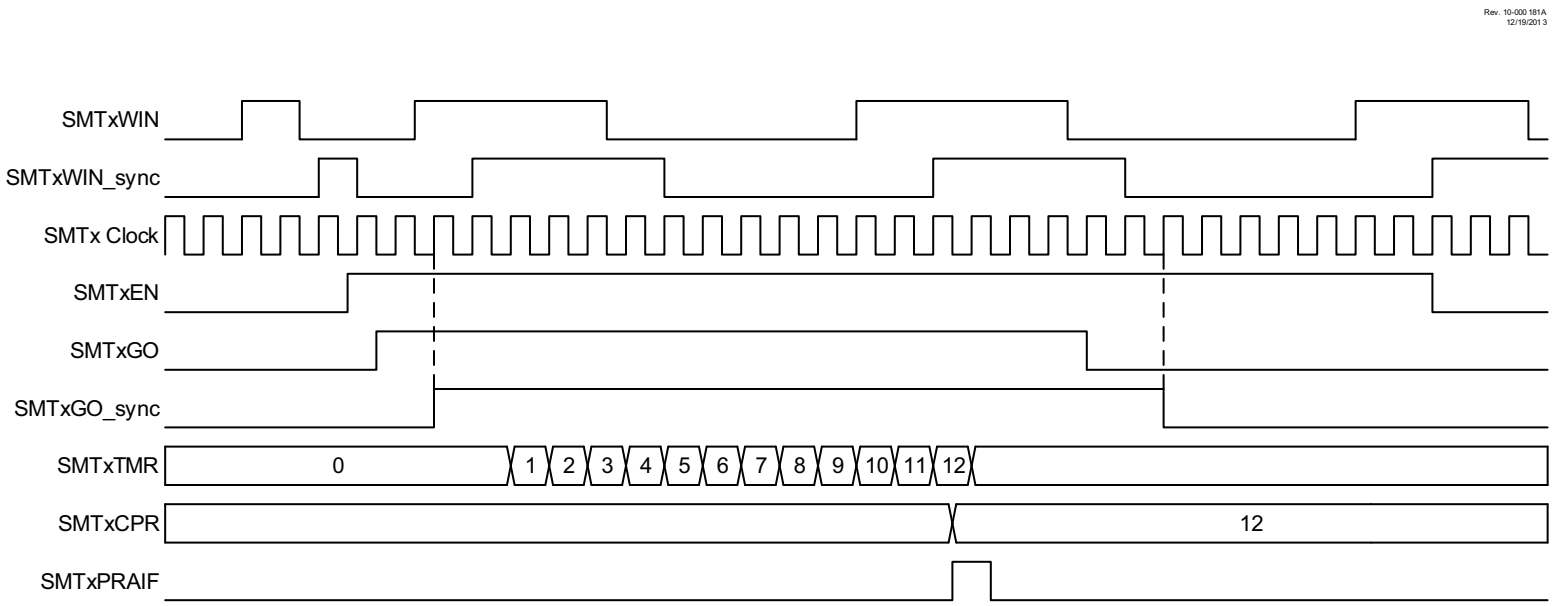
This mode measures the window duration of the SMTWINx input of the SMT. It begins incrementing the timer on a rising edge of the SMTWINx input and updates the SMT1CPR register with the value of the timer and resets the timer on a second rising edge. See Figure 25-10 and Figure 25-11.

FIGURE 25-10: WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-11: WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM

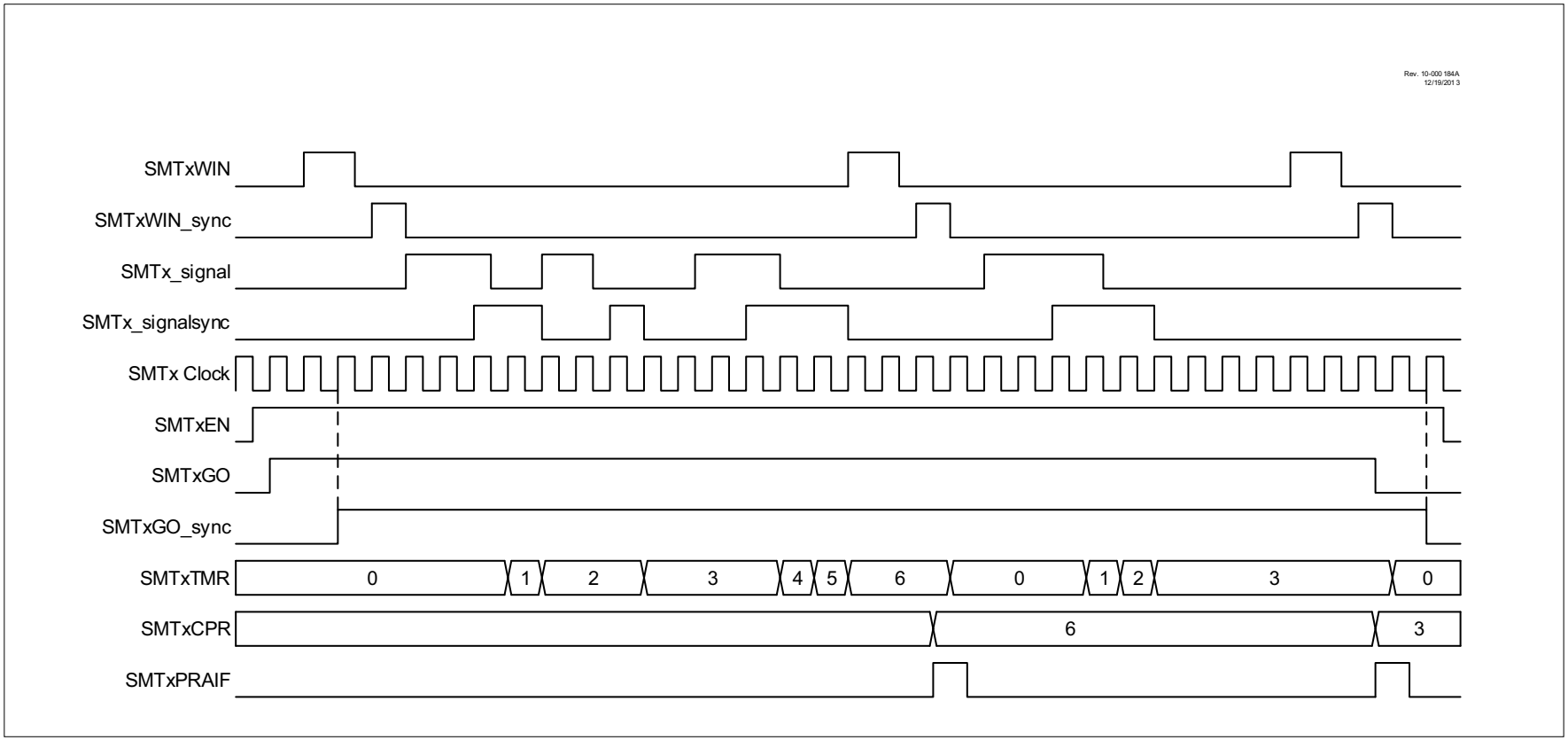


© 2017-2020 Microchip Technology Inc.

25.6.6 GATED WINDOWED MEASURE MODE

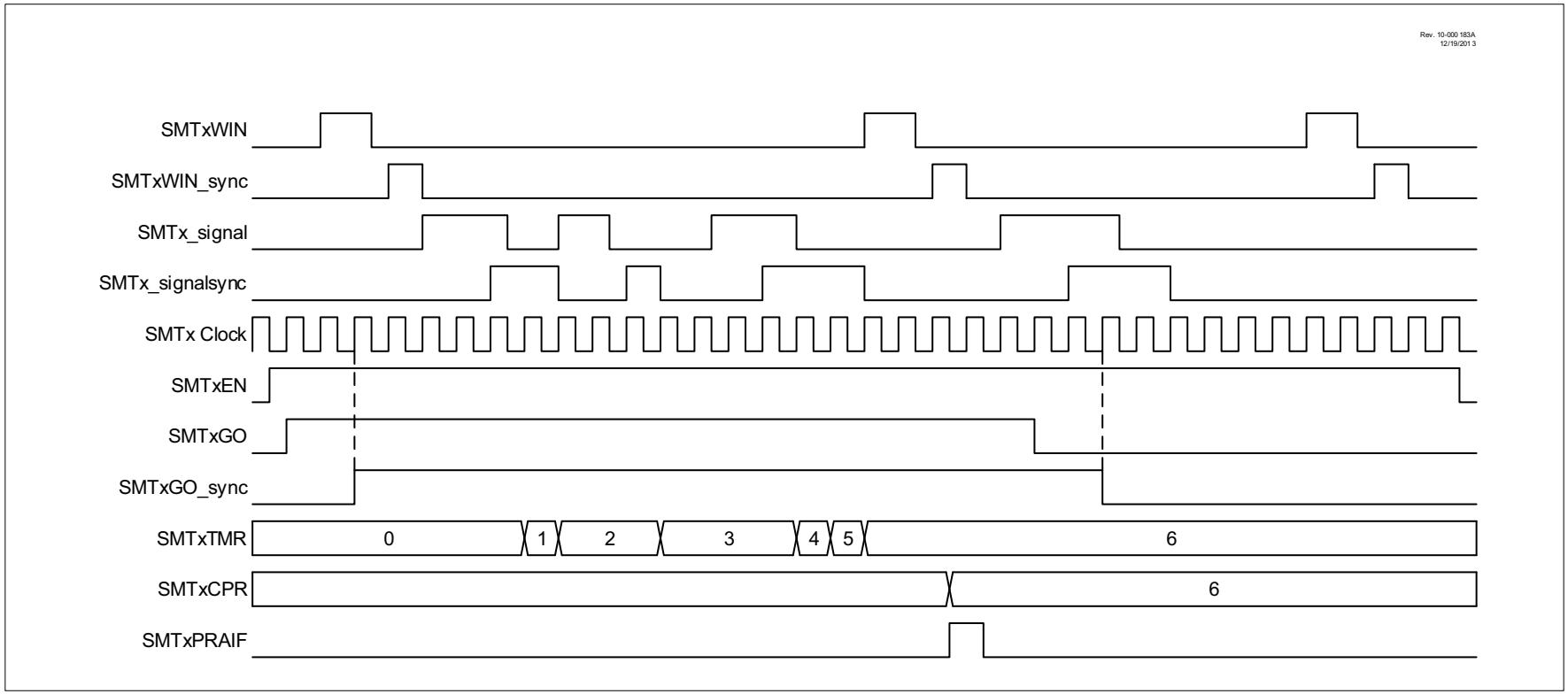
This mode measures the duty cycle of the SMT1_signal input over a known input window. It does so by incrementing the timer on each pulse of the clock signal while the SMT1_signal input is high, updating the SMT1CPR register and resetting the timer on every rising edge of the SMTWINx input after the first. See Figure 25-12 and Figure 25-13.

FIGURE 25-12: GATED WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-13: GATED WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAMS

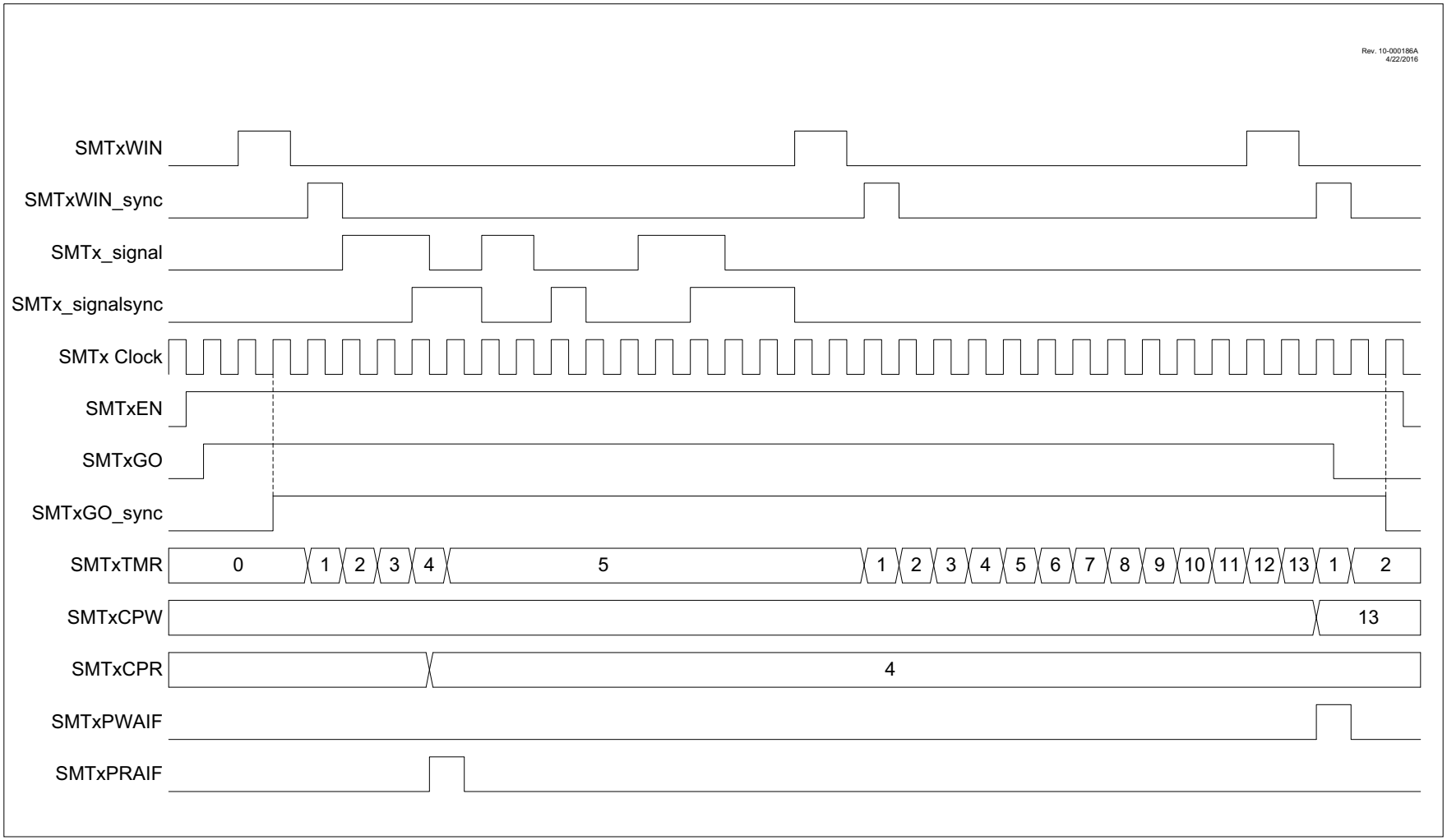


© 2017-2020 Microchip Technology Inc.

25.6.7 TIME OF FLIGHT MEASURE MODE

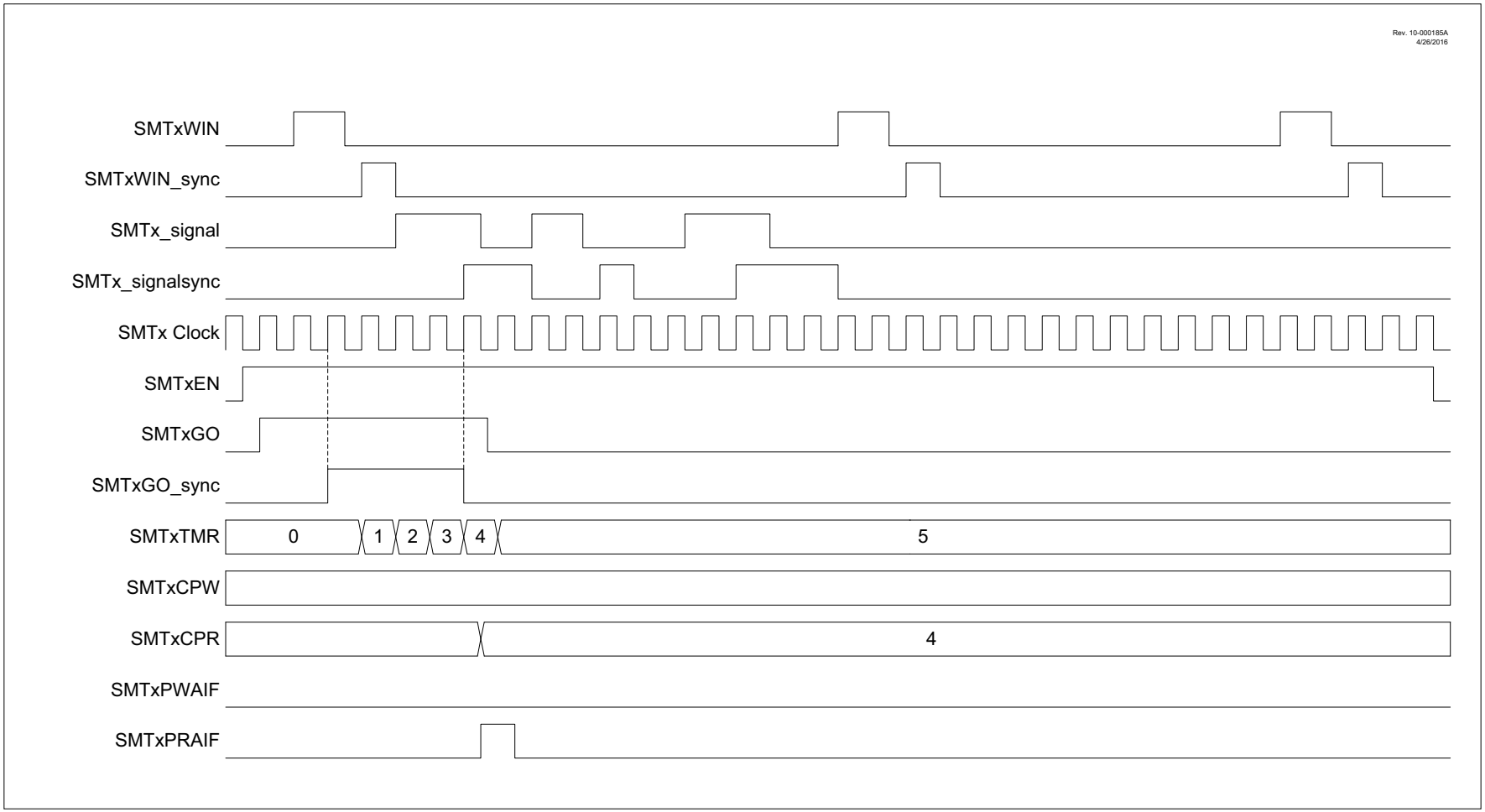
This mode measures the time interval between a rising edge on the SMTWINx input and a rising edge on the SMT1_signal input, beginning to increment the timer upon observing a rising edge on the SMTWINx input, while updating the SMT1CPR register and resetting the timer upon observing a rising edge on the SMT1_signal input. In the event of two SMTWINx rising edges without an SMT1_signal rising edge, it will update the SMT1CPW register with the current value of the timer and reset the timer value. See Figure 25-14 and Figure 25-15.

FIGURE 25-14: TIME OF FLIGHT MODE REPEAT ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-15: TIME OF FLIGHT MODE SINGLE ACQUISITION TIMING DIAGRAM

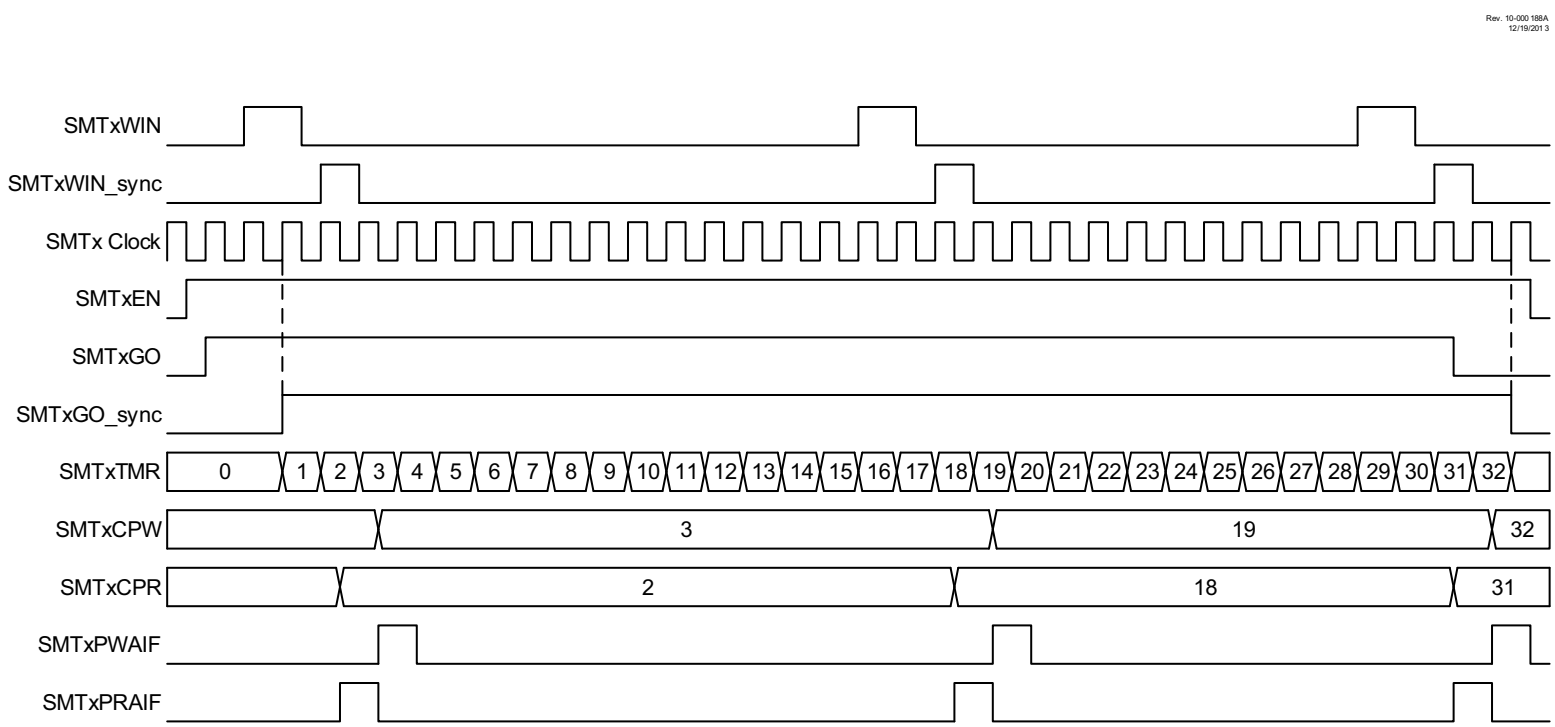


© 2017-2020 Microchip Technology Inc.

25.6.8 CAPTURE MODE

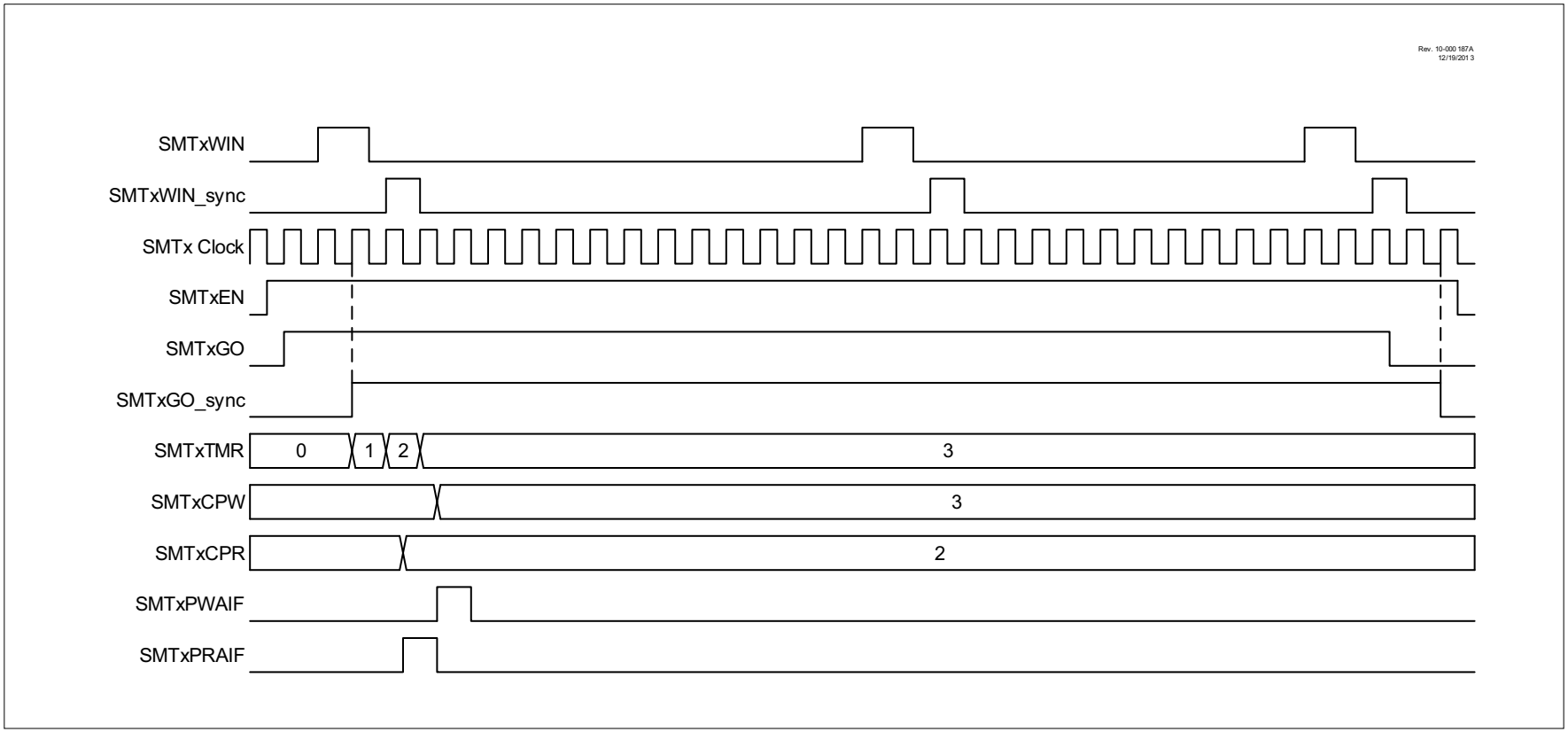
This mode captures the Timer value based on a rising or falling edge on the SMTWINx input and triggers an interrupt. This mimics the capture feature of a CCP module. The timer begins incrementing upon the GO bit being set, and updates the value of the SMT1CPR register on each rising edge of SMTWINx, and updates the value of the CPW register on each falling edge of the SMTWINx. See Figure 25-16 and Figure 25-17.

FIGURE 25-16: CAPTURE MODE REPEAT ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-17: CAPTURE MODE SINGLE ACQUISITION TIMING DIAGRAM

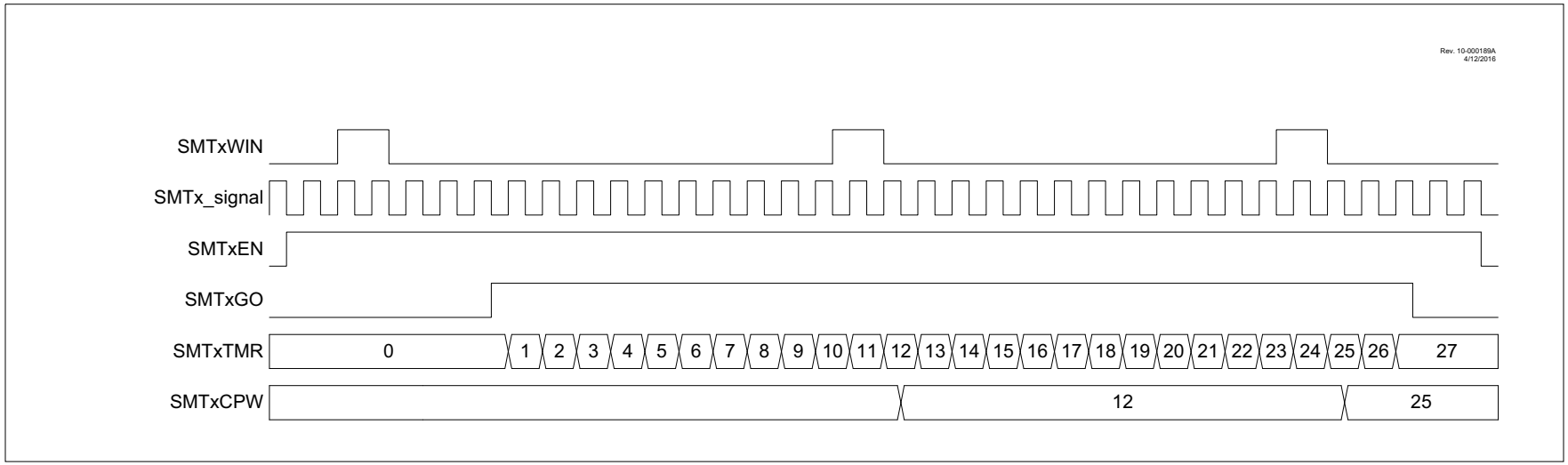


© 2017-2020 Microchip Technology Inc.

25.6.9 COUNTER MODE

This mode increments the timer on each pulse of the SMT1_signal input. This mode is asynchronous to the SMT clock and uses the SMT1_signal as a time source. The SMT1CPW register will be updated with the current SMT1TMR value on the rising edge of the SMT1WIN input. See Figure 25-18.

FIGURE 25-18: COUNTER MODE TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

25.6.10 GATED COUNTER MODE

This mode counts pulses on the SMT1_signal input, gated by the SMT1WIN input. It begins incrementing the timer upon seeing a rising edge of the SMT1WIN input and updates the SMT1CPW register upon a falling edge on the SMT1WIN input. See Figure 25-19 and Figure 25-20.

FIGURE 25-19: GATED COUNTER MODE REPEAT ACQUISITION TIMING DIAGRAM

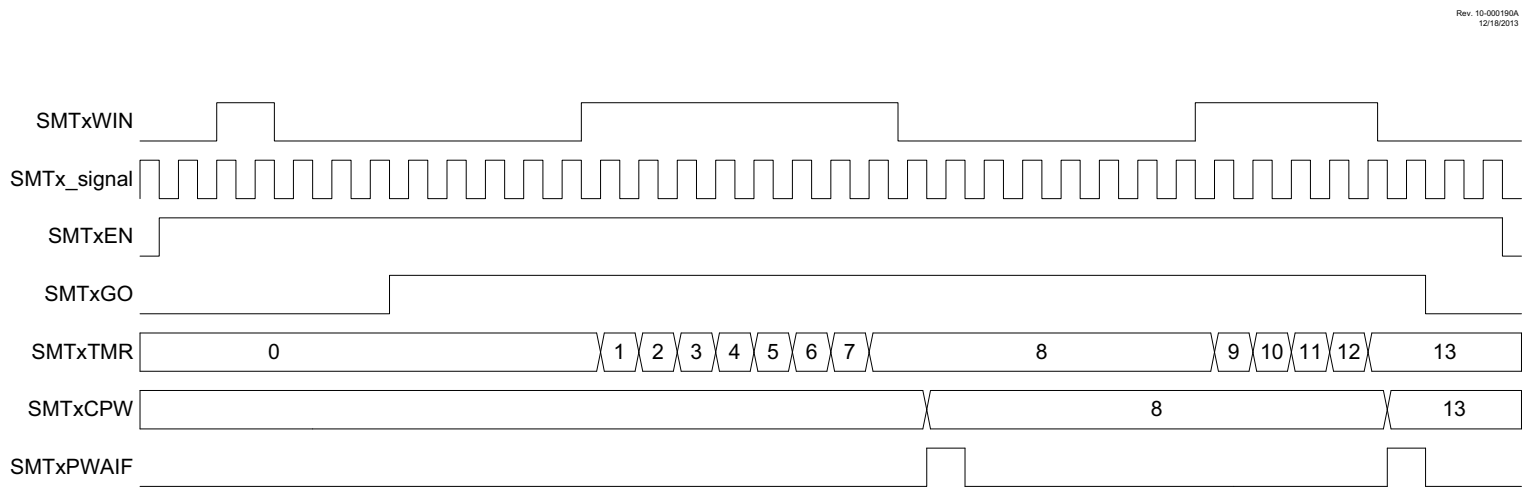
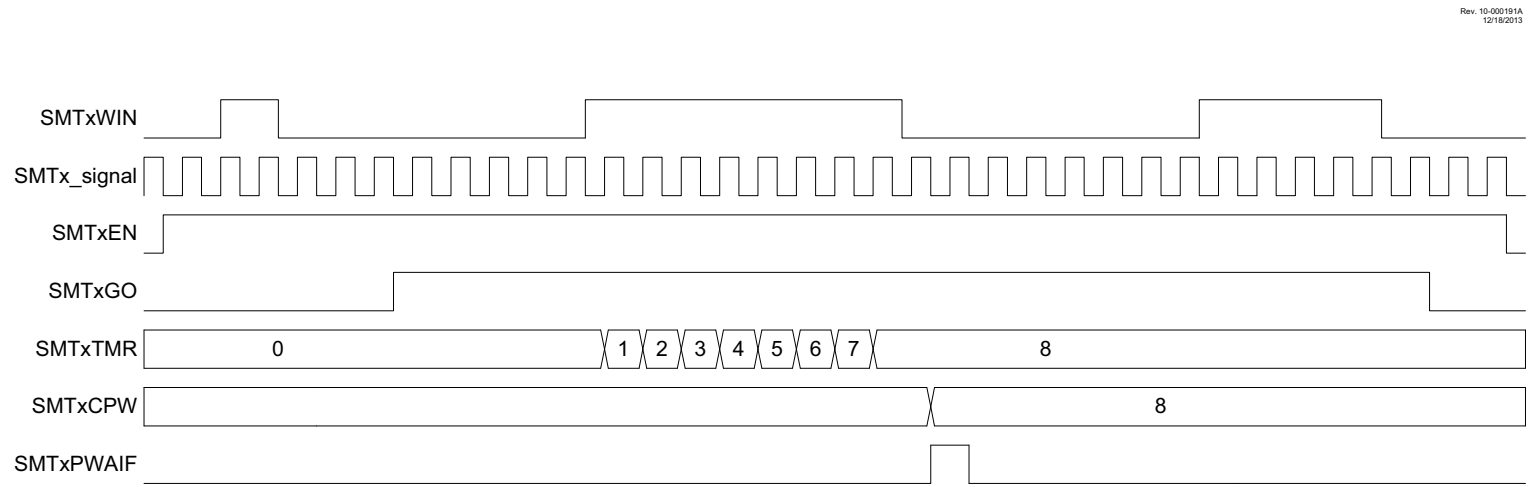


FIGURE 25-20: GATED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM

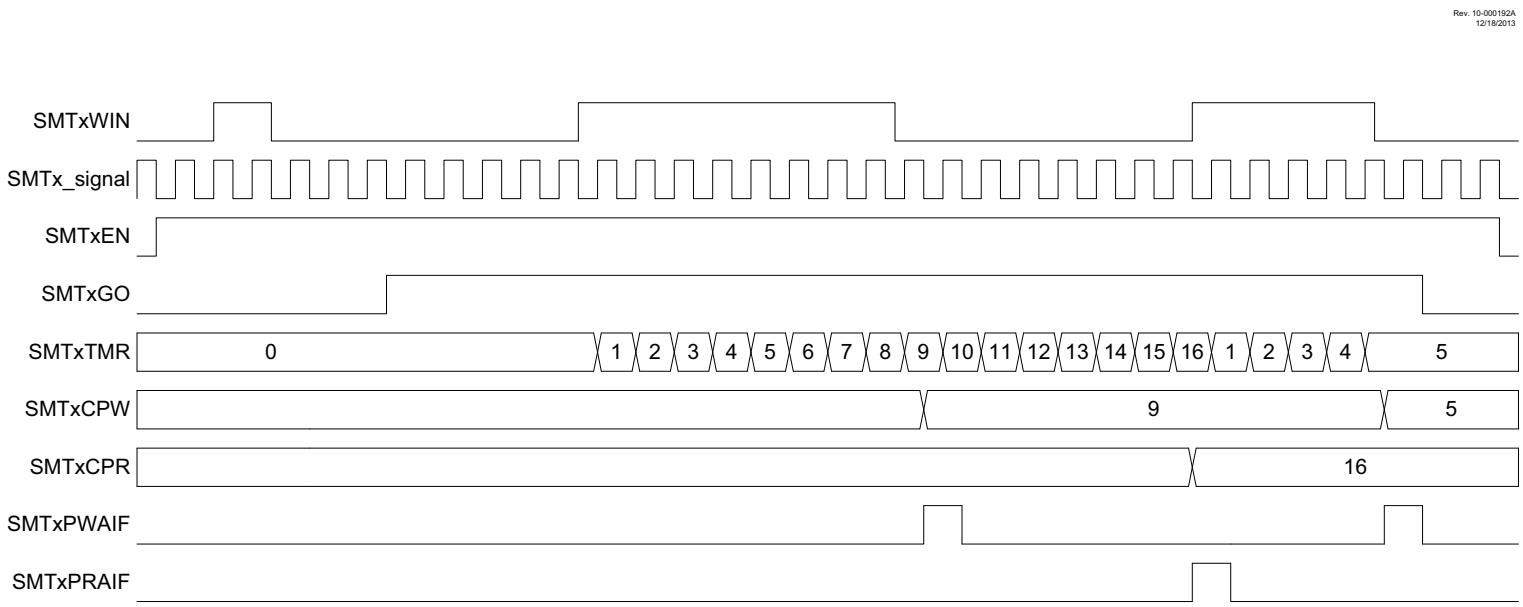


© 2017-2020 Microchip Technology Inc.

25.6.11 WINDOWED COUNTER MODE

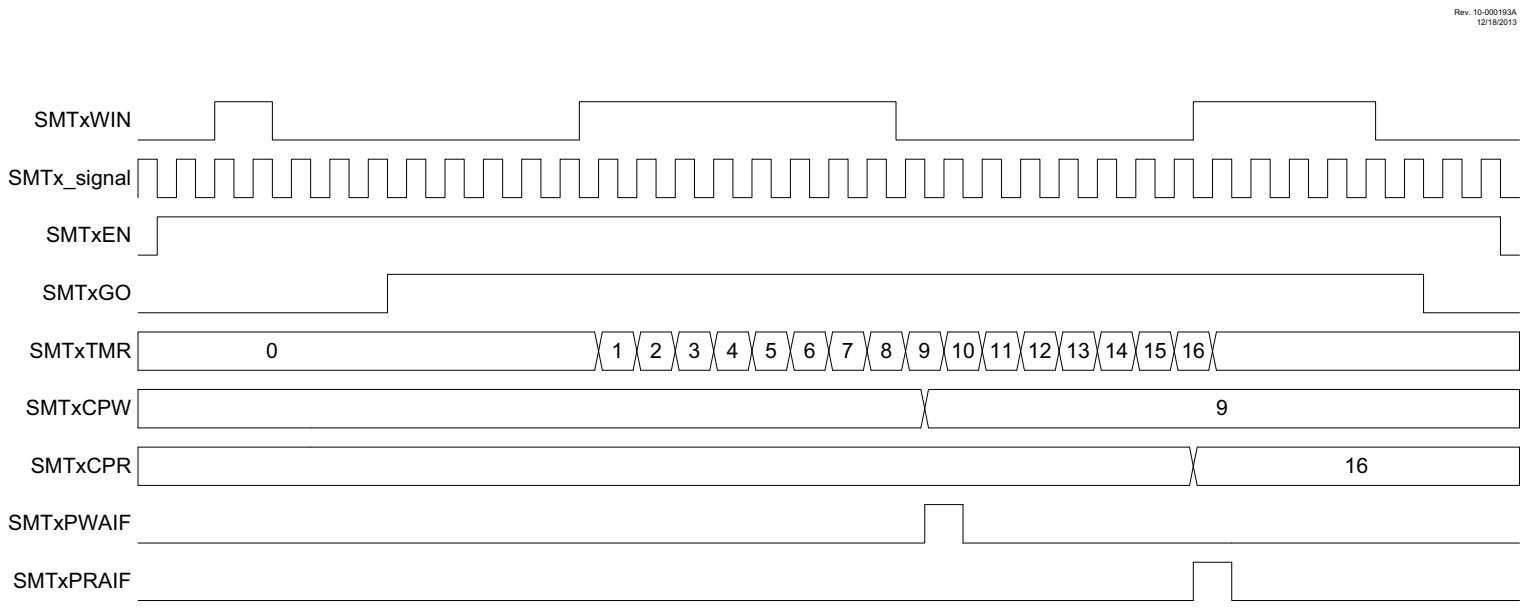
This mode counts pulses on the SMT1_signal input, within a window dictated by the SMT1WIN input. It begins counting upon seeing a rising edge of the SMT1WIN input, updates the SMT1CPW register on a falling edge of the SMT1WIN input, and updates the SMT1CPR register on each rising edge of the SMT1WIN input beyond the first. See Figure 25-21 and Figure 25-22.

FIGURE 25-21: WINDOWED COUNTER MODE REPEAT ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

FIGURE 25-22: WINDOWED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM



© 2017-2020 Microchip Technology Inc.

25.7 Interrupts

The SMT can trigger an interrupt under three different conditions:

- PW Acquisition Complete
- PR Acquisition Complete
- Counter Period Match

The interrupts are controlled by the PIR and PIE registers of the device.

25.7.1 PW AND PR ACQUISITION INTERRUPTS

The SMT can trigger interrupts whenever it updates the SMT1CPW and SMT1CPR registers, the circumstances for which are dependent on the SMT mode, and are discussed in each mode's specific section. The SMT1CPW interrupt is controlled by SMT1PWAIF and SMT1PWAIE bits in the respective PIR and PIE registers. The SMT1CPR interrupt is controlled by the SMT1PRAIF and SMT1PRAIE bits, also located in the respective PIR and PIE registers.

In synchronous SMT modes, the interrupt trigger is synchronized to the SMT1CLK. In Asynchronous modes, the interrupt trigger is asynchronous. In either mode, once triggered, the interrupt will be synchronized to the CPU clock.

25.7.2 COUNTER PERIOD MATCH INTERRUPT

As described in **Section 25.1.2 "Period Match interrupt"**, the SMT will also interrupt upon SMT1TMR, matching SMT1PR with its period match limit functionality described in **Section 25.3 "Halt Operation"**. The period match interrupt is controlled by SMT1IF and SMT1IE, located in the respective PIR and PIE registers.

25.8 Register Definitions: SMT Control

Long bit name prefixes for the Signal Measurement Timer peripherals are shown in **Section 1.3 "Register and Bit naming conventions"**.

TABLE 25-2: LONG BIT NAMES PREFIXES FOR SMT PERIPHERALS

Peripheral	Bit Name Prefix
SMT1	SMT1

REGISTER 25-1: SMT1CON0: SMT CONTROL REGISTER 0

R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN(1)	—	STP	WPOL	SPOL	CPOL	PS[1:0]	
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 **EN:** SMT Enable bit(1)
1 = SMT is enabled
0 = SMT is disabled; internal states are reset, clock requests are disabled

bit 6 **Unimplemented**: Read as '0'

bit 5 **STP:** SMT Counter Halt Enable bit
When SMT1TMR = SMT1PR:
1 = Counter remains SMT1PR; period match interrupt occurs when clocked
0 = Counter resets to 24'h000000; period match interrupt occurs when clocked

bit 4 **WPOL:** SMT1WIN Input Polarity Control bit
1 = SMT1WIN signal is active-low/falling edge enabled
0 = SMT1WIN signal is active-high/rising edge enabled

bit 3 **SPOL:** SMT1SIG Input Polarity Control bit
1 = SMT1_signal is active-low/falling edge enabled
0 = SMT1_signal is active-high/rising edge enabled

bit 2 **CPOL:** SMT Clock Input Polarity Control bit
1 = SMT1TMR increments on the falling edge of the selected clock signal
0 = SMT1TMR increments on the rising edge of the selected clock signal

bit 1-0 **PS[1:0]:** SMT Prescale Select bits
11 = Prescaler = 1:8
10 = Prescaler = 1:4
01 = Prescaler = 1:2
00 = Prescaler = 1:1

Note 1: Setting EN to '0' does not affect the register contents.

REGISTER 25-2: SMT1CON1: SMT CONTROL REGISTER 1

R/W/HC-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
GO	REPEAT	—	—	MODE[3:0]			
bit 7							bit 0

Legend:		
HC = Bit is cleared by hardware		HS = Bit is set by hardware
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 **GO:** GO Data Acquisition bit
1 = Incrementing, acquiring data is enabled
0 = Incrementing, acquiring data is disabled

bit 6 **REPEAT:** SMT Repeat Acquisition Enable bit
1 = Repeat Data Acquisition mode is enabled
0 = Single Acquisition mode is enabled

bit 5-4 **Unimplemented**: Read as ‘0’

bit 3-0 **MODE[3:0]** SMT Operation Mode Select bits
1111 = Reserved
•
•
•
1011 = Reserved
1010 = Windowed counter
1001 = Gated counter
1000 = Counter
0111 = Capture
0110 = Time of flight
0101 = Gated windowed measure
0100 = Windowed measure
0011 = High and low time measurement
0010 = Period and Duty-Cycle Acquisition
0001 = Gated Timer
0000 = Timer

REGISTER 25-3: SMT1STAT: SMT STATUS REGISTER

R/W/HC-0/0	R/W/HC-0/0	R/W/HC-0/0	U-0	U-0	R-0/0	R-0/0	R-0/0
CPRUP	CPWUP	RST	—	—	TS	WS	AS
bit 7							bit 0

Legend:		
HC = Bit is cleared by hardware		HS = Bit is set by hardware
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 **CPRUP:** SMT Manual Period Buffer Update bit
1 = Request update to SMT1CPRx registers
0 = SMT1CPRx registers update is complete

bit 6 **CPWUP:** SMT Manual Pulse Width Buffer Update bit
1 = Request update to SMT1CPW registers
0 = SMT1CPW registers update is complete

bit 5 **RST:** SMT Manual Timer Reset bit
1 = Request Reset to SMT1TMR registers
0 = SMT1TMR registers update is complete

bit 4-3 **Unimplemented**: Read as ‘0’

bit 2 **TS:** GO Value Status bit
1 = SMT timer is incrementing
0 = SMT timer is not incrementing

bit 1 **WS:** SMT1WIN Value Status bit
1 = SMT window is open
0 = SMT window is closed

bit 0 **AS:** SMT_signal Value Status bit
1 = SMT acquisition is in progress
0 = SMT acquisition is not in progress

REGISTER 25-4: SMT1CLK: SMT CLOCK SELECTION REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	CSEL[2:0]		
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 7-3 **Unimplemented**: Read as ‘0’

bit 2-0 **CSEL[2:0]:** SMT Clock Selection bits

111 = Reference Clock Output
110 = SOSC
101 = MFINTOSC/16 (32 kHz)
100 = MFINTOSC (500 kHz)
011 = LFINTOSC
010 = HFINTOSC 16 MHz
001 = FOSC
000 = FOSC/4

REGISTER 25-5: SMT1WIN: SMT1 WINDOW INPUT SELECT REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	WSEL[4:0]				
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 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 **WSEL[4:0]:** SMT1 Window Selection bits
11111 = Reserved
•
•
•
11011 = Reserved
11010 = CLC4_out
11001 = CLC3_out
11000 = CLC2_out
10111 = CLC1_out
10110 = ZCD1_out
10101 = CMP2_out
10100 = CMP1_out
10011 = NCO1_out
10010 = Reserved
10001 = Reserved
10000 = PWM8_out
01111 = PWM7_out
01110 = PWM6_out
01101 = PWM5_out
01100 = CCP4_out
01011 = CCP3_out
01010 = CCP2_out
01001 = CCP1_out
01000 = TMR6_postscaled
00111 = TMR4_postscaled
00110 = TMR2_postscaled
00100 = CLKREF
00011 = SOSC
00010 = MFINTOSC/16 (32 kHz)
00001 = LFINTOSC
00000 = SMTxWINPPS

REGISTER 25-6: SMT1SIG: SMT1 SIGNAL INPUT SELECT REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	SSEL[4:0]				
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 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 **SSEL[4:0]:** SMT1 Signal Selection bits
11111 = Reserved
•
•
•
11010 = Reserved
11001 = CLC4_out
11000 = CLC3_out
10111 = CLC2_out
10110 = CLC1_out
10101 = ZCD1_out
10100 = CMP2_out
10011 = CMP1_out
10010 = NCO1_out
10001 = Reserved
10000 = Reserved
01111 = PWM8_out
01110 = PWM7_out
01101 = PWM6_out
01100 = PWM5_out
01011 = CCP4_out
01010 = CCP3_out
01001 = CCP2_out
01000 = CCP1_out
00111 = TMR6_postscaled
00110 = TMR5_postscaled
00101 = TMR4_postscaled
00100 = TMR3_postscaled
00011 = TMR2_postscaled
00010 = TMR1_postscaled
00001 = TMR0_overflow
00000 = SMTxSIGPPS

REGISTER 25-7: SMT1TMRL: SMT TIMER REGISTER – LOW BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SMT1TMR[7:0]							
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 7-0 **SMT1TMR[7:0]**: Significant bits of the SMT Counter – Low Byte

REGISTER 25-8: SMT1TMRH: SMT TIMER REGISTER – HIGH BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SMT1TMR[15:8]							
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 7-0 **SMT1TMR[15:8]**: Significant bits of the SMT Counter – High Byte

REGISTER 25-9: SMT1TMRU: SMT TIMER REGISTER – UPPER BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
SMT1TMR[23:16]							
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 7-0 **SMT1TMR[23:16]**: Significant bits of the SMT Counter – Upper Byte

REGISTER 25-10: SMT1CPRL: SMT CAPTURED PERIOD REGISTER – LOW BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMT1CPR[7:0]							
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 7-0 **SMT1CPR[7:0]**: Significant bits of the SMT Period Latch – Low Byte

REGISTER 25-11: SMT1CPRH: SMT CAPTURED PERIOD REGISTER – HIGH BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMT1CPR[15:8]							
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 7-0 **SMT1CPR[15:8]**: Significant bits of the SMT Period Latch – High Byte

REGISTER 25-12: SMT1CPRU: SMT CAPTURED PERIOD REGISTER – UPPER BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMT1CPR[23:16]							
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 7-0 **SMT1CPR[23:16]**: Significant bits of the SMT Period Latch – Upper Byte

REGISTER 25-13: SMT1CPWL: SMT CAPTURED PULSE WIDTH REGISTER – LOW BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMT1CPW[7:0]							
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 7-0 **SMT1CPW[7:0]**: Significant bits of the SMT PW Latch – Low Byte

REGISTER 25-14: SMT1CPWH: SMT CAPTURED PULSE WIDTH REGISTER – HIGH BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMT1CPW[15:8]							
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 7-0 **SMT1CPW[15:8]**: Significant bits of the SMT PW Latch – High Byte

REGISTER 25-15: SMT1CPWU: SMT CAPTURED PULSE WIDTH REGISTER – UPPER BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMT1CPW[23:16]							
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 7-0 **SMT1CPW[23:16]**: Significant bits of the SMT PW Latch – Upper Byte

REGISTER 25-16: SMT1PRL: SMT PERIOD REGISTER – LOW BYTE

R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1
SMT1PR[7:0]							
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 7-0 **SMT1PR[7:0]**: Significant bits of the SMT Timer Value for Period Match – Low Byte

REGISTER 25-17: SMT1PRH: SMT PERIOD REGISTER – HIGH BYTE

R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1
SMT1PR[15:8]							
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 7-0 **SMT1PR[15:8]**: Significant bits of the SMT Timer Value for Period Match – High Byte

REGISTER 25-18: SMT1PRU: SMT PERIOD REGISTER – UPPER BYTE

R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1	R/W-x/1
SMT1PR[23:16]							
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 7-0 **SMT1PR[23:16]**: Significant bits of the SMT Timer Value for Period Match – Upper Byte

TABLE 25-3: SUMMARY OF REGISTERS ASSOCIATED WITH SMT1

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
SMT1CON0	EN	—	STP	WPOL	SPOL	CPOL	SMT1PS[1:0]		395
SMT1CON1	GO	REPEAT	—	—	MODE[3:0]				396
SMT1STAT	CPRUP	CPWUP	RST	—	—	TS	WS	AS	397
SMT1CLK	—	—	—	—	—	CSEL[2:0]			398
SMT1SIG	—	—	—	SSEL[4:0]					400
SMT1WIN	—	—	—	WSEL[4:0]					399
SMT1TMRL	TMR[7:0]								401
SMT1TMRH	TMR[15:8]								401
SMT1TMRU	TMR[23:16]								401
SMT1CPRL	CPR[7:0]								402
SMT1CPRH	CPR[15:8]								402
SMT1CPRU	CPR[23:16]								402
SMT1CPWL	CPW[7:0]								403
SMT1CPWH	CPW[15:8]								403
SMT1CPWU	CPW[23:16]								403
SMT1PRL	PR[7:0]								404
SMT1PRH	PR[15:8]								404
SMT1PRU	PR[23:16]								404

Legend: — = unimplemented read as '0'. Shaded cells are not used for SMT1 module.

26.0 COMPLEMENTARY WAVEFORM GENERATOR (CWG) MODULE

The Complementary Waveform Generator (CWG) produces half-bridge, full-bridge, and steering of PWM waveforms. It is backwards compatible with previous CCP functions. There are three instances of the CWG module present on the device.

Each of the CWG modules has the following features:

- Six operating modes:
  - Synchronous Steering mode
  - Asynchronous Steering mode
  - Full-Bridge mode, Forward
  - Full-Bridge mode, Reverse
  - Half-Bridge mode
  - Push-Pull mode
- Output polarity control
- Output steering
- Independent 6-bit rising and falling event dead-band timers
  - Clocked dead band
  - Independent rising and falling dead-band enables
- Auto-shutdown control with:
  - Selectable shutdown sources
  - Auto-restart option
  - Auto-shutdown pin override control

26.1 Fundamental Operation

The CWG generates two output waveforms from the selected input source.

The off-to-on transition of each output can be delayed from the on-to-off transition of the other output, thereby creating a time delay immediately where neither output is driven. This is referred to as dead time and is covered in **Section 26.6 “Dead-Band Control”**.

It may be necessary to guard against the possibility of circuit faults or a feedback event arriving too late or not at all. In this case, the active drive must be terminated before the Fault condition causes damage. This is referred to as auto-shutdown and is covered in **Section 26.10 “Auto-Shutdown”**.

26.2 Operating Modes

The CWG module can operate in six different modes, as specified by the MODE[2:0] bits of the CWGxCON0 register:

- Half-Bridge mode
- Push-Pull mode
- Asynchronous Steering mode
- Synchronous Steering mode
- Full-Bridge mode, Forward
- Full-Bridge mode, Reverse

All modes accept a single pulse data input, and provide up to four outputs as described in the following sections.

All modes include auto-shutdown control as described in **Section 26.10 “Auto-Shutdown”**.

Note:	Except as noted for Full-bridge mode (**Section 26.2.3 “Full-Bridge Modes”**), mode changes should only be performed while EN = 0 (Register 26-1).

26.2.1 HALF-BRIDGE MODE

In Half-Bridge mode, two output signals are generated as true and inverted versions of the input as illustrated in Figure 26-2. A non-overlap (dead-band) time is inserted between the two outputs as described in **Section 26.6 “Dead-Band Control”**. The output steering feature cannot be used in this mode. A basic block diagram of this mode is shown in Figure 26-1.

The unused outputs CWGxC and CWGxD drive similar signals as CWGxA and CWGxB, with polarity independently controlled by the POLC and POLD bits of the CWGxCON1 register, respectively.

FIGURE 26-1: SIMPLIFIED CWG BLOCK DIAGRAM (HALF-BRIDGE MODE, MODE[2:0] = 100)

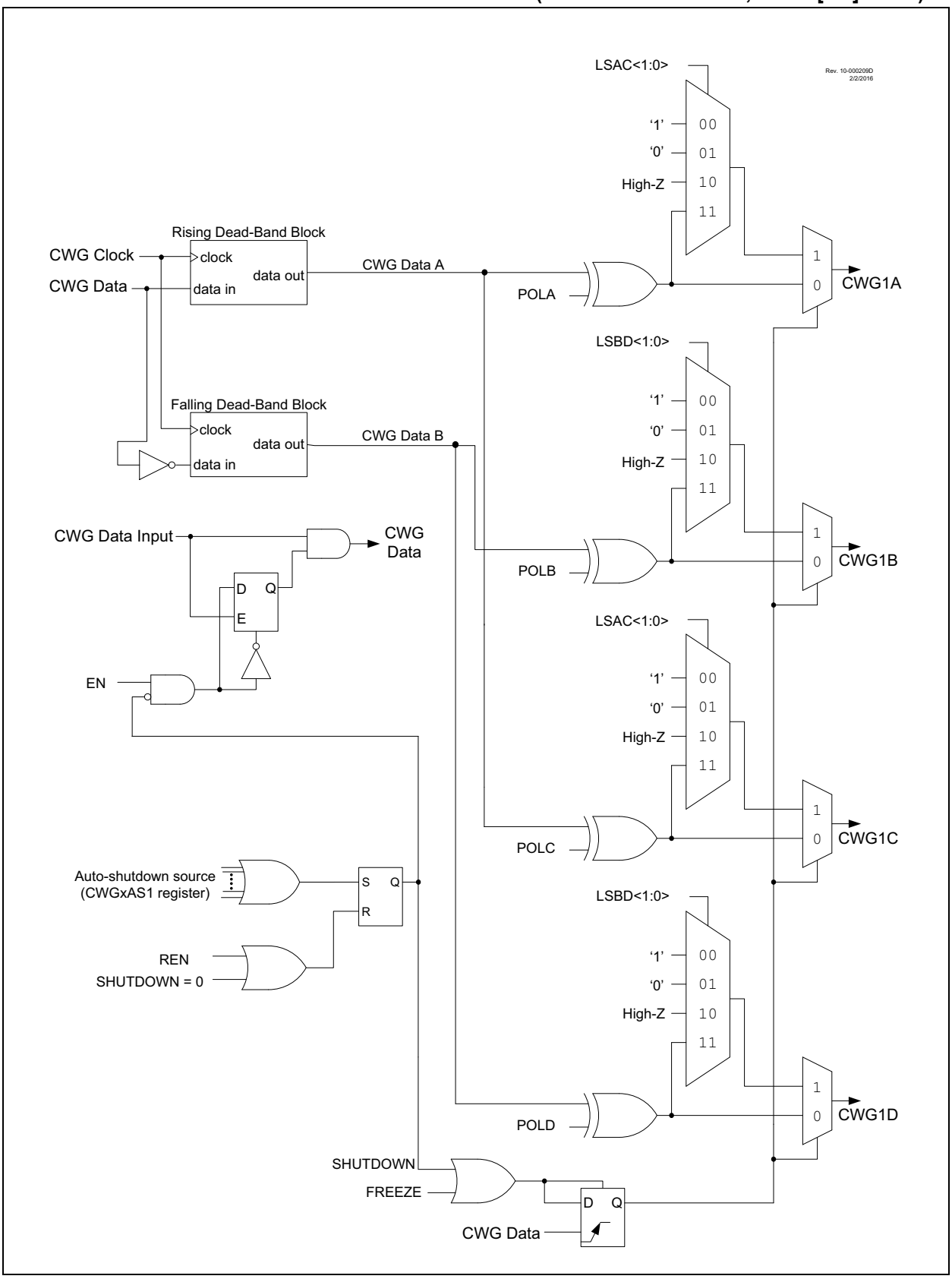
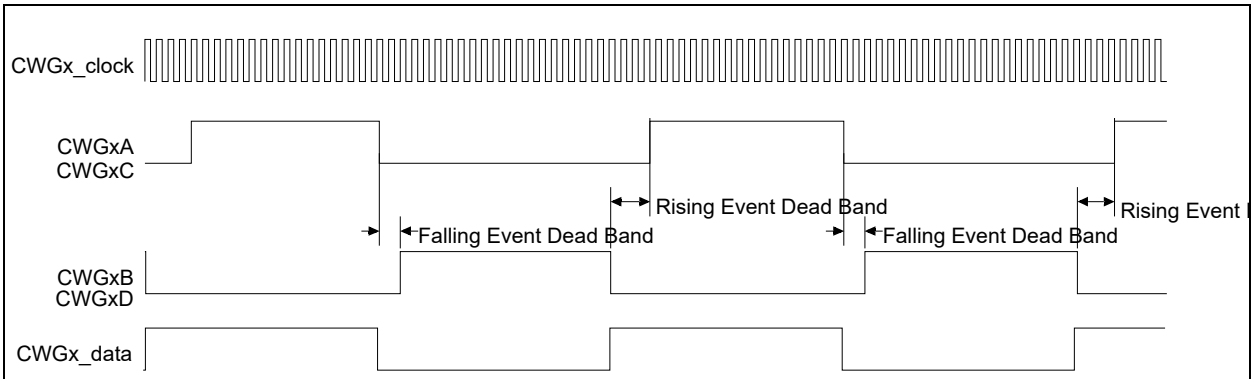


FIGURE 26-2: CWGx HALF-BRIDGE MODE OPERATION



Note: CWGx_rising_src = CCP1_out, CWGx_falling_src = ~CCP1_out

26.2.2 PUSH-PULL MODE

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in Figure 26-4. This alternation creates the push-pull effect required for driving some transformer-based power supply designs. Steering modes are not used in Push-Pull mode. A basic block diagram for the Push-Pull mode is shown in Figure 26-3.

The push-pull sequencer is reset whenever EN = 0 or if an auto-shutdown event occurs. The sequencer is clocked by the first input pulse, and the first output appears on CWGxA.

The unused outputs CWGxC and CWGxD drive copies of CWGxA and CWGxB, respectively, but with polarity controlled by the POLC and POLD bits of the CWGxCON1 register, respectively.

FIGURE 26-3: SIMPLIFIED CWG BLOCK DIAGRAM (PUSH-PULL MODE, MODE[2:0] = 101)

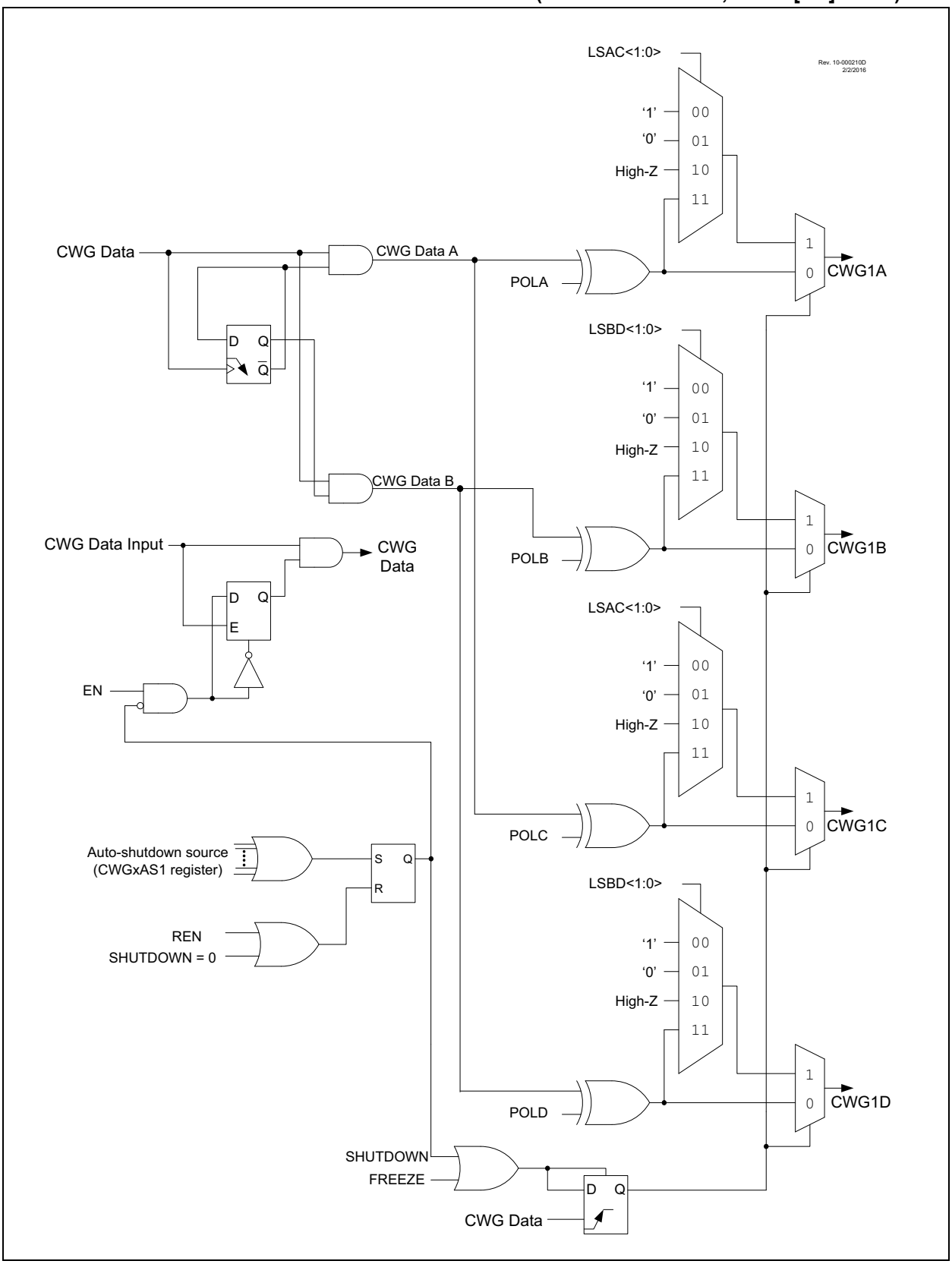
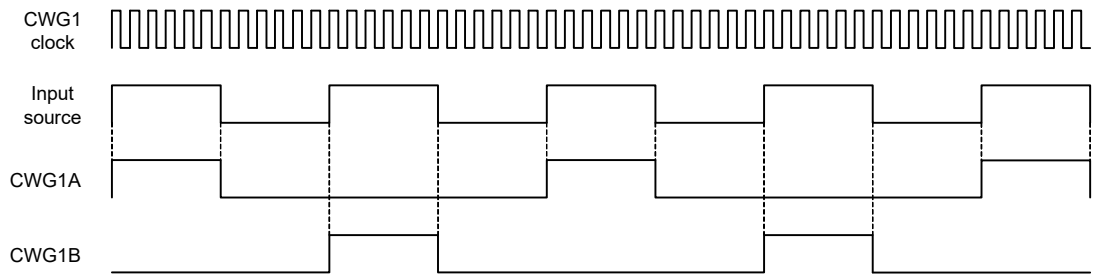


FIGURE 26-4: CWGx PUSH-PULL MODE OPERATION



26.2.3 FULL-BRIDGE MODES

In Forward and Reverse Full-Bridge modes, three outputs drive static values while the fourth is modulated by the input data signal. The mode selection may be toggled between forward and reverse by toggling the MODE[0] bit of the CWGxCON0 while keeping MODE[2:1] static, without disabling the CWG module. When connected as shown in Figure 26-5, the outputs are appropriate for a full-bridge motor driver. Each CWG output signal has independent polarity control, so the circuit can be adapted to high-active and low-active drivers. A simplified block diagram for the Full-Bridge modes is shown in Figure 26-6.

FIGURE 26-5: EXAMPLE OF FULL-BRIDGE APPLICATION

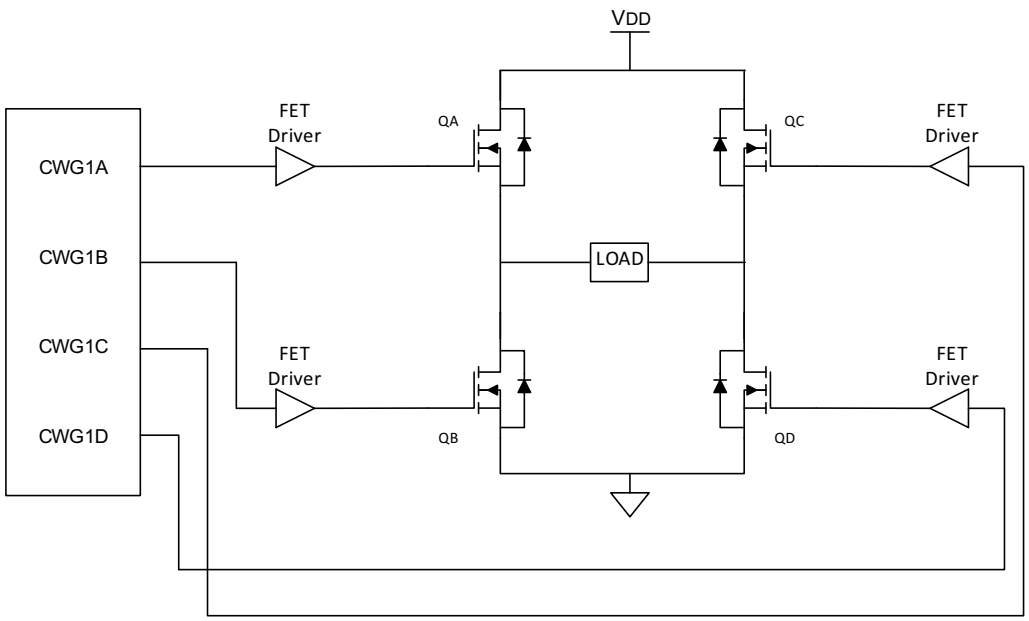
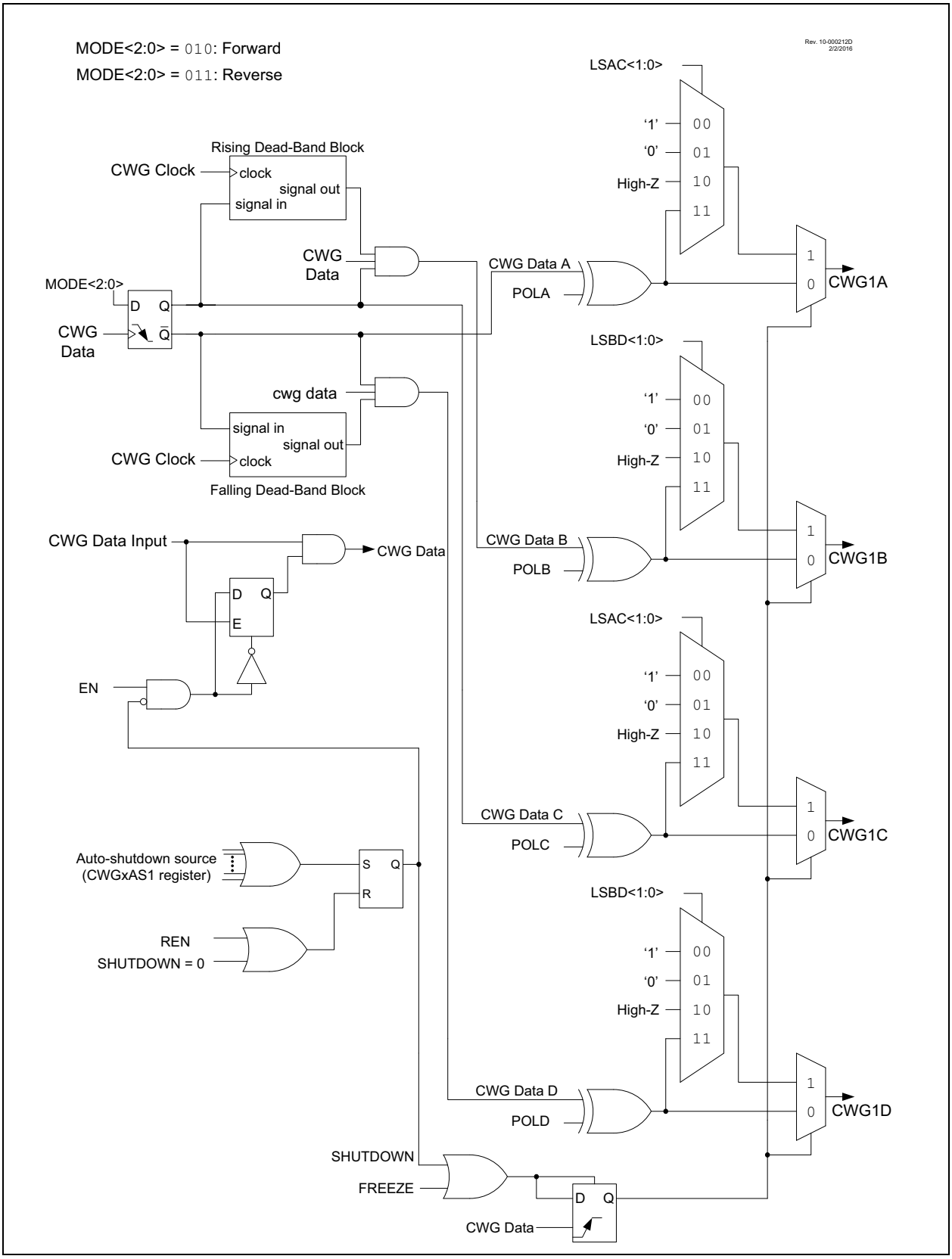


FIGURE 26-6: SIMPLIFIED CWG BLOCK DIAGRAM (FORWARD AND REVERSE FULL-BRIDGE MODES)

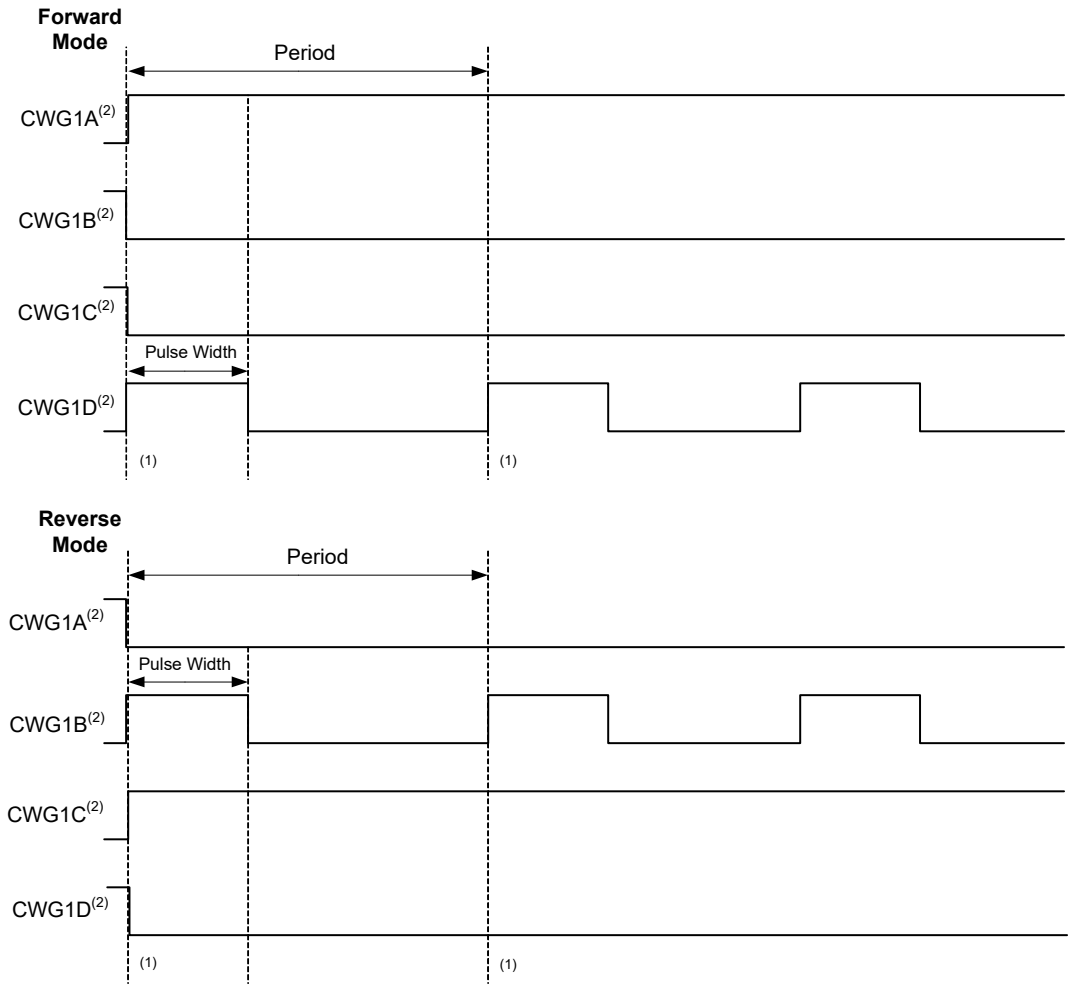


In Forward Full-Bridge mode (MODE[2:0] = 010), CWGxA is driven to its active state, CWGxB and CWGxC are driven to their inactive state, and CWGxD is modulated by the input signal, as shown in Figure 26-7.

In Reverse Full-Bridge mode (MODE[2:0] = 011), CWGxC is driven to its active state, CWGxA and CWGxD are driven to their inactive states, and CWGxB is modulated by the input signal, as shown in Figure 26-7.

In Full-Bridge mode, the dead-band period is used when there is a switch from forward to reverse or vice-versa. This dead-band control is described in **Section 26.6 "Dead-Band Control"**, with additional details in **Section 26.7 "Rising Edge and Reverse Dead Band"** and **Section 26.8 "Falling Edge and Forward Dead Band"**. Steering modes are not used with either of the Full-Bridge modes. The mode selection may be toggled between forward and reverse toggling the MODE[0] bit of the CWGxCON0 while keeping MODE[2:1] static, without disabling the CWG module.

FIGURE 26-7: EXAMPLE OF FULL-BRIDGE OUTPUT



Note 1: A rising CWG data input creates a rising event on the modulated output.
2: Output signals shown as active-high; all POLy bits are clear.

26.2.3.1 Direction Change in Full-Bridge Mode

In Full-Bridge mode, changing MODE[2:0] controls the forward/reverse direction. Changes to MODE[2:0] change to the new direction on the next rising edge of the modulated input.

A direction change is initiated in software by changing the MODE[2:0] bits of the CWGxCON0 register. The sequence is illustrated in Figure 26-8.

- The associated active output CWGxA and the inactive output CWGxC are switched to drive in the opposite direction.
- The previously modulated output CWGxD is switched to the inactive state, and the previously inactive output CWGxB begins to modulate.
- CWG modulation resumes after the direction-switch dead band has elapsed.

26.2.3.2 Dead-Band Delay in Full-Bridge Mode

Dead-band delay is important when either of the following conditions is true:

1. The direction of the CWG output changes when the duty cycle of the data input is at or near 100%, or
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

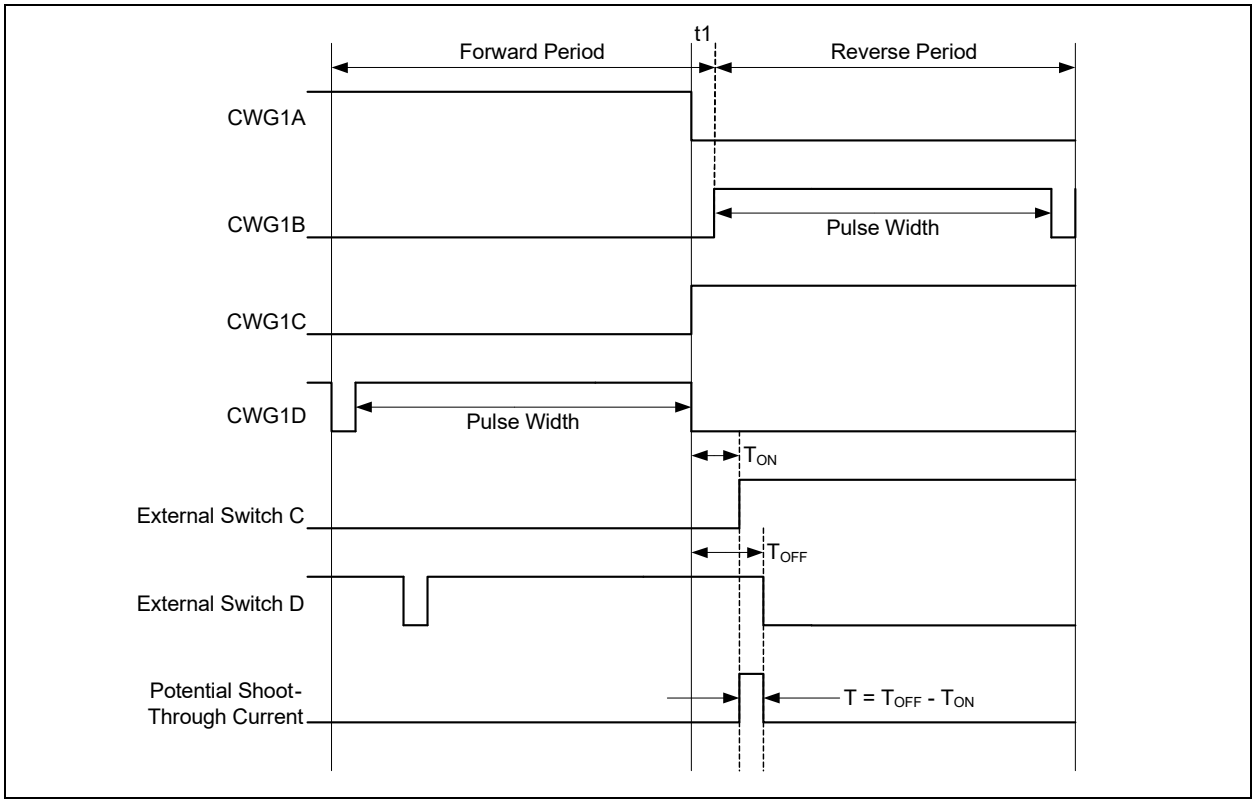
The dead-band delay is inserted only when changing directions, and only the modulated output is affected. The statically-configured outputs (CWGxA and CWGxC) are not afforded dead band, and switch essentially simultaneously.

Figure 26-8 shows an example of the CWG outputs changing directions from forward to reverse, at near 100% duty cycle. In this example, at time t1, the output of CWGxA and CWGxD become inactive, while output CWGxC becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices QC and QD for the duration of ‘t’. The same phenomenon will occur to power devices QA and QB for the CWG direction change from reverse to forward.

When changing the CWG direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

1. Reduce the CWG duty cycle for one period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

FIGURE 26-8: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



26.2.4 STEERING MODES

In both Synchronous and Asynchronous Steering modes, the modulated input signal can be steered to any combination of four CWG outputs and a fixed-value will be presented on all the outputs not used for the PWM output. Each output has independent polarity, steering, and shutdown options. Dead-band control is not used in either steering mode.

When STRx = 0 (Register 26-5), then the corresponding pin is held at the level defined by OVRx (Register 26-5). When STRx = 1, then the pin is driven by the modulated input signal.

The POLx bits (Register 26-2) control the signal polarity only when STRx = 1.

The CWG auto-shutdown operation also applies to steering modes as described in **Section 26.14 "Register Definitions: CWG Control"**.

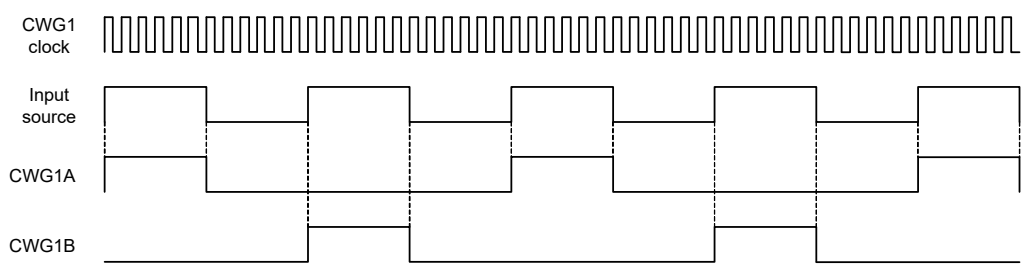
Note:	Only the STRx bits are synchronized; the SDATx (data) bits are not synchronized.

The CWG auto-shutdown operation also applies in Steering modes as described in **Section 26.10 "Auto-Shutdown"**. An auto-shutdown event will only affect pins that have STRx = 1.

26.2.4.1 Synchronous Steering Mode

In Synchronous Steering mode (MODE[2:0] bits = 001, Register 26-1), changes to steering selection registers take effect on the next rising edge of the modulated data input (Figure 26-9). In Synchronous Steering mode, the output will always produce a complete waveform.

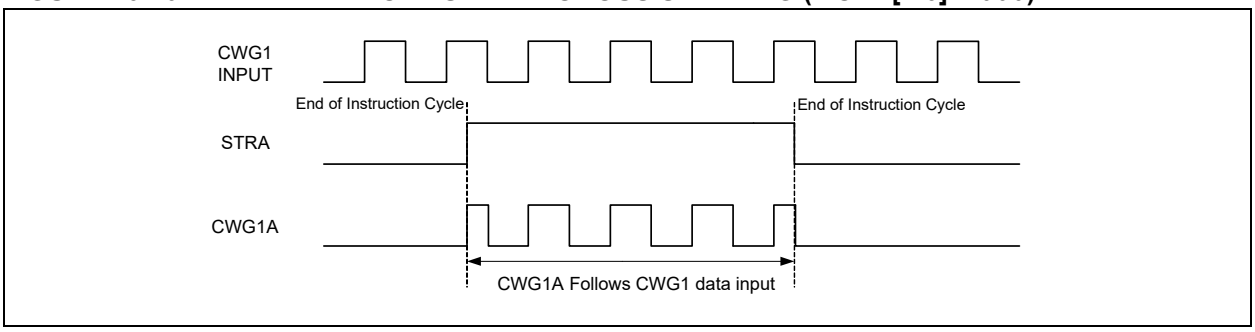
FIGURE 26-9: EXAMPLE OF SYNCHRONOUS STEERING (MODE[2:0] = 001)



26.2.4.2 Asynchronous Steering Mode

In Asynchronous mode (MODE[2:0] bits = 000, Register 26-1), steering takes effect at the end of the instruction cycle that writes to STR. In Asynchronous Steering mode, the output signal may be an incomplete waveform (Figure 26-10). This operation may be useful when the user firmware needs to immediately remove a signal from the output pin.

FIGURE 26-10: EXAMPLE OF ASYNCHRONOUS STEERING (MODE[2:0] = 000)

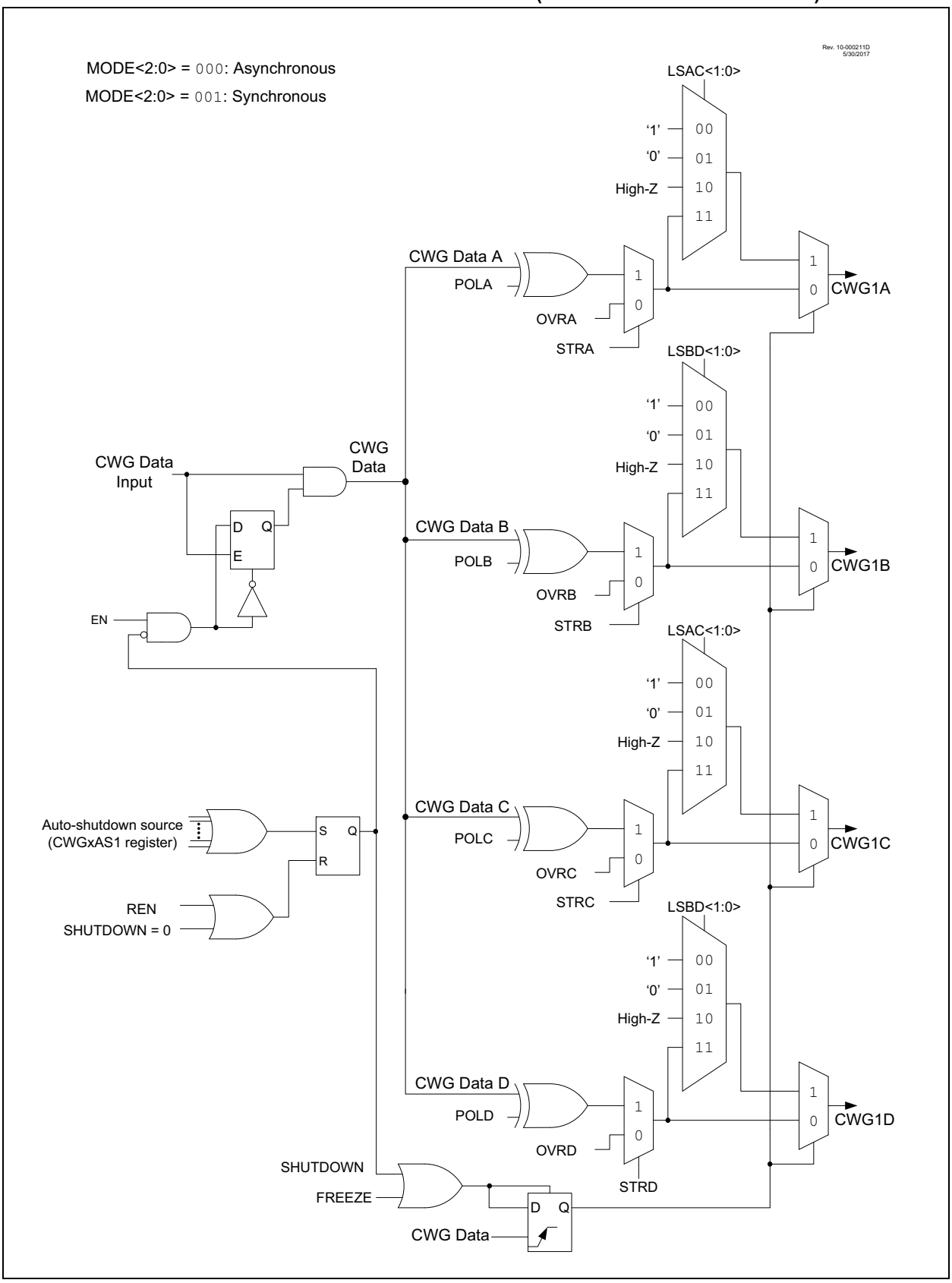


26.2.4.3 Start-up Considerations

The application hardware must use the proper external pull-up and/or pull-down resistors on the CWG output pins. This is required because all I/O pins are forced to high-impedance at Reset.

The POLy bits (Register 26-2) allow the user to choose whether the output signals are active-high or active-low.

FIGURE 26-11: SIMPLIFIED CWG BLOCK DIAGRAM (OUTPUT STEERING MODES)



26.3 Clock Source

The clock source is used to drive the dead-band timing circuits. The CWG module allows the following clock sources to be selected:

- FOSC (system clock)
- HFINTOSC

When the HFINTOSC is selected, the HFINTOSC will be kept running during Sleep. Therefore, CWG modes requiring dead band can operate in Sleep, provided that the CWG data input is also active during Sleep.The clock sources are selected using the CS bit of the CWGxCLKCON register (Register 26-3). The system clock FOSC, is disabled in Sleep and thus dead-band control cannot be used.

26.4 Selectable Input Sources

The CWG generates the output waveforms from the following input sources:

TABLE 26-1: SELECTABLE INPUT SOURCES

Source Peripheral	Signal Name	ISM[2:0]
CWGxPPS	Pin selected by CWGxPPS	000
CCP1	CCP1 Output	001
CCP2	CCP2 Output	010
PWM3	PWM3 Output	011
PWM4	PWM4 Output	100
CMP1	Comparator 1 Output	101
CMP2	Comparator 2 Output	110
DSM	Data signal modulator output	111

The input sources are selected using the IS[4:0] bits in the CWGxISM register (Register 26-4).

26.5 Output Control

26.5.1 CWG OUTPUTS

Each CWG output can be routed to a Peripheral Pin Select (PPS) output via the RxyPPS register (see **Section 17.0 “Peripheral Pin Select (PPS) Module”**).

26.5.2 POLARITY CONTROL

The polarity of each CWG output can be selected independently. When the output polarity bit is set, the corresponding output is active-high. Clearing the output polarity bit configures the corresponding output as active-low. However, polarity does not affect the override levels. Output polarity is selected with the POLy bits of the CWGxCON1. Auto-shutdown and steering options are unaffected by polarity.

26.6 Dead-Band Control

The dead-band control provides non-overlapping PWM signals to prevent shoot-through current in PWM switches. Dead-band operation is employed for Half-Bridge and Full-Bridge modes. The CWG contains two 6-bit dead-band counters. One is used for the rising edge of the input source control in Half-Bridge mode or for reverse dead-band Full-Bridge mode. The other is used for the falling edge of the input source control in Half-Bridge mode or for forward dead band in Full-Bridge mode.

Dead band is timed by counting CWG clock periods from zero up to the value in the rising or falling dead-band counter registers. See CWGxDBR and CWGxDBF registers, respectively.

26.6.1 DEAD-BAND FUNCTIONALITY IN HALF-BRIDGE MODE

In Half-Bridge mode, the dead-band counters dictate the delay between the falling edge of the normal output and the rising edge of the inverted output. This can be seen in Figure 26-2.

26.6.2 DEAD-BAND FUNCTIONALITY IN FULL-BRIDGE MODE

In Full-Bridge mode, the dead-band counters are used when undergoing a direction change. The MODE[0] bit of the CWGxCON0 register can be set or cleared while the CWG is running, allowing for changes from Forward to Reverse mode. The CWGxA and CWGxC signals will change immediately upon the first rising input edge following a direction change, but the modulated signals (CWGxB or CWGxD, depending on the direction of the change) will experience a delay dictated by the dead-band counters.

© 2017-2020 Microchip Technology Inc.

26.7 Rising Edge and Reverse Dead Band

In Half-Bridge mode, the rising edge dead band delays the turn-on of the CWGxA output after the rising edge of the CWG data input. In Full-Bridge mode, the reverse dead-band delay is only inserted when changing directions from Forward mode to Reverse mode, and only the modulated output CWGxB is affected.

The CWGxDBR register determines the duration of the dead-band interval on the rising edge of the input source signal. This duration is from 0 to 64 periods of the CWG clock.

Dead band is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWGxDBR register value is double-buffered. When EN = 0 (Register 26-1), the buffer is loaded when CWGxDBR is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input, after the LD bit (Register 26-1) is set. Refer to Figure 26-12 for an example.

26.8 Falling Edge and Forward Dead Band

In Half-Bridge mode, the falling edge dead band delays the turn-on of the CWGxB output at the falling edge of the CWG data input. In Full-Bridge mode, the forward dead-band delay is only inserted when changing directions from Reverse mode to Forward mode, and only the modulated output CWGxD is affected.

The CWGxDBF register determines the duration of the dead-band interval on the falling edge of the input source signal. This duration is from zero to 64 periods of CWG clock.

Dead-band delay is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWGxDBF register value is double-buffered. When EN = 0 (Register 26-1), the buffer is loaded when CWGxDBF is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input after the LD (Register 26-1) is set. Refer to Figure 26-13 for an example.

FIGURE 26-12: DEAD-BAND OPERATION, CWGxDBR = 0x01, CWGxDBF = 0x02

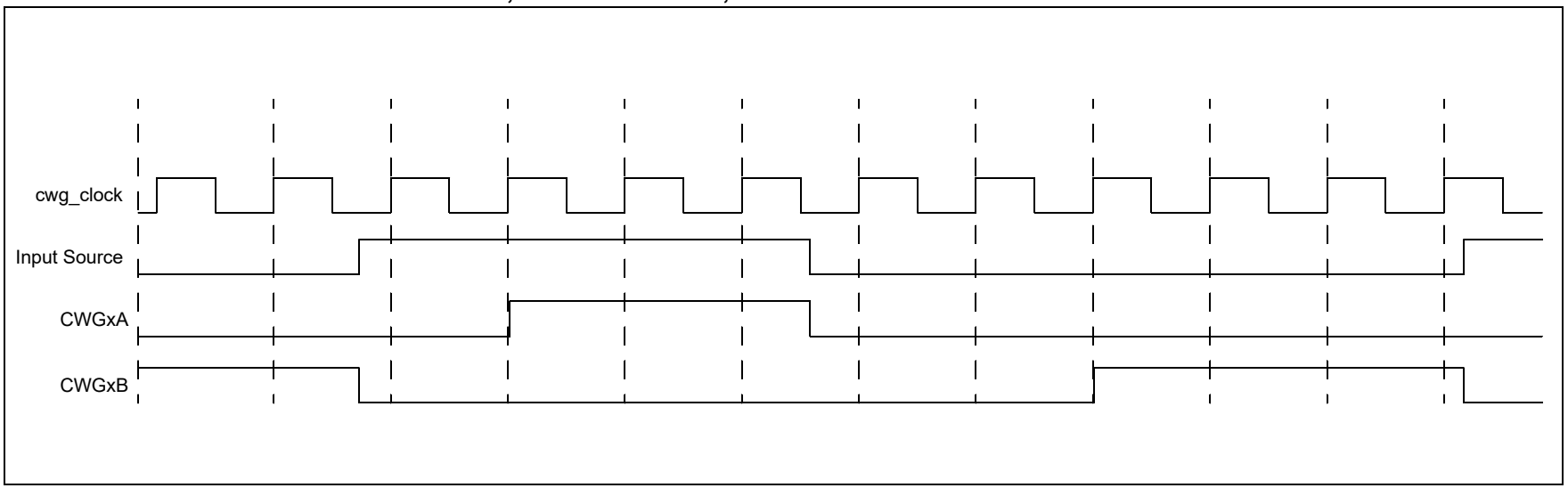
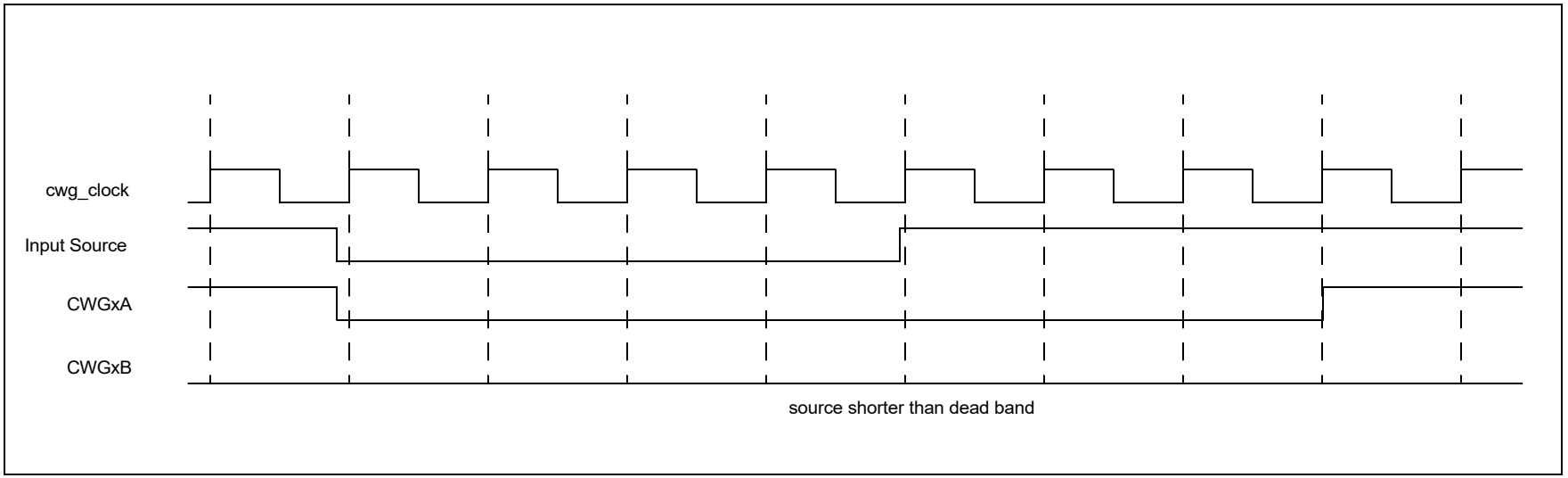


FIGURE 26-13: DEAD-BAND OPERATION, CWGxDBR = 0x03, CWGxDBF = 0x06, SOURCE SHORTER THAN DEAD BAND



© 2017-2020 Microchip Technology Inc.

26.9 Dead-Band Jitter

When the rising and falling edges of the input source are asynchronous to the CWG clock, it creates jitter in the dead-band time delay. The maximum jitter is equal to one CWG clock period. Refer to Equation 26-1 for more details.

EQUATION 26-1: DEAD-BAND DELAY TIME CALCULATION

$$T_{DEAD-BAND_MIN} = \frac{1}{F_{CWG\ CLOCK}} \bullet DBx<4:0>$$

$$T_{DEAD-BANDMAX} = \frac{1}{F_{CWG\ CLOCK}} \bullet DBx<4:0>+1$$

$$T_{JITTER} = T_{DEAD-BAND_MAX} - T_{DEAD-BAND_MIN}$$

$$T_{JITTER} = \frac{1}{F_{CWG_CLOCK}}$$

$$T_{DEAD-BAND_MAX} = T_{DEAD-BAND_MIN} + T_{JITTER}$$

$$EXAMPLE$$

$$DBR<4:0> = 0x0A = 10$$

$$F_{CWG_CLOCK} = 8\ MHz$$

$$T_{JITTER} = \frac{1}{8MHz} = 125\ \text{ns}$$

$$T_{DEAD-BAND_MIN} = 125\ ns * 10 = 125\ \mu s$$

$$T_{DEAD-BAND_MAX} = 1.25\ \mu s + 0.125 \mu s = 1.37 \mu s$$

26.10 Auto-Shutdown

Auto-shutdown is a method to immediately override the CWG output levels with specific overrides that allow for safe shutdown of the circuit. The shutdown state can be either cleared automatically or held until cleared by software. The auto-shutdown circuit is illustrated in Figure 26-14.

26.10.1 SHUTDOWN

The shutdown state can be entered by either of the following two methods:

- Software generated
- External Input

26.10.1.1 Software Generated Shutdown

Setting the SHUTDOWN bit of the CWGxAS0 register will force the CWG into the shutdown state.

When the auto-restart is disabled, the shutdown state will persist as long as the SHUTDOWN bit is set.

When auto-restart is enabled, the SHUTDOWN bit will clear automatically and resume operation on the next rising edge event. The SHUTDOWN bit indicates when a shutdown condition exists. The bit may be set or cleared in software or by hardware.

26.10.1.2 External Input Source

External shutdown inputs provide the fastest way to safely suspend CWG operation in the event of a Fault condition. When any of the selected shutdown inputs goes active, the CWG outputs will immediately go to the specified override levels without software delay. The override levels are selected by the LSBD[1:0] and LSAC[1:0] bits of the CWGxAS0 register (Register 26-6). Several input sources can be selected to cause a shutdown condition. All input sources are active-low. The sources are:

- Pin selected by CWGxPPS
- Timer2 postscaled output
- Timer4 postscaled output
- Timer6 postscaled output
- Comparator 1 output
- Comparator 2 output
- CLC2 output

Shutdown input sources are individually enabled by the ASxE bits of the CWGxAS1 register (Register 26-7).

Note:	Shutdown inputs are level sensitive, not edge sensitive. The shutdown state cannot be cleared, except by disabling auto-shutdown, as long as the shutdown input level persists.

26.10.1.3 Pin Override Levels

The levels driven to the CWG outputs during an auto-shutdown event are controlled by the LSBD[1:0] and LSAC[1:0] bits of the CWGxAS0 register (Register 26-6). The LSBD[1:0] bits control CWGxB/D output levels, while the LSAC[1:0] bits control the CWGxA/C output levels.

26.10.1.4 Auto-Shutdown Interrupts

When an auto-shutdown event occurs, either by software or hardware setting SHUTDOWN, the CWGxIF flag bit of the respective PIR register is set.

26.11 Auto-Shutdown Restart

After an auto-shutdown event has occurred, there are two ways to resume operation:

- Software controlled
- Auto-restart

In either case, the shutdown source must be cleared before the restart can take place. That is, either the shutdown condition must be removed, or the corresponding ASxE bit must be cleared.

26.11.1 SOFTWARE-CONTROLLED RESTART

If the REN bit of the CWGxAS0 register is clear (REN = `0`), the CWG module must be restarted after an auto-shutdown event through software.

Once all auto-shutdown sources are removed, the software must clear SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

Note:	The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

26.11.2 AUTO-RESTART

If the REN bit of the CWGxAS0 register is set (REN = `1`), the CWG module will restart from the shutdown state automatically.

Once all auto-shutdown conditions are removed, the hardware will automatically clear SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

Note:	The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

26.12 Operation During Sleep

The CWG module operates independently from the system clock and will continue to run during Sleep, provided that the clock and input sources selected remain active.

The HFINTOSC remains active during Sleep when all the following conditions are met:

- CWG module is enabled
- Input source is active
- HFINTOSC is selected as the clock source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as system clock and CWG clock, when the CWG is enabled and the input source is active, then the CPU will go idle during Sleep, but the HFINTOSC will remain active and the CWG will continue to operate. This will have a direct effect on the Sleep mode current.

26.13 Configuring the CWG

1. Ensure that the TRIS control bits corresponding to CWG outputs are set so that all are configured as inputs, ensuring that the outputs are inactive during setup. External hardware should ensure that pin levels are held to safe levels.
2. Clear the EN bit, if not already cleared.
3. Configure the MODE[2:0] bits of the CWGx-CON0 register to set the output operating mode.
4. Configure the POLy bits of the CWGxCON1 register to set the output polarities.
5. Configure the ISM[4:0] bits of the CWGxISM register to select the data input source.
6. If a steering mode is selected, configure the STRx bits to select the desired output on the CWG outputs.
7. Configure the LSBD[1:0] and LSAC[1:0] bits of the CWGxASD0 register to select the auto-shutdown output override states (this is necessary even if not using auto-shutdown because start-up will be from a shutdown state).
8. If auto-restart is desired, set the REN bit of CWGxAS0.
9. If auto-shutdown is desired, configure the ASxE bits of the CWGxAS1 register to select the shutdown source.
10. Set the desired rising and falling dead-band times with the CWGxDBR and CWGxDBF registers.
11. Select the clock source in the CWGxCLKCON register.
12. Set the EN bit to enable the module.
13. Clear the TRIS bits that correspond to the CWG outputs to set them as outputs.

If auto-restart is to be used, set the REN bit and the SHUTDOWN bit will be cleared automatically. Otherwise, clear the SHUTDOWN bit in software to start the CWG.

FIGURE 26-14: CWG SHUTDOWN BLOCK DIAGRAM

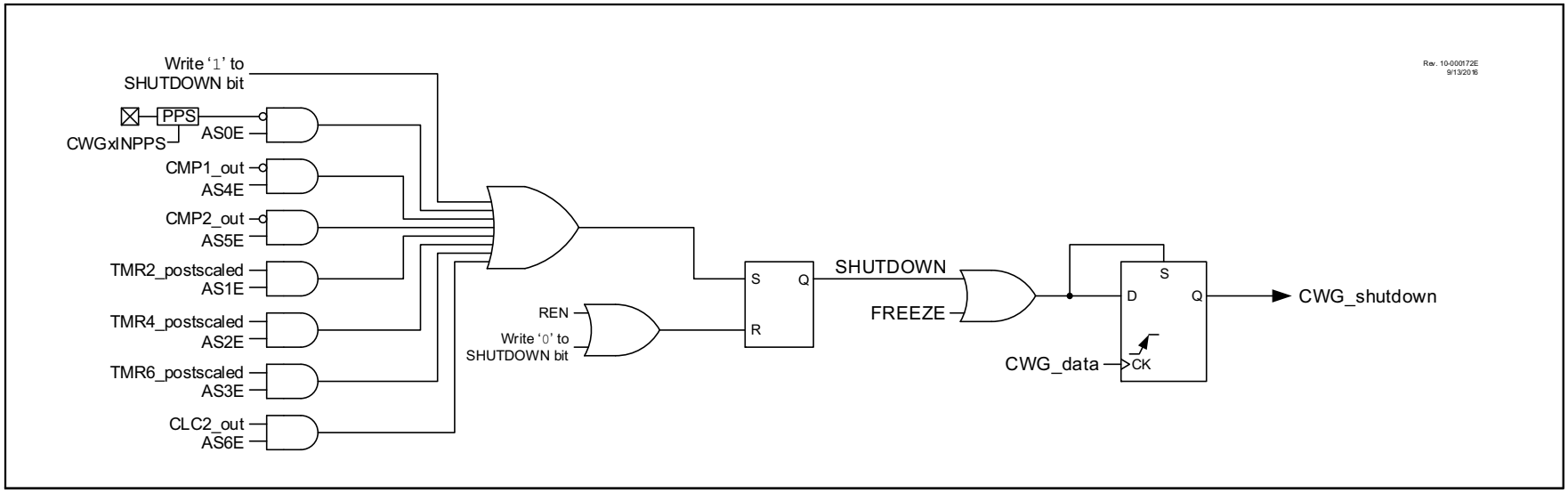
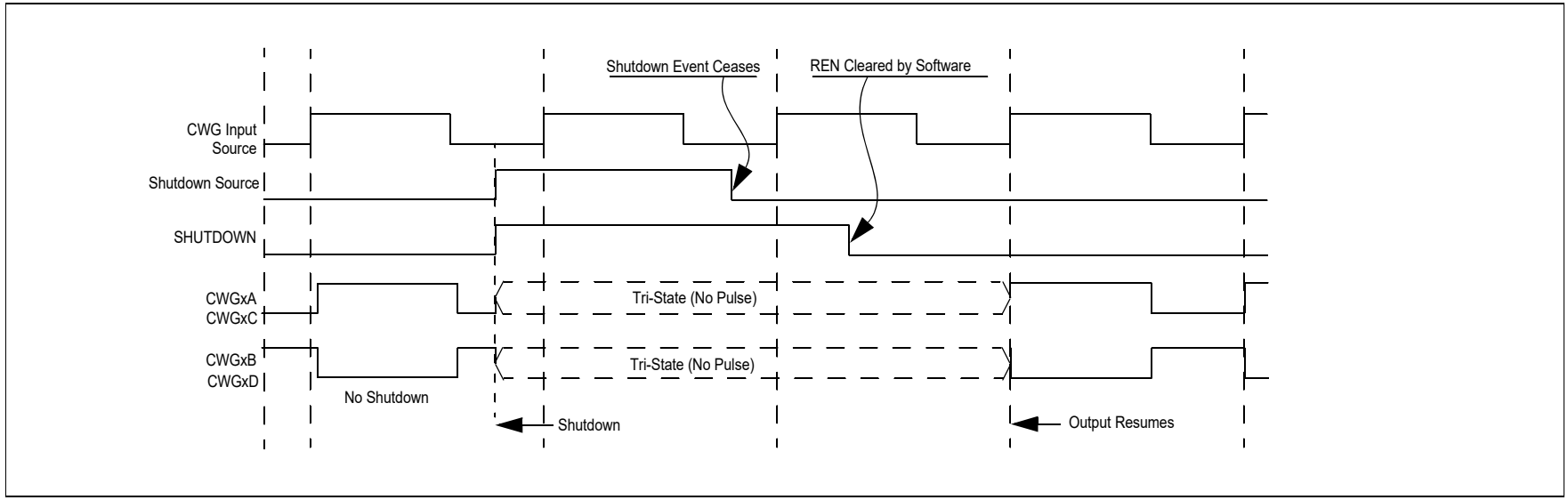
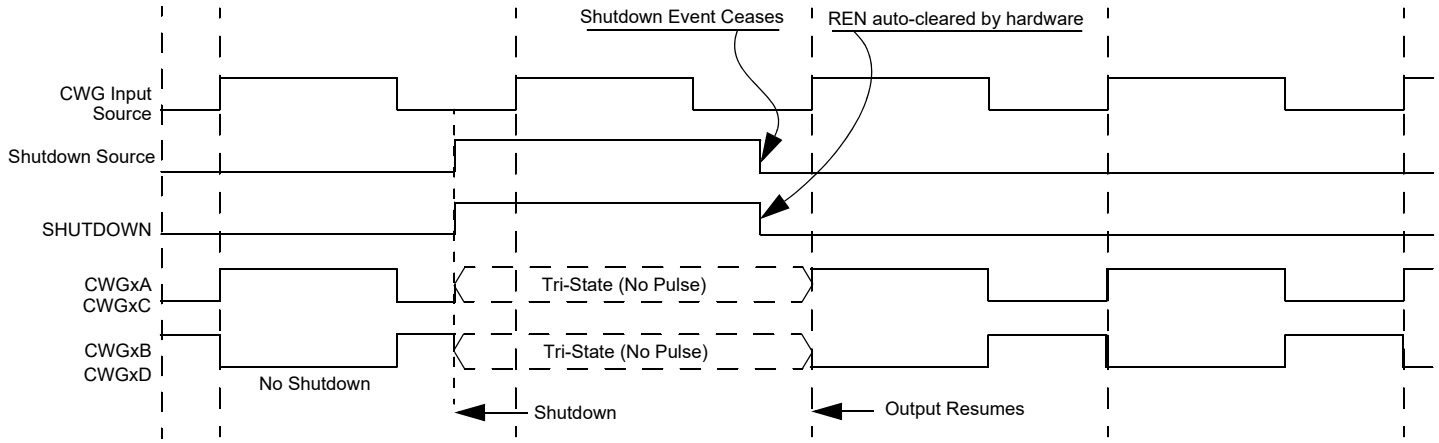


FIGURE 26-15: SHUTDOWN FUNCTIONALITY, AUTO-RESTART DISABLED (REN = 0, LSAC = 01, LSBD = 01)



© 2017-2020 Microchip Technology Inc.

FIGURE 26-16: SHUTDOWN FUNCTIONALITY, AUTO-RESTART ENABLED (REN = 1, LSAC = 01, LSBD = 01)



© 2017-2020 Microchip Technology Inc.

26.14 Register Definitions: CWG Control

Long bit name prefixes for the CWG peripheral is shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
CWG1	CWG1
CWG2	CWG2
CWG3	CWG3

REGISTER 26-1: CWGxCON0: CWG CONTROL REGISTER 0

R/W-0/0	R/W/HC-0/0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
EN	LD(1)	—	—	—	MODE[2:0]		
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	HC = Bit is cleared by hardware

bit 7 **EN:** CWGx Enable bit
1 = Module is enabled
0 = Module is disabled

bit 6 **LD:** CWGx Load Buffers bit(1)
1 = Dead-band count buffers to be loaded on CWG data rising edge, following first falling edge after this bit is set
0 = Buffers remain unchanged

bit 5-3 **Unimplemented**: Read as '0'

bit 2-0 **MODE[2:0]**: CWGx Mode bits
111 = Reserved
110 = Reserved
101 = CWG outputs operate in Push-Pull mode
100 = CWG outputs operate in Half-Bridge mode
011 = CWG outputs operate in Reverse Full-Bridge mode
010 = CWG outputs operate in Forward Full-Bridge mode
001 = CWG outputs operate in Synchronous Steering mode
000 = CWG outputs operate in Asynchronous Steering mode

Note 1: This bit can only be set after EN = 1; it cannot be set in the same cycle when EN is set.

REGISTER 26-2: CWGxCON1: CWG CONTROL REGISTER 1

U-0	U-0	R-x	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	IN	—	POLD	POLC	POLB	POLA
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 7-6 **Unimplemented:** Read as ‘0’

bit 5 **IN:** CWG Input Value bit (read-only)

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **POLD:** CWGxD Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 2 **POLC:** CWGxC Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 1 **POLB:** CWGxB Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 0 **POLA:** CWGxA Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

REGISTER 26-3: CWGxCLK: CWGx CLOCK INPUT SELECTION REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	CS
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 7-1 **Unimplemented:** Read as ‘0’

bit 0 **CS:** CWG Clock Source Selection bits

CS	CWG1	CWG2	CWG3
1	HFINTOSC (1)	HFINTOSC (1)	HFINTOSC (1)
0	FOSC	FOSC	FOSC

Note 1: HFINTOSC remains operating during Sleep.

REGISTER 26-4: CWGxISM: CWGx INPUT SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ISM[4:0]				
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 7-5 **Unimplemented** Read as ‘0’

bit 4-0 **ISM[4:0]:** CWG Data Input Selection Multiplexer Select bits

ISM[4:0]	CWG1	CWG2	CWG3
	Input Selection	Input Selection	Input Selection
11111-10011	Reserved	Reserved	Reserved
10010	CLC4_out	CLC4_out	CLC4_out
10001	CLC3_out	CLC3_out	CLC3_out
10000	CLC2_out	CLC2_out	CLC2_out
01111	CLC1_out	CLC1_out	CLC1_out
01110	DSM_out	DSM_out	DSM_out
01101	CMP2OUT	CMP2OUT	CMP2OUT
01100	CMP1OUT	CMP1OUT	CMP1OUT
01011	NCO1OUT	NCO1OUT	NCO1OUT
01010-01001	Reserved	Reserved	Reserved
01000	PWM8OUT	PWM8OUT	PWM8OUT
00111	PWM7OUT	PWM7OUT	PWM7OUT
00110	PWM6OUT	PWM6OUT	PWM6OUT
00101	PWM5OUT	PWM5OUT	PWM5OUT
00100	CCP4_out	CCP4_out	CCP4_out
00011	CCP3_out	CCP3_out	CCP3_out
00010	CCP2_out	CCP2_out	CCP2_out
00001	CCP1_out	CCP1_out	CCP1_out
00000	Pin selected by CWG1PPS	Pin selected by CWG2PPS	Pin selected by CWG3PPS

REGISTER 26-5: CWGxSTR(1): CWG STEERING CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
OVRD	OVRC	OVRB	OVRA	STRD(2)	STRC(2)	STRB(2)	STRA(2)
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 **OVRD:** Steering Data D bit

bit 6 **OVRC:** Steering Data C bit

bit 5 **OVRB:** Steering Data B bit

bit 4 **OVRA:** Steering Data A bit

bit 3 **STRD:** Steering Enable bit D(2)
`1` = CWGxD output has the CWG data input waveform with polarity control from POLD bit
`0` = CWGxD output is assigned to value of OVRD bit

bit 2 **STRC:** Steering Enable bit C(2)
`1` = CWGxC output has the CWG data input waveform with polarity control from POLC bit
`0` = CWGxC output is assigned to value of OVRC bit

bit 1 **STRB:** Steering Enable bit B(2)
`1` = CWGxB output has the CWG data input waveform with polarity control from POLB bit
0 = CWGxB output is assigned to value of OVRB bit

bit 0 **STRA:** Steering Enable bit A(2)
`1` = CWGxA output has the CWG data input waveform with polarity control from POLA bit
`0` = CWGxA output is assigned to value of OVRA bit

Note 1: The bits in this register apply only when MODE[2:0] = `00x` (Register 26-1, Steering modes).
2: This bit is double-buffered when MODE[2:0] = `001`.

REGISTER 26-6: CWGxAS0: CWG AUTO-SHUTDOWN CONTROL REGISTER 0

R/W/HS/HC-0/0	R/W-0/0	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	U-0	U-0
SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]		—	—
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	HS/HC = Bit is set/cleared by hardware
q = Value depends on condition		

bit 7 **SHUTDOWN:** Auto-Shutdown Event Status bit[1,2]
- 1 = An auto-shutdown state is in effect
- 0 = No auto-shutdown event has occurred

bit 6 **REN:** Auto-Restart Enable bit
- 1 = Auto-restart is enabled
- 0 = Auto-restart is disabled

bit 5-4 **LSBD[1:0]:** CWGxB and CWGxD Auto-Shutdown State Control bits
- 11 = A logic ‘1’ is placed on CWGxB/D when an auto-shutdown event occurs.
- 10 = A logic ‘0’ is placed on CWGxB/D when an auto-shutdown event occurs.
- 01 = Pin is tri-stated on CWGxB/D when an auto-shutdown event occurs.
- 00 = The inactive state of the pin, including polarity, is placed on CWGxB/D after the required dead-band interval when an auto-shutdown event occurs.

bit 3-2 **LSAC[1:0]:** CWGxA and CWGxC Auto-Shutdown State Control bits
- 11 = A logic ‘1’ is placed on CWGxA/C when an auto-shutdown event occurs.
- 10 = A logic ‘0’ is placed on CWGxA/C when an auto-shutdown event occurs.
- 01 = Pin is tri-stated on CWGxA/C when an auto-shutdown event occurs.
- 00 = The inactive state of the pin, including polarity, is placed on CWGxA/C after the required dead-band interval when an auto-shutdown event occurs.

bit 1-0 **Unimplemented:** Read as ‘0’

Note 1: This bit may be written while EN = 0 (Register 26-1), to place the outputs into the shutdown configuration.

2: The outputs will remain in auto-shutdown state until the next rising edge of the CWG data input after this bit is cleared.

REGISTER 26-7: CWGxAS1: CWG AUTO-SHUTDOWN CONTROL REGISTER 1

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
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 **Unimplemented** Read as ‘0’

bit 6 **AS6E:** CWG Auto-shutdown Source 6 Enable bit
1 = Auto-shutdown for Source 6 is enabled

CWG Module	CWG1	CWG2	CWG3
Auto-shutdown Source 6	CLC2 OUT	CLC3 OUT	CLC4 OUT

0 = Auto-shutdown for Source 6 is disabled

bit 5 **AS5E:** CWG Auto-shutdown Source 5 (CMP2 OUT) Enable bit
1 = Auto-shutdown for CMP2 OUT is enabled
0 = Auto-shutdown for CMP2 OUT is disabled

bit 4 **AS4E:** CWG Auto-shutdown Source 4 (CMP1 OUT) Enable bit
1 = Auto-shutdown for CMP1 OUT is enabled
0 = Auto-shutdown for CMP1 OUT is disabled

bit 3 **AS3E:** CWG Auto-shutdown Source 3 (TMR6_Postscaled) Enable bit
1 = Auto-shutdown for TMR6_Postscaled is enabled
0 = Auto-shutdown for TMR6_Postscaled is disabled

bit 2 **AS2E:** CWG Auto-shutdown Source 2 (TMR4_Postscaled) Enable bit
1 = Auto-shutdown for TMR4_Postscaled is enabled
0 = Auto-shutdown for TMR4_Postscaled is disabled

bit 1 **AS1E:** CWG Auto-shutdown Source 1 (TMR2_Postscaled) Enable bit
1 = Auto-shutdown for TMR2_Postscaled is enabled
0 = Auto-shutdown for TMR2_Postscaled is disabled

bit 0 **AS0E:** CWG Auto-shutdown Source 0 (Pin selected by CWGxPPS) Enable bit
1 = Auto-shutdown for CWGxPPS Pin is enabled
0 = Auto-shutdown for CWGxPPS Pin is disabled

REGISTER 26-8: CWGxDBR: CWG RISING DEAD-BAND COUNT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	DBR[5:0]					
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 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **DBR[5:0]:** CWG Rising Edge Triggered Dead-Band Count bits
11 1111 = 63-64 CWG clock periods
11 1110 = 62-63 CWG clock periods
.
.
.
00 0010 = 2-3 CWG clock periods
00 0001 = 1-2 CWG clock periods
00 0000 = 0 CWG clock periods. Dead-band generation is by-passed

REGISTER 26-9: CWGxDBF: CWG FALLING DEAD-BAND COUNT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	DBF[5:0]					
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 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **DBF[5:0]:** CWG Falling Edge Triggered Dead-Band Count bits
11 1111 = 63-64 CWG clock periods
11 1110 = 62-63 CWG clock periods
.
.
.
00 0010 = 2-3 CWG clock periods
00 0001 = 1-2 CWG clock periods
00 0000 = 0 CWG clock periods. Dead-band generation is by-passed.

TABLE 26-2: SUMMARY OF REGISTERS ASSOCIATED WITH CWG

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CWGxCON0	EN	LD	—	—	—	MODE[2:0]			425
CWGxCON1	—	—	IN	—	POLD	POLC	POLB	POLA	426
CWGxCLK	—	—	—	—	—	—	—	CS	427
CWGxISM	—	—	—	ISM[4:0]					428
CWGxSTR	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA	429
CWGxAS0	SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]		—	—	430
CWGxAS1	—	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E	431
CWGxDBR	—	—	DBR[5:0]						432
CWGxDBF	—	—	DBF[5:0]						432

Legend: – = unimplemented locations read as ‘0’. Shaded cells are not used by CWG.

27.0 CONFIGURABLE LOGIC CELL (CLC)

The Configurable Logic Cell (CLCx) module provides programmable logic that operates outside the speed limitations of software execution. The logic cell takes up the input signals and, through the use of configurable gates, reduces the inputs to four logic lines that drive one of eight selectable single-output logic functions.

Input sources are a combination of the following:

- I/O pins
- Internal clocks
- Peripherals
- Register bits

The output can be directed internally to peripherals and to an output pin.

There are four CLC modules available on this device - CLC1, CLC2, CLC3 and CLC4.

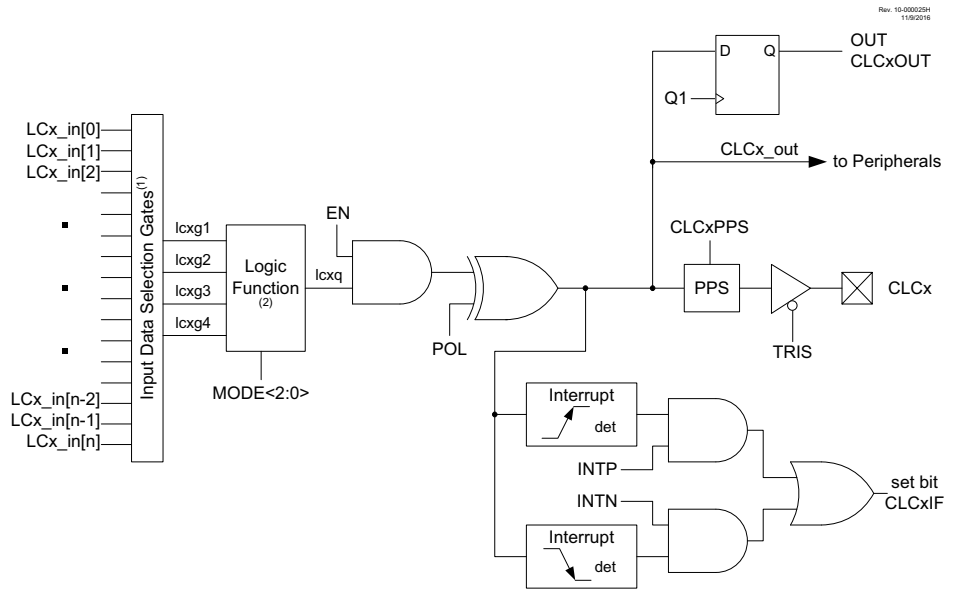
Note:	The CLC1, CLC2, CLC3 and CLC4 are four separate module instances of the same CLC module design. Throughout this section, the lower case ‘x’ in register names is a generic reference to the CLC number (which should be substituted with 1, 2, 3, or 4 during code development). For example, the control register is generically described in this chapter as CLCxCON, but the actual device registers are CLC1CON, CLC2CON, CLC3CON and CLC4CON.

Refer to Figure 27-1 for a simplified diagram showing signal flow through the CLCx.

Possible configurations include:

- Combinatorial Logic
  - AND
  - NAND
  - AND-OR
  - AND-OR-INVERT
  - OR-XOR
  - OR-XNOR
- Latches
  - S-R
  - Clocked D with Set and Reset
  - Transparent D with Set and Reset

FIGURE 27-1: CLCx SIMPLIFIED BLOCK DIAGRAM



Note 1: See Figure 27-2: Input Data Selection and Gating

2: See Figure 27-3: Programmable Logic Functions.

27.1 CLCx Setup

Programming the CLCx module is performed by configuring the four stages in the logic signal flow. The four stages are:

- Data selection
- Data gating
- Logic function selection
- Output polarity

Each stage is setup at run time by writing to the corresponding CLCx Special Function Registers. This has the added advantage of permitting logic reconfiguration on-the-fly during program execution.

27.1.1 DATA SELECTION

There are many signals available as inputs to the configurable logic. Four input multiplexers are used to select the inputs to pass on to the next stage.

Data selection is through four multiplexers as indicated on the left side of Figure 27-2. Data inputs in the figure are identified by a generic numbered input name.

Table 27-1 correlates the generic input name to the actual signal for each CLC module. The column labeled 'DyS[5:0] Value' indicates the MUX selection code for the selected data input. DyS is an abbreviation for the MUX select input codes: D1S[5:0] through D4S[5:0].

Data inputs are selected with CLCxSEL0 through CLCxSEL3 registers (Register 27-3 through Register 27-6).

Note: Data selections are undefined at power-up.

TABLE 27-1: CLCx DATA INPUT SELECTION

DyS[5:0] Value	CLCx Input Source
111111 [63]	Reserved
.	
.	
.	
110100 [52]	Reserved
110011 [51]	CWG3B_out
110010 [50]	CWG3A_out
110001 [49]	CWG2B_out
110000 [48]	CWG2A_out
101111 [47]	CWG1B_out
101110 [46]	CWG1A_out
101101 [45]	SS1
101100 [44]	SCK1
101011 [43]	SDO1
101010 [42]	Reserved
101001 [41]	UART2_tx_out
101000 [40]	UART1_tx_out
100111 [39]	CLC4_out
100110 [38]	CLC3_out
100101 [37]	CLC2_out
100100 [36]	CLC1_out
100011 [35]	DSM1_out
100010 [34]	IOC_flag
100001 [33]	ZCD_out
100000 [32]	CMP2_out
011111 [31]	CMP1_out
011110 [30]	NCO1_out
011101 [29]	Reserved
011100 [28]	Reserved
011011 [27]	PWM8_out
011010 [26]	PWM7_out
011001 [25]	PWM6_out
011000 [24]	PWM5_out
010111 [23]	CCP4_out
010110 [22]	CCP3_out
010101 [21]	CCP2_out
010100 [20]	CCP1 _out
010011 [19]	SMT1_out
010010 [18]	TMR6_out
010001 [17]	TMR5 _overflow
010000 [16]	TMR4 _out
001111 [15]	TMR3 _overflow

TABLE 27-1: CLCx DATA INPUT SELECTION (CONTINUED)

DyS[5:0] Value	CLCx Input Source
001110 [14]	TMR2 _out
001101 [13]	TMR1 _overflow
001100 [12]	TMR0 _overflow
001011 [11]	CLKR _out
001010 [10]	ADCRC
001001 [9]	SOSC
001000 [8]	MFINTOSC (32 kHz)
000111 [7]	MFINTOSC (500 kHz)
000110 [6]	LFINTOSC
000101 [5]	HFINTOSC
000100 [4]	FOSC
000011 [3]	CLCIN3PPS
000010 [2]	CLCIN2PPS
000001 [1]	CLCIN1PPS
000000 [0]	CLCIN0PPS

27.1.2 DATA GATING

Outputs from the input multiplexers are directed to the desired logic function input through the data gating stage. Each data gate can direct any combination of the four selected inputs.

Note: Data gating is undefined at power-up.

The gate stage is more than just signal direction. The gate can be configured to direct each input signal as inverted or non-inverted data. Directed signals are ANDed together in each gate. The output of each gate can be inverted before going on to the logic function stage.

The gating is in essence a 1-to-4 input AND/NAND/OR/NOR gate. When every input is inverted and the output is inverted, the gate is an OR of all enabled data inputs. When the inputs and output are not inverted, the gate is an AND or all enabled inputs.

Table 27-2 summarizes the basic logic that can be obtained in gate 1 by using the gate logic select bits. The table shows the logic of four input variables, but each gate can be configured to use less than four. If no inputs are selected, the output will be zero or one, depending on the gate output polarity bit.

TABLE 27-2: DATA GATING LOGIC

CLCxGLSy	GyPOL	Gate Logic
0x55	1	AND
0x55	0	NAND
0xAA	1	NOR
0xAA	0	OR
0x00	0	Logic 0
0x00	1	Logic 1

It is possible (but not recommended) to select both the true and negated values of an input. When this is done, the gate output is zero, regardless of the other inputs, but may emit logic glitches (transient-induced pulses). If the output of the channel must be zero or one, the recommended method is to set all gate bits to zero and use the gate polarity bit to set the desired level.

Data gating is configured with the logic gate select registers as follows:

- Gate 1: CLCxGLS0 (Register 27-7)
- Gate 2: CLCxGLS1 (Register 27-8)
- Gate 3: CLCxGLS2 (Register 27-9)
- Gate 4: CLCxGLS3 (Register 27-10)

Register number suffixes are different than the gate numbers because other variations of this module have multiple gate selections in the same register.

Data gating is indicated in the right side of Figure 27-2. Only one gate is shown in detail. The remaining three gates are configured identically with the exception that the data enables correspond to the enables for that gate.

27.1.3 LOGIC FUNCTION

There are eight available logic functions including:

- AND-OR
- OR-XOR
- AND
- S-R Latch
- D Flip-Flop with Set and Reset
- D Flip-Flop with Reset
- J-K Flip-Flop with Reset
- Transparent Latch with Set and Reset

Logic functions are shown in Figure 27-2. Each logic function has four inputs and one output. The four inputs are the four data gate outputs of the previous stage. The output is fed to the inversion stage and from there to other peripherals, an output pin, and back to the CLCx itself.

27.1.4 OUTPUT POLARITY

The last stage in the Configurable Logic Cell is the output polarity. Setting the POL bit of the CLCxPOL register inverts the output signal from the logic stage. Changing the polarity while the interrupts are enabled will cause an interrupt for the resulting output transition.

27.2 CLCx Interrupts

An interrupt will be generated upon a change in the output value of the CLCx when the appropriate interrupt enables are set. A rising edge detector and a falling edge detector are present in each CLC for this purpose.

The CLCxIF bit of the associated PIRx register will be set when either edge detector is triggered and its associated enable bit is set. The INTP enables rising edge interrupts and the INTN bit enables falling edge interrupts. Both are located in the CLCxCON register.

To fully enable the interrupt, set the following bits:

- CLCxIE bit of the respective PIE register
- INTP bit of the CLCxCON register (for a rising edge detection)
- INTN bit of the CLCxCON register (for a falling edge detection)
- GIE bits of the INTCON0 register

The CLCxIF bit of the respective PIR register, must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

27.3 Output Mirror Copies

Mirror copies of all CON output bits are contained in the CLCxDATA register. Reading this register reads the outputs of all CLCs simultaneously. This prevents any reading skew introduced by testing or reading the OUT bits in the individual CLCxCON registers.

27.4 Effects of a Reset

The CLCxCON register is cleared to zero as the result of a Reset. All other selection and gating values remain unchanged.

27.5 Operation During Sleep

The CLC module operates independently from the system clock and will continue to run during Sleep, provided that the input sources selected remain active.

The HFINTOSC remains active during Sleep when the CLC module is enabled and the HFINTOSC is selected as an input source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and as a CLC input source, when the CLC is enabled, the CPU will go idle during Sleep, but the CLC will continue to operate and the HFINTOSC will remain active.

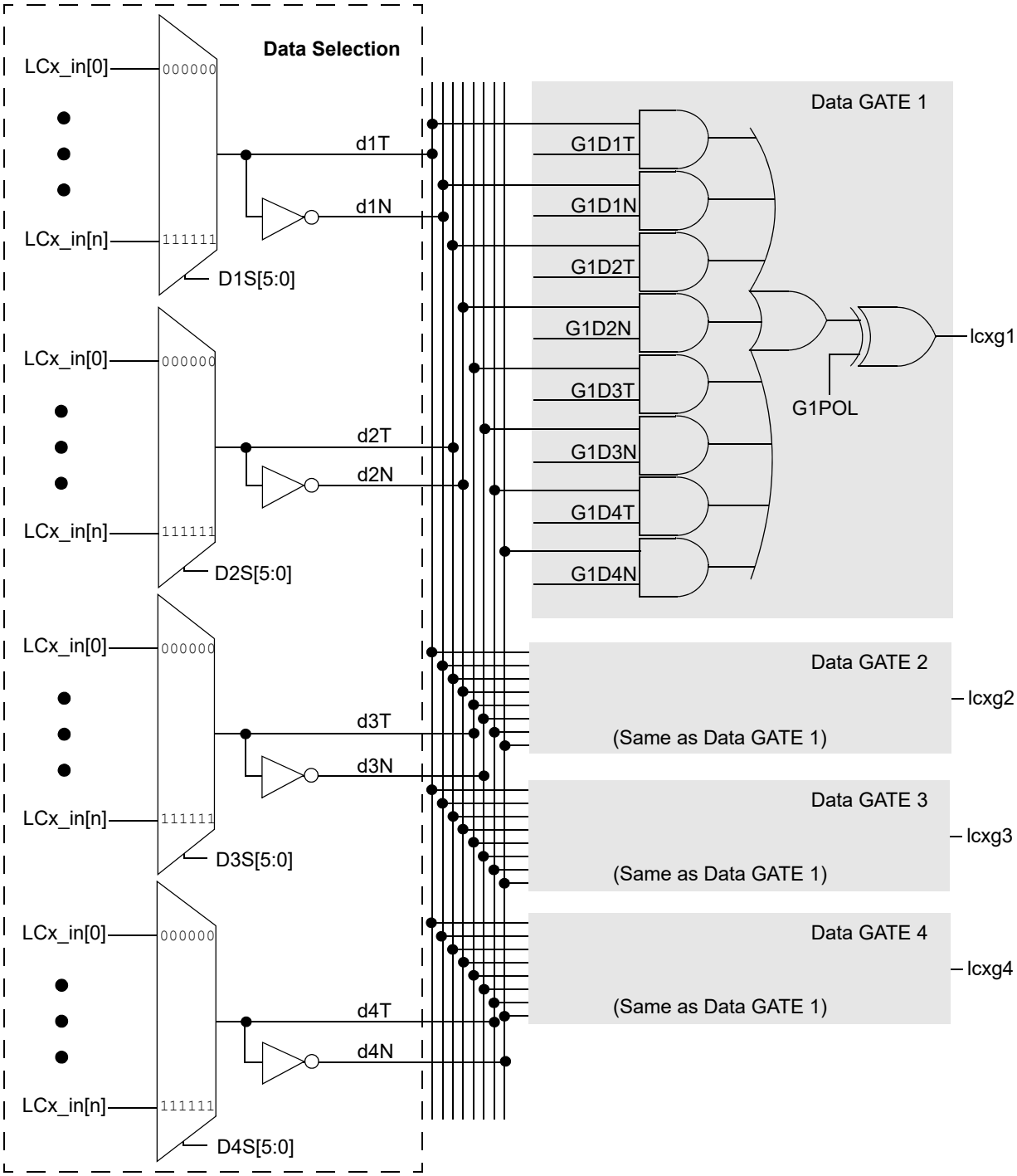
This will have a direct effect on the Sleep mode current.

27.6 CLCx Setup Steps

The following steps should be followed when setting up the CLCx:

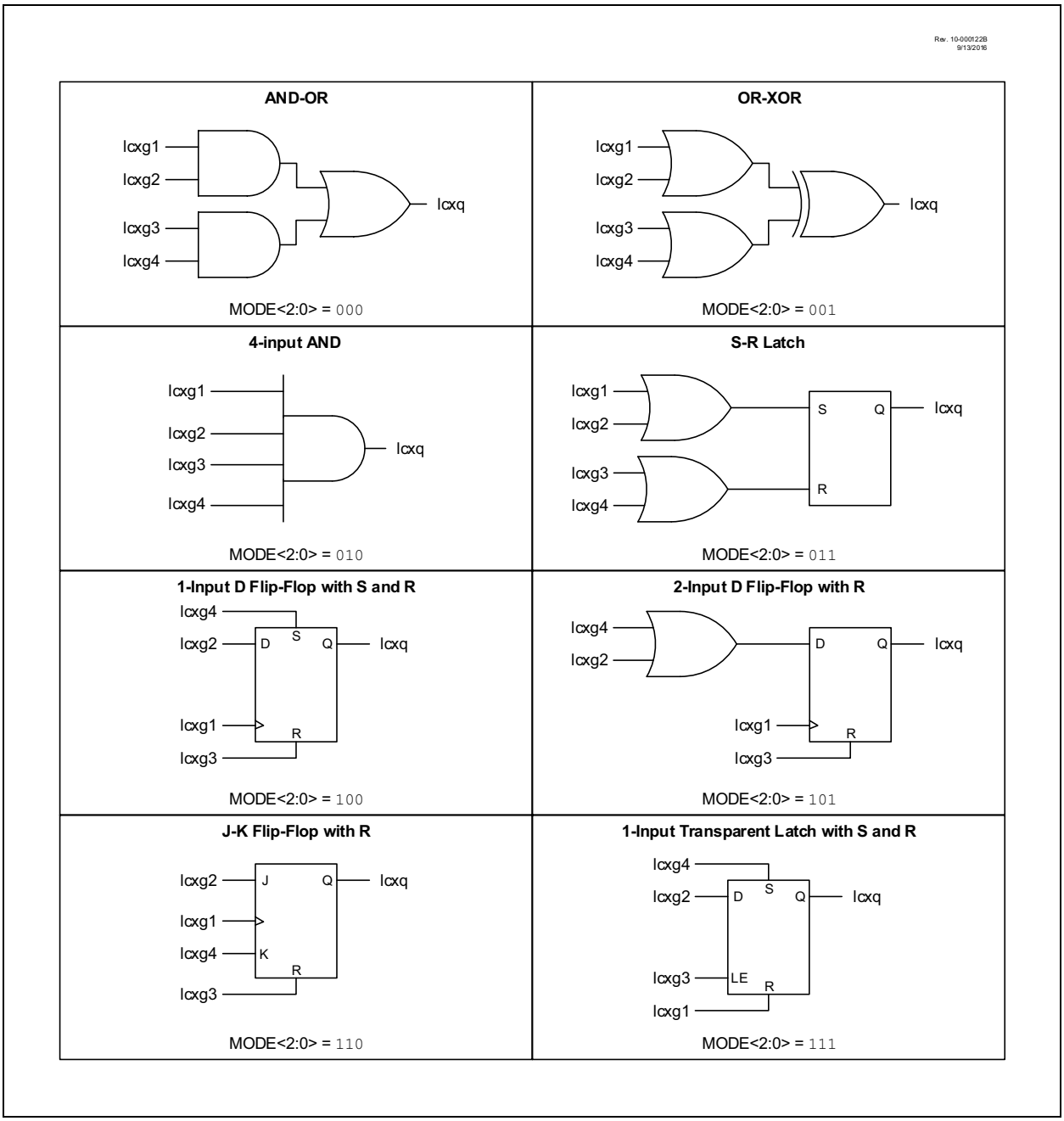
- Disable CLCx by clearing the EN bit.
- Select desired inputs using CLCxSEL0 through CLCxSEL3 registers (See Table 27-1).
- Clear any associated ANSEL bits.
- Set all TRIS bits associated with inputs.
- Clear all TRIS bits associated with outputs.
- Enable the chosen inputs through the four gates using CLCxGLS0, CLCxGLS1, CLCxGLS2, and CLCxGLS3 registers.
- Select the gate output polarities with the GyPOL bits of the CLCxPOL register.
- Select the desired logic function with the MODE[2:0] bits of the CLCxCON register.
- Select the desired polarity of the logic output with the POL bit of the CLCxPOL register. (This step may be combined with the previous gate output polarity step).
- If driving a device pin, set the desired pin PPS control register and also clear the TRIS bit corresponding to that output.
- If interrupts are desired, configure the following bits:
  - Set the INTP bit in the CLCxCON register for rising event.
  - Set the INTN bit in the CLCxCON register for falling event.
  - Set the CLCxIE bit of the respective PIE register.
  - Set the GIE bits of the INTCON0 register.
- Enable the CLCx by setting the EN bit of the CLCxCON register.

FIGURE 27-2: INPUT DATA SELECTION AND GATING



Note: All controls are undefined at power-up.

FIGURE 27-3: PROGRAMMABLE LOGIC FUNCTIONS



27.7 Register Definitions: CLC Control

REGISTER 27-1: CLCxCON: CONFIGURABLE LOGIC CELL CONTROL REGISTER

R/W-0/0	U-0	R-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	OUT	INTP	INTN	MODE[2:0]		
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 **EN:** Configurable Logic Cell Enable bit
1 = Configurable logic cell is enabled and mixing input signals
0 = Configurable logic cell is disabled and has logic zero output

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **OUT:** Configurable Logic Cell Data Output bit
Read-only: logic cell output data, after LCPOL; sampled from CLCxOUT

bit 4 **INTP:** Configurable Logic Cell Positive Edge Going Interrupt Enable bit
1 = CLCxIF will be set when a rising edge occurs on CLCxOUT
0 = CLCxIF will not be set

bit 3 **INTN:** Configurable Logic Cell Negative Edge Going Interrupt Enable bit
1 = CLCxIF will be set when a falling edge occurs on CLCxOUT
0 = CLCxIF will not be set

bit 2-0 **MODE[2:0]:** Configurable Logic Cell Functional Mode bits
111 = Cell is 1-input transparent latch with S and R
110 = Cell is J-K flip-flop with R
101 = Cell is 2-input D flip-flop with R
100 = Cell is 1-input D flip-flop with S and R
011 = Cell is S-R latch
010 = Cell is 4-input AND
001 = Cell is OR-XOR
000 = Cell is AND-OR

REGISTER 27-2: CLCxPOL: SIGNAL POLARITY CONTROL REGISTER

R/W-0/0	U-0	U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
POL	—	—	—	G4POL	G3POL	G2POL	G1POL
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 **POL:** CLCxOUT Output Polarity Control bit
1 = The output of the logic cell is inverted
0 = The output of the logic cell is not inverted

bit 6-4 **Unimplemented:** Read as ‘0’

bit 3 **G4POL:** Gate 3 Output Polarity Control bit
1 = The output of gate 3 is inverted when applied to the logic cell
0 = The output of gate 3 is not inverted

bit 2 **G3POL:** Gate 2 Output Polarity Control bit
1 = The output of gate 2 is inverted when applied to the logic cell
0 = The output of gate 2 is not inverted

bit 1 **G2POL:** Gate 1 Output Polarity Control bit
1 = The output of gate 1 is inverted when applied to the logic cell
0 = The output of gate 1 is not inverted

bit 0 **G1POL:** Gate 0 Output Polarity Control bit
1 = The output of gate 0 is inverted when applied to the logic cell
0 = The output of gate 0 is not inverted

© 2017-2020 Microchip Technology Inc.

REGISTER 27-3: CLCxSEL0: GENERIC CLCx DATA 0 SELECT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	D1S[5:0]					
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 **Unimplemented:** Read as ‘0’

bit 5-0 **D1S[5:0]:** CLCx Data1 Input Selection bits
See Table 27-1.

REGISTER 27-4: CLCxSEL1: GENERIC CLCx DATA 1 SELECT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	D2S[5:0]					
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 **Unimplemented:** Read as ‘0’

bit 5-0 **D2S[5:0]:** CLCx Data 2 Input Selection bits
See Table 27-1.

REGISTER 27-5: CLCxSEL2: GENERIC CLCx DATA 2 SELECT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	D3S[5:0]					
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 **Unimplemented:** Read as ‘0’

bit 5-0 **D3S[5:0]:** CLCx Data 3 Input Selection bits
See Table 27-1.

REGISTER 27-6: CLCxSEL3: GENERIC CLCx DATA 3 SELECT REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	D4S[5:0]					
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 **Unimplemented:** Read as ‘0’

bit 5-0 **D4S[5:0]: CLCx Data 4 Input Selection bits**
See Table 27-1.

REGISTER 27-7: CLCxGLS0: GATE 0 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
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 **G1D4T:** Gate 0 Data 4 True (non-inverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 0
0 = CLCIN3 (true) is not gated into CLCx Gate 0

bit 6 **G1D4N:** Gate 0 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 0
0 = CLCIN3 (inverted) is not gated into CLCx Gate 0

bit 5 **G1D3T:** Gate 0 Data 3 True (non-inverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 0
0 = CLCIN2 (true) is not gated into CLCx Gate 0

bit 4 **G1D3N:** Gate 0 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 0
0 = CLCIN2 (inverted) is not gated into CLCx Gate 0

bit 3 **G1D2T:** Gate 0 Data 2 True (non-inverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 0
0 = CLCIN1 (true) is not gated into l CLCx Gate 0

bit 2 **G1D2N:** Gate 0 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 0
0 = CLCIN1 (inverted) is not gated into CLCx Gate 0

bit 1 **G1D1T:** Gate 0 Data 1 True (non-inverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 0
0 = CLCIN0 (true) is not gated into CLCx Gate 0

bit 0 **G1D1N:** Gate 0 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 0
0 = CLCIN0 (inverted) is not gated into CLCx Gate 0

© 2017-2020 Microchip Technology Inc.

REGISTER 27-8: CLCxGLS1: GATE 1 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
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 **G2D4T:** Gate 1 Data 4 True (noninverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 1
0 = CLCIN3 (true) is not gated into CLCx Gate 1

bit 6 **G2D4N:** Gate 1 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 1
0 = CLCIN3 (inverted) is not gated into CLCx Gate 1

bit 5 **G2D3T:** Gate 1 Data 3 True (noninverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 1
0 = CLCIN2 (true) is not gated into CLCx Gate 1

bit 4 **G2D3N:** Gate 1 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 1
0 = CLCIN2 (inverted) is not gated into CLCx Gate 1

bit 3 **G2D2T:** Gate 1 Data 2 True (noninverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 1
0 = CLCIN1 (true) is not gated into CLCx Gate 1

bit 2 **G2D2N:** Gate 1 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 1
0 = CLCIN1 (inverted) is not gated into CLCx Gate 1

bit 1 **G2D1T:** Gate 1 Data 1 True (noninverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 1
0 = CLCIN0 (true) is not gated into CLCx Gate1

bit 0 **G2D1N:** Gate 1 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 1
0 = CLCIN0 (inverted) is not gated into CLCx Gate 1

© 2017-2020 Microchip Technology Inc.

REGISTER 27-9: CLCxGLS2: GATE 2 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
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 **G3D4T:** Gate 2 Data 4 True (noninverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 2
0 = CLCIN3 (true) is not gated into CLCx Gate 2

bit 6 **G3D4N:** Gate 2 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 2
0 = CLCIN3 (inverted) is not gated into CLCx Gate 2

bit 5 **G3D3T:** Gate 2 Data 3 True (noninverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 2
0 = CLCIN2 (true) is not gated into CLCx Gate 2

bit 4 **G3D3N:** Gate 2 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 2
0 = CLCIN2 (inverted) is not gated into CLCx Gate 2

bit 3 **G3D2T:** Gate 2 Data 2 True (noninverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 2
0 = CLCIN1 (true) is not gated into CLCx Gate 2

bit 2 **G3D2N:** Gate 2 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 2
0 = CLCIN1 (inverted) is not gated into CLCx Gate 2

bit 1 **G3D1T:** Gate 2 Data 1 True (noninverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 2
0 = CLCIN0 (true) is not gated into CLCx Gate 2

bit 0 **G3D1N:** Gate 2 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 2
0 = CLCIN0 (inverted) is not gated into CLCx Gate 2

© 2017-2020 Microchip Technology Inc.

REGISTER 27-10: CLCxGLS3: GATE 3 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
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 **G4D4T:** Gate 3 Data 4 True (non-inverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 3
0 = CLCIN3 (true) is not gated into CLCx Gate 3

bit 6 **G4D4N:** Gate 3 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 3
0 = CLCIN3 (inverted) is not gated into CLCx Gate 3

bit 5 **G4D3T:** Gate 3 Data 3 True (non-inverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 3
0 = CLCIN2 (true) is not gated into CLCx Gate 3

bit 4 **G4D3N:** Gate 3 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 3
0 = CLCIN2 (inverted) is not gated into CLCx Gate 3

bit 3 **G4D2T:** Gate 3 Data 2 True (non-inverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 3
0 = CLCIN1 (true) is not gated into CLCx Gate 3

bit 2 **G4D2N:** Gate 3 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 3
0 = CLCIN1 (inverted) is not gated into CLCx Gate 3

bit 1 **G4D1T:** Gate 4 Data 1 True (non-inverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 3
0 = CLCIN0 (true) is not gated into CLCx Gate 3

bit 0 **G4D1N:** Gate 3 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 3
0 = CLCIN0 (inverted) is not gated into CLCx Gate 3

© 2017-2020 Microchip Technology Inc.

REGISTER 27-11: CLCDATA: CLC DATA OUTPUT

U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	CLC4OUT	CLC3OUT	CLC2OUT	CLC1OUT
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 7-4 **Unimplemented:** Read as ‘0’
bit 3 **CLC4OUT:** Mirror copy of OUT bit of CLC4CON register
bit 2 **CLC3OUT:** Mirror copy of OUT bit of CLC3CON register
bit 1 **CLC2OUT:** Mirror copy of OUT bit of CLC2CON register
bit 0 **CLC1OUT:** Mirror copy of OUT bit of CLC1CON register

TABLE 27-3: SUMMARY OF REGISTERS ASSOCIATED WITH CLCx

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CLCxCON	EN	—	OUT	INTP	INTN	MODE[2:0]			441
CLCxPOL	POL	—	—	—	G4POL	G3POL	G2POL	G1POL	442
CLCxSEL0	—	—	D1S[5:0]						443
CLCxSEL1	—	—	D2S[5:0]						443
CLCxSEL2	—	—	D3S[5:0]						443
CLCxSEL3	—	—	D4S[5:0]						443
CLCxGLS0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N	444
CLCxGLS1	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N	445
CLCxGLS2	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N	446
CLCxGLS3	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N	447
CLCDATA	—	—	—	—	CLC4OUT	CLC3OUT	CLC2OUT	CLC1OUT	448

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

28.0 NUMERICALLY CONTROLLED OSCILLATOR (NCO) MODULE

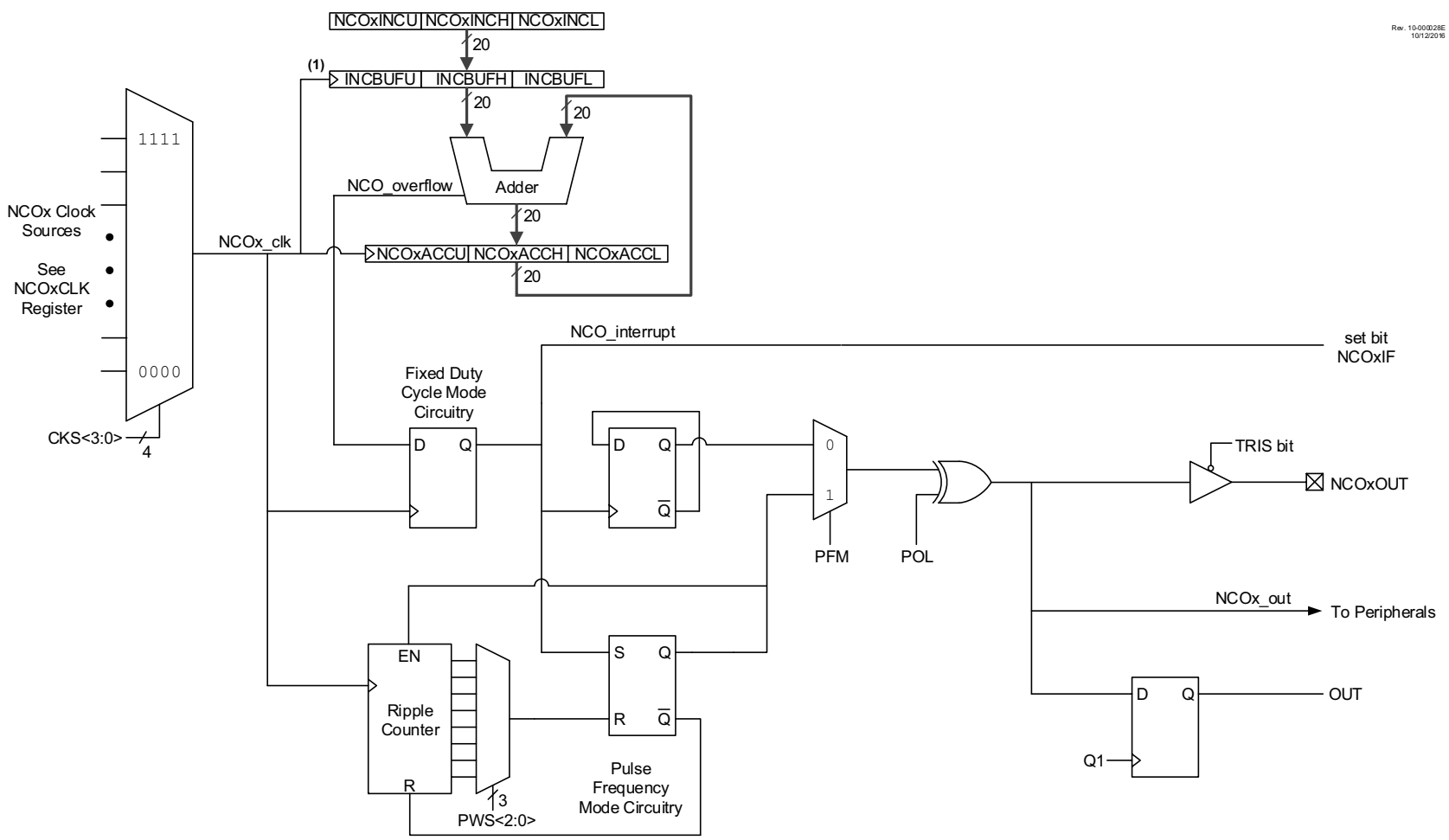
The Numerically Controlled Oscillator (NCO) module is a timer that uses overflow from the addition of an increment value to divide the input frequency. The advantage of the addition method over simple counter driven timer is that the output frequency resolution does not vary with the divider value. The NCO is most useful for applications that require frequency accuracy and fine resolution at a fixed duty cycle.

Features of the NCO include:

- 20-bit Increment Function
- Fixed Duty Cycle mode (FDC) mode
- Pulse Frequency (PF) mode
- Output Pulse-Width Control
- Multiple Clock Input Sources
- Output Polarity Control
- Interrupt Capability

Figure 28-1 is a simplified block diagram of the NCO module.

FIGURE 28-1: DIRECT DIGITAL SYNTHESIS MODULE SIMPLIFIED BLOCK DIAGRAM



Note 1: The increment registers are double-buffered to allow for value changes to be made without first disabling the NCO module. The full increment value is loaded into the buffer registers on the second rising edge of the NCOx_clk signal that occurs immediately after a write to NCOxINCL register. The buffers are not user-accessible and are shown here for reference.

© 2017-2020 Microchip Technology Inc.

28.1 NCO Operation

The NCO operates by repeatedly adding a fixed value to an accumulator. Additions occur at the input clock rate. The accumulator will overflow with a carry periodically, which is the raw NCO output (NCO_overflow). This effectively reduces the input clock by the ratio of the addition value to the maximum accumulator value. See Equation 28-1.

The NCO output can be further modified by stretching the pulse or toggling a flip-flop. The modified NCO output is then distributed internally to other peripherals and can be optionally output to a pin. The accumulator overflow also generates an interrupt (NCO_overflow).

The NCO period changes in discrete steps to create an average frequency. This output depends on the ability of the receiving circuit (i.e., CWG or external resonant converter circuitry) to average the NCO output to reduce uncertainty.

EQUATION 28-1: NCO OVERFLOW FREQUENCY

$$F_{OVERFLOW} = \frac{NCO\ Clock\ Frequency \times Increment\ Value}{2^{20}}$$

28.1.1 NCO CLOCK SOURCES

Clock sources available to the NCO include:

- FOSC
- HFINTOSC
- LFINTOSC
- MFINTOSC/4 (32 kHz)
- MFINTOSC (500 kHz)
- CLC1/2/3/4_out
- CLKREF
- SOSC

The NCO clock source is selected by configuring the N1CKS[2:0] bits in the NCO1CLK register.

28.1.2 ACCUMULATOR

The accumulator is a 20-bit register. Read and write access to the accumulator is available through three registers:

- NCO1ACCL
- NCO1ACCH
- NCO1ACCU

28.1.3 ADDER

The NCO Adder is a full adder, which operates independently from the source clock. The addition of the previous result and the increment value replaces the accumulator value on the rising edge of each input clock.

28.1.4 INCREMENT REGISTERS

The increment value is stored in three registers making up a 20-bit incrementer. In order of LSB to MSB they are:

- NCO1INCL
- NCO1INCH
- NCO1INCU

When the NCO module is enabled, the NCO1INCU and NCO1INCH registers should be written first, then the NCO1INCL register. Writing to the NCO1INCL register initiates the increment buffer registers to be loaded simultaneously on the second rising edge of the NCO_clk signal.

The registers are readable and writable. The increment registers are double-buffered to allow value changes to be made without first disabling the NCO module.

When the NCO module is disabled, the increment buffers are loaded immediately after a write to the increment registers.

Note: The increment buffer registers are not user-accessible.

28.2 FIXED DUTY CYCLE MODE

In Fixed Duty Cycle (FDC) mode, every time the accumulator overflows (NCO_overflow), the output is toggled. This provides a 50% duty cycle, provided that the increment value remains constant. For more information, see Figure 28-2.

28.3 PULSE FREQUENCY MODE

In Pulse Frequency (PF) mode, every time the Accumulator overflows, the output becomes active for one or more clock periods. Once the clock period expires, the output returns to an inactive state. This provides a pulsed output. The output becomes active on the rising clock edge immediately following the overflow event. For more information, see Figure 28-2.

The value of the active and inactive states depends on the polarity bit, POL in the NCO1CON register.

The PF mode is selected by setting the PFM bit in the NCO1CON register.

28.3.1 OUTPUT PULSE-WIDTH CONTROL

When operating in PF mode, the active state of the output can vary in width by multiple clock periods. Various pulse widths are selected with the PWS[2:0] bits in the NCO1CLK register.

When the selected pulse width is greater than the Accumulator overflow time frame, then DDS operation is undefined.

28.4 OUTPUT POLARITY CONTROL

The last stage in the NCO module is the output polarity. The POL bit in the NCO1CON register selects the output polarity. Changing the polarity while the interrupts are enabled will cause an interrupt for the resulting output transition. The NCO output signal is available to most of the other peripherals available on the device.

28.5 Interrupts

When the accumulator overflows (NCO_overflow), the NCO Interrupt Flag bit, NCO1IF, of the PIR4 register is set. To enable the interrupt event (NCO_interrupt), the following bits must be set:

- EN bit of the NCO1CON register
- NCO1IE bit of the PIE4 register
- GIE/GIEH bit of the INTCON0 register

The interrupt must be cleared by software by clearing the NCO1IF bit in the Interrupt Service Routine.

28.6 Effects of a Reset

All of the NCO registers are cleared to zero as the result of a Reset.

28.7 Operation in Sleep

The NCO module operates independently from the system clock and will continue to run during Sleep, provided that the clock source selected remains active.

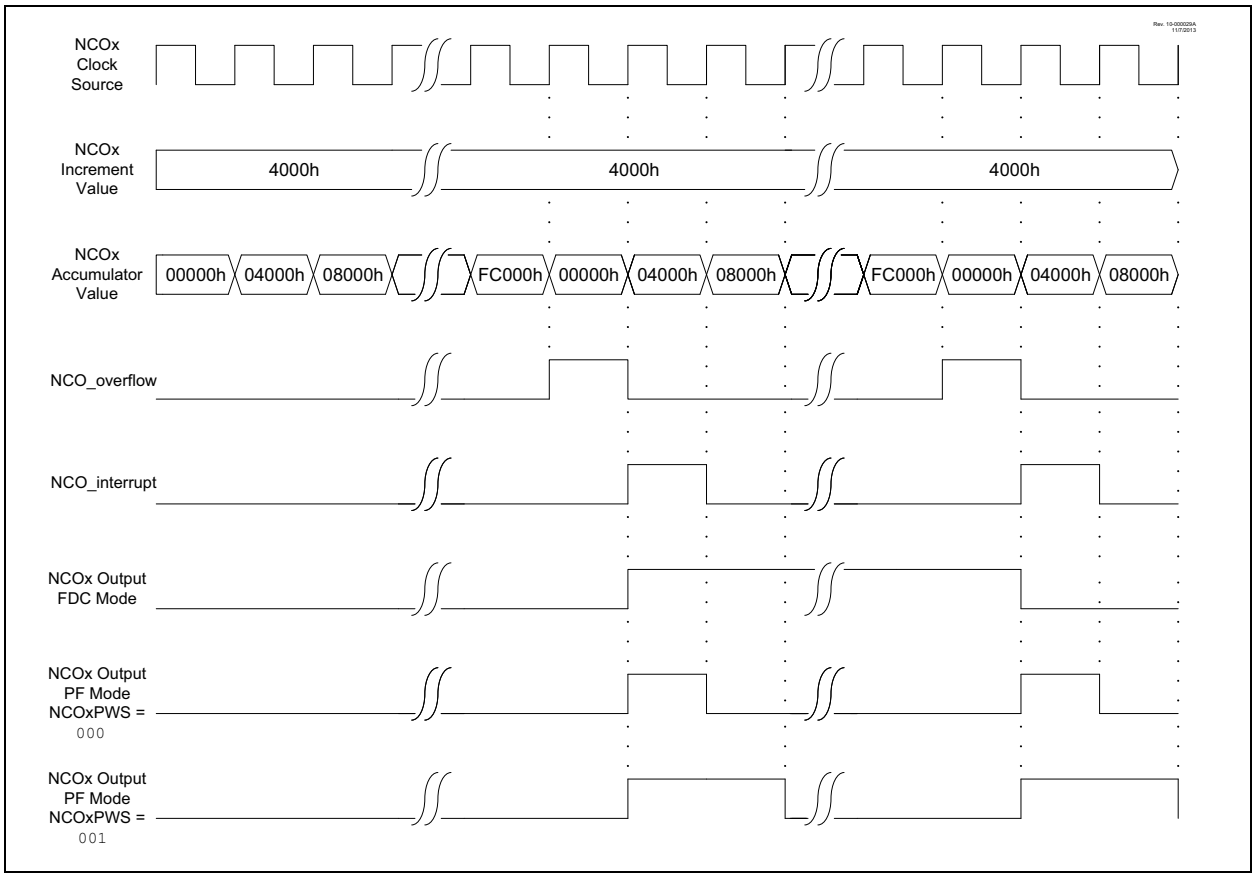
The HFINTOSC remains active during Sleep when the NCO module is enabled and the HFINTOSC is selected as the clock source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and the NCO clock source, when the NCO is enabled, the CPU will go idle during Sleep, but the NCO will continue to operate and the HFINTOSC will remain active.

This will have a direct effect on the Sleep mode current.

© 2017-2020 Microchip Technology Inc.

FIGURE 28-2: FDC OUTPUT MODE OPERATION DIAGRAM



28.8 NCO Control Registers

REGISTER 28-1: NCO1CON: NCO CONTROL REGISTER

R/W-0/0	U-0	R-0/0	R/W-0/0	U-0	U-0	U-0	R/W-0/0
EN	—	OUT	POL	—	—	—	PFM
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 **EN:** NCO1 Enable bit
1 = NCO1 module is enabled
0 = NCO1 module is disabled

bit 6 **Unimplemented**: Read as ‘0’

bit 5 **OUT:** NCO1 Output bit
Displays the current output value of the NCO1 module.

bit 4 **POL:** NCO1 Polarity
1 = NCO1 output signal is inverted
0 = NCO1 output signal is not inverted

bit 3-1 **Unimplemented**: Read as ‘0’

bit 0 **PFM:** NCO1 Pulse Frequency Mode bit
1 = NCO1 operates in Pulse Frequency mode
0 = NCO1 operates in Fixed Duty Cycle mode, divide by 2

REGISTER 28-2: NCO1CLK: NCO1 INPUT CLOCK CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
PWS[2:0](1,2)			—	CKS[3:0]			
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 7-5 **PWS[2:0]:** NCO1 Output Pulse Width Select bits(1,2)
- 111 = NCO1 output is active for 128 input clock periods
- 110 = NCO1 output is active for 64 input clock periods
- 101 = NCO1 output is active for 32 input clock periods
- 100 = NCO1 output is active for 16 input clock periods
- 011 = NCO1 output is active for 8 input clock periods
- 010 = NCO1 output is active for 4 input clock periods
- 001 = NCO1 output is active for 2 input clock periods
- 000 = NCO1 output is active for 1 input clock period

bit 4 **Unimplemented:** Read as ‘0’

bit 3-0 **CKS[3:0]:** NCO1 Clock Source Select bits
- 1111 = Reserved
- •
- •
- •
- 1011 = Reserved
- 1010 = CLC4_out
- 1001 = CLC3_out
- 1000 = CLC2_out
- 0111 = CLC1_out
- 0110 = CLKREF_out
- 0101 = SOSC
- 0100 = MFINTOSC/4 (32 kHz)
- 0011 = MFINTOSC (500 kHz)
- 0010 = LFINTOSC
- 0001 = HFINTOSC
- 0000 = FOSC

Note 1: N1PWS applies only when operating in Pulse Frequency mode.
2: If NCO1 pulse width is greater than NCO1 overflow period, operation is undefined.

REGISTER 28-3: NCO1ACCL: NCO1 ACCUMULATOR REGISTER – LOW BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ACC[7:0]							
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 7-0 **ACC[7:0]:** NCO1 Accumulator, Low Byte

REGISTER 28-4: NCO1ACCH: NCO1 ACCUMULATOR REGISTER – HIGH BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ACC[15:8]							
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 7-0 **ACC[15:8]:** NCO1 Accumulator, High Byte

REGISTER 28-5: NCO1ACCU: NCO1 ACCUMULATOR REGISTER – UPPER BYTE(1)

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	ACC[19:16]			
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 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **ACC[19:16]:** NCO1 Accumulator, Upper Byte

Note 1: The accumulator spans registers NCO1ACCU:NCO1ACCH: NCO1ACCL. The 24 bits are reserved but not all are used.This register updates in real time, asynchronously to the CPU; there is no provision to guarantee atomic access to this 24-bit space using an 8-bit bus. Writing to this register while the module is operating will produce undefined results.

REGISTER 28-6: NCO1INCL: NCO1 INCREMENT REGISTER – LOW BYTE(1,2)

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-1/1
INC[7:0]							
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 7-0 **INC[7:0]:** NCO1 Increment, Low Byte

Note 1: The logical increment spans NCO1INCU:NCO1INCH:NCO1INCL.

2: NCO1INC is double-buffered as INCBUF; INCBUF is updated on the next falling edge of NCOCLK after writing to NCO1INCL; NCO1INCU and NCO1INCH should be written prior to writing NCO1INCL.

REGISTER 28-7: NCO1INCH: NCO1 INCREMENT REGISTER – HIGH BYTE(1)

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
INC[15:8]							
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 7-0 **INC[15:8]:** NCO1 Increment, High Byte

Note 1: The logical increment spans NCO1INCU:NCO1INCH:NCO1INCL.

REGISTER 28-8: NCO1INCU: NCO1 INCREMENT REGISTER – UPPER BYTE(1)

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	INC[19:16]			
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 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **INC[19:16]:** NCO1 Increment, Upper Byte

Note 1: The logical increment spans NCO1INCU:NCO1INCH:NCO1INCL.

TABLE 28-1: SUMMARY OF REGISTERS ASSOCIATED WITH NCO

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
NCO1CON	N1EN	—	N1OUT	N1POL	—	—	—	N1PFM	454
NCO1CLK	N1PWS[2:0]			—	—	N1CKS[2:0]			455
NCO1ACCL	NCO1ACC[7:0]								456
NCO1ACCH	NCO1ACC[15:8]								456
NCO1ACCU	—	—	—	—	NCO1ACC[19:16]				457
NCO1INCL	NCO1INC[7:0]								457
NCO1INCH	NCO1INC[15:8]								457
NCO1INCU	—	—	—	—	NCO1INC[19:16]				458

Legend: — = unimplemented read as ‘0’. Shaded cells are not used for NCO module.

29.0 ZERO-CROSS DETECTION (ZCD) MODULE

The ZCD module detects when an A/C signal crosses through the ground potential. The actual zero-crossing threshold is the zero-crossing reference voltage, VCPINV, which is typically 0.75V above ground.

The connection to the signal to be detected is through a series current-limiting resistor. The module applies a current source or sink to the ZCD pin to maintain a constant voltage on the pin, thereby preventing the pin voltage from forward biasing the ESD protection diodes. When the applied voltage is greater than the reference voltage, the module sinks current. When the applied voltage is less than the reference voltage, the module sources current. The current source and sink action keeps the pin voltage constant over the full range of the applied voltage. The ZCD module is shown in the simplified block diagram Figure 29-2.

The ZCD module is useful when monitoring an A/C waveform for, but not limited to, the following purposes:

- A/C period measurement
- Accurate long term time measurement
- Dimmer phase delayed drive
- Low EMI cycle switching

29.1 External Resistor Selection

The ZCD module requires a current-limiting resistor in series with the external voltage source. The impedance and rating of this resistor depends on the external source peak voltage. Select a resistor value that will drop all of the peak voltage when the current through the resistor is nominally 300 µA. Refer to Equation 29-1 and Figure 29-1. Make sure that the ZCD I/O pin internal weak pull-up is disabled so it does not interfere with the current source and sink.

EQUATION 29-1: EXTERNAL RESISTOR

$$R_{SERIES} = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

FIGURE 29-1: EXTERNAL VOLTAGE

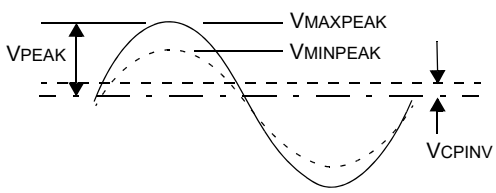
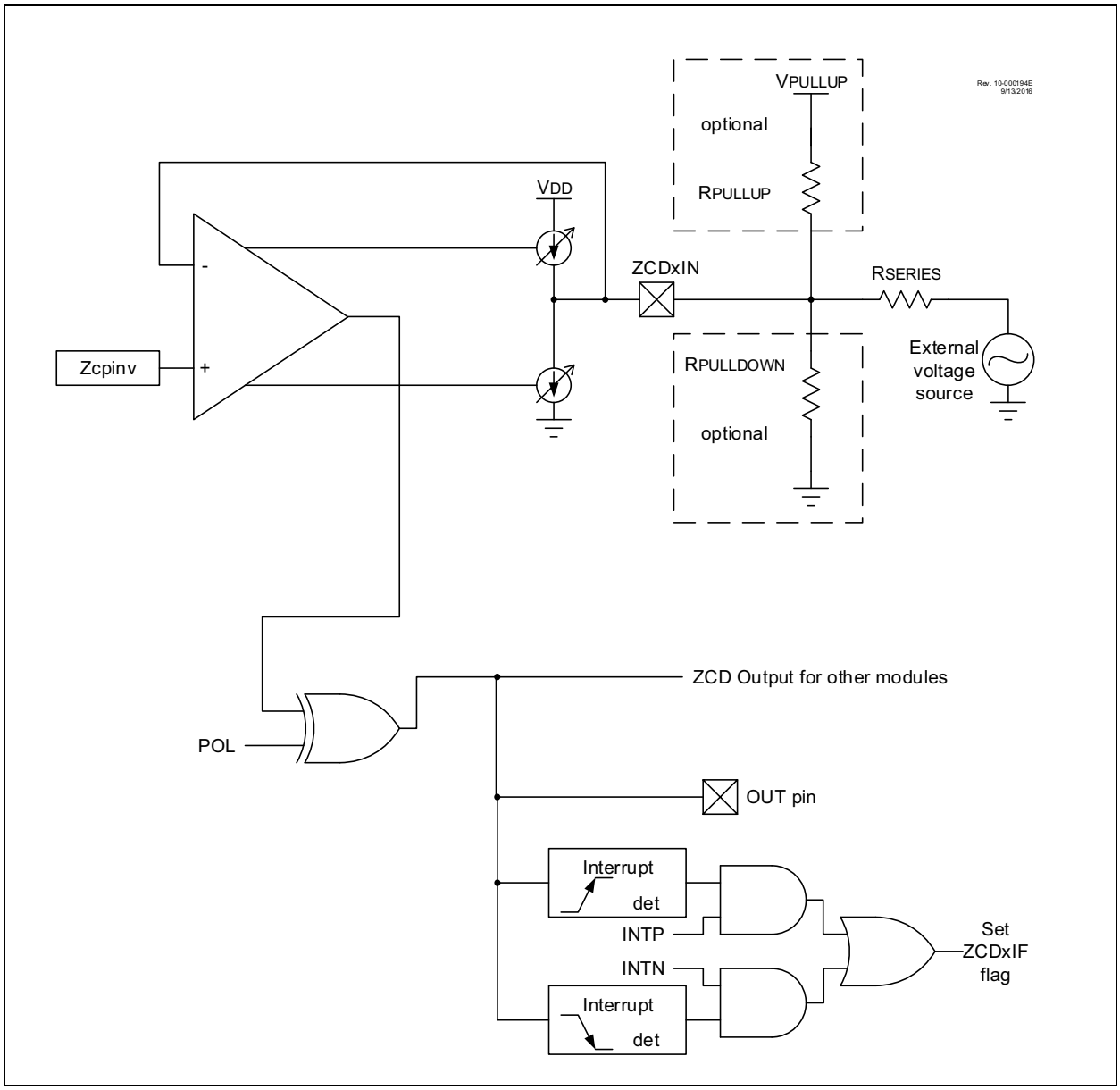


FIGURE 29-2: SIMPLIFIED ZCD BLOCK DIAGRAM



29.2 ZCD Logic Output

The ZCD module includes a Status bit, which can be read to determine whether the current source or sink is active. The OUT bit of the ZCDCON register is set when the current sink is active, and cleared when the current source is active. The OUT bit is affected by the polarity bit, even if the module is disabled.

The OUT signal can also be used as input to other modules. This is controlled by the registers of the corresponding module. OUT can be used as follows:

- Gate source for TMR1/3/5
- Clock source for TMR2/4/6
- Reset source for TMR2/4/6

29.3 ZCD Logic Polarity

The POL bit of the ZCDCON register inverts the OUT bit relative to the current source and sink output. When the POL bit is set, a OUT high indicates that the current source is active, and a low output indicates that the current sink is active.

The POL bit affects the ZCD interrupts.

29.4 ZCD Interrupts

An interrupt will be generated upon a change in the ZCD logic output when the appropriate interrupt enables are set. A rising edge detector and a falling edge detector are present in the ZCD for this purpose.

The ZCDIF bit of the respective PIR register will be set when either edge detector is triggered and its associated enable bit is set. The INTP enables rising edge interrupts and the INTN bit enables falling edge interrupts. Both are located in the ZCDCON register. Priority of the interrupt can be changed if the IPEN bit of the INTCON register is set. The ZCD interrupt can be made high or low priority by setting or clearing the ZCDIP bit of the respective IPR register.

To fully enable the interrupt, the following bits must be set:

- ZCDIE bit of the respective PIE register
- INTP bit of the ZCDCON register (for a rising edge detection)
- INTN bit of the ZCDCON register (for a falling edge detection)
- GIE bits of the INTCON0 register

Changing the POL bit can cause an interrupt, regardless of the level of the SEN bit.

The ZCDIF bit of the respective PIR register must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

29.5 Correcting for VCPINV offset

The actual voltage at which the ZCD switches is the reference voltage at the noninverting input of the ZCD op amp. For external voltage source waveforms other than square waves, this voltage offset from zero causes the zero-cross event to occur either too early or too late. When the waveform is varying relative to VSS, then the zero cross is detected too early as the waveform falls and too late as the waveform rises. When the waveform is varying relative to VDD, then the zero cross is detected too late as the waveform rises and too early as the waveform falls. The actual offset time can be determined for sinusoidal waveforms with the corresponding equations shown in Equation 29-2.

EQUATION 29-2: ZCD EVENT OFFSET

When External Voltage Source is relative to VSS:

$$T_{OFFSET} = \frac{\operatorname{asin}\left(\frac{V_{CPINV}}{V_{PEAK}}\right)}{2\pi \bullet Freq}$$

When External Voltage Source is relative to VDD:

$$T_{OFFSET} = \frac{\operatorname{asin}\left(\frac{V_{DD}-V_{CPINV}}{V_{PEAK}}\right)}{2\pi \bullet Freq}$$

This offset time can be compensated for by adding a pull-up or pull-down biasing resistor to the ZCD pin. A pull-up resistor is used when the external voltage source is varying relative to VSS. A pull-down resistor is used when the voltage is varying relative to VDD. The resistor adds a bias to the ZCD pin so that the target external voltage source must go to zero to pull the pin voltage to the VCPINV switching voltage. The pull-up or pull-down value can be determined with the equations shown in Equation 29-3 or Equation 29-4.

EQUATION 29-3: ZCD PULL-UP/DOWN

When External Signal is relative to Vss:

$$R_{PULLUP} = \frac{R_{SERIES}(V_{PULLUP} - V_{CPINV})}{V_{CPINV}}$$

When External Signal is relative to VDD:

$$R_{PULLDOWN} = \frac{R_{SERIES}(V_{CPINV})}{(V_{DD} - V_{CPINV})}$$

Measuring VCPINV can be difficult, especially when the waveform is relative to VDD. However, by combining Equations 29-2 and 29-3, the resistor value can be determined from the time difference between the ZCD_output high and low intervals. Note that the time difference, ΔT, is 4*TOFFSET. The equation for determining the pull-up and pull-down resistor values from the high and low ZCD_output periods is shown in Equation 29-4.

EQUATION 29-4: PULL-UP/DOWN RESISTOR VALUES

$$R = R_{SERIES}\left(\frac{V_{BIAS}}{V_{PEAK}\left(\sin\left(\pi Freq\frac{(\Delta T)}{2}\right)\right)} - 1\right)$$

R is pull-up or pull-down resistor.

VBIAS is VPULLUP when R is pull-up or VDD when R is pull-down.

ΔT is the ZCDOUT high and low period difference.

29.6 Handling VPEAK Variations

If the peak amplitude of the external voltage is expected to vary, the series resistor must be selected to keep the ZCD current source and sink below the design maximum range of ± 600 μA and above a reasonable minimum range. A general rule of thumb is that the maximum peak voltage can be no more than six times the minimum peak voltage. To ensure that the maximum current does not exceed ± 600 μA and the minimum is at least ± 100 μA, compute the series resistance as shown in Equation 29-5. The compensating pull-up for this series resistance can be determined with Equation 29-3 because the pull-up value is not dependent to the peak voltage.

EQUATION 29-5: SERIES R FOR V RANGE

$$R_{SERIES} = \frac{V_{MAXPEAK} + V_{MINPEAK}}{7\times10^{-4}}$$

29.7 Operation During Sleep

The ZCD current sources and interrupts are unaffected by Sleep.

29.8 Effects of a Reset

The ZCD circuit can be configured to default to the active or inactive state on Power-on-Reset (POR). When the $\overline{ZCD}$ Configuration bit is cleared, the ZCD circuit will be active at POR. When the $\overline{ZCD}$ Configuration bit is set, the SEN bit of the ZCDCON register must be set to enable the ZCD module.

29.9 Disabling the ZCD Module

The ZCD module can be disabled in two ways:

1. Configuration Word 2H has the $\overline{ZCD}$ bit which disables the ZCD module when set, but it can be enabled using the SEN bit of the ZCDCON register (Register 29-1). If the $\overline{ZCD}$ bit is clear, the ZCD is always enabled.
2. The ZCD can also be disabled using the ZCDMD bit of the respective PMD2 register (Register 19-3). This is subject to the status of the $\overline{ZCD}$ bit.

29.10 Register Definitions: ZCD Control

REGISTER 29-1: ZCDCON: ZERO-CROSS DETECT CONTROL REGISTER

R/W-0/0	U-0	R-x	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
SEN	—	OUT	POL	—	—	INTP	INTN
bit 7							bit 0

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

bit 7 **SEN:** Zero-Cross Detect Software Enable bit
This bit is ignored when ZCDSEN configuration bit is set.
1= Zero-cross detect is enabled.
0= Zero-cross detect is disabled. ZCD pin operates according to PPS and TRIS controls.

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **OUT:** Zero-Cross Detect Data Output bit
ZCDPOL bit = 0:
1 = ZCD pin is sinking current
0 = ZCD pin is sourcing current
ZCDPOL bit = 1:
1 = ZCD pin is sourcing current
0 = ZCD pin is sinking current

bit 4 **POL:** Zero-Cross Detect Polarity bit
1 = ZCD logic output is inverted
0 = ZCD logic output is not inverted

bit 3-2 **Unimplemented:** Read as ‘0’

bit 1 **INTP:** Zero-Cross Detect Positive-Going Edge Interrupt Enable bit
1 = ZCDIF bit is set on low-to-high ZCD_output transition
0 = ZCDIF bit is unaffected by low-to-high ZCD_output transition

bit 0 **INTN:** Zero-Cross Detect Negative-Going Edge Interrupt Enable bit
1 = ZCDIF bit is set on high-to-low ZCD_output transition
0 = ZCDIF bit is unaffected by high-to-low ZCD_output transition

TABLE 29-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE ZCD MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
ZCDCON	SEN	—	OUT	POL	—	—	INTP	INTN	463

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

30.0 DATA SIGNAL MODULATOR (DSM) MODULE

The Data Signal Modulator (DSM) is a peripheral which allows the user to mix a data stream, also known as a modulator signal, with a carrier signal to produce a modulated output.

Both the carrier and the modulator signals are supplied to the DSM module either internally, from the output of a peripheral, or externally through an input pin.

The modulated output signal is generated by performing a logical “AND” operation of both the carrier and modulator signals and then provided to the MDOUT pin.

The carrier signal is comprised of two distinct and separate signals. A carrier high (CARH) signal and a carrier low (CARL) signal. During the time in which the modulator (MOD) signal is in a logic high state, the DSM mixes the carrier high signal with the modulator signal. When the modulator signal is in a logic low state, the DSM mixes the carrier low signal with the modulator signal.

Using this method, the DSM can generate the following types of Key Modulation schemes:

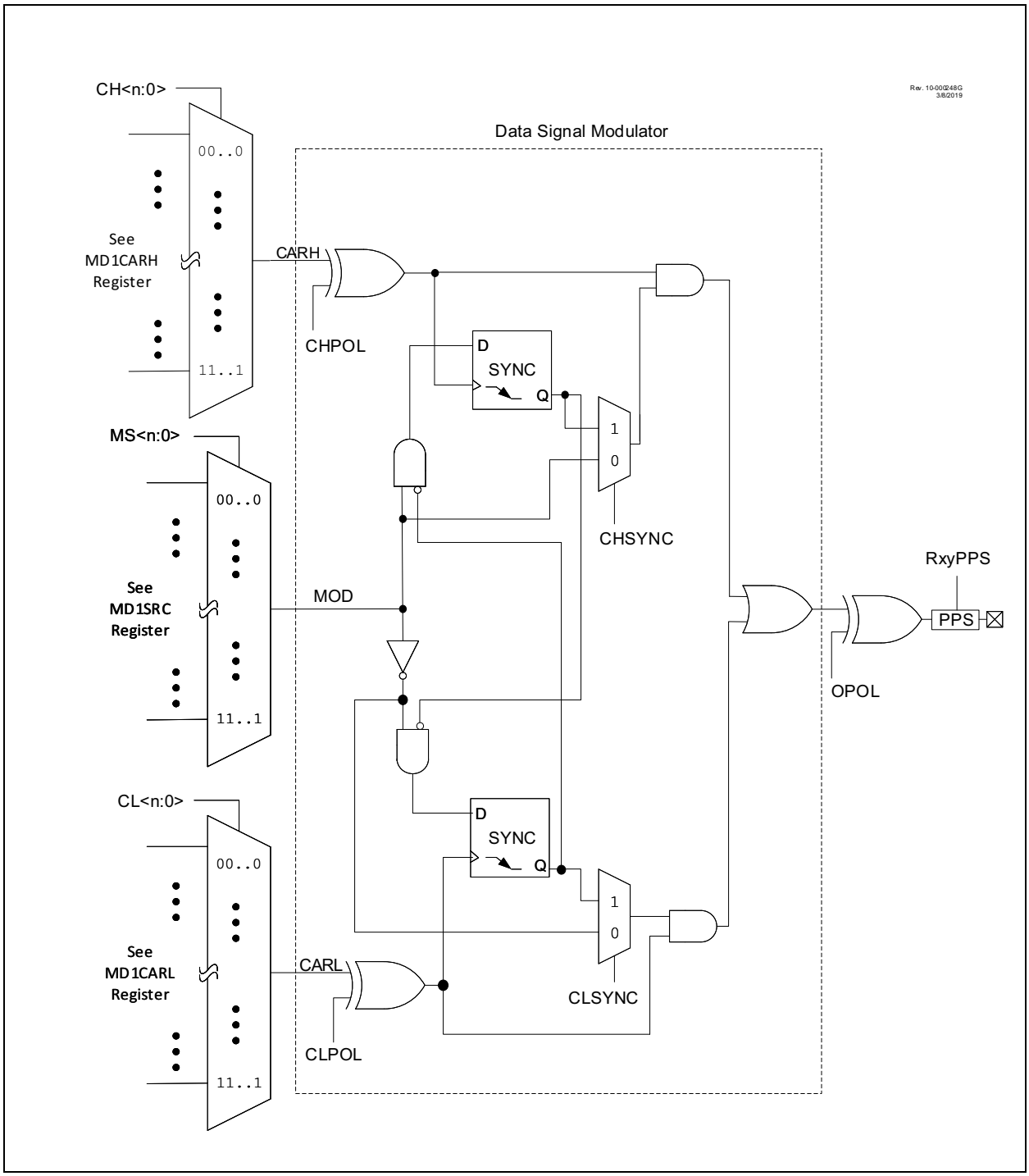
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

Additionally, the following features are provided within the DSM module:

- Carrier Synchronization
- Carrier Source Polarity Select
- Programmable Modulator Data
- Modulated Output Polarity Select
- Peripheral Module Disable, which provides the ability to place the DSM module in the lowest power consumption mode

Figure 30-1 shows a Simplified Block Diagram of the Data Signal Modulator peripheral.

FIGURE 30-1: SIMPLIFIED BLOCK DIAGRAM OF THE DATA SIGNAL MODULATOR



30.1 DSM Operation

The DSM module can be enabled by setting the EN bit in the MD1CON0 register. Clearing the EN bit in the MD1CON0 register, disables the DSM module output and switches the carrier high and carrier low signals to the default option of MD1CARHPPS and MD1CARLPPS, respectively. The modulator signal source is also switched to the BIT in the MD1CON0 register.

The values used to select the carrier high, carrier low, and modulator sources held by the Modulation Source, Modulation High Carrier, and Modulation Low Carrier control registers are not affected when the EN bit is cleared and the DSM module is disabled. The values inside these registers remain unchanged while the DSM is inactive. The sources for the carrier high, carrier low and modulator signals will once again be selected when the EN bit is set and the DSM module is again enabled and active.

30.2 Modulator Signal Sources

The modulator signal can be supplied from the sources specified in Table 30-3.

The modulator signal is selected by configuring the MS[4:0] bits in the MD1SRC register.

30.3 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from the sources specified in Table 30-1.

The carrier high signal is selected by configuring the CH[4:0] bits in the MD1CARH register. The carrier low signal is selected by configuring the CL[4:0] bits in the MD1CARL register.

30.4 Carrier Synchronization

During the time when the DSM switches between carrier high and carrier low signal sources, the carrier data in the modulated output signal can become truncated. To prevent this, the carrier signal can be synchronized to the modulator signal. When synchronization is enabled, the carrier pulse that is being mixed at the time of the transition is allowed to transition low before the DSM switches over to the next carrier source.

Synchronization is enabled separately for the carrier high and carrier low signal sources. Synchronization for the carrier high signal is enabled by setting the CHSYNC bit in the MD1CON1 register. Synchronization for the carrier low signal is enabled by setting the CLSYNC bit in the MD1CON1 register.

Figure 30-2 through Figure 30-6 show timing diagrams of using various synchronization methods.

FIGURE 30-2: On Off Keying (OOK) Synchronization

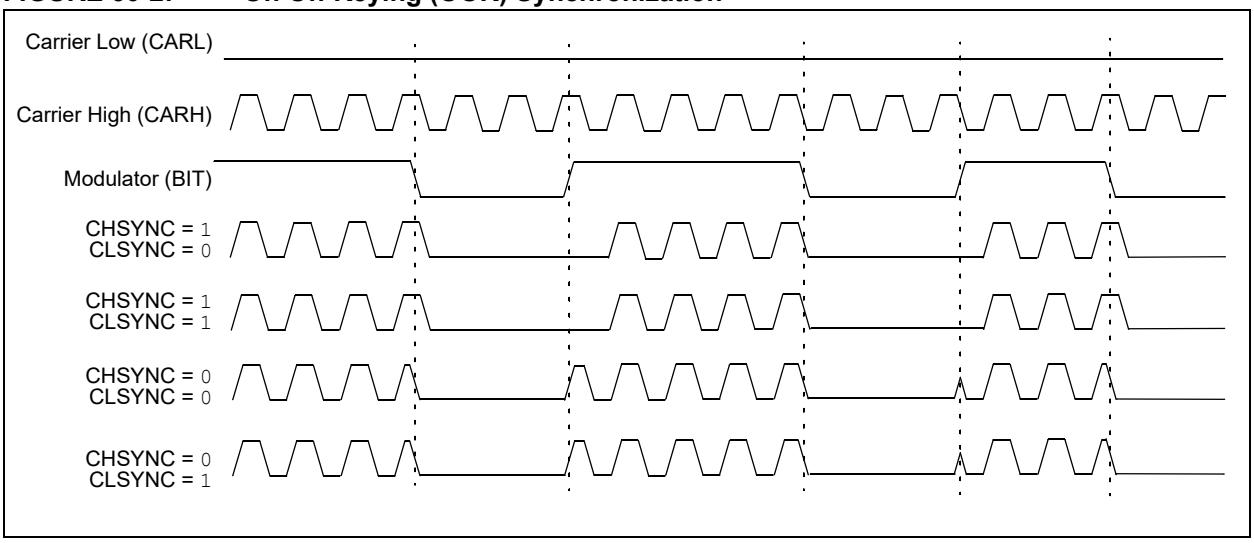


FIGURE 30-3: No Synchronization (CHSYNC = 0, CLSYNC = 0)

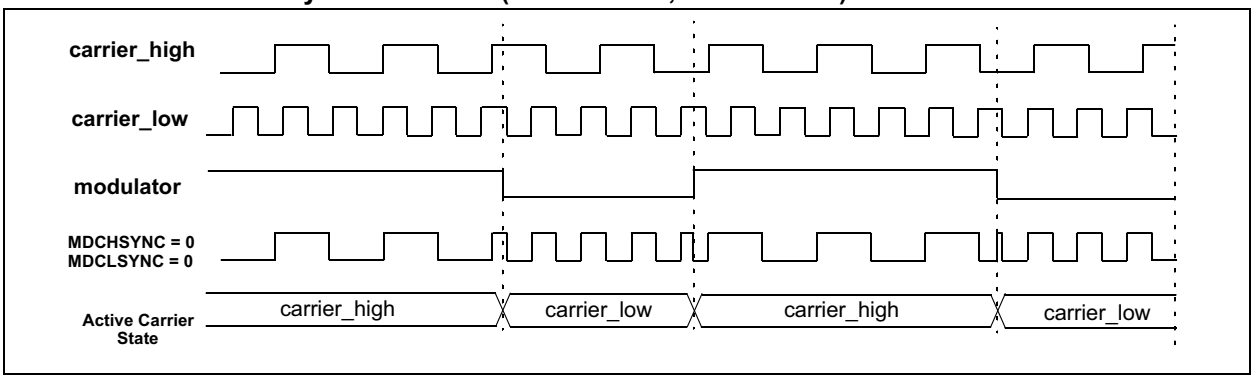


FIGURE 30-4: Carrier High Synchronization (CHSYNC = 1, CLSYNC = 0)

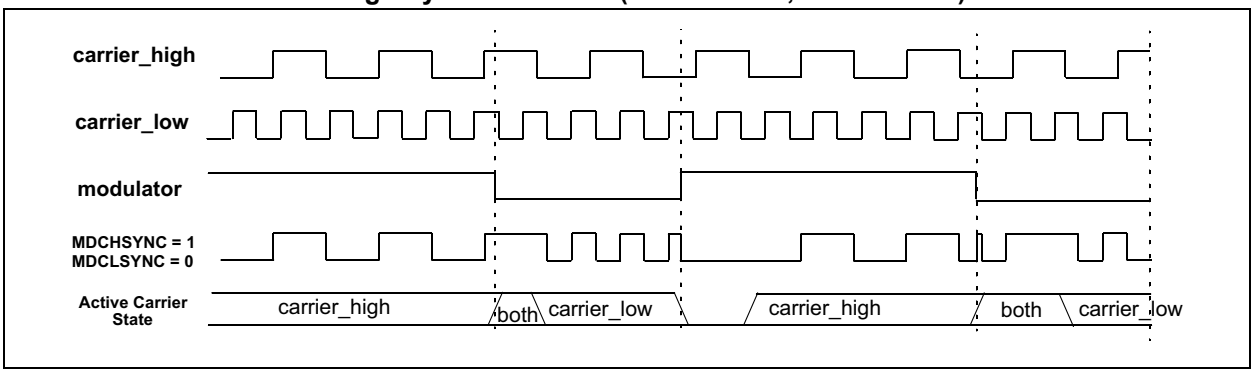


FIGURE 30-5: Carrier Low Synchronization (CHSYNC = 0, CLSYNC = 1)

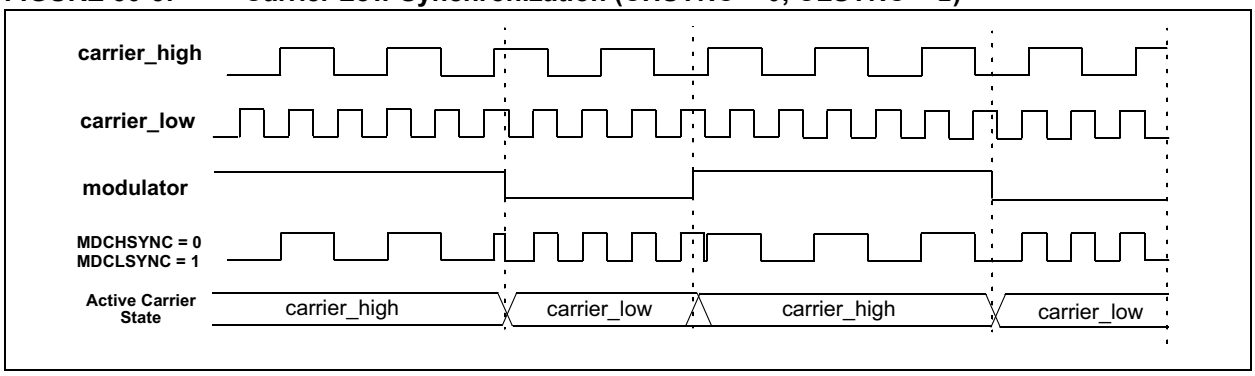
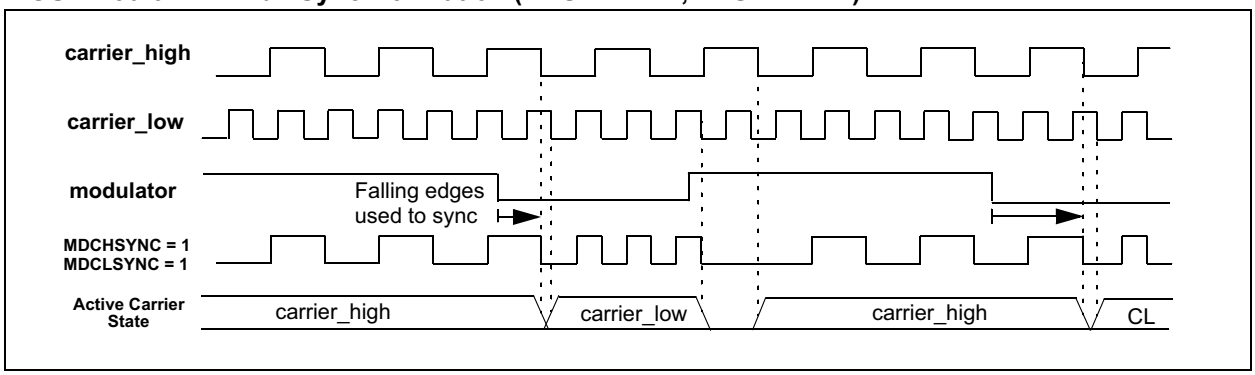


FIGURE 30-6: Full Synchronization (CHSYNC = 1, CLSYNC = 1)



30.5 Carrier Source Polarity Select

The signal provided from any selected input source for the carrier high and carrier low signals can be inverted. Inverting the signal for the carrier high source is enabled by setting the CHPOL bit of the MD1CON1 register. Inverting the signal for the carrier low source is enabled by setting the CLPOL bit of the MD1CON1 register.

30.6 Programmable Modulator Data

The BIT of the MD1CON0 register can be selected as the source for the modulator signal. This gives the user the ability to program the value used for modulation.

30.7 Modulated Output Polarity

The modulated output signal provided on the DSM pin can also be inverted. Inverting the modulated output signal is enabled by setting the OPOL bit of the MD1CON0 register.

30.8 Operation in Sleep Mode

The DSM module is not affected by Sleep mode. The DSM can still operate during Sleep, if the Carrier and Modulator input sources are also still operable during Sleep. Refer to **Section 10.0 “Power-Saving Operation Modes”** for more details.

30.9 Effects of a Reset

Upon any device Reset, the DSM module is disabled. The user’s firmware is responsible for initializing the module before enabling the output. The registers are reset to their default values.

30.10 Peripheral Module Disable

The DSM module can be completely disabled using the PMD module to achieve maximum power saving. The DSMMD bit of PMD6 (Register 19-7) when set disables the DSM module completely. When enabled again, all the registers of the DSM module default to POR status.

30.11 Register Definitions: Modulation Control

Long bit name prefixes for the Modulation peripheral is shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
MD1	MD1

REGISTER 30-1: MD1CON0: MODULATION CONTROL REGISTER 0

R/W-0/0	U-0	R-0/0	R/W-0/0	U-0	U-0	U-0	R/W-0/0
EN	—	OUT	OPOL	—	—	—	BIT
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 **EN:** Modulator Module Enable bit
1 = Modulator module is enabled and mixing input signals
0 = Modulator module is disabled and has no output

bit 6 **Unimplemented:** Read as '0'

bit 5 **OUT:** Modulator Output bit
Displays the current output value of the Modulator module.(1)

bit 4 **OPOL:** Modulator Output Polarity Select bit
1 = Modulator output signal is inverted; idle high output
0 = Modulator output signal is not inverted; idle low output

bit 3-1 **Unimplemented:** Read as '0'

bit 0 **BIT:** Allows software to manually set modulation source input to module(2)
1 = Modulator selects Carrier High
0 = Modulator selects Carrier Low

Note 1: The modulated output frequency can be greater and asynchronous from the clock that updates this register bit, the bit value may not be valid for higher speed modulator or carrier signals.

2: BIT bit must be selected as the modulation source in the MD1SRC register for this operation.

REGISTER 30-2: MD1CON1: MODULATION CONTROL REGISTER 1

U-0	U-0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	CHPOL	CHSYNC	—	—	CLPOL	CLSYNC
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 **Unimplemented:** Read as ‘0’

bit 5 **CHPOL:** Modulator High Carrier Polarity Select bit
1 = Selected high carrier signal is inverted
0 = Selected high carrier signal is not inverted

bit 4 **CHSYNC:** Modulator High Carrier Synchronization Enable bit
1 = Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier
0 = Modulator output is not synchronized to the high time carrier signal(1)

bit 3-2 **Unimplemented:** Read as ‘0’

bit 1 **CLPOL:** Modulator Low Carrier Polarity Select bit
1 = Selected low carrier signal is inverted
0 = Selected low carrier signal is not inverted

bit 0 **CLSYNC:** Modulator Low Carrier Synchronization Enable bit
1 = Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier
0 = Modulator output is not synchronized to the low time carrier signal(1)

Note 1: Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

REGISTER 30-3: MD1CARH: MODULATION HIGH CARRIER CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	CH[4:0](1)				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **CH[4:0]:** Modulator Carrier High Selection bits(1)
See Table 30-1 for signal list

Note 1: Unused selections provide an input value.

REGISTER 30-4: MD1CARL: MODULATION LOW CARRIER CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	CL[4:0](1)				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **CL[4:0]:** Modulator Carrier Low Input Selection bits(1)
See Table 30-1 for signal list

Note 1: Unused selections provide a zero as the input value.

TABLE 30-1: MD1CARH/MD1CARL SELECTION MUX CONNECTIONS

MD1CARH			MD1CARL		
CH[4:0]		Connection	CL[4:0]		Connection
11111-10011	31-19	Reserved	11111-10011	31-19	Reserved
10010	18	CLC4OUT	10010	18	CLC4OUT
10001	17	CLC3OUT	10001	17	CLC3OUT
10000	16	CLC2OUT	10000	16	CLC2OUT
01111	15	CLC1OUT	01111	15	CLC1OUT
01110	14	NCO1OUT	01110	14	NCO1OUT
01101-01100	13-12	Reserved	01101-01100	13-12	Reserved
01011	11	PWM8 OUT	01011	11	PWM8 OUT
01010	10	PWM7 OUT	01010	10	PWM7 OUT
01001	9	PWM6 OUT	01001	9	PWM6 OUT
01000	8	PWM5 OUT	01000	8	PWM5 OUT
00111	7	CCP4 OUT	00111	7	CCP4 OUT
00110	6	CCP3 OUT	00110	6	CCP3 OUT
00101	5	CCP2 OUT	00101	5	CCP2 OUT
00100	4	CCP1 OUT	00100	4	CCP1 OUT
00011	3	CLKREF output	00011	3	CLKREF output
00010	2	HFINTOSC	00010	2	HFINTOSC
00001	1	FOSC (system clock)	00001	1	FOSC (system clock)
00000	0	Pin selected by MD1CARHPPS	00000	0	Pin selected by MD1CARLPPS

REGISTER 30-5: MD1SRC: MODULATION SOURCE CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	MS[4:0]				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **MS[4:0]:** Modulator Source Selection bits(1)
See Table 30-2 for signal list

Note 1: Unused selections provide a zero as the input value.

TABLE 30-2: MD1SRC SELECTION MUX CONNECTIONS

MS[4:0]		Connection
1 1111 – 1 0111	31-23	Reserved
1 0110	22	SPI1 SDO
1 0101	21	Reserved
1 0100	20	UART2 TX
1 0011	19	UART1 TX
1 0010	18	CLC4 OUT
1 0001	17	CLC3 OUT
1 0000	16	CLC2 OUT
0 1111	15	CLC1 OUT
0 1110	14	CMP2 OUT
0 1101	13	CMP1 OUT
0 1100	12	NCO1 OUT
0 1011	11	Reserved
0 1010	10	Reserved
0 1001	9	PWM8 OUT
0 1000	8	PWM7 OUT
0 0111	7	PWM6 OUT
0 0110	6	PWM5 OUT

TABLE 30-2: MD1SRC SELECTION MUX CONNECTIONS

MS[4:0]		Connection
0 0101	5	CCP4 OUT
0 0100	4	CCP3 OUT
0 0011	3	CCP2 OUT
0 0010	2	CCP1 OUT
0 0001	1	DSM1 BIT
0 0000	0	Pin selected by MDSRCPPS

TABLE 30-3: SUMMARY OF REGISTERS ASSOCIATED WITH DATA SIGNAL MODULATOR MODE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
MD1CON0	EN	—	OUT	OPOL	—	—	—	BIT	470
MD1CON1	—	—	CHPOL	CHSYNC	—	—	CLPOL	CLSYNC	471
MD1CARH	—	—	—	CH[4:0]					472
MD1CARL	—	—	—	CL[4:0]					472
MD1SRC	—	—	—	MS[4:0]					473

Legend: — = unimplemented, read as '0'. Shaded cells are not used in the Data Signal Modulator mode.

31.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) WITH PROTOCOL SUPPORT

The Universal Asynchronous Receiver Transmitter (UART) 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 UART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or one of several automated protocols. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers.

Supported protocols include:

- LIN Master and Slave
- DMX mode
- DALI control gear and control device

The UART module includes the following capabilities:

- Full-duplex asynchronous transmit and receive
- Two-character input buffer
- One-character output buffer
- Programmable 7-bit or 8-bit character length
- 9th bit Address detection
- 9th bit even or odd parity
- Input buffer overrun error detection
- Received character framing error detection
- Hardware and software flow control
- Automatic checksums
- Programmable 1, 1.5, and 2 Stop bits
- Programmable data polarity
- Manchester encoder/decoder
- Operation in Sleep
- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- Automatic and user timed Break period generation
- RX and TX inactivity timeouts (with Timer2)

Block diagrams of the UART transmitter and receiver are shown in Figure 31-1 and Figure 31-2.

The UART transmit output (TX_out) is available to the TX pin and internally to various peripherals.

FIGURE 31-1: UART TRANSMIT BLOCK DIAGRAM

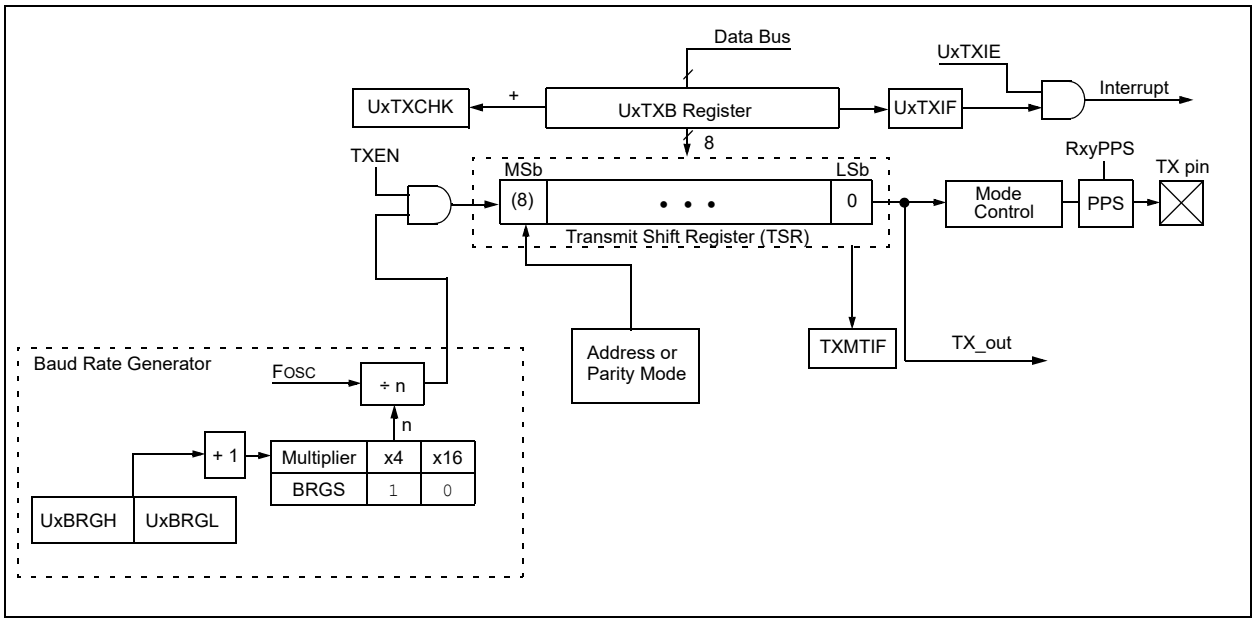
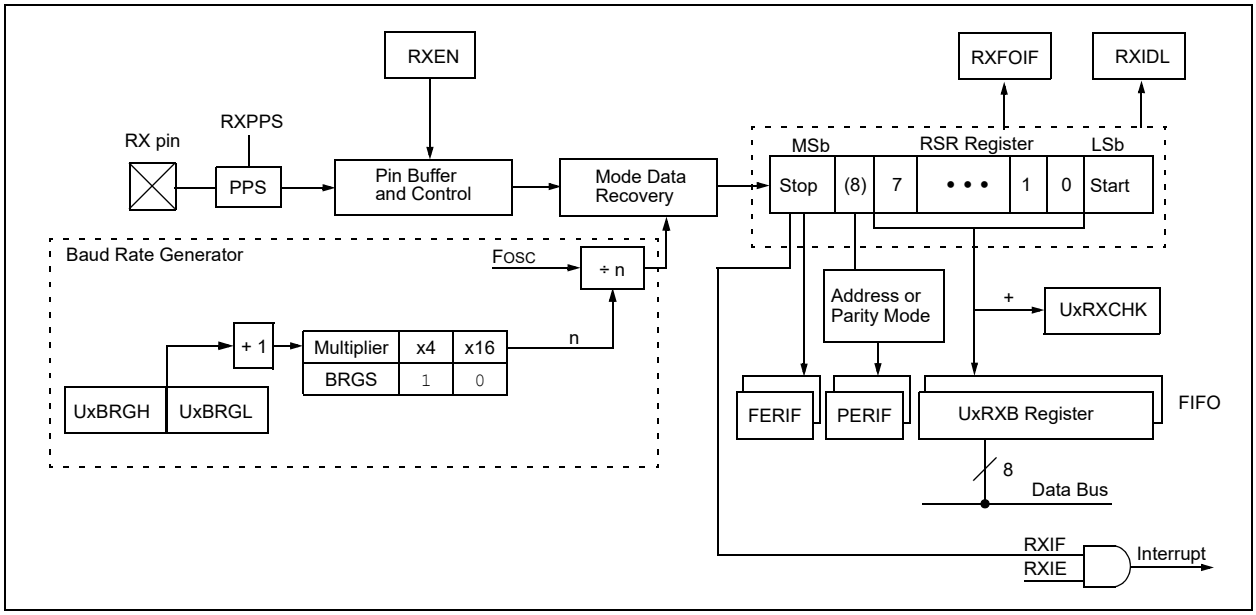


FIGURE 31-2: UART RECEIVE BLOCK DIAGRAM



The operation of the UART module is controlled through nineteen registers:

- Three control registers (UxCON0-UxCON2)
- Error enable and status (UxERRIE, UxERRIR, UxUIR)
- UART buffer status and control (UxFIFO)
- Three 9-bit protocol parameters (UxP1-UxP3)
- 16-bit baud rate generator (UxBRGH:L)
- Transmit buffer write (UxTXB)
- Receive buffer read (UxRXB)
- Receive checksum (UxRXCHK)
- Transmit checksum (UxTXCHK)

These registers are detailed in **Section 31.21 "Register Definitions: UART Control"**.

31.1 UART I/O Pin Configuration

The RX input pin is selected with the UxRPPS register. The TX output pin is selected with each pin's RxyPPS register. When the TRIS control for the pin corresponding to the TX output is cleared, then the UART will maintain control and the logic level on the TX pin. Changing the TXPOL bit in UxCON2 will immediately change the TX pin logic level regardless of the value of EN or TXEN.

31.2 UART Asynchronous Modes

The UART has five asynchronous modes:

- 7-bit
- 8-bit
- 8-bit with even parity in the 9th bit
- 8-bit with odd parity in the 9th bit
- 8-bit with address indicator in the 9th bit

The UART 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 seven or eight data bits, one optional parity or address bit, 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 with no parity. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See **Section 31.17 "UART Baud Rate Generator (BRG)"** for more information.

In all the asynchronous modes, the UART transmits and receives the LSb first. The UART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is supported by the hardware by even and odd parity modes.

31.2.1 UART ASYNCHRONOUS TRANSMITTER

The UART transmitter block diagram is shown in Figure 31-1. 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 UxTXB register.

31.2.1.1 Enabling the Transmitter

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

- TXEN = 1
- MODE[3:0] = 0h through 3h
- UxBRGH:L = desired baud rate
- UxBRGS = desired baud rate multiplier
- RxyPPS = code for desired output pin
- ON = 1

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

Setting the TXEN bit in the UxCON0 register enables the transmitter circuitry of the UART. The MODE[3:0] bits in the UxCON0 register select the desired mode. Setting the ON bit in the UxCON1 register enables the UART. When TXEN is set and the transmitter is not idle, the TX pin is automatically configured as an output. When the transmitter is idle, the TX pin drive is relinquished to the port TRIS control. If the TX pin is shared with an analog peripheral, the analog I/O function should be disabled by clearing the corresponding ANSEL bit.

Note:	The UxTXIF Transmitter Interrupt flag is set when the TXEN enable bit is set and the UxTXB register can accept data.

31.2.1.2 Transmitting Data

A transmission is initiated by writing a character to the UxTXB register. If this is the first character, or the previous character has been completely transmitted from the TSR, the data in the UxTXB 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 UxTXB until the previous character transmission is complete. The pending character in the UxTXB is then transferred to the TSR at the beginning of the previous character Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the completion of all of the previous character's Stop bits.

31.2.1.3 Transmit Data Polarity

The polarity of the transmit data is controlled with the TXPOL bit in the UxCON2 register. The default state of this bit is '0' which selects high true transmit idle and data bits. Setting the TXPOL bit to '1' will invert the transmit data, resulting in low true idle and data bits. The TXPOL bit controls transmit data polarity in all modes.

31.2.1.4 Transmit Interrupt Flag

The UxTXIF interrupt flag bit in the PIR register is set whenever the UART transmitter is enabled and no character is being held for transmission in the UxTXB. In other words, the UxTXIF bit is clear only when the TSR is busy with a character and a new character has been queued for transmission in the UxTXB.

The UxTXIF interrupt can be enabled by setting the UxTXIE interrupt enable bit in the PIE register. However, the UxTXIF flag bit will be set whenever the UxTXB is empty, regardless of the state of UxTXIE enable bit.The UxTXIF bit is read-only and cannot be set or cleared by software.

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

31.2.1.5 TSR Status

The TXMTIF bit in the UxERRIR register indicates the status of the TSR. This is a read-only bit. The TXMTIF bit is set when the TSR is empty and idle. The TXMTIF bit is cleared when a character is transferred to the TSR from the UxTXB. The TXMTIF bit remains clear until all bits, including the Stop bits, have been shifted out of the TSR and a byte is not waiting in the UxTXB register.

The TXMTIF will generate an interrupt when the TXMTIE bit in the UxERRIE register is set.

Note:	The TSR is not mapped in data memory, so it is not available to the user.

31.2.1.6 Transmitter 7-bit Mode

7-Bit mode is selected when the MODE[3:0] bits are set to '0001'. In 7-bit mode, only the seven Least Significant bits of the data written to UxTXB are transmitted. The Most Significant bit is ignored.

31.2.1.7 Transmitter Parity Modes

When the Odd or even Parity mode is selected, all data is sent as nine bits. The first eight bits are data and the 9th bit is parity. Even and odd parity is selected when the MODE[3:0] bits are set to '0011' and '0010', respectively. Parity is automatically determined by the module and inserted in the serial data stream.

31.2.1.8 Asynchronous Transmission Setup

1. Initialize the UxBRGH, UxBRGL register pair and the BRGS bit to achieve the desired baud rate (see **Section 31.17 "UART Baud Rate Generator (BRG)"**).
2. Set the MODE[3:0] bits to the desired asynchronous mode.
3. Set TXPOL bit if inverted TX output is desired.
4. Enable the asynchronous serial port by setting the ON bit.
5. Enable the transmitter by setting the TXEN control bit. This will cause the UxTXIF interrupt flag to be set.
6. If the device has PPS, configure the desired I/O pin RxyPPS register with the code for TX output.
7. If interrupts are desired, set the UxTXIE interrupt enable bit in the respective PIE register. An interrupt will occur immediately provided that the GIE bits in the INTCON0 register are also set.
8. Write one byte of data into the UxTXB register. This will start the transmission.
9. Subsequent bytes may be written when the UxTXIF bit is '1'.

FIGURE 31-3: ASYNCHRONOUS TRANSMISSION

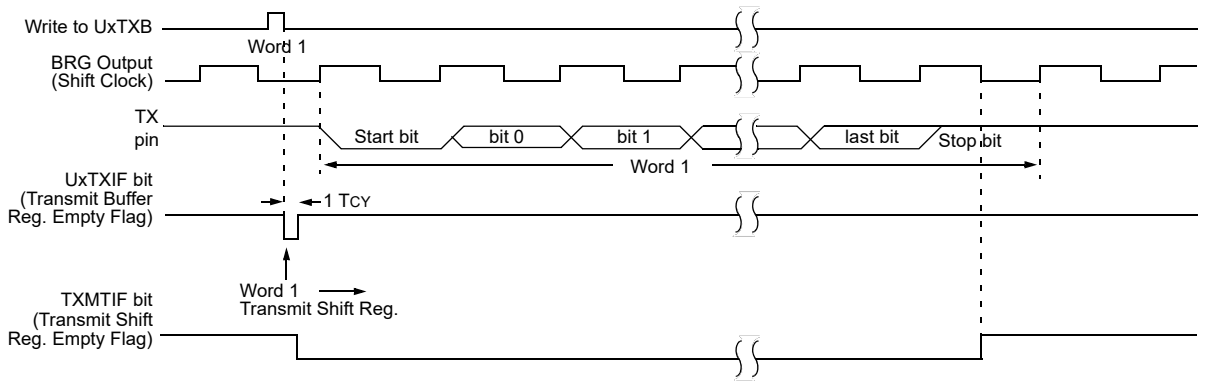
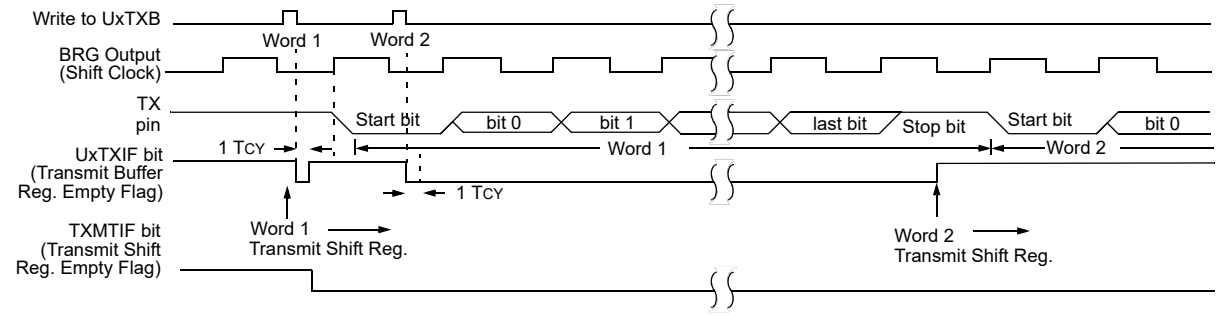


FIGURE 31-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)



Note: This timing diagram shows the first transmission and the start of the second consecutive transmission.

31.2.2 UART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 31-2. The data is received on the RX pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 4 or 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the UART receiver. The FIFO registers and RSR are not directly accessible by software. Access to the received data is via the UxRXB register.

31.2.2.1 Enabling the Receiver

The UART receiver is enabled for asynchronous operation by configuring the following control bits:

- RXEN = 1
- MODE[3:0] = 0h through 3h
- UxBRGH:L = desired baud rate
- RXPPS = code for desired input pin
- Input pin ANSEL bit = 0
- ON = 1

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

Setting the RXEN bit in the UxCON0 register enables the receiver circuitry of the UART. Setting the MODE[3:0] bits in the UxCON0 register configures the UART for the desired asynchronous mode. Setting the ON bit in the UxCON1 register enables the UART. The TRIS bit corresponding to the selected RX I/O pin must be set to configure the pin as an input.

Note:	If the RX function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

31.2.2.2 Receiving Data

Data is recovered from the bit stream by timing to the center of the bits and sampling the input level. In High-Speed mode, there are four BRG clocks per bit and only one sample is taken per bit. In Normal-Speed mode, there are 16 BRG clocks per bit and three samples are taken per bit.

The receiver data recovery circuit initiates character reception on the falling edge of the Start bit. The Start bit, is always a '0'. The Start bit is qualified in the middle of the bit. In Normal-Speed mode only, the Start bit is also qualified at the leading edge of the bit. The following paragraphs describe the majority detect sampling of Normal-Speed mode.

The falling edge starts the baud rate generator (BRG) clock. The input is sampled at the first and second BRG clocks.

If both samples are high then the falling edge is deemed a glitch and the UART returns to the Start bit detection state without generating an error.

If either sample is low, the data recovery circuit continues counting BRG clocks and takes samples at clock counts 7, 8, and 9. When less than two samples are low, the Start bit is deemed invalid and the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit.

When two or more samples are low, the Start bit is deemed valid and the data recovery continues. After a valid Start bit is detected, the BRG clock counter continues and resets at count 16. This is the beginning of the first data bit.

The data recovery circuit counts BRG clocks from the beginning of the bit and takes samples at clocks 7, 8, and 9. The bit value is determined from the majority of the samples. The resulting '0' or '1' is shifted into the RSR.The BRG clock counter continues and resets at count 16. This sequence repeats until all data bits have been sampled and shifted into the RSR.

After all data bits have been shifted in, the first Stop bit is sampled. Stop bits are always a '1'. If the bit sampling determines that a '0' is in the Stop bit position, the framing error is set for this character. Otherwise, the framing error is cleared for this character. See **Section 31.2.2.4 "Receive Framing Error"** for more information on framing errors.

31.2.2.3 Receive Interrupts

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the UART receive FIFO. The UxRXIF interrupt flag in the respective PIR register is set at this time, provided it is not being suppressed.

The UxRXIF is suppressed by any of the following:

- FERIF if FERIE is set
- PERIF if PERIE is set

This suspends DMA transfer of data until software processes the error and reads UxRXB to advance the FIFO beyond the error.

UxRXIF interrupts are enabled by setting all of the following bits:

- UxRXIE, Interrupt Enable bit in the PIE register
- GIE, Global Interrupt Enable bits in the INTCON0 register

The UxRXIF interrupt flag bit will be set when not suppressed and there is an unread character in the FIFO, regardless of the state of interrupt enable bits. Reading the UxRXB register will transfer the top character out of the FIFO and reduce the FIFO contents by one. The UxRXIF interrupt flag bit is read-only, it cannot be set or cleared by software.

31.2.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error flag bit. A framing error indicates that the Stop bit was not seen at the expected time. The framing error flag is accessed via the FERIF bit in the UxERRIR register. The FERIF bit represents the frame status of the top unread character of the receive FIFO. Therefore, the FERIF bit must be read before reading UxRXB.

The FERIF bit is read-only and only applies to the top unread character of the receive FIFO. A framing error (FERIF = 1) does not preclude reception of additional characters. It is neither necessary nor possible to clear the FERIF bit directly. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERIF bit is cleared when the character at the top of the FIFO does not have a framing error or when all bytes in the receive FIFO have been read. Clearing the ON bit resets the receive FIFO, thereby also clearing the FERIF bit.

A framing error will generate a summary UxERR interrupt when the FERIE bit in the UxERRIE register is set. The summary error is reset when the FERIF bit of the top of the FIFO is '0' or when all FIFO characters have been retrieved.

When FERIE is set, UxRXIF interrupts are suppressed when FERIF is '1'.

31.2.2.5 Receiver Parity Modes

Even and odd parity is automatically detected when the MODE[3:0] bits are set to '0011' and '0010', respectively. Parity modes receive eight data bits and one parity bit for a total of nine bits for each character. The PERIF bit in the UxERRIR register represents the parity error of the top unread character of the receive FIFO rather than the parity bit itself. The parity error must be read before reading the UxRXB register advances the FIFO.

A parity error will generate a summary UxERR interrupt when the PERIE bit in the UxERRIE register is set.The summary error is reset when the PERIF bit of the top of the FIFO is '0' or when all FIFO characters have been retrieved.

When PERIE is set, UxRXIF interrupts are suppressed when PERIF is '1'.

31.2.2.6 Receive FIFO Overflow

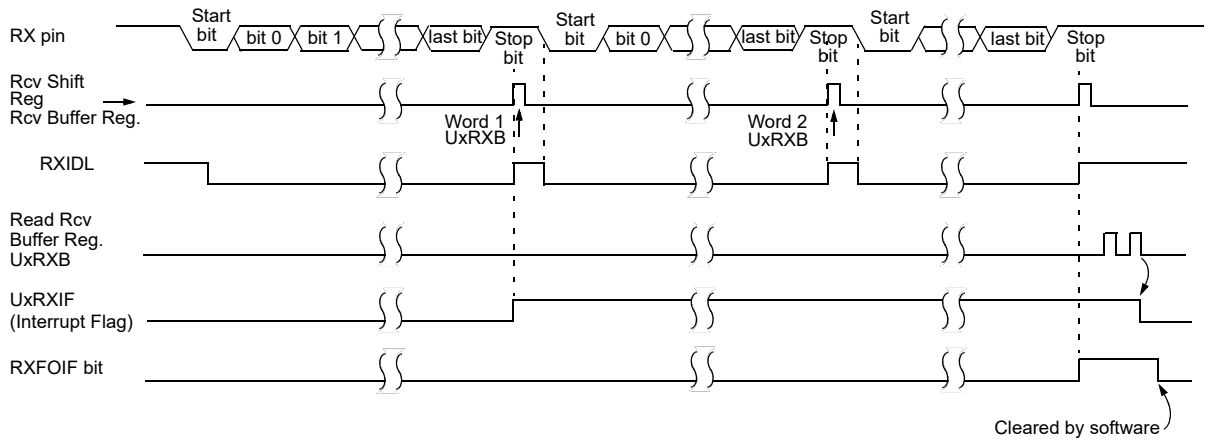
When more characters are received than the receive FIFO can hold, the RXFOIF bit in the UxERRIR register is set. The character causing the overflow condition is discarded. The RUNOVF bit in the UxCON2 register determines how the receive circuit responds to characters while the overflow condition persists. When RUNOVF is set, the receive shifter stays synchronized to the incoming data stream by responding to Start, data, and Stop bits. However, all received bytes not already in the FIFO are discarded. When RUNOVF is cleared, the receive shifter ceases operation and Start, data, and Stop bits are ignored. The receive overflow condition is cleared by reading the UxRXB register and clearing the RXFOIF bit. If the UxRXB register is not read to open a space in the FIFO, the next character received will be discarded and cause another overflow condition.

A receive overflow error will generate a summary UxEIF interrupt when the RXFOIE bit in the UxERRIE register is set.

31.2.2.7 Asynchronous Reception Setup

1. Initialize the UxBRGH, UxBRGL register pair and the BRGS bit to achieve the desired baud rate (see **Section 31.17 "UART Baud Rate Generator (BRG)"**).
2. Configure the RXPPS register for the desired RX pin
3. Clear the ANSEL bit for the RX pin (if applicable).
4. Set the MODE[3:0] bits to the desired asynchronous mode.
5. Set the RXPOL bit if the data stream is inverted.
6. Enable the serial port by setting the ON bit.
7. If interrupts are desired, set the UxRXIE bit in the PIEx register and the GIE bits in the INTCON0 register.
8. Enable reception by setting the RXEN bit.
9. The UxRXIF 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 UxRXIE interrupt enable bit is also set.
10. Read the UxERRIR register to get the error flags.
11. Read the UxRXB register to get the received byte.
12. If an overrun occurred, clear the RXFOIF bit.

FIGURE 31-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The UxRXB (receive buffer) is not read before the third word is received, causing the RXFOIF (FIFO overrun) bit to be set. STPMD = 0, STP[1:0] = 00.

31.3 Asynchronous Address Mode

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems.

When Asynchronous Address mode is enabled, all data is transmitted and received as 9-bit characters. The 9th bit determines whether the character is an address or data. When the 9th bit is set, the eight Least Significant bits are the address. When the 9th bit is clear, the Least Significant bits are data. In either case, the 9th bit is stored in PERIF when the byte is written to the receive FIFO. When PERIE is also set, the RXIF will be suppressed, thereby suspending DMA transfers allowing software to process the received address.

An address character will enable all receivers that match the address and disable all other receivers. Once a receiver is enabled, all non-address characters will be received until an address character is received that does not match.

31.3.1 ADDRESS MODE TRANSMIT

The UART transmitter is enabled for asynchronous address operation by configuring the following control bits:

- TXEN = `1`
- MODE[3:0] = `0100`
- UxBRGH:L = desired baud rate
- RxyPPS = code for desired output pin
- ON = `1`

Addresses are sent by writing to the UxP1L register. This transmits the written byte with the 9th bit set, which indicates that the byte is an address.

Data is sent by writing to the UxTXB register. This transmits the written byte with the 9th bit cleared, which indicates that the byte is data.

To send data to a particular device on the transmission bus, first transmit the address of the intended device. All subsequent data will be accepted only by that device until an address of another device is transmitted.

Writes to UxP1L take precedence over writes to UxTXB. When both the UxP1L and UxTXB registers are written while the TSR is busy, the next byte to be transmitted will be from UxP1L.

To ensure that all data intended for one device is sent before the address is changed, wait until the TXMTIF bit is high before writing UxP1L with the new address.

31.3.2 ADDRESS MODE RECEIVE

The UART receiver is enabled for asynchronous address operation by configuring the following control bits:

- RXEN = `1`
- MODE[3:0] = `0100`
- UxBRGH:L = desired baud rate
- RXPPS = code for desired input pin
- Input pin ANSEL bit = `0`
- UxP2L = receiver address
- UxP3L = address mask
- ON = `1`

In Address mode, no data will be transferred to the input FIFO until a valid address is received. This is the default state. Any of the following conditions will cause the UART to revert to the default state:

- ON = `0`
- RXEN = `0`
- Received address does not match

When a character with the 9th bit set is received, the Least Significant eight bits of that character will be qualified by the values in the UxP2L and UxP3L registers.

The byte is XOR'd with UxP2L then AND'd with UxP3L. A match occurs when the result is 0h, in which case, the unaltered received character is stored in the receive FIFO, thereby setting the UxRXIF interrupt bit. The 9th bit is stored in the corresponding PERIF bit, identifying this byte as an address.

An address match also enables the receiver for all data such that all subsequent characters without the 9th bit set will be stored in the receive FIFO.

When the 9th bit is set and a match does not occur, the character is not stored in the receive FIFO and all subsequent data is ignored.

The UxP3L register mask allows a range of addresses to be accepted. Software can then determine the sub-address of the range by processing the received address character.

© 2017-2020 Microchip Technology Inc.

31.4 DMX Mode (UART1 only)

DMX is a protocol used in stage and show equipment. This includes lighting, fog machines, motors, etc. The protocol consists of a controller that sends out commands, and receiver such as theater lights that receive these commands. DMX protocol is usually unidirectional, but can be a bidirectional protocol in either Half or Full-duplex modes. An example of Half-duplex mode is the RDM (Remote Device Management) protocol that sits on DMX512A. The controller transmits commands and the receiver receives them. Also there are no error conditions or re-transmit mechanisms.

DMX, or DMX512A as it is known, consists of a “Universe” of 512 channels. This means that one controller can output up to 512 bytes on a single DMX link. Each equipment on the line is programmed to listen to a consecutive sequence of one or more of these bytes.

For example, a fog machine connected to one of the universes may be programmed to receive one byte, starting at byte number 10, and a lighting unit may be programmed to receive four bytes starting at byte number 22.

31.4.1 DMX CONTROLLER

DMX Controller mode is configured with the following settings:

- MODE[3:0] = `1010`
- TXEN = `1`
- RXEN = `0`
- TXPOL = `0`
- UxP1 = One less than the number of bytes to transmit (excluding the Start code)
- UxBRGH:L = Value to achieve 250K baud rate
- STP[1:0] = `10` for 2 Stop bits
- RxyPPS = TX pin output code
- ON = `1`

Each DMX transmission begins with a Break followed by a byte called the ‘Start Code’. The width of the BREAK is fixed at 25 bit times. The Break is followed by a “Mark After Break” (MAB) Idle period. After this Idle period, the 1st through ‘n’th byte is transmitted, where ‘n-1’ is the value in UxP1. See Figure 31-6.

Software sends the Start Code and the ‘n’ data bytes by writing the UxTXB register with each byte to be sent in the desired order. A UxTXIF value of ‘`1`’ indicates when the UxTXB is ready to accept the next byte.

The internal byte counter is not accessible to software. Software needs to keep track of the number of bytes written to UxTXB to ensure that no more and no less than ‘n’ bytes are sent because the DMX state machine will automatically insert a Break and reset its internal counter after ‘n’ bytes are written. One way to ensure synchronization between hardware and software is to toggle TXEN after the last byte of the universe is completely free of the transmit shift register as indicated by the TXMTIF bit.

31.4.2 DMX RECEIVER

DMX Receiver mode is configured with the following settings:

- MODE[3:0] = `1010`
- TXEN = `0`
- RXEN = `1`
- RXPOL = `0`
- UxP2 = number of first byte to receive
- UxP3 = number of last byte to receive
- UxBRGH:L = Value to achieve 250K baud rate
- STP[1:0] = `10` for 2 Stop bits
- ON = `1`
- UxRXPPS = code for desired input pin
- Input pin ANSEL bit = `0`

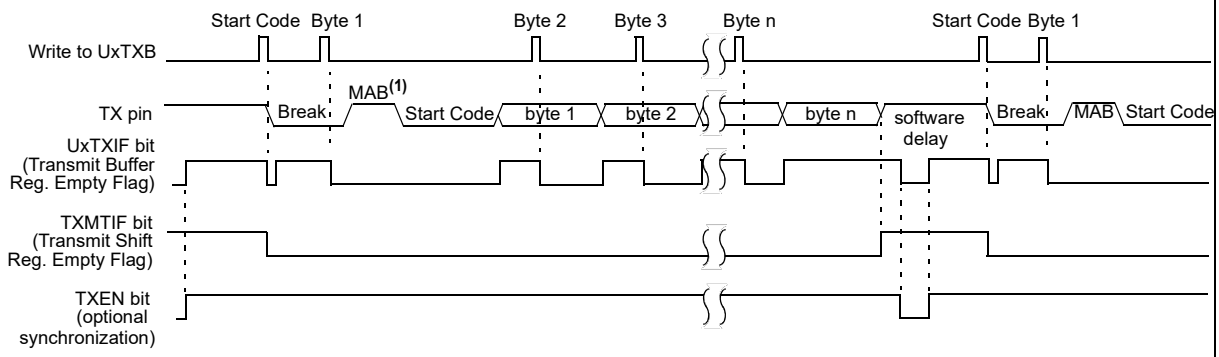
When configured as DMX Receiver, the UART listens for a Break character that is at least 23 bit periods wide. If the Break is shorter than 23 bit times, the Break is ignored and the DMX state machine remains in Idle mode. Upon receiving the Break, the DMX counters will be reset to align with the incoming data stream. Immediately after the Break, the UART will see the “Mark after Break” (MAB). This space is ignored by the UART. The Start Code follows the MAB and will always be stored in the receive FIFO.

After the Start Code, the 1st through 512th byte will be received, but not all of them are stored in the receive FIFO. The UART ignores all received bytes until the ones of interest are received. This is done using the UxP2 and UxP3 registers. The UxP2 register holds the value of the byte number to start the receive process. The byte counter starts at 0 for the first byte after the Start Code. For example, to receive four bytes starting at the 10th byte after the Start Code, write 009h (9 decimal) to UxP2H:L and 00Ch (12 decimal) to UxP3H:L. The receive FIFO is only 2 bytes deep, therefore the bytes must be retrieved by reading UxRXB as they come in to avoid a receive FIFO overrun condition.

Typically two Stop bits are inserted between bytes. If either Stop bit is detected as a ‘`0`’ then the framing error for that byte will be set.

Since the DMX sequence always starts with a Break, the software can verify that it is in sync with the sequence by monitoring the RXBKIF flag to ensure that the next byte received after the RXBKIF is processed as the Start Code and subsequent bytes are processed as the expected data.

FIGURE 31-6: DMX TRANSMIT SEQUENCE



Note 1: The MAB period is fixed at 3-bits period.

31.5 LIN Modes (UART1 only)

LIN is a protocol used primarily in automotive applications. The LIN network consists of two kinds of software processes: a Master process and a Slave process. Each network has only one Master process and one or more Slave processes.

From a physical layer point of view, the UART on one processor may be driven by both a Master and a Slave process, as long as only one Master process exists on the network.

A LIN transaction consists of a Master process followed by a Slave process. The Slave process may involve more than one Slave where one is transmitting and the other(s) are receiving. The transaction begins by the following Master process transmission sequence:

1. Break
2. Delimiter bit
3. Sync Field
4. PID byte

The PID determines which Slave processes are expected to respond to the Master. When the PID byte is complete, the TX output remains in the Idle state. One or more of the Slave processes may respond to the Master process. If no one responds within the inter-byte period, the Master is free to start another transmission. The inter-byte period is timed by software using a means other than the UART.

The Slave process follows the Master process. When the Slave software recognizes the PID then that Slave process responds by either transmitting the required response or by receiving the transmitted data. Only Slave processes send data. Therefore, Slave processes receiving data are receiving that of another Slave process.

When a Slave sends data, the Slave UART automatically calculates the checksum for the transmitted bytes as they are sent and appends the inverted checksum byte to the slave response.

When a Slave receives data, the checksum is accumulated on each byte as it is received using the same algorithm as the sending process. The last byte, which is the inverted checksum value calculated by the sending process, is added to the locally calculated checksum by the UART. The check passes when the result is all '1's, otherwise the check fails and the CERIF bit is set.

Two methods for computing the checksum are available: legacy and enhanced. The legacy checksum includes only the data bytes. The enhanced checksum includes the PID and the data. The C0EN control bit in the UxCON2 register determines the checksum method. Setting C0EN to '1' selects the enhanced method. Software must select the appropriate method before the Start bit of the checksum byte is received.

31.5.1 LIN MASTER/SLAVE MODE

The LIN Master mode includes capabilities to generate Slave processes. The Master process stops at the PID transmission. Any data that is transmitted in Master/Slave mode is done as a Slave process. LIN Master/Slave mode is configured by the following settings:

- MODE[3:0] = 1100
- TXEN = 1
- RXEN = 1
- UxBRGH:L = Value to achieve desired baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection
- C0EN = desired checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

Note: The TXEN bit must be set before the Master process is received and remain set while in LIN mode whether or not the slave process is a transmitter.

The Master process is started by writing the PID to the UxP1L register when UxP2 is '0' and the UART is idle. The UxTXIF will not be set in this case. Only the six Least Significant bits of UxP1L are used in the PID transmission.

The two Most Significant bits of the transmitted PID are PID parity bits. PID[6] is the exclusive-or of PID bits 0,1,2,and 4. PID[7] is the inverse of the exclusive-or of PID bits 1,3,4,and 5.

The UART calculates and inserts these bits in the serial stream.

Writing UxP1L automatically clears the UxTXCHK and UxRXCHK registers and generates the Break, delimiter bit, Sync character (55h), and PID transmission portion of the transaction. The data portion of the transaction that follows, if there is one, is a Slave process. See **Section 31.5.2 "LIN Slave Mode"** for more details of that process. The Master receives it's own PID when RXEN is set. Software performs the Slave process corresponding to the PID that was sent and received. Attempting to write UxP1L before an active master process is complete will not succeed. Instead, the TXWRE bit will be set.

31.5.2 LIN SLAVE MODE

LIN Slave mode is configured by the following settings:

- MODE[3:0] = 1011
- TXEN = 1
- RXEN = 1
- UxP2 = Number of data bytes to transmit
- UxP3 = Number of data bytes to receive
- UxBRGH:L = Value to achieve default baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection
- C0EN = desired checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

The Slave process starts upon detecting a Break on the RX pin. The Break clears the UxTXCHK, UxRXCHK, UxP2, and UxP3 registers. At the end of the Break, the auto-baud circuity is activated and the baud rate is automatically set using the Sync character following the Break. The character following the Sync character is received as the PID code and is saved in the receive FIFO. The UART computes the two PID parity bits from the six Least Significant bits of the PID. If either parity bit does not match the corresponding bit of the received PID code, the PERIF flag is set and saved at the same FIFO location as the PID code. The UxRXIF bit is set indicating that the PID is available.

Software retrieves the PID by reading the UxRXB register and determines the Slave process to execute from that. The checksum method, number of data bytes, and whether to send or receive data, is defined by software according to the PID code.

31.5.2.1 LIN Slave Receiver

When the Slave process is a receiver, the software performs the following tasks:

- UxP3 register is written with a value equal to the number of data bytes to receive.
- C0EN bit is set or cleared to select the appropriate checksum. This must be completed before the Start bit of the checksum byte is received.
- Each byte of the process response is read from UxRXB when UxRXIF is set.

The UART updates the checksum on each received byte. When the last data byte is received, the computed checksum total is stored in the UxRXCHK register. The next received byte is saved in the receive FIFO and added with the value in UxRXCHK. The result of this addition is not accessible. However, if the result is not all '1's, the CERIF bit in the UxERRIR is set. The CERIF flag persists until cleared by software. Software needs to read UxRXB to remove the checksum byte from the FIFO, but the byte can be discarded if not needed for any other purpose.

After the checksum is received, the UART ignores all activity on the RX pin until a Break starts the next transaction.

31.5.2.2 LIN Slave Transmitter

When the Slave process is a transmitter, software performs the following tasks in the order shown:

- UxP2 register is written with a value equal to the number of bytes to transmit. This will enable TXIF flag which is disabled when UxP2 is '0'.
- C0EN bit is set or cleared to select the appropriate checksum
- Inter-byte delay is performed
- Each byte of the process response is written to UxTXB when UxTXIF is set

The UART accumulates the checksum as each byte is written to UxTXB. After the last byte is written, the UART stores the calculated checksum in the UxTXCHK register and transmits the inverted result as the last byte in the response.

The TXIF flag is disabled when UxP2 bytes have been written. Any writes to UxTXB that exceed the UxP2 count will be ignored and set the TXWRE flag in the UxFIFO register.

© 2017-2020 Microchip Technology Inc.

31.6 DALI Mode (UART1 only)

DALI is a protocol used for intelligent lighting control for building automation. The protocol consists of Control Devices and Control Gear. A Control Device is an application controller that sends out commands to the light fixtures. The light fixture itself is termed as a Control Gear. The communication is done using Manchester encoding, which is performed by the UART hardware.

Manchester encoding consists of the clock and data in a single bit stream. A high-to-low or a low-to-high transition always occurs in the middle of the bit period and is not guaranteed to occur at the bit period boundaries. When the consecutive bits in the bit stream are of the same value (i.e., consecutive '1's or consecutive '0's), a transition occurs at the bit boundary. However, when the bit value changes, there is no transition at the bit boundary. According to the standard, a half-bit time is typically 416.7 µs long. A double half-bit time or a single bit is typically 833.3 µs.

The protocol is inherently half-duplex. Communication over the bus occurs in the form of forward and backward frames. Wait times between the frames are defined in the standard to prevent collision between the frames.

A Control Device transmission is termed as the forward frame. In the DALI 2.0 standard, a forward frame can be two or three bytes in length. The two-byte forward frame is used for communication between Control Device and Control Gear whereas the three-byte forward frame is used for communication between Control Devices on the bus. The first byte in the forward frame is the control byte and is followed by either one or two data bytes. The transaction begins when the Control Device starts a transmission. Unlike other protocols, each byte in the frame is transmitted MSB first. Typical frame timing is as shown in Figure 31-8.

During communication between two Control Devices, three bytes are required to be transmitted. In this case, the software must write the third byte to UxTXB as soon as UxTXIF goes True and before the output shifter becomes empty. This ensures that the three bytes of the forward frame are transmitted back-to-back without any interruption.

All Control Gear on the bus receive the forward frame. If the forward frame requires a reply to be sent, one of the Control Gear may respond with a single byte, called the backward frame. The 2.0 standard requires the Control Gear to begin transmission of the backward frame between 5.5 ms to 10.5 ms (~14 to 22 half-bit times) after reception of the forward frame. Once the backward frame is received by the Control Device, it is required to wait a minimum of 2.4 ms (~6 half-bit times). After this wait time, the Control Device is free to transmit another forward frame (see Figure 31-9).

A Start bit is used to indicate the start of the forward and backward frames. The receiver bit rate is determined by the BRG register. The low period of the Start bit is measured and is used as the timing reference for all data bits in the forward and backward frames. The ABDOVF bit is set if the Stat bit low period causes the measurement counter to overflow. All the bits following the Start bit are data bits. The bit stream terminates when no transition is detected in the middle of a bit period (see Figure 31-7).

Forward and backward frames are terminated by two Idle bit periods or Stop bits. Normally, these start in the first bit period of a byte. If both Stop bits are valid, the byte reception is terminated and the CERIF bit in UxERRIR register is set. This bit needs to be cleared in the software.

If either of the Stop bits is invalid, the frame is tagged as invalid by saving it as a null byte and setting the framing error in the receive FIFO.

A framing error also occurs when no transition is detected on the bus in the middle of a bit period when the byte reception is not complete. In such a scenario, the byte will be saved with the FERIF bit.

31.6.1 CONTROL DEVICE

Control Device mode is configured with the following settings:

- MODE = 0b1000
- TXEN = 1
- RXEN = 1
- UxP1 = Forward frames are held for transmission with this number of half-bit periods after the completion of a forward or backward frame.
- UxP2 = Forward/backward frame threshold delimiter. Any reception that starts this number of half bit periods after the completion of a forward or backward frame is detected as forward frame and sets the PERIF flag of the corresponding received byte.
- UxBRGH:L = Value to achieve 1200 baud rate
- TXPOL = appropriate polarity for interface circuit
- STP = 0b10 for two Stop bits
- CERIE = 1 to enable interrupt when STP bit is received (if applicable)
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1.

A forward frame is initiated by writing the control byte to the UxTXB register. After sending the control byte, each data byte must be written to the UxTXB register as soon as UxTXIF goes true. It is necessary to perform every write after UxTXIF goes true, to ensure that the transmit buffer is ready to accept the byte.

Each write must also occur before the TXMTIF bit goes true, to ensure that the bit stream of forward frame is generated without an interruption.

When TXMTIF goes true, indicating the transmit shift register has completed sending the last byte in the frame, the TX output is held in Idle state for the number of half-bit periods selected by the STP bits in the UxCON2 register and the CERIF bit in UxERRIR register is set. This bit needs to be cleared in the software.

After the last Stop bit, the TX output is held in Idle state for an additional wait time determined by the half-bit period count in the UxP1 register. For example, a 2450 µs delay (~6 half-bit times) requires a value of 6 in UxP1L.

Any writes to the UxTXB register that occur after TXMTIF goes true, but before the UxP1 wait time expires, are held and then transmitted immediately following the wait time. If a backward frame is received during the wait time, any bytes that may have been written to UxTXB will be transmitted after completion of the backward frame reception plus the UxP1 wait time.

The wait timer is reset by the backward frame and starts over immediately following the reception of the Stop bits of the backward frame. Data pending in the transmit shift register will be sent when the wait time elapses.

To replace or delete any pending forward frame data, the TXBE bit needs to be set to flush the shift register and transmit buffer. A new control byte can then be written to the UxTXB register. The control byte will be held in the buffer and sent at the beginning of the next forward frame following the UxP1 wait time.

In Control Device mode, PERIF is set when a forward frame is received. This helps the software to determine whether the received byte is part of a forward frame from a Control Device (either from the Control Device under consideration or from another Control Device on the bus) or a backward frame from a Control Gear.

31.6.2 CONTROL GEAR

The Control Gear mode is configured with the following settings:

- MODE = `0b1001`
- TXEN = `1`
- RXEN = `1`
- UxP1 = Back Frames are held for transmission this number of half-bit periods after the completion of a Forward Frame.
- UxP2 = Forward/Back Frame threshold delimiter. Idle periods more than this number of half-bit periods are detected as Forward Frames.
- UxBRGH:L = Value to achieve 1200 baud rate
- TXPOL = appropriate polarity for interface circuit
- RXPOL = same as TXPOL
- STP = `0b10` for two Stop bits
- CERIE = `1` to enable interrupt when STP bit is received (if applicable)
- RxyPPS = TX pin output code
- TX pin TRIS control = `0`
- RXPPS = RX pin selection code
- RX pin TRIS control = `1`
- Input pin ANSEL bit = `0`
- ON = `1`

The UART starts listening for a forward frame when the Control Gear mode is entered. Only the frames that follow an Idle period longer than UxP2 half-bit periods are detected as forward frames. Backward frames from other Control Gear are ignored. Only forward frames will be stored in UxRXB. This is necessary because a backward frame can be sent only as a response to a forward frame.

The forward frame is received one byte at a time in the receive FIFO and retrieved by reading the UxRXB register. At the end of the forward frame, when the stop bit is received, the CERIF bit in UxERRIR register is set. This bit needs to be cleared in the software. The end of the forward frame starts a timer to delay the backward frame response by wait time equal to the number of half-bit periods stored in UxP1.

The data received in the forward frame is processed by the application software. If the application decides to send a backward frame in response to the forward frame, the value of the backward frame is written to UxTXB. This value is held for transmission in the transmit shift register until the wait time expires and is then transmitted.

If the backward frame data is written to UxTXB after the wait time has expired, it is held in the UxTXB register until the end of the wait time following the next forward frame. The TXMTIF bit is false when the backward frame data is held in the transmit shift register. Receiving a UxRXIF interrupt before the TXMTIF goes true indicates that the backward frame write was too late and another forward frame was received before sending the backward frame. The pending backward frame has to be flushed by setting the TXBE bit, to prevent it from being sent after the next Forward Frame.

FIGURE 31-7: MANCHESTER TIMING

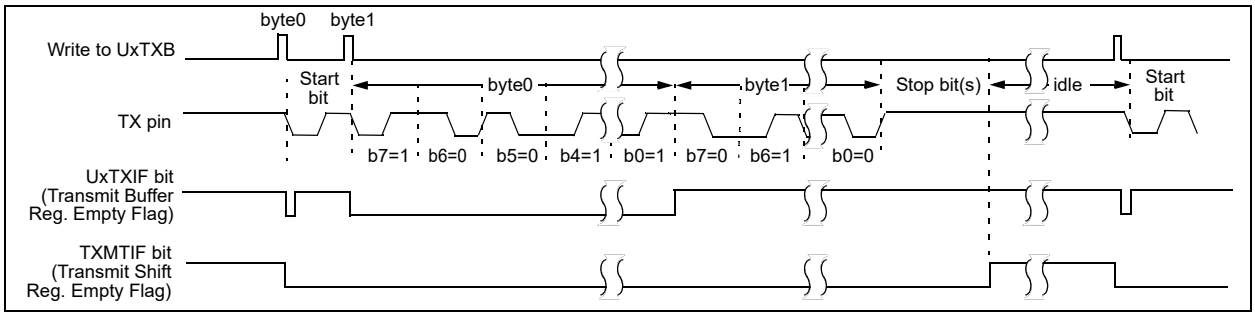


FIGURE 31-8: DALI FRAME TIMING

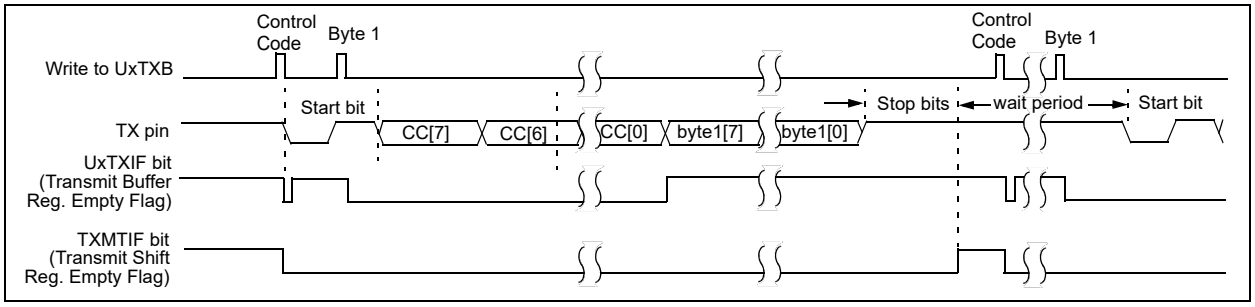
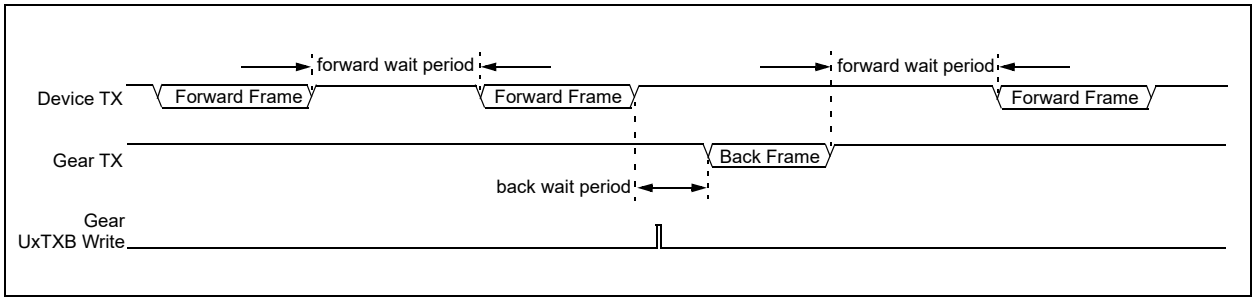


FIGURE 31-9: DALI FORWARD/BACK FRAME TIMING



31.7 General Purpose Manchester (UART1 only)

General purpose Manchester is a subset of the DALI mode. When the UxP1L register is cleared, there is no minimum wait time between frames. This allows full and half-duplex operation because writes to the UxTXB are not held waiting for a receive operation to complete.

General purpose Manchester operation maintains all other aspects of DALI mode such as:

- Single-pulse Start bit
- Most Significant bit first
- No stop periods between back-to-back bytes

General purpose Manchester mode is configured with the following settings:

- MODE[3:0] = `1000`
- TXEN = `1`
- RXEN = `1`
- UxP1 = 0h
- UxBRGH:L = desired baud rate
- TXPOL and RXPOL = desired Idle state
- STP = desired number of stop periods
- RxyPPS = TX pin selection code
- TX pin TRIS control = `0`
- RXPPS = RX pin selection code
- RX pin TRIS control = `1`
- Input pin ANSEL bit = `0`
- ON = `1`

The Manchester bit stream timing is shown in Figure 31-7.

31.8 Polarity

Receive and transmit polarity is user selectable and affects all modes of operation.

The idle level is programmable with the polarity control bits in the UxCON2 register. The control bits default to '0', which select a high idle level. The low level Idle state is selected by setting the control bit to '1'. TXPOL controls the TX idle level. RXPOL controls the RX idle level.

31.9 Stop Bits

The number of Stop bits is user selectable with the STP bits in the UxCON2 register.The STP bits affect all modes of operation.

Stop bits selections include:

- 1 transmit with receive verify on first
- 1.5 transmit with receive verify on first
- 2 transmit with receive verify on both
- 2 transmit with receive verify on first only

In all modes, except DALI, the transmitter is idle for the number of Stop bit periods between each consecutively transmitted word. In DALI, the Stop bits are generated after the last bit in the transmitted data stream.

The input is checked for the idle level in the middle of the first Stop bit, when receive verify on first is selected, as well as in the middle of the second Stop bit, when verify on both is selected. If any Stop bit verification indicates a non-idle level, the framing error FERIF bit is set for the received word.

31.9.1 DELAYED UXRXIF

When operating in Half-Duplex mode, where the microcontroller needs to reverse the transceiver direction after a reception, it may be more convenient to hold off the UxRXIF interrupt until the end of the Stop bits to avoid line contention. The user selects when the UxRXIF interrupt occurs with the STPMD bit in the UxFIFO register. When STPMD is '1', the UxRXIF occurs at the end of the last Stop bit. When STPMD is '0', UxRXIF occurs when the received byte is stored in the receive FIFO. When STP[1:0] = 10, the store operation is performed in the middle of the second Stop bit, otherwise, it is performed in the middle of the first Stop bit. The FERIF and PERIF interrupts are not delayed with STPMD. Only UxRXIF is delayed when STPMD is set and should be the only indicator for reversing transceiver direction.

31.10 Operation after FIFO overflow

The Receive Shift Register (RSR) can be configured to stop or continue running during a receive FIFO overflow condition. Stopped operation is the Legacy mode.

When the RSR continues to run during an overflow condition, the first word received after clearing the overflow will always be valid.

When the RSR is stopped during an overflow condition, synchronization with the Start bits is lost. Therefore, the first word received after the overflow is cleared may start in the middle of a word.

Operation during overflow is selected with the RUNOVF bit in the UxCON2 register. Setting the RUNOVF bit selects the run during overflow method.

31.11 Receive and Transmit Buffers

The UART uses small buffer areas to transmit and receive data. These are sometimes referred to as FIFOs.

The receiver has a Receive Shift Register (RSR) and two buffer registers. The buffer at the top of the FIFO (earliest byte to enter the FIFO) is by retrieved by reading the UxRXB register.

The transmitter has one Transmit Shift Register (TSR) and one buffer register. Writes to UxTXB go to the transmit buffer then immediately to the TSR, if it is empty. When the TSR is not empty, writes to UxTXB are held then transferred to the TSR when it becomes available.

31.11.1 FIFO STATUS

The UxFIFO register contains several status bits for determining the state of the receive and transmit buffers.

The RXBE bit indicates that the receive FIFO is empty. This bit is essentially the inverse of UxRXIF. The RXBF bit indicates that the receive FIFO is full.

The transmitter has only one buffer register so the status bits are essentially a copy and inverse of the UxTXIF bit. The TXBE bit indicates that the buffer is empty (same as UxTXIF) and the TXBF bit indicates that the buffer is full (UxTXIF inverse). A third transmitter status bit, TXWRE (transmit write error), is set whenever a UxTXB write is performed when the TXBF bit is set. This indicates that the write was unsuccessful.

31.11.2 FIFO RESET

All modes support resetting the receive and transmit buffers.

The receive buffer is flushed and all unread data discarded when the RXBE bit in the UxFIFO register is written to '1'. The MOVWF instruction with the TXBE bit cleared should be used to avoid inadvertently clearing a byte pending in the TSR when UxTXB is empty.

Data written to UxTXB when TXEN is low will be held in the Transmit Shift Register (TSR) then sent when TXEN is set. The transmit buffer and inactive TSR are flushed by setting the TXBE bit in the UxFIFO register. Setting TXBE while a character is actively transmitting from the TSR will complete the transmission without being flushed.

Clearing the ON bit will discard all received data and transmit data pending in the TSR and UxTXB.

31.12 Flow Control

This section does not apply to the LIN, DALI, or DMX modes.

Flow control is the means by which a sending UART data stream can be suspended by a receiving UART. Flow control prevents input buffers from overflowing without software intervention. The UART supports both hardware and XON/XOFF methods of flow control.

The flow control method is selected with the FLO[1:0] bits in the UxCON2 register. Flow control is disabled when are both bits are cleared.

31.12.1 HARDWARE FLOW CONTROL

Hardware flow control is selected by setting the FLO[1:0] bits to '10'.

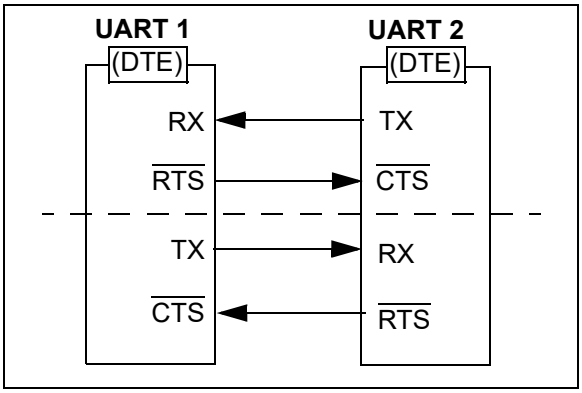
Hardware flow control consists of three lines. The RS-232 signal names for two of these are $\overline{RTS}$, and $\overline{CTS}$. Both are low true. The third line may be used to control an RS-485 transceiver. The signal name for this is TXDE for transmit drive enable. This output is high when the TX output is actively sending a character and low at all other times. The UART is configured as DTE (computer) equipment which means $\overline{RTS}$ is an output and $\overline{CTS}$ is an input.

The $\overline{RTS}$ and $\overline{CTS}$ signals work as a pair to control the transmission flow. A DTE-to-DTE configuration connects the $\overline{RTS}$ output of the receiving UART to the $\overline{CTS}$ input of the sending UART. Refer to Figure 31-10.

The UART receiving data asserts the $\overline{RTS}$ output low when the input FIFO is empty. When a character is received, the $\overline{RTS}$ output goes high until the UxRXB is read to free up both FIFO locations.

When the $\overline{CTS}$ input goes high after a byte has started to transmit, the transmission will complete normally. The receiver accommodates this by accepting the character in the second FIFO location even when the $\overline{CTS}$ input is high.

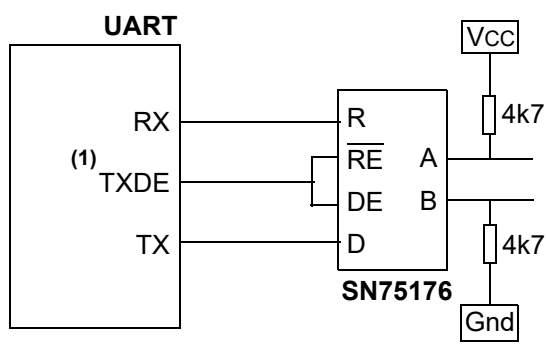
FIGURE 31-10: FLOW CONTROL



31.12.2 RS-485 TRANSCEIVER CONTROL

Hardware flow control can be used to control the direction of an RS-485 transceiver as shown in Figure 31-11. Configure the CTS input to be always enabled by setting the UxCTSPPS selection to an unimplemented port pin such as RD0. When the signal and control lines are configured as shown in Figure 31-11, then the UART will not receive its own transmissions. To verify that there are no collisions on the RS-485 lines then the transceiver RE control can be disconnected from TXDE and tied low thereby enabling loop-back reception of all transmissions. See **Section 31.14 "Collision Detection (UART1 Only)"** for more information.

FIGURE 31-11: RS-485 CONFIGURATION



Note 1: Configure UxCTSPPS to an unimplemented input such as RD0 (UxCTSPPS = 0x18).

31.12.3 XON/XOFF FLOW CONTROL

XON/XOFF flow control is selected by setting the FLO[1:0] bits to '01'.

XON/XOFF is a data based flow control method. The signals to suspend and resume transmission are special characters sent by the receiver to the transmitter The advantage is that additional hardware lines are not needed.

XON/XOFF flow control requires full duplex operation because the transmitter must be able to receive the signal to suspend transmitting while the transmission is in progress. Although XON and XOFF are not defined in the ASCII code, the generally accepted values are 13h for XOFF and 11h for XON. The UART uses those codes.

The transmitter defaults to XON, or transmitter enabled. This state is also indicated by the read-only XON bit in the UxFIFO register.

When an XOFF character is received, the transmitter stops transmitting after completing the character actively being transmitted. The transmitter remains disabled until an XON character is received.

XON will be forced on when software toggles the TXEN bit.

When the RUNOVF bit in the UxCON2 register is set then XON and XOFF characters continue to be received and processed without the need to clear the input FIFO by reading the UxRXB. However, if the RUNOVF bit is clear then the UxRXB must be read to avoid a receive overflow which will suspend flow control when the receive buffer overflows.

31.13 Checksum (UART1 only)

This section does not apply to the LIN mode, which handles checksums automatically.

The transmit and receive checksum adders are enabled when the C0EN bit in the UxCON2 register is set. When enabled, the adders accumulate every byte that is transmitted or received. The accumulated sum includes the carry of the addition. Software is responsible for clearing the checksum registers before a transaction and performing the check at the end of the transaction.

The following is an example of how the checksum registers could be used in the asynchronous modes.

31.13.1 TRANSMIT CHECKSUM METHOD

1. Clear the UxTXCHK register.
2. Set the C0EN bit.
3. Send all bytes of the transaction output.
4. Invert UxTXCHK and send the result as the last byte of the transaction.

31.13.2 RECEIVE CHECKSUM METHOD

1. Clear the UxRXCHK register.
2. Set the C0EN bit.
3. Receive all bytes in the transaction including the checksum byte.
4. Set MSb of UxRXCHK if 7-bit mode is selected.
5. Add 1 to UxRXCHK.
6. If the result is '0', the checksum passes, otherwise it fails.

31.14 Collision Detection (UART1 Only)

External forces that interfere with the transmit line are detected in all modes of operation with collision detection. Collision detection is always active when RXEN and TXEN are both set.

When the receive input is connected to the transmit output through either the same I/O pin or external circuitry, a character will be received for every character transmitted. The collision detection circuit provides a warning when the word received does not match the word transmitted.

The TXCIF flag in the UxERRIR register is used to signal collisions. This signal is only useful when the TX output is looped back to the RX input and everything that is transmitted is expected to be received. If more than one transmitter is active at the same time, it can be assumed that the TX word will not match the RX word. The TXCIF detects this mismatch and flags an interrupt. The TXCIF bit will also be set in DALI mode transmissions when the received bit is missing the expected mid-bit transition.

Collision detection is always active, regardless of whether or not the RX input is connected to the TX output. It is up to the user to disable the TXCIE bit when collision interrupts are not required.

The software overhead of unloading the receive buffer of transmitted data is avoided by setting the RUNOVF bit in UxCON2 and ignoring the receive interrupt and letting the receive buffer overflow. When the transmission is complete, prepare for receiving data by flushing the receive buffer (see Section 31.11.2, FIFO Reset) and clearing the RXFOIF overflow flag in the UxERRIR register.

31.15 RX/TX Activity Timeout

The UART works in conjunction with the HLT timers to monitor activity on the RX and TX lines. Use this feature to determine when there has been no activity on the receive or transmit lines for a user specified period of time.

To use this feature, set the HLT to the desired timeout period by a combination of the HLT clock source, timer prescale value, and timer period registers. Configure the HLT to reset on the UART TX or RX line and start the HLT at the same time the UART is started. UART activity will keep resetting the HLT to prevent a full HLT period from elapsing. When there has been no activity on the selected TX or RX line for longer than the HLT period then an HLT interrupt will occur signaling the timeout event.

For example, the following register settings will configure HLT2 for a 5 ms timeout of no activity on U1RX:

- T2PR = 0x9C (156 prescale periods)
- T2CLKCON = 0x05 (500 kHz internal oscillator)
- T2HLT = 0x04 (free running, reset on rising edge)
- T2RST = 0x15 (reset on U1RX)
- T2CON = 0xC0 (Timer2 on with 1:16 prescale)

31.16 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as VDD or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value of the OSCTUNE register allows for fine resolution changes to the system clock source. See **Section 7.2.2.3 "Internal Oscillator Frequency Adjustment"** for more information.

The other method adjusts the value of the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see **Section 31.17.1 "Auto-Baud Detect"**). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change of the peripheral clock frequency.

31.17 UART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is a 16-bit timer that is dedicated to the support of the UART operation.

The UxBRGH, UxBRGL register pair determines the period of the free running baud rate timer. The multiplier of the baud rate period is determined by the BRGS bit in the UxCON0 register.

Table 31-1 contains the formulas for determining the baud rate. Example 31-1 provides a sample calculation for determining the baud rate and baud rate error.

The high baud rate range (BRGS = 1) is intended to extend the baud rate range up to a faster rate when the desired baud rate is not possible otherwise. Using the normal baud rate range (BRGS = 0) is recommended when the desired baud rate is achievable with either range.

Writing a new value to the UxBRGH, UxBRGL 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 RXIDL bit to make sure that the receive operation is idle before changing the system clock.

EXAMPLE 31-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, BRGS = 0:

$$Desired\ Baud\ Rate = \frac{FOSC}{16([UxBRG] + 1)}$$

$$X = \frac{\frac{FOSC}{Desired\ Baud\ Rate}}{16} - 1$$

$$= \frac{\frac{16000000}{9600}}{16} - 1$$

$$= [103.17] = 103$$

$$Calculated\ Baud\ Rate = \frac{16000000}{16(103 + 1)}$$

$$= 9615$$

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

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

TABLE 31-1: BAUD RATE FORMULAS

BRGS	BRG/UART Mode	Baud Rate Formula
1	High Rate	FOSC/[4 (n+1)]
0	Normal Rate	FOSC/[16(n+1)]

Legend: n = value of UxBRGH, UxBRGL register pair.

31.17.1 AUTO-BAUD DETECT

The UART module supports automatic detection and calibration of the baud rate in the 8-bit asynchronous and LIN modes. However, setting ABDEN to start auto-baud detection is neither necessary, nor possible in LIN mode because that mode supports auto-baud detection automatically at the beginning of every data packet. Enabling auto-baud detect with the ABDEN bit applies to the asynchronous modes only.

> **Note:** In DALI Mode, ABDEN is ignored. The baud rate needs to be manually set to 1200 using the BRG registers.

When Auto-Baud Detect (ABD) is active, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII “U”), which is the Sync character for the LIN bus. The unique feature of this character is that it has five falling edges, including the Start bit edge, five rising edges including the Stop bit edge.

In 8-bit Asynchronous mode, setting the ABDEN bit in the UxCON0 register enables the auto-baud calibration sequence. The first falling edge of the RX input after ABDEN is set will start the auto-baud calibration sequence. While the ABD sequence takes place, the UART state machine is held in idle. On the first falling edge of the receive line, the UxBRG begins counting up using the BRG counter clock as shown in Figure 31-12. The fifth falling edge will occur on the RX pin at the beginning of the bit 7 period. At that time, an accumulated value totaling the proper BRG period is left in the UxBRGH, UxBRGL register pair, the ABDEN bit is automatically cleared and the ABDIF interrupt flag is set. ABDIF must be cleared by software.

RXIDL indicates that the sync input is active. RXIDL will go low on the first falling edge and go high on the fifth rising edge.

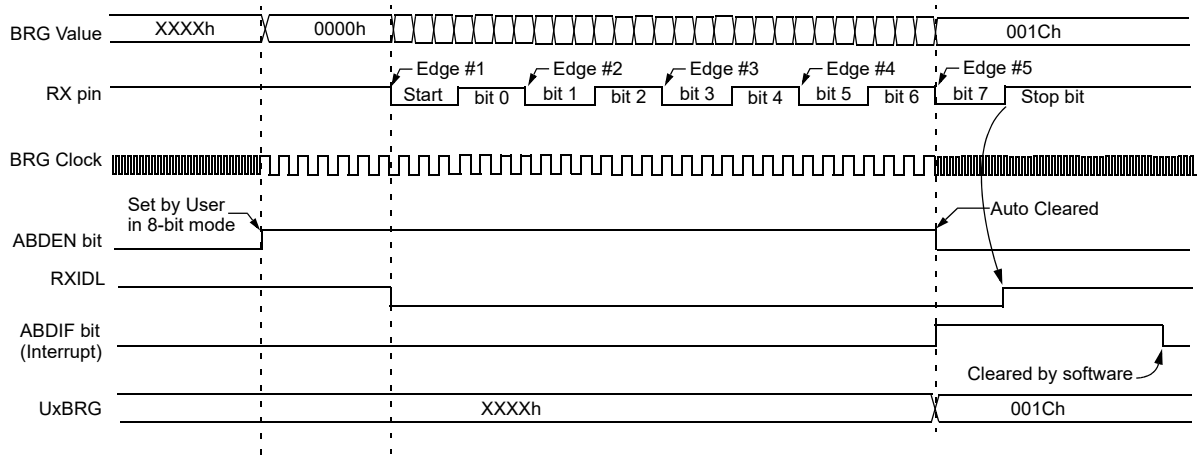
The BRG auto-baud clock is determined by the BRGS bit as shown in Table 31-2. During ABD, the internal BRG register is used as a 16-bit counter. However, the UxBRGH and UxBRGL registers retain the previous BRG value until the auto-baud process is successfully completed. While calibrating the baud rate period, the internal BRG register is clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed and is transferred to the UxBRGH and UxBRGL registers when complete.

> **Note 1:** If the WUE bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see **Section 31.17.3 “Auto-Wake-up on Break”**).
>
> **2:** It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and UART baud rates are not possible.

TABLE 31-2: BRG COUNTER CLOCK RATES

BRGS	BRG Base Clock	BRG ABD Clock
1	FOSC/4	FOSC/32
0	FOSC/16	FOSC/128

FIGURE 31-12: AUTOMATIC BAUD RATE CALIBRATION



Note 1: Auto-baud is supported in LIN and 8-bit asynchronous modes only.

31.17.2 AUTO-BAUD OVERFLOW

During the course of automatic baud detection, the ABDOVF bit in the UxERRIR register will be set if the baud rate counter overflows before the fifth falling edge is detected on the RX pin. The ABDOVF bit indicates that the counter has exceeded the maximum count that can fit in the 16 bits of the UxBRGH:UxBRGL register pair. After the ABDOVF bit has been set, the state machine continues to search until the fifth falling edge is detected on the RX pin. Upon detecting the fifth falling RX edge, the hardware will set the ABDIF interrupt flag and clear the ABDEN bit in the UxCON0 register. The UxBRGH and UxBRGL register values retain their previous value. The ABDIF flag in the UxUIR register and ABDOVF flag in the UxERRIR register can be cleared by software directly. To generate an interrupt on an auto-baud overflow condition, all the following bits must be set:

- ABDOVE bit in the UxERRIE register
- UxEIE bit in the PIEx register
- PIE and GIE bits in the INTCON register

To terminate the auto-baud process before the ABDIF flag is set, clear the ABDEN bit, then clear the ABDOVF bit in the UxERRIR register.

31.17.3 AUTO-WAKE-UP ON BREAK

During Sleep mode, all clocks to the UART are suspended. Because of this, the Baud Rate Generator is inactive and a proper character reception cannot be performed. The Auto-Wake-up feature allows the controller to wake-up due to activity on the RX line.

The Auto-Wake-up feature is enabled by setting both the WUE bit in the UxCON1 register and the UxIE bit in the PIEx register. Once set, the normal receive sequence on RX is disabled, and the UART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a transition out of the Idle state on the RX line. (This coincides with the start of a Break or a wake-up signal character for the LIN protocol.)

The UART module generates a WUIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes (Figure 31-13), and asynchronously, if the device is in Sleep mode (Figure 31-14). The interrupt condition is cleared by clearing the WUIF bit in the UxUIR register. To generate an interrupt on a wake-up event, all the following bits must be set:

- UxIE bit in the PIEx register
- PIE and GIE bits in the INTCON register

The WUE bit is automatically cleared by the transition to the Idle state on the RX line at the end of the Break. This signals to the user that the Break event is over. At this point, the UART module is in Idle mode, waiting to receive the next character.

31.17.3.1 Special Considerations

Break Character

To avoid character errors or character fragments during a wake-up event, the wake-up character must be all zeros.

When the wake-up is enabled, the function works independent of the low time on the data stream. If the WUE bit is set and a valid non-zero character is received, the low time from the Start bit to the first rising edge will be interpreted as the wake-up event. The remaining bits of the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

Therefore, the initial character of the transmission must be all zeros. This must be eleven or more bit times, 13-bit times recommended for LIN bus, or any number of bit times for standard RS-232 devices.

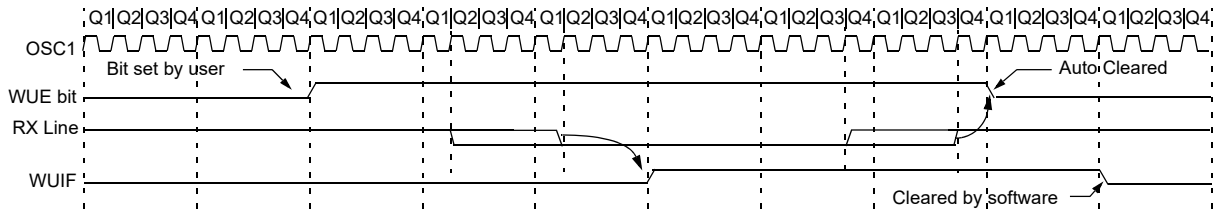
Oscillator Start-up Time

Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL modes). 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 UART.

WUE Bit

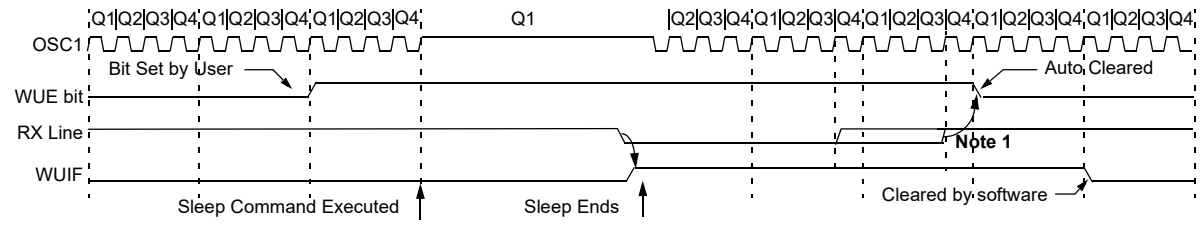
To ensure that no actual data is lost, check the RXIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

FIGURE 31-13: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION



Note 1: The **UART** remains in idle while the WUE bit is set.

FIGURE 31-14: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



Note 1: If the wake-up event requires long oscillator warm-up time, the automatic clearing 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 **UART** remains in idle while the WUE bit is set.

31.18 Transmitting a Break

The UART module has the capability of sending either a fixed length Break period or a software timed Break period. The fixed length Break consists of a Start bit, followed by 12 ‘`0`’ bits and a Stop bit. The software timed Break is generated by setting and clearing the BRKOVR bit in the UxCON1 register.

To send the fixed length Break, set the SENDB and TXEN bits in the UxCON0 register. The Break sequence is then initiated by a write to UxTXB. The timed Break will occur first, followed by the character written to UxTXB that initiated the Break. The initiating character is typically the Sync character of the LIN specification.

SENB is disabled in the LIN and DMX modes because those modes generate the Break sequence automatically.

The SENDB bit is automatically reset by hardware after the Break Stop bit is complete.

The TXMTIF bit in the UxERRIR register indicates when the transmit operation is active or idle, just as it does during normal transmission. See Figure 31-15 for the timing of the Break sequence.

31.19 Receiving a Break

The UART has counters to detect when the RX input remains in the space state for an extended period of time. When this happens, the RXBKIF bit in the UxERRIR register is set.

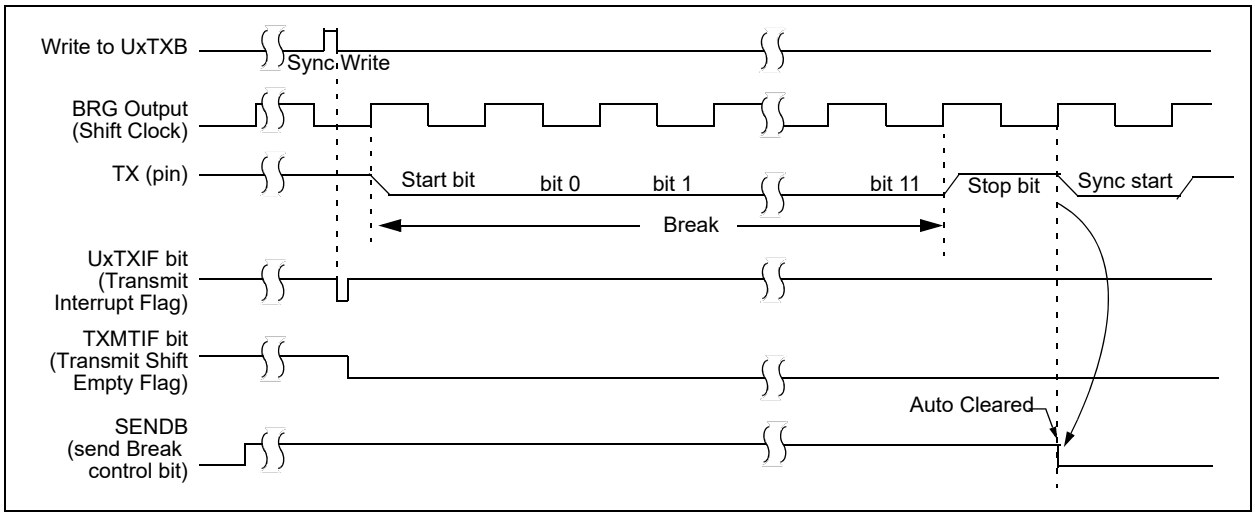
A Break is detected when the RX input remains in the space state for 11 bit periods for asynchronous and LIN modes, and 23 bit periods for DMX mode.

The user can select to receive the Break interrupt as soon as the Break is detected or at the end of the Break, when the RX input returns to the Idle state. When the RXBIMD bit in the UxCON1 is ‘`1`’ then RXBKIF is set immediately upon Break detection. When RXBIMD is ‘`0`’ then RXBKIF is set when the RX input returns to the Idle state.

31.20 UART Operation During Sleep

The UART ceases to operate during Sleep. The safe way to wake the device from Sleep by a serial operation is to use the Wake-on-Break feature of the UART. See Section 31.17.3, Auto-Wake-up on Break

FIGURE 31-15: SEND BREAK CHARACTER SEQUENCE



31.21 Register Definitions: UART Control

Long bit name prefixes for the UART peripherals are shown below. Refer to **Section 1.3 "Register and Bit naming conventions"** for more information.

Peripheral	Bit Name Prefix
UART 1	U1
UART 2	U2

REGISTER 31-1: UxCON0: UART CONTROL REGISTER 0

R/W-0/0	R/W/HS/HC-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
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	HC = Hardware clear

bit 7 **BRGS:** Baud rate Generator Speed Select bit
1 = Baud rate generator is high speed with 4 baud clocks per bit
0 = Baud rate generator is normal speed with 16 baud clocks per bit

bit 6 **ABDEN:** Auto-baud Detect Enable bit(3)
1 = Auto-baud is enabled. Receiver is waiting for Sync character (0x55)
0 = Auto-baud is not enabled or auto-baud is complete

bit 5 **TXEN:** Transmit Enable Control bit(2)
1 = Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is idle.
0 = Transmit is disabled. TX output pin drive is controlled by PORT TRIS control

bit 4 **RXEN:** Receive Enable Control bit(2)
1 = Receiver is enabled
0 = Receiver is disabled

bit 3-0 **MODE[3:0]:** UART Mode Select bits(1)
1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = LIN Master/Slave mode(4)
1011 = LIN Slave-Only mode(4)
1010 = DMX mode(4)
1001 = DALI Control Gear mode(4)
1000 = DALI Control Device mode(4)
0111 = Reserved
0110 = Reserved
0101 = Reserved
0100 = Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data
0011 = Asynchronous 8-bit UART mode with 9th bit even parity
0010 = Asynchronous 8-bit UART mode with 9th bit odd parity
0001 = Asynchronous 7-bit UART mode
0000 = Asynchronous 8-bit UART mode

Note 1: Changing the UART MODE while ON = 1 may cause unexpected results.
2: Clearing TXEN or RXEN will not clear the corresponding buffers. Use TXBE or RXBE to clear the buffers.
3: When MODE = 100x, then ABDEN bit is ignored.
4: UART1 only.

REGISTER 31-2: UxCON1: UART CONTROL REGISTER 1

R/W-0/0	U-0	U-0	R/W/HC-0/0	R/W-0/0	U-0	R/W-0/0	R/W/HC-0/0
ON	—	—	WUE	RXBIMD	—	BRKOVR	SENDB
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	HC = Hardware clear

bit 7 **ON:** Serial Port Enable bit
1 = Serial port enabled
0 = Serial port disabled (held in Reset)

bit 6-5 **Unimplemented:** Read as ‘0’

bit 4 **WUE:** Wake-up Enable bit
1 = Receiver is waiting for falling RX input edge which will set the UxIF bit. Cleared by hardware on wake event. Also requires UxIE bit of PIEx to enable wake
0 = Receiver operates normally

bit 3 **RXBIMD:** Receive Break Interrupt Mode Select bit
1 = Set RXBKIF immediately when RX in has been low for the minimum Break time
0 = Set RXBKIF on rising RX input after RX in has been low for the minimum Break time

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **BRKOVR:** Send Break Software Override bit
1 = TX output is forced to non-idle state
0 = TX output is driven by transmit shift register

bit 0 **SENDB:** Send Break Control bit(1)
1 = Output Break upon UxTXB write. Written byte follows Break. Bit is cleared by hardware.
0 = Break transmission completed or disabled

Note 1: This bit is read-only in LIN, DMX, and DALI modes.

REGISTER 31-3: UxCON2: UART CONTROL REGISTER 2

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
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 **RUNOVF:** Run During Overflow Control bit
1 = RX input shifter continues to synchronize with Start bits after overflow condition
0 = RX input shifter stops all activity on receiver overflow condition

bit 6 **RXPOL**: Receive Polarity Control bit
1 = Invert RX polarity, Idle state is low
0 = RX polarity is not inverted, Idle state is high

bit 5-4 **STP[1:0]:** Stop Bit Mode Control bits(1)
11 = Transmit 2 Stop bits, receiver verifies first Stop bit
10 = Transmit 2 Stop bits, receiver verifies first and second Stop bits
01 = Transmit 1.5 Stop bits, receiver verifies first Stop bit
00 = Transmit 1 Stop bit, receiver verifies first Stop bit

bit 3 **C0EN**: Checksum Mode Select bit(2)
LIN mode:
1 = Checksum Mode 1, enhanced LIN checksum includes PID in sum
0 = Checksum Mode 0, legacy LIN checksum does not include PID in sum
Other modes:
1 = Add all TX and RX characters
0 = Checksums disabled

bit 2 **TXPOL:** Transmit Polarity Control bit
1 = Output data is inverted, TX output is low in Idle state
0 = Output data is not inverted, TX output is high in Idle state

bit 1-0 **FLO[1:0]:** Handshake Flow Control bits
11 = Reserved
10 = RTS/CTS and TXDE Hardware flow control
01 = XON/XOFF Software flow control
00 = Flow control is off

Note 1: All modes transmit selected number of Stop bits. Only DMX and DALI receivers verify selected number of Stop bits and all others verify only the first Stop bit.
2: UART1 only.

REGISTER 31-4: UxERRIR: UART ERROR INTERRUPT FLAG REGISTER

R/S/C-1/1	R/S/C-0/0	R/W/S-0/0	R/W/S-0/0	R/S/C-0/0	R/W/S-0/0	R/W/S-0/0	R/W/S-0/0
TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
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	S = Hardware set	C = Hardware clear

bit 7 **TXMTIF:** Transmit Shift Register Empty Interrupt Flag bit
1 = Transmit shift register is empty (Set at end of Stop bits)
0 = Transmit shift register is actively shifting data

bit 6 **PERIF**: Parity Error Interrupt Flag bit
LIN and Parity modes:
1 = Unread byte at top of input FIFO has parity error
0 = Unread byte at top of input FIFO does not have parity error
DALI Device mode:
1 = Unread byte at top of input FIFO received as Forward Frame
0 = Unread byte at top of input FIFO received as Back Frame
Address mode:
1 = Unread byte at top of input FIFO received as address
0 = Unread byte at top of input FIFO received as data
Other modes:
Not used

bit 5 **ABDOVF:** Auto-baud Detect Overflow Interrupt Flag bit
DALI mode:
1 = Start bit measurement overflowed counter
0 = No overflow during Start bit measurement
Other modes:
1 = Baud rate generator overflowed during the auto detection sequence
0 = Baud rate generator has not overflowed

bit 4 **CERIF**: Checksum Error/DALI STP bit Interrupt Flag bit
DALI modes:
1 = Stop bit detected
0 = Stop bit not detected
LIN Mode:
1 = Checksum error
0 = No Checksum error

bit 3 **FERIF:** Framing Error Interrupt Flag bit
1 = Unread byte at top of input FIFO has framing error
0 = Unread byte at top of input FIFO does not have framing error

bit 2 **RXBKIF:** Break Reception Interrupt Flag bit
1 = Break detected
0 = No Break detected

bit 1 **RXFOIF:** Receive FIFO Overflow Interrupt Flag bit
1 = Receive FIFO has overflowed
0 = Receive FIFO has not overflowed

bit 0 **TXCIF:** Transmit Collision Interrupt Flag bit(1)
1 = Transmitted word is not equal to the word received during transmission
0 = Transmitted word equals the word received during transmission

Note 1: UART1 only.

REGISTER 31-5: UxERRIE: UART ERROR INTERRUPT ENABLE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TXMTIE	PERIE	ABDOVE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
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 **TXMTIE:** Transmit Shift Register Empty Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 6 **PERIE**: Parity Error Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 5 **ABDOVE:** Auto-baud Detect Overflow Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 4 **CERIE**: Checksum Error/DALI STP bit Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 3 **FERIE:** Framing Error Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 2 **RXBKIE:** Break Reception Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 1 **RXFOIE:** Receive FIFO Overflow Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 0 **TXCIE:** Transmit Collision Interrupt Enable bit(1)
1 = Interrupt enabled
0 = Interrupt not enabled

Note 1: UART1 only.

REGISTER 31-6: UxUIR: UART GENERAL INTERRUPT REGISTER

R/S/W-0/0	R/S/W-0/0	U-0	U-0	U-0	R/W-0/0	U-0	U-0
WUIF	ABDIF	—	—	—	ABDIE	—	—
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	S = Hardware set

bit 7 **WUIF:** Wake-up Interrupt bit
- 1 = Idle to non-idle transition on RX line detected when WUE is set. Also sets UxIF. (WUIF must be cleared by software to clear UxIF)
- 0 = WUE not enabled by software or no transition detected

bit 6 **ABDIF**: Auto-baud detect interrupt bit
- 1 = Auto-baud detection complete. Status shown in UxIF when ABDIE is set. (Must be cleared by software)
- 0 = Auto-baud not enabled or auto-baud enabled and auto-baud detection not complete

bit 5-3 **Unimplemented:** Read as ‘0’

bit 2 **ABDIE:** Auto-baud Detect Interrupt Enable bit
- 1 = ABDIF will set UxIF bit in PIRx register
- 0 = ABDIF will not set UxIF

bit 1-0 **Unimplemented:** Read as ‘0’

REGISTER 31-7: UxFIFO: UART FIFO STATUS REGISTER

R/W/S-0/0	R/W-0/0	R/W/S/C-1/1	R/S/C-0/0	R/S/C-1/1	S/C-1/1	R/W/S/C-1/1	R/S/C-0/0
TXWRE	STPMD	TXBE	TXBF	RXIDL	XON	RXBE	RXBF
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	S = Hardware set	C = Hardware clear

bit 7 **TXWRE:** Transmit Write Error Status bit (Must be cleared by software)

LIN Master mode:
1 = UxP1L was written when a master process was active

LIN Slave mode:
1 = UxTXB was written when UxP2 = 0 or more than UxP2 bytes have been written to UxTXB since last Break

Address Detect mode:
1 = UxP1L was written before the previous data in UxP1L was transferred to TX shifter

All modes:
1 = A new byte was written to UxTXB when the output FIFO was full
0 = No error

bit 6 **STPMD**: Stop Bit Detection Mode bit
1 = Assert UxRXIF at end of last Stop bit or end of first Stop bit when STP = 11
0 = Assert UxRXIF in middle of first Stop bit

bit 5 **TXBE:** Transmit Buffer Empty Status bit
1 = Transmit buffer is empty. Setting this bit will clear the transmit buffer and output shift register.
0 = Transmit buffer is not empty. Software cannot clear this bit.

bit 4 **TXBF**: Transmit Buffer Full Status bit
1 = Transmit buffer is full
0 = Transmit buffer is not full

bit 3 **RXIDL:** Receive Pin Idle Status bit
1 = Receive pin is in Idle state
0 = UART is receiving Start, Stop, Data, Auto-baud, or Break

bit 2 **XON:** Software Flow Control Transmit Enable Status bit
1 = Transmitter is enabled
0 = Transmitter is disabled

bit 1 **RXBE:** Receive Buffer Empty Status bit
1 = Receive buffer is empty. Setting this bit will clear the RX buffer(1)
0 = Receive buffer is not empty. Software cannot clear this bit.

bit 0 **RXBF:** Receive Buffer Full Status bit
1 = Receive buffer is full
0 = Receive buffer is not full

Note 1: The BSF instruction should not be used to set RXBE because doing so will clear a byte pending in the transmit shift register when the UxTXB register is empty. Instead, use the MOVWF instruction with a ‘0’ in the TXBE bit location.

REGISTER 31-8: UxBRGL: UART BAUD RATE GENERATOR LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BRG[7:0]							
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 7-0 **BRG[7:0]:** Least Significant Byte of Baud Rate Generator

REGISTER 31-9: UxBRGH: UART BAUD RATE GENERATOR HIGH REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BRG[15:8]							
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 7-0 **BRG[15:8]:** Most Significant Byte of Baud Rate Generator

Note 1: The UxBRG registers should only be written when ON = `0`.

2: Maximum BRG value when MODE = ‘`100x`’ and BRGS = `1` is 0x7FFE.

3: Maximum BRG value when MODE = ‘`100x`’ and BRGS = `0` is 0x1FFE.

REGISTER 31-10: UxRXB: UART RECEIVE REGISTER

R-x/u	R-x/u	R-x/u	R-x/u	R-x/u	R-x/u	R-x/u	R-x/u
RXB[7:0]							
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 7-0 **RXB[7:0]:** Top of Receive Buffer

REGISTER 31-11: UxTXB: UART TRANSMIT REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TXB[7:0]							
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 7-0 **TXB[7:0]:** Bottom of Transmit Buffer

REGISTER 31-12: UxP1H: UART PARAMETER 1 HIGH REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	P1[8]
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 **Unimplemented:** Read as ‘0’

bit 0 **P1[8]:** Most Significant Bit of Parameter 1

DMX mode:
Most Significant bit of number of bytes to transmit between Start Code and automatic Break generation

DALI Control Device mode:
Most Significant bit of idle time delay after which a Forward Frame is sent. Measured in half-bit periods

DALI Control Gear mode:
Most Significant bit of delay between the end of a Forward Frame and the start of the Back Frame Measured in half-bit periods

Other modes:
Not used

REGISTER 31-13: UxP1L: UART PARAMETER 1 LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
P1[7:0]							
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 7-0 **P1[7:0]:** Least Significant Bits of Parameter 1

DMX mode:
Least Significant Byte of number of bytes to transmit between Start Code and automatic Break generation

DALI Control Device mode:
Least Significant Byte of idle time delay after which a Forward Frame is sent. Measured in half-bit periods

DALI Control Gear mode:
Least Significant Byte of delay between the end of a Forward Frame and the start of the Back Frame Measured in half-bit periods

LIN mode:
PID to transmit (Only Least Significant 6 bits used)

Asynchronous Address mode:
Address to transmit (9th transmit bit automatically set to ‘1’)

Other modes:
Not used

REGISTER 31-14: UxP2H: UART PARAMETER 2 HIGH REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	P2[8]
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 **Unimplemented:** Read as ‘0’

bit 0 **P2[8]:** Most Significant Bit of Parameter 2
DMX mode:
Most Significant bit of first address of receive block
DALI mode:
Most Significant bit of number of half-bit periods of idle time in Forward Frame detection threshold
Other modes:
Not used

REGISTER 31-15: UxP2L: UART PARAMETER 2 LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
P2[7:0]							
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 7-0 **P2[7:0]:** Least Significant Bits of Parameter 2
DMX mode:
Least Significant Byte of first address of receive block
LIN Slave mode:
Number of data bytes to transmit
DALI mode:
Least Significant Byte of number of half-bit periods of idle time in Forward Frame detection threshold
Asynchronous Address mode:
Receiver address
Other modes:
Not used

REGISTER 31-16: UxP3H: UART PARAMETER 3 HIGH REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	P3[8]
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 **Unimplemented:** Read as ‘0’

bit 0 **P3[8]:** Most Significant Bit of Parameter 3
DMX mode:
Most Significant bit of last address of receive block
Other modes:
Not used

REGISTER 31-17: UxP3L: UART PARAMETER 3 LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
P3[7:0]							
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 7-0 **P3[7:0]:** Least Significant Bits of Parameter 3
DMX mode:
Least Significant Byte of last address of receive block
LIN Slave mode:
Number of data bytes to receive
Asynchronous Address mode:
Receiver address mask. Received address is XOR’d with UxP2L then AND’d with UxP3L
Match occurs when result is zero
Other modes:
Not used

REGISTER 31-18: UxTXCHK: UART TRANSMIT CHECKSUM RESULT REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TXCHK[7:0]							
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 7-0 **TXCHK[7:0]:** Checksum calculated from TX bytes
LIN mode and C0EN = 1:
Sum of all transmitted bytes including PID
LIN mode and C0EN = 0:
Sum of all transmitted bytes except PID
All other modes and C0EN = 1:
Sum of all transmitted bytes since last clear
All other modes and C0EN = 0:
Not used

REGISTER 31-19: UxRXCHK: UART RECEIVE CHECKSUM RESULT REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
RXCHK[7:0]							
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 7-0 **RXCHK[7:0]:** Checksum calculated from RX bytes
LIN mode and C0EN = 1:
Sum of all received bytes including PID
LIN mode and C0EN = 0:
Sum of all received bytes except PID
All other modes and C0EN = 1:
Sum of all received bytes since last clear
All other modes and C0EN = 0:
Not used

TABLE 31-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE UART

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
UxCON0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]				499
UxCON1	ON	—	—	WUE	RXBIMD	—	BRKOVR	SENDB	500
UxCON2	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]		501
UxERRIR	TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF	502
UxERRIE	TXMTIE	PERIE	ABDOVE	CERIE	FERIE;	RXBKIE	RXFOIE	TXCIE	503
UxUIR	WUIF	ABDIF	—	—	—	ABDIE	—	—	504
UxFIFO	TXWRE	STPMD	TXBE	TXBF	RXIDL	XON	RXBE	RXBF	505
UxBRGL	BRG[7:0]								506
UxBRGH	BRG[15:8]								506
UxRXB	RXB[7:0]								507
UxTXB	TXB[7:0]								507
UxP1H	—	—	—	—	—	—	—	P1[8]	508
UxP1L	P1[7:0]								508
UxP2H	—	—	—	—	—	—	—	P2[8]	509
UxP2L	P2[7:0]								509
UxP3H	—	—	—	—	—	—	—	P3[8]	510
UxP3L	P3[7:0]								510
UxTXCHK	TXCHK[7:0]								511
UxRXCHK	RXCHK[7:0]								511

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

32.0 SERIAL PERIPHERAL INTERFACE (SPI) MODULE

32.1 SPI Module Overview

The SPI (Serial Peripheral Interface) module is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select. Example slave devices include serial EEPROMs, shift registers, display drivers, A/D converters, or another PIC® device.

The SPI bus specifies four signal connections:

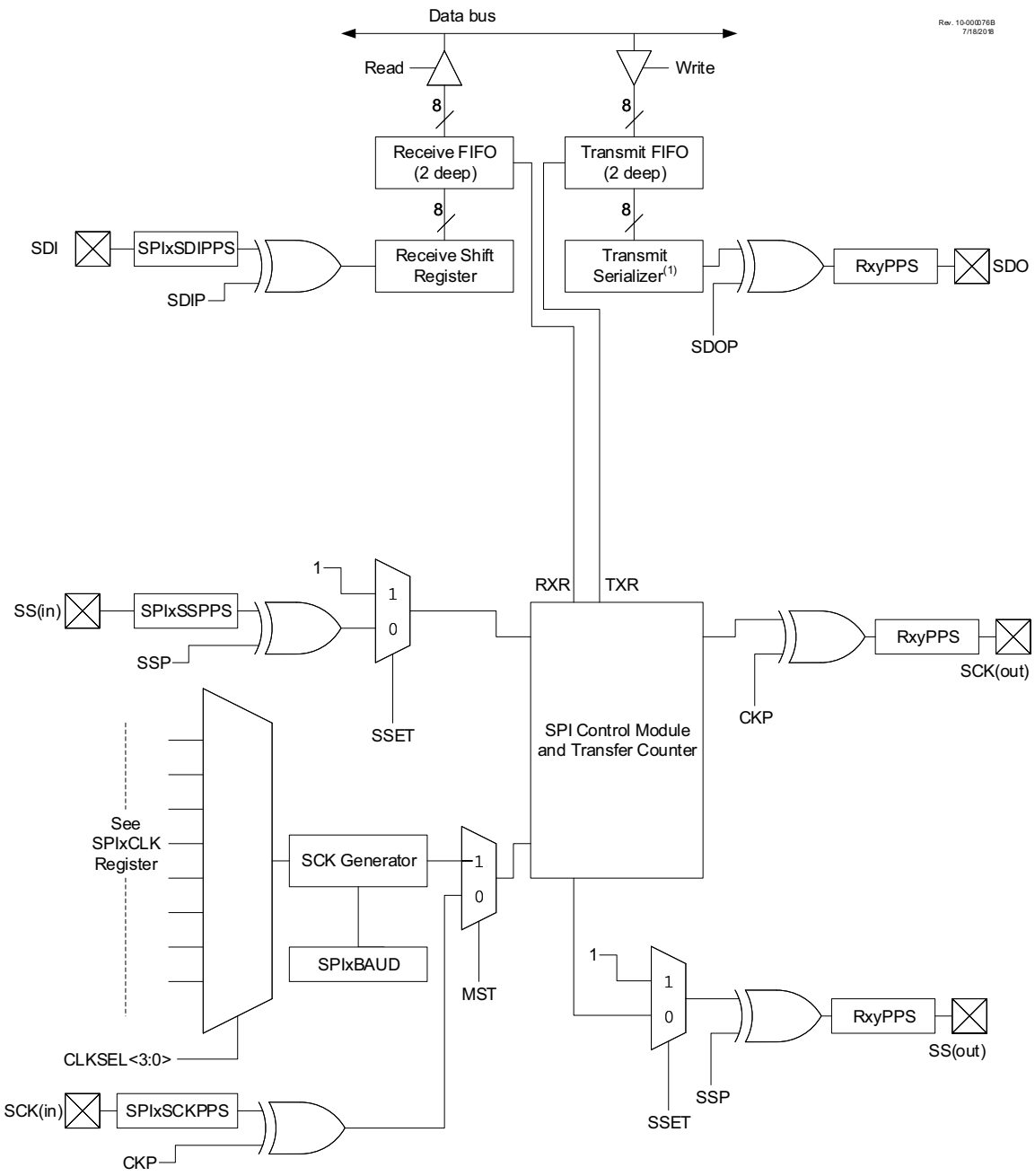
- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data IN (SDI)
- Slave Select (SS)

The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Polarity and Edge Select
- SDI, SDO, and SS Polarity Control
- Separate Transmit and Receive Enables
- Slave Select Synchronization
- Daisy-chain connection of slave devices
- Separate Transmit and Receive Buffers with 2-byte FIFO and DMA capabilities

Figure 32-1 shows the block diagram of the SPI module.

FIGURE 32-1: SPI MODULE SIMPLIFIED BLOCK DIAGRAM



Note 1: If TXR=1 and the transmit FIFO is empty, the previous value of the receive shift register will be sent to the transmit serializer.

The SPI transmit output (SDO_out) is available to the remappable PPS SDO pin and internally to the following peripherals:

- Configurable Logic Cell (CLC)
- Data Signal Modulator (DSM)

The SPI bus typically operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions typically involve shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new bit is shifted into the device. Unlike older Microchip devices, the SPI on the this device contains two separate registers for incoming and outgoing data. Both registers also have 2-byte FIFO buffers and allow for DMA bus connections.

Figure 32-2 shows a typical connection between two devices configured as master and slave devices.

Data is shifted out of the transmit FIFO on the programmed clock edge and into the receive shift register on the opposite edge of the clock.

The master device transmits information on its SDO output pin which is connected to, and received by, the slave's SDI input pin. The slave device transmits information on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

The master device sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its output register (on its SDO pin) and the slave device is reading this bit and saving as the LSb of its input register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its input register.

After eight bits have been shifted out, the master and slave have exchanged register values and stored the incoming data into the receiver FIFOs.

If there is more data to exchange, the registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data) depends on the application software. This leads to three scenarios for data transmission:

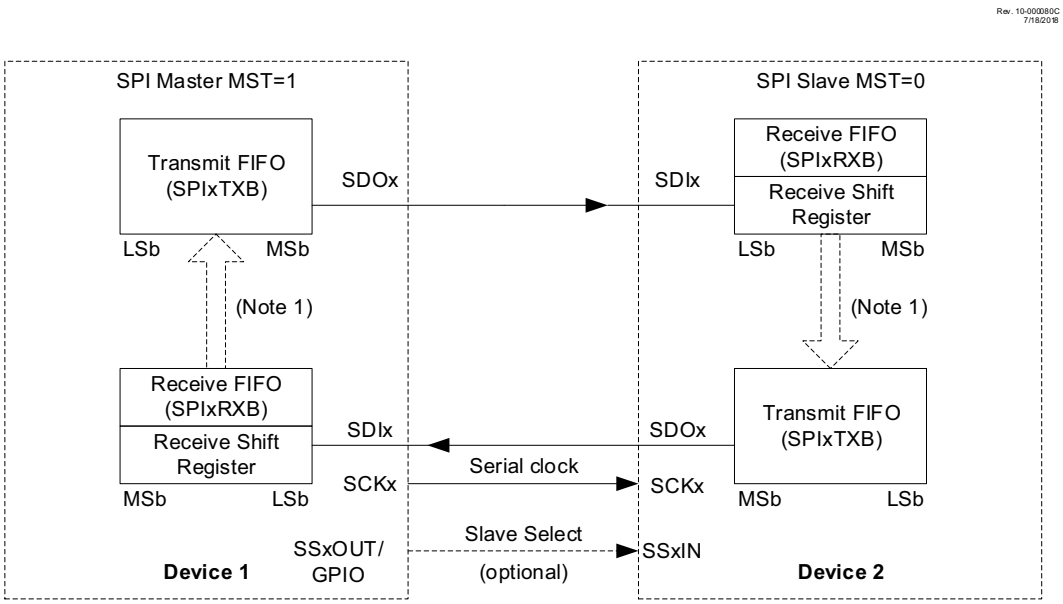
- Master sends useful data and slave sends dummy data
- Master sends useful data and slave sends useful data
- Master sends dummy data and slave sends useful data

In this particular SPI module, dummy data may be sent without software involvement, by clearing either the RXR bit (for receiving dummy data) or the TXR bit (for sending dummy data) (see Table 32-1 as well as **Section 32.5 "Master mode"** and **Section 32.6 "Slave Mode"** for further TXR/RXR setting details). This SPI module can send transmissions of any number of bits, and can send information in segments of varying size (from 1-8 bits in width). As such, transmissions may involve any number of clock cycles, depending on the amount of data to be transmitted.

When there is no more data to be transmitted, the master stops sending the clock signal and deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line disregards the clock and transmission signals and does not transmit out any data of its own.

FIGURE 32-2: SPI MASTER/SLAVE CONNECTION WITH FIFOs



Note 1: In some modes, if the Transmit FIFO is empty, the most recently received byte of data will be transmitted

2: This diagram assumes that the LSBF bit is cleared (communications are MSb-first). If LSBF is set, the communications will be LSb-first.

32.2 SPI REGISTERS

- SPI Interrupt Flag Register (SPIxINTF)
- SPI Interrupt Enable Register (SPIxINTE)
- SPI Byte Count High and Low Registers (SPIxTCNTH/L)
- SPI Bit Count Register (SPIxTWIDTH)
- SPI Baud Rate Register (SPIxBAUD)
- SPI Control Register 0 (SPIxCON0)
- SPI Control Register 1 (SPIxCON1)
- SPI Control Register 2 (SPIxCON2)
- SPI FIFO Status Register (SPIxSTATUS)
- SPI Receiver Buffer Register (SPIxRXB)
- SPI Transmit Buffer Register (SPIxTXB)
- SPI Clock Select Register (SPIxCLK)

SPIxCON0, SPIxCON1, and SPIxCON2 are control registers for the SPI module.

SPIxSTATUS contains several Status bits that indicate the status of both the SPI module and the receive and transmit FIFOs.

SPIxBAUD and SPIxCLK control the baud rate generator of the SPI module when in Master mode. The SPIxCLK selects the clock source that is used. The SPIxBAUD configures the clock divider used on that clock. More information on the baud rate generator is available in **Section 32.5.6 “Master Mode SPI Clock Configuration”**.”

SPIxTxB and SPIxRxB are the transmit and receive buffer registers used to send and receive data on the SPI bus. They both offer indirect access to shift registers that are used for shifting the data in and out. Both registers access the two-byte FIFOs, allowing for multiple transmissions/receptions to be stored between software transfers the data.

The SPIxTCNTH:L register pair either count or control the number of bits or bytes in a data transfer. When BMODE = `1`, the SPIxTCNT value signifies bytes and the SPIxTWIDTH value signifies the number of bits in a byte. When BMODE = `0`, the SPIxTCNT value is concatenated with the SPIxTWIDTH register to signify bits. In Master Receive-only mode (TXR = `0` and RXR = `1`), the data transfer is initiated by writing SPIxTCNT with the desired bit or byte value to transfer. In Master Transmit mode (TXR = `1`), the data transfer is initiated by writing the SPIxTxB register, in which case the SPIxTCNT is a down counter for the bits or bytes transferred.

The SPIxINTF and SPIxINTE are the flags and enables, respectively, for SPI-specific interrupts. They are tied to the SPIxIF flag and SPIxIE enable in the PIR and PIE registers, which is triggered when any interrupt contained in the SPIxINTF/SPIxINTE registers is triggered. The PIR/PIE registers also contain SPIxTXIF/SPIxTXIE bits, which are the interrupt flag and enable for the SPI Transmit Interrupt, as well as the SPIxRXIF/SPIxRXIE bits, which are the interrupt flag and enable for the SPI Receive Interrupt.

32.3 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SPIxCON0[2:0], SPIxCON1[7:4], SPIxCON1[2:0], and SPIxCON2[2:0]). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Input, Output, and Slave Select Polarity
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on first/second edge of SCK)
- Clock Rate (Master mode only)
- Slave Select Mode (Master or Slave mode)
- MSB-First or LSB-First
- Receive/Transmit Modes
  - Full duplex
  - Receive-without-transmit
  - Transmit-without-receive
- Transfer Counter Mode (Transmit-without-receive mode)

32.3.1 ENABLING AND DISABLING THE SPI MODULE

To enable the serial peripheral, the SPI enable bit (EN in SPIxCON0) must be set. To reset or reconfigure SPI mode, clear the EN bit, re-initialize the SPIxCONx registers and then set the EN bit. Setting the EN bit enables the SPI inputs and outputs: SDI, SDO, SCK(out), SCK(in), SS(out), and SS(in). All of these inputs and outputs are steered by PPS, and thus must have their functions properly mapped to device pins to function (see **Section 17.0 "Peripheral Pin Select (PPS) Module"**). In addition, SS(out) and SCK(out) must have the pins they are steered to set as outputs (TRIS bits must be '0') in order to properly output. Clearing the TRIS bit of the SDO pin will cause the SPI module to always control that pin, but is not necessary for SDO functionality. (see **Section 32.3.5 "Input and Output Polarity Bits"**). Configurations selected by the following registers should not be changed while the EN bit is set:

- SPIxBAUD
- SPIxCON1
- SPIxCON0 (except to clear the EN bit)

Clearing the EN bit aborts any transmissions in progress, disables the setting of interrupt flags by hardware, and resets the FIFO occupancy (see **Section 32.3.3 "Transmit and Receive FIFOs"** for more FIFO details).

32.3.2 BUSY BIT

While a data transfer is in progress, the SPI module sets the BUSY bit of SPIxCON2. This bit can be polled by the user to determine the current status of the SPI module, and to know when a communication is complete. The following registers/bits should not be written by software while the BUSY bit is set:

- SPIxTCNTH/L
- SPIxTWIDTH
- SPIxCON2
- The CLRBF bit of SPIxSTATUS

Note 1: The BUSY bit is subject to synchronization delay of up to two instruction cycles. The user must wait for it to set after loading the transmit buffer (SPIxTXB register) before using it to determine the status of the SPI module.

2: It is also not recommended to read SPIxTCNTH/L while the BUSY bit is set, as the value in the registers may not be a reliable indicator of the Transfer Counter. Use the Transfer Count Zero Interrupt Flag (the TCZIF bit of SPIxINTF) to accurately determine that the Transfer Counter has reached zero.

32.3.3 TRANSMIT AND RECEIVE FIFOS

The transmission and reception of data from the SPI module is handled by two FIFOs, one for reception and one for transmission (addressed by the SFRs SPIxRXB and SPIxTXB, respectively.). The TXFIFO is written by software and is read by the SPI module to shift the data onto the SDO pin. The RXFIFO is written by the SPI module as it shifts in the data from the SDI pin and is read by software. Setting the CLRBF bit of SPIxSTATUS resets the occupancy for both FIFOs, emptying both buffers. The FIFOs are also reset by disabling the SPI module.

Note:	TXFIFO occupancy and RXFIFO occupancy simply refer to the number of bytes that are currently being stored in each FIFO. These values are used in this chapter to illustrate the function of these FIFOs and are not directly accessible through software.

The SPIxRXB register addresses the receive FIFO and is read-only. Reading from this register will read from the first FIFO location that was written to by hardware and decrease the RXFIFO occupancy. If the FIFO is empty, reading from this register will instead return a value of zero and set the RXRE (Receive Buffer Read Error) bit of the SPIxSTATUS register. The RXRE bit must then be cleared in software in order to properly reflect the status of the read error. When RXFIFO is full, the RXBF bit of the SPIxSTATUS register will be set. When the device receives data on the SDI pin, the receive FIFO may be written to by hardware and the occupancy increased, depending on the mode and receiver settings, as summarized in Table 32-1.

The SPIxTXB register addresses the transmit FIFO and is write-only. Writing to the register will write to the first empty FIFO location and increase the occupancy. If the FIFO is full, writing to this register will not affect the data and will set the TXWE bit of the SPIxSTATUS register. When the TXFIFO is empty, the TXBE bit of SPIxSTATUS will be set. When a data transfer occurs, data may be read from the first FIFO location written to and the occupancy decreases, depending on mode and transmitter settings, as summarized in Table 32-1 and **Section 32.6.1 “Slave Mode Transmit options”**.

32.3.4 LSB VS. MSB-FIRST OPERATION

Typically, SPI communication is output Most-Significant bit first, but some devices/buses may not conform to this standard. In this case, the LSBF bit may be used to alter the order in which bits are shifted out during the data exchange. In both Master and Slave mode, the LSBF bit of SPIxCON0 controls if data is shifted MSb or LSb first. Clearing the bit (default) configures the data to transfer MSb first, which is traditional SPI operation, while setting the bit configures the data to transfer LSb first.

32.3.5 INPUT AND OUTPUT POLARITY BITS

SPIxCON1 has three bits that control the polarity of the SPI inputs and outputs. The SDIP bit controls the polarity of the SDI input, the SDOP bit controls the polarity of the SDO output, and the SSP bit controls the polarity of both the slave SS input and the master SS output. For all three bits, when the bit is clear, the input or output is active-high, and when the bit is set, the input or output is active-low. When the EN bit of SPIxCON0 is cleared, SS(out) and SCK(out) both revert to the inactive state dictated by their polarity bits. The SDO output state when the EN bit of SPIxCON0 is cleared is determined by several factors.

- When the associated TRIS bit for the SDO pin is cleared, and the SPI goes idle after a transmission, the SDO output will remain at the last bit level. The SDO pin will revert to the Idle state if EN is cleared.
- When the associated TRIS bit for the SDO pin is set, behavior varies in Slave and Master mode.
  - In Slave mode, the SDO pin tri-states when:
  - Slave Select is inactive,
  - the EN bit of SPIxCON0 is cleared, or when
  - the TXR bit of SPIxCON2 is cleared.
  - In Master mode, the SDO pin tri-states when TXR = `0`. When TXR = `1` and the SPI goes idle after a transmission, the SDO output will remain at the last bit level. The SDO pin will revert to the Idle state if EN is cleared.

32.4 Transfer Counter

In all master modes, the transfer counter can be used to determine how many data transfers the SPI will send/receive. The transfer counter is comprised of the SPIxTCNTH/L set of registers, and is also partially controlled by the SPIxTWIDTH register. The Transfer Counter has two primary modes, determined by the BMODE bit of the SPIxCON0 register. Each mode uses the SPIxTCNTH/L and SPIxTWIDTH registers to determine the number and size of the transfers. In both modes, when the transfer counter reaches zero, the TCZIF interrupt flag is set.

Note:	When BMODE=1 in all master modes (and at all times in slave modes), the Transfer Counter will still decrement as transfers occur and can be used to count the number of messages sent/received, as well as to control SS(out) and to trigger TCZIF. Also when BMODE = 1, the SPIxTWIDTH register can be used in Master and Slave modes to determine the size of messages sent and received by the SPI, even if the Transfer Counter is not being actively used to control the number of messages being sent/received by the SPI module.

32.4.1 TOTAL BIT COUNT MODE (BMODE = 0)

In this mode, SPIxTCNTH/L and SPIxTWIDTH are concatenated to determine the total number of bits to be transferred. These bits will be loaded from/into the transmit/receive FIFOs in 8-bit increments and the transfer counter will be decremented by eight until the total number of remaining bits is less than eight. If there are any remaining bits (SPIxTWIDTH ≠ 0), the transmit FIFO will send out one final message with any extra bits greater than the remainder ignored. The SPIxTWIDTH is the remaining bit count but the value does not change as it does for the SPIxTCNT value. Similarly, the receiver will load a final byte into the receiver FIFO, and pad the extra bits with zeros. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of this final byte are ignored/padded. For example, when LSBF = 0 and the final transfer contains only two bits then if the last byte sent was 5Fh then the RXB of the receiver will contain 40h which are the two MSbits of the final byte padded with zeros in the LSbits.

In this mode, the SPI master will only transmit messages when the SPIxTCNT value is greater than zero, regardless of TXR and RXR settings. In Master Transmit mode, the transfer starts with the data write to the SPIxTXB register or the count value written to the SPIxTCNTL register, which ever occurs last. In Master Receive-only mode, the transfer clocks start when the SPIxTCNTL value is written. Transfer clocks are suspended when the receive FIFO is full and resume as the FIFO is read.

32.4.2 VARIABLE TRANSFER SIZE MODE (BMODE = 1)

In this mode, SPIxTWIDTH specifies the width of every individual piece of the data transfer in bits. SPIxTCNTH/SPIxTCNTL specifies the number of transfers of this bit length. If SPIxTWIDTH = 0, each piece is a full byte of data. If SPIxTWIDTH ≠ 0, then only the specified number of bits from the transmit FIFO are shifted out, with the unused bits ignored. Received data is padded with zeros in the unused bit areas when transfered into the receive FIFO. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of the transfers are ignored/padded. In this mode, the transfer counter being zero only stops messages from being sent/received when in “Receive only” mode.

Note:	With BMODE = 1, it is possible for the transfer counter (SPIxTCNTH/L) to decrement below zero, although when in “Receive only” Master mode, transfer clocks will cease when the transfer counter reaches zero.

32.4.3 TRANSFER COUNTER IN SLAVE MODE

In Slave Mode, the transfer counter will still decrement as data is shifted in and out of the SPI module, but it will not control data transfers. In addition, in slave mode, the BMODE bit along with the transfer counter is used to determine when the device should look for Slave Select faults. If BMODE = 0, the SSFLT bit will be set if Slave Select transitions from its active to inactive state during bytes of data, as well as if it transitions before the last bit sent during the final byte (if SPIxTWIDTH ≠ 0). If BMODE = 1, the SSFLT bit will be set if Slave Select transitions from its active to inactive state before the final bit of each individual transfer is completed. Note that SSFLT does not have an associated interrupt, so it should be checked in software. An ideal time to do this is when the End of Slave Select Interrupt (EOSIF) is triggered (see **Section 32.8.3.3 "Start of Slave Select and End of Slave Select Interrupts"**).

32.5 Master mode

In master mode, the device controls the SCK line, and as such, initiates data transfers and determines when any slaves broadcast data onto the SPI bus.

Master mode of this device can be configured in four different modes, configured by the TXR and RXR bits:

- Full Duplex mode
- Receive Only mode
- Transmit Only mode
- Transfer Off mode

The modes are illustrated in Table 32-1, below:

TABLE 32-1: MASTER MODE TXR/RXR SETTINGS

	TXR = 1	**TXR = 0**
RXR = 1	Full Duplex Mode If BMODE = 1, transfer when RxFIFO is not full and TxFIFO is not empty If BMODE = 0, Transfer when RXFIFO is not full, TXFIFO is not empty, and the Transfer Counter is non-zero	Receive Only mode Transfer when RxFIFO is not full and the Transfer Counter is non-zero Transmitted data is either the top of the FIFO or the most recently received data
RXR = 0	Transmit Only Mode If BMODE = 1, transfer when TxFIFO is not empty If BMODE = 0, Transfer when TXFIFO is not empty and the Transfer Counter is non-zero Received data is not stored	No Transfers

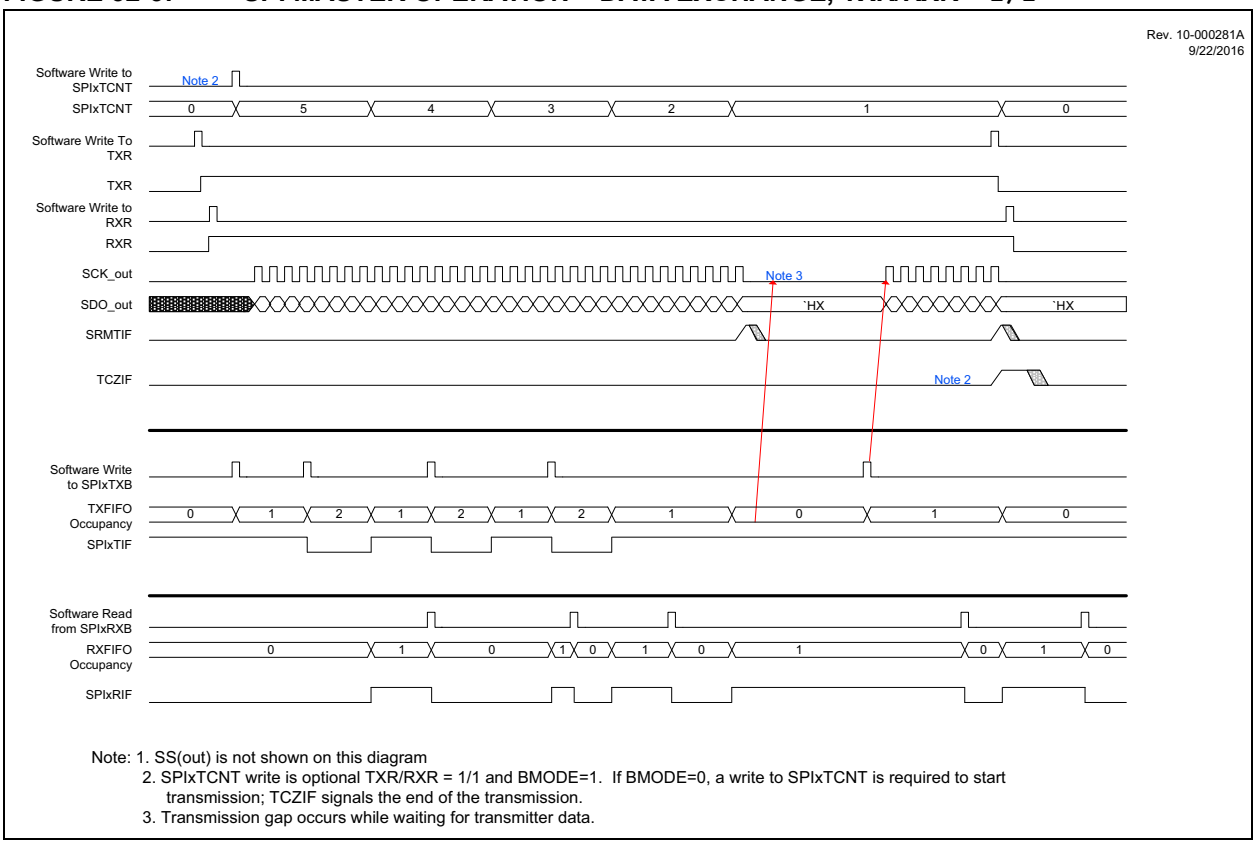
32.5.1 FULL DUPLEX MODE

When both TXR and RXR are set, the SPI master is in Full Duplex mode. In this mode, data transfer triggering is affected by the BMODE bit of SPIxCON0.

When BMODE = `1`, data transfers will occur whenever both the RXFIFO is not full and there is data present in the TXFIFO. In practice, as long as the RXFIFO is not full, data will be transmitted/received as soon as the SPIxTxB register is written to, matching functionality of SPI (MSSP) modules on older 8-bit Microchip devices. The SPIxTCNT will decrement with each transfer. However, when SPIxTCNT is zero the next transfer is not inhibited and the corresponding SPIxTCNT decrement will cause the count to roll over to the maximum value. Figure 32-3 shows an example of a communication using this mode.

When BMODE = `0`, the transfer counter (SPIxTCNTH/SPIxTCNTL) must also be written to before transfers will occur, and transfers will cease when the transfer counter reaches '`0`'. For example, if SPIxTXB is written twice and then SPIxTCNTL is written with '3' then the transfer will start with the SPIxTCNTL write. The two bytes in the TXFIFO will be sent after which the transfer will suspend until the third and last byte is written to SPIxTXB.

FIGURE 32-3: SPI MASTER OPERATION – DATA EXCHANGE, TXR/RXR = `1/1`



32.5.2 TRANSMIT ONLY MODE

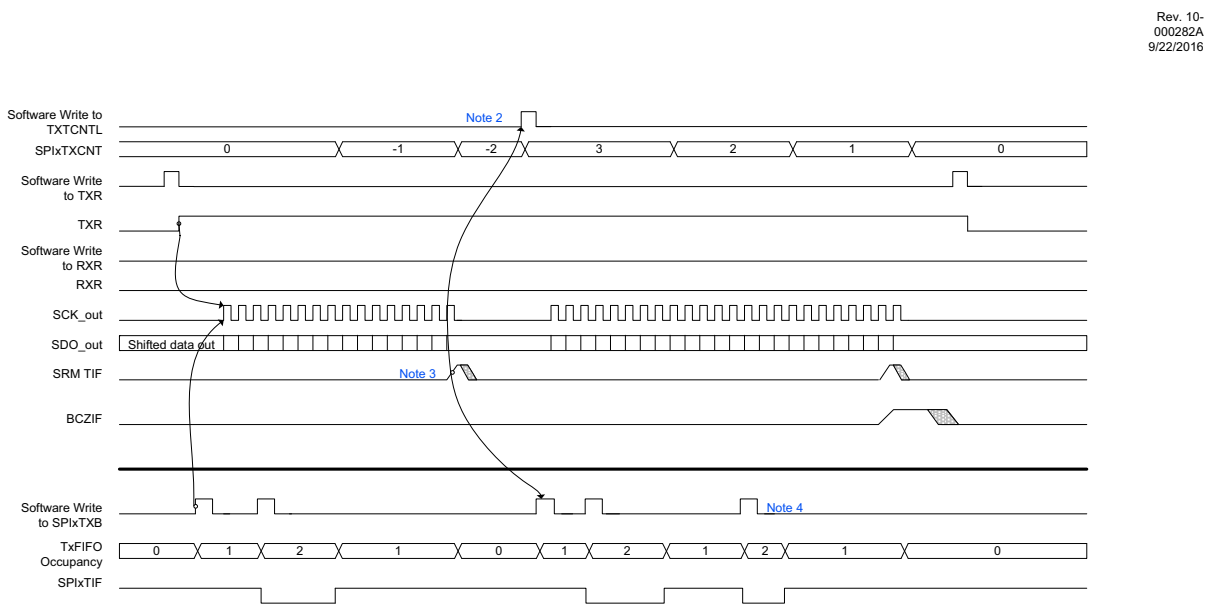
When TXR is set and RXR is clear, the SPI master is in Transmit Only mode. In this mode, data transfer triggering is affected by the BMODE bit of SPIxCON0.

When BMODE = `1`, data transfers will occur whenever TXFIFO is not empty. Data will be transmitted as soon as the TXFIFO register is written to, matching functionality of SPI (MSSP) modules on previous 8-bit Microchip devices. The SPIxTCNT will decrement with each transfer. However, when SPIxTCNT is zero the next transfer is not inhibited and the corresponding SPIxTCNT decrement will cause the count to roll over to the maximum value. Any data received in this mode is not stored in RXFIFO. Figure 32-4 shows an example of sending a command and then sending a byte of data, using this mode.

When BMODE = `0`, the transfer counter (SPIxTCNTH/L) must also be written to before transfers will occur, and transfers will cease when the transfer counter reaches '`0`'.

For example, if SPIxTXB is written twice and then SPIxTCNTL is written with '3', the transfer will start with the SPIxTCNTL write. The two bytes in the TXFIFO will be sent after which the transfer will suspend until the third and last byte is written to SPIxTXB.

FIGURE 32-4: SPI MASTER OPERATION, COMMAND+WRITE DATA, TXR/RXR=`1/0`

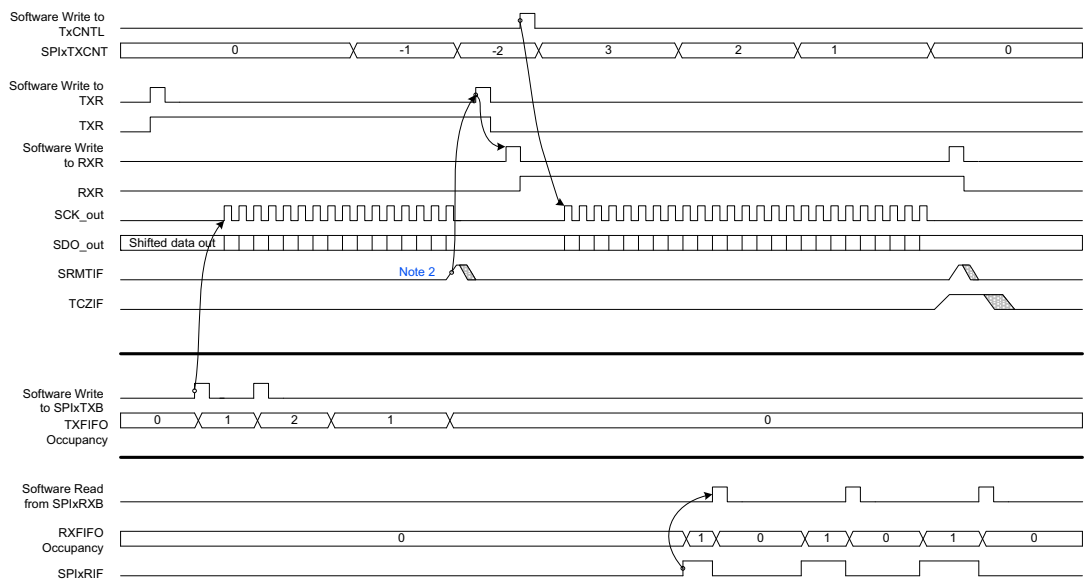


Note: 1. SS(out) is not shown
2. The byte counter is optional when TXR/RXR = 1/0;
3. After the command bytes, wait for SRMTIF before loading SPIxBYTESL otherwise the command bytes would decrement BYTES. Alternatively, load BC = 5 and count the command bytes also; TCZIF signals the end of the transmission.
4. Transmit data interrupt handler (or DMA) must write only the bytes necessary; the byte counter is not available as an indicator.
5. Reading the RXFIFO is not required because RXR = 0.

32.5.3 RECEIVE ONLY MODE

When RXR is set and TXR is clear, the SPI master is in Receive Only mode. In this mode, data transfers when the RXFIFO is not full and the Transfer Counter is non-zero. In this mode, writing a value to SPIxTCNTL will start the clocks for transfer. The clocks will suspend while the RXFIFO is full and cease when the SPIxTCNT reaches zero (see **Section 32.4 "Transfer Counter"**). If there is any data in the TXFIFO, the first data written to the TXFIFO will be transmitted on each data exchange, although the TXFIFO occupancy will not change, meaning that the same message will be sent on each transmission. If there is no data in the TXFIFO, the most recently received data will instead be transmitted. Figure 32-5 shows an example of sending a command using **Section 32.5.2 "Transmit Only Mode"** and then receiving a byte of data using this mode.

FIGURE 32-5: SPI MASTER OPERATION, COMMAND+READ DATA, TXR/RXR=`0/1`



Note: 1. SS(out) is not shown
2. Software must wait for shift-register empty (SRMTIF) before changing TXR, RXR, BYTES and BITS controls. This is not considered an imposition in this case, because the slave probably needs time to load output data (see also Figure 4-14).

32.5.4 TRANSFER OFF MODE

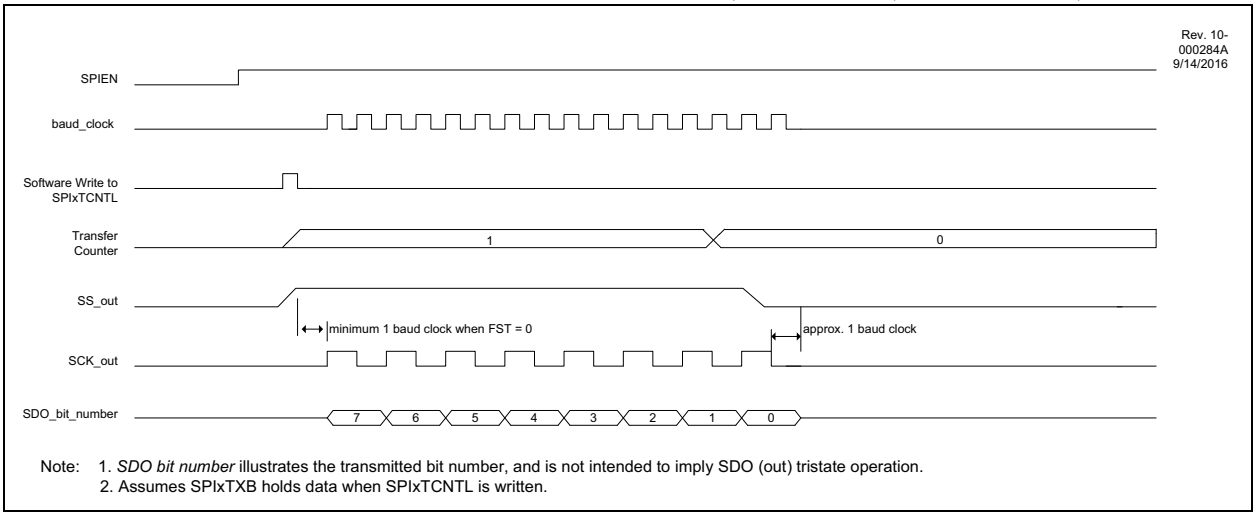
When both TXR and RXR are cleared, the SPI master is in Transfer Off mode. In this mode, SCK will not toggle and no data is exchanged. However, writes to SPIxTXB will be transferred to the TXFIFO which will be transmitted if the TXR bit is set.

32.5.5 MASTER MODE SLAVE SELECT CONTROL

32.5.5.1 Hardware Slave Select Control

This SPI module allows for direct hardware control of a Slave Select output. The Slave Select output SS(out) is controlled both directly, through the SSET bit of SPIxCON2, as well indirectly by the hardware while the transfer counter is non-zero (see **Section 32.4 "Transfer Counter"**). SS(out) is steered by the PPS registers to pins (see **Section 17.2 "PPS Outputs"**) and its polarity is controlled by the SSP bit of SPIxCON1. Setting the SSET bit will also assert SS(out). Clearing the SSET bit will leave SS(out) to be controlled by the Transfer Counter. When the Transfer Counter is loaded, the SPI module will automatically assert the SS. When the Transfer Counter decrements to zero, the SPI module will deassert SS either one baud period after the final SCK pulse of the final transfer (if CKE/SMP = `0/1`) or one half baud period otherwise (see Figure 32-6).

FIGURE 32-6: SPI MASTER SS OPERATION- CKE = `0`, BMODE = `1`, TCWIDTH = `0`, SSP = `0`



Note: 1. *SDO bit number* illustrates the transmitted bit number, and is not intended to imply SDO (out) tristate operation.
2. Assumes SPIxTXB holds data when SPIxTCNTL is written.

32.5.5.2 Software Slave Select Control

Slave Select can also be controlled through software via a general purpose I/O pin. In this case, ensure that the pin in question is configured as a GPIO through PPS (see **Section 17.2 "PPS Outputs"**), and ensure that the pin is set as an output (clear the appropriate bit in the appropriate TRIS register). In this case, SSET will not affect the slave select, the Transfer Counter will not automatically control the slave select output, and all setting and clearing of the slave select output line must be directly controlled by software.

32.5.6 MASTER MODE SPI CLOCK CONFIGURATION

32.5.6.1 SPI Clock Selection

The clock source for SPI master modes is selected by the SPIxCLK register. Selections include the following:

- FOSC
- HFINTOSC
- CLKREF
- Timer0_overflow
- Timer2_Postscaled
- Timer4_Postscaled
- Timer6_Postscaled
- SMT_match

The SPIxBAUD register allows for dividing this clock. The frequency of the SCK output is defined by Equation 32-1:

EQUATION 32-1: FREQUENCY OF SCK OUTPUT SIGNAL

$$F_{BAUD} = \frac{F_{CSEL}}{(2 \cdot (BAUD + 1))}$$

where FBAUD is the baud rate frequency output on the SCK pin, FCSEL is the frequency of the input clock selected by the SPIxCLK register, and BAUD is the value contained in the SPIxBAUD register.

32.5.6.2 CKE, CKP and SMP

The CKP, CKE, and SMP bits control the relationship between the SCK clock output, SDO output data changes, and SDI input data sampling. The bit functions are as follows:

- CKP - SCK output polarity
- CKE - SDO output change relative to the SCK clock
- SMP - SDI input sampling relative to the clock edges

The CKE bit, when set, inverts the low Idle state of the SCK output to a high Idle state.

Figure 32-7 through Figure 32-10 illustrate the eight possible combinations of the CKP, CKE, and SMP bit selections.

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. When the CKE bit is cleared, the SDO data is undefined prior to the first SCK edge.

Note: All timing diagrams assume the LSBF bit of SPIxCON0 is cleared.

FIGURE 32-7: CLOCKING DETAIL-MASTER MODE, CKE/SMP = 0/0

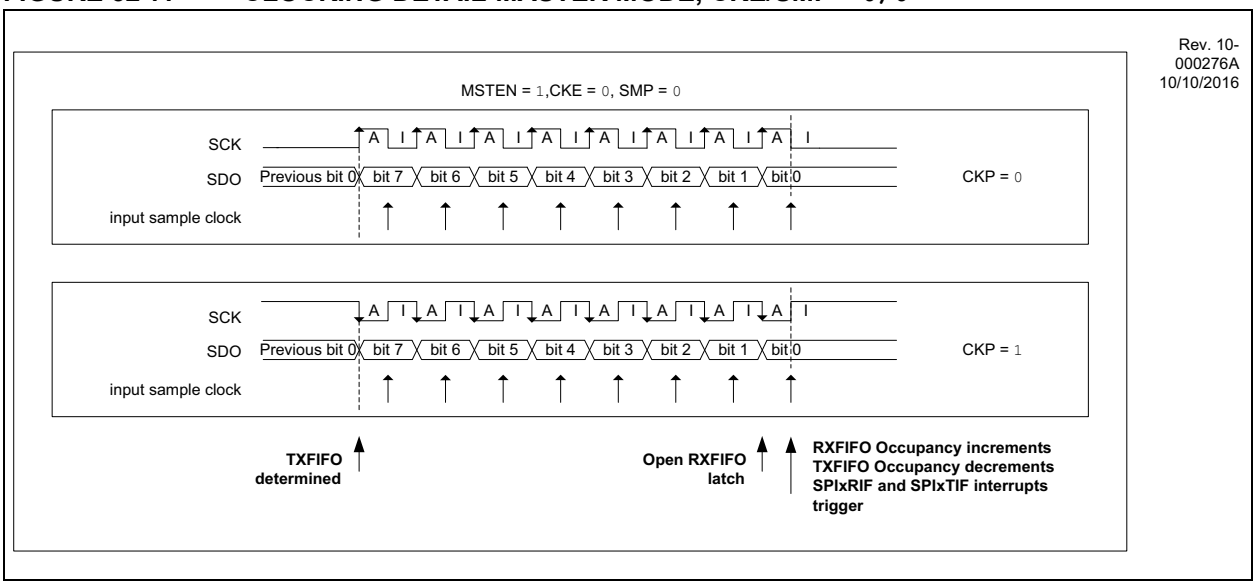


FIGURE 32-8: CLOCKING DETAIL - MASTER MODE, CKE/SMP = 1/1

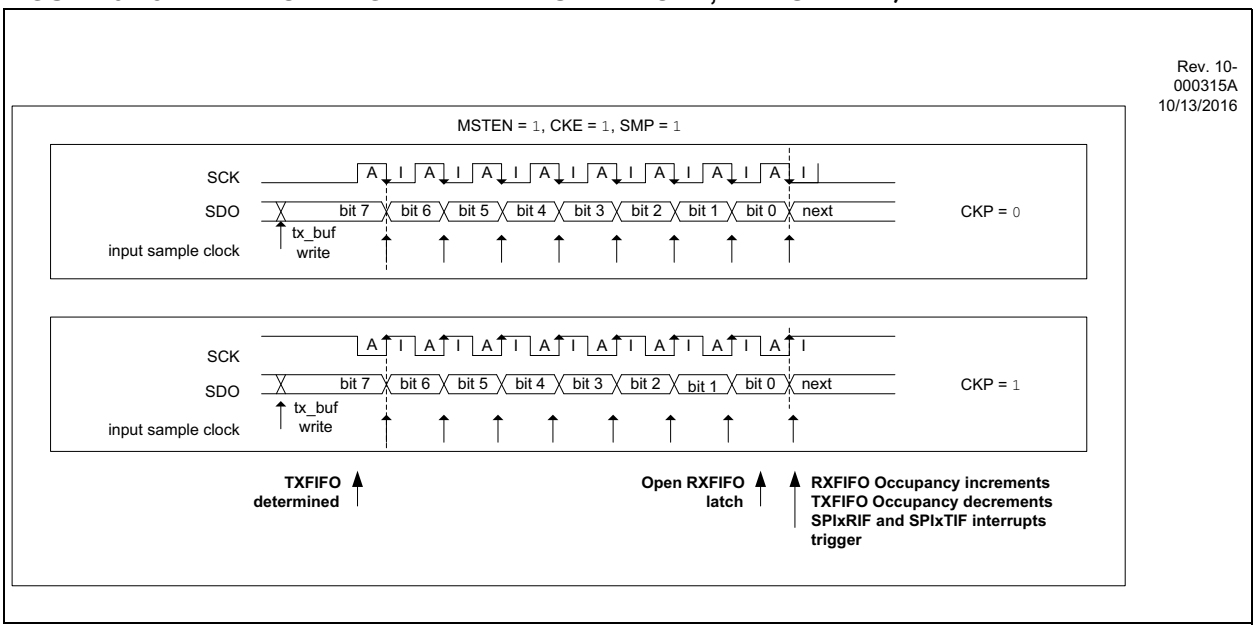


FIGURE 32-9: CLOCKING DETAIL-MASTER MODE, CKE = 0, SMP = 1

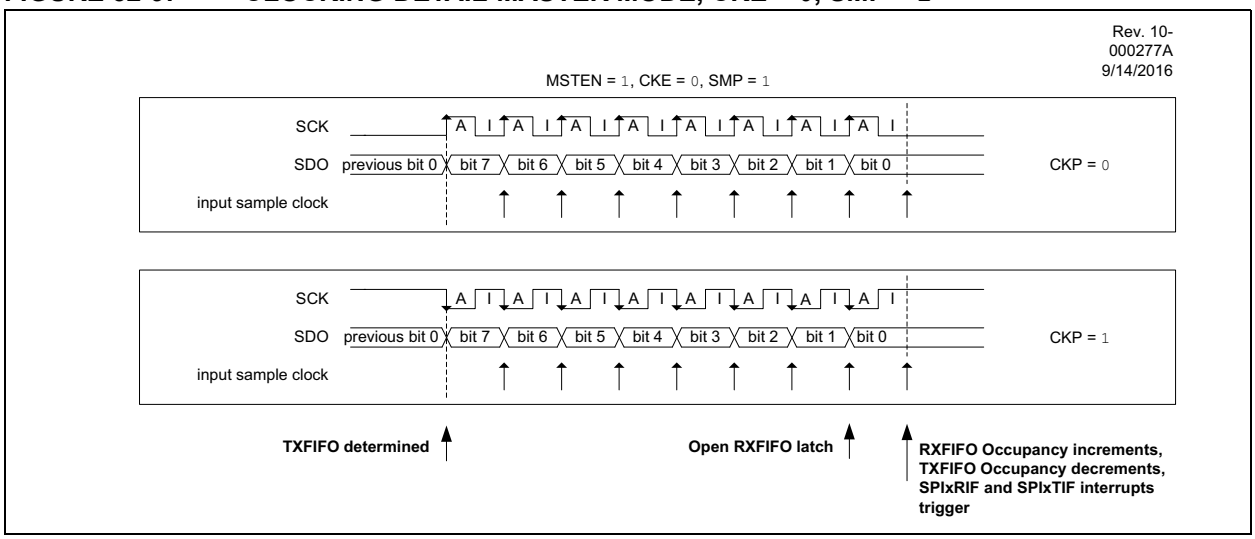
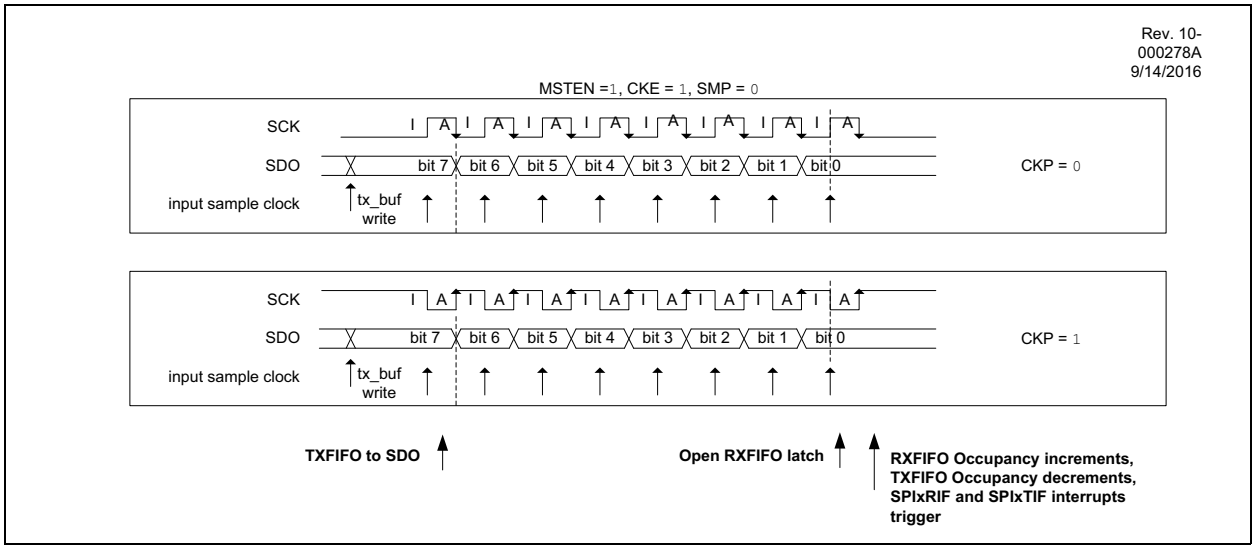


FIGURE 32-10: CLOCKING DETAIL-MASTER MODE, CKE = 1, SMP = 0



32.5.6.3 SCK Start-Up Delay

When starting an SPI data exchange, the master device sets the SS output (either through hardware or software) and then triggers the module to send data. These data triggers are synchronized to the clock selected by the SPIxCLK register before the first SCK pulse appears, usually requiring one or two clocks of the selected clock.

The SPI module includes synchronization delays on SCK generation specifically designed to ensure that the Slave Select output timing is correct, without requiring precision software timing loops.

When the value of the SPIxBAUD register is a small number (indicating higher SCK frequencies), the synchronization delay can be relatively long between setting SS and the first SCK. With larger values of SPIxBAUD (indicating lower SCK frequencies), this delay is much smaller and the first SCK can appear relatively quickly after SS is set.

By default, the SPI module inserts a ½ baud delay (half of the period of the clock selected by the SPIxCLK register) before the first SCK pulse. This allows for systems with a high SPIxBAUD value to have extra set-up time before the first clock. Setting the FST bit in SPIxCON1 removes this additional delay, allowing systems with low SPIxBAUD values (and thus, long synchronization delays) to forego this unnecessary extra delay.

32.6 Slave Mode

32.6.1 SLAVE MODE TRANSMIT OPTIONS

The SDO output of the SPI module in Slave mode is controlled by the TXR bit of SPIxCON2, the TRIS bit associated with the SDO pin, the Slave Select input, and the current state of the TXFIFO. This control is summarized in Table 32-2. In this table, TRISxn refers to the bit in the TRIS register corresponding to the pin that SDO has been assigned with PPS, TXR is the Transmit Data Required Control bit of SPIxCON2, SS is the state of the Slave Select input, and TXBE is the TXFIFO Buffer Empty bit of SPIxSTATUS.

32.6.1.1 SDO Drive/Tri-state

The TRIS bit associated with the SDO pin controls whether the SDO pin will tri-state. When this TRIS bit is cleared, the pin will always be driving to a level, even when the SPI module is inactive. When the SPI module is inactive (either due to the master not clocking the SCK line or the SS being false), the SDO pin will be driven to the value of the LAT bit associated with the SDO pin. When the SPI module is active, its output is determined by both TXR and whether there is data in the TXFIFO.

When the TRIS bit associated with the SDO pin is set, the pin will only have an output level driven to it when TXR = 1 and the slave select input is true. In all other cases, the pin will be tri-stated.

32.6.1.2 SDO Output Data

The TXR bit controls the nature of the data that is transmitted in Slave mode. When TXR is set, transmitted data is taken from the TXFIFO. If the FIFO is empty, the most recently received data will be transmitted and the TXUIF flag will be set to indicate that a transmit FIFO underflow has occurred.

When TXR is cleared, the data will be taken from the TXFIFO, and the TXFIFO occupancy will not decrease. If the TXFIFO is empty, the most recently received data will be transmitted, and the TXUIF bit will not be set. However, if the TRIS bit associated with the SDO pin is set, clearing the TXR bit will cause the SPI module to not output any data to the SDO pin.

TABLE 32-2: SLAVE MODE TRANSMIT

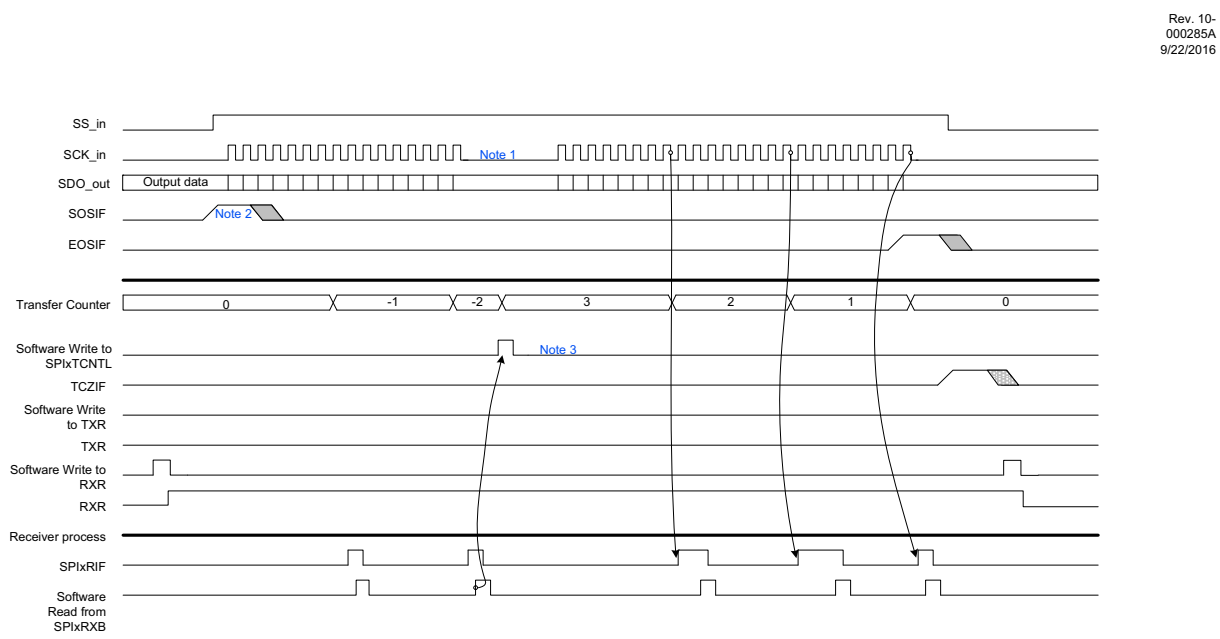
TRISxn(1)	TXR	SS	TXBE	SDO State
0	0	FALSE	0	Drives state determined by LATxn(2)
0	0	FALSE	1	Drives state determined by LATxn(2)
0	0	TRUE	0	Outputs the oldest byte in the TXFIFO Does not remove data from the TXFIFO
0	0	TRUE	1	Outputs the most recently received byte
0	1	FALSE	0	Drives state determined by LATxn(2)
0	1	FALSE	1	Drives state determined by LATxn(2)
0	1	TRUE	0	Outputs the oldest byte in the TXFIFO Removes transmitted byte from the TXFIFO Decrements occupancy of TXFIFO
0	1	TRUE	1	Outputs the most recently received byte Sets the TXUIF bit of SPIxINTF
1	0	FALSE	0	Tri-stated
1	0	FALSE	1	Tri-stated
1	0	TRUE	0	Tri-stated
1	0	TRUE	1	Tri-stated
1	1	FALSE	0	Tri-stated
1	1	FALSE	1	Tri-stated
1	1	TRUE	0	Outputs the oldest byte in the TXFIFO Removes transmitted byte from the TXFIFO Decrements occupancy of TXFIFO
1	1	TRUE	1	Outputs the most recently received byte Sets the TXUIF bit of SPIxINTF

Note 1: TRISxn is the bit in the TRISx register corresponding to the pin that SDO has been assigned with PPS.
2: LATxn is the bit in the LATx register corresponding to the pin that SDO has been assigned with PPS.

32.6.2 SLAVE MODE RECEIVE OPTIONS

The RXR bit controls the nature of receptions in slave mode. When RXR is set, the SDI input data will be stored in the RXFIFO if it is not full. If the RXFIFO is full, the RXOIF bit will be set to indicate an RXFIFO overflow error and the data is discarded. When RXR is cleared, all received data will be ignored and not stored in the RXFIFO (although it may still be used for transmission if TXFIFO is empty). Figure 32-11 shows a typical slave mode communication, showing a case where the master writes two then three bytes, showing interrupts as well as the behavior of the transfer counter in slave mode (see **Section 32.4.3 "Transfer Counter in Slave mode"** for more details on t**Section 32.8 "SPI Interrupts"**he transfer counter in slave mode as well as Section X.8 for more information on interrupts).

FIGURE 32-11: SPI SLAVE MODE OPERATION – INTERRUPT-DRIVEN, MASTER WRITES 2+3 BYTES



Note: 1. This delay is exaggerated for illustration, and can be as short as1/2 bit period.
2. If the device is sleeping, SOSIF will wake it up for interrupt service.
3. Setting SPIxTCNTL is optional in this example, otherwise it will count -3, -4, -5, and TCZIF will not occur

32.6.3 SLAVE MODE SLAVE SELECT

In Slave mode, an external Slave Select Signal can be used to synchronize communication with the Master device. The Slave Select line is held in its inactive state (high by default) until the master device is ready to communicate. When the Slave Select transitions to its active state, the slave knows that a new transmission is starting.

When the Slave Select goes false at the end of the transmission the receive function of the selected SPI Slave device returns to the inactive state. The slave is then ready to receive a new transmission when the Slave Select goes True again.

The Slave Select signal is received on the SS input pin. This pin is remappable with the SPIxSSPPS register (see **Section 17.1 "PPS Inputs"**). When the input on this pin is true, transmission and reception are enabled, and the SDO pin is driven. When the input on this pin is false, the SDO pin is either tri-stated (if the TRIS bit associated with the SDO pin is set) or driven to the value of the LAT bit associated with the SDO pin (if the TRIS bit associated with the SDO pin is cleared). In addition, the SCK input is ignored.

If the SS input goes False, while a data transfer is still in progress, it is considered a slave select fault. The SSFLT bit of SPIxCON2 indicates whether such an event has occurred. The transfer counter value determines the number of bits in a valid data transfer (see **Section 32.4 "Transfer Counter"** for more details).

The Slave Select polarity is controlled by the SSP bit of SPIxCON1. When SSP is set (its default state), the Slave Select input is active-low, and when it is cleared, the Slave Select input is active-high.

The Slave Select for the SPI module is controlled by the SSET bit of SPIxCON2. When the bit is cleared (its default state), the slave select will act as described above. When the bit is set, the SPI module will behave as if the SS input was always in its active state.

Note:	When SSET is set, the effective SS(in) signal is always active. Hence, the SSFLT bit may be disregarded.

32.6.4 SLAVE MODE CLOCK CONFIGURATION

In Slave Mode, SCK is an input, and must be configured to the same polarity and clock edge as the master device. As in Master mode, the polarity of the clock input is controlled by the CKP bit of SPIxCON1 and the clock edge used for transmitting data is controlled by the CKE bit of SPIxCON1.

32.6.5 DAISY-CHAIN CONFIGURATION

The SPI bus can be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device connected to all slave devices (alternately, the slave devices can be configured to ignore the slave select line by setting the SSET bit). In a typical Daisy-Chain configuration, the SCK signal from the master is connected to each of the slave device SCK inputs. However, the SCK input and output are separate signals selected by the PPS control. When the PPS selection is made to configure the SCK input and SCK output on separate pins then, the SCK output will follow the SCK input, allowing for SCK signals to be daisy-chained like the SDO/SDI signals.

Figure 32-12 shows the block diagram of a typical daisy-chain connection, and Figure 32-13 shows the block diagram of a daisy-chain connection possible using this SPI module.

FIGURE 32-12: TRADITIONAL SPI DAISY-CHAIN CONNECTION

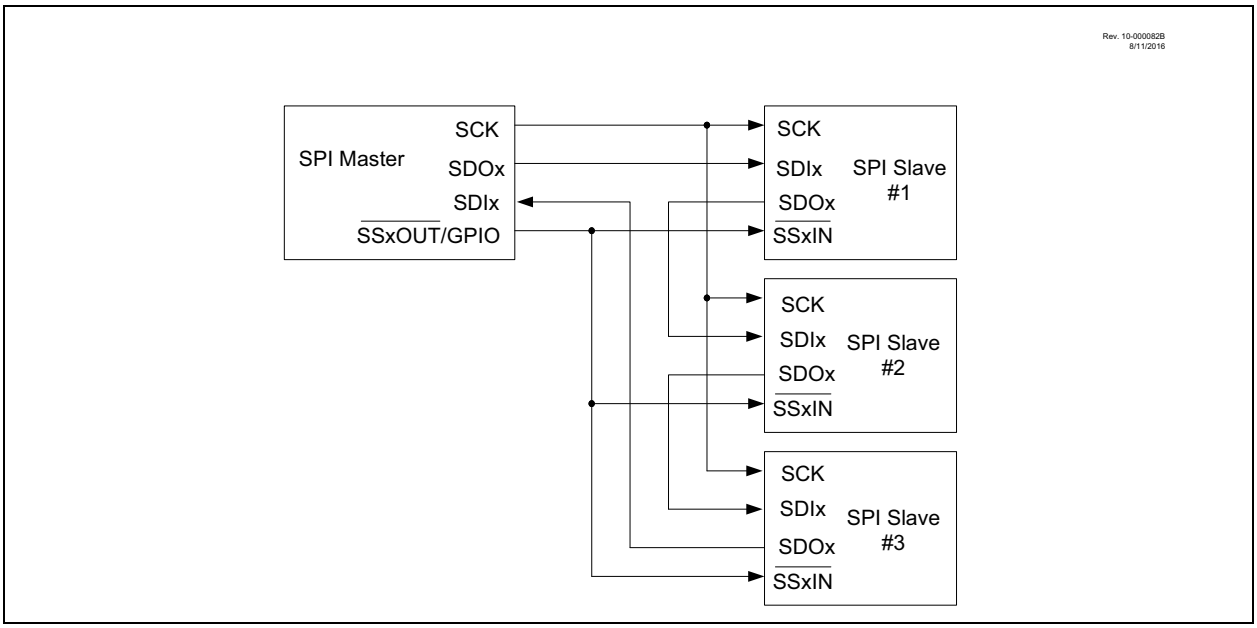
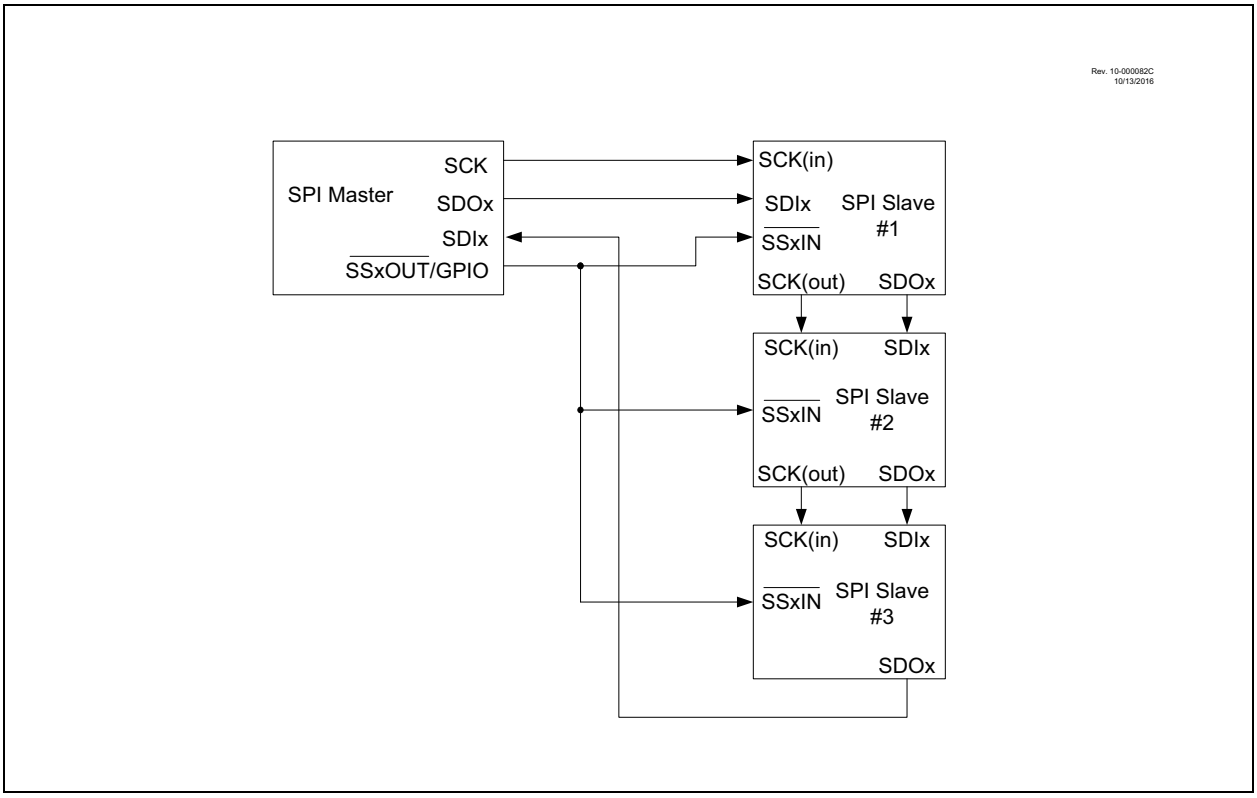


FIGURE 32-13: SPI DAISY-CHAIN CONNECTION WITH CHAINED SCK



32.7 SPI Operation in Sleep Mode

SPI master mode will operate in Sleep, provided the clock source selected by SPIxCLK is active in Sleep mode. FIFOs will operate as they would when the part is awake. When TXR = 1, the TXFIFO will need to contain data in order for transfers to take place in Sleep. All interrupts will still set the interrupt flags in Sleep but only enabled interrupts will wake the device from Sleep.

SPI Slave mode will operate in Sleep, because the clock is provided by an external master device. FIFOs will still operate and interrupts will set interrupt flags, and enabled interrupts will wake the device from Sleep.

32.8 SPI Interrupts

There are three top level SPI interrupts in the PIRx register:

- SPI Transmit
- SPI Receive
- SPI Module status

The status interrupts are enabled at the module level in the SPIxINTE register. Only enabled status interrupts will cause the single top level SPIxIF flag to be set.

32.8.1 SPI RECEIVER DATA INTERRUPT

The SPI Receiver Data Interrupt is set when RXFIFO contains data, and is cleared when the RXFIFO is empty. The interrupt flag SPI1RXIF is located in PIRx and the interrupt enable SPI1RXIE is located in PIEx. This interrupt flag is read-only.

32.8.2 SPI TRANSMITTER DATA INTERRUPT

The SPI Transmitter Data Interrupt is set when TXFIFO is not full, and is cleared when the TXFIFO is full. The interrupt flag SPI1TXIF is located in PIRx and the interrupt enable SPI1TXIE is located in PIEx. The interrupt flag is read-only.

32.8.3 SPI MODULE STATUS INTERRUPTS

The SPIxIF flag in the respective PIR register is set when any of the individual status flags in SPIxINTF and their respective SPIxINTE bits are set. In order for the setting of any specific interrupt flag to interrupt normal program flow both the SPIxIE bit as well as the specific bit in SPIxINTE associated with that interrupt must be set.

The Status Interrupts are:

- Shift Register Empty Interrupt
- Transfer Counter is Zero Interrupt
- Start of Slave Select Interrupt
- End of Slave Select Interrupt
- Receiver Overflow Interrupt
- Transmitter Underflow Interrupt

32.8.3.1 Shift Register Empty Interrupt

The Shift Register Empty interrupt flag and enable are the SRMTIF and SRMTIE bits respectively. This interrupt is only available in master mode and triggers when a data transfer completes and conditions are not present to start a new transfer, as dictated by the TXR and RXR bits (see Table 32-1 for conditions for starting a new Master mode data transfer with different TXR/RXR settings). This interrupt will be triggered at the end of the last full bit period, after SCK has been low for one ½-baud period. See Figure 32-14 for more details of the timing of this interrupt as well as other interrupts. This bit will not clear itself when the conditions for starting a new transfer occur, and must be cleared in software.

32.8.3.2 Transfer Counter is Zero Interrupt

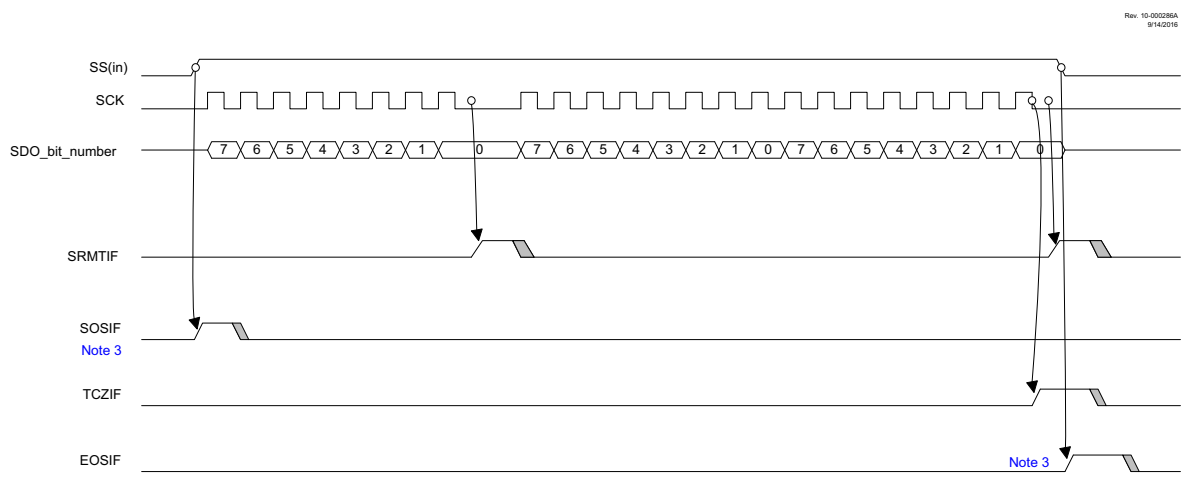
The Transfer Counter is zero interrupt flag and enable are the TCZIF and TCZIE bits, respectively. This interrupt will trigger when the transfer counter (defined by BMODE, SPIxTCNTH/L and SPIxTWIDTH) decrements from one to zero. See Figure 32-14 for more details on the timing of this interrupt as well as other interrupts. This bit must be cleared in software.

> **Note:** The TCZIF flag only indicates that the transfer counter has decremented from one to zero, and may not indicate that the entire data transfer process is complete. Either poll the BUSY bit of SPIxCON2 and wait for it to be cleared or use the Shift Register Empty Interrupt (SRMTIF) to determine if a data transfer is fully complete.

32.8.3.3 Start of Slave Select and End of Slave Select Interrupts

The start of slave select interrupt flag and enable are the SOSIF and SOSIE bits, respectively, and the end of slave select interrupt flag and enable are similarly designated by the EOSIF and EOSIE bits. These interrupts trigger at the leading and trailing edges of the slave select input. Note that the interrupts are active in both master and slave mode, and will trigger on transitions of the slave select input regardless of which mode the SPI is in. In Master mode, PPS should be used to route the slave select input to the same pin as the slave select output, allowing these interrupts to trigger on changes to the slave select output. Also note that in slave mode, changing the SSET bit can trigger these interrupts, as it changes the effective input value of slave select. Both SOSIF and EOSIF must be cleared in software

FIGURE 32-14: TRANSFER AND SLAVE SELECT INTERRUPT TIMINGS



Note 1: SRMTIF available only in Master mode
2: Clearing of interrupt flags is shown for illustration; actual interrupt flags must be cleared in software
3: SOSIF and EOSIF are set according to SS(in), even in Master mode.

32.8.3.4 Receiver Overflow and Transmitter Underflow Interrupts

The receiver overflow interrupt triggers if data is received when the RXFIFO is already full and RXR = 1. In this case, the data will be discarded and the RXOIF bit will be set. The receiver overflow interrupt flag is the RXOIF bit of SPIxINTF. The receiver overflow interrupt enable bit is the RXOIE bit of SPIxINTE.

The Transmitter Underflow interrupt flag triggers if a data transfer begins when the TXFIFO is empty and TXR = 1. In this case, the most recently received data will be transmitted and the TXUIF bit will be set. The transmitter underflow interrupt flag is the TXUIF bit of SPIxINTF. The transmitter underflow interrupt enable bit is the TXUIE bit of SPIxINTE.

Both of these interrupts will only occur in Slave mode, as Master mode will not allow the RXFIFO to overflow or the TXFIFO to underflow.

32.9 Register definitions: SPI

REGISTER 32-1: SPIxINTF: SPI INTERRUPT FLAG REGISTER

R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	U-0	R/W/HS-0/0	R/W/HS-0/0	U-0
SRMTIF	TCZIF	SOSIF	EOSIF	—	RXOIF	TXUIF	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’ HS = Bit can be set by hardware

bit 7 **SRMTIF**: Shift Register Empty Interrupt Flag bit

Slave mode:

This bit is ignored

Master mode:

1 = The data transfer is complete

0 = Either no data transfers have occurred or a data transfer is in progress

bit 6 **TCZIF**: Transfer Counter is Zero Interrupt Flag bit

1 = The transfer counter (as defined by BMODE in Register 32-7, TCNTH/L, and TWIDTH) has decremented to zero

0= No interrupt pending

bit 5 **SOSIF**: Start of Slave Select Interrupt Flag bit

1 = SS(in) transitioned from false to true

0 = No interrupt pending

bit 4 **EOSIF**: End of Slave Select Interrupt Flag bit

1 = SS(in) transitioned from true to false

0 = No interrupt pending

bit 3 **Unimplemented**: Read as ‘0’

bit 2 **RXOIF**: Receiver Overflow Interrupt Flag bit

1 = Data transfer completed when RXBF = 1 (edge triggered) and RXR = 1

0 = No interrupt pending

bit 1 **TXUIF**: Transmitter Underflow Interrupt Flag bit

1 = Slave Data transfer started when TXBE = 1 and TXR = 1

0 = No interrupt pending

bit 0 **Unimplemented**: Read as ‘0’

REGISTER 32-2: SPIxINTE: SPI INTERRUPT ENABLE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	U-0
SRMTIE	TCZIE	SOSIE	EOSIE	—	RXOIE	TXUIE	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7 **SRMTIE**: Shift Register Empty Interrupt Enable bit
1 = Enables the Shift Register Empty Interrupt
0 = Disables the Shift Register Empty Interrupt

bit 6 **TCZIE**: Transfer Counter is Zero Interrupt Enable bit
1 = Enables the Transfer Counter is Zero Interrupt
0 = Disables the Transfer Counter is Zero Interrupt

bit 5 **SOSIE**: Start of Slave Select Interrupt Enable bit
1 = Enables the Start of Slave Select Interrupt
0 = Disables the Start of Slave Select Interrupt

bit 4 **EOSIE**: End of Slave Select Interrupt Enable bit
1 = Enables the End of Slave Select Interrupt
0 = Disables the End of Slave Select Interrupt

bit 3 **Unimplemented**: Read as ‘0’

bit 2 **RXOIE**: Receiver Overflow Interrupt Enable bit
1 = Enables the Receiver Overflow Interrupt
0 = Disables the Receiver Overflow Interrupt

bit 1 **TXUIE**: Transmitter Underflow Interrupt Enable bit
1 = Enables the Transmitter Underflow Interrupt
0 = Disables the Transmitter Underflow Interrupt

bit 0 **Unimplemented**: Read as ‘0’

REGISTER 32-3: SPIxTCNTL – SPI TRANSFER COUNTER LSB REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7-0 TCNT[7:0]
BMODE = 0
Bits 10-3 of the Transfer Counter, counting the total number of bits to transfer
BMODE = 1
Bits 7-0 of the Transfer Counter, counting the total number of bytes to transfer

Note: This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

REGISTER 32-4: SPIxTCNTH: SPI TRANSFER COUNTER MSB REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	TCNT10	TCNT9	TCNT8
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7-3 **Unimplemented**: Read as ‘0’

bit 2-0 TCNT[10:8]:

BMODE = 0

Bits 13-11 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Bits 10-8 of the Transfer Counter, counting the total number of bytes to transfer

Note: This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

REGISTER 32-5: SPIxTWIDTH: SPI TRANSFER WIDTH REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	TWIDTH2	TWIDTH1	TWIDTH0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7-3 **Unimplemented**: Read as ‘0’

bit 2-0 TWIDTH[2:0]:

BMODE = 0

Bits 2-0 of the Transfer Counter, counting the total number of bits to transfer

BMODE = 1

Size (in bits) of each transfer counted by the transfer counter

111 = 7 bits

110 = 6 bits

101 = 5 bits

100 = 4 bits

011 = 3 bits

010 = 2 bits

001 = 1 bit

000 = 8 bits

Note: This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

REGISTER 32-6: SPIxBAUD: SPI BAUD RATE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7-0 BAUD[7:0]: Baud Clock Prescaler Select bits

SCK high or low time: TSC=SPI Clock Period*(BAUD+1)

SCK toggle frequency: FSCK=FBAUD= SPI Clock Frequency/(2*(BAUD+1))

Note: This register should not be written while the SPI is enabled (EN bit of SPIxCON0 = 1)

REGISTER 32-7: SPIxCON0: SPI CONFIGURATION REGISTER 0

R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	—	—	—	LSBF	MST	BMODE
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7 **EN**: SPI Module Enable Control bit

1 =SPI is enabled

0 = SPI is disabled,

bit 6-3 **Unimplemented**: Read as ‘0’

bit 2 **LSBF**: LSb-First Data Exchange bit

1 = Data is exchanged LSb first

0 = Data is exchanged MSb first (traditional SPI operation)

bit 1 **MST**: SPI Operating Mode Master Select bit

1 = SPI module operates as the bus master

0 = SPI module operates as a bus slave

bit 0 **BMODE**: Bit-Length Mode Select bit

1 = SPIxTWIDTH setting applies to every byte: total bits sent is SPIxTWIDTH*SPIxTCNT, end-of-packet occurs when SPIxTCNT = 0

0 = SPIxTWIDTH setting applies only to the last byte exchanged; total bits sent is SPIxTWIDTH + (SPIxTCNT*8)

Note: This register should only be written when the EN bit is cleared, or to clear the EN bit.

REGISTER 32-8: SPIxCON1: SPI CONFIGURATION REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-1/1	R/W-0/0	R/W-0/0
SMP	CKE	CKP	FST	—	SSP	SDIP	SDOP
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7 **SMP**: SPI Input Sample Phase Control bit

Slave mode:

1 = Reserved

0 = SDI input is sampled in the middle of data output time

Master mode:

1 = SDI input is sampled at the end of data output time

0 = SDI input is sampled in the middle of data output time

bit 6 **CKE**: Clock Edge Select bit

1 = Output data changes on transition from active to idle clock state

0 = Output data changes on transition from idle to active clock state

bit 5 **CKP**: Clock Polarity Select bit

1 = Idle state for SCK is high level

0 = Idle state for SCK is low level

bit 4 **FST**: Fast Start Enable bit

Slave mode:

This bit is ignored

Master mode:

1 = Delay to first SCK may be less than ½ baud period

0 = Delay to first SCK will be at least ½ baud period

bit 3 **Unimplemented**: Read as ‘0’

bit 2 **SSP**: SS Input/Output Polarity Control bit

1 = SS is active-low

0 = SS is active-high

bit 1 **SDIP**: SDI Input Polarity Control bit

1 = SDI input is active-low

0 = SDI input is active-high

bit 0 **SDOP**: SDI Output Polarity Control bit

1 = SDO output is active-low

0 = SDO output is active-high

REGISTER 32-9: SPIxCON2: SPI CONFIGURATION REGISTER 2

R-0/0	R-0/0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
BUSY(1)	SSFLT	—	—	—	SSET	TXR(2)	RXR(2)
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7 **BUSY**: SPI Module Busy Status bit(1)

1 = Data exchange is busy

0 = Data exchange is not taking place

bit 6 **SSFLT**: SS(in) Fault Status bit

If SSET = 0

1 = SS(in) ended the transaction unexpectedly, and the data byte being received was lost

0 = SS(in) ended normally

If SSET = 1

This bit is unchanged.

bit 5-3 **Unimplemented**: Read as ‘0’

bit 2 **SSET**: Slave Select Enable bit

Master mode:

1 = SS(out) is driven to the active state continuously

0 = SS(out) is driven to the active state while the transmit counter is not zero

Slave mode:

1 = SS(in) is ignored and data is clocked on all SCK(in) (as though SS = TRUE at all times)

0 = SS(in) enables/disables data input and tri-states SDO if the TRIS bit associated with the SDO pin is set (see Table 32-2 for details)

bit 1 **TXR**: Transmit Data-Required Control bit(2)

1 = TxFIFO data is required for a transfer

0 = TxFIFO data is not required for a transfer

bit 0 **RXR**: Receive FIFO Space-Required Control bit(2)

1 = Data transfers are suspended if the RxFIFO is full

0 = Received data is not stored in the FIFO

Note 1: The BUSY bit is subject to synchronization delay of up to two instruction cycles. The user must wait for it to set after loading the transmit buffer (SPIxTXB register) before using it to determine the status of the SPI module.

2: See Table 32-1 as well as **Section 32.5 “Master mode”** and **Section 32.6 “Slave Mode”** for more details pertaining to TXR and RXR function.

3: This register should not be written to while a transfer is in progress (BUSY bit of SPIxCON2 is set).

REGISTER 32-10: SPIxSTATUS: SPI STATUS REGISTER

R/C/HS-0/0	U-0	R-1/1	U-0	R/C/HS-0/0	S-0/0	U-0	R-0/0
TXWE	—	TXBE	—	RXRE	CLRBF	—	RXBF
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’ C = Clearable bit S = Settable bit HS = Bit can be set by hardware

bit 7 **TXWE**: Transmit Buffer Write Error bit
1 = SPIxTxB was written while TxFIFO was full
0 = No error has occurred

bit 6 **Unimplemented**: Read as ‘0’

bit 5 **TXBE**: Transmit Buffer Empty bit (read-only)
1 = Transmit buffer TxFIFO is empty
0 = Transmit buffer is not empty

bit 4 **Unimplemented**: Read as ‘0’

bit 3 **RXRE**: Receive Buffer Read Error bit
1 = SPIxRB was read while RxFIFO was empty
0 = No error has occurred

bit 2 **CLRBF**: Clear Buffer Control bit (write only)
1 = Reset the receive and transmit buffers, making both buffers empty
0 = Take no action

bit 1 **Unimplemented**: Read as ‘0’

bit 0 **RXBF**: Receive Buffer Full bit (read-only)
1 = Receive buffer is full
0 = Receive buffer is not full

REGISTER 32-11: SPIxRxB: SPI READ BUFFER REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7-0 **RXB[7:0]**: Receiver Buffer bits (read-only)

If RX buffer is not empty:

Contains the top-most byte of RXFIFO, and reading this register will remove the top-most byte RXFIFO and decrease the occupancy of the RXFIFO

If RX buffer is empty:

Reading this register will read as ‘0’, leave the occupancy unchanged, and set the RXRE bit of SPIxSTATUS

REGISTER 32-12: SPIxTxB: SPI TRANSMIT BUFFER REGISTER

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7-0 **TXB[7:0]**: Transmit Buffer bits (write only)

If TXFIFO is not full:

Writing to this register adds the data to the top of the TXFIFO and increases the occupancy of the TXFIFO write pointer

If TXFIFO is full:

Writing to this register does not affect the data in the TXFIFO or the write pointer, and the TXWE bit of SPIxSTATUS will be set

REGISTER 32-13: SPIxCLK: SPI CLOCK SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CLKSEL3	CLKSEL2	CLKSEL1	CLKSEL0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’

bit 7-4 **Unimplemented**: Read as ‘0’

bit 3-0 **CLKSEL[3:0]**: SPI Clock Source Selection bits

1111-1001 = Reserved
1000 = SMT_match
0111 = TMR6_Postscaled
0110 = TMR4_Postscaled
0101 = TMR2_Postscaled
0100 = TMR0_overflow
0011 = CLKREF
0010 = MFINTOSC
0001 = HFINTOSC
0000 = FOSC

TABLE 32-3: SUMMARY OF REGISTERS ASSOCIATED WITH SPI

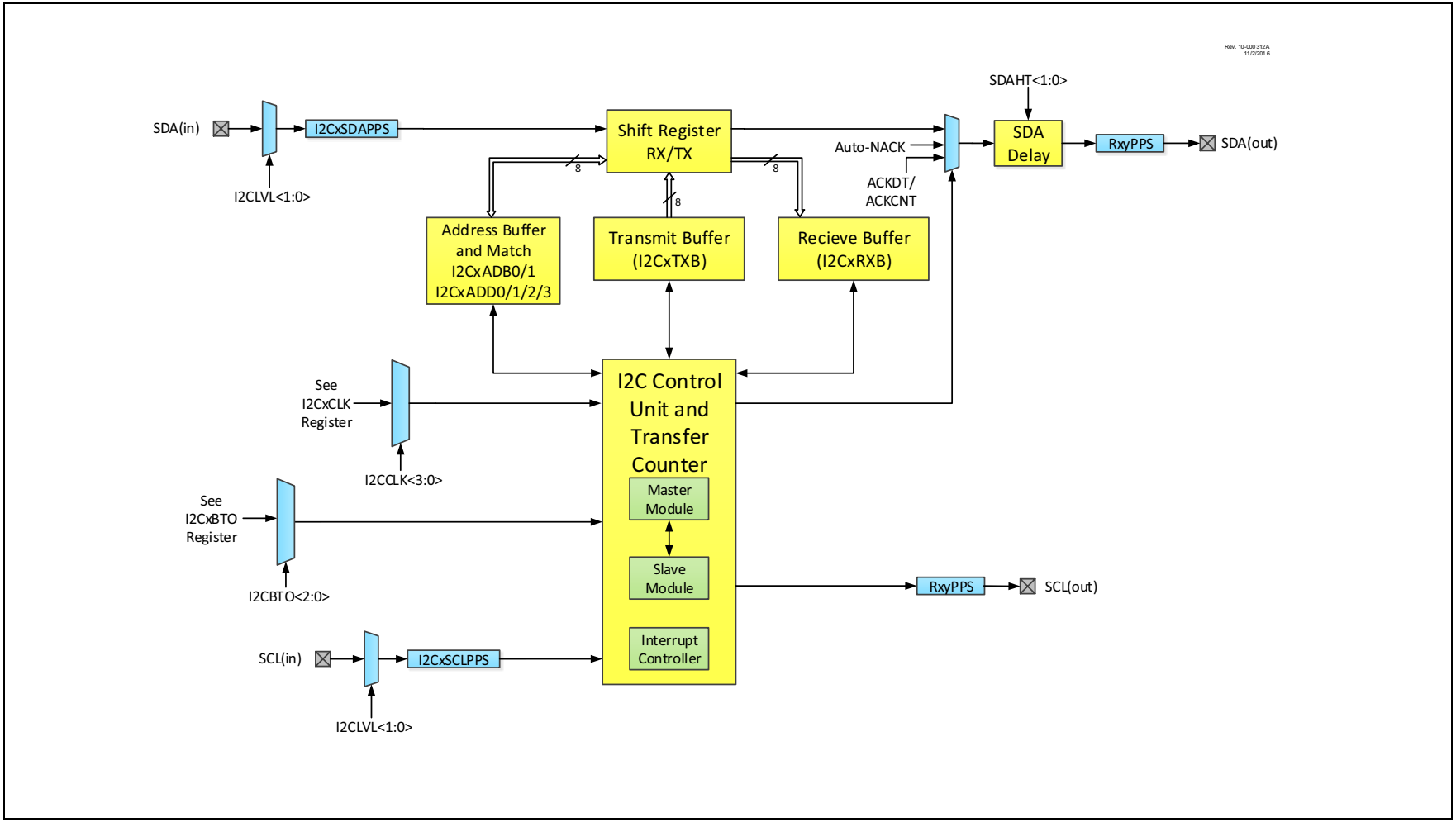
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
SPIxINTF	SRMTIF	TCZIF	SOSIF	EOSIF	—	RXOIF	TXUIF	—	536
SPIxINTE	SRMTIE	TCZIE	SOSIE	EOSIE	—	RXOIE	TXUIE	—	537
SPIxTCNTH	—	—	—	—	—	TCNT10	TCNT9	TCNT8	538
SPIxTCNTL	TCNT7	TCNT6	TCNT5	TCNT4	TCNT3	TCNT2	TCNT1	TCNT0	537
SPIxTWIDTH	—	—	—	—	—	TWIDTH2	TWIDTH1	TWITDH0	538
SPIxBAUD	BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0	539
SPIxCON0	EN	—	—	—	—	LSBF	MST	BMODE	539
SPIxCON1	SMP	CKE	CKP	FST	—	SSP	SDIP	SDOP	540
SPIxCON2	BUSY	SSFLT	—	—	—	SSET	TXR	RXR	541
SPIxSTATUS	TXWE	—	TXBE	—	RXRE	CLRBF	—	RXBF	542
SPIxRXB	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	542
SPIxTXB	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0	543
SPIxCLK	—	—	—	—	CLKSEL3	CLKSEL2	CLKSEL1	CLKSEL0	543

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the SPI module.

33.0 I²C MODULE

The device has two dedicated, independent I²C modules. Figure 33-1 is a block diagram of the I²C interface module. The figure shows both the Master and Slave modes together.

FIGURE 33-1: I²C MODULE BLOCK DIAGRAM



© 2017-2020 Microchip Technology Inc.

33.1 I²C Features

- Inter-Integrated Circuit (I²C) interface supports the following modes in hardware:
  - Master mode
  - Slave mode with byte NACKing
  - Multi-Master mode
- Dedicated Address, Receive and Transmit buffers
- Up to four Slave addresses matching
- General Call address matching
- 7-bit and 10-bit addressing with masking
- Start, Restart, Stop, Address, Write, and $\overline{ACK}$ Interrupts
- Clock Stretching hardware for:
  - RX Buffer Full
  - TX Buffer Empty
  - After Address, Write, and $\overline{ACK}$
- Bus Collision Detection with arbitration
- Bus Timeout Detection
- SDA hold time selection
- I²C, SMBus 2.0, and SMBus 3.0 input level selections

33.2 I²C Module Overview

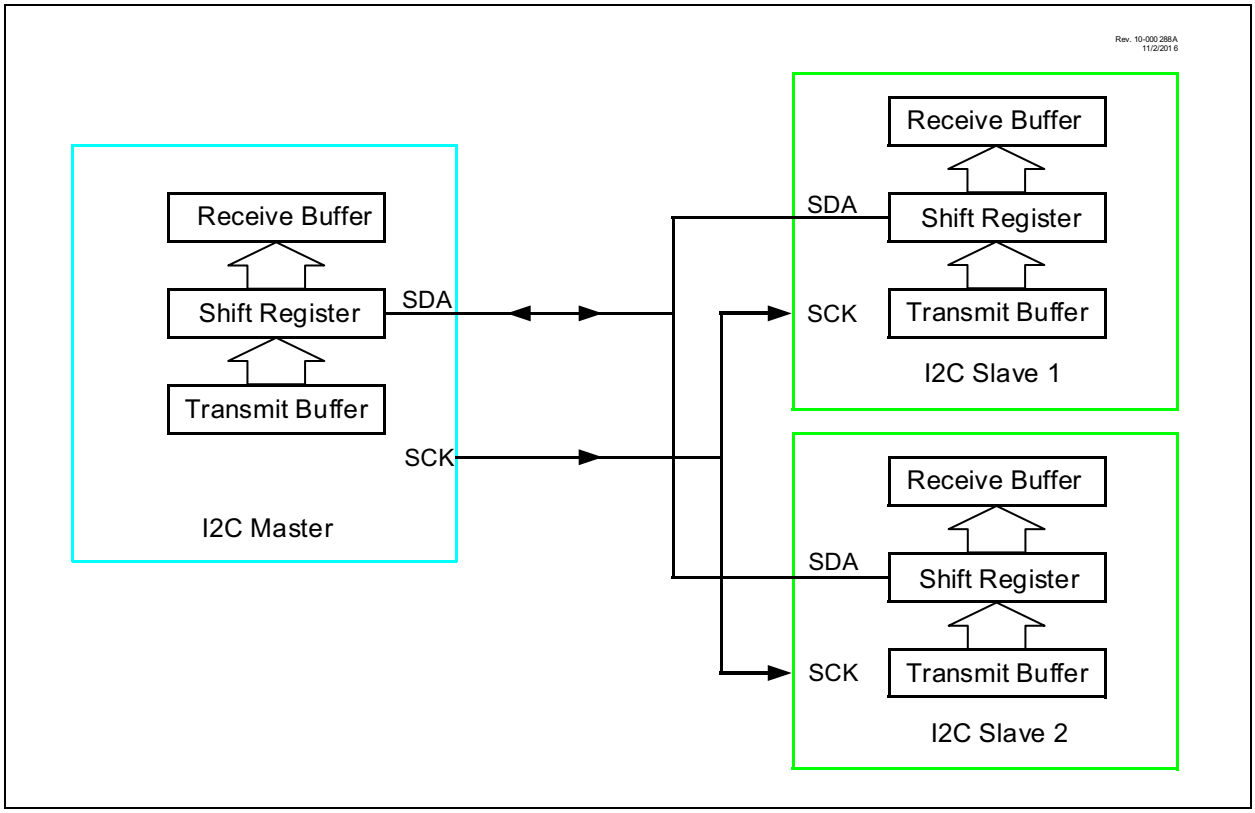
The I²C module provides a synchronous interface between the microcontroller and other I²C-compatible devices using the two-wire I²C serial bus. Devices communicate in a master/slave environment. The I²C bus specifies two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors to the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one. Every transaction on the I²C bus has to be initiated by the Master.

Figure 33-2 shows a typical connection between a master and more than one slave.

FIGURE 33-2: I²C MASTER/SLAVE CONNECTIONS



There are four main operations based on the direction of the data being shared during I²C communication.

- Master Transmit (master is transmitting data to a slave)
- Master Receive (master is receiving data from a slave)
- Slave Transmit (slave is transmitting data to a master)
- Slave Receive (slave is receiving data from the master)

To begin any I²C communication, the master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an $\overline{\text{ACK}}$. The master then continues to shift data in or out of the slave until it terminates the message with a Stop.

Further details about the I²C module are discussed in the section below.

33.3 I²C Mode Operation

All I²C communication is 8-bit data and 1-bit acknowledge and shifted out MSb first. The user can control the interaction between the software and the module using several control registers and interrupt flags. Two pins, SDA and SCL, are exercised by the module to communicate with other external I²C devices.

33.3.1 DEFINITION OF I²C TERMINOLOGY

The I²C communication protocol terminologies are defined for reference below in Table 33-1. These terminologies are used throughout this document. Table 33-1 has been adapted from the Phillips I²C specification.

TABLE 33-1: I²C BUS TERMS

TERM	Description
Transmitter	The device which shifts data out onto the bus
Receiver	The device which shifts data in from the bus
Master	The device that initiates a transfer, generates clock signals and terminates a transfer
Slave	The device addressed by the master
Multi-master	A bus with more than one device that can initiate data transfers
Arbitration	Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted
Synchronization	Procedure to synchronize the clocks of two or more devices on the bus.
Idle	No master is controlling the bus, and both SDA and SCL lines are high
Active	Any time one or more master devices are controlling the bus
Addressed Slave	Slave device that has received a matching address and is actively being clocked by a master
Matching Address	Address byte that is clocked into a slave that matches the value stored in I2CxADR
Write Request	Slave receives a matching address with R/$\overline{\text{W}}$ bit clear and is ready to clock in data
Read Request	Master sends an address byte with the R/$\overline{\text{W}}$ bit set, indicating that it wishes to clock data out of the Slave. This data is the next and all following bytes until a Restart or Stop.
Clock Stretching	When a device on the bus holds SCL low to stall communication
Bus Collision	Any time the SDA line is sampled low by the module while it is outputting and expected high state.
Bus Timeout	A device holds the bus longer than specified by the I2CxBTO register, causing a module Reset.

33.3.2 BYTE FORMAT

All communication in I^{2}C is done in 9-bit segments. A byte is sent from a master to a slave or vice-versa, followed by an Acknowledge bit sent by the receiver. After the 8th falling edge of the SCL line, the device transmitting data on the SDA line releases control of that pin to an input, and reads in an acknowledge value on the next clock pulse. The clock signal is provided by the master. Data is valid to change while the SCL line is low, and sampled on the rising edge of the clock. Changes on the SDA line while the SCL line is high define Start and Stop conditions on the bus which are explained further in the chapter.

33.3.3 SDA AND SCL PINS

The user must configure these pins as open-drain inputs. This is done by clearing the appropriate TRIS bits and setting the appropriate and ODCON bits. The user may also select the input threshold, slew-rate and internal pull-up settings using the RxyI2C control registers (Register 16-9).

33.3.4 SDA HOLD TIME

The hold time of the SDA pin is selected by the SDAHT[1:0] bits of the I2CxCON2 register. Hold time is the time SDA is held valid after the falling edge of SCL. A longer hold time setting may help on buses with large capacitance.

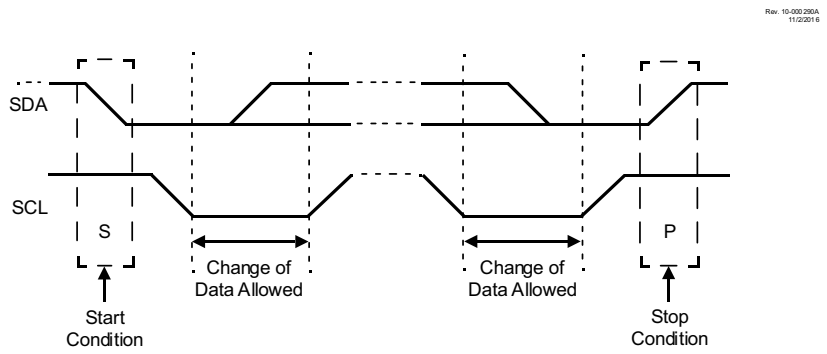
33.3.5 START CONDITION

The I^{2}C specification defines a Start condition as a transition of SDA line from a high to a low state while SCL line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an Active state. Figure 33-3 shows waveforms for Start conditions. Master hardware waits for the BFRE bit of I2CxSTAT0 to be set, before asserting a Start condition on the SCL and SDA lines. If two masters assert a start at the same time, a collision will occur during the addressing phase.

33.3.6 STOP CONDITION

A Stop condition is a transition of the SDA line from low to high while the SCL line is high. Figure 33-3 shows waveforms for Stop conditions.

FIGURE 33-3: START AND STOP CONDITIONS



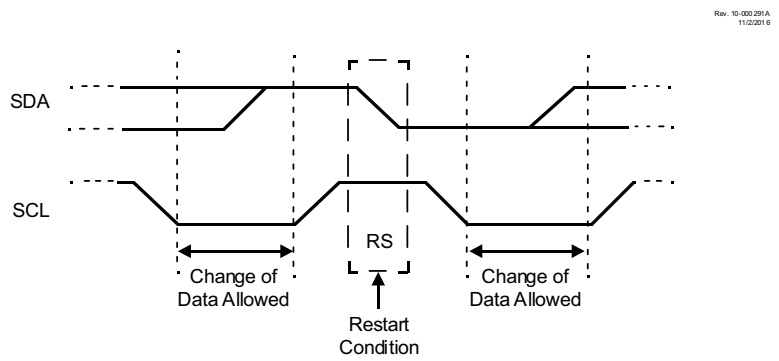
Note: At least one SCL low time must appear before a Stop is valid. Therefore if the SDA line goes low then high again while the SCL line is high, only the Start condition is detected.

33.3.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave. Figure 33-4 shows the waveform for a Restart condition.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes (SMA = 1), the master can issue a Restart and the high address byte with the R/$\overline{W}$ bit set. The slave logic will then hold the clock and prepare to clock out data.

FIGURE 33-4: RESTART CONDITION



33.3.8 ACKNOWLEDGE SEQUENCE

The ninth SCL pulse for any transferred byte in I^2C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge ($\overline{ACK}$) is an active-low signal, pulling the SDA line low indicates to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an $\overline{ACK}$ is placed in the ACKSTAT bit of the I2CxCON1 register. The ACKSTAT bit is cleared when the receiving device sends an Acknowledge and is set when the receiving device does not Acknowledge. A slave sends an Acknowledge when it has recognized its address. When in a mode that is receiving data, the $\overline{ACK}$ data being sent to the transmitter depends on the value of I2CxCNT register. ACKDT is the value sent when I2CxCNT! = 0. When I2CxCNT = 0, the ACKCNT value is used instead.

In Slave mode, if the ADRIE or WRIE bits are set, clock stretching is initiated when there is an address match or when there is an attempt to write to slave. This allows the user to set the $\overline{ACK}$ value sent back to the transmitter. The ACKDT bit of the I2CxCON1 register is set/cleared to determine the response. Slave hardware will generate an $\overline{ACK}$ response if the ADRIE or WRIE bits are clear.

Certain conditions will cause a not-ACK (NACK) to be sent automatically. If any of the RXRE, TXWE, RXO, or TXU bits is set, the hardware response is forced to NACK. All subsequent responses from the device for address matches or data will be a NACK response.

33.3.9 BUS TIME-OUT

The I2CxBTO register can be used to select the timeout source for the module. The I^2C module is reset when the selected bus time out signal goes high. This feature is useful for SMBus and PMBus™ compatibility.

For example, Timer2 can be selected as the bus timeout source and configured to count when the SCL pin is low. If the timer runs over before the SCL pin transitioned high, the timer-out pulse will reset the module.

If the module is configured as a slave and a BTO event occurs when the slave is active, i.e., the SMA bit is set, the module is immediately reset. The SMA and CSTR bits are also cleared, and the BTOIF bit is set.

If a BTO event occurs when the module is configured as a master and is active, (i.e., MMA bit is set), and the module immediately tries to assert a Stop condition and also sets the BTOIF bit. The actual generation of the Stop condition may be delayed if the bus is been clock stretched by some slave device. The MMA bit will be cleared only after the Stop condition is generated.

33.3.10 ADDRESS BUFFERS

The I^2C module has two address buffer registers, I2CxADB0 and I2CxADB1. Depending on the mode, these registers are used as either receive or transmit address buffers. See Table 33-2 for data flow directions in these registers. In Slave modes, these registers are only updated when there is an address match. The ADB bit in the I2CxCON2 register is used to enable/disable the address buffer functionality. When disabled, the address data is sourced from the transmit buffer and is stored in the receive buffer.

TABLE 33-2: ADDRESS BUFFER DIRECTION AS PER I^2C MODE

Modes	MODE[2:0]	I2CxADB0	I2CxADB1
Slave (7-bit)	000	RX	—
	001	RX	—
Slave (10-bit)	010	RX	RX
	011	RX	RX
Master (7-bit)	100	—	TX
Master (10-bit)	101	TX	TX
Multi-Master (7-bit)	110	RX	TX
	111	RX	TX

33.3.10.1 Slave Mode (7-bit)

In 7-bit Slave mode, I2CxADB0 is loaded with the received matching address and R/$\overline{W}$ data. The I2CxADB1 register is ignored in this mode.

33.3.10.2 Slave Mode (10-bit)

In 10-bit Slave mode, I2CxADB0 is loaded with the lower eight bits of the matching received address. I2CxADB1 is loaded with full eight bits of the high address byte, including the R/$\overline{W}$ bit.

33.3.10.3 Master Mode (7-bit)

The I2CxADB0 register is ignored in this mode. In 7-bit Master mode, the I2CxADB1 register is used to copy address data byte, including the R/$\overline{W}$ value, to the Shift register.

33.3.10.4 Master Mode (10-bit)

In 10-bit Master mode, the I2CxADB0 register stores the low address data byte value that will be copied to the Shift register after the high address byte is shifted out. The I2CxADB1 register stores the high address byte value that will be copied to the Shift register. It is up to the user to specify all eight of these bits, even though the I^2C specification defines the upper five bits as a constant.

33.3.10.5 Multi-Master Mode (7-bit only)

In Multi-Master mode, the device can be both master and slave depending on the sequence of events on the bus. If being addressed as a slave, the I2CxADB0 register stores the received matching slave address byte. If the device is trying to communicate as a master on the bus, the contents of the I2CxADB1 register are copied to the Shift register for addressing a slave device.

33.3.11 RECEIVE AND TRANSMIT BUFFER

The receive buffer holds one byte of data while another is shifted into the SDA pin. The user can access the buffer by software (or DMA) through the I2CxRXB register. When new data is loaded into the I2CxRXB register, the receive buffer full Status bit (RXBF) is set and reading the I2CxRXB register clears this bit.

If the user tries to read I2CxRXB when it is empty (i.e., RXBF = 0), receive read error bit (RXRE) is set and a NACK will be generated. The user must clear the error bit to resume normal operation.

The transmit buffer holds one byte of data while another can be shifted out through the SDA pin. The user can access the buffer by software (or DMA) through the I2CxTXB register. When the I2CxTXB does not contain any transmit data, the transmit buffer empty status bit (TXBE) is set. At this point, the user can load another byte into the buffer.

If the user tries to write I2CxTXB when it is NOT empty (i.e. TXBE = 0), transmit write error flag bit (TXWE) is set and the new data is discarded. When TXWE is set, the user must clear this error condition to resume normal operation.

By setting the CLRBF bit in the I2CxSTAT1 register, the user can clear both receive and transmit buffers. CLRBF will also clear the I2CxRXIF and I2CxTXIF bits.

33.3.12 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Since the SCL connection is open-drain, the slave has the ability to hold the line low until it is ready to continue communicating. Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

Clock stretching can be enabled or disabled by the clearing or setting of CSD (clock stretching disable) bit in the I2CxCON1 register. This bit is valid only in the Multi-Master and Slave modes of operation.

33.3.12.1 Clock Stretching for Buffer Operations

If enabled, clock stretching is forced during buffer read/write operations. For example, in Slave mode if RXBF = 1 (receive buffer full), the clock will be stretched after the seventh falling edge of SCL. The SCL line is released only after the user reads data from the receive buffer. This ensures that there is never a receive data overflow. In this situation, if clock stretching is disabled, the RXO bit in I2CxCON1 is set indicating a receive overflow. When set, the module will always respond with a NACK.

Similarly, when TXBE = 1 (transmit buffer empty) and I2CxCNT! = 0, the clock is stretched after the 8th falling edge of SCL. The SCL line is released only after the user loads new data into the transmit buffer. This ensures that there is never a transmit underflow. In this situation, if clock stretching is disabled, the TXU bit in I2CxCON1 is set indicating a transmit underflow. When set, the module will always respond with a NACK.

33.3.12.2 Clock Stretching for Other Slave Operations

There are three Interrupt and Hold bits that provide clock stretching in Slave mode. These bits can also be used in conjunction with the I2CxIE bit in PIRx register to generate system level interrupts.

- Incoming address match interrupt
  - Clock stretching after an incoming matching address byte is enabled by the Address Interrupt and Hold (ADRIE) bit of the I2CxPIE register. When ADRIE = 1, the CSTR bit is set and the SCL line is stretched following the 8th falling edge of SCL of a received matching address. This allows the user to read the received address from the I2CADB0/1 registers and selectively $\overline{\text{ACK}}$/NACK based on the received address. Clock stretching from ADRIE is released by software clearing the CSTR bit.
- Data Write Interrupt
  - The data write interrupt and hold enable (WRIE) bit is used to enable clock stretching after a received data byte. When WRIE = 1, the CSTR bit is set, and the SCL line is stretched, following the 8th falling SCL edge for incoming slave data. This bit allows user software to selectively $\overline{\text{ACK}}$/NACK each received data byte. Clock stretching from WRIE is released by software clearing the CSTR bit.
- Acknowledge status
  - The acknowledge status time interrupt and hold enable (ACKTIE) bit is used to enable clock stretching after the $\overline{\text{ACK}}$ phase of a transmission. This bit enables clock stretching for all address/data transactions; address, write, or read. Following the $\overline{\text{ACK}}$, the slave hardware will set CSTR. Clock stretching from ACKTIE is released by software clearing the CSTR bit.

33.3.13 DATA BYTE COUNT

The I2CxCNT register is used to specify the number of bytes in a complete I^2C packet. The value in this register will decrement every time a data byte is received or transmitted from the I^2C module. The I2CxCNT register will not decrement past zero.

If a byte transfer causes the I2CxCNT register to decrement to zero, the Count Interrupt Flag bit (CNTIF) in I2CxPIR is set. This flag bit is set on the 9th falling edge of SCL for transmit and receive operations.

The I2CxCNT register can be auto-loaded if the ACNT bit in the I2CxCON2 register is set. When ACNT bit is set, the data byte following the address byte is loaded into the I2CxCNT register.

Note 1:	I2CxCNT decrements on the eighth (receive) or ninth (transmit) falling edge of SCL; writes during this bit time can corrupt the value.
2:	If the block size of the message is greater than 255, the I2CxCNT register can be updated mid-message to prevent decrement to zero.

33.4 I^2C Slave Mode

The I^2C Slave mode operates in one of four modes selected in the Mode bits of I2CxCON0. The modes can be divided into 7- and 10-bit Addressing modes. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

33.4.1 SLAVE ADDRESSING MODES

The I2CxADR/1/2/3 registers contain the Slave mode addresses. The first byte received after a Start or Restart condition is compared against the values stored in these registers. If the byte matches a value, it is loaded into the I2CxADB0/1 registers. If the value does not match, there is no response from the module. The I^2C module can be configured in the following Slave configurations.

33.4.1.1 7-bit Addresses Mode

In this mode, the LSb of the received data byte is ignored when determining if there is an address match. All four I2CxADR registers are independently compared to the received address byte.

33.4.1.2 7-bit Addresses with Masking

In this mode, the value in I2CxADR0 is masked with the value in I2CxADR1 to determine if an address match occurred. A second address and mask are also compared from I2CxADR2/3. When Mode[2:0] = `001` or `111`, the I2CxADR1/3 registers serve as the mask value for I2CxADR0/2. All seven bits of the address can be masked

33.4.1.3 10-bit Addresses

In this mode, the values stored in I2CxADR0 and I2CxADR1 registers are used to create a 10-bit address. A second 10-bit compare address is formed from I2CxADR2 and I2CxADR3.

33.4.1.4 10-bit Address with Masking

In this mode, the I2CxADR0/1 registers are used to form a 10-bit address, and the I2CxADR2/3 registers are used to form a 10-bit mask for that address. When MODE[2:0] = `011`, the I2CxADR2/3 registers serve as the mask value for the 10-bit address stored in I2CxADR0/1.

> **Note:** Even though 10-bit addressing calls out only 10-bits used in the address comparison, all 15 address bits in I2CxADR0/1 are compared in these modes.

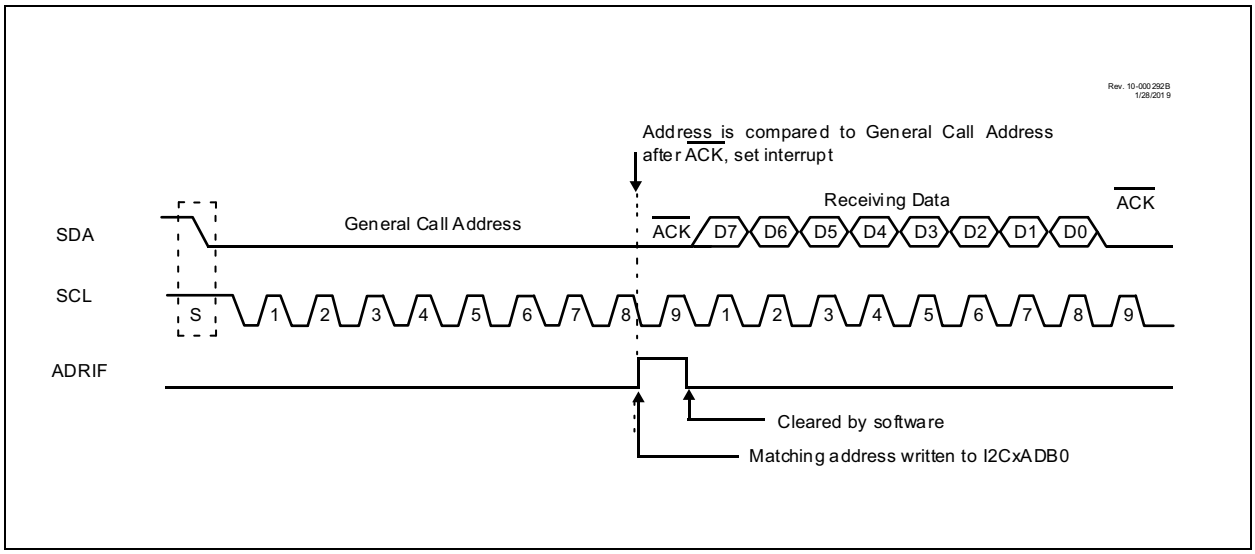
33.4.2 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I^{2}C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master device. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an $\overline{\text{ACK}}$. The general call address is a reserved address in the I^{2}C protocol, defined as address `0x00`. In order for the slave hardware to $\overline{\text{ACK}}$ this address, it must be enabled by setting the GCEN bit in the I2CxCON2 register. Setting one of the I2CxADR0/1/2/3 registers to `0x00` is not required. Figure 33-5 shows a General Call reception sequence.

If the ADRIE bit is set, the module will clock stretch after the eighth SCL pulse just like any other address match.

> **Note:** General Call addressing is supported in only 7-bit Addressing modes

FIGURE 33-5: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE



33.4.3 SLAVE OPERATION IN 7-BIT ADDRESSING MODE

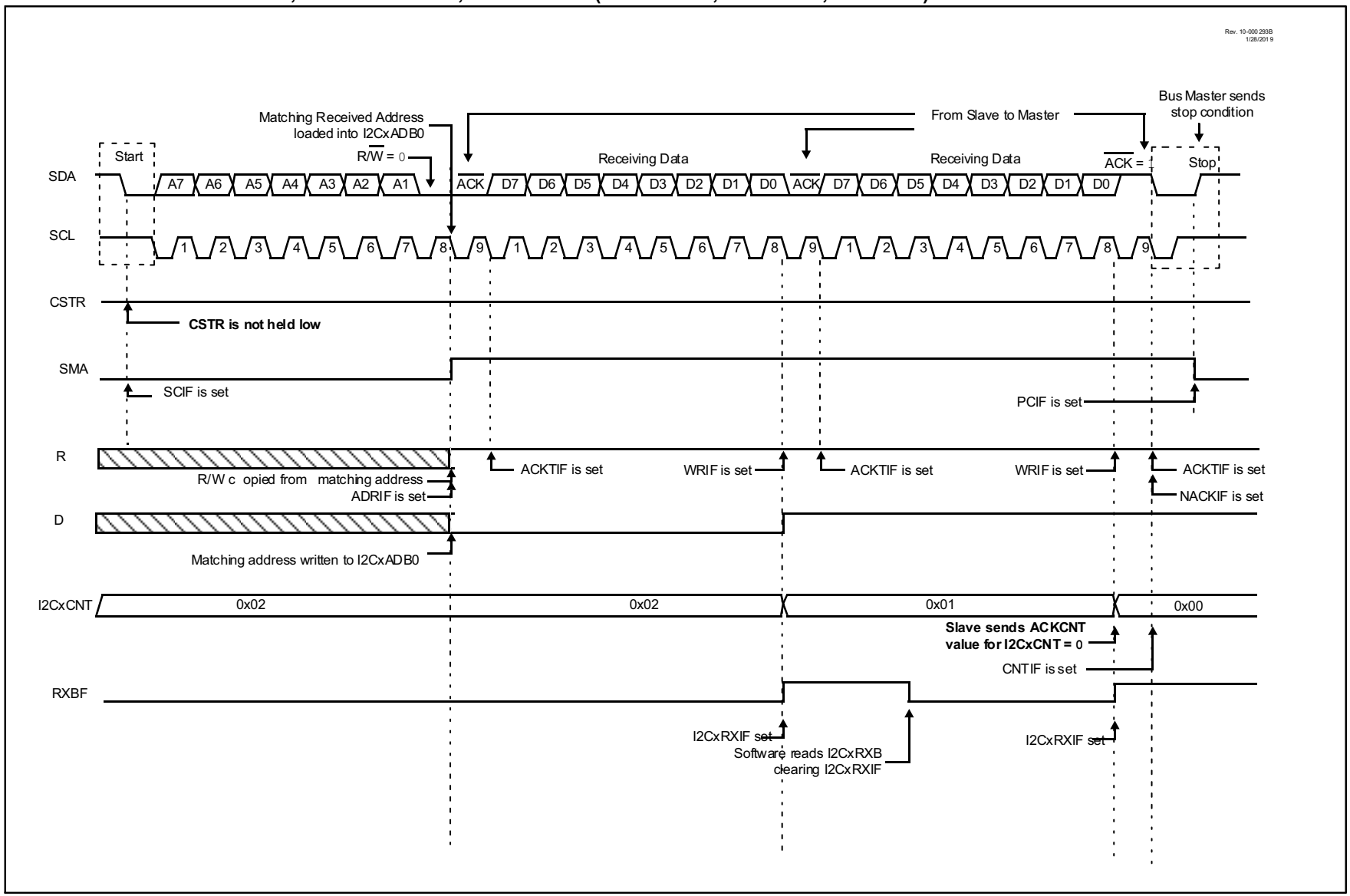
The Least Significant bit (LSb) in an address byte transmitted by the master is used to determine if the Master wants to read from or write to the Slave device. If set, it denotes that the Master wants to read from the slave and if cleared it means the master wants to write to the slave device. If there is an address match, the R/$\overline{W}$ bit is copied to the R bit of the I2CxSTAT0 register.

33.4.3.1 Slave Reception (7-bit Addressing Mode)

This section describes the sequence of events for the I^2C module configured as an I^2C slave in 7-bit Addressing mode and is receiving data. Figure 33-6, Figure 33-7, and Figure 33-8 are used as a visual reference for this description.

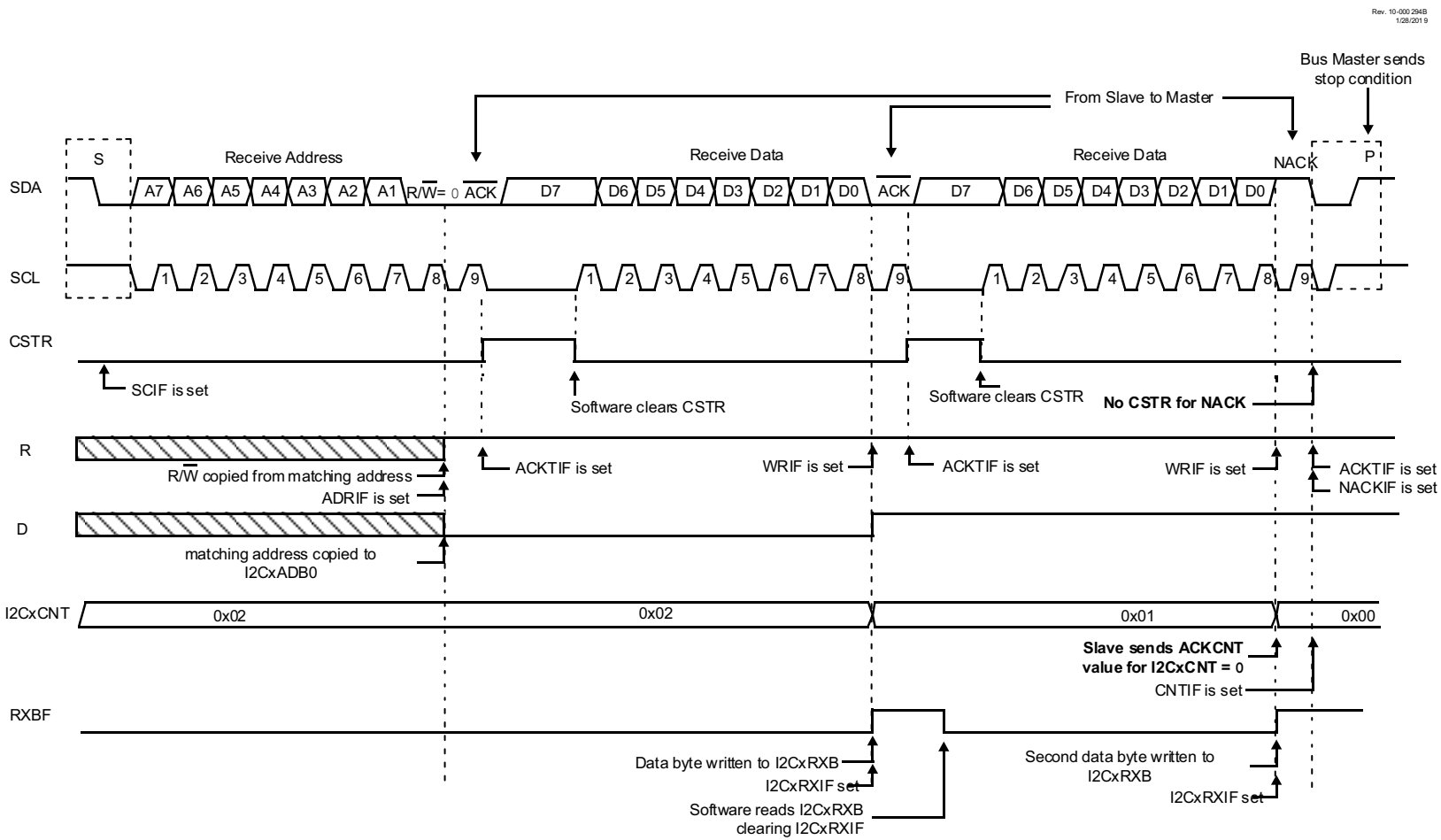
1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set.
2. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
3. Master transmits eight bits – 7-bit address and R/$\overline{W}$ = 0.
4. Received address is compared with the values in I2CxADR0/I2CxADR1/I2CxADR2/I2CxADR3 registers. Refer to section **Section 33.4.1 "Slave Addressing Modes"** for slave addressing modes.
5. If address matches; SMA in I2CxSTAT0 register is set, R/$\overline{W}$ is copied to R bit, D bit is cleared. If the address does not match; module becomes idle.
6. The matched address data is loaded into I2CxADB0 (If ABD = 0) or I2CxRXB (if ABD = 1) and ADRIF in I2CxPIR register is set.
7. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set. Slave software can read address from I2CxADB0 and set/clear ACKDT before releasing SCL.
8. If there are any previous error conditions, e.g., Receive buffer overflow or transmit buffer underflow errors, Slave will force a NACK and the module becomes idle.
9. ACKDT value is copied out to SDA for $\overline{ACK}$ pulse to be read by the Master on the 9th SCL pulse.
10. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set, then Slave software can read address from I2CxADB0 register and change the value of ACKDT before releasing SCL by clearing CSTR.
11. Master sends first seven SCL pulses of the data byte or a Stop condition (in the case of NACK).
12. If Stop condition; PCIF in I2CxPIR register is set, module becomes idle.
13. If the receive buffer is full from the previous transaction i.e. RXBF = 1 (I2CxRXIF = 1), CSTR is set. Slave software must read data out of I2CxRXB to resume communication.
14. Master sends 8th SCL pulse of the data byte. D bit is set, WRIF is set.
15. I2CxRXB is loaded with new data, RXBF bit is set, I2CxRXIF is set.
16. If Data write interrupt and hold is enabled (WRIE = 1), CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB and set/clear ACKDT before releasing SCL by clearing CSTR.
17. If I2CxCNT = 0, the ACKCNT value is output to the SDA; else, if I2CxCNT!= 0, the ACKDT value is used and the value of I2CxCNT is decremented.
18. The $\overline{ACK}$ value is copied out to SDA to be read by the Master on the 9th SCL pulse.
19. If I2CxCNT = 0, CNTIF is set.
20. If a NACK was sent, NACKIF is set, module becomes idle.
21. If ACKTIE = 1, CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB clearing RXBF, before releasing SCL by clearing CSTR.
22. Go to step 11.

FIGURE 33-6: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (ACKTIE = 0, ADRIE = 0, WRIE = 0)



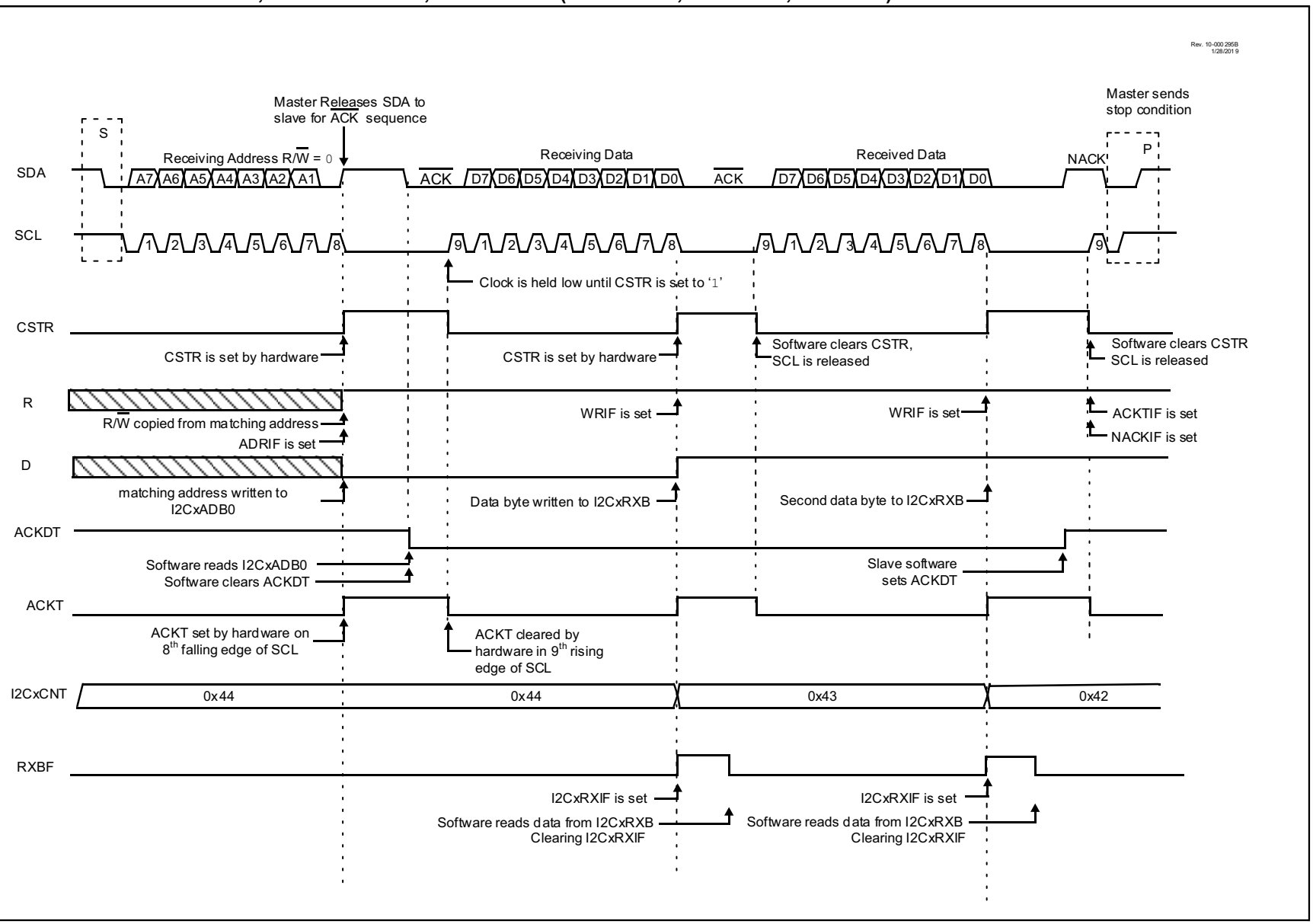
© 2017-2020 Microchip Technology Inc.

FIGURE 33-7: I²C SLAVE, 7-BIT ADDRESS, RECEPTION WITH I2CxCNT (ACKTIE = 1, ADRIE = 0, WRIE = 0)



© 2017-2020 Microchip Technology Inc.

FIGURE 33-8: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (ACKTIE = 0, ADRIE = 1, WRIE = 1)



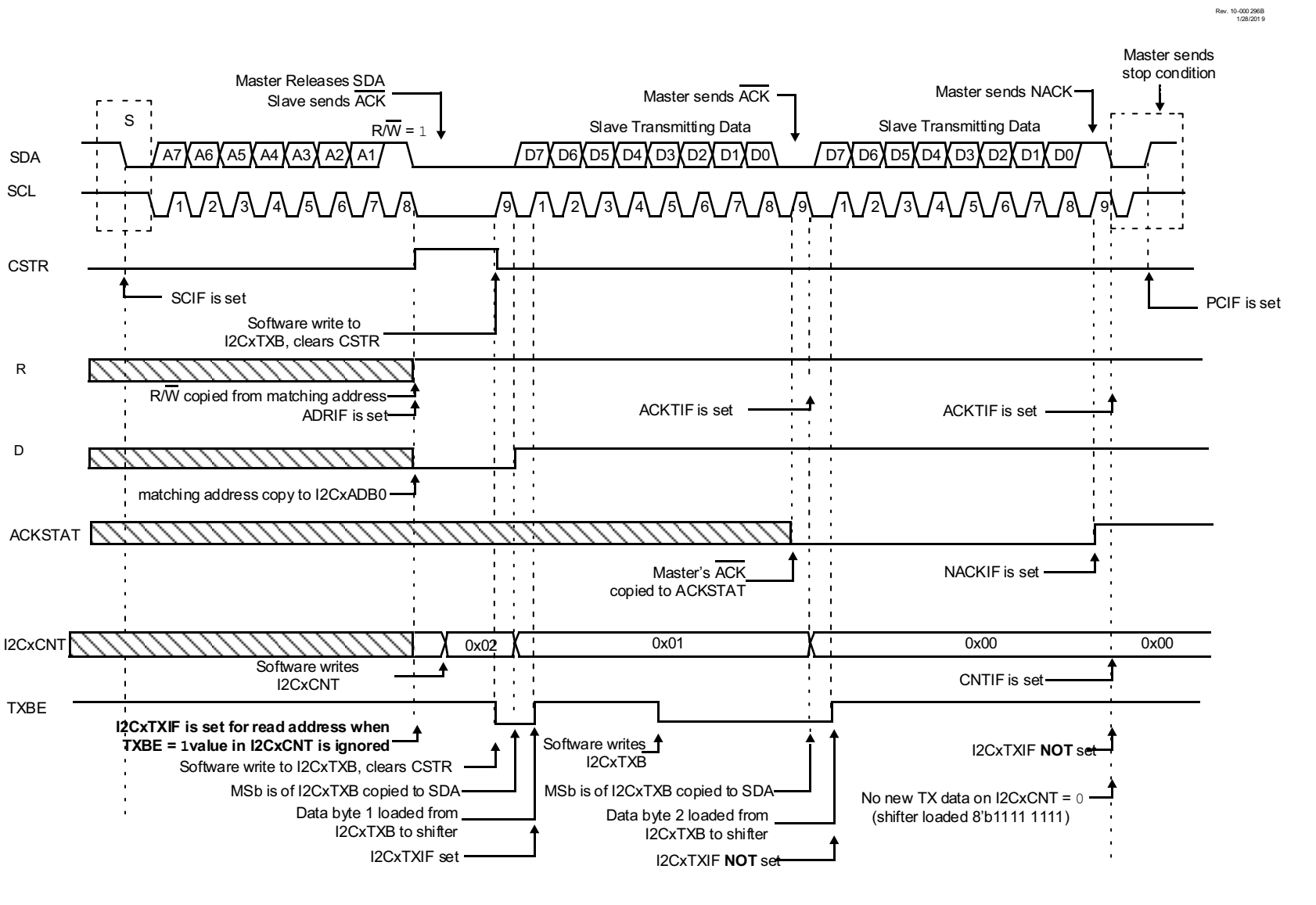
© 2017-2020 Microchip Technology Inc.

33.4.3.2 Slave Transmission (7-bit Addressing Mode)

This section describes the sequence of events for the I^2C module configured as an I^2C slave in 7-bit Addressing mode and is transmitting data. Figure 33-9 and Figure 33-10 are used as a visual reference for this description.

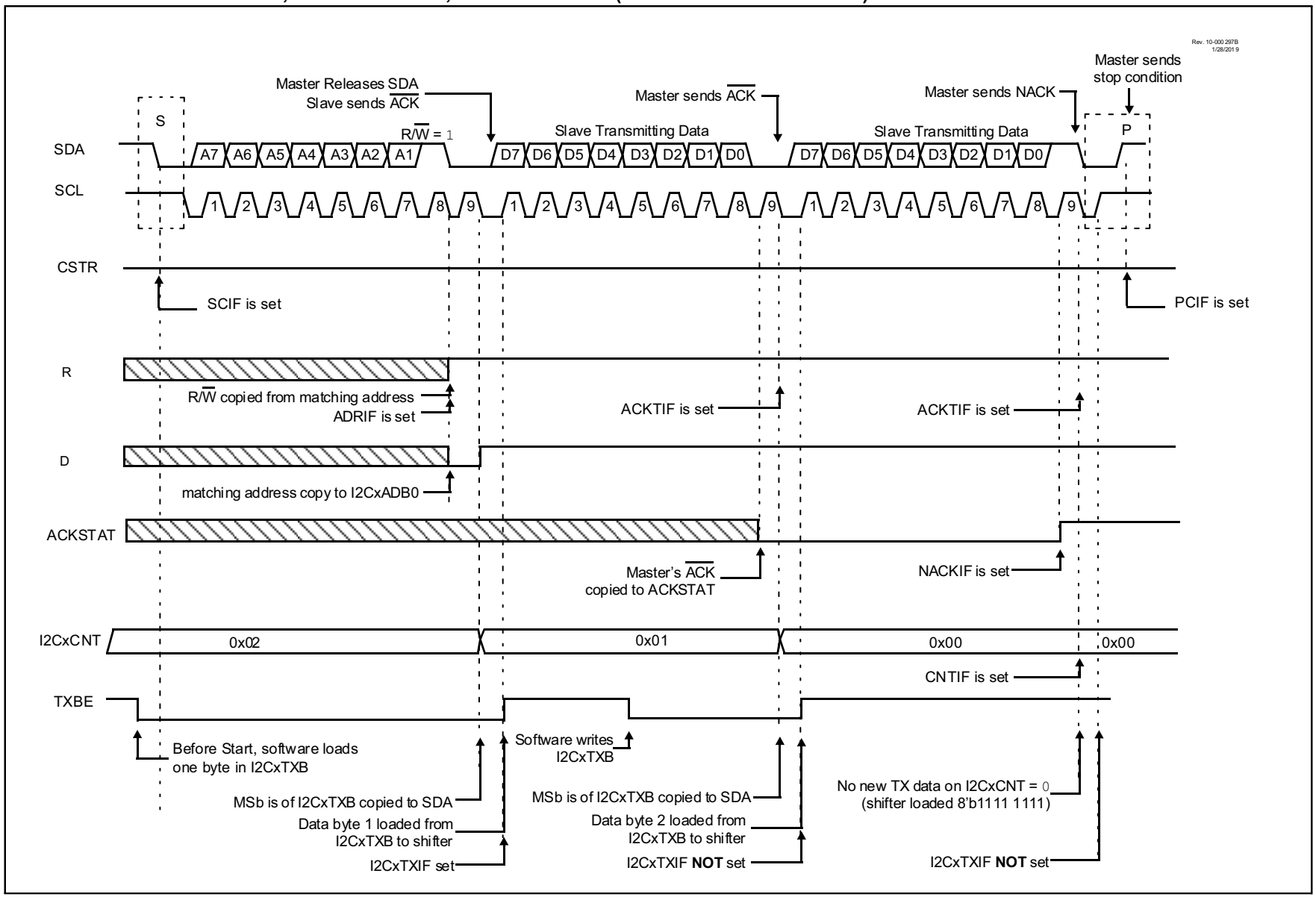
1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set.
2. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
3. Master transmits eight bits – 7-bit address and R/$\overline{W}$ = 1.
4. Received address is compared with the values in I2CxADR0/I2CxADR1/I2CxADR2/I2CxADR3 registers. Refer to **Section 33.4.1 "Slave Addressing Modes"** for Slave Addressing modes.
5. If address matches; SMA in I2CxSTAT0 register is set, R/$\overline{W}$ is copied to R bit, D bit is cleared. If the address does not match; module becomes idle.
6. The matched address data is loaded into I2CxADB0 (If ABD = 0) or I2CxRXB (if ABD = 1) and ADRIF in I2CxPIR register is set.
7. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set. Slave software can read address from I2CxADB0 and set/clear ACKDT before releasing SCL. SCL line can be released by clearing CSTR.
8. If the transmit buffer is empty from the previous transaction, i.e. TXBE = 1 and I2CxCNT!= 0 (I2CxTXIF = 1), CSTR is set. Slave software must load data into I2CxTXB to release SCL. I2CxCNT decrements after the byte is loaded into the Shift register.
9. Slave hardware waits for 9th SCL pulse with $\overline{ACK}$ data from Master.
10. If I2CxCNT = 0, CNTIF is set.
11. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set.
12. Slave software can change the value of ACKDT before releasing SCL by clearing CSTR.
13. Master sends eight SCL pulses to clock out data or asserts a Stop condition to end the transaction.
14. Go to step 8.

FIGURE 33-9: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION



© 2017-2020 Microchip Technology Inc.

FIGURE 33-10: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (NO CLOCK STRETCHING)



© 2017-2020 Microchip Technology Inc.

33.4.3.3 Slave operation in 10-bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of '11110A9A80'. A9 and A8 are the two MSb of the 10-bit address. The first byte is compared with the value in I2CxADR1 and I2CxADR3 registers. After the high byte is acknowledged, the low address byte is clocked in and all eight bits are compared to the low address value in the I2CxADR0 and I2CxADR2 registers. A high and low address match as a write request is required at the start of all 10-bit addressing communication. To initiate a read, the Master needs to issue a Restart once the slave is addressed and clock in the high address with the R/$\overline{W}$ bit set. The slave hardware will then acknowledge the read request and prepare to clock out data. The SMA (slave active) bit is set only when both the high and low address bytes match.

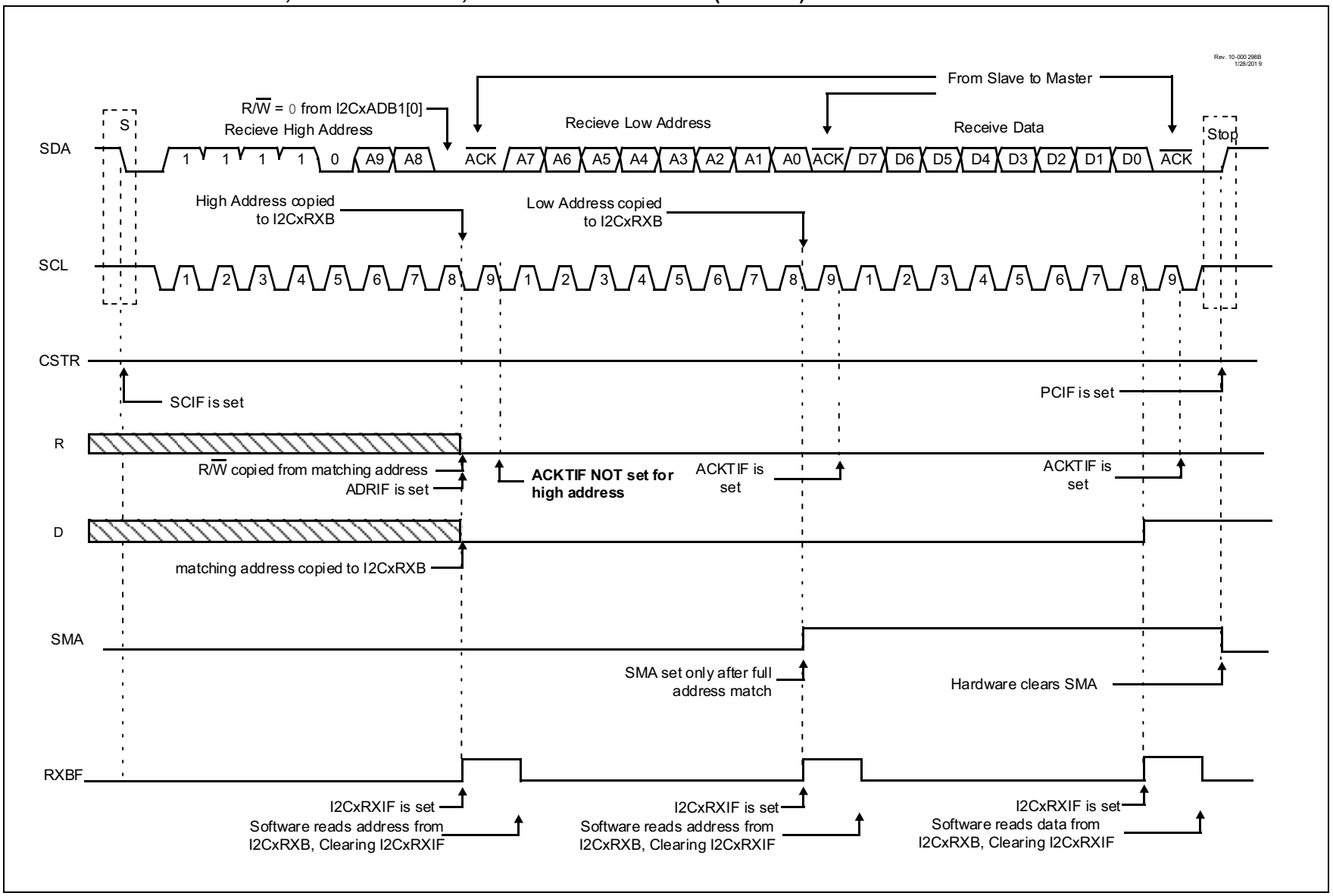
Note:	All seven bits of the received high address are compared to the values in the I2CxADR1 and I2CxADR3 registers. The five-bit '11110' high address format is not enforced by module hardware. It is up to the user to configure these bits correctly.

33.4.3.4 Slave Reception (10-bit Addressing Mode)

This section describes the sequence of events for the I^2C module configured as an I^2C slave in 10-bit Addressing mode and is receiving data. Figure 33-11 is used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
2. Master transmits high address byte with R/$\overline{W}$ = 0.
3. The received high address is compared with the values in I2CxADR1 and I2CxADR3 registers.
4. If high address matches; R/$\overline{W}$ is copied to R bit, D bit is cleared, high address data is copied to I2CxADB1. If the address does not match; module becomes idle.
5. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set.
6. Slave software can read high address from I2CxADB1 and set/clear ACKDT before releasing SCL.
7. ACKDT value is copied out to SDA for $\overline{ACK}$ pulse. SCL line is released by clearing CSTR.
8. Master sends ninth SCL pulse for $\overline{ACK}$.
9. Slave can force a NACK at this point due to previous error not being cleared. E.g. Receive buffer overflow or transmit buffer underflow errors. In these cases the Slave hardware forces a NACK and the module becomes idle.
10. Master transmits low address data byte.
11. If the low address matches; SMA is set, ADRIF is set, R/$\overline{W}$ is copied to R/$\overline{W}$ bit, D/A bit is cleared, low address data is copied to I2CxADB0, and ACKDT is copied to SDA. If the address does not match; module becomes idle.
12. If address hold interrupt is enabled, the CSTR bit is set as mentioned in step 6. Slave software can read low address byte from I2CxADB0 register and change ACKDT value before releasing SCL.
13. Master sends ninth SCL pulse for $\overline{ACK}$.
14. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set.
15. Slave software can read address from I2CxADB0 and I2CxADB1 registers and change the value of ACKDT before releasing SCL by clearing CSTR.
16. Master sends first seven SCL pulses of the data byte or a Stop condition (in the case of NACK).
17. If Stop condition; PCIF in I2CxPIR register is set, module becomes idle.
18. If the receive buffer is full from the previous transaction i.e. RXBF = 1, I2CxRXIF = 1, CSTR is set. Slave software must read data out of I2CxRXB to resume communication.
19. Master sends eighth SCL pulse of the data byte. D bit is set, WRIF is set. I2CxRXB is loaded with new data, RXBF bit is set.
20. If Data write interrupt and hold is enabled (WRIE = 1), CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB and set/clear ACKDT before releasing SCL by clearing CSTR.
21. If I2CxCNT = 0, the ACKCNT value is output to the SDA; else, the ACKDT value is used and the value of I2CxCNT is decremented.
22. Master sends SCL pulse for $\overline{ACK}$.
23. If I2CxCNT = 0, CNTIF is set.
24. If the response was a NACK; NACKIF is set, module becomes idle.
25. If ACKTIE = 1, CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB clearing RXBF; before releasing SCL by clearing CSTR.
26. Go to step 16.

FIGURE 33-11: I²C SLAVE, 10-BIT ADDRESS, RECEPTION WITH STOP (ADB = 1)



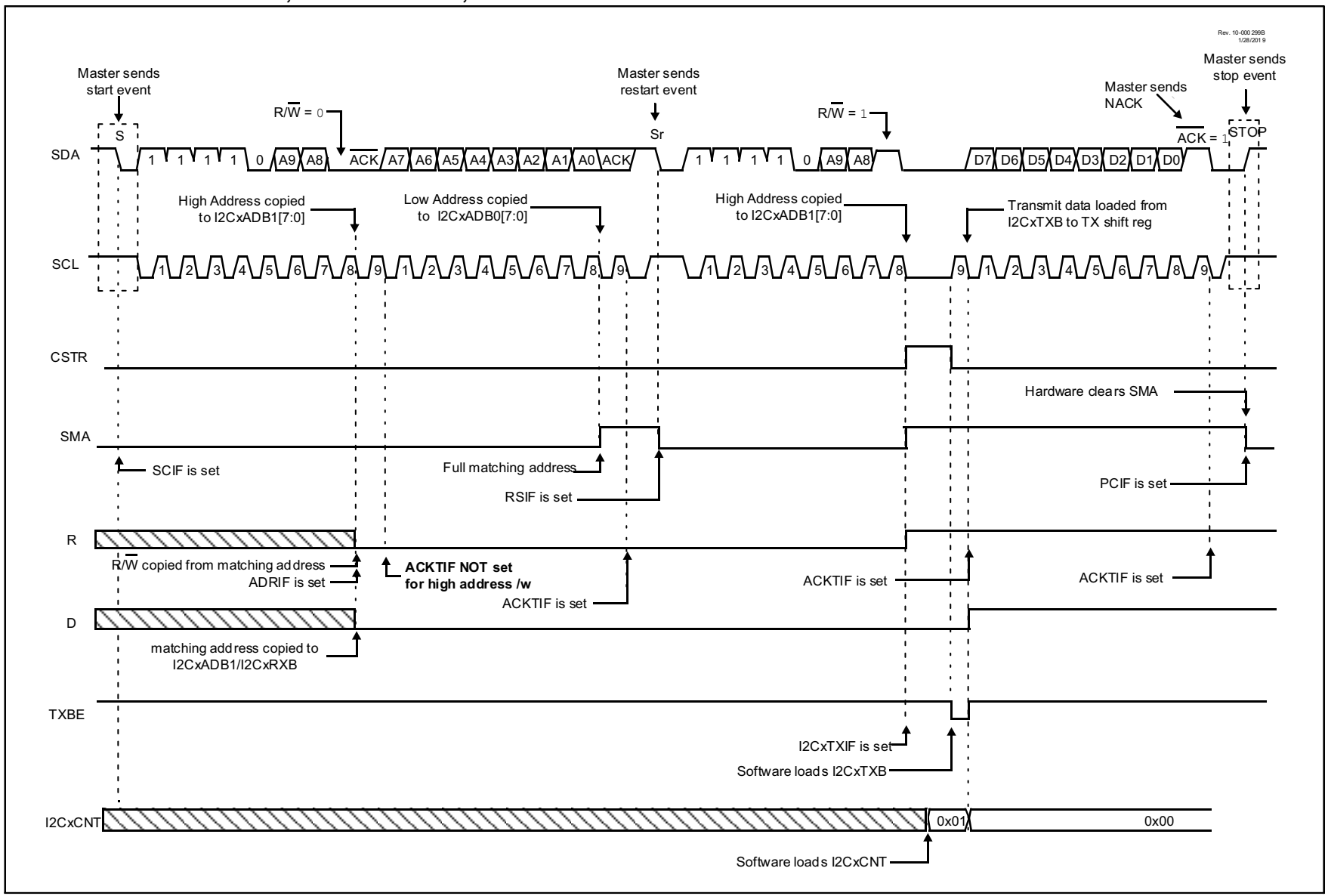
© 2017-2020 Microchip Technology Inc.

33.4.3.5 Slave Transmission (10-bit Addressing Mode)

This section describes the sequence of events for the I^2C module configured as an I^2C slave in 10-bit Addressing mode and is transmitting data. Figure 33-12 is used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
2. Master transmits high address byte with R/$\overline{W}$ = 0.
3. The received high address is compared with the values in I2CxADR1 and I2CxADR3 registers.
4. If high address matches; R/$\overline{W}$ is copied to R bit, D bit is cleared, high address data is copied to I2CxADB1. If the address does not match; module becomes idle.
5. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set.
6. Slave software can read high address from I2CxADB1 and set/clear ACKDT before releasing SCL.
7. ACKDT value is copied out to SDA for $\overline{ACK}$ pulse. SCL line is released by clearing CSTR.
8. Master sends ninth SCL pulse for $\overline{ACK}$.
9. Slave can force a NACK at this point due to previous error not being cleared. E.g. Receive buffer overflow or transmit buffer underflow errors. In these cases the Slave hardware forces a NACK and the module becomes idle.
10. Master transmits low address data byte.
11. If the low address matches; SMA is set, ADRIF is set, low address data is copied to I2CxADB0, and ACKDT is copied to SDA. If the address does not match; module becomes idle.
12. If address hold interrupt is enabled, the CSTR bit is set as mentioned in step 6. Slave software can read low address byte from I2CxADB0 register and change ACKDT value before releasing SCL.
13. Master sends 9th SCL pulse for $\overline{ACK}$.
14. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set.
15. Slave software can read address from I2CxADB0 and I2CxADB1 registers and change the value of ACKDT before releasing SCL by clearing CSTR.
16. Master asserts Restart condition (cannot be Start) on the bus. Restart Condition Interrupt Flag (RSCIF) is set. If the Restart Condition Interrupt is enabled, generic interrupt I2CxIF is set.
17. Master transmits high address byte with R/$\overline{W}$ = 1.
18. If SMA = 1, and if high address matches; R/$\overline{W}$ is copied to R bit, D bit is cleared, high address data is copied to I2CxADB1, and ACKDT is output to SDA. If the address does not match or SMA = 0; module become idle.
19. If ADRIE = 1, CSTR is set. I2CxIF is set. Slave software can read address from I2CxADB0/1 and set/clear ACKDT. The ACKDT value is copied out to SDA. SCL is released by clearing CSTR bit.
20. If TXBE = 1 and I2CxCNT!= 0, I2CxTXIF and CSTR is set. Slave software must load data into I2CxTXB to release SCL.
21. Master sends SCL pulse for $\overline{ACK}$. If I2CxCNT = 0, CNTIF is set.
22. If NACK; NACKIF is set, slave goes idle.
23. If ACKTIE = 1, CSTR is set, I2CxIF is set.
24. Master sends eight SCL pulses to clock out data.
25. Go to step 20.

FIGURE 33-12: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION



© 2017-2020 Microchip Technology Inc.

33.5 I²C Master Mode

Master mode is enabled by setting and clearing the appropriate MODE[2:0] bits in I2CxCON0 and then by setting the EN bit. Master mode of operation is supported by interrupt generation on buffer full (RXBF), buffer empty (TXBE), and the detection of the Start, Restart, and Stop conditions. The Restart (RS) and Start (S) bits are cleared from a Reset or when the I²C module is disabled. Control of the I²C bus is asserted when the BFRE bit of I2CSTAT0 is set.

33.5.1 I²C MASTER MODE OPERATION

The master device generates all of the serial clock pulses and the Start, Restart, and Stop conditions. A transfer is ended with a Stop condition or with a Restart condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released, and MMA bit will stay set signifying that the Master module is still active.

The steps to initiate a transaction depends on the setting of the address buffer disable bit (ABD) of the I2CxCON2 register.

- ABD = 0 (Address buffers are enabled)

In this case, the master module will use the address stored in the address buffer registers (I2CxADB0/1) to initiate communication with a slave device. User software needs to set the Start bit (S) in the I2CxCON0 register to start communication. This is valid for both 7-bit and 10-bit Addressing modes.

- ABD = 1 (Address buffers are disabled)

In this case, the slave address is transmitted through the transmit buffer and the contents of the address buffers are ignored. User software needs to write the slave address to the transmit buffer (I2CxTXB) to initiate communication. Writing to the Start bit is ignored in this mode. This is valid for both 7-bit and 10-bit Addressing modes.

33.5.1.1 Master Transmitter

In Master Transmitter mode, the first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In the case of master transmitter, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

33.5.1.2 Master Receiver

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of the transmission.

33.5.2 MASTER CLOCK SOURCE AND ARBITRATION

The I²C module clock source is selected by the I2CxCLK register. The I²C Clock provides the SCL output clock for Master mode and is used by the Bus Free timer. The I²C clock can be sourced from several peripherals.

33.5.3 BUS FREE TIME

In Master modes, the BFRE bit of the I2CxSTAT0 register gives an indication of the bus idle status. The master hardware cannot assert a Start condition until this bit is set by the hardware. This prevents the master from colliding with other masters that may already be talking on the bus. The BFRET[1:0] bits of I2CxCON1 allow selection of 8 to 64 pulses of the I²C clock input before asserting the BFRE bit. The BFRET bits are used to ensure that the I²C module always follows the minimum Stop Hold Time. The I²C timing requirements are listed in the electrical specifications chapter.

Note:	I²C clock is not required to have a 50% duty cycle.

33.5.4 MASTER CLOCK TIMING

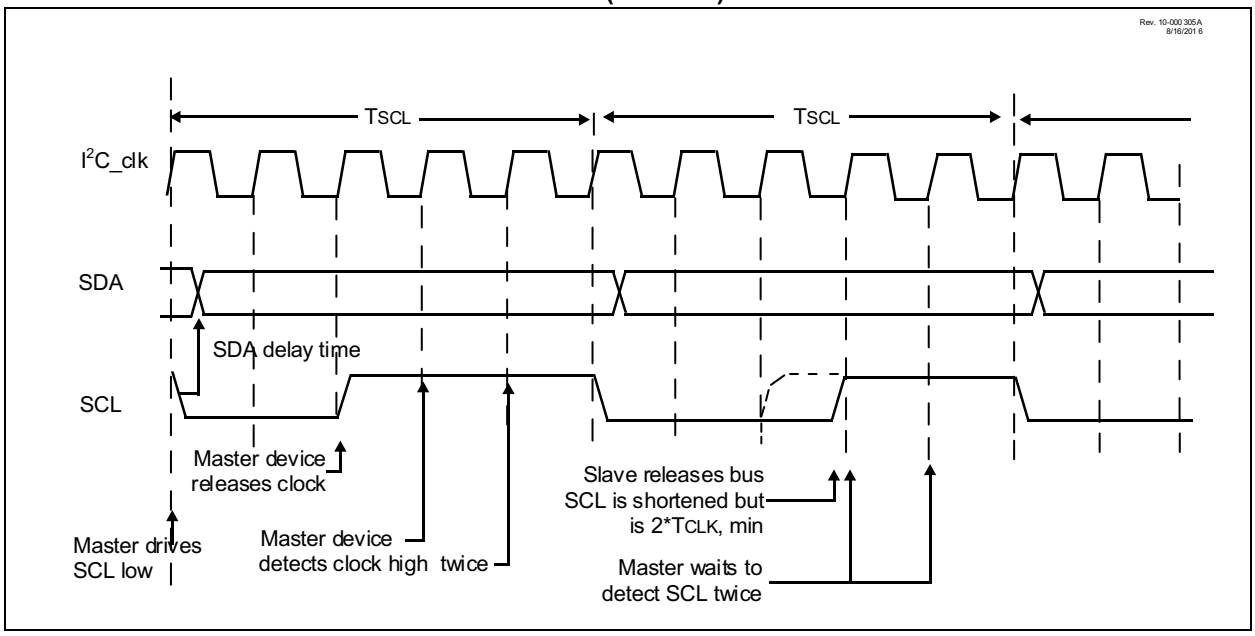
The clock generation in the I²C module can be configured using the Fast Mode Enable (FME) bit of the I2CxCON2 register. This bit controls the number of times the SCL pin is sampled before the master hardware drives it.

33.5.4.1 Clock Timing with FME = 0

One TSCL, consists of five clocks of the I²C clock input. The first clock is used to drive SCL low, the third releases SCL high. The fourth and fifth clocks are used to detect if the SCL pin is, in fact, high or being stretched by a slave.

If a slave is clock stretching, the hardware waits; checking SCL on each successive I²C clock, proceeding only after detecting SCL high. Figure 33-13 shows the clock synthesis timing when FME = 0.

FIGURE 33-13: CLOCK SYNTHESIS TIMING (FME = 0)

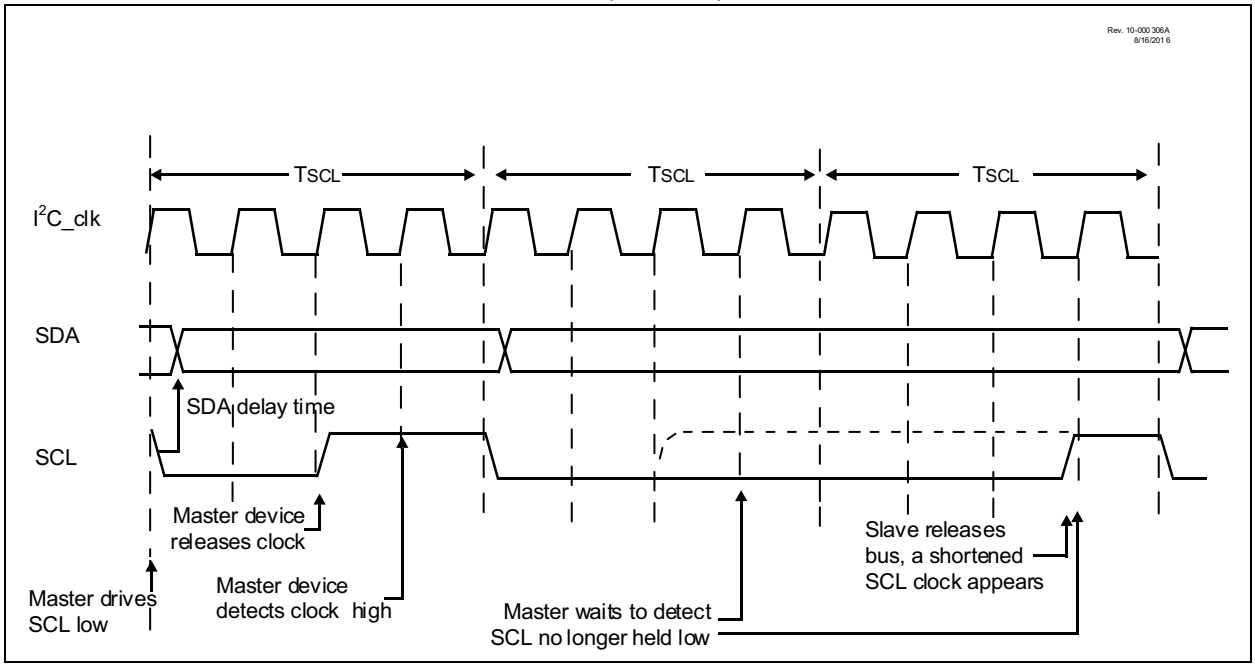


33.5.4.2 Clock Timing with FME = 1

One TSCL, consists of four clocks of the I^{2}C clock input. The first clock is used to drive SCL low, the third releases SCL high, and the fourth is used to detect if the clock is, in fact, high or being stretched by a slave.

If a slave is clock stretching, the hardware waits; checking SCL on each successive I^{2}C clock, proceeding only after detecting SCL high. Figure 33-14 shows the clock synthesis timing when FME = 1.

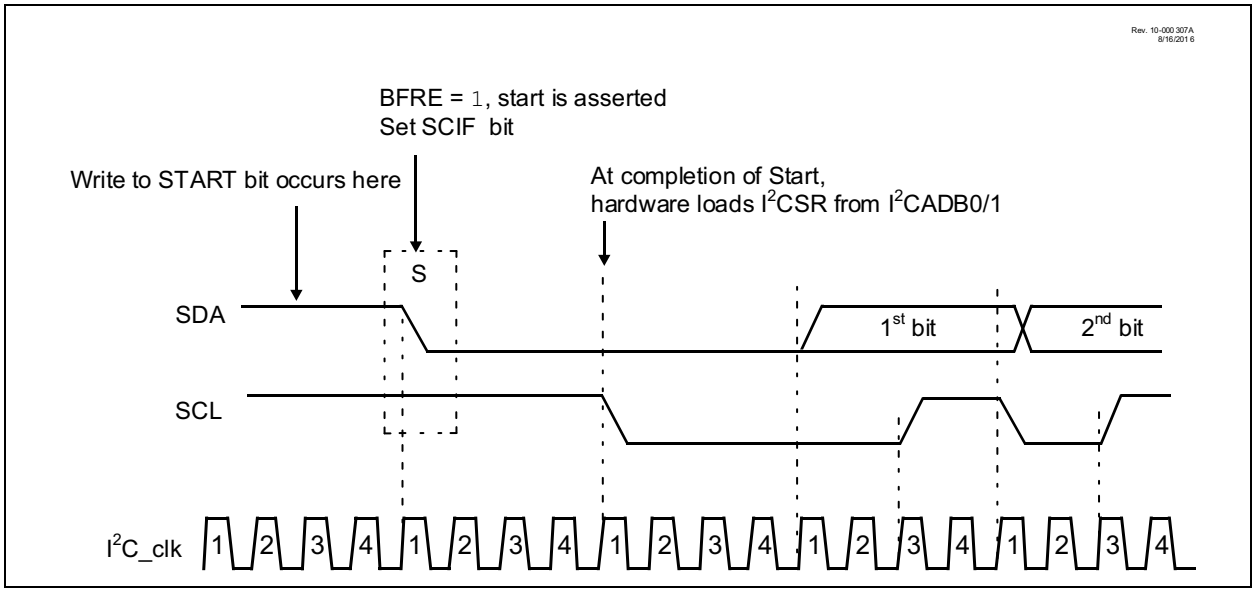
FIGURE 33-14: CLOCK SYNTHESIS TIMING (FME = 1)



33.5.5 I²C MASTER MODE START CONDITION TIMING

The user can initiate a Start condition by either writing to the Start bit (S) of the I2CxCON0 register or by writing to the I2CxTXB register based on the ABD bit setting. Master hardware waits for BFRE = 1, before asserting the Start condition. The action of the SDA being driven low while SCL is high is the Start condition, causing the SCIF bit to be set. One TSCL later the SCL is asserted low, ending the start sequence. Figure 33-15 shows the Start condition timing.

FIGURE 33-15: START CONDITION TIMING

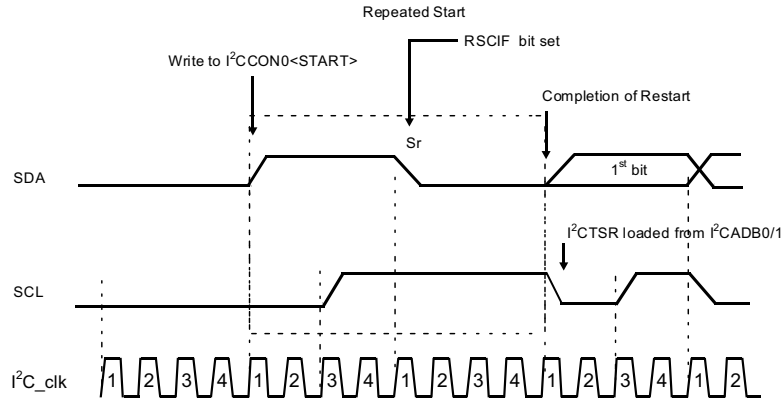


33.5.6 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the Start bit of the I2CxCON0 register is set and the master module is waiting from a Restart clock stretch event (RSEN = 1 and I2CxCNT = 0).

When the Start bit is set, the SDA pin is released high for TSCL/2. Then the SCL pin is released floated high) for TSCL/2. If the SDA pin is detected low, bus collision flag (BCLIF) is set and the master goes idle. If SDA is detected high, the SDA pin will be pulled low (Start condition) for TSCL. Last, SCL is asserted low and I2CxADB0/1 is loaded into the Shift register. As soon as a Restart condition is detected on the SDA and SCL pins, the RSCIF bit is set. Figure 33-16 shows the timings for repeated Start Condition.

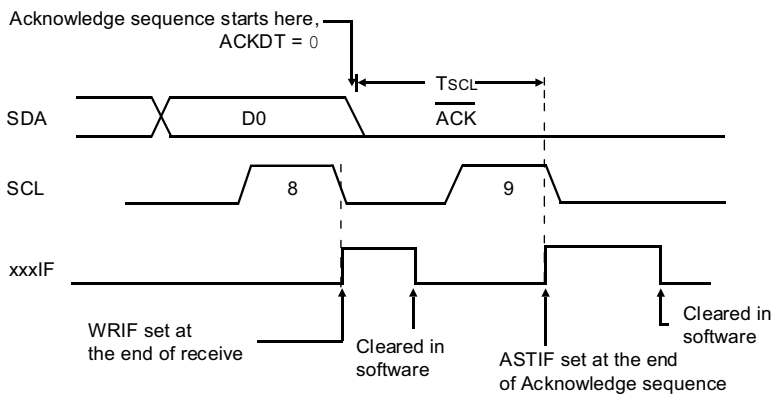
FIGURE 33-16: REPEATED START CONDITION TIMING



33.5.7 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled automatically following an address/data byte transmission. The SCL pin is pulled low and the contents of the Acknowledge Data bits (ACKDT/ACKCNT) are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The master then waits one clock period (TSCL) and the SCL pin is released high. When the SCL pin is sampled high (clock arbitration), the master counts another TSCL. The SCL pin is then pulled low. Figure 33-17 shows the timings for Acknowledge sequence.

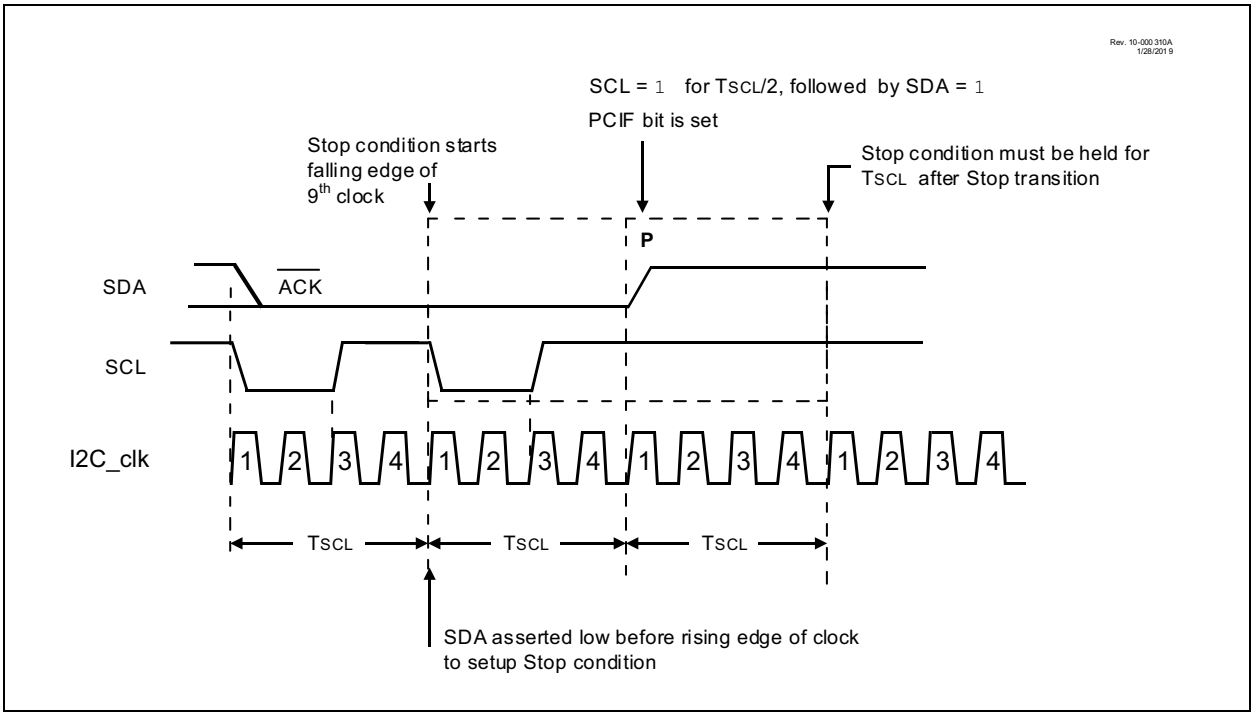
FIGURE 33-17: ACKNOWLEDGE SEQUENCE TIMING



33.5.8 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of receive/transmit when I2CxCNT = 0. After the last byte of a receive/transmit sequence, the SCL line is held low. The master asserts the SDA line low. The SCL pin is then released high TSCL/2 later and is detected high. The SDA pin is then released. When the SDA pin transitions high while SCL is high, the PCIF bit of the I2CxIF register is set. Figure 33-18 shows the timings for a Stop condition.

FIGURE 33-18: STOP CONDITION DURING RECEIVE OR TRANSMIT



33.5.9 MASTER TRANSMISSION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the I^2C module configured as an I^2C master in 7-bit Addressing mode and is transmitting data. Figure 33-19 is used as a visual reference for this description.

1. If ABD = 0; i.e., Address buffers are enabled

Master software loads number of bytes to be transmitted in one sequence in I2CxCNT, slave address in I2CxADB1 with R/$\overline{W}$ = 0 and the first byte of data in I2CxTXB. Master software has to set the Start (S) bit to initiate communication.

If ABD = 1; i.e., Address buffers are disabled

Master software loads the number of bytes to be transmitted in one sequence in I2CxCNT and the slave address with R/$\overline{W}$ = 0 into the I2CxTXB register. Writing to the I2CxTXB will assert the start condition on the bus and sets the S bit. Software writes to the S bit are ignored in this case.

2. Master hardware waits for BFRE bit to be set; then shifts out start and address.
3. If the transmit buffer is empty (i.e., TXBE = 1) and I2CxCNT!= 0, the I2CxTXIF and MDR bits are set and the clock is stretched on the 8th falling SCL edge. Clock can be started by loading the next data byte in I2CxTXB register.
4. Master sends out the 9th SCL pulse for $\overline{ACK}$.
5. If the Master hardware receives $\overline{ACK}$ from Slave device, it loads the next byte from the transmit buffer (I2CxTXB) into the Shift register and the value of I2CxCNT register is decremented.
6. If a NACK was received, Master hardware asserts Stop or Restart
7. If ABD = 0; i.e., Address buffers are enabled

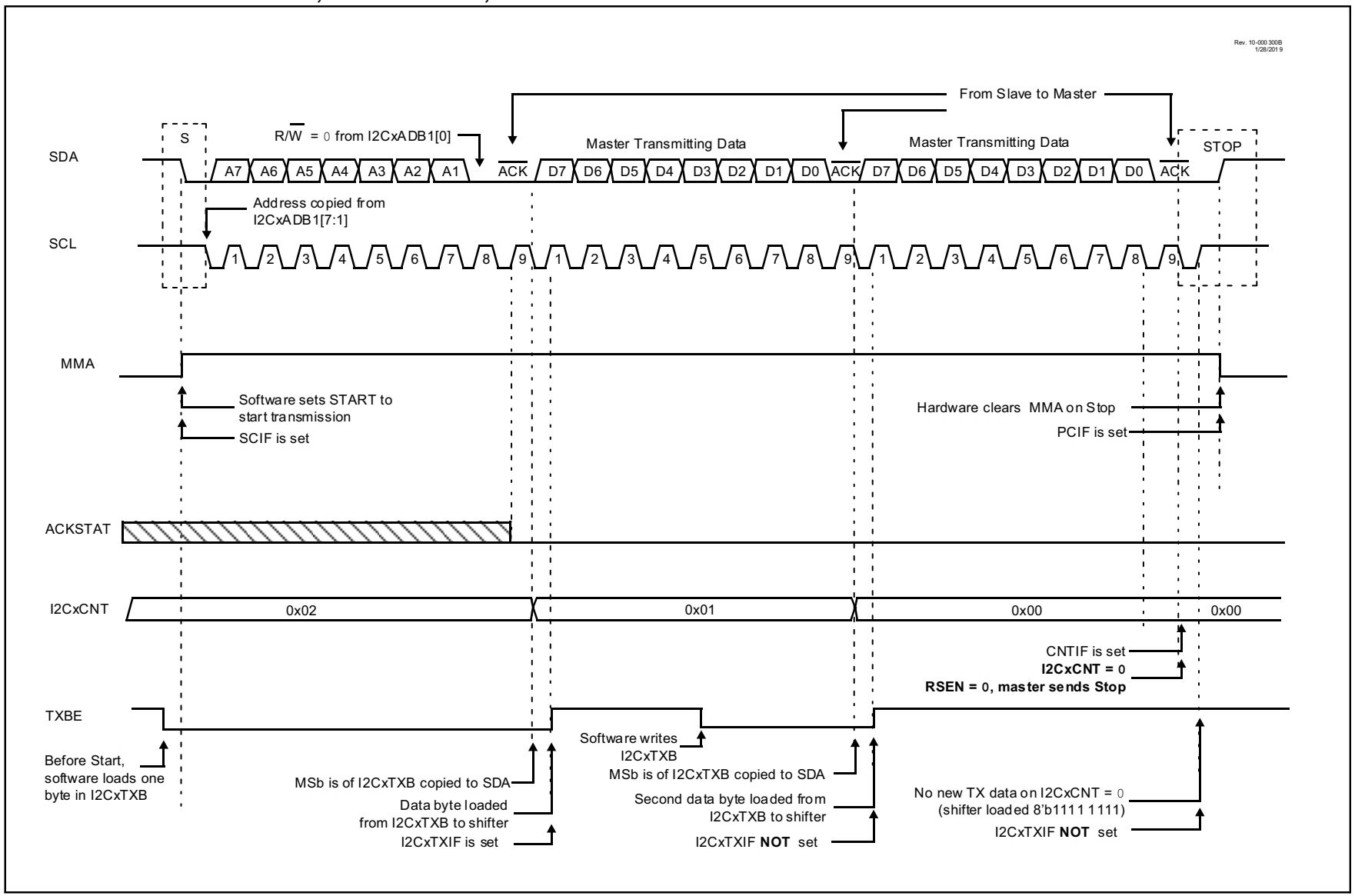
If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to set the Start bit again to issue a restart condition.

If ABD = 1; i.e., Address buffers are disabled

If I2CxCNT = 0, Master hardware sends Stop or sets MDR if RSEN = 1 and waits for the software to write the new address to the I2CxTXB register. Software writes to the S bit are ignored in this case.

8. Master hardware outputs data on SDA.
9. If TXBE = 1 and I2CxCNT! = 0, I2CxTXIF and MDR bits are set and the clock is stretched on 8th falling SCL edge. The user can release the clock by writing the next data byte to I2CxTXB register.
10. Master hardware clocks in $\overline{ACK}$ from Slave, and loads the next data byte from I2CTXB to the Shift register. The value of I2CxCNT is decremented.
11. Go to step 7.

FIGURE 33-19: I²C MASTER, 7-BIT ADDRESS, TRANSMISSION



© 2017-2020 Microchip Technology Inc.

33.5.10 MASTER RECEPTION IN 7-BIT ADDRESSING MODE

This section describes the sequence of events for the I^2C module configured as an I^2C master in 7-bit Addressing mode and is receiving data. Figure 33-20 is used as a visual reference for this description.

1. Master software loads slave address in I2CxADB1 with R/$\overline{W}$ bit = 1 and number of bytes to be received in one sequence in I2CxCNT register.
2. Master hardware waits for BFRE bit to be set; then shifts out start and address with R/$\overline{W}$ = 1.
3. Master sends out the 9th SCL pulse for $\overline{ACK}$, master hardware clocks in $\overline{ACK}$ from Slave
4. If ABD = 0; i.e., Address buffers are enabled

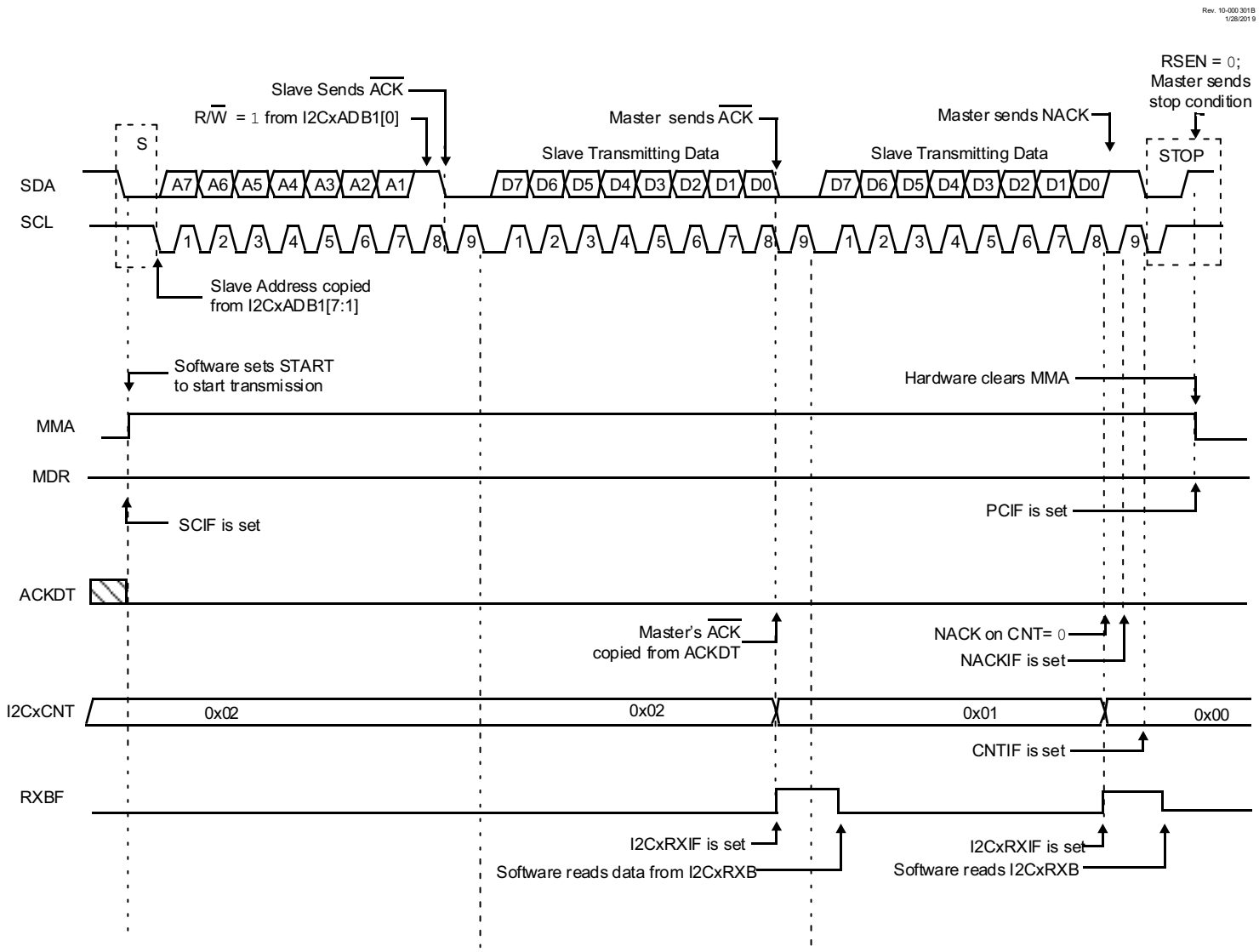
If NACK, master hardware sends Stop or sets MDR (if RSEN = 1) and waits for user software to write to S bit for restart.

If ABD = 1; i.e., Address buffers are disabled

If NACK, master hardware sends Stop or sets MDR (if RSEN = 1) and waits for user software to load the new address into I2CxTXB. Software writes to the S bit are ignored in this case.

5. If $\overline{ACK}$, master hardware receives 7-bits of data into the Shift register.
6. If the receive buffer is full (i.e., RXBF = 1), clock is stretched on 7th falling SCL edge.
7. Master software must read previous data out of I2CxRXB to clear RXBF.
8. Master hardware receives 8th bit of data into the shift register and loads it into I2CxRXB, sets I2CxRXIF and RXBF bits. I2CxCNT is decremented.
9. If I2CxCNT! = 0, master hardware clocks out ACKDT as $\overline{ACK}$ value to slave. If I2CxCNT = 0, master hardware clocks out ACKCNT as $\overline{ACK}$ value to slave. It is up to the user to set the values of ACKDT and ACKCNT correctly. If the user does not set ACKCNT to '1', the master hardware will never send a NACK when I2CxCNT becomes zero. Since a NACK was not seen on the bus, the master hardware will also not assert a Stop condition.
10. Go to step 4.

FIGURE 33-20: I²C MASTER, 7-BIT ADDRESS, RECEPTION



33.5.11 MASTER TRANSMISSION IN 10-BIT ADDRESSING MODE

This section describes the sequence of events for the I^2C module configured as an I^2C master in 10-bit Addressing mode and is transmitting data. Figure 33-21 is used as a visual reference for this description

1. If ABD = 0; i.e., Address buffers are enabled

Master software loads number of bytes to be transmitted in one sequence in I2CxCNT, high address byte of slave address in I2CxADB1 with R/$\overline{W}$ = 0, low address byte in I2CxADB0 and the first byte of data in I2CxTXB. Master software has to set the Start (S) bit to initiate communication.

If ABD = 1; i.e., Address buffers are disabled

Master software loads the number of bytes to be transmitted in one sequence in I2CxCNT and the high address byte of the slave address with R/$\overline{W}$ = 0 into the I2CxTXB register. Writing to the I2CxTXB will assert the start condition on the bus and sets the S bit. Software writes to the S bit are ignored in this case.

2. Master hardware waits for BFRE bit to be set; then shifts out the start and high address and waits for acknowledge.
3. If NACK, master hardware sends Stop.
4. If ABD = 0; i.e., Address buffer are enabled

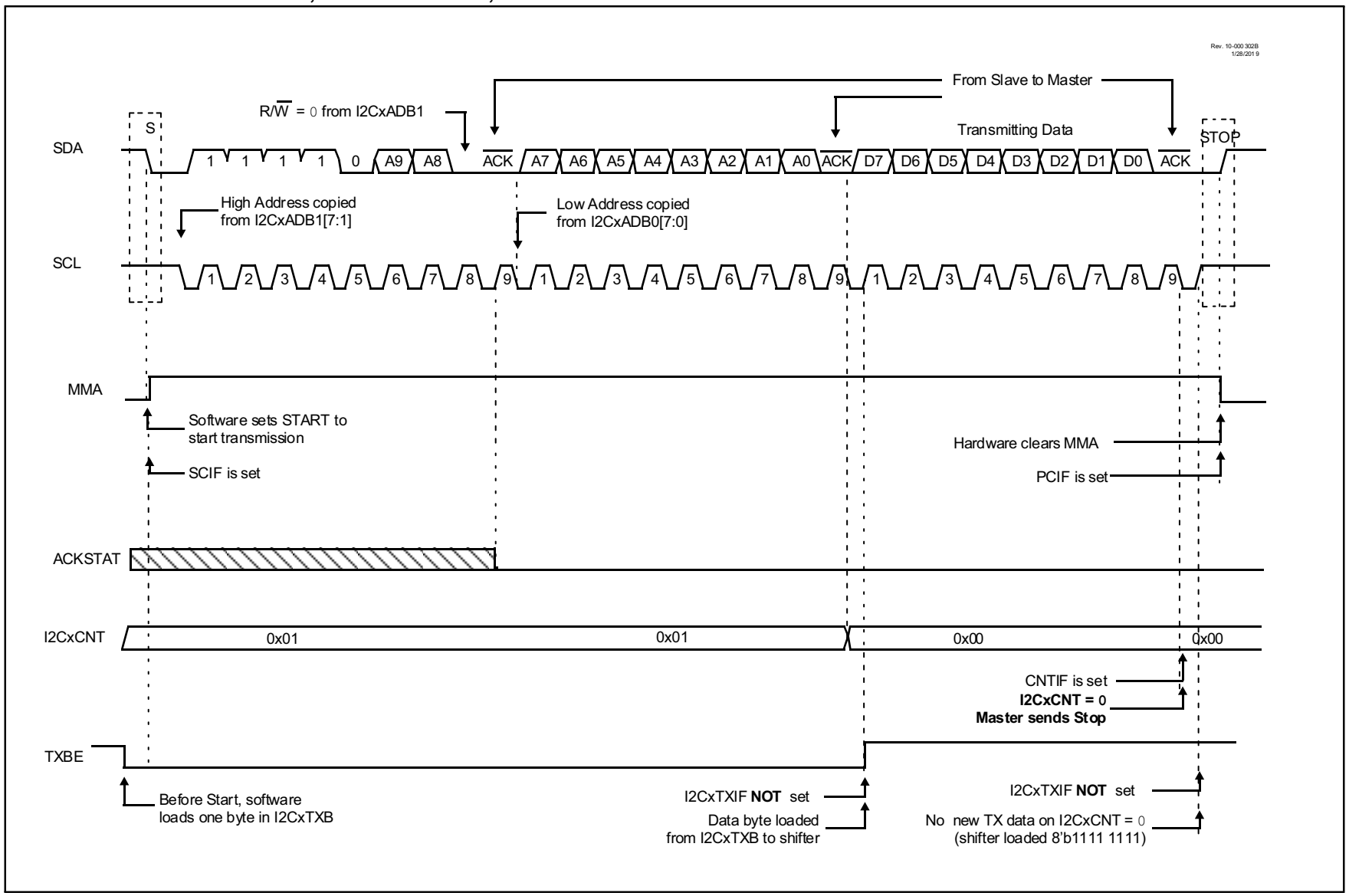
If $\overline{ACK}$, master hardware sends the low address byte from I2CxADB0.

If ABD = 1; i.e., Address buffer are disabled

If $\overline{ACK}$, master hardware sets TXIF and MDR bits and the software has to write the low address byte into I2CxTXB. Writing to I2CxTXB sends the low address on the bus.

5. If TXBE = 1 and I2CxCNT! = 0, I2CxTXIF and MDR bits are set. Clock is stretched on 8th falling SCL edge until master software writes next data byte to I2CxTXB.
6. Master hardware sends ninth SCL pulse for $\overline{ACK}$ from slave and loads the shift register from I2CxTXB. I2CxCNT is decremented.
7. If slave sends a NACK, master hardware sends Stop and ends transmission.
8. If slave sends an $\overline{ACK}$, master hardware outputs data in the shift register on SDA. I2CxCNT value is checked on the 8th falling SCL edge. If I2CxCNT = 0; master hardware sends 9th SCL pulse for $\overline{ACK}$ and CNTIF is set.
9. If I2CxCNT! = 0; go to step 5.

FIGURE 33-21: I²C MASTER, 10-BIT ADDRESS, TRANSMISSION



© 2017-2020 Microchip Technology Inc.

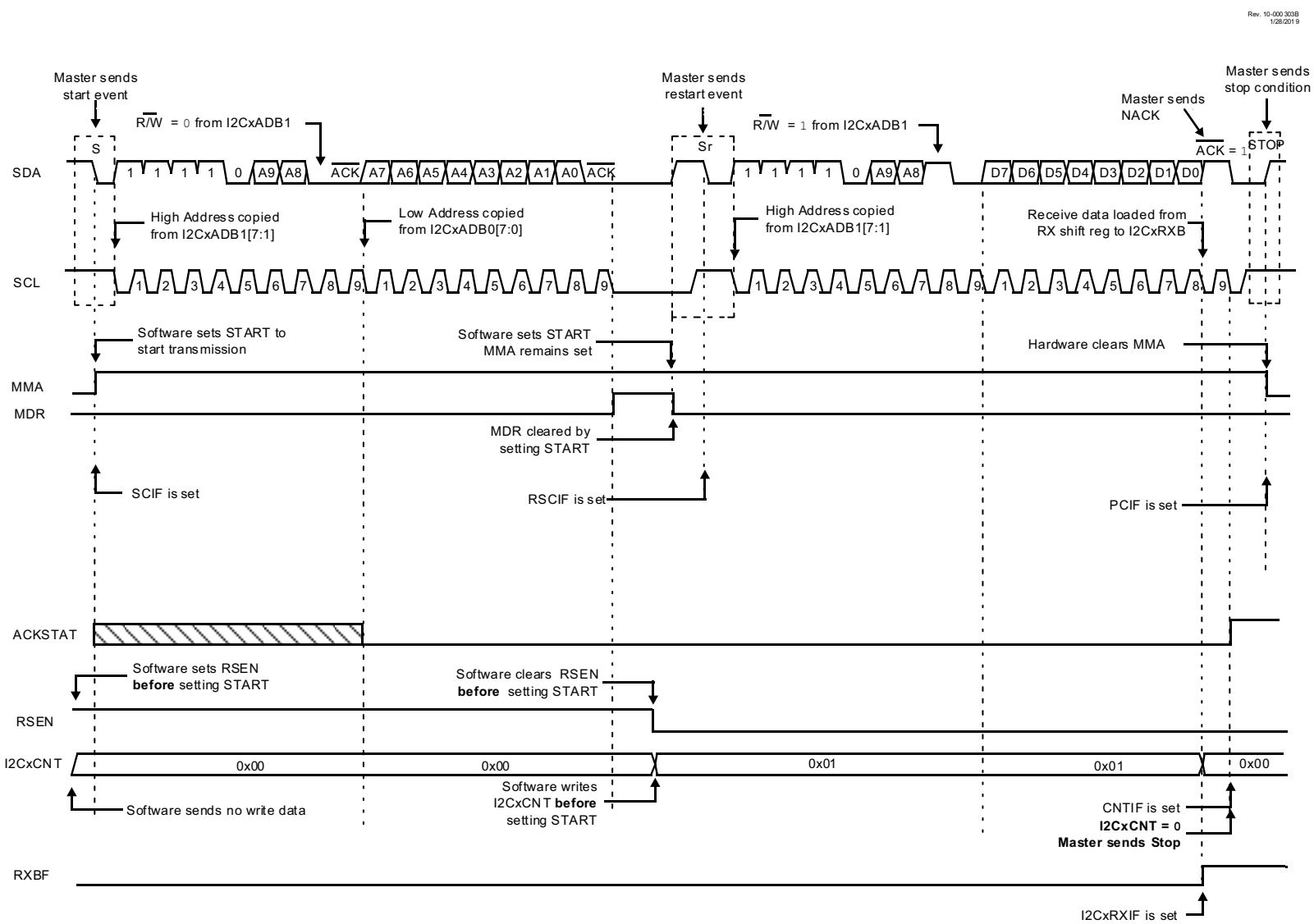
33.5.12 MASTER RECEPTION IN 10-BIT ADDRESSING MODE

This section describes the sequence of events for the I^2C module configured as an I^2C master in 10-bit Addressing mode and is receiving data. Figure 33-22 is used as a visual reference for this description.

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
   a) When ABD is clear (ABD = 0), the address buffers, I2CxADB0 and I2CxADB1, are enabled. In this case, the address high byte and R/$\overline{W}$ bit are loaded into I2CxADB1, with R/$\overline{W}$ clear (R/$\overline{W}$ = 0). The address low byte is loaded into I2CxADB0, and the Restart Enable (RSEN) bit of I2CxCON0 is set by software. After these registers are loaded, software must set the Start bit to begin communication. Once the S bit is set, master hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
   b) When ABD is set (ABD = 1), the address buffers are disabled. In this case, the number of expected received bytes are loaded into I2CxCNT, the address high byte and R/$\overline{W}$ bit are loaded into I2CxTXB, with R/$\overline{W}$ clear (R/$\overline{W}$ = 0). A write to I2CxTXB will cause master hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Master hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Master Mode Active (MMA) bit of I2CxSTAT0 and the Start Condition Interrupt Flag (SCIF) of I2CxPIR. If the Start Condition Interrupt Enable (SCIE) bit of I2CxPIE is also set, the generic I2CxIF is also set.
3. Master hardware transmits the address high byte and R/$\overline{W}$ bit.
4. Master hardware samples SCL to determine if the slave is stretching the clock, and continues to sample SCL until the line is sampled high.
5. Master hardware transmits the 9th clock pulse, and receives the $\overline{ACK}$/NACK response from the slave. If a NACK was received, the NACK Detect Interrupt Flag (NACKIF) is set and the master immediately issues a Stop condition. If an $\overline{ACK}$ was received, module hardware transmits the address low byte.
6. Master hardware samples SCL to determine if the slave is stretching the clock, and continues to sample SCL until the line is sampled high.
7. Master hardware transmits the 9th clock pulse, and receives the $\overline{ACK}$/NACK response from the slave. If an $\overline{ACK}$ was received, hardware sets MDR and waits for hardware or software to set the Start bit. If a NACK is received, hardware sets the NACK Detect Interrupt Flag (NACKIF), and:
   a) ABD = 0: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
   b) ABD = 1: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored. If the NACK Detect Interrupt Enable (NACKIE) is also set, hardware sets the generic I2CxEIF bit.
8. Software loads I2CxCNT with the expected number of received bytes.
9. If the ABD is clear (ABD = 0), software sets the Start bit. If the ABD is set (ABD = 1), software writes the address high byte with R/$\overline{W}$ bit into I2CxTXB, with R/$\overline{W}$ set (R/$\overline{W}$ = 1).
10. Master hardware transmits the Restart condition, which sets the Restart Condition Interrupt Flag (RSCIF) bit of I2CxPIR. If the Restart Condition Interrupt Enable (RSCIE) bit of I2CxPIE is also set, the generic I2CxIF is set by hardware.
11. Master hardware transmits the high address byte and R/$\overline{W}$ bit.
12. Master hardware samples SCL to determine if the slave is stretching the clock, and continues to sample SCL until the line is sampled high.
13. Master hardware transmits the 9th clock pulse, and receives the $\overline{ACK}$/NACK response from the slave. If an $\overline{ACK}$ is received, master hardware receives the first seven bits of the data byte into the Receive Shift Register (RSR). If a NACK is received, and:
    a) ABD = 0: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
    b) ABD = 1: Master generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored.
14. If previous data is currently in I2CxRXB (RXBF = 1) when the first seven bits are received by the Receive Shift Register, hardware sets MDR, and the clock is stretched after the 7th falling edge of SCL. This allows software to read I2CxRXB, which clears the RXBF bit, and prevents a receive buffer overflow. Once the RXBF bit is clear, hardware releases SCL.

15. Master hardware clocks in the 8th bit of the data byte into the Receive Shift register, then transfers the complete byte into I2CxRXB, which sets the I2CxRXIF and RXBF bits. If I2CxRXIE is also set, hardware sets the generic I2CxIF bit. I2CxCNT is decremented by one.
16. Hardware checks I2CxCNT for a zero value. If I2CxCNT is non-zero (I2CxCNT != 0), hardware transmits the value of the Acknowledge Data (ACKDT) bit as the acknowledgment response to the slave. It is up to user software to properly configure ACKDT. In most cases, ACKDT should be clear (ACKDT = 0), which indicates an $\overline{\text{ACK}}$ response. If I2CxCNT is zero (I2CxCNT = 0), hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the acknowledgment response to the slave. CNTIF is set, and master hardware either issues a Stop condition or a Restart condition. It is up to user software to properly configure ACKCNT. In most cases, ACKCNT should be set (ACKCNT = 1), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled bus condition.
17. Master hardware receives the first seven bits of the next data byte into the receive Shift register.
18. Repeat Steps 14 – 17 until all expected bytes have been received.

FIGURE 33-22: I²C MASTER, 10-BIT ADDRESS, RECEPTION (USING RSEN BIT)



© 2017-2020 Microchip Technology Inc.

33.6 I²C Multi-Master Mode

In Multi-Master mode, the bus-free (BFRE) bit allows the master to determine when the bus is free. Control of the I²C bus may be taken when the BFRE bit of the I2CxSTAT0 register is set. Interrupt generation on the detection of a slave address match, ADRIE; causes a clock stretch and allows user software to respond to the Master being addressed as a slave device. The slave active (SMA) bit is set for a matching received slave address.

Clock arbitration occurs when the master, during any receive, transmit or Restart/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the SCL line is monitored to see if the pin is actually sampled high.

Note:	In this mode, the slave hardware has priority over the master hardware. Master mode communication can only be initiated when the SMA = 0.

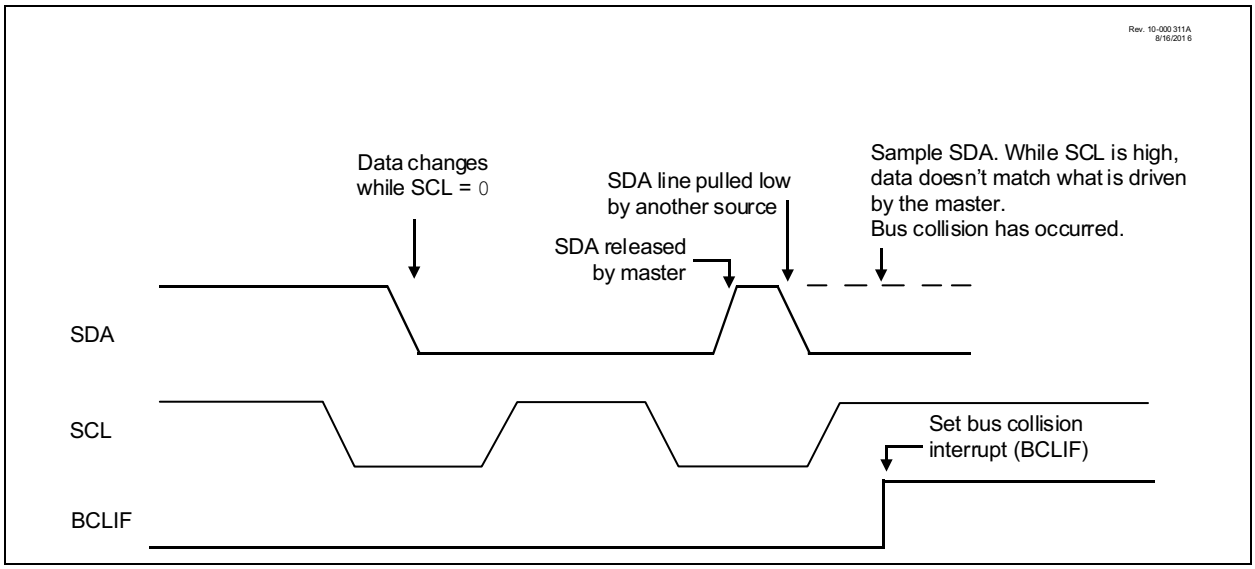
In master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLIF bit. MMA is cleared when BCLIF is set. The states where arbitration can be lost are:

- Address Transfer
- Data Transfer (master write)
- Repeated Start Condition
- Acknowledge Condition

33.6.1 MULTI-MASTER MODE BUS COLLISION

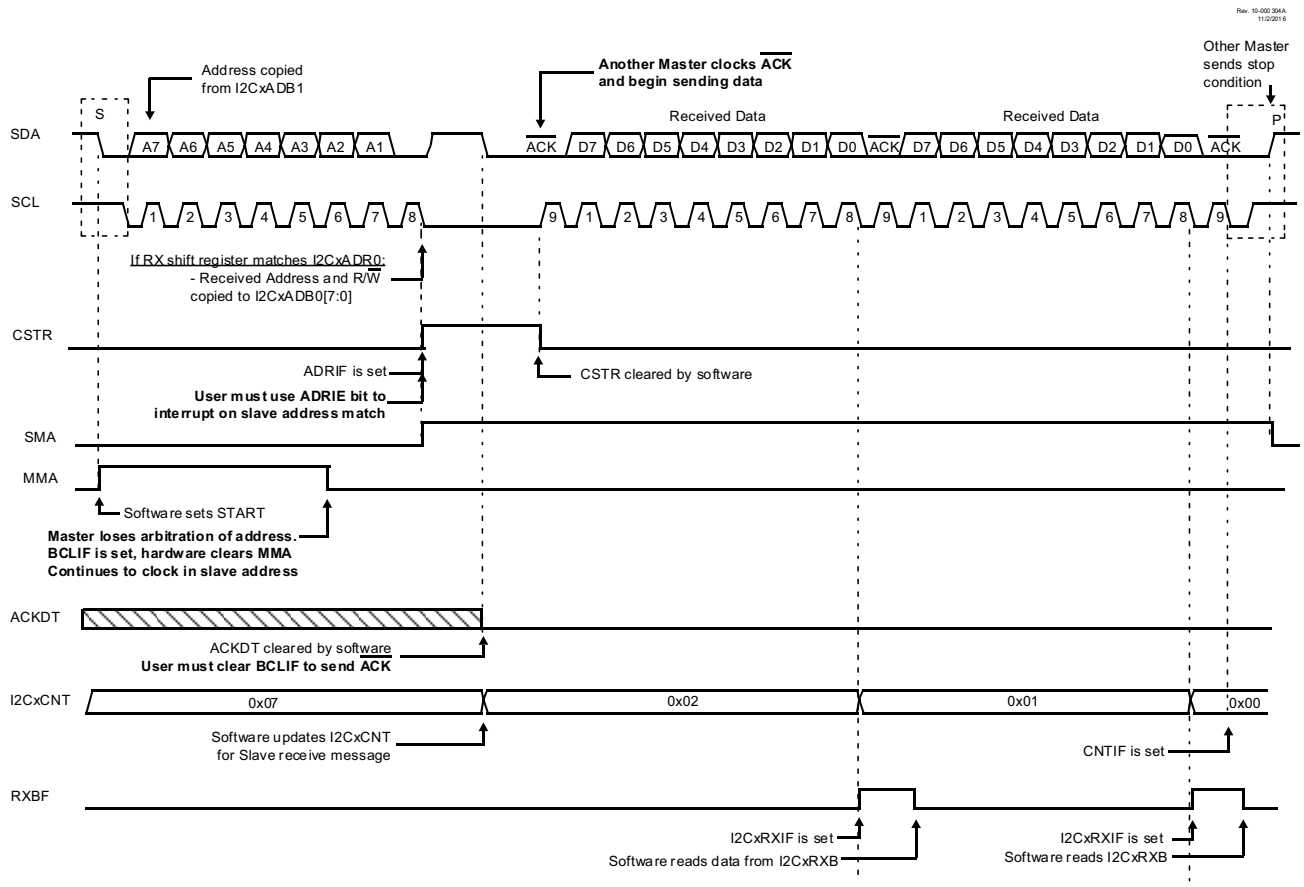
Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data is stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the I²C bus to its Idle state. Refer to Figure 33-23 for a detailed timing diagram.

FIGURE 33-23: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



If transmission was in progress when the bus collision occurred, the SDA and SCL lines are released. If a Repeated Start, Stop or Acknowledge was in progress when the bus collision occurred, the action is aborted; the SDA and SCL lines are released. The BCLIF condition must be cleared by software to allow an $\overline{\text{ACK}}$ to be shifted out on the bus again, until then the module will always respond with a NACK. Refer to Figure 33-24 for a detailed timing diagram of a transaction in Multi-Master mode.

FIGURE 33-24: I²C MULTI-MASTER, 7-BIT ADDRESS, WRITE (ADRIE = 1, WRIE = 0)



© 2017-2020 Microchip Technology Inc.

33.7 Register Definitions: I²C Control

This section defines all the registers associated with the control and status of the I²C bus.

REGISTER 33-1: I2CxCON0: I²C CONTROL REGISTER 0

R/W-0	R/W-0	R/W/HC/HS-0	R/C/HS/HC-0	R-0	R/W-0	R/W-0	R/W-0
EN(1,2)	RSEN	S	CSTR(3)	MDR	MODE [2:0]		
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	HS = Hardware set HC = Hardware clear

bit 7 **EN:** I²C Module Enable bit
1 = Enables the I²C module(1,2)
0 = Disables the I²C module.

bit 6 **RSEN:** Restart Enable bit (Only MODE[2:0] = 1xx)
1 = When (I2CxCNT = 0 or ACKSTAT = 1), on 9th falling SCL sets MDR.
0 = When (I2CxCNT = 0 or ACKSTAT = 1), on 9th falling SCL; master shifts out a Stop condition

bit 5 **S:** Master Start/Restart bit (Only MODE[2:0] = 1xx)
When MMA = 0
1 = Set by user set of Start bit or write to I2CxTXB, waits for BFRE = 1 to begin with a Start
0 = Cleared by hardware after sending Start
When MMA = 1 & MDR = 1
1 = Set by user set of Start bit or write to I2CxTXB, resumes communication with a Restart
0 = Cleared by hardware after sending Restart
Else - Writes to I2CxTXB or Start bit (S) has no effect on Start bit

bit 4 **CSTR:** Slave Clock Stretching bit (3)
1 = Clock is held low (clock stretching)
0 = Enable clocking, SCL control is released

SMA = 1 and RXBF = 1(6)
- Set by hardware on 7th falling SCL edge
- User must read byte I2CxRXB to release SCL
SMA = 1 and TXBE = 1 and I2CxCNT!= 0
- Set by hardware on 8th falling SCL edge
- User must write byte to I2CxTXB to release SCL
when ADRIE is set (4)
- Set by hardware on 8th falling SCL edge of matching received address
- User must clear CSTR to release SCL
SMA = 1 & WRIE = 1
- Set by hardware on 8th falling SCL edge of received data byte
- User must clear CSTR to release SCL
SMA = 1 & ACKTIE = 1
- Set by hardware on 9th falling SCL edge
- User must clear CSTR to release SCL

bit 3 **MDR:** Master Data Request *(Master pause)*
1 = Master state mechine pauses until data is read/written to proceed (SCL is output held low)
0 = Master clocking of data is enabled.

MMA = 1 & RXBF = 1
pause_for_rx - Set by hardware on 7th falling SCL edge
- User must read from I2CxRXB to release SCL
MMA = 1 & TXBE = 1 & I2CxCNT!= 0
pause_for_tx - Set by hardware on 8th falling SCL edge
- User must write to I2CxTXB to release SCL
pause_for_restart - Set by hardware on 9th falling SCL edge
RSEN = 1 & MMA = 1 && I2CxCNT = 0 || ACKSTAT = 1
- User must set Start or write to I2CxTXB to release SCL and shift Restart onto bus

bit 2-0 **MODE[2:0]:** I^2C Mode Select bits
111 = I^2C Muti-Master mode (SMBus 2.0 Host), (5)
Works as both MODE[2:0] = 001 and MODE[2:0] = 100
110 = I^2C Muti-Master mode (SMBus 2.0 Host), (5)
Works as both MODE[2:0] = 000 and MODE[2:0] = 100
101 = I^2C Master mode, 10-bit address
100 = I^2C Master mode, 7-bit address
011 = I^2C Slave mode, one 10-bit address with masking
010 = I^2C Slave mode, two 10-bit address
001 = I^2C Slave mode, two 7-bit address with masking
000 = I^2C Slave mode, four 7-bit address

Note 1: SDA and SCL pins must be configured for open-drain with internal or external pull-up
2: SDA and SCL pins must be selected as both input and output in PPS
3: CSTR can be set by more than one hardware source, all sources must be addressed by user software before the SCL line is released. CSTR is a module status bit, and does not show the true bus state.
4: SMA is set on the same SCL edge as CSTR for a matching received address
5: In this mode, ADRIE should be set, this allows an interrupt to clear the BCLIF condition and allow the $\overline{ACK}$ of matching address.
6: In 10-bit Slave mode, when ADB = 1, CSTR will set when the high address has not been read out of I2CxRXB before the low address is shifted in.

REGISTER 33-2: I2CxCON1: I²C CONTROL REGISTER 1

R/W-0	R/W-0	R-0	R-0	U-0	R/W/HS-0	R/W/HS-0	R/W-0
ACKCNT(2)	ACKDT(1,2)	ACKSTAT	ACKT	—	RXO	TXU	CSD
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	HS = Hardware set HC = Hardware clear

bit 7 **ACKCNT:** Acknowledge End of Count bit(2)
Acknowledge value transmitted after received data, when I2CxCNT = 0
1 = Not Acknowledge (copied to SDA output)
0 = Acknowledge (copied to SDA output)

bit 6 **ACKDT:** Acknowledge Data bit(1,2)
Acknowledge value transmitted after matching address
Acknowledge value transmitted after received data, when I2CxCNT! = 0
1 = Not Acknowledge (copied to SDA output)
0 = Acknowledge (copied to SDA output)

bit 5 **ACKSTAT:** Acknowledge Status bit (Transmission only)
1 = Acknowledge was not received for most recent transmission
0 = Acknowledge was received for most recent transmission

bit 4 **ACKT:** Acknowledge Time Status bit
1 = Indicates the I²C bus is in an Acknowledge sequence, set on 8th falling edge of SCL clock
0 = Not in Acknowledge sequence, cleared on 9th rising edge of SCL

bit 3 **Unimplemented:** Read as 1'b0

bit 2 **RXO:** Receive Overflow Status bit (MODE[2:0] = 0xx & 11x)
This bit can only be set when CSD= 1
1 = Set when SMA = 1, and a master clocks in data when RXBF = 1
0 = No slave overflow condition

bit 1 **TXU:** Transmit Underflow Status bit (MODE[2:0] = 0xx & 11x)
This bit can only be set when CSD = 1
1 = Set when SMA = 1, and a master clocks out data when TXBE = 1
0 = No slave underflow condition

bit 0 **CSD:** Clock Stretching Disable bit (MODE[2:0] = 0xx & 11x)
1 = When SMA = 1, the CSTR bit will never be set
0 = Slave clock stretching proceeds normally

Note 1: Software writes to ACKDT bit must be followed by a minimum SDA data-setup time before clearing CSTR.
2: NACK may still be generated by I²C hardware when bus errors are indicated in the I2CxSTAT1 or I2CxERR registers.

REGISTER 33-3: I2CxCON2: I²C CONTROL REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ACNT	GCEN	FME	ADB	SDAHT[1:0]		BFRET[1:0]	
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	HS = Hardware set	HC = Hardware clear

bit 7 **ACNT:** Auto-Load I²C Count Register Enable bit
1 = The first received or transmitted byte after the address, is automatically loaded into the I2CxCNT register. The I2CxCNT register is loaded at the same time as the value is moved to/from the shifter. ACKDT is used to determine the ACK/NACK value for the address bytes and first data byte of a received message. This prevents a NACK from being sent for the byte that would update the I2CxCNT register.
0 = Auto-load of I2CxCNT disabled

bit 6 **GCEN:** General Call Address Enable bit (MODE[2:0] = 00x & 11x)
1 = General call address, 0x00, causes address match event
0 = General call address disabled

bit 5 **FME:** Fast Mode Enable bit
1 = SCL is sampled high only once before driving SCL low. ($F_{SCL} = F_{I2CXCLK}/4$)
0 = SCL is sampled high twice before driving SCL low. ($F_{SCL} = F_{I2CXCLK}/5$)

bit 4 **ADB:** Address Data Buffer Disable bit
1 = Received address data is loaded into I2CxRXB
Transmitted address data is loaded from the I2CxTXB
0 = Received address data is loaded only into the I2CxADB
Transmitted address data is loaded from the I2CxADB0/1 registers.

bit 3-2 **SDAHT[1:0]:** SDA Hold Time Selection bits
11 = Reserved
10 = Minimum of 30 ns hold time on SDA after the falling edge of SCL
01 = Minimum of 100 ns hold time on SDA after the falling edge of SCL
00 = Minimum of 300 ns hold time on SDA after the falling edge of SCL

bit 1-0 **BFRET[1:0]:** Bus Free Time Selection bits
11 = 64 I²C Clock pulses
10 = 32 I²C Clock pulses
01 = 16 I²C Clock pulses
00 = 8 I²C Clock pulses

REGISTER 33-4: I2CxCLK: I²C CLOCK SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CLK[3:0]			
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	HS = Hardware set HC = Hardware clear

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **CLK[3:0]:** I²C Clock Selection Bits

CLK[3:0]	I²Cx Clock Selection
1010-1111	Reserved
1001	SMT1 overflow
1000	TMR6 post scaled output
0111	TMR4 post scaled output
0110	TMR2 post scaled output
0101	TMR0 overflow
0100	Clock Reference output
0011	MFINTOSC (500 kHz)
0010	HFINTOSC
0001	FOSC
0000	FOSC/4

REGISTER 33-5: I2CxBTO: I²C BUS TIMEOUT SELECTION REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	BTO[2:0]		
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	HS = Hardware set	HC = Hardware clear

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **BTO[2:0]:** I²C Bus Timeout Selection bits

BTO[2:0]	I²Cx Bus Timeout Selection
111	CLC4OUT
110	CLC3OUT
101	CLC2OUT
100	CLC1OUT
011	TMR6 post scaled output
010	TMR4 post scaled output
001	TMR2 post scaled output
000	Reserved

REGISTER 33-6: I2CxSTAT0: I²C STATUS REGISTER 0

R-0	R-0	R-0	R-0	R-0	U-0	U-0	U-0
BFRE(3)	SMA	MMA	R(1, 2)	D	—	—	—
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	HS = Hardware set HC = Hardware clear

bit 7 **BFRE:** Bus Free Status bit(3)
1 = Indicates the I²C bus is idle
Both SCL and SDA have been high for time-out selected by I2CxCON2[BFRET[1:0]] bits.
I2CxCLK must select a valid clock source for this bit to function.
0 = Bus not idle (When no I2CxCLK source is selected, this bit remains clear)

bit 6 **SMA:** Slave Module Active Status bit
1 = Set after the 8th falling SCL edge of a received matching 7-bit slave address
Set after the 8th falling SCL edge of a received matching 10-bit slave **low** address
Set after the 8th falling SCL edge of a received matching 10-bit slave **high** w/ read address, only after a previous matching high and low w/ write.
0 = Cleared by any Restart/Stop detected on the bus
Cleared by BTOIF and BCLIF conditions

bit 5 **MMA:** Master Module Active Status bit
1 = Master Mode state machine is active
Set when master state machine asserts a Start on bus
0 = Master state machine is idle
Cleared when BCLIF is set
Cleared when Stop is shifted out by master.
Cleared for BTOIF condition, after the master successfully shifts out a Stop condition.

bit 4 **R:** Read Information bit (1, 2)
1 = Indicates the last matching received (high) address was a Read request
0 = Indicates the last matching received (high) address was a Write

bit 3 **D:** Data bit
1 = Indicates the last byte received or transmitted was data
0 = Indicates the last byte received or transmitted was an address

bit 2-0 **Unimplemented:** Read as 1'b0

Note 1: This bit holds the R bit information following the last received address match. Addresses transmitted by the Master or appearing on the bus without a match do not affect this bit.
2: Clock requests and input from I2CxCLK register are disabled in Slave modes.
3: Software must use the EN bit to force Master or Slave hardware to idle.

REGISTER 33-7: I2CxSTAT1: I²C STATUS REGISTER 1

R/W/HS-0	U-0	R-1	U-0	R/W/HS-0	R/S-0/0	U-0	R-0
TXWE(2)	—	TXBE(1, 3)	—	RXRE(2)	CLRBF	—	RXBF(1,3)
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	HS = Hardware set HC = Hardware clear

bit 7 **TXWE:** Transmit Write Error Status bit (2)
1 = A new byte of data was written to I2CxTXB when it was full (Must be cleared by software)
0 = No transmit write error

bit 6 **Unimplemented:** Read as '0'

bit 5 **TXBE:** Transmit Buffer Empty Status bit
1 = I2CxTXB is empty (Cleared by writing the I2CTXB register)
0 = I2CxTXB is full

bit 4 **Unimplemented:** Read as '0'

bit 3 **RXRE:** Receive Read Error Status bit
1 = A byte of data was read from I2CxRXB when it was empty. (Must be cleared by software)
0 = No receive overflow

bit 2 **CLRBF:** Clear Buffer bit
Setting this bit clears/empties the receive and transmit buffers, causing reset of RXBF and TXBE.
Setting this bit clears the I2CxRXIF and I2CxTXIF interrupt flags.
This bit is set-only special function, and always reads '0'

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

bit 0 **RXBF:** Receive Buffer Full Status bit
1 = I2CxRXB has received new data (Cleared by reading the I2CxRXB register)
0 = I2CxRXB is empty

Note 1: The bits are held in Reset when EN = 0.
2: Will cause NACK to be sent for slave address and master/slave data read bytes.
3: Used as triggers for DMA operation.

REGISTER 33-8: I2CxERR: I²C ERROR REGISTER

U-0	R/W/HS-0	R/W/HS-0	R/W/HS-0	U-0	R/W-0	R/W-0	R/W-0
—	BTOIF(1,2)	BCLIF(1)	NACKIF(1)	—	BTOIE	BCLIE	NACKIE
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	HS = Hardware set HC = Hardware clear

bit 7 **Unimplemented:** Read as '0'

bit 6 **BTOIF:** Bus Time-Out Interrupt Flag bit(1,2)
1 = Bus Timeout occurred
0 = No bus timeout

bit 5 **BCLIF:** Bus Collision Detect Interrupt Flag bit(1)
1 = Bus collision detected (On the rising edge of SCL input, SDA output is high and input is sampled low)
Slave and Master Mode the module immediately goes idle
Multi-Master Mode attempts to match slave addresses, and/or goes idle
0 = No bus collision detected

bit 4 **NACKIF:** NACK Detect Interrupt Flag bit(1)
1 = When (SMA = 1 || MMA = 1) and a NACK is detected on the bus
NACKIF is also set when any of the TXWE, RXRE, TXU, or RXO bits are set.
0 = No NACK/Error detected
NACKIF is **not** set by the NACK send for non-matching slave addresses

bit 3 **Unimplemented:** Read as '0'

bit 2 **BTOIE:** Bus Time-Out Interrupt Enable bit
1 = Enable interrupt on bus time out
0 = Bus time-out not enabled

bit 1 **BCLIE:** Bus Collision Detect Interrupt Enable bit
1 = Enable interrupt on bus collision
0 = Bus collision interrupts are disabled

bit 0 **NACKIE:** NACK Detect Interrupt Enable bit
1 = Enable interrupt on NACKIF
0 = NACKIF interrupt is disabled

Note 1: Enabled error interrupt flags are OR'd to produce the PIRx[I2CxEIF] bit.
2: User software must select the Bus Time-out Source in the I2CxBTO register.

REGISTER 33-9: I2CxCNT: I²C BYTE COUNT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
CNT[7:0]							
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	HS = Hardware set	HC = Hardware clear

bit 7-0 **CNT[7:0]:** I²C Byte Count Register bits

If receiving data,

decremented 8th SCL edge, when a new data byte is loaded into I2CxRXB

If transmitting data,

decremented 9th SCL edge, when a new data byte is moved from I2CxTXB

CNTIF flag is set on 9th falling SCL edge, when I2CxCNT = 0. (Byte count cannot decrement past ‘0’)

Note 1: It is recommended to write this register only when the module is Idle (MMA = 0, SMA = 0) or when clock stretching (CSTR = 1 || MDR = 1).

REGISTER 33-10: I2CxPIR: I2CxIF INTERRUPT FLAG REGISTER

R/W/HS-0	R/W/HS-0	U-0	R/W/HS-0	R/W/HS-0	R/W/HS-0	R/W/HS-0	R/W/HS-0
CNTIF	ACKTIF	—	WRIF	ADRIF	PCIF	RSCIF	SCIF
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	HS = Hardware set HC = Hardware clear

bit 7 **CNTIF:** Byte Count Interrupt Flag bit
1 = When I2CxCNT = 0, set by the 9th falling edge of SCL.
0 = I2CxCNT condition has not occurred.

bit 6 **ACKTIF:** Acknowledge Status Time Interrupt Flag bit (2) (MODE[2:0] = 0xx OR 11x)
1 = Set by the 9th falling edge of SCL for any byte when addressed as a Slave
0 = Acknowledge condition not detected.

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **WRIF:** Data Write Interrupt Flag bit (MODE[2:0] = 0xx OR 11x)
1 = Set the 8th falling edge of SCL for a received data byte.
0 = Data Write condition not detected

bit 3 **ADRIF:** Address Interrupt Flag bit (MODE[2:0] = 0xx OR 11x)
1 = Set the 8th falling edge of SCL for a matching received (high/low) address byte
0 = Address condition not detected

bit 2 **PCIF:** Stop Condition Interrupt Flag
1 = Set on detection of Stop condition
0 = No Stop condition detected

bit 1 **RSCIF:** Restart Condition Interrupt Flag
1 = Set on detection of Restart condition
0 = No Restart condition detected

bit 0 **SCIF:** Start Condition Interrupt Flag
1 = Set on detection of Start condition
0 = No Start condition detected

Note 1: Enabled interrupt flags are OR’d to produce the PIRx[I2CxIF] bit.
2: ACKTIF is not set by a matching, 10-bit, high address byte with the R/$\overline{W}$ bit clear. It is only set after the matching low address byte is shifted in.

REGISTER 33-11: I2CxPIE: I2CxIE INTERRUPT AND HOLD ENABLE REGISTER

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE
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	HS = Hardware set	HC = Hardware clear

bit 7 **CNTIE:** Byte Count Interrupt Enable bit
1 = When CNTIF is set
0 = Byte count interrupts are disabled

bit 6 **ACKTIE:** Acknowledge Interrupt and Hold Enable bit
1 = When ACKTIF is set
If $\overline{\text{ACK}}$ is generated, CSTR is also set.
If NACK is generated, CSTR is unchanged
0 = Acknowledge holding and interrupt is disabled

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **WRIE:** Data Write Interrupt and Hold Enable bit
1 = When WRIF is set; CSTR is set
0 = Data Write holding and interrupt is disabled

bit 3 **ADRIE:** Address Interrupt and Hold Enable bit
1 = When ADRIF is set; CSTR is set
0 = Address holding and interrupt is disabled

bit 2 **PCIE:** Stop Condition Interrupt Enable
1 = Enable interrupt on detection of Stop condition
0 = Stop detection interrupts are disabled

bit 1 **RSCIE:** Restart Condition Interrupt Enable
1 = Enable interrupt on detection of Restart condition
0 = Start detection interrupts are disabled

bit 0 **SCIE:** Start Condition Interrupt Enable
1 = Enable interrupt on detection of Start condition
0 = Start detection interrupts are disabled

Note 1: Enabled interrupt flags are OR’d to produce the PIRx[I2CxIF] bit.

REGISTER 33-12: I2CxADR0: I²C ADDRESS 0 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
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	HS = Hardware set	HC = Hardware clear

bit 7-0 **ADR[7-0]:** Address 0 bits

MODE[2:0] = 00x | 11x - 7-bit Slave/Multi-Master Modes

ADR0[7:1]:7-bit Slave Address

ADR0[0]: Unused in this mode; bit state is a don’t care

MODE[2:0] = 01x - 10-bit Slave Modes

ADR0[7:0]: Eight Least Significant bits of 10-bit address 0

REGISTER 33-13: I2CxADR1: I²C ADDRESS 1 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8	—
bit 7							bit 0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	—
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	HS = Hardware set	HC = Hardware clear

bit 7-1 **ADR[7-1]:** Address 1 bits

MODE[2:0] = 000 | 110 - 7-bit Slave/Multi-Master modes
ADR[7:1]:7-bit Slave Address

MODE[2:0] = 001 | 111 - 7-bit Slave/Multi-Master modes w/Masking
ADR[7:1]: 7-bit Slave Address Mask

MODE[2:0] = 01x - 10-bit Slave Modes
ADR[14-10]: Bit pattern sent by master is fixed by I²C specification and must be equal to ‘11110’. However, these bit values are compared by hardware to the received data to determine a match. It is up to the user to set these bits as ‘11110’.
ADR[9-8]: Two Most Significant bits of 10-bit address

bit 0 **Unimplemented:** Read as ‘0’.

REGISTER 33-14: I2CxADR2: I²C ADDRESS 2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
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	HS = Hardware set	HC = Hardware clear

bit 7-0 **ADR[7-0]:** Address 2 bits

MODE[2:0] = 000 | 110 - 7-bit Slave/Multi-Master Modes
ADR[7:1]: 7-bit Slave Address

MODE[2:0] = 001 | 111 - 7-bit Slave/Multi-Master Modes with Masking
ADR[7:1]: 7-bit Slave Address

MODE[2:0] = 010 - 10-Bit Slave Mode
ADR[7:0]: Eight Least Significant bits of second 10-bit address

MODE[2:0] = 011 - 10-Bit Slave Mode with Masking
ADR[7-0]: Eight least significant bits of 10-bit address mask

REGISTER 33-15: I2CXADR3: I²C ADDRESS 3 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8	—
bit 15							bit 8
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	—
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	HS = Hardware set	HC = Hardware clear

bit 7-1 **ADR[7-1]:** Address 3 bits

MODE[2:0] = 000 | 110 - 7-bit Slave/Multi-Master modes
ADR[7:1]: 7-bit Slave Address

MODE[2:0] = 001 | 111 - 7-bit Slave/Multi-Master mode with Masking
ADR[7:1]: 7-bit Slave Address

MODE[2:0] = 010 - 10-Bit Slave Mode
ADR[14-10]: Bit pattern sent by master is fixed by I²C specification and must be equal to ‘11110’. However, these bit values are compared by hardware to the received data to determine a match. It is up to the user to set these bits as ‘11110’
ADR[9-8]: Two Most Significant bits of 10-bit address

MODE[2:0] = 011 - 10-Bit Slave Mode with Masking
ADR[14-8]: 10-bit high address mask

bit 0 **Unimplemented:** Read as ‘0’.

REGISTER 33-16: I2CxADB0: I²C ADDRESS DATA BUFFER 0 REGISTER(1)

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
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	HS = Hardware set	HC = Hardware clear

bit 7-0 MODE[2:0] = 00x
ADB[7:1]: Address Data byte
Received matching 7-bit slave address data
R/$\overline{W}$: Read/not-Write Data bit
Received read/write value from 7-bit address byte

MODE[2:0] = 01x
ADB[7:0]: Address Data byte
Received matching lower 8-bits of 10-bit slave address data

MODE[2:0] = 100
Unused in this mode; bit state is a don't care

MODE[2:0] = 101
ADB[7:0]: Low Address Data byte
Low 10-bit address value copied to transmit Shift register

MODE[2:0] = 11x
ADB[7:1]: Address Data byte
Received matching 7-bit slave address
R/$\overline{W}$: Read/not-Write Data bit
Received read/write value received 7-bit slave address byte

Note 1: This register is read only except in master, 10-bit Address mode (MODE[2:0] = 101).

REGISTER 33-17: I2CxADB1: I²C ADDRESS DATA BUFFER 1 REGISTER(1)

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
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	HS = Hardware set	HC = Hardware clear

bit 7-0 MODE[2:0] = 00x
Unused in this mode; bit state is a don't care
MODE[2:0] = 01x
ADB[7:1]: 10-bit Address High byte
Received matching 10-bit high address data
R/$\overline{W}$: Read/not-Write Data bit
Received read/write value from matching 10-bit high address
MODE[2:0] = 100
ADB[7:1]: Address Data byte
7-bit address value copied to transmit Shift register
R/$\overline{W}$: Read/not-Write Data bit
Read/write value copied to transmit Shift register
MODE[2:0] = 101
ADB[7:1]: 10-bit Address High Data byte
10-bit high address value copied to transmit Shift register
R/$\overline{W}$: Read/not-Write Data bit
Read/write value copied to transmit Shift register
MODE[2:0] = 11x
ADB[7:1]: Address Data byte
7-bit address value copied to transmit Shift register
R/$\overline{W}$: Read/not-Write Data bit
Read/write value copied to transmit Shift register

Note 1: This register is read only in slave, 7-bit Addressing modes (MODE[2:0] = 0xx)

TABLE 33-18: SUMMARY OF REGISTERS FOR I²C 8-BIT MACRO

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
I2CxBTO	—	—	—	—	—	BTO[2:0]			584
I2CxCLK	—	—	—	—	—	CLK[2:0]			583
I2CxPIE	CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE	590
I2CxPIR	CNTIF	ACKTIF	—	WRIF	ADRIF	PCIF	RSCIF	SCIF	589
I2CxERR	—	BTOIF	BCLIF	NACKIF	—	BTOIE	BCLIE	NACKIE	587
I2CxSTAT0	BFRE	SMA	MMA	R	D	—	—	—	585
I2CxSTAT1	TXWE	—	TXBE	—	RXRE	CLRBF	—	RXBF	586
I2CxCON0	EN	RSEN	S	CSTR	MDR	MODE[2:0]			579
I2CxCON1	ACKCNT	ACKDT	ACKSTAT	ACKT	—	RXO	TXU	CSD	581
I2CxCON2	ACNT	GCEN	FME	ABD	SDAHT[3:2]		BFRET[1:0]		582
I2CxADR0	ADR[7:0]								591
I2CxADR1	ADR[7:1]							—	592
I2CxADR2	ADR[7:0]								593
I2CxADR3	ADR[7:1]							—	594
I2CxADB0	ADB[7:0]								595
I2CxADB1	ADB[7:0]								596
I2CxCNT	CNT[7:0]								588

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the I²C module.

34.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of VDD, with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to the following:

- ADC input channel
- ADC positive reference
- Comparator input
- Digital-to-Analog Converter (DAC)

The FVR can be enabled by setting the EN bit of the FVRCON register.

Note:	Fixed Voltage Reference output cannot exceed VDD.

34.1 Independent Gain Amplifiers

The output of the FVR, which is connected to the ADC, Comparators, and DAC, is routed through two independent programmable gain amplifiers. Each amplifier can be programmed for a gain of 1x, 2x or 4x, to produce the three possible voltage levels.

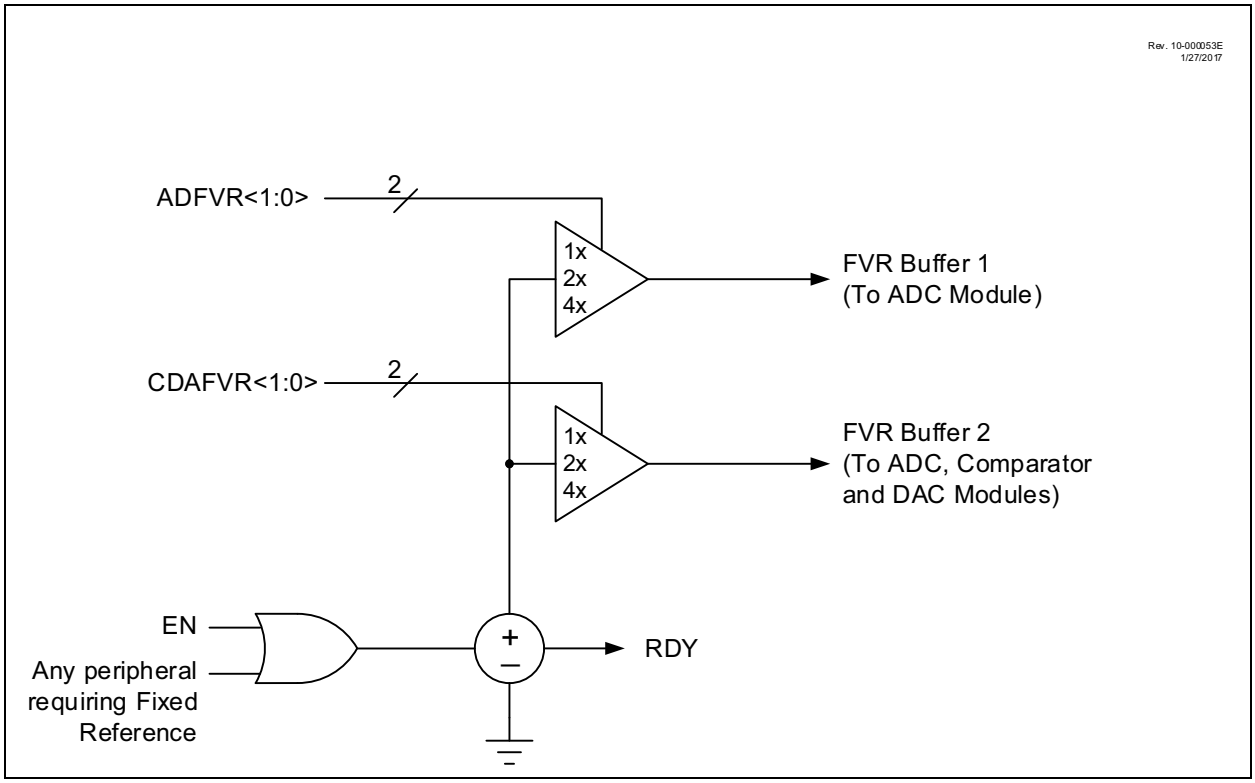
The ADFVR[1:0] bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference **Section 36.0 "Analog-to-Digital Converter with Computation (ADC2) Module"** for additional information.

The CDAFVR[1:0] bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and comparator module. Reference **Section 37.0 "5-Bit Digital-to-Analog Converter (DAC) Module"** and **Section 38.0 "Comparator Module"** for additional information.

34.2 FVR Stabilization Period

When the Fixed Voltage Reference module is enabled, it requires time for the reference and amplifier circuits to stabilize. Once the circuits stabilize and are ready for use, the RDY bit of the FVRCON register will be set.

FIGURE 34-1: VOLTAGE REFERENCE BLOCK DIAGRAM



34.3 Register Definitions: FVR Control

REGISTER 34-1: FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER

R/W-0/0	R-q/q	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	RDY(1)	TSEN(3)	TSRNG(3)	CDAFVR[1:0]		ADFVR[1:0]	
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 **EN:** Fixed Voltage Reference Enable bit
1 = Fixed Voltage Reference is enabled
0 = Fixed Voltage Reference is disabled

bit 6 **RDY:** Fixed Voltage Reference Ready Flag bit(1)
1 = Fixed Voltage Reference output is ready for use
0 = Fixed Voltage Reference output is not ready or not enabled

bit 5 **TSEN:** Temperature Indicator Enable bit(3)
1 = Temperature Indicator is enabled
0 = Temperature Indicator is disabled

bit 4 **TSRNG:** Temperature Indicator Range Selection bit(3)
1 = V_{OUT} = 3V_T (High Range)
0 = V_{OUT} = 2V_T (Low Range)

bit 3-2 **CDAFVR[1:0]:** FVR Buffer 2 Gain Selection bits
11 = FVR Buffer 2 Gain is 4x, (4.096V)(2)
10 = FVR Buffer 2 Gain is 2x, (2.048V)(2)
01 = FVR Buffer 2 Gain is 1x, (1.024V)
00 = FVR Buffer 2 is off

bit 1-0 **ADFVR[1:0]:** FVR Buffer 1 Gain Selection bit
11 = FVR Buffer 1 Gain is 4x, (4.096V)(2)
10 = FVR Buffer 1 Gain is 2x, (2.048V)(2)
01 = FVR Buffer 1 Gain is 1x, (1.024V)
00 = FVR Buffer 1 is off

Note 1: RDY is always ‘1’.
2: Fixed Voltage Reference output cannot exceed V_{DD}.
3: See **Section 35.0 “Temperature Indicator Module”** for additional information.

TABLE 34-1: SUMMARY OF REGISTERS ASSOCIATED WITH FIXED VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
FVRCON	EN	RDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]		599

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used with the Fixed Voltage Reference.

35.0 TEMPERATURE INDICATOR MODULE

This family of devices is equipped with a temperature circuit designed to measure the operating temperature of the silicon die.

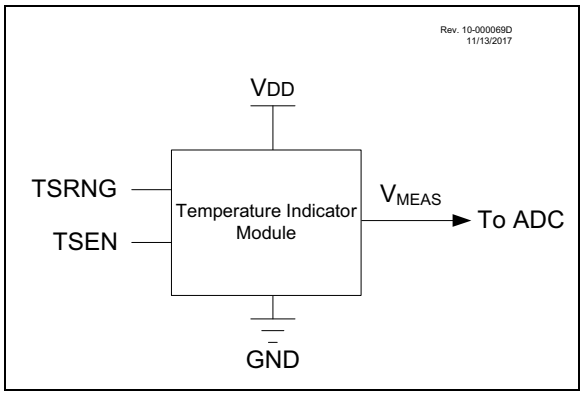
The circuit's range of operating temperature falls between -40°C and +125°C. A one-point calibration allows the circuit to indicate a temperature closely surrounding that point. A two-point calibration allows the circuit to sense the entire range of temperature more accurately.

35.1 Module Operation

The temperature indicator module consists of a temperature-sensing circuit that provides a voltage to the device ADC. The analog voltage output, VMEAS, varies inversely to the device temperature. The output of the temperature indicator is referred to as VMEAS.

Figure 35-1 shows a simplified block diagram of the temperature indicator module.

FIGURE 35-1: TEMPERATURE INDICATOR MODULE BLOCK DIAGRAM



The output of the circuit is measured using the internal Analog-to-Digital Converter. A channel is reserved for the temperature circuit output. Refer to **Section 36.0 "Analog-to-Digital Converter with Computation (ADC2) Module"** for detailed information.

The ON/OFF bit for the module is located in the FVRCON register. See **Section 34.0 "Fixed Voltage Reference (FVR)"** for more information. The circuit is enabled by setting the TSEN bit of the FVRCON register. When the module is disabled, the circuit draws no current.

The circuit operates in either High or Low range. Refer to the next section for more details on the range settings.

35.1.1 TEMPERATURE INDICATOR RANGE

The temperature indicator circuit operates in either high or low range. The high range, selected by setting the TSRNG bit of the FVRCON register, provides a wider output voltage. This provides more resolution over the temperature range. High range requires a higher-bias voltage to operate and thus, a higher VDD is needed. The low range is selected by clearing the TSRNG bit of the FVRCON register. The low range generates a lower sensor voltage and thus, a lower VDD voltage is needed to operate the circuit.

The output voltage of the sensor is the highest value at -40°C and the lowest value at +125°C.

High Range: The High range is used in applications with the reference for the ADC, VREF = 2.048V. This range may not be suitable for battery-powered applications. The ADC reading (in counts) at 90°C for the high range setting is stored in the DIA Table (Table 5-3) as parameter TSHR2.

Low Range: This mode is useful in applications in which the VDD is too low for high-range operation. The VDD in this mode can be as low as 1.8V. VDD must, however, be at least 0.5V higher than the maximum sensor voltage depending on the expected low operating temperature. The ADC reading (in counts) at 90°C for the Low range setting is stored in the DIA Table (Table 5-3) as parameter TSLR2.

35.1.2 MINIMUM OPERATING VDD

When the temperature circuit is operated in low range, the device may be operated at any operating voltage that is within specifications. When the temperature circuit is operated in high range, the device operating voltage, VDD, must be high enough to ensure that the temperature circuit is correctly biased.

Table 35-1 shows the recommended minimum VDD vs. Range setting.

TABLE 35-1: RECOMMENDED VDD vs. RANGE

Min.VDD, TSRNG = 1 (High Range)	Min. VDD, TSRNG = 0 (Low Range)
≥ 2.5	≥ 1.8

35.2 Temperature Calculation

This section describes the steps involved in calculating the die temperature, TMEAS:

1. Obtain the ADC count value of the measured analog voltage: The analog output voltage, VMEAS is converted to a digital count value by the Analog to Digital Converter (ADC) and is referred to as ADCMEAS.
2. Obtain the ADC count value, ADCDIA at 90 degrees, from the DIA table. This parameter is TSLR2 for the low range setting or TSHR2 for the high range setting of the temperature indicator module.
3. Obtain the output analog voltage (in mV) value of the Fixed Reference Voltage (FVR) for 2x setting, from the DIA Table. This parameter is FVRA2X in the DIA table (Table 5-3).
4. Obtain the value of the temperature indicator voltage sensitivity, parameter Mv, from Table 44-26 for the corresponding range setting.

Equation 35-1 provides an estimate for the die temperature based on the above parameters.

EQUATION 35-1: SENSOR TEMPERATURE

$$T_{MEAS} = 90 + \frac{(ADC_{MEAS} - ADC_{DIA}) \times FVRA2X}{(2^N - 1) \times Mv}$$

Where:
ADCMEAS = ADC reading at temperature being estimated
ADCDIA = ADC reading stored in the DIA
FVRA2X = FVR value stored in the DIA for 2x setting
N = Resolution of the ADC
Mv = Temperature Indicator voltage sensitivity (mV/°C)

Note: It is recommended to take the average of 10 measurements of ADCmeas to reduce noise and improve accuracy.

35.2.1 CALIBRATION

35.2.1.1 Higher-Order Calibration

If the application requires more precise temperature measurement, additional calibrations steps will be necessary. For these applications, two-point or three-point calibration is recommended.

35.2.2 TEMPERATURE RESOLUTION

The resolution of the ADC reading, Ma (°C/count), depends on both the ADC resolution N and the reference voltage used for conversion, as shown in Equation 35-1. It is recommended to use the smallest VREF value, such as the ADC FVR1 Output Voltage for 2x setting (FVRA2X) value from the DIA. Refer to Table 5-3 for DIA location.

Note: Refer to Table 44-18 for FVR reference voltage accuracy.

35.3 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait a certain minimum acquisition time (parameter TS01 in Table 44-26) for the ADC value to settle, after the ADC input multiplexer is connected to the temperature indicator output, before the conversion is performed.

TABLE 35-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR(1)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
FVRCON	EN	RDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]		599

Legend: — = Unimplemented location, read as '0'. Shaded cells are unused by the temperature indicator module.
Note 1: It is recommended to take the average of ten measurements of ADCMEAS to reduce noise and improve accuracy.

36.0 ANALOG-TO-DIGITAL CONVERTER WITH COMPUTATION (ADC^2) MODULE

The Analog-to-Digital Converter with Computation (ADC^2) allows conversion of an analog input signal to a 12-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 12-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair).

Additionally, the following features are provided within the ADC module:

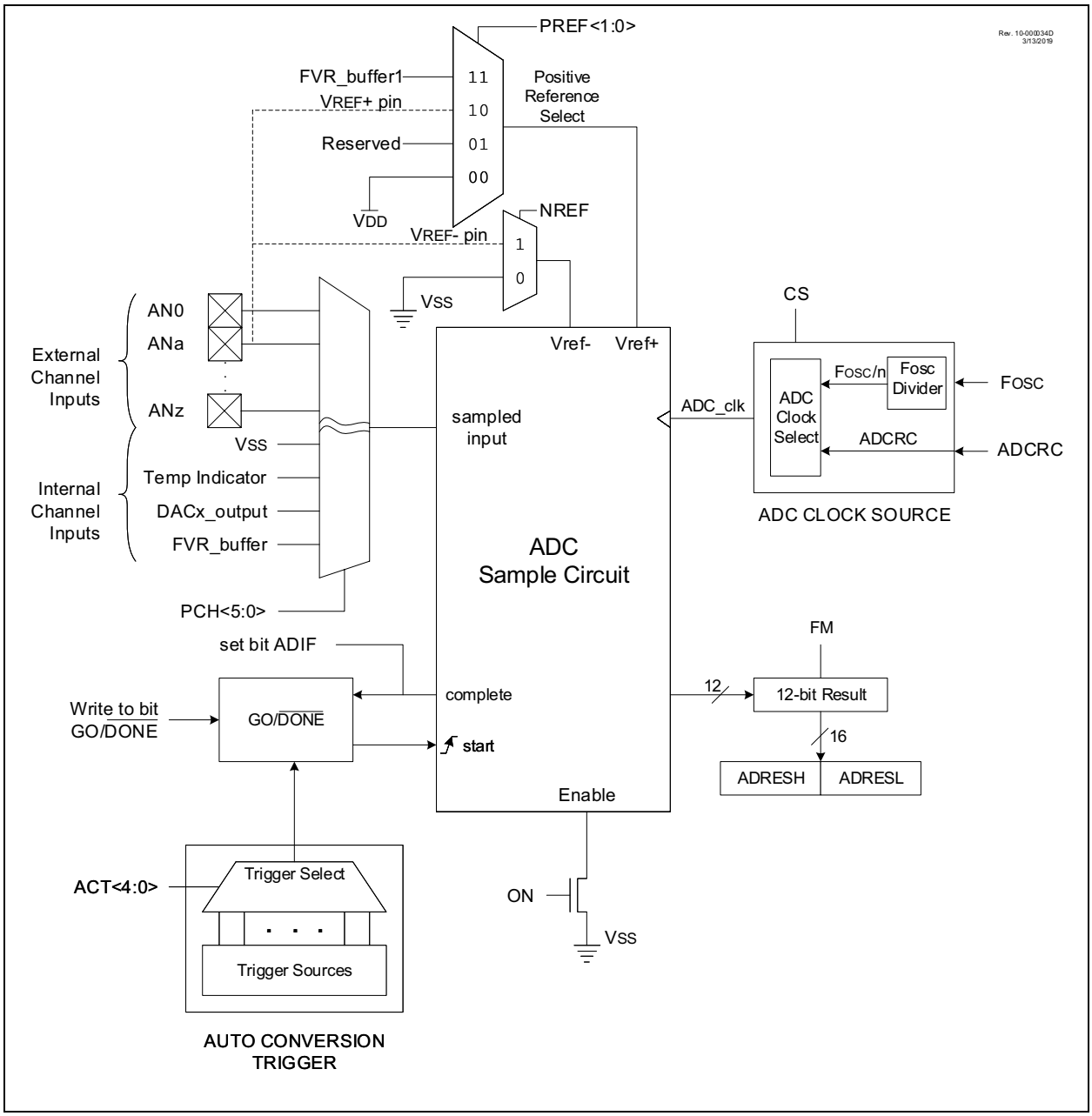
- 13-bit Acquisition Timer
- Hardware Capacitive Voltage Divider (CVD) support:
  - 13-bit Precharge Timer
  - Adjustable sample and hold capacitor array
  - Guard ring digital output drive
- Automatic repeat and sequencing:
  - Automated double sample conversion for CVD
  - Two sets of result registers (Result and Previous result)
  - Auto-conversion trigger
  - Internal retrigger
- Computation features:
  - Averaging and Low-Pass Filter functions
  - Reference Comparison
  - 2-level Threshold Comparison
  - Selectable Interrupts

Figure 36-1 shows the block diagram of the ADC.

The ADC voltage reference is software selectable to be either internally generated or externally supplied.

The ADC can generate an interrupt upon completion of a conversion and upon threshold comparison. These interrupts can be used to wake up the device from Sleep.

FIGURE 36-1: ADC² BLOCK DIAGRAM



36.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting
- Conversion Trigger Selection
- ADC Acquisition Time
- ADC Precharge Time
- Additional Sample and Hold Capacitor
- Single/Double Sample Conversion
- Guard Ring Outputs

36.1.1 PORT CONFIGURATION

The ADC will convert the voltage level on a pin whether or not the ANSEL bit is set. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to **Section 16.0 "I/O Ports"** for more information.

Note:	Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

36.1.2 CHANNEL SELECTION

There are several channel selections available:

- Eight PORTA pins (RA[7:0])
- Eight PORTB pins (RB[7:0])
- Eight PORTC pins (RC[7:0])
- Eight PORTD pins (RD[7:0], PIC18(L)F45/46/47/55/56/57K42 only)
- Three PORTE pins (RE[2:0], PIC18(L)F45/46/47/55/56/57K42 only)
- Eight PORTF pins (RD[7:0], PIC18(L)F55/56/57K42 only)
- Temperature Indicator
- DAC output
- Fixed Voltage Reference (FVR)
- VSS (ground)

The ADPCH register determines which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion.

Refer to **Section 36.2 "ADC Operation"** for more information.

36.1.3 ADC VOLTAGE REFERENCE

The PREF[1:0] bits of the ADREF register provide control of the positive voltage reference. The positive voltage reference can be:

- VREF+ pin
- VDD
- FVR outputs

The NREF bit of the ADREF register provides control of the negative voltage reference. The negative voltage reference can be:

- VREF- pin
- VSS

See **Section 34.0 "Fixed Voltage Reference (FVR)"** for more details on the Fixed Voltage Reference.

36.1.4 CONVERSION CLOCK

The conversion clock source is selected with the CS bit in the ADCON0 register. When CS = `1` the ADC clock source is an internal fixed-frequency clock referred to as ADCRC. When CS = `0` the ADC clock source is derived from FOSC.

Note:	When ADCON0.CS = `0`, the clock can be divided using the ADCLK register to meet the ADC clock period requirements.

The time to complete one bit conversion is defined as TAD. Refer Figure 36-2 for the complete timing details of the ADC conversion.

For correct conversion, the appropriate TAD specification must be met. Refer to Table 44-15 for more information. Table 36-1 gives examples of appropriate ADC clock selections.

Note 1:	Unless using the ADCRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.
2:	The internal control logic of the ADC runs off of the clock selected by the CS bit of ADCON0. What this can mean is when the CS bit of ADCON0 is set to '`1`' (ADC runs on ADCRC), there may be unexpected delays in operation when setting ADC control bits.

TABLE 36-1: ADC CLOCK PERIOD (TAD) VS. DEVICE OPERATING FREQUENCIES(1,3)

ADC Clock Source	CS[5:0]	Device Frequency (FOSC)						
		64 MHz TAD	32 MHz TAD	20 MHz TAD	16 MHz TAD	8 MHz TAD	4 MHz TAD	1 MHz TAD
FOSC/2	000000	31.25 ns(2)	62.5 ns(2)	100 ns(2)	125 ns(2)	250 ns(2)	500 ns	2.0 μs
FOSC/4	000001	62.5 ns(2)	125 ns(2)	200 ns(2)	250 ns(2)	500 ns	1.0 μs	4.0 μs
FOSC/6	000010	93.75 ns(2)	187.5 ns(2)	300 ns(2)	375 ns(2)	750 ns	1.5 μs	6.0 μs
FOSC/8	000011	125 ns(2)	250 ns(2)	400 ns(2)	500 ns	1.0 μs	2.0 μs	8.0 μs
...	...	...	...	...	...	...	...	...
FOSC/16	000111	250 ns(2)	500 ns	800 ns	1.0 μs	2.0 μs	4.0 μs	16.0 μs(2)
...	...	...	...	...	...	...	...	...
FOSC/128	111111	2.0 μs	4.0 μs	6.4 μs	8.0 μs	16.0 μs(2)	32.0 μs(2)	128.0 μs(2)
ADCRC	ADCON0.CS = 1	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs

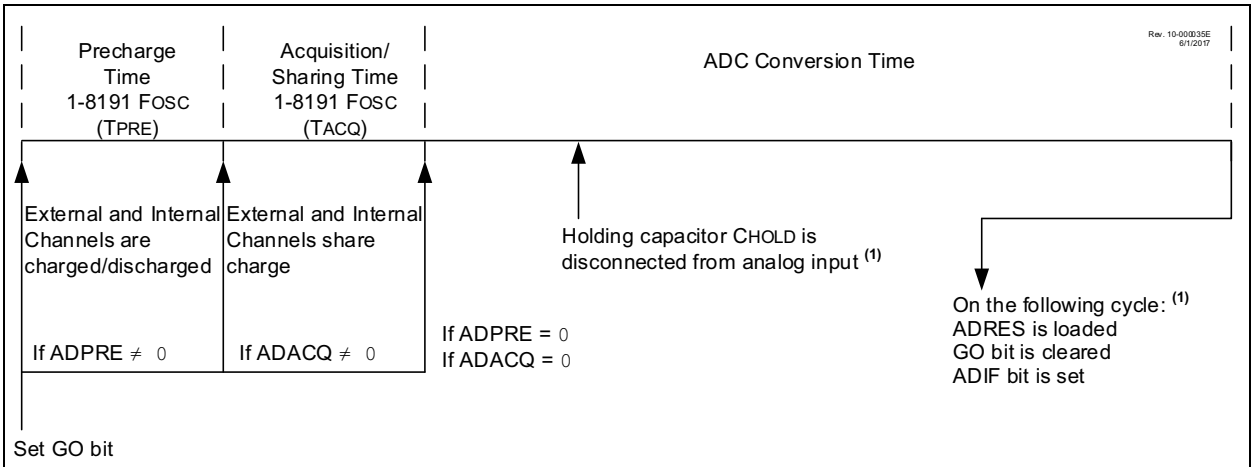
Legend: Shaded cells are outside of recommended range.

Note 1: See TAD parameter for ADCRC source typical TAD value.

2: These values violate the required TAD time.

3: The ADC clock period (TAD) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the ADCRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

FIGURE 36-2: ANALOG-TO-DIGITAL CONVERSION CYCLES



Note 1: Refer to Table 44-15 for ADC Conversion Timing Specifications.

36.1.5 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADC Interrupt Flag is the ADIF bit in the PIRx register. The ADC Interrupt Enable is the ADIE bit in the PIEx register. The ADIF bit must be cleared in software.

> **Note 1:** The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.
>
> **2:** The ADC operates during Sleep only when the ADCRC oscillator is selected.

This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake up the device. Upon waking from Sleep, the next instruction following the `SLEEP` instruction is always executed. If the user is attempting to wake-up from Sleep and resume in-line code execution, the ADIE bit of the PIEx register and the GIE bits of the INTCON0 register must both be set. If all these bits are set, the execution will switch to the Interrupt Service Routine.

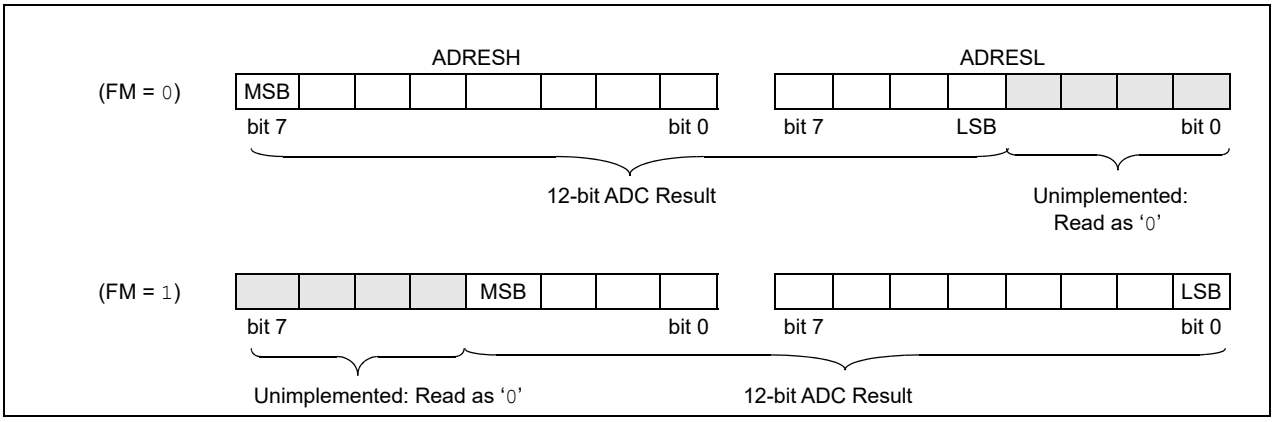
36.1.6 RESULT FORMATTING

The 12-bit ADC conversion result can be supplied in two formats, left justified or right justified. The FM bits of the ADCON0 register controls the output format.

Figure 36-3 shows the two output formats.

Writes to the ADRES register pair are always right justified regardless of the selected format mode. Therefore, data read after writing to ADRES when FM = `0` will be shifted left four places.

FIGURE 36-3: 12-BIT ADC CONVERSION RESULT FORMAT



36.2 ADC Operation

36.2.1 STARTING A CONVERSION

To enable the ADC module, the ON bit of the ADCON0 register must be set to a '1'. A conversion may be started by any of the following:

- Software setting the GO bit of ADCON0 to '1'
- An external trigger (selected by Register 36-3)
- A continuous-mode retrigger (see section **Section 36.5.8 "Continuous Sampling mode"**)

.

Note:	The GO bit should not be set in the same instruction that turns on the ADC. Refer to **Section 36.2.6 "ADC Conversion Procedure (Basic Mode)"**.

36.2.2 COMPLETION OF A CONVERSION

When any individual conversion is complete, the value already in ADRES is written into PREV (if PSIS = 1) and the new conversion results appear in ADRES. When the conversion completes, the ADC module will:

- Clear the GO bit (unless the CONT bit of ADCON0 is set)
- Set the ADIF Interrupt Flag bit
- Set the MATH bit
- Update ACC

When DSEN = 0 then after every conversion, or when DSEN = 1 then after every other conversion, the following events occur:

- ERR is calculated
- ADTIF is set if ERR calculation meets threshold comparison

Importantly, filter and threshold computations occur after the conversion itself is complete. As such, interrupt handlers responding to ADIF should check ADTIF before reading filter and threshold results.

36.2.3 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the ADCRC option. When the ADCRC oscillator source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the device remains in Sleep and the ADC module is turned off after the conversion completes, although the ON bit remains set.

36.2.4 EXTERNAL TRIGGER DURING SLEEP

If the external trigger is received during Sleep while ADC clock source is set to the ADCRC, ADC module will perform the conversion and set the ADIF bit upon completion.

If an external trigger is received when the ADC clock source is something other than ADCRC, the trigger will be recorded, but the conversion will not begin until the device exits Sleep.

36.2.5 AUTO-CONVERSION TRIGGER

The auto-conversion trigger allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the GO bit is set by hardware.

The auto-conversion trigger source is selected by the ADACT register.

Using the auto-conversion trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met. See Register 36-33 for auto-conversion sources.

36.2.6 ADC CONVERSION PROCEDURE (BASIC MODE)

This is an example procedure for using the ADC to perform an analog-to-digital conversion:

1. Configure Port:
   - Disable pin output driver (Refer to the TRISx register)
   - Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   - Select ADC conversion clock
   - Select voltage reference
   - Select ADC input channel
   - Precharge and acquisition
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable global interrupt(1)
4. If ADACQ = 0, software must wait the required acquisition time(2).
5. Start conversion by setting the GO bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the GO bit
   - Polling the ADIF bit
   - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

2: Refer to **Section 36.3 "ADC Acquisition Requirements"**.

EXAMPLE 36-1: ADC CONVERSION

```
/*This code block configures the ADC
for polling, VDD and VSS references, ADCRC
oscillator and AN0 input.
Conversion start & polling for completion
are included.
 */
void main() {
    //System Initialize
    initializeSystem();

    //Setup ADC
    ADCON0bits.FM = 1; //right justify
    ADCON0bits.CS = 1; //ADCRC Clock
    ADPCH = 0x00; //RA0 is Analog channel
    TRISAbits.TRISA0 = 1; //Set RA0 to input
    ANSELAbits.ANSELA0 = 1; //Set RA0 to analog
    ADCON0bits.ON = 1; //Turn ADC On

    while (1) {
        ADCON0bits.GO = 1; //Start conversion
        while (ADCON0bits.GO); //Wait for conversion done
        resultHigh = ADRESH; //Read result
        resultLow = ADRESL; //Read result
    }
}
```

36.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 36-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 36-4. Refer to Parameter AD08 mentioned in Table 44-14 for the maximum recommended impedance for analog sources. If the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be completed before the conversion can be started. To calculate the minimum acquisition time, Equation 36-1 may be used. This equation assumes that 1/2 LSb error is used (4,096 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

EQUATION 36-1: ACQUISITION TIME EXAMPLE

Assumptions: *Temperature = 50°C and external impedance of 1kΩ 5.0V VDD*

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 2\mu s + T_C + [(\text{Temperature} - 25°C)(0.05\mu s/°C)] \end{aligned}$$

The value for TC can be approximated with the following equations:

$$V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) = V_{CHOLD}$$ *;[1] VCHOLD charged to within 1/2 lsb*

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD}$$ *;[2] VCHOLD charge response to VAPPLIED*

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right)$$ *;combining [1] and [2]*

Note: Where n = number of bits of the ADC.

Solving for TC:

$$\begin{aligned} T_C &= -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ \ln(1/8191) \\ &= -28pF(1k\Omega + 7k\Omega + 1k\Omega)\ \ln(0.0001221) \\ &= 2.27\mu s \end{aligned}$$

Therefore:

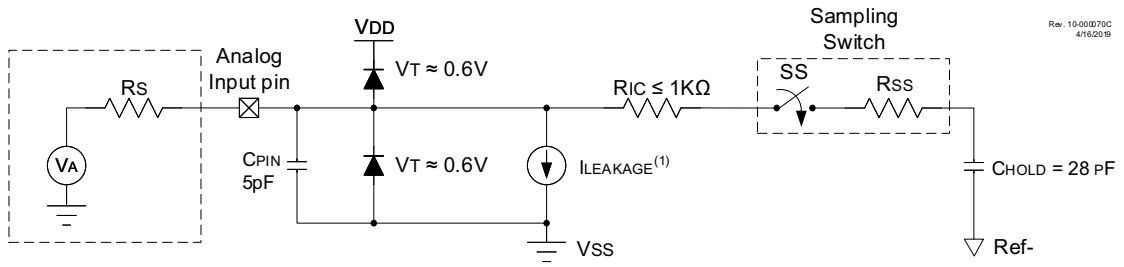
$$\begin{aligned} T_{ACQ} &= 2\mu s + 2.27\mu s + [(50°C - 25°C)(0.05\mu s/°C)] \\ &= 5.52\mu s \end{aligned}$$

Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

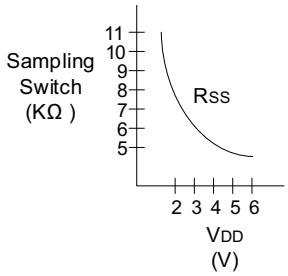
3: The maximum recommended impedance for analog sources is mentioned in Parameter AD08 in Table 44-14. This is required to meet the pin leakage specification.

FIGURE 36-4: ANALOG INPUT MODEL



Legend:

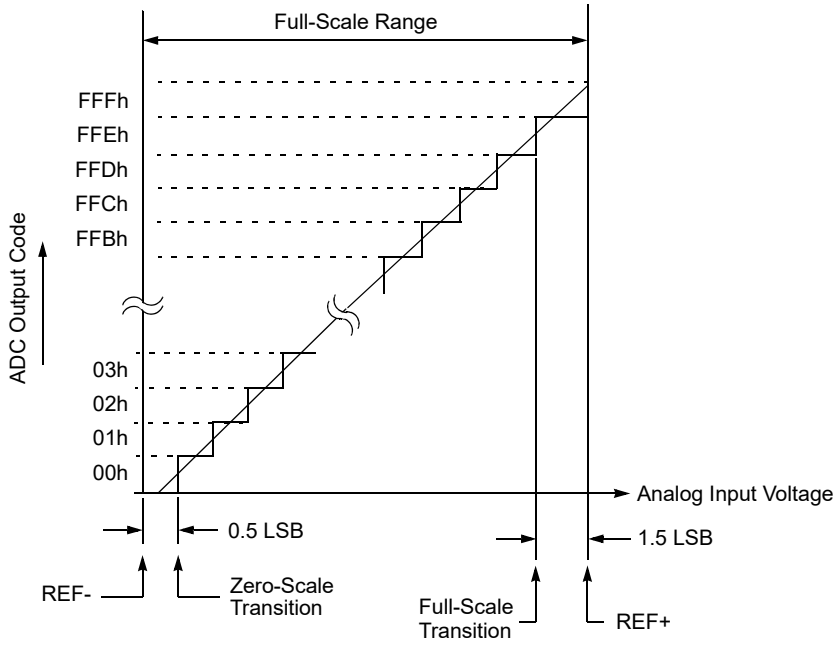
CPIN	= Input Capacitance
ILEAKAGE	= Leakage Current at the pin due to various junctions
RIC	= Interconnect Resistance
RS	= Source Impedance
VA	= Analog Voltage
VT	= Diode Forward Voltage
SS	= Sampling Switch
RSS	= Resistance of the Sampling Switch
CHOLD	= Sample/Hold Capacitance



Note:

1. Refer to Table 44-4 (parameter D340 and D341).

FIGURE 36-5: ADC TRANSFER FUNCTION



36.4 ADC Charge Pump

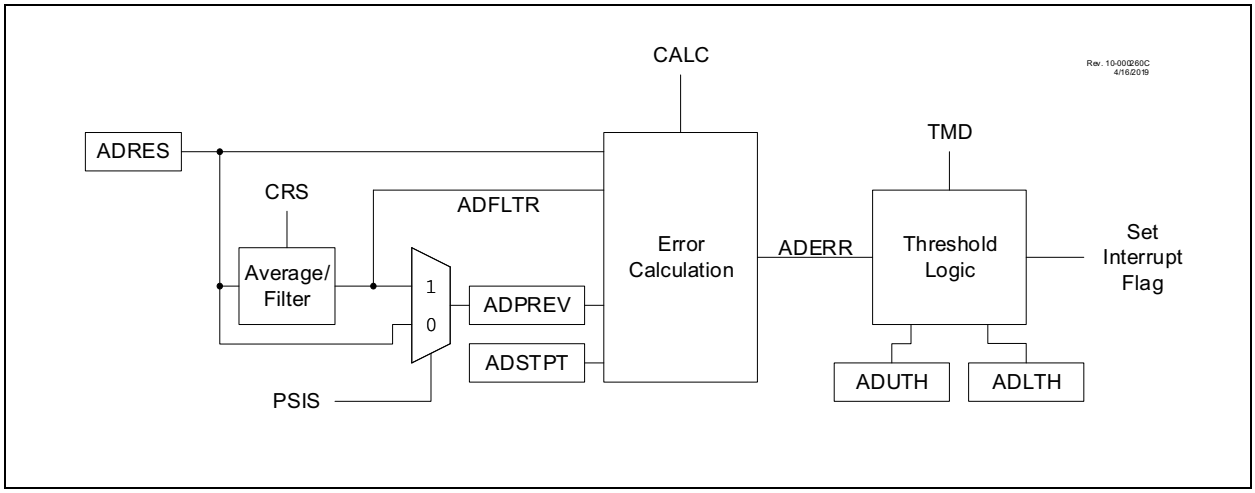
The ADC module has a dedicated charge pump which can be controlled through the ADCP register (Register 36-36). The primary purpose of the charge pump is to supply a constant voltage to the gates of transistor devices in the A/D converter, signal and reference input pass-gates, to prevent degradation of transistor performance at low operating voltage.

The charge pump can be enabled by setting the CPON bit in the ADCP register. Once enabled, the pump will undergo a start-up time to stabilize the charge pump output. Once the output stabilizes and is ready for use, the CPRDY bit of the ADCP register will be set.

36.5 Computation Operation

The ADC module hardware is equipped with post conversion computation features. These features provide data post-processing functions that can be operated on the ADC conversion result, including digital filtering/averaging and threshold comparison functions.

FIGURE 36-6: COMPUTATIONAL FEATURES SIMPLIFIED BLOCK DIAGRAM



The operation of the ADC computational features is controlled by MD[2:0] bits in the ADCON2 register.

The module can be operated in one of five modes:

• **Basic**: In this mode, ADC conversion occurs on single (DSEN = 0) or double (DSEN = 1) samples. ADIF is set after all the conversion are complete.

• **Accumulate**: With each trigger, the ADC conversion result is added to accumulator and CNT increments. ADIF is set after each conversion. ADTIF is set according to the calculation mode.

• **Average**: With each trigger, the ADC conversion result is added to the accumulator. When the RPT number of samples have been accumulated, a threshold test is performed. Upon the next trigger, the accumulator is cleared. For the subsequent tests, additional RPT samples are required to be accumulated.

• **Burst Average**: At the trigger, the accumulator is cleared. The ADC conversion results are then collected repetitively until RPT samples are accumulated and finally the threshold is tested.

• **Low-Pass Filter (LPF)**: With each trigger, the ADC conversion result is sent through a filter. When RPT samples have occurred, a threshold test is performed. Every trigger after that the ADC conversion result is sent through the filter and another threshold test is performed.

The five modes are summarized in Table 36-2 below.

TABLE 36-2: COMPUTATION MODES

		Bit Clear Conditions	Value after Trigger completion		Threshold Operations			Value at ADTIF interrupt		
Mode	MD	ACC and CNT	ACC	CNT	Retrigger	Threshold Test	Interrupt	ADAOV	FLTR	CNT
Basic	0	ACLR = 1	Unchanged	Unchanged	No	Every Sample	If threshold=true	N/A	N/A	count
Accumulate	1	ACLR = 1	S + ACC or (S2-S1) + ACC	If (CNT=0xFF): CNT, otherwise: CNT+1	No	Every Sample	If threshold=true	ACC Overflow	$ACC/2^{CRS}$	count
Average	2	ACLR = 1 or CNT>=RPT at GO or retrigger	S + ACC or (S2-S1) + ACC	If (CNT=0xFF): CNT, otherwise: CNT+1	No	If CNT>=RPT	If threshold=true	ACC Overflow	$ACC/2^{CRS}$	count
Burst Average	3	ACLR = 1 or GO set or retrigger	Each repetition: same as Average End with sum of all samples	Each repetition: same as Average End with CNT=RPT	Repeat while CNT<RPT	If CNT>=RPT	If threshold=true	ACC Overflow	$ACC/2^{CRS}$	RPT
Low-pass Filter	4	ACLR = 1	$S+ACC-ACC/2^{CRS}$ or $(S2-S1)+ACC-ACC/2^{CRS}$	Count up, stop counting when CNT = 0xFF	No	If CNT>=RPT	If threshold=true	ACC Overflow	$ACC/2^{CRS}$ (Filtered Value)	count

Note: S1 and S2 are abbreviations for Sample 1 and Sample 2, respectively. When DSEN = 0, S1 = ADRES; When DSEN = 1, S1 = PREV and S2 = ADRES.

© 2017-2020 Microchip Technology Inc.

36.5.1 DIGITAL FILTER/AVERAGE

The digital filter/average module consists of an accumulator with data feedback options, and control logic to determine when threshold tests need to be applied. The ADACC register is a 24-bit wide register which can be accessed through the ADACCU:ADACCH:ADACCL register pair. It contains 18-bit accumulator value ACC [17:0] and one extended sign bit.

Upon each trigger event (the GO bit set or external event trigger), the ADC conversion result is added to the accumulator. If the accumulated result exceeds $2^{(accumulator_width)}$-1 = 262143, the overflow bit ADAOV in the ADSTAT register is set.

The number of samples to be accumulated is determined by the RPT (A/D Repeat Setting) register. Each time a sample is added to the accumulator, the ADCNT register is incremented. Once RPT samples are accumulated (CNT = RPT), an accumulator clear command can be issued by the software by setting the ACLR bit in the ADCON2 register. Setting the ACLR bit will also clear the ADAOV (Accumulator overflow) bit in the ADSTAT register, as well as the ADCNT register. The ACLR bit is cleared by the hardware when accumulator clearing action is complete.

Note:	When ADC is operating from ADCRC, five ADCRC clock cycles are required to execute the ACC clearing operation.

The CRS [2:0] bits in the ADCON2 register control the data shift on the accumulator result, which effectively divides the value in accumulator (ADACCU:ADACCH:ADACCL) register pair. The right-shifted value is stored in the signed ADFLTRH:ADFLTRL register pair. When the value in the ADFLTR register overflows, the overflow bit ADAOV in the ADSTAT register is set. For the Accumulate mode of the digital filter, the shift provides a simple scaling operation. For the Average/Burst Average mode, the shift bits are used to determine the number of logical right shifts to be performed on the accumulated result. For the Low-pass Filter mode, the shift is an integral part of the filter, and determines the cut-off frequency of the filter. Table 36-3 shows the -3 dB cut-off frequency in ωT (radians) and the highest signal attenuation obtained by this filter at nyquist frequency (ωT = π).

TABLE 36-3: LOW-PASS FILTER -3 dB CUT-OFF FREQUENCY

ADCRS	ωT (radians) @ -3 dB Frequency	dB @ $F_{nyquist}$=1/(2T)
1	0.72	-9.5
2	0.284	-16.9
3	0.134	-23.5
4	0.065	-29.8
5	0.032	-36.0
6	0.016	-42.0

36.5.2 BASIC MODE

Basic mode (MD = `000`) disables all additional computation features. In this mode, no accumulation occurs but threshold error comparison is performed. Double sampling, Continuous mode, and all CVD features are still available, but no features involving the digital filter/average features are used.

36.5.3 ACCUMULATE MODE

In Accumulate mode (MD = `001`), after every conversion, the ADC result is added to the ADACC register. The ADACC register is right-shifted by the value of the CRS bits in the ADCON2 register. This right-shifted value is copied in to the ADFLTR register. The Formatting mode does not affect the right-justification of the ACC value. Upon each sample, CNT is also incremented, incrementing the number of samples accumulated. After each sample and accumulation, the ACC value has a threshold comparison performed on it (see **Section 36.5.7 "Threshold Comparison"**) and the ADTIF interrupt may trigger.

36.5.4 AVERAGE MODE

In Average mode (MD = `010`), the ADACC registers accumulate with each ADC sample, much as in Accumulate mode, and the ADCNT register increments with each sample. The ADFLTR register is also updated with the right-shifted value of the ADACC register. The value of the CRS bits governs the number of right shifts. However, in Average mode, the threshold comparison is performed upon CNT being greater than or equal to a user-defined RPT value. In this mode when RPT = 2^CNT, then the final accumulated value will be divided by number of samples, allowing for a threshold comparison operation on the average of all gathered samples.

36.5.5 BURST AVERAGE MODE

The Burst Average mode (MD = `011`) acts the same as the Average mode in most respects. The one way it differs is that it continuously retriggers ADC sampling until the CNT value is greater than or equal to RPT, even if Continuous Sampling mode (see **Section 36.5.8 "Continuous Sampling mode"**) is not enabled. This allows for a threshold comparison on the average of a short burst of ADC samples.

36.5.6 LOW-PASS FILTER MODE

The Low-pass Filter mode (MD = `100`) acts similarly to the Average mode in how it handles samples (accumulates samples until CNT value greater than or equal to RPT, then triggers threshold comparison), but instead of a simple average, it performs a low-pass filter operation on all of the samples, reducing the effect of high-frequency noise on the average, then performs a threshold comparison on the results. (see Table 36-2 for a more detailed description of the mathematical operation). In this mode, the CRS bits determine the cut-off frequency of the low-pass filter (as demonstrated by Table 36-3).

36.5.7 THRESHOLD COMPARISON

At the end of each computation:

- The conversion results are latched and held stable at the end-of-conversion.
- The error is calculated based on a difference calculation which is selected by the CALC[2:0] bits in the ADCON3 register and stored in the signed ADERRH:ADERRL register pair. If the value of the ADERR register overflows, the ADAOV overflow bit is set in the ADSTAT register. The value can be one of the following calculations (see Register 36-4 for more details):
  - The first derivative of single measurements
  - The CVD result in CVD mode
  - The current result vs. a setpoint
  - The current result vs. the filtered/average result
  - The first derivative of the filtered/average value
  - Filtered/average value vs. a setpoint
- The result of the calculation (ERR) is compared to the upper and lower thresholds, ADUTH and ADLTH registers, to set the UTHR and LTHR flag bits. The threshold logic is selected by TMD[2:0] bits in the ADCON3 register. The threshold trigger option can be one of the following:
  - Never interrupt
  - Error is less than lower threshold
  - Error is greater than or equal to lower threshold
  - Error is between thresholds (inclusive)
  - Error is outside of thresholds
  - Error is less than or equal to upper threshold
  - Error is greater than upper threshold
  - Always interrupt regardless of threshold test results
  - If the threshold condition is met, the threshold interrupt flag ADTIF is set.

Note 1:	The threshold tests are signed operations.
2:	If ADAOV is set, a threshold interrupt is signaled.

36.5.8 CONTINUOUS SAMPLING MODE

Setting the CONT bit in the ADCON0 register automatically retriggers a new conversion cycle after updating the ADACC register. The GO bit remains set and re-triggering occurs automatically.

If SOI = `1`, a threshold interrupt condition will clear GO and the conversions will stop.

36.5.9 DOUBLE SAMPLE CONVERSION

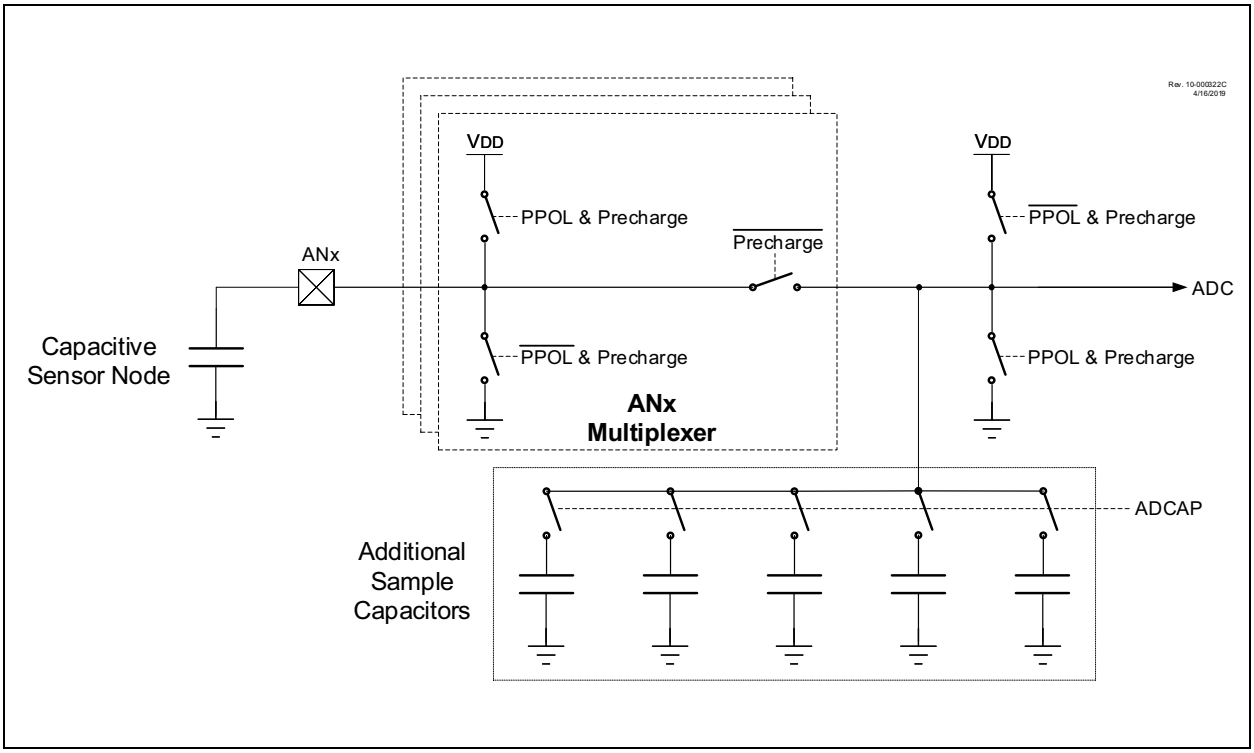
Double sampling is enabled by setting the DSEN bit of the ADCON1 register. When this bit is set, two conversions are required before the module will calculate threshold error (each conversion must still be triggered separately). The first conversion will set the MATH bit of the ADSTAT register and update ADACC, but will not calculate ERR or trigger ADTIF. When the second conversion completes, the first value is transferred to PREV (depending on the setting of PSIS) and the value of the second conversion is placed into ADRES. Only upon the completion of the second conversion is ERR calculated and ADTIF triggered (depending on the value of CALC).

© 2017-2020 Microchip Technology Inc.

36.6 Capacitive Voltage Divider (CVD) Features

The ADC module contains several features that allow the user to perform a relative capacitance measurement on any ADC channel using the internal ADC sample and hold capacitance as a reference. This relative capacitance measurement can be used to implement capacitive touch or proximity sensing applications. Figure 36-7 shows the basic block diagram of the CVD portion of the ADC module.

FIGURE 36-7: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM



This is an example to configure ADC for CVD operation:

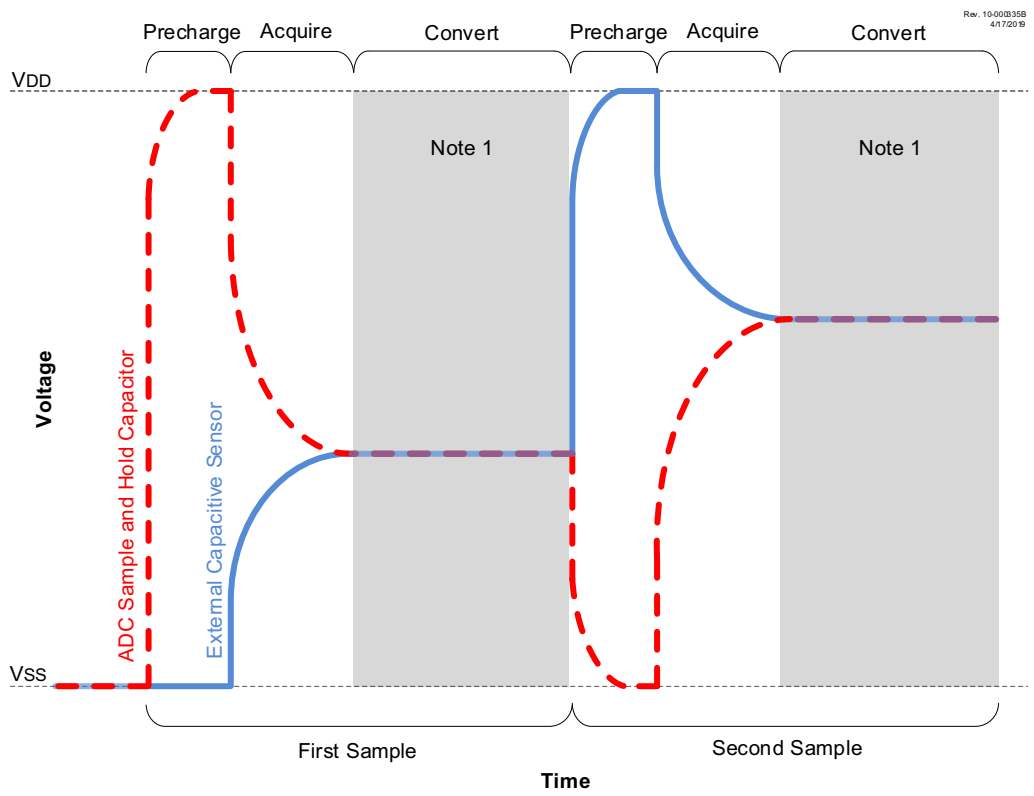
1. Configure Port:
   1.1 Disable pin output driver (Refer to the TRISx register)
   1.2 Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   2.1. Select ADC conversion clock
   2.2. Configure voltage reference
   2.3. Select ADC input channel
   2.4. Configure precharge (ADPRE) and acquisition (ADACQ) time period
   2.5. Select precharge polarity (PPOL bit)
   2.6. Enable Double Sampling (DSEN bit)
   2.7. Turn on ADC module
3. Configure ADC interrupt (optional):
   3.1. Clear ADC interrupt flag
   3.2. Enable ADC interrupt
   3.3. Enable global interrupt (GIE bit)(1)
4. Start double sample conversion by setting the GO bit.
5. Wait for ADC conversion to complete by one of the following:
   - Polling the GO bit
   - Waiting for the ADC interrupt (if interrupt is enabled)
6. Second ADC conversion depends on the state of CONT:
   6.1. If CONT = 1, both conversion will repeat automatically form a single trigger
   6.2. If CONT = 0, each conversion must be triggered separately
7. ADERR register contains the CVD result
8. Clear the ADC interrupt flag (if interrupt is enabled).

Note 1: With global interrupts disabled (GIE = 0), the device will wake from Sleep but will not enter an Interrupt Service Routine.

36.6.1 CVD OPERATION

A CVD operation begins with the ADC's internal Sample-and-Hold capacitor (CHOLD) being disconnected from the path which connects it to the external capacitive sensor node. While disconnected, CHOLD is precharged to VDD or discharged to VSS. The sensor node is either discharged or charged to VSS or VDD, respectively to the opposite level of CHOLD. When the precharge phase is complete, the VDD/VSS bias paths for the two nodes are disconnected and the paths between CHOLD and the external sensor node is reconnected, at which time the acquisition phase of the CVD operation begins. During acquisition, a capacitive voltage divider is formed between the precharged CHOLD and sensor nodes, which results in a final voltage level setting on CHOLD which is determined by the capacitances and precharge levels of the two nodes. After acquisition, the ADC converts the voltage level on CHOLD. This process is then repeated with the selected precharge levels inverted for both the CHOLD and the sensor nodes. Figure 36-8 shows the waveform for two inverted CVD measurements, which is known as differential CVD measurement.

FIGURE 36-8: DIFFERENTIAL CVD MEASUREMENT WAVEFORM



Note 1: External Capacitive Sensor voltage during the conversion phase may vary as per the configuration of the corresponding pin.

36.6.2 PRECHARGE CONTROL

The precharge stage is an optional period of time that brings the external channel and internal sample and hold capacitor to known voltage levels. Precharge is enabled by writing a non-zero value to the ADPRE register. This stage is initiated when an ADC conversion begins, either from setting the GO bit, a special event trigger, or a conversion restart from the computation functionality. If the ADPRE register is cleared when an ADC conversion begins, this stage is skipped.

During the precharge time, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the PPOL bit of ADCON1. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is also determined by the PPOL bit of ADCON1. The amount of time that this charging receives is controlled by the ADPRE register.

Note 1:	The external charging overrides the TRIS setting of the respective I/O pin.
2:	If there is a device attached to this pin, Precharge should not be used.

36.6.3 ACQUISITION CONTROL FOR CVD (ADPRE > 0)

The Acquisition stage allows time for the voltage on the internal Sample-and-Hold capacitor to charge or discharge from the selected analog channel. This acquisition time is controlled by the ADACQ register. The acquisition stage begins when precharge stage ends.

At the start of the acquisition stage, the port pin logic of the selected analog channel is overridden to turn off the digital high/low output drivers so they do not affect the final result of the charge averaging. Also, the selected ADC channel is connected to CHOLD. This allows charge averaging to proceed between the precharged channel and the CHOLD capacitor.

Note:	When PRE > 0, acquisition time cannot be '0'. In this case, setting ADACQ to '0' will set a maximum acquisition time (8191 ADC clock cycles). When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

36.6.4 GUARD RING OUTPUTS

Figure 36-9 shows a typical guard ring circuit. CGUARD represents the capacitance of the guard ring trace placed on the PCB board. The user selects values for RA and RB that will create a voltage profile on CGUARD, which will match the selected acquisition channel.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, *"mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider"* (DS01478).

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs can be routed through PPS controls to I/O pins (see **Section 17.0 "Peripheral Pin Select (PPS) Module"** for details) and the polarity of these outputs are controlled by the GPOL and IPEN bits of ADCON1.

At the start of the first precharge stage, both outputs are set to match the GPOL bit of ADCON1. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the IPEN bit of ADCON1 causes both guard ring outputs to transition to the opposite polarity of GPOL at the start of the second precharge stage, and ADGRDA toggles again for the second acquisition. For more information on the timing of the guard ring output, refer to Figure 36-9 and Figure 36-10.

FIGURE 36-9: GUARD RING CIRCUIT

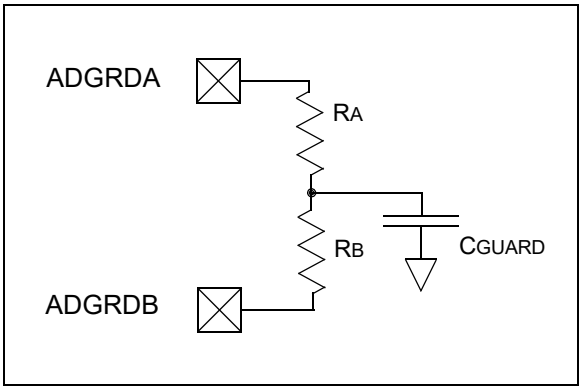
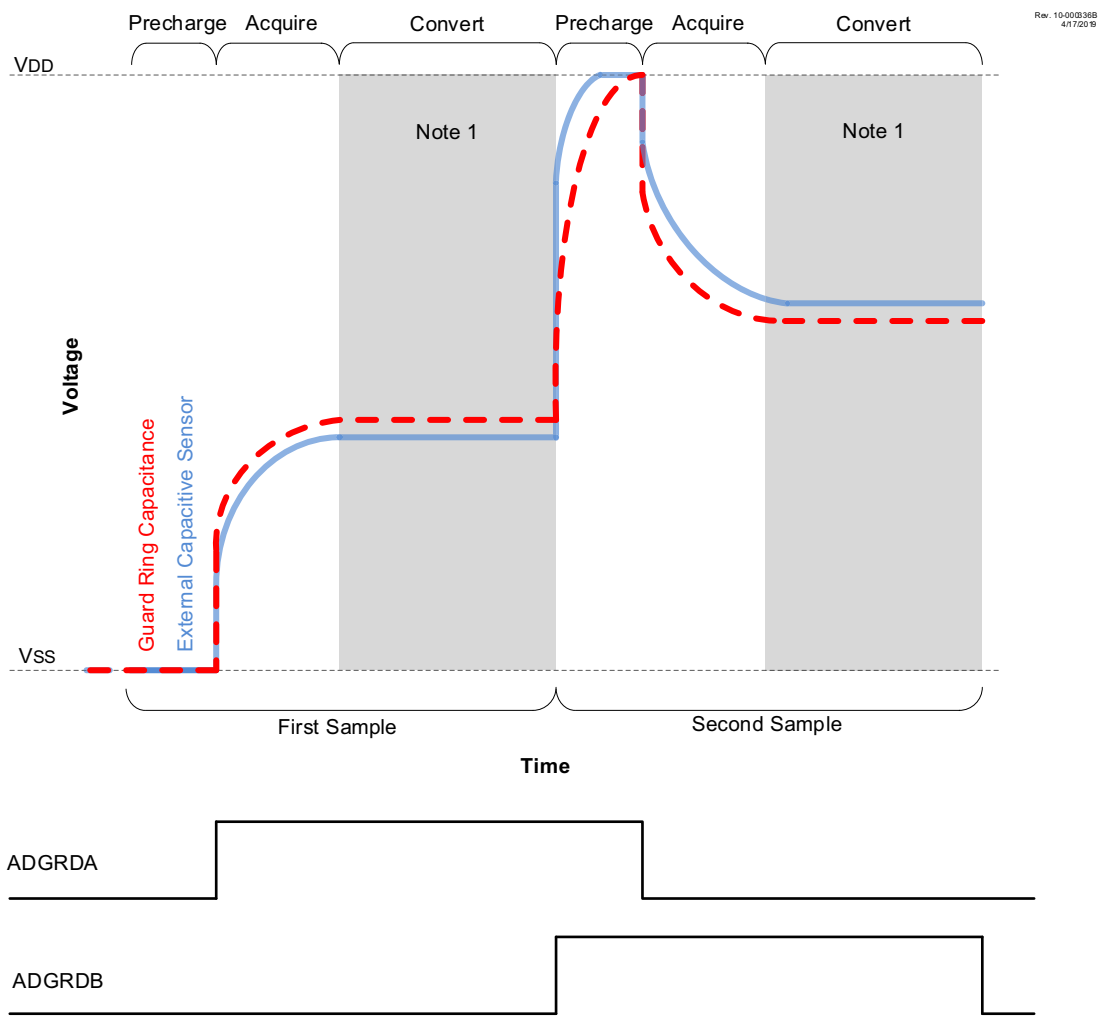


FIGURE 36-10: DIFFERENTIAL CVD WITH GUARD RING OUTPUT WAVEFORM



Note 1: External Capacitive Sensor voltage during the conversion phase may vary as per the configuration of the corresponding pin.

36.6.5 ADDITIONAL SAMPLE AND HOLD CAPACITANCE

Additional capacitance can be added in parallel with the internal sample and hold capacitor (CHOLD) by using the ADCAP register. This register selects a digitally programmable capacitance which is added to the ADC conversion bus, increasing the effective internal capacitance of the sample and hold capacitor in the ADC module. This is used to improve the match between internal and external capacitance for a better sensing performance. The additional capacitance does not affect analog performance of the ADC because it is not connected during conversion. See Figure 36-6.

36.7 Register Definitions: ADC Control

REGISTER 36-1: ADCON0: ADC CONTROL REGISTER 0

R/W-0/0	R/W-0/0	U-0	R/W-0/0	U-0	R/W-0/0	U-0	R/W/HS/HC-0/0
ON	CONT	—	CS	—	FM	—	GO
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	HC = Bit is cleared by hardware
		HS = Bit is set by hardware

bit 7 **ON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled

bit 6 **CONT:** ADC Continuous Operation Enable bit
1 = GO is retriggered upon completion of each conversion trigger until ADTIF is set (if SOI is set) or until GO is cleared (regardless of the value of SOI)
0 = ADC is cleared upon completion of each conversion trigger

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **CS:** ADC Clock Selection bit
1 = Clock supplied from ADCRC dedicated oscillator
0 = Clock supplied by FOSC, divided according to ADCLK register

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **FM:** ADC results Format/alignment Selection
1 = ADRES and PREV data are right-justified
0 = ADRES and PREV data are left-justified, zero-filled

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **GO:** ADC Conversion Status bit(1,2)
1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the CONT bit
0 = ADC conversion completed/not in progress

Note 1: This bit requires ON bit to be set.

2: If cleared by software while a conversion is in progress, the results of the conversion up to this point will be transfered to ADRES and the state machine will be reset, but the ADIF interrupt flag bit will not be set; filter and threshold operations will not be performed.

REGISTER 36-2: ADCON1: ADC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0
PPOL	IPEN	GPOL	—	—	—	—	DSEN
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 **PPOL:** Precharge Polarity bit
If ADPRE != 0x00:

PPOL	Action During 1st Precharge Stage	
	External (selected analog I/O pin)	Internal (AD sampling capacitor)
1	Connected to VDD	C_{HOLD} connected to VSS
0	Connected to VSS	C_{HOLD} connected to VDD

Otherwise:
The bit is ignored

bit 6 **IPEN:** A/D Inverted Precharge Enable bit
If DSEN = 1
1 = The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle
0 = Both Conversion cycles use the precharge and guards specified by PPOL and GPOL
Otherwise:
The bit is ignored

bit 5 **GPOL:** Guard Ring Polarity Selection bit
1 = ADC guard Ring outputs start as digital high during Precharge stage
0 = ADC guard Ring outputs start as digital low during Precharge stage

bit 4-1 **Unimplemented:** Read as ‘0’

bit 0 **DSEN:** Double-sample enable bit
1 = Two conversions are performed on each trigger. Data from the first conversion appears in PREV
0 = One conversion is performed for each trigger

REGISTER 36-3: ADCON2: ADC CONTROL REGISTER 2

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W/HC-0	R/W-0/0	R/W-0/0	R/W-0/0
PSIS	CRS[2:0]			ACLR	MD[2:0]		
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	HC = Bit is cleared by hardware

bit 7 **PSIS:** ADC Previous Sample Input Select bits
`1` = PREV is the FLTR value at start-of-conversion
`0` = PREV is the RES value at start-of-conversion

bit 6-4 **CRS[2:0]:** ADC Accumulated Calculation Right Shift Select bits
If MD = `100`:
Low-pass filter time constant is 2^{CRS}, filter gain is 1:1
If MD = `001`, `010` or `011`:
The accumulated value is right-shifted by CRS (divided by 2^{CRS})**(1,2)**
Otherwise:
Bits are ignored

bit 3 **ACLR:** A/D Accumulator Clear Command bit**(3)**
`1` = ACC, ADAOV and CNT registers are cleared
`0` = Clearing action is complete (or not started)

bit 2-0 **MD[2:0]:** ADC Operating Mode Selection bits**(4)**
`111-101` = Reserved
`100` = Low-pass Filter mode
`011` = Burst Average mode
`010` = Average mode
`001` = Accumulate mode
`000` = Basic mode

Note 1: To correctly calculate an average, the number of samples (set in RPT) must be 2^{CRS}.
2: CRS = `0b111` is a reserved option.
3: This bit is cleared by hardware when the accumulator operation is complete; depending on oscillator selections, the delay may be many instructions.
4: See Table 36-2 for Full mode descriptions.

REGISTER 36-4: ADCON3: ADC CONTROL REGISTER 3

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W/HC-0	R/W-0/0	R/W-0/0	R/W-0/0
—	CALC[2:0]			SOI	TMD[2:0]		
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	HC = Bit is cleared by hardware

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **CALC[2:0]:** ADC Error Calculation Mode Select bits

CALC	DSEN = 0 Single-Sample Mode	DSEN = 1 CVD Double-Sample Mode(1)	Application
111	Reserved	Reserved	Reserved
110	Reserved	Reserved	Reserved
101	FLTR-STPT	FLTR-STPT	Average/filtered value vs. setpoint
100	PREV-FLTR	PREV-FLTR	First derivative of filtered value(3) (negative)
011	Reserved	Reserved	Reserved
010	RES-FLTR	(RES-PREV)-FLTR	Actual result vs. averaged/ filtered value
001	RES-STPT	(RES-PREV)-STPT	Actual result vs.setpoint
000	RES-PREV	RES-PREV	First derivative of single measurement(2)
			Actual CVD result in CVD mode(2)

bit 3 **SOI:** ADC Stop-on-Interrupt bit

If CONT = 1:

1 = GO is cleared when the threshold conditions are met, otherwise the conversion is retriggered

0 = GO is not cleared by hardware, must be cleared by software to stop retriggers

bit 2-0 **TMD[2:0]:** Threshold Interrupt Mode Select bits

111 = Interrupt regardless of threshold test results

110 = Interrupt if ERR>UTH

101 = Interrupt if ERR≤UTH

100 = Interrupt if ERR<LTH or ERR>UTH

011 = Interrupt if ERR>LTH and ERR<UTH

010 = Interrupt if ERR≥LTH

001 = Interrupt if ERR<LTH

000 = Never interrupt

Note 1: When PSIS = 0, the value of (RES-PREV) is the value of (S2-S1) from Table 36-2.

2: When PSIS = 0

3: When PSIS = 1.

REGISTER 36-5: ADSTAT: ADC STATUS REGISTER

R-0/0	R-0/0	R-0/0	R/W/HC-0/0	U-0	R-0/0	R-0/0	R-0/0
ADAOV	UTHR	LTHR	MATH	—	STAT[2:0]		
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	HS/HC = Bit is set/cleared by hardware

bit 7 **ADAOV**: ADC Accumulator Overflow bit
`1` = ADACC or ADFLTR or ADERR registers have overflowed
`0` = ADACC, ADFLTR and ADERR registers have not overflowed

bit 6 **UTHR**: ADC Module Greater-than Upper Threshold Flag bit
`1` = ERR >UTH
`0` = ERR ≤ UTH

bit 5 **LTHR**: ADC Module Less-than Lower Threshold Flag bit
`1` = ERR < LTH
`0` = ERR ≥ LTH

bit 4 **MATH**: ADC Module Computation Status bit(1)
`1` = Registers ADACC, ADFLTR, ADUTH, ADLTH and the ADAOV bit are updating or have already updated
`0` = Associated registers/bits have not changed since this bit was last cleared

bit 3 **Unimplemented:** Read as ‘`0`’

bit 2-0 **STAT[2:0]**: ADC Module Cycle Multistage Status bits
`111` = ADC module is in 2nd conversion stage
`110` = ADC module is in 2nd acquisition stage
`101` = ADC module is in 2nd precharge stage
`100` = ADC computation is suspended between 1st and 2nd sample; the computation results are incomplete and awaiting data from the 2nd sample(2, 3)
`011` = ADC module is in 1st conversion stage
`010` = ADC module is in 1st acquisition stage
`001` = ADC module is in 1st precharge stage
`000` = ADC module is not converting

Note 1: MATH bit cannot be cleared by software while STAT = `0b100`.
2: If the selected clock is ADCRC and FOSC < ADCRC, this reading may be invalid.
3: STAT = 0b`100` appears between the two triggers when DSEN = `1` and CONT = `0`.

REGISTER 36-6: ADCLK: ADC CLOCK SELECTION REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	CS[5:0]					
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 **Unimplemented**: Read as ‘0’

bit 5-0 **CS[5:0]**: ADC Clock Divider Select bits
111111 = FOSC/128
111110 = FOSC/126
111101 = FOSC/124
•
•
•
000000 = FOSC/2

Note: ADC clock divider is only available if FOSC is selected as the ADC clock source (ADCON0.CS = 0).

REGISTER 36-7: ADREF: ADC REFERENCE SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	NREF	—	—	PREF[1:0]	
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 7-5 **Unimplemented**: Read as ‘0’

bit 4 **NREF**: ADC Negative Voltage Reference Selection bit
1 = VREF- is connected to external VREF-
0 = VREF- is connected to VSS

bit 3-2 **Unimplemented**: Read as ‘0’

bit 1-0 **PREF**: ADC Positive Voltage Reference Selection bits
11 = VREF+ is connected to internal Fixed Voltage Reference (FVR) module
10 = VREF+ is connected to external VREF+
01 = Reserved
00 = VREF+ is connected to VDD

REGISTER 36-8: ADPCH: ADC POSITIVE CHANNEL SELECTION REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	PCH[5:0]					
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 **Unimplemented**: Read as ‘0’

bit 5-0 **PCH[5:0]**: ADC Positive Input Channel Selection bits

111111 = FVR buffer 2(2)
111110 = FVR buffer 1(2)
111101 = DAC1 output(1)
111100 = Temperature indicator(3)
111011 = Vss (Analog Ground)
111010 = Reserved. No channel connected.
•
•
•
110000 = Reserved. No channel connected.
101111 = ANF7(4)
101110 = ANF6(4)
101101 = ANF5(4)
101100 = ANF4(4)
101011 = ANF3(4)
101010 = ANF2(4)
101001 = ANF1(4)
101000 = ANF0(4)
100111 = Reserved. No channel connected.
•
•
100011 = Reserved. No channel connected.
100010 = ANE2(5)
100001 = ANE1(5)
100000 = ANE0(5)
011111 = AND7(5)
011110 = AND6(5)
011101 = AND5(5)
011100 = AND4(5)
011011 = AND3(5)
011010 = AND2(5)
011001 = AND1(5)
011000 = AND0(5)
010111 = ANC7
010110 = ANC6
010101 = ANC5
010100 = ANC4
010011 = ANC3
010010 = ANC2
010001 = ANC1
010000 = ANC0
001111 = ANB7
001110 = ANB6
001101 = ANB5
001100 = ANB4
001011 = ANB3
001010 = ANB2
001001 = ANB1
001000 = ANB0
000111 = ANA7
000110 = ANA6
000101 = ANA5
000100 = ANA4
000011 = ANA3
000010 = ANA2
000001 = ANA1
000000 = ANA0

Note 1: See **Section 37.0 “5-Bit Digital-to-Analog Converter (DAC) Module”** for more information.
2: See **Section 34.0 “Fixed Voltage Reference (FVR)”** for more information.
3: See **Section 35.0 “Temperature Indicator Module”** for more information.
4: Reserved on PIC18(L)F26/27/45/46/47K42 parts.
5: Reserved on PIC18(L)F26K42 parts.

REGISTER 36-9: ADPREL: ADC PRECHARGE TIME CONTROL REGISTER (LOW BYTE)

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
PRE[7:0]							
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 7-0 **PRE[7:0]**: Precharge Time Select bits
See Table 36-4.

REGISTER 36-10: ADPREH: ADC PRECHARGE TIME CONTROL REGISTER (HIGH BYTE)

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	PRE[12:8]				
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 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 PRE[12:8]: Precharge Time Select bits
See Table 36-4.

TABLE 36-4: PRECHARGE TIME

ADPRE	Precharge time	
	CS! = ADCRC	CS = ADCRC
1 1111 1111 1111	8191 clocks of FOSC	8191 clocks of ADCRC
1 1111 1111 1110	8190 clocks of FOSC	8190 clocks of ADCRC
1 1111 1111 1101	8189 clocks of FOSC	8189 clocks of ADCRC
...	...	...
0 0000 0000 0010	2 clocks of FOSC	2 clocks of ADCRC
0 0000 0000 0001	1 clock of FOSC	1 clock of ADCRC
0 0000 0000 0000	Not included in the data conversion cycle	

REGISTER 36-11: ADACQL: ADC ACQUISITION TIME CONTROL REGISTER (LOW BYTE)

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ACQ[7:0]							
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 7-0 ACQ[7:0]: Acquisition (charge share time) Select bits
See Table 36-5.

REGISTER 36-12: ADACQH: ADC ACQUISITION TIME CONTROL REGISTER (HIGH BYTE)

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ACQ[12:8]				
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 7-5 **Unimplemented**: Read as ‘0’
bit 4-0 ACQ[12:8]: Acquisition (charge share time) Select bits
See Table 36-5.

TABLE 36-5: ACQUISITION TIME

ADACQ	Acquisition time	
	ADCS! = ADCRC	ADCS = ADCRC
1 1111 1111 1111	8191 clocks of FOSC	8191 clocks of ADCRC
1 1111 1111 1110	8190 clocks of FOSC	8190 clocks of ADCRC
1 1111 1111 1101	8189 clocks of FOSC	8189 clocks of ADCRC
...	...	...
0 0000 0000 0010	2 clocks of FOSC	2 clocks of ADCRC
0 0000 0000 0001	1 clock of FOSC	1 clock of ADCRC
0 0000 0000 0000	Not included in the data conversion cycle(1)	

Note 1: If ADPRE is not equal to ‘0’, then ADACQ = 0 means Acquisition time is 8192 clocks of FOSC or ADCRC.

REGISTER 36-13: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	CAP[4:0]				
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 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 **CAP[4:0]**: ADC Additional Sample Capacitor Selection bits
11111 = 31 pF
11110 = 30 pF
11101 = 29 pF
•
•
•
00011 = 3 pF
00010 = 2 pF
00001 = 1 pF
00000 = No additional capacitance

REGISTER 36-14: ADRPT: ADC REPEAT SETTING REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
RPT[7:0]							
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 7-0 **RPT[7:0]**: ADC Repeat Threshold bits
Determines the number of times that the ADC is triggered before the threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See Table 36-2 for more details.

REGISTER 36-15: ADCNT: ADC REPEAT COUNTER REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
CNT[7:0]							
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 7-0 **CNT[7:0]**: ADC Repeat Count bits
Counts the number of times that the ADC has been triggered and is used along with CNT to determine when the error threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See Table Table 36-2 for more details.

REGISTER 36-16: ADFLTRH: ADC FILTER HIGH BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
FLTR[15:8]							
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 7-0 **FLTR[15:8]**: ADC Filter Output Most Significant bits — Signed 2’s Complement
In Accumulate, Average, and Burst Average mode, this is equal to ACC right shifted by the CRS bits of ADCON2. In LPF mode, this is the output of the Low-pass Filter.

REGISTER 36-17: ADFLTRL: ADC FILTER LOW BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
FLTR[7:0]							
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 7-0 **FLTR[7:0]**: ADC Filter Output Least Significant bits — Signed 2’s Complement
In Accumulate, Average, and Burst Average mode, this is equal to ACC right shifted by the CRS bits of ADCON2. In LPF mode, this is the output of the Low-pass Filter.

REGISTER 36-18: ADRESH: ADC RESULT REGISTER HIGH, FM = 0

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
RES[11:4]							
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 7-0 **RES[11:4]**: ADC Result Register bits
Upper eight bits of 12-bit conversion result.

REGISTER 36-19: ADRESL: ADC RESULT REGISTER LOW, FM = 0

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0
RES[3:0]				—	—	—	—
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 7-4 **RES[3:0]**: ADC Result Register bits. Lower four bits of 12-bit conversion result.
bit 3-0 **Reserved**

REGISTER 36-20: ADRESH: ADC RESULT REGISTER HIGH, FM = 1

U-0	U-0	U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	—	—	RES[11:8]			
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 7-4 **Reserved**

bit 3-0 **RES[11:8]**: ADC Sample Result bits. Upper four bits of 12-bit conversion result.

REGISTER 36-21: ADRESL: ADC RESULT REGISTER LOW, FM = 1

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
RES[7:0]							
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 7-0 **RES[7:0]**: ADC Result Register bits. Lower eight bits of 12-bit conversion result.

REGISTER 36-22: ADPREVH: ADC PREVIOUS RESULT REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
PREV[15:8]							
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 7-0 **PREV[15:8]**: Previous ADC Results bits
If PSIS = 1:
Upper byte of FLTR at the start of current ADC conversion
If PSIS = 0:
Upper bits of ADRES at the start of current ADC conversion(1)

Note 1: If PSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the FM bit.

REGISTER 36-23: ADPREVL: ADC PREVIOUS RESULT REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
PREV[7:0]							
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 7-0 **PREV[7:0]**: Previous ADC Results bits
If PSIS = 1:
Lower byte of FLTR at the start of current ADC conversion
If PSIS = 0:
Lower bits of ADRES at the start of current ADC conversion(1)

Note 1: If PSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the FM bit.

REGISTER 36-24: ADACCU: ADC ACCUMULATOR REGISTER UPPER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
(sign)	(sign)	(sign)	(sign)	(sign)	(sign)	ACC[17:16]	
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 7-2 Six copies of sign bit (1)

bit 1-0 **ACC[17:16]**: ADC Accumulator MSB — Signed 2’s Complement. Upper two bits of accumulator value. See Table 36-2 for more details.

Note 1: The ADACC register is a 24-bit wide register which contains the 18-bit accumulator value and six copies of the sign bit.

2: This register can only be written when GO=0.

REGISTER 36-25: ADACCH: ADC ACCUMULATOR REGISTER HIGH

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ACC[15:8]							
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 7-0 **ACC[15:8]**: ADC Accumulator middle bits — Signed 2’s Complement. Middle eight bits of accumulator value. See Table 36-2 for more details.

Note 1: The ADACC register is a 24-bit wide register which contains the 18-bit accumulator value and six copies of the sign bit.

2: This register can only be written when GO=0.

REGISTER 36-26: ADACCL: ADC ACCUMULATOR REGISTER LOW

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ACC[7:0]							
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 7-0 **ACC[7:0]**: ADC Accumulator LSB — Signed 2’s Complement. Lower eight bits of accumulator value. See Table 36-2 for more details.

Note 1: The ADACC register is a 24-bit wide register which contains the 18-bit accumulator value and six copies of the sign bit.

2: This register can only be written when GO=0.

REGISTER 36-27: ADSTPTH: ADC THRESHOLD SETPOINT REGISTER HIGH

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
STPT[15:8]							
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 7-0 **STPT[15:8]**: ADC Threshold Setpoint MSB — Signed 2’s Complement. Upper byte of ADC threshold setpoint, depending on CALC, may be used to determine ERR, see Register 36-29 for more details.

REGISTER 36-28: ADSTPTL: ADC THRESHOLD SETPOINT REGISTER LOW

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
STPT[7:0]							
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 7-0 **STPT[7:0]**: ADC Threshold Setpoint LSB — Signed 2’s Complement. Lower byte of ADC threshold setpoint, depending on CALC, may be used to determine ERR, see Register 36-30 for more details.

REGISTER 36-29: ADERRH: ADC SETPOINT ERROR REGISTER HIGH

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ERR[15:8]							
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 7-0 **ERR[15:8]**: ADC Setpoint Error MSB — Signed 2’s Complement. Upper byte of ADC Setpoint Error. Setpoint Error calculation is determined by CALC bits of ADCON3, see Register 36-4 for more details.

REGISTER 36-30: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ERR[7:0]							
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 7-0 **ERR[7:0]**: ADC Setpoint Error LSB — Signed 2’s Complement. Lower byte of ADC Setpoint Error calculation is determined by CALC bits of ADCON3, see Register 36-4 for more details.

REGISTER 36-31: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LTH[15:8]							
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 7-0 **LTH[15:8]**: ADC Lower Threshold MSB — Signed 2’s Complement. LTH and UTH are compared with ERR to set the UTHR and LTHR bits of ADSTAT. Depending on the setting of TMD, an interrupt may be triggered by the results of this comparison.

REGISTER 36-32: ADLTHL: ADC LOWER THRESHOLD LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LTH[7:0]							
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 7-0 **LTH[7:0]**: ADC Lower Threshold LSB — Signed 2’s Complement. LTH and UTH are compared with ERR to set the UTHR and LTHR bits of ADSTAT. Depending on the setting of TMD, an interrupt may be triggered by the results of this comparison.

REGISTER 36-33: ADUTHH: ADC UPPER THRESHOLD HIGH BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
UTH[15:8]							
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 7-0 **UTH[15:8]**: ADC Upper Threshold MSB — Signed 2’s Complement. LTH and UTH are compared with ERR to set the UTHR and LTHR bits of ADSTAT. Depending on the setting of TMD, an interrupt may be triggered by the results of this comparison.

REGISTER 36-34: ADUTHL: ADC UPPER THRESHOLD LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
UTH[7:0]							
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 7-0 **UTH[7:0]**: ADC Upper Threshold LSB — Signed 2’s Complement. LTH and UTH are compared with ERR to set the UTHR and LTHR bits of ADSTAT. Depending on the setting of TMD, an interrupt may be triggered by the results of this comparison.

REGISTER 36-35: ADACT: ADC AUTO CONVERSION TRIGGER CONTROL REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ACT[4:0]				
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 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 **ACT[4:0]**: Auto-Conversion Trigger Select Bits
11111 = Reserved, do not use
•
•
•
11110 = Reserved, do not use
11101 = Software write to ADPCH
11100 = Reserved, do not use
11011 = Software read of ADRESH
11010 = Software read of ADERRH
11001 = CLC4_out
11000 = CLC3_out
10111 = CLC2_out
10110 = CLC1_out
10101 = Logical OR of all Interrupt-on-change Interrupt Flags
10100 = CMP2_out
10011 = CMP1_out
10010 = NCO1_out
10001 = PWM8_out
10000 = PWM7_out
01111 = PWM6_out
01110 = PWM5_out
01101 = CCP4_trigger
01100 = CCP3_trigger
01011 = CCP2_trigger
01010 = CCP1_trigger
01001 = SMT1_trigger
01000 = TMR6_postscaled
00111 = TMR5_overflow
00110 = TMR4_postscaled
00101 = TMR3_overflow
00100 = TMR2_postscaled
00011 = TMR1_overflow
00010 = TMR0_overflow
00001 = Pin selected by ADACTPPS
00000 = External Trigger Disabled

REGISTER 36-36: ADCP: ADC CHARGE PUMP CONTROL REGISTER

R/W-0/0	U-0	U-0	U-0	U-0	U-0	U-0	R-0/0
CPON	—	—	—	—	—	—	CPRDY
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	HS= Hardware set

bit 7 **CPON**: Charge Pump On Control bit
1 = Charge Pump On when requested by the ADC
0 = Charge Pump Off

bit 6-1 **Unimplemented:** Read as ‘0’

bit 0 CPRDY: Charge Pump Ready Status bit
1 = Charge Pump is ready
0 = Charge Pump is not ready (or never started)

TABLE 36-6: SUMMARY OF REGISTERS ASSOCIATED WITH ADC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ADCON0	ON	CONT	—	CS	—	FM	—	GO	619
ADCON1	PPOL	IPEN	GPOL	—	—	—	—	DSEN	620
ADCON2	PSIS	CRS[2:0]			ACLR	MD[2:0]			621
ADCON3	—	CALC[2:0]			SOI	TMD[2:0]			622
ADSTAT	ADAOV	UTHR	LTHR	MATH	STAT[3:0]				623
ADCLK	—	—	CS[5:0]						624
ADREF	—	—	—	NREF	—	—	PREF[1:0]		624
ADPCH	—	—	PCH[5:0]						625
ADPREL	PRE[7:0]								626
ADPREH	—	—	—	PRE[12:8]					626
ADACQL	ACQ[7:0]								627
ADACQH	—	—	—	ACQ[12:8]					624
ADCAP	—	—	—	CAP[4:0]					628
ADRPT	RPT[7:0]								628
ADCNT	CNT[7:0]								629
ADFLTRL	FLTR[7:0]								629
ADFLTRH	FLTR[15:8]								629
ADRESL	RESL[7:0]								630, 631
ADRESH	RESH[7:0]								630, 631
ADPREVH	PREV[15:8]								632
ADPREVL	PREV[7:0]								632
ADACCH	ACC[15:8]								633
ADACCL	ACC[7:0]								634
ADACCU	(sign)	(sign)	(sign)	(sign)	(sign)	(sign)	ACC[17:16]		633
ADSTPTL	STPT[7:0]								634
ADSTPTH	STPT[15:8]								634

TABLE 36-6: SUMMARY OF REGISTERS ASSOCIATED WITH ADC (CONTINUED)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ADERRL	ERR[7:0]								635
ADERRH	ERR[15:8]								635
ADLTHH	LTH[15:8]								635
ADLTHL	LTH[7:0]								636
ADUTHH	UTH[15:8]								636
ADUTHL	UTH[7:0]								636
ADERRL	ERR[15:8]								635
ADACT	—	—	—	ACT[5:0]					637
ADCP	CPON	—	—	—	—	—	—	CPRDY	638

Legend: — = unimplemented read as '0'. Shaded cells are not used for the ADC module.

37.0 5-BIT DIGITAL-TO-ANALOG CONVERTER (DAC) MODULE

The Digital-to-Analog Converter supplies a variable voltage reference, ratiometric with the input source, with 32 selectable output levels.

The positive input source (VSOURCE+) of the DAC can be connected to:

- FVR Buffer
- External VREF+ pin
- VDD supply voltage

The negative input source (VSOURCE-) of the DAC can be connected to:

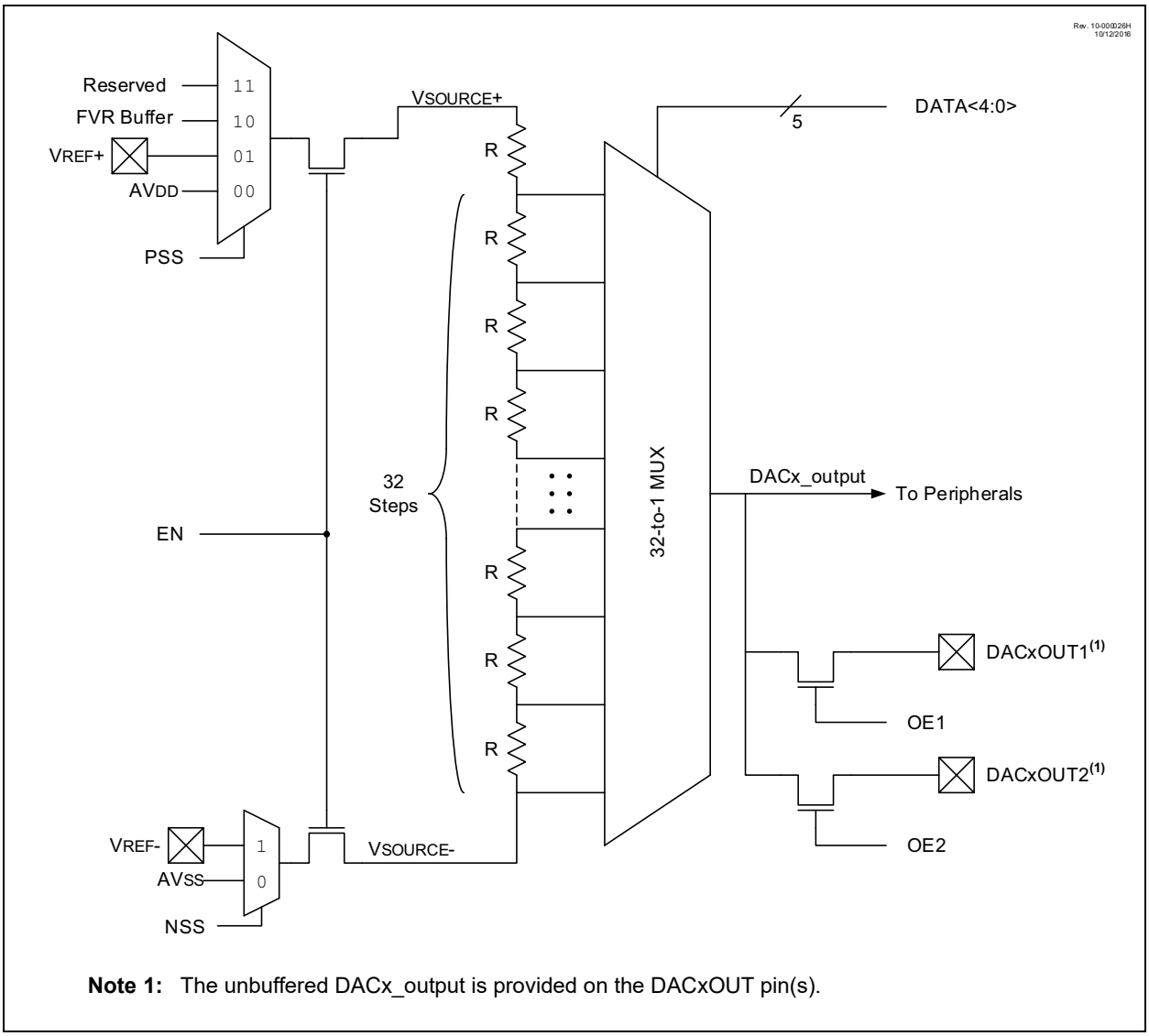
- External VREF- pin
- Vss

The output of the DAC (DAC1_output) can be selected as a reference voltage to the following:

- Comparator positive input
- ADC input channel
- DAC1OUT1 pin
- DAC1OUT2 pin

The Digital-to-Analog Converter (DAC) can be enabled by setting the EN bit of the DAC1CON0 register.

FIGURE 37-1: DIGITAL-TO-ANALOG CONVERTER BLOCK DIAGRAM



Note 1: The unbuffered DACx_output is provided on the DACxOUT pin(s).

37.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DATA[4:0] bits of the DAC1CON1 register.

The DAC output voltage can be determined by using Equation 37-1.

37.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in Table 44-17.

37.3 DAC Voltage Reference Output

The unbuffered DAC voltage can be output to the DAC1OUTn pin(s) by setting the respective DACOEn bit(s) of the DAC1CON0 register. Selecting the DAC reference voltage for output on either DAC1OUTn pin automatically overrides the digital output buffer, the weak pull-up and digital input threshold detector functions of that pin.

Reading the DAC1OUTn pin when it has been configured for DAC reference voltage output will always return a '0'.

Note: The unbuffered DAC output (DAC1OUTn) is not intended to drive an external load.

37.4 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Windowed Watchdog Timer Time-out, the contents of the DAC1CON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

37.5 Effects of a Reset

A device Reset affects the following:

- DAC1 is disabled.
- DAC1 output voltage is removed from the DAC1OUTn pin(s).
- The DAC1R[4:0] range select bits are cleared.

EQUATION 37-1: DAC OUTPUT VOLTAGE

IF DACEN = 1

$$DACx_output = \left((V_{REF+} - V_{REF-}) \times \frac{DATA[4:0]}{2^5} \right) + V_{REF-}$$

Note: See the DAC1CON0 register for the available $V_{SOURCE+}$ and $V_{SOURCE-}$ selections.

37.6 Register Definitions: DAC Control

Long bit name prefixes for the DAC peripheral is shown below. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

Peripheral	Bit Name Prefix
DAC1	DAC1

REGISTER 37-1: DAC1CON0: DAC CONTROL REGISTER

R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0
EN	—	OE1	OE2	PSS[1:0]		—	NSS
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 **EN:** DAC Enable bit
1 = DAC is enabled
0 = DAC is disabled(1)

bit 6 **Unimplemented:** Read as '0'

bit 5 **OE1:** DAC Voltage Output Enable bit
1 = DAC voltage level is output on the DAC1OUT1 pin
0 = DAC voltage level is disconnected from the DAC1OUT1 pin

bit 4 **OE2:** DAC Voltage Output Enable bit
1 = DAC voltage level is output on the DAC1OUT2 pin
0 = DAC voltage level is disconnected from the DAC1OUT2 pin

bit 3-2 **PSS[1:0]:** DAC Positive Source Select bit
11 = Reserved
10 = FVR buffer 2
01 = V_{REF+}
00 = V_{DD}

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

bit 0 **NSS:** DAC Negative Source Select bit
1 = V_{REF-}
0 = V_{SS}

Note 1: DAC1OUTx output pins are still active.

REGISTER 37-2: DAC1CON1: DAC DATA REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	DATA[4:0]				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **DATA[4:0]:** Data Input Register for DAC bits

TABLE 37-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
DAC1CON0	EN	—	OE1	OE2	PSS[1:0]		—	NSS	642
DAC1CON1	—	—	—	DATA[4:0]					643

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used with the DAC module.

38.0 COMPARATOR MODULE

Note: The PIC18(L)F26/27/45/46/47/55/56/57K42 devices have two comparators. Therefore, all information in this section refers to both C1 and C2.

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.

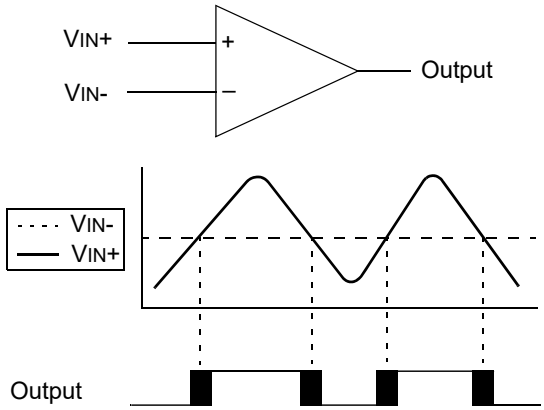
The analog comparator module includes the following features:

- Programmable input selection
- Programmable output polarity
- Rising/falling output edge interrupts

38.1 Comparator Overview

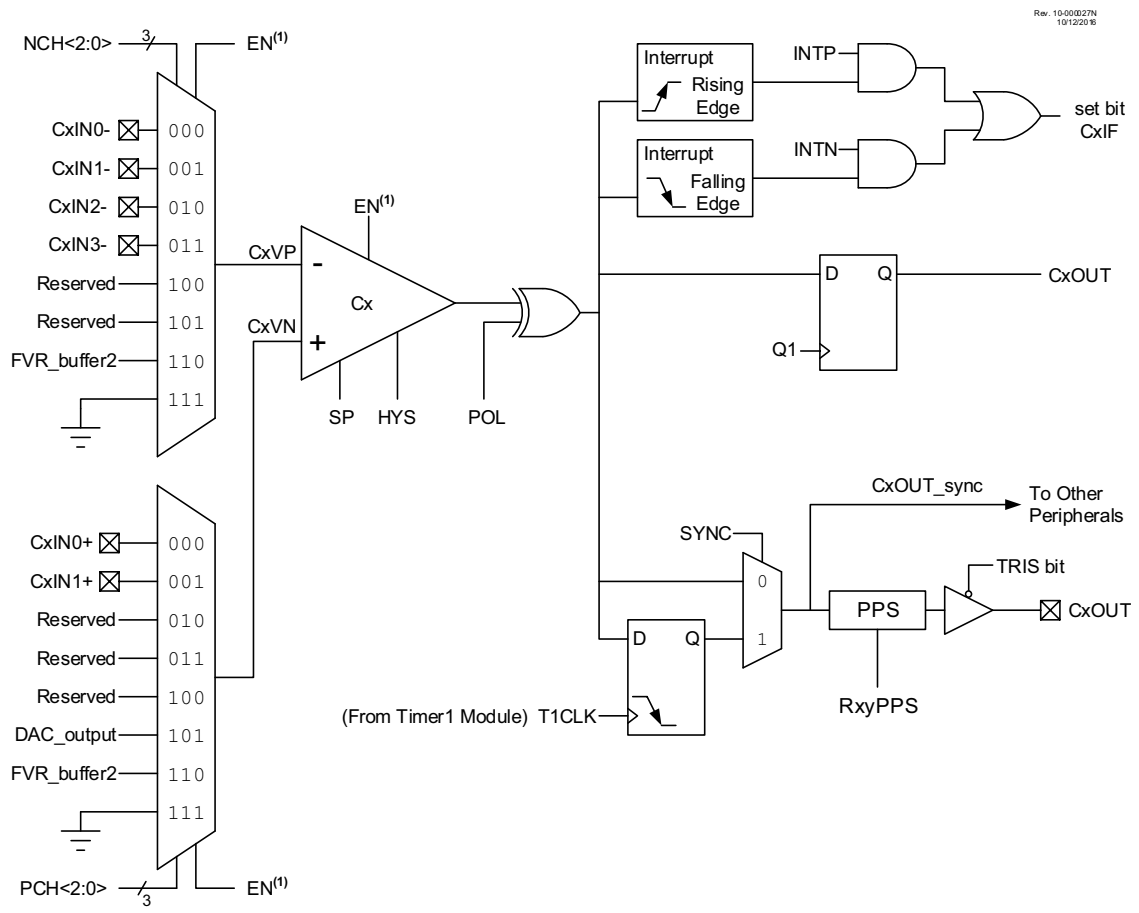
A single comparator is shown in Figure 38-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at VIN+ is less than the analog voltage at VIN-, the output of the comparator is a digital low level. When the analog voltage at VIN+ is greater than the analog voltage at VIN-, the output of the comparator is a digital high level.

FIGURE 38-1: SINGLE COMPARATOR



Note: The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.

FIGURE 38-2: 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.

38.2 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see Register 38-1) contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register (see Register 38-2) contains Control bits for the following:

- Interrupt on positive/negative edge enables

The CMxPCH and CMxNCH registers are used to select the positive and negative input channels, respectively.

38.2.1 COMPARATOR ENABLE

Setting the EN bit of the CMxCON0 register enables the comparator for operation. Clearing the EN bit disables the comparator resulting in minimum current consumption.

38.2.2 COMPARATOR OUTPUT

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the CxOUT bit of the CMOUT register.

The comparator output can also be routed to an external pin through the RxyPPS register (Register 17-2). The corresponding TRIS bit must be clear to enable the pin as an output.

> **Note 1:** The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

38.2.3 COMPARATOR OUTPUT POLARITY

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the POL bit of the CMxCON0 register. Clearing the POL bit results in a noninverted output.

Table 38-1 shows the output state versus input conditions, including polarity control.

TABLE 38-1: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS

Input Condition	POL	CxOUT
CxVN > CxVP	0	0
CxVN < CxVP	0	1
CxVN > CxVP	1	1
CxVN < CxVP	1	0

38.3 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 HYS bit of the CMxCON0 register.

See Comparator Specifications in Table 44-16 for more information.

38.3.1 COMPARATOR OUTPUT SYNCHRONIZATION

The output from a comparator can be synchronized with Timer1 by setting the SYNC 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, the CxOUT bit is synchronized with the timer, so that the software sees no ambiguity due to timing. See the Comparator Block Diagram (Figure 38-2) and the Timer1 Block Diagram (Figure 21-1) for more information.

38.4 Comparator Interrupt

An interrupt can be generated for every rising or falling edge of the comparator output.

When either edge detector is triggered and its associated enable bit is set (INTP and/or INTN bits of the CMxCON1 register), the Corresponding Interrupt Flag bit (CxIF bit of the respective PIR register) will be set.

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

- EN bit of the CMxCON0 register
- CxIE bit of the respective PIE register
- INTP bit of the CMxCON1 register (for a rising edge detection)
- INTN bit of the CMxCON1 register (for a falling edge detection)
- GIE bit of the INTCON0 register

The associated interrupt flag bit, CxIF bit of the respective PIR 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 POL bit of the CMxCON0 register, or by switching the comparator on or off with the EN bit of the CMxCON0 register.

38.5 Comparator Positive Input Selection

Configuring the PCH[2:0] bits of the CMxPCH register directs an internal voltage reference or an analog pin to the non-inverting input of the comparator:

- CxIN0+, CxIN1+ analog pin
- DAC output
- FVR (Fixed Voltage Reference)
- VSS (Ground)

See **Section 34.0 “Fixed Voltage Reference (FVR)”** for more information on the Fixed Voltage Reference module.

See **Section 37.0 “5-Bit Digital-to-Analog Converter (DAC) Module”** for more information on the DAC input signal.

Any time the comparator is disabled (EN = 0), all comparator inputs are disabled.

38.6 Comparator Negative Input Selection

The NCH[2:0] bits of the CMxNCH register direct an analog input pin and internal reference voltage or analog ground to the inverting input of the comparator:

- CxIN0-, CxIN1-, CxIN2-, CxIN3- analog pin
- FVR (Fixed Voltage Reference)
- Analog Ground

Note:	To use CxINy+ and CxINy- pins as analog input, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

38.7 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 Table 44-16 and Table 44-18 for more details.

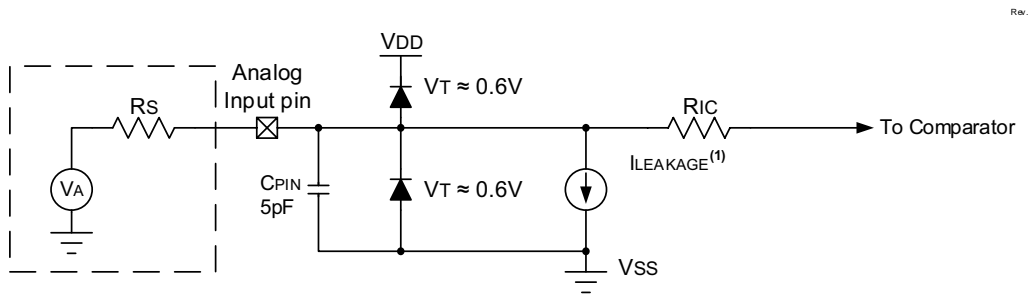
38.8 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 38-3. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

The maximum source impedance for analog sources is mentioned in Parameter AD08 in Table 44-14. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

> **Note 1:** When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.
>
> **2:** Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

FIGURE 38-3: ANALOG INPUT MODEL



Legend:
CPIN = Input Capacitance
ILEAKAGE = Leakage Current at the pin due to various junctions
RIC = Interconnect Resistance
RS = Source Impedance
VA = Analog Voltage
VT = Threshold Voltage

Note 1: See **Section 44.0 "Electrical Specifications"**.

38.9 CWG1 Auto-Shutdown Source

The output of the comparator module can be used as an auto-shutdown source for the CWG1 module. When the output of the comparator is active and the corresponding WGASxE is enabled, the CWG operation will be suspended immediately (see **Section 26.10.1.2 "External Input Source"**).

38.10 ADC Auto-Trigger Source

The output of the comparator module can be used to trigger an ADC conversion. When the ADACT register is set to trigger on a comparator output, an ADC conversion will trigger when the Comparator output goes high.

38.11 TMR2/4/6 Reset

The output of the comparator module can be used to reset Timer2. When the TxRST register is appropriately set, the timer will reset when the Comparator output goes high.

38.12 Operation in Sleep Mode

The comparator module can operate during Sleep. The comparator clock source is based on the Timer1 clock source. If the Timer1 clock source is either the system clock (FOSC) or the instruction clock (FOSC/4), Timer1 will not operate during Sleep, and synchronized comparator outputs will not operate.

A comparator interrupt will wake the device from Sleep. The CxIE bits of the respective PIE register must be set to enable comparator interrupts.

38.13 Register Definitions: Comparator Control

Long bit name prefixes for the Comparators are shown in Table 38-2. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

TABLE 38-2:

Peripheral	Bit Name Prefix
C1	C1
C2	C2

REGISTER 38-1: CMxCON0: COMPARATOR x CONTROL REGISTER 0

R/W-0/0	R-0/0	U-0	R/W-0/0	U-0	U-1	R/W-0/0	R/W-0/0
EN	OUT	—	POL	—	—	HYS	SYNC
bit 7							bit 0

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

bit 7 **EN:** Comparator Enable bit
1 = Comparator is enabled
0 = Comparator is disabled and consumes no active power

bit 6 **OUT:** Comparator Output bit
If POL = 0 (noninverted polarity):
1 = CxVP > CxVN
0 = CxVP < CxVN
If POL = 1 (inverted polarity):
1 = CxVP < CxVN
0 = CxVP > CxVN

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

bit 4 **POL:** Comparator Output Polarity Select bit
1 = Comparator output is inverted
0 = Comparator output is not inverted

bit 3 **Unimplemented:** Read as '0'

bit 2 **Unimplemented:** Read as '1'

bit 1 **HYS:** Comparator Hysteresis Enable bit
1 = Comparator hysteresis enabled
0 = Comparator hysteresis disabled

bit 0 **SYNC:** Comparator Output Synchronous Mode bit
1 = Comparator output to Timer1 and I/O pin is synchronous to changes on Timer1 clock source. Output updated on the falling edge of Timer1 clock source.
0 = Comparator output to Timer1 and I/O pin is asynchronous

REGISTER 38-2: CMxCON1: COMPARATOR x CONTROL REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	INTP	INTN
bit 7							bit 0

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

bit 7-2 Unimplemented: Read as ‘0’

bit 1 **INTP**: Comparator Interrupt on Positive-Going Edge Enable bit
- 1 = The CxIF interrupt flag will be set upon a positive-going edge of the CxOUT bit
- 0 = No interrupt flag will be set on a positive-going edge of the CxOUT bit

bit 0 **INTN**: Comparator Interrupt on Negative-Going Edge Enable bit
- 1 = The CxIF interrupt flag will be set upon a negative-going edge of the CxOUT bit
- 0 = No interrupt flag will be set on a negative-going edge of the CxOUT bit

REGISTER 38-3: CMxNCH: COMPARATOR x INVERTING CHANNEL SELECT REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	NCH[2:0]		
bit 7							bit 0

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

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **NCH[2:0]:** Comparator Inverting Input Channel Select bits
- 111 = Vss
- 110 = FVR_Buffer2
- 101 = NCH not connected
- 100 = NCH not connected
- 011 = CxIN3-
- 010 = CxIN2-
- 001 = CxIN1-
- 000 = CxIN0-

REGISTER 38-4: CMxPCH: COMPARATOR x NON-INVERTING CHANNEL SELECT REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	PCH[2:0]		
bit 7							bit 0

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

bit 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **PCH[2:0]:** Comparator Non-Inverting Input Channel Select bits
111 = Vss
110 = FVR_Buffer2
101 = DAC_Output
100 = PCH not connected
011 = PCH not connected
010 = PCH not connected
001 = CxIN1+
000 = CxIN0+

REGISTER 38-5: CMOUT: COMPARATOR OUTPUT REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	R-0/0	R-0/0
—	—	—	—	—	—	C2OUT	C1OUT
bit 7							bit 0

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

bit 7-2 **Unimplemented:** Read as ‘0’

bit 1 **C2OUT:** Mirror copy of C2OUT bit

bit 0 **C1OUT:** Mirror copy of C1OUT bit

TABLE 38-3: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CMxCON0	EN	OUT	—	POL	—	—	HYS	SYNC	650
CMxCON1	—	—	—	—	—	—	INTP	INTN	651
CMxNCH	—	—	—	—	—	NCH[2:0]			651
CMxPCH	—	—	—	—	—	PCH[2:0]			652
CMOUT	—	—	—	—	—	—	C2OUT	C1OUT	652

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

39.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18(L)F26/27/45/46/47/55/56/57K42 family of devices has a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point (positive going, negative going or both). If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

Complete control of the HLVD module is provided through the HLVDCON0 and HLVDCON1 register. This allows the circuitry to be “turned off” by the user under software control, which minimizes the current consumption for the device.

The module’s block diagram is shown in Figure 39-1.

Since the HLVD can be software enabled through the EN bit, setting and clearing the enable bit does not produce a false HLVD event glitch. Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The RDY bit (HLVDCON0[4]) is a read-only bit used to indicate when the band gap reference voltages are stable.

The module can only generate an interrupt after the module is turned ON and the band gap reference voltages are ready.

The INTH and INTL bits determine the overall operation of the module. When INTH is set, the module monitors for rises in V_{DD} above the trip point set by the HLVDCON1 register. When INTL is set, the module monitors for drops in V_{DD} below the trip point set by the HLVDCON1 register. When both the INTH and INTL bits are set, any changes above or below the trip point set by the HLVDCON1 register can be monitored.

The OUT bit can be read to determine if the voltage is greater than or less than the voltage level selected by the HLVDCON1 register.

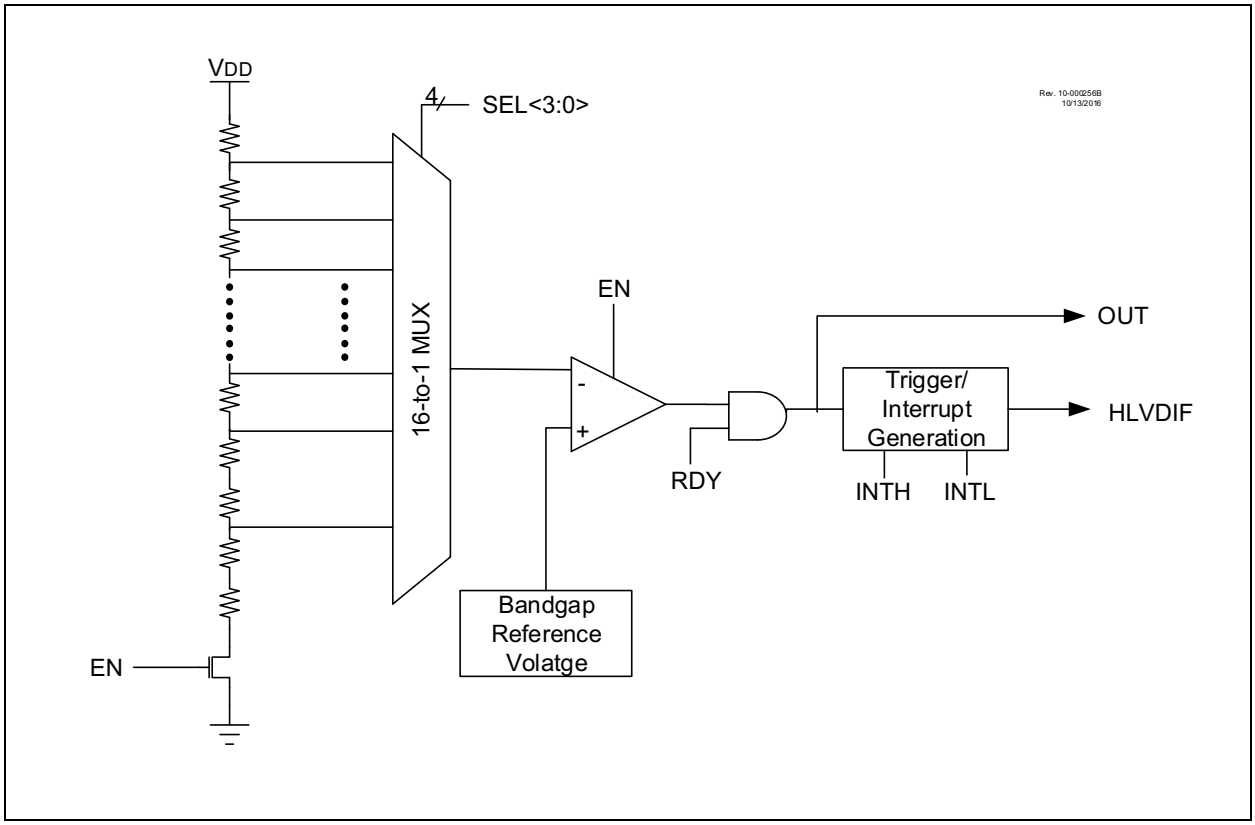
39.1 Operation

When the HLVD module is enabled, a comparator uses an internally generated voltage reference as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any of SEL[3:0] bits (HLVDCON1[3:0]).

FIGURE 39-1: HLVD MODULE BLOCK DIAGRAM



39.2 HLVD Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the SEL[3:0] bits of the HLVDCON1 register.
2. Depending on the application to detect high-voltage peaks or low-voltage drops or both, set the INTH or INTL bit appropriately.
3. Enable the HLVD module by setting the EN bit.
4. Clear the HLVD interrupt flag (PIR2 register), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE in the PIE2 register and GIE bits.

   An interrupt will not be generated until the RDY bit is set.

Note:	Before changing any module settings (INTH, INTL, SEL[3:0]), first disable the module (EN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events.

39.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and consume static current. The total current consumption, when enabled, is specified in electrical specification Parameter **D206** (Table 44-4).

Depending on the application, the HLVD module does not need to operate constantly. To reduce current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After such a check, the module could be disabled.

39.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification (Table 44-18), may be used by other internal circuitry, such as the programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, T_{FVRST}, is an interval that is independent of device clock speed. It is specified in electrical specification (Table 44-18).

The HLVD interrupt flag is not enabled until T_{FVRST} has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (see Figure 39-2 or Figure 39-3).

FIGURE 39-2: LOW-VOLTAGE DETECT OPERATION (INTL = 1)

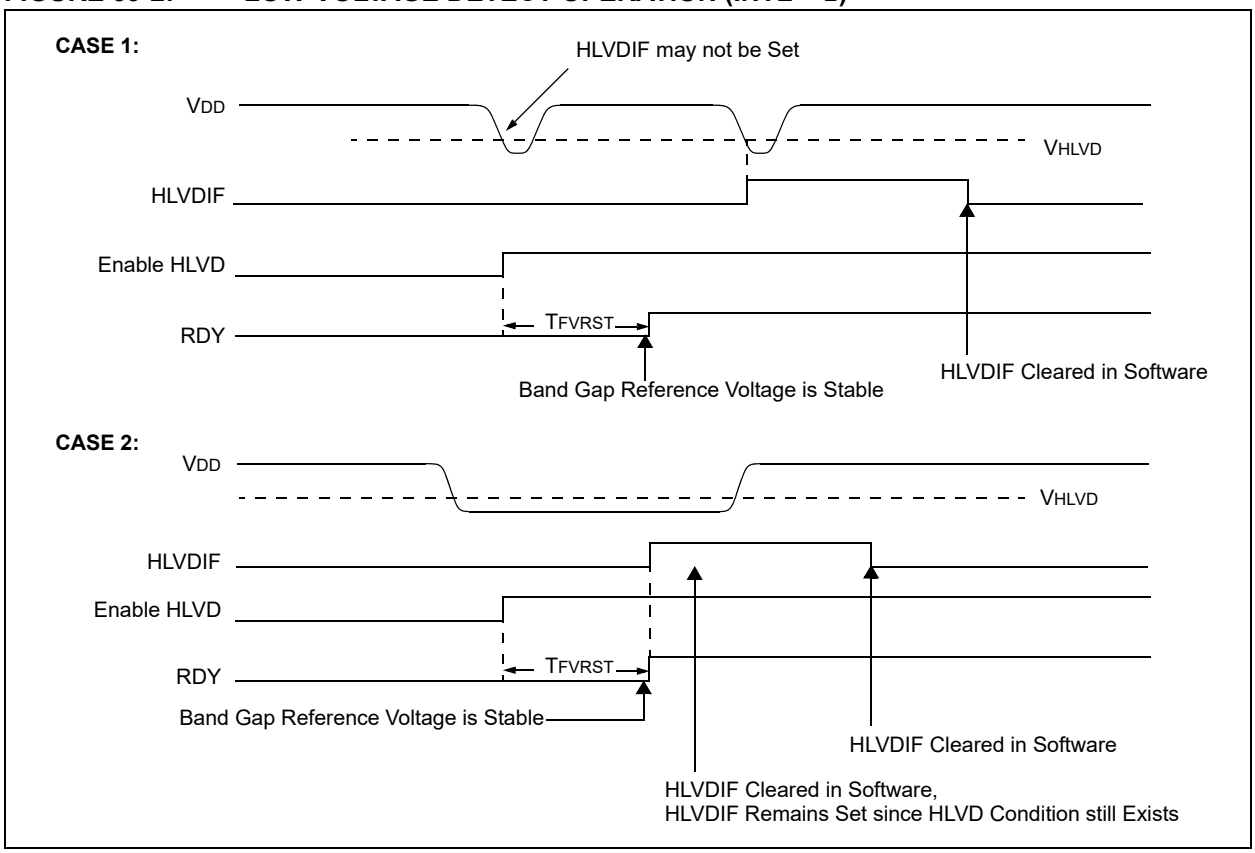
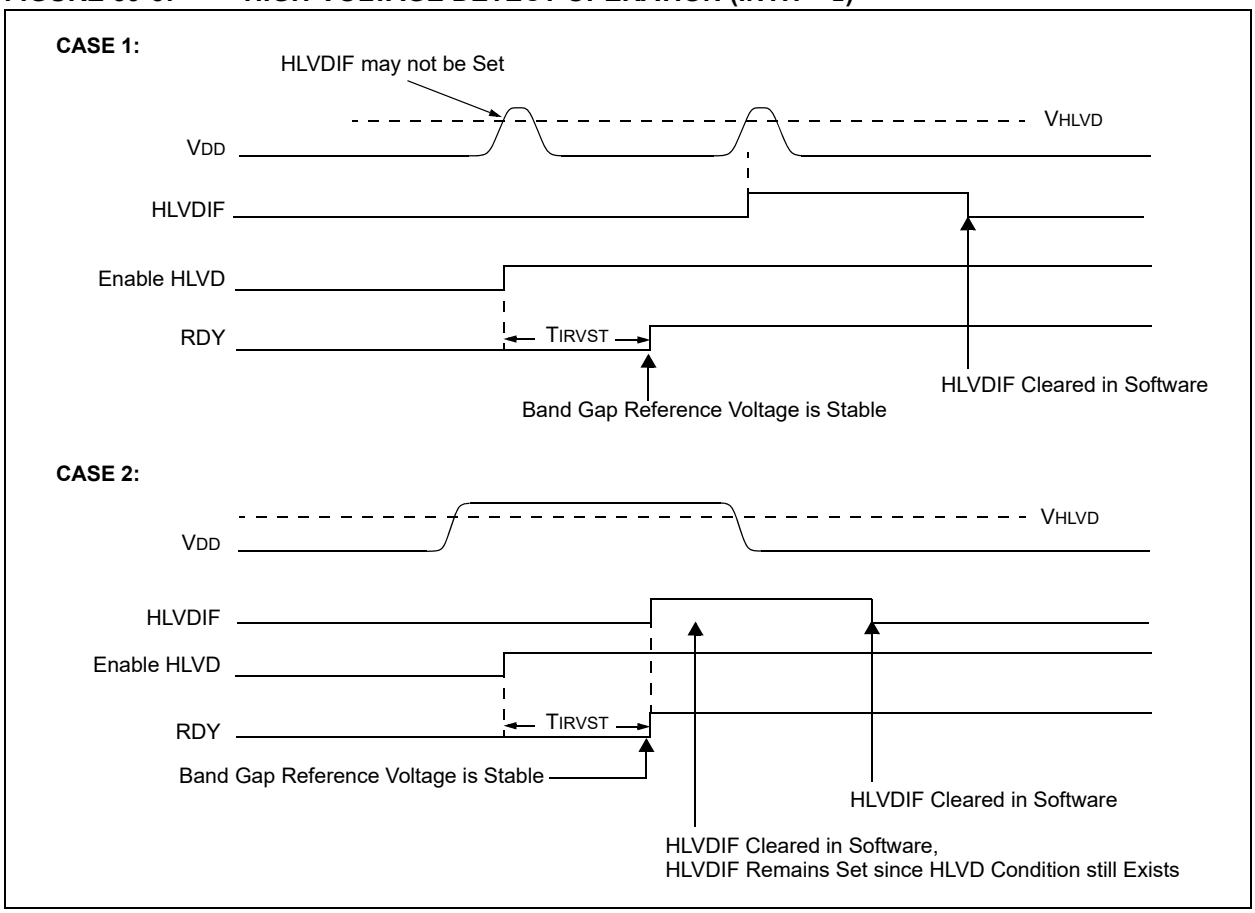


FIGURE 39-3: HIGH-VOLTAGE DETECT OPERATION (INTH = 1)

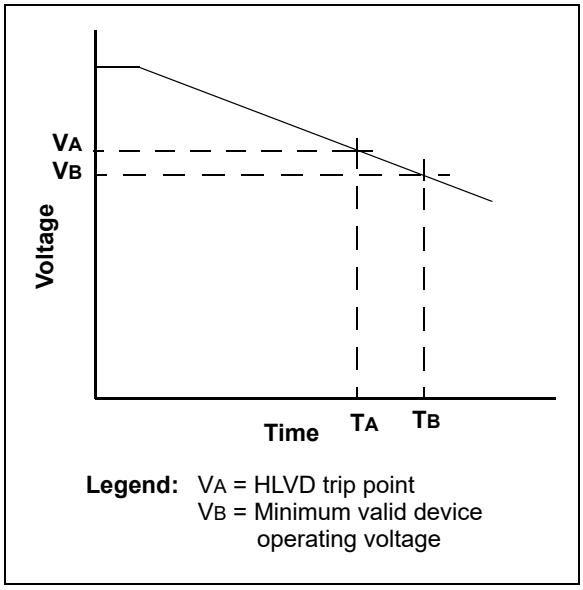


39.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a High-Voltage Detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 39-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, V_A, the HLVD logic generates an interrupt at time, T_A. The interrupt could cause the execution of an Interrupt Service Routine (ISR), which would allow the application to perform “housekeeping tasks” and a controlled shutdown before the device voltage exits the valid operating range at T_B. This would give the application a time window, represented by the difference between T_A and T_B, to safely exit.

FIGURE 39-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION



39.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

39.7 Operation During Idle and Doze Modes

In both Idle and Doze modes, the module is active and events are generated if peripheral is enabled.

39.8 Operation During Freeze

When debugging in Freeze mode, no new event or interrupt can be generated. The state of the RDY bit is frozen.

Register reads and writes through the CPU interface are allowed.

39.9 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

39.10 Register Definitions: HLVD Control

Long bit name prefixes for the HLVD peripheral is shown in Table 39-1. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

TABLE 39-1:

Peripheral	Bit Name Prefix
HLVD	HLVD

REGISTER 39-1: HLVDCON0: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER 0

R/W-0/0	U-0	R-x	R-x	U-0	U-0	R/W-0/0	R/W-0/0
EN	—	OUT	RDY	—	—	INTH	INTL
bit 7							bit 0

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

bit 7 **EN:** High/Low-voltage Detect Power Enable bit
- 1 = Enables HLVD, powers up HLVD circuit and supporting reference circuitry
- 0 = Disables HLVD, powers down HLVD and supporting circuitry

bit 6 **Unimplemented:** Read as '0'

bit 5 **OUT:** HLVD Comparator Output bit
- 1 = Voltage ≤ selected detection limit (HLVDL[3:0])
- 0 = Voltage ≥ selected detection limit (HLVDL[3:0])

bit 4 **RDY:** Band Gap Reference Voltages Stable Status Flag bit
- 1 = Indicates HLVD Module is ready and output is stable
- 0 = Indicates HLVD Module is not ready

bit 3-2 **Unimplemented:** Read as '0'

bit 1 **INTH:** HLVD Positive going (High Voltage) Interrupt Enable
- 1 = HLVDIF will be set when voltage ≥ selected detection limit (SEL[3:0])
- 0 = HLVDIF will not be set

bit 0 **INTL:** HLVD Negative going (Low Voltage) Interrupt Enable
- 1 = HLVDIF will be set when voltage ≤ selected detection limit (SEL[3:0])
- 0 = HLVDIF will not be set

REGISTER 39-2: HLVDCON1: LOW-VOLTAGE DETECT CONTROL REGISTER 1

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	SEL[3:0]			
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **SEL[3:0]:** High/Low Voltage Detection Limit Selection bits
Refer to Table 44-13 for voltage detection limits.

TABLE 39-2: SUMMARY OF REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
HLVDCON0	EN	—	OUT	RDY	—	—	INTH	INTL	659
HLVDCON1	—	—	—	—	SEL[3:0]				660

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

40.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

ICSP™ programming allows customers to manufacture circuit boards with unprogrammed devices. Programming can be done after the assembly process, allowing the device to be programmed with the most recent firmware or a custom firmware. Five pins are needed for ICSP™ programming:

- ICSPCLK
- ICSPDAT
- $\overline{\text{MCLR}}$/VPP
- VDD
- VSS

In Program/Verify mode the program memory, User IDs and the Configuration Words are programmed through serial communications. The ICSPDAT pin is a bidirectional I/O used for transferring the serial data and the ICSPCLK pin is the clock input. For more information on ICSP™ refer to the **"**PIC18F26/27/45/46/47/55/56/57K42 *Memory Programming Specification***"** (DS40001886).

40.1 High-Voltage Programming Entry Mode

The device is placed into High-Voltage Programming Entry mode by holding the ICSPCLK and ICSPDAT pins low then raising the voltage on $\overline{\text{MCLR}}$/VPP to VIHH.

40.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using VDD only, without high voltage. When the LVP bit of Configuration Words is set to '1', the low-voltage ICSP™ programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to '0'.

Entry into the Low-Voltage Programming Entry mode requires the following steps:

1. $\overline{\text{MCLR}}$ is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, $\overline{\text{MCLR}}$ must be held at VIL for as long as Program/Verify mode is to be maintained.

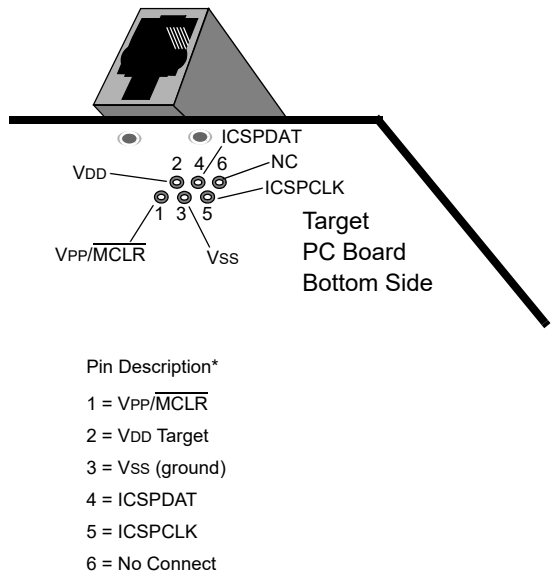
If low-voltage programming is enabled (LVP = 1), the $\overline{\text{MCLR}}$ Reset function is automatically enabled and cannot be disabled. See **Section 6.5 "MCLR"** for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

40.3 Common Programming Interfaces

Connection to a target device is typically done through an ICSP™ header. A commonly found connector on development tools is the RJ-11 in the 6P6C (6-pin, 6-connector) configuration. See Figure 40-1.

FIGURE 40-1: ICD RJ-11 STYLE CONNECTOR INTERFACE

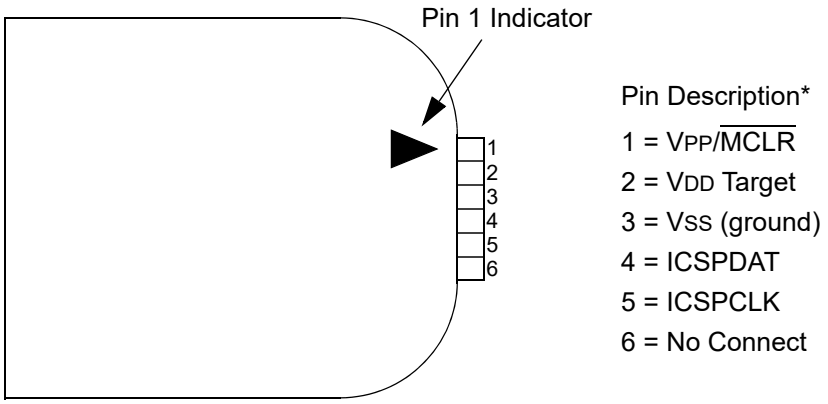


Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 40-2.

For additional interface recommendations, refer to your specific device programmer manual prior to PCB design.

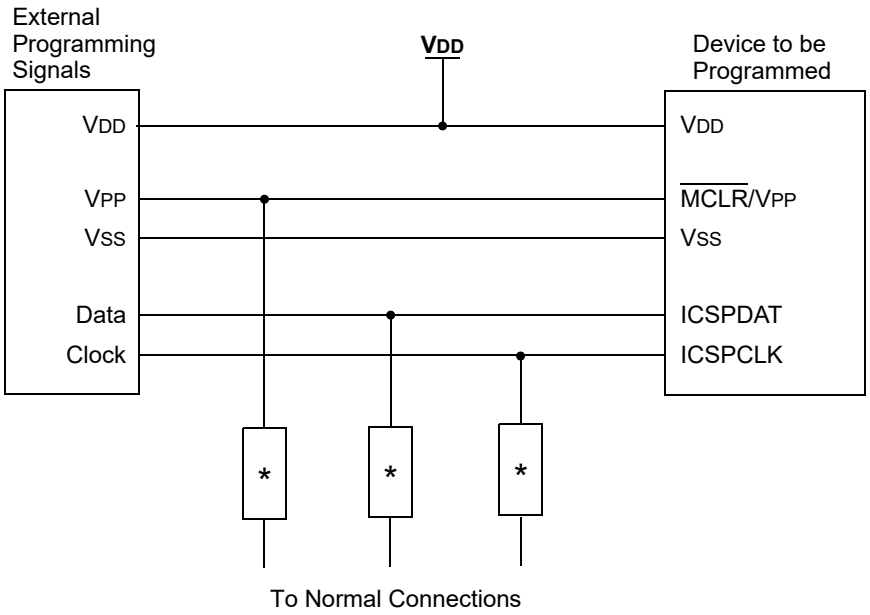
It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 40-3 for more information.

FIGURE 40-2: PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE



* The 6-pin header (0.100" spacing) accepts 0.025" square pins.

FIGURE 40-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING



* Isolation devices (as required).

41.0 INSTRUCTION SET SUMMARY

PIC18(L)F26/27/45/46/47/55/56/57K42 devices incorporate the standard set of PIC18 core instructions, as well as an extended set of instructions, for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

41.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC® MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are few instructions that require two- or three-program memory locations and two that require three-program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 41-3 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 41-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The destination of the result (specified by ‘d’)
3. The accessed memory (specified by ‘a’)

The file register designator ‘f’ specifies which file register is to be used by the instruction. The destination designator ‘d’ specifies where the result of the operation is to be placed. If ‘d’ is zero, the result is placed in the WREG register. If ‘d’ is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The bit in the file register (specified by ‘b’)
3. The accessed memory (specified by ‘a’)

The bit field designator ‘b’ selects the number of the bit affected by the operation, while the file register designator ‘f’ represents the number of the file in which the bit is located.

The literal instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The control instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the `CALL` or `RETURN` instructions (specified by ‘s’)
- The mode of the table read and table write instructions (specified by ‘m’)
- No operand required (specified by ‘—’)

All instructions are a single word, except for few two or three word instructions. These instructions were made two- or three-word to contain the required information in 32 or 48 bits. In the second word and third words, the four MSbs are ‘`1`’s. If this second or third word is executed as an instruction (by itself), it will execute as a `NOP`.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a `NOP`.

The two-word instructions execute in two instruction cycles and three-word instructions execute in three instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs. Two-word branch instructions (if true) would take 3 μs.

Figure 41-1 shows the general formats that the instructions can have. All examples use the convention ‘nnh’ to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 41-3, lists the standard instructions recognized by the Microchip Assembler (MPASM™).

Section 41.1.1 “Standard Instruction Set” provides a description of each instruction.

TABLE 41-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
ACCESS	ACCESS = 0: RAM access bit symbol
BANKED	BANKED = 1: RAM access bit symbol
bbb	Bit address within an 8-bit file register (0 to 7)
BSR	Bank Select Register. Used to select the current RAM bank.
d	Destination select bit; d = 0: store result in WREG, d = 1: store result in file register f.
dest	Destination either the WREG register or the specified register file location
f	8-bit Register file address (00h to FFh)
f_n	FSR Number (0 to 2)
f_s	12-bit Register file address (000h to FFFh) or 14-bit Register file address (0000h to 3FFFh). This is the source address.
f_d	12-bit Register file address (000h to FFFh) or 14-bit Register file address (0000h to 3FFFh). This is the destination address.
z_s	7-bit literal offset for FSR2 to used as register file address (000h to FFFh). This is the source address.
z_d	7-bit literal offset for FSR2 to used as register file address (000h to FFFh). This is the destination address.
k	Literal field, constant data or label (may be a 6-bit, 8-bit, 12-bit or a 20-bit value)
label	Label name
mm	The mode of the TBLPTR register for the Table Read and Table Write instructions Only used with Table Read and Table Write instructions:
*	No Change to register (such as TBLPTR with Table reads and writes)
*+	Post-Increment register (such as TBLPTR with Table reads and writes)
*-	Post-Decrement register (such as TBLPTR with Table reads and writes)
+*	Pre-Increment register (such as TBLPTR with Table reads and writes)
n	The relative address (2's complement number) for relative branch instructions, or the direct address for Call/Branch and Return instructions
PRODH	Product of Multiply high byte
PRODL	Product of Multiply low byte
s	Fast Call / Return mode select bit. s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
u	Unused or Unchanged
W	W = 0: Destination select bit symbol
WREG	Working register (accumulator)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
TBLPTR	21-bit Table Pointer (points to a Program Memory location)
TABLAT	8-bit Table Latch
TOS	Top of Stack
PC	Program Counter
PCL	Program Counter Low Byte
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch
PCLATU	Program Counter Upper Byte Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
TO	Time-out bit
PD	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[ ]	Indexed address
( )	Contents
→	Assigned to

TABLE 41-1: OPCODE FIELD DESCRIPTIONS (CONTINUED)

Field	Description
{ }	Optional argument
[expr]<n>	Specifies bit n of the register indicated by the pointer expr
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

FIGURE 41-1: General Format for Instructions (1/2)

Byte-oriented file register operations — **Example Instruction**

15 … 10	9	8	7 … 0
OPCODE	d	a	f (FILE #)

`ADDWF MYREG, W, B`

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

Byte to Byte move operations (2-word)

15 … 12	11 … 0
OPCODE	f (Source FILE #)
1111	f (Destination FILE #)

`MOVFF MYREG1, MYREG2`

f = 12-bit file register address

Byte to Byte move operations (3-word)

15 … 4	3 … 0
OPCODE	FILE #
1111	FILE #
1111	FILE #

`MOVFFL MYREG1, MYREG2`

Bit-oriented file register operations

15 … 12	11 … 9	8	7 … 0
OPCODE	b (BIT #)	a	f (FILE #)

`BSF MYREG, bit, B`

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

Literal operations

15 … 8	7 … 0
OPCODE	k (literal)

`MOVLW 7Fh`

k = 8-bit immediate value

FIGURE 41-2: General Format for Instructions (2/2)

Control operations

CALL, GOTO and Branch operations

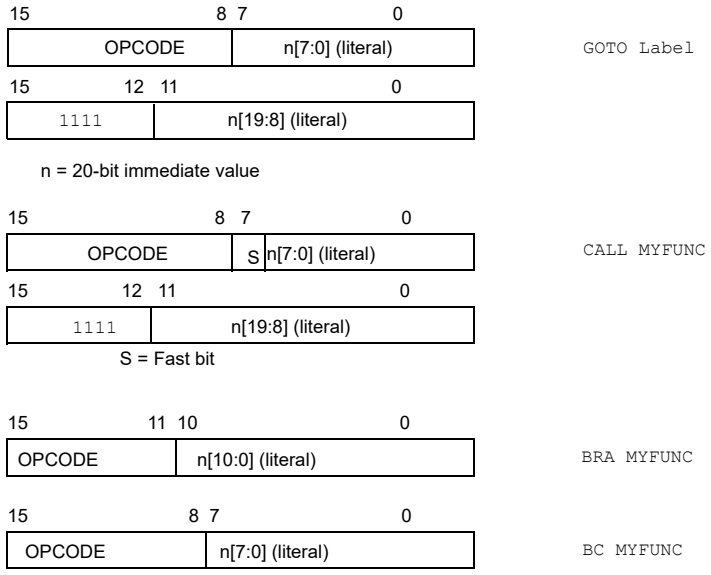


TABLE 41-2: INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER INSTRUCTIONS									
ADDWF	f, d ,a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	
MOVF	f, d, a	Move f to WREG or f	1	0101	00da	ffff	ffff	Z, N	
MOVFF	f_s, f_d	Move f_s (source) to 1st word	2	1100	ffff	ffff	ffff	None	2, 3
		f_d (destination) 2nd word		1111	ffff	ffff	ffff		
MOVFFL	f_s, f_d	Move f_s (source) to	3	0000	0000	0110	ffff	None	2
		g (full destination)		1111	ffff	ffff	ffgg		
		f_d (full destination)3rd word		1111	gggg	gggg	gggg		
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	
BYTE-ORIENTED SKIP INSTRUCTIONS									
CPFSEQ	f, a	Compare f with WREG, skip =	1 - 4	0110	001a	ffff	ffff	None	1
CPFSGT	f, a	Compare f with WREG, skip >	1 - 4	0110	010a	ffff	ffff	None	1
CPFSLT	f, a	Compare f with WREG, skip <	1 - 4	0110	000a	ffff	ffff	None	1
DECFSZ	f, d, a	Decrement f, Skip if 0	1 - 4	0010	11da	ffff	ffff	None	1
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 - 4	0100	11da	ffff	ffff	None	1
INCFSZ	f, d, a	Increment f, Skip if 0	1 - 4	0011	11da	ffff	ffff	None	1
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 - 4	0100	10da	ffff	ffff	None	1
TSTFSZ	f, a	Test f, skip if 0	1 - 4	0110	011a	ffff	ffff	None	1
BIT-ORIENTED FILE REGISTER INSTRUCTIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	
BIT-ORIENTED SKIP INSTRUCTIONS									
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 - 4	1011	bbba	ffff	ffff	None	1
BTFSS	f, b, a	Bit Test f, Skip if Set	1 - 4	1010	bbba	ffff	ffff	None	1

Note 1: If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a `NOP`.

2: Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

3: f_s and f_d do not cover the full memory range. 2 MSBs of bank selection are forced to ‘b00 to limit the range of these instructions to lower 4k addressing space.

TABLE 41-2: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
CONTROL INSTRUCTIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	1
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	1
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	1
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	1
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	1
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	1
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	1
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	1
CALL	k, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None	2
		2nd word		1111	kkkk	kkkk	kkkk		
CALLW	—	Call subroutine using WREG	2	0000	0000	0001	0100	None	1
GOTO	k	Go to address 1st word	2	1110	1111	kkkk	kkkk	None	2
	—	2nd word		1111	kkkk	kkkk	kkkk		
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	1
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	None	1
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	1
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	1
INHERENT INSTRUCTIONS									
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	None	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	2
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RESET		Software device Reset	1	0000	0000	1111	1111	All	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	None	

Note 1: If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a `NOP`.

2: Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

3: f_s and f_d do not cover the full memory range. 2 MSBs of bank selection are forced to ‘b00 to limit the range of these instructions to lower 4k addressing space.

TABLE 41-2: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
LITERAL INSTRUCTIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f_n, k	Load FSR(f_n) with a 14-bit literal (k)	2	1110 1111	1110 00kk	00ff kkkk	kkkk kkkk	None	
ADDFSR	f_n, k	Add FSR(f_n) with (k)	1	1110	1000	ffkk	kkkk	None	
SUBFSR	f_n, k	Subtract (k) from FSR(f_n)	1	1110	1001	ffkk	kkkk	None	
MOVLB	k	Move literal to BSR[5:0]	1	0000	0001	00kk	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY – PROGRAM MEMORY INSTRUCTIONS									
TBLRD*		Table Read	2 - 5	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2 - 5	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a `NOP`.

2: Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

3: f_s and f_d do not cover the full memory range. 2 MSBs of bank selection are forced to ‘b00 to limit the range of these instructions to lower 4k addressing space.

41.1.1 STANDARD INSTRUCTION SET

ADDFSR — Add Literal to FSR

Syntax:	ADDFSR f, k
Operands:	$0 \le k \le 63$ $f \in [0, 1, 2]$
Operation:	FSR(f) + k → FSR(f)
Status Affected:	None
Encoding:	1110 | 1000 | ffkk | kkkk
Description:	The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

ADDLW — ADD literal to W

Syntax:	ADDLW k
Operands:	$0 \le k \le 255$
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 15h`

Before Instruction
W = 10h
After Instruction
W = 25h

ADDWF — ADD W to f

Syntax:	ADDWF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
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 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ADDWF REG, 0, 0`

Before Instruction
W = 17h
REG = 0C2h
After Instruction
W = 0D9h
REG = 0C2h

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

ADDWFC — ADD W and CARRY bit to f

Syntax:	ADDWFC f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) + (f) + (C) → dest
Status Affected:	N,OV, C, DC, Z
Encoding:	0010 | 00da | ffff | ffff
Description:	Add W, the CARRY flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ADDWFC REG, 0, 1`

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

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

ANDLW — AND literal with W

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

Q Cycle Activity:

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

Example: `ANDLW 05Fh`

Before Instruction
W = A3h

After Instruction
W = 03h

ANDWF — AND W with f

Syntax:	ANDWF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) .AND. (f) → dest
Status Affected:	N, Z
Encoding:	0001 | 01da | ffff | ffff
Description:	The contents of W are AND'ed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ANDWF REG, 0, 0`

Before Instruction
W = 17h
REG = C2h
After Instruction
W = 02h
REG = C2h

BC — Branch if Carry

Syntax:	BC n
Operands:	$-128 \le n \le 127$
Operation:	if CARRY bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0010 | nnnn | nnnn
Description:	If the CARRY bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BC 5`

Before Instruction
PC = address (HERE)
After Instruction
If CARRY = 1;
PC = address (HERE + 12)
If CARRY = 0;
PC = address (HERE + 2)

BCF — Bit Clear f

Syntax:	BCF f, b {,a}
Operands:	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$
Operation:	$0 \rightarrow f<b>$
Status Affected:	None
Encoding:	1001 | bbba | ffff | ffff
Description:	Bit 'b' in register 'f' is cleared. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

BN — Branch if Negative

Syntax:	BN n
Operands:	$-128 \le n \le 127$
Operation:	if NEGATIVE bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0110 | nnnn | nnnn
Description:	If the NEGATIVE bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BN Jump`

Before Instruction
PC = address (HERE)
After Instruction
If NEGATIVE = 1;
PC = address (Jump)
If NEGATIVE = 0;
PC = address (HERE + 2)

BNC — Branch if Not Carry

Syntax:	BNC n
Operands:	-128 ≤ n ≤ 127
Operation:	if CARRY bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0011 | nnnn | nnnn
Description:	If the CARRY bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNC Jump`

```
Before Instruction
    PC          =   address (HERE)
After Instruction
    If CARRY    =   0;
        PC      =   address (Jump)
    If CARRY    =   1;
        PC      =   address (HERE + 2)
```

BNN — Branch if Not Negative

Syntax:	BNN n
Operands:	-128 ≤ n ≤ 127
Operation:	if NEGATIVE bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0111 | nnnn | nnnn
Description:	If the NEGATIVE bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNN Jump`

```
Before Instruction
    PC          =   address (HERE)
After Instruction
    If NEGATIVE =   0;
        PC      =   address (Jump)
    If NEGATIVE =   1;
        PC      =   address (HERE + 2)
```

BNOV — Branch if Not Overflow

Syntax:	BNOV n
Operands:	-128 ≤ n ≤ 127
Operation:	if OVERFLOW bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0101 | nnnn | nnnn
Description:	If the OVERFLOW bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNOV Jump`

Before Instruction
PC = address (HERE)
After Instruction
If OVERFLOW = 0;
PC = address (Jump)
If OVERFLOW = 1;
PC = address (HERE + 2)

BNZ — Branch if Not Zero

Syntax:	BNZ n
Operands:	-128 ≤ n ≤ 127
Operation:	if ZERO bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0001 | nnnn | nnnn
Description:	If the ZERO bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNZ Jump`

Before Instruction
PC = address (HERE)
After Instruction
If ZERO = 0;
PC = address (Jump)
If ZERO = 1;
PC = address (HERE + 2)

BRA — Unconditional Branch

Syntax:	BRA n
Operands:	-1024 ≤ n ≤ 1023
Operation:	(PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1101 | 0nnn | nnnn | nnnn
Description:	Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE BRA Jump`

Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

BSF — Bit Set f

Syntax:	BSF f, b {,a}
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
Operation:	1 → f[b>
Status Affected:	None
Encoding:	1000 | bbba | ffff | ffff
Description:	Bit 'b' in register 'f' is set. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `BSF FLAG_REG, 7, 1`

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

BTFSC — Bit Test File, Skip if Clear

Syntax: BTFSC f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

1011	bbba	ffff	ffff

Description: If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh).
See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

BTFSS — Bit Test File, Skip if Set

Syntax: BTFSS f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

1010	bbba	ffff	ffff

Description: If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh).
See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

BTG — Bit Toggle f

Syntax:	BTG f, b {,a}
Operands:	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$
Operation:	$(\overline{f<b>}) \rightarrow f<b>$
Status Affected:	None
Encoding:	0111 | bbba | ffff | ffff
Description:	Bit 'b' in data memory location 'f' is inverted. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

BOV — Branch if Overflow

Syntax:	BOV n
Operands:	$-128 \le n \le 127$
Operation:	if OVERFLOW bit is '1' $(PC) + 2 + 2n \rightarrow PC$
Status Affected:	None
Encoding:	1110 | 0100 | nnnn | nnnn
Description:	If the OVERFLOW bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BOV Jump`

Before Instruction
PC = address (HERE)
After Instruction
If OVERFLOW = 1;
PC = address (Jump)
If OVERFLOW = 0;
PC = address (HERE + 2)

BZ — Branch if Zero

Syntax:	BZ n
Operands:	-128 ≤ n ≤ 127
Operation:	if ZERO bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0000 | nnnn | nnnn
Description:	If the ZERO bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BZ Jump`

```
Before Instruction
    PC          =   address (HERE)
After Instruction
    If ZERO     =   1;
        PC      =   address (Jump)
    If ZERO     =   0;
        PC      =   address (HERE + 2)
```

CALL — Subroutine Call

Syntax:	CALL k {,s}
Operands:	0 ≤ k ≤ 1048575 s ∈ [0,1]
Operation:	(PC) + 4 → TOS, k → PC<20:1>, if s = 1 (W) → WREG_CSHAD, (Status) → STATUS_CSHAD, (BSR) → BSR_CSHAD
Status Affected:	None
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 | 110s | $k_7$kkk | kkkk$_0$ 1111 | k_{19}kkk | kkkk | kkkk$_8$
Description:	Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WREG_CSHAD, STATUS_CSHAD and BSR_CSHAD. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a 2-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE, 1`

```
Before Instruction
    PC          =   address (HERE)
After Instruction
    PC          =   address (THERE)
    TOS         =   address (HERE + 4)
    WREG_CSHAD  =   W
    BSR_CSHAD   =   BSR
    STATUS_CSHAD =  Status
```

CALLW — Subroutine Call Using WREG

Syntax:	CALLW
Operands:	None
Operation:	(PC + 2) → TOS, (W) → PCL, (PCLATH) → PCH, (PCLATU) → PCU
Status Affected:	None
Encoding:	0000 0000 0001 0100
Description	First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched. Unlike CALL, there is no option to update W, Status or BSR.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read WREG	PUSH PC to stack	No operation
No operation	No operation	No operation	No operation

Example: `HERE CALLW`

Before Instruction
- PC = address (HERE)
- PCLATH = 10h
- PCLATU = 00h
- W = 06h

After Instruction
- PC = 001006h
- TOS = address (HERE + 2)
- PCLATH = 10h
- PCLATU = 00h
- W = 06h

CLRF — Clear f

Syntax:	CLRF f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	000h → f 1 → Z
Status Affected:	Z
Encoding:	0110 101a ffff ffff
Description:	Clears the contents of the specified register. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `CLRF FLAG_REG, 1`

Before Instruction
- FLAG_REG = 5Ah

After Instruction
- FLAG_REG = 00h

CLRWDT — Clear Watchdog Timer

Syntax:	CLRWDT
Operands:	None
Operation:	000h → WDT, 000h → WDT postscaler, 1 → $\overline{TO}$, 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	0000 | 0000 | 0000 | 0100
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the post-scaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: CLRWDT

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 00h
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF — Complement f

Syntax:	COMF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(\bar{f}) \rightarrow$ dest
Status Affected:	N, Z
Encoding:	0001 | 11da | ffff | ffff
Description:	The contents of register ‘f’ are complemented. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank. If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: COMF REG, 0, 0

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

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

Syntax:	CPFSEQ f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [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 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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 = Address (HERE)
- W = ?
- REG = ?

After Instruction
- If REG = W;
  - PC = Address (EQUAL)
- If REG ≠ W;
  - PC = Address (NEQUAL)

CPFSGT — Compare f with W, skip if f > W

Syntax:	CPFSGT f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(f) – (W), skip if (f) > (W) (unsigned comparison)
Status Affected:	None
Encoding:	0110 010a ffff ffff
Description:	Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

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

If skip and followed by 2-word instruction:

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

Example:

```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```

Before Instruction
- PC = Address (HERE)
- W = ?

After Instruction
- If REG > W;
  - PC = Address (GREATER)
- If REG ≤ W;
  - PC = Address (NGREATER)

CPFSLT — Compare f with W, skip if f < W

Syntax:	CPFSLT 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 | 000a | ffff | ffff
Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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    CPFSLT REG, 1
NLESS   :
LESS    :
```

Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

DAW — Decimal Adjust W Register

Syntax:	DAW
Operands:	None

Operation:	If (W<3:0>) > 9] or [DC = 1] then
	(W<3:0>) + 6 → W<3:0>;
	else
	(W<3:0>) → W[3:0>;
	
	If [(W<7:4>) + DC > 9] or [C = 1] then
	(W<7:4>) + 6 + DC → W<7:4> ;
	else
	(W<7:4>) + DC → W<7:4>

Status Affected:	C
Encoding:	0000 | 0000 | 0000 | 0111
Description:	DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example1:

```
DAW
```

Before Instruction
W = A5h
C = 0
DC = 0
After Instruction
W = 05h
C = 1
DC = 0

Example 2:

Before Instruction
W = CEh
C = 0
DC = 0
After Instruction
W = 34h
C = 1
DC = 0

DECF — Decrement f

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

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

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

0000	01da	ffff	ffff

Description: Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `DECF CNT, 1, 0`

Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1

DECFSZ — Decrement f, skip if 0

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

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

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 2-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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:	DCFSNZ f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
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 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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)

GOTO — Unconditional Branch

Syntax:	GOTO k
Operands:	$0 \le k \le 1048575$
Operation:	k → PC<20:1>
Status Affected:	None
Encoding: 1st word (k[7:0>) 2nd word(k[19:8>)	1110 | 1111 | $k_7$kkk | kkkk$_0$ 1111 | k_{19}kkk | kkkk | kkkk$_8$
Description:	GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a 2-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: GOTO THERE

After Instruction
PC = Address (THERE)

INCF — Increment f

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

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

Operation: (f) + 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

0010	10da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `INCF CNT, 1, 0`

Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?

After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1

INCFSZ — Increment f, skip if 0

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

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

Operation: (f) + 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0011	11da	ffff	ffff

Description: The contents of register 'f' are incremented. 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 2-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

INFSNZ — Increment f, skip if not 0

Syntax:	INFSNZ f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) + 1 → dest, skip if result ≠ 0
Status Affected:	None
Encoding:	0100 | 10da | ffff | ffff
Description:	The contents of register 'f' are incremented. 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 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction. 4 cycles if skip and followed by a 3-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    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

IORLW — Inclusive OR literal with W

Syntax:	IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. k → W
Status Affected:	N, Z
Encoding:	0000 | 1001 | kkkk | kkkk
Description:	The contents of W are ORed with the 8-bit literal 'k'. The result is placed in W.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `IORLW 35h`

Before Instruction
W = 9Ah
After Instruction
W = BFh

IORWF — Inclusive OR W with f

Syntax:	IORWF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) .OR. (f) → dest
Status Affected:	N, Z
Encoding:	0001 00da ffff ffff
Description:	Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `IORWF RESULT, 0, 1`

Before Instruction
RESULT = 13h
W = 91h

After Instruction
RESULT = 13h
W = 93h

LFSR — Load FSR

Syntax:	LFSR f, k
Operands:	$0 \le f \le 2$ $0 \le k \le 16383$
Operation:	k → FSRf
Status Affected:	None
Encoding:	1110 1111 / 1110 $00k_9k_8$ / 00ff kkkk / k_{13}kkk kkkk
Description:	The 14-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL

Example: `LFSR 2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

MOVF — Move f

Syntax:	MOVF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) → dest
Status Affected:	N, Z
Encoding:	0101 | 00da | ffff | ffff
Description:	The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVF REG, 0, 0`

Before Instruction
REG = 22h
W = FFh
After Instruction
REG = 22h
W = 22h

MOVFF — Move f to f

Syntax:	MOVFF f_s,f_d
Operands:	$0 \le f_s \le 4095$ $0 \le f_d \le 4095$
Operation:	$(f_s) \rightarrow f_d$
Status Affected:	None
Encoding: 1st word (source) 2nd word (destin.)	1100 | ffff | ffff | $ffff_s$ 1111 | ffff | ffff | $ffff_d$
Description:	The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh. MOVFF has curtailed the source and destination range to the lower 4 Kbyte space of memory (Banks 1 through 15). For everything else, use MOVFFL.
Words:	2
Cycles:	2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: `MOVFF REG1, REG2`

Before Instruction
REG1 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

MOVFFL — Move f to f (Long Range)

Syntax: MOVFFL f_s,f_d

Operands: $0 \le f_s \le 16383$
$0 \le f_d \le 16383$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word	0000	0000	0110	$f_sf_sf_sf_s$
2nd word	1111	$f_sf_sf_sf_s$	$f_sf_sf_sf_s$	$f_sf_sf_df_d$
3rd word	1111	$f_df_df_df_d$	$f_df_df_df_d$	$f_df_df_df_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' and destination 'f_d' can be anywhere in the 16 Kbyte data space (0000h to 3FFFh).
Either source or destination can be W (a useful special situation).
`MOVFFL` is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).
The `MOVFFL` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 3

Cycles: 3

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
Decode	Read register 'f_s' (src)	Process data	No operation
Decode	No operation No dummy read	No operation	Write register 'f_d' (dest)

Example: `MOVFFL 2000h, 200Ah`

Before Instruction
Contents of 2000h = 33h
Contents of 200Ah = 11h
After Instruction
Contents of 2000h = 33h
Contents of 200Ah = 33h

MOVLB — Move literal to BSR

Syntax: MOVLB k

Operands: $0 \le k \le 63$

Operation: $k \rightarrow$ BSR

Status Affected: None

Encoding:

0000	0001	00kk	kkkk

Description: The 6-bit literal 'k' is loaded into the Bank Select Register (BSR<5:0>). The value of BSR<7:6> always remains '0'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `MOVLB 5`

Before Instruction
BSR Register = 02h
After Instruction
BSR Register = 05h

MOVLW — Move literal to W

Syntax:	MOVLW k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow W$
Status Affected:	None
Encoding:	0000 | 1110 | kkkk | kkkk
Description:	The 8-bit literal 'k' is loaded into W.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVLW 5Ah`

After Instruction

W = 5Ah

MOVWF — Move W to f

Syntax:	MOVWF f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	$(W) \rightarrow f$
Status Affected:	None
Encoding:	0110 | 111a | ffff | ffff
Description:	Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVWF REG, 0`

Before Instruction

W = 4Fh
REG = FFh

After Instruction

W = 4Fh
REG = 4Fh

MULLW — Multiply literal with W

Syntax:	MULLW k
Operands:	$0 \le k \le 255$
Operation:	(W) x k → PRODH:PRODL
Status Affected:	None
Encoding:	0000 | 1101 | kkkk | kkkk
Description:	An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged. None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: `MULLW 0C4h`

Before Instruction
- W = E2h
- PRODH = ?
- PRODL = ?

After Instruction
- W = E2h
- PRODH = ADh
- PRODL = 08h

MULWF — Multiply W with f

Syntax:	MULWF f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	(W) x (f) → PRODH:PRODL
Status Affected:	None
Encoding:	0000 | 001a | ffff | ffff
Description:	An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged. None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example: `MULWF REG, 1`

Before Instruction
- W = C4h
- REG = B5h
- PRODH = ?
- PRODL = ?

After Instruction
- W = C4h
- REG = B5h
- PRODH = 8Ah
- PRODL = 94h

NEGF — Negate f

Syntax:	NEGF f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	$(\overline{f}) + 1 \rightarrow f$
Status Affected:	N, OV, C, DC, Z
Encoding:	0110 110a ffff ffff
Description:	Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `NEGF REG, 1`

Before Instruction
REG = 0011 1010 [3Ah]
After Instruction
REG = 1100 0110 [C6h]

NOP — No Operation

Syntax:	NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	0000 1111 0000 xxxx 0000 xxxx 0000 xxxx
Description:	No operation.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

POP	Pop Top of Return Stack
Syntax:	POP
Operands:	None
Operation:	(TOS) → bit bucket
Status Affected:	None
Encoding:	0000 0000 0000 0110
Description:	The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:

```
POP
GOTO      NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

PUSH	Push Top of Return Stack
Syntax:	PUSH
Operands:	None
Operation:	(PC) + 2 → TOS
Status Affected:	None
Encoding:	0000 0000 0000 0101
Description:	The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:

```
PUSH
```

Before Instruction
TOS = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah

RCALL — Relative Call

Syntax:	RCALL n
Operands:	-1024 ≤ n ≤ 1023
Operation:	(PC) + 2 → TOS, (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1101 | 1nnn | nnnn | nnnn
Description:	Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' PUSH PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE RCALL Jump`

Before Instruction
PC = Address (HERE)
After Instruction
PC = Address (Jump)
TOS = Address (HERE + 2)

RESET — Reset

Syntax:	RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.
Status Affected:	All
Encoding:	0000 | 0000 | 1111 | 1111
Description:	This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset by software.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: `RESET`

After Instruction
Registers = Reset Value
Flags* = Reset Value

RETFIE — Return from Interrupt

Syntax:	RETFIE {s}
Operands:	s ∈ [0,1]
Operation:	(TOS) → PC, if s = 1, context is restored into WREG, STATUS, BSR, FSR0H, FSR0L, FSR1H, FSR1L, FSR2H, FSR2L, PRODH, PRODL, PCLATH and PCLATU registers from the corresponding shadow registers. if s = 0, there is no change in status of any register.
Status Affected:	STAT<1:0> in INTCON1 register
Encoding:	0000 0000 0001 000s
Description:	Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WREG, STATUS, BSR, FSR0H, FSR0L, FSR1H, FSR1L, FSR2H, FSR2L, PRODH, PRODL, PCLATH and PCLATU, are loaded into corresponding registers. There are two sets of shadow registers, main context and low context. The set retrieved on RETFIE instruction execution depends on what the state of operation of the CPU was when RETFIE was executed. If 's' = 0, no update of these registers occurs (default).
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	POP PC from stack Set INTCONx.STAT bits and restore context
No operation	No operation	No operation	No operation

Example: `RETFIE 1`

After Interrupt

PC	=	TOS
WREG	=	WREG_SHAD
BSR	=	BSR_SHAD
STATUS	=	STATUS_SHAD
FSR0L/H	=	FSR0L/H_SHAD
FSR1L/H	=	FSR1L/H_SHAD
FSR2L/H	=	FSR2L/H_SHAD
PRODL/H	=	PRODL/H_SHAD
PCLATH/U	=	PCLATH/U_SHAD

RETLW — Return literal to W

Syntax:	RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → W, (TOS) → PC, PCLATU, PCLATH are unchanged
Status Affected:	None
Encoding:	0000 1100 kkkk kkkk
Description:	W is loaded with the 8-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The upper and high address latches (PCLATU/H) remains unchanged.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	POP PC from stack, Write to W
No operation	No operation	No operation	No operation

Example:

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
   :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
   :
   :
    RETLW kn   ; End of table
```

Before Instruction
W = 07h
After Instruction
W = v

TABLE 44-8: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS (CONTINUED)

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
OS21	F_{CY}	Instruction Frequency	—	FOSC/4	—	MHz	
OS22	T_{CY}	Instruction Period	62.5	$1/F_{CY}$	—	ns	

* 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: Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min” values with an external clock applied to OSC1 pin. When an external clock input is used, the “max” cycle time limit is “DC” (no clock) for all devices.

2: The system clock frequency (FOSC) is selected by the “main clock switch controls” as described in **Section 10.0 “Power-Saving Operation Modes”**.

3: The system clock frequency (FOSC) must meet the voltage requirements defined in the **Section 44.2 “Standard Operating Conditions”**.

4: LP, XT and HS oscillator modes require an appropriate crystal or resonator to be connected to the device. For clocking the device with the external square wave, one of the EC mode selections must be used.

TABLE 44-9: INTERNAL OSCILLATOR PARAMETERS(1)

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
OS50	FHFOSC	Precision Calibrated HFINTOSC Frequency	—	4 8 12 16 48 64	—	MHz	(**Note 2**)
OS51	FHFOSCLP	Low-Power Optimized HFINTOSC Frequency	0.92 1.84 0.88 1.76	1 2 1 2	1.08 2.16 1.12 2.24	MHz MHz MHz MHz	-40°C to 85°C -40°C to 85°C -40°C to 125°C -40°C to 125°C
OS53*	FLFOSC	Internal LFINTOSC Frequency	24.80	31	37.2	kHz	
OS54*	THFOSCST	HFINTOSC Wake-up from Sleep Start-up Time	— —	11 50	20 —	μs μs	VREGPM = 0 VREGPM = 1
OS56	TLFOSCST	LFINTOSC Wake-up from Sleep Start-up Time	—	0.2	—	ms	

* 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: To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

2: See Figure 44-6: Precision Calibrated HFINTOSC and MFINTOSC Frequency Accuracy Over Device VDD and Temperature.

FIGURE 44-6: PRECISION CALIBRATED HFINTOSC AND MFINTOSC FREQUENCY ACCURACY OVER DEVICE VDD AND TEMPERATURE

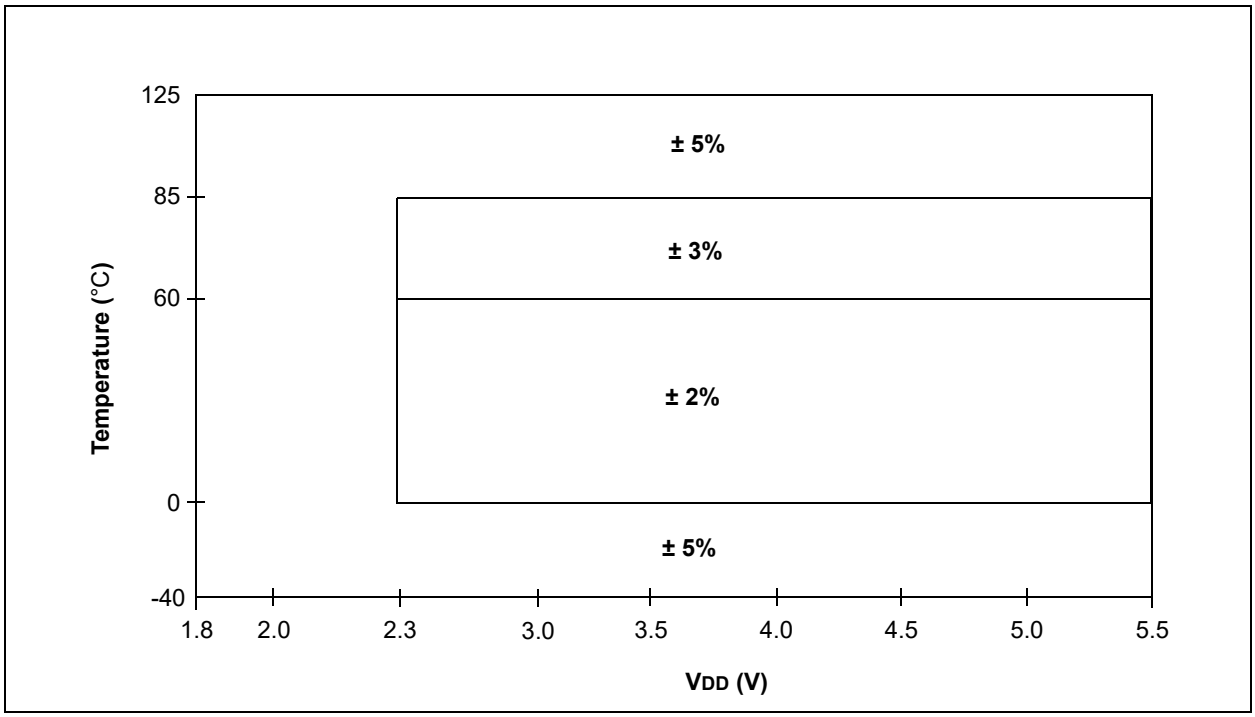


TABLE 44-10: PLL SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated) VDD ≥ 2.5V							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
PLL01	FPLLIN	PLL Input Frequency Range	4	—	16	MHz	
PLL02	FPLLOUT	PLL Output Frequency Range	16	—	64	MHz	**Note 1**
PLL03	TPLLST	PLL Lock Time from Start-up	—	200	—	µs	
PLL04	FPLLJIT	PLL Output Frequency Stability (Jitter)	-0.25	—	0.25	%	

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: The output frequency of the PLL must meet the FOSC requirements listed in Parameter D002.

FIGURE 44-7: CLKOUT AND I/O TIMING

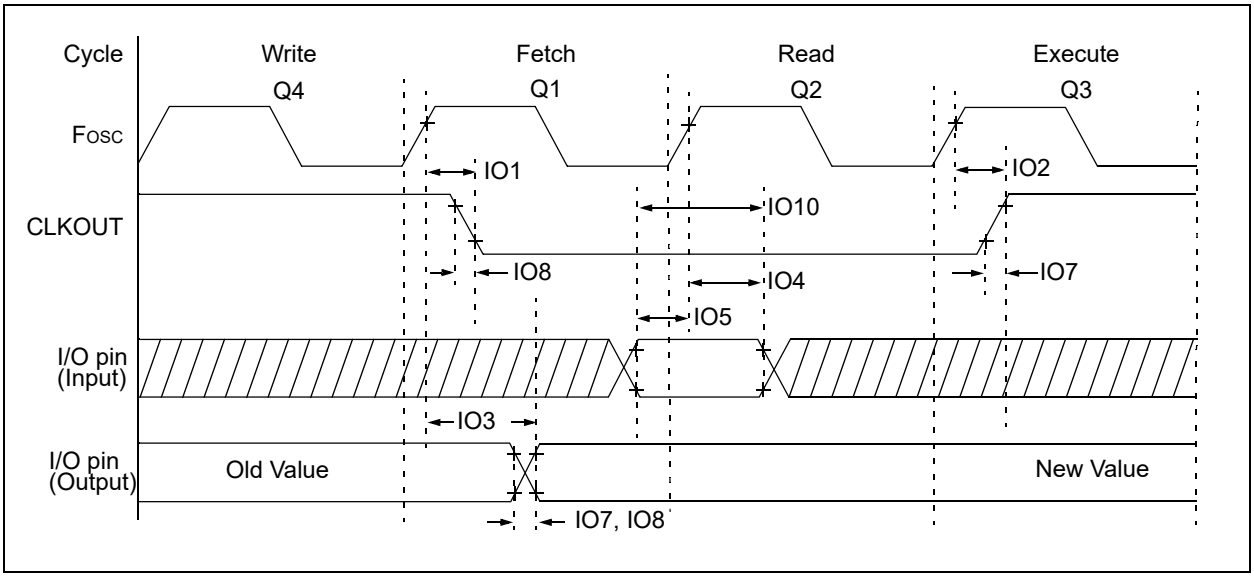
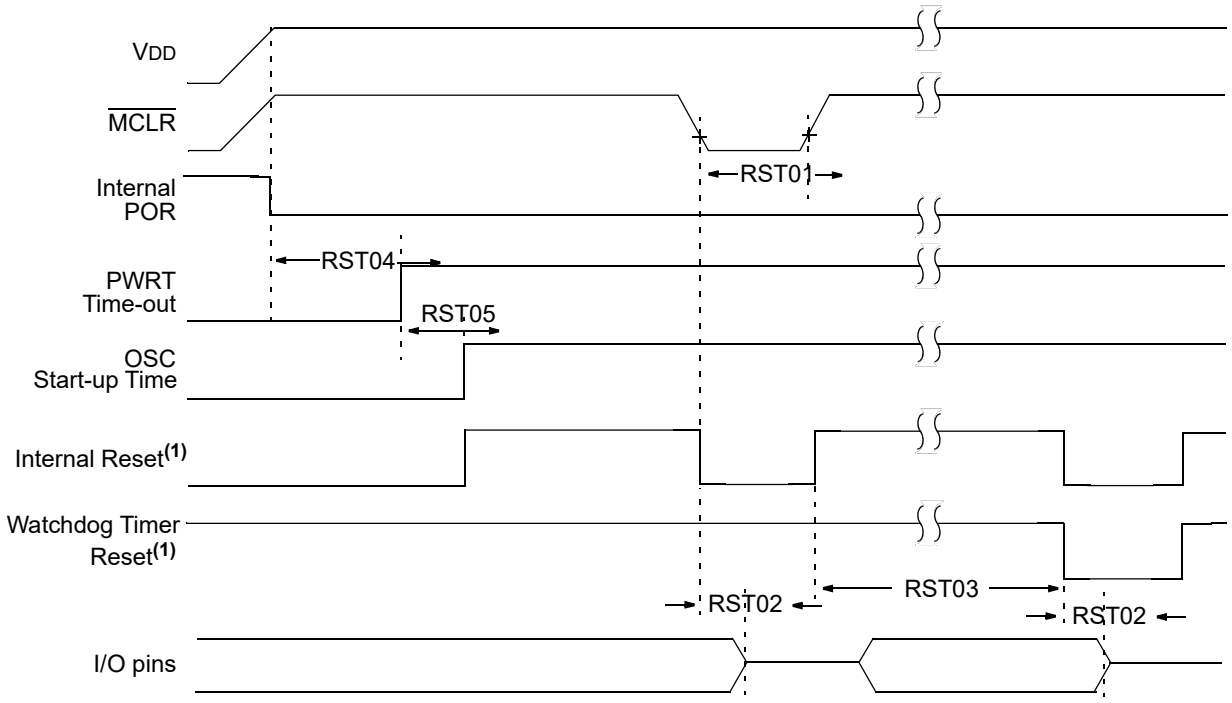


TABLE 44-11: I/O AND CLKOUT TIMING SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
IO1*	$T_{CLKOUTH}$	CLKOUT rising edge delay (rising edge Fosc (Q1 cycle) to falling edge CLKOUT	—	—	70	ns	
IO2*	$T_{CLKOUTL}$	CLKOUT falling edge delay (rising edge Fosc (Q3 cycle) to rising edge CLKOUT	—	—	72	ns	
IO3*	T_{IO_VALID}	Port output valid time (rising edge Fosc (Q1 cycle) to port valid)	—	50	70	ns	
IO4*	T_{IO_SETUP}	Port input setup time (Setup time before rising edge Fosc – Q2 cycle)	20	—	—	ns	
IO5*	T_{IO_HOLD}	Port input hold time (Hold time after rising edge Fosc – Q2 cycle)	50	—	—	ns	
IO6*	T_{IOR_SLREN}	Port I/O rise time, slew rate enabled	—	25	—	ns	V_{DD} = 3.0V
IO7*	T_{IOR_SLRDIS}	Port I/O rise time, slew rate disabled	—	5	—	ns	V_{DD} = 3.0V
IO8*	T_{IOF_SLREN}	Port I/O fall time, slew rate enabled	—	25	—	ns	V_{DD} = 3.0V
IO9*	T_{IOF_SLRDIS}	Port I/O fall time, slew rate disabled	—	5	—	ns	V_{DD} = 3.0V
IO10*	T_{INT}	INT pin high or low time to trigger an interrupt	25	—	—	ns	
IO11*	T_{IOC}	Interrupt-on-Change minimum high or low time to trigger interrupt	25	—	—	ns	

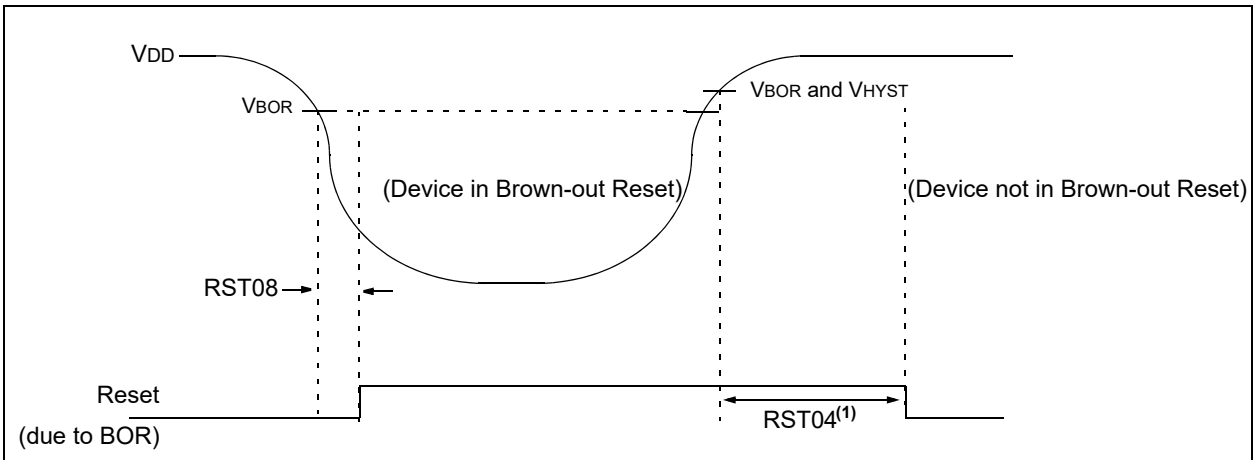
*These parameters are characterized but not tested.

FIGURE 44-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note 1: Asserted low.

FIGURE 44-9: BROWN-OUT RESET TIMING AND CHARACTERISTICS



Note 1: Delay depends on PWRTS[1:0] Configuration bits.

TABLE 44-12: RESET, WDT, OSCILLATOR START-UP TIMER, POWER-UP TIMER, BROWN-OUT RESET AND LOW-POWER BROWN-OUT RESET SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
RST01*	TMCLR	$\overline{\text{MCLR}}$ Pulse Width Low to ensure Reset	2	—	—	μs	
RST02*	TIOZ	I/O high-impedance from Reset detection	—	—	2	μs	
RST03	TWDT	Watchdog Timer Time-out Period	—	16	—	ms	1:512 Prescaler
RST04*	TPWRT	Power-up Timer Period	—	1 16 64	—	ms ms ms	PWRTS = 00 PWRTS = 01 PWRTS = 10
RST05	TOST	Oscillator Start-up Timer Period(1,2)	—	1024	—	TOSC	
RST06	VBOR	Brown-out Reset Voltage(4)	2.7 2.55 2.3 2.3 1.8	2.85 2.7 2.45 2.45 1.9	3.0 2.85 2.6 2.6 2.1	V V V V V	BORV = 00 BORV = 01 BORV = 10 BORV = 11 (PIC18Fxxx) BORV = 11 (PIC18LFxxx)
RST07	VBORHYS	Brown-out Reset Hysteresis	—	40	—	mV	
RST08	TBORDC	Brown-out Reset Response Time	—	3	—	μs	
RST09	VLPBOR	Low-Power Brown-out Reset Voltage	1.8	2	2.5	V	PIC18LFXXX only

* 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: By design, the Oscillator Start-up Timer (OST) counts the first 1024 cycles, independent of frequency.

2: To ensure these voltage tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

TABLE 44-13: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
HLVD01	V_{DET}	Voltage Detection	1.73(1)	1.90	2.07	V	HLVDSEL[3:0]=0000
			1.91	2.10	2.29	V	HLVDSEL[3:0]=0001
			2.05	2.25	2.45	V	HLVDSEL[3:0]=0010
			2.28	2.50	2.73	V	HLVDSEL[3:0]=0011
			2.37	2.60	2.83	V	HLVDSEL[3:0]=0100
			2.50	2.75	3.00	V	HLVDSEL[3:0]=0101
			2.64	2.90	3.16	V	HLVDSEL[3:0]=0110
			2.87	3.15	3.43	V	HLVDSEL[3:0]=0111
			3.05	3.35	3.65	V	HLVDSEL[3:0]=1000
			3.28	3.60	3.92	V	HLVDSEL[3:0]=1001
			3.41	3.75	4.09	V	HLVDSEL[3:0]=1010
			3.64	4.00	4.36	V	HLVDSEL[3:0]=1011
			3.82	4.20	4.58	V	HLVDSEL[3:0]=1100
			3.96	4.35	4.74	V	HLVDSEL[3:0]=1101
			4.23	4.65	5.07	V	HLVDSEL[3:0]=1110

* 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: Device operation below VDD = 1.8 V is not recommended.

TABLE 44-14: ANALOG-TO-DIGITAL CONVERTER (ADC) ACCURACY SPECIFICATIONS(1,2):

Operating Conditions (unless otherwise stated) VDD = 3.0V, TA = 25°C, TAD = 1μs							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
AD01	NR	Resolution	—	—	12	bit	
AD02	EIL	Integral Error	—	±0.1	±2.0	LSb	ADCREF+ = 3.0V, ADCREF-= 0V
AD03	EDL	Differential Error	—	±0.1	±1.0	LSb	ADCREF+ = 3.0V, ADCREF-= 0V
AD04	EOFF	Offset Error	—	0.5	6.0	LSb	ADCREF+ = 3.0V, ADCREF-= 0V
AD05	EGN	Gain Error	—	±0.2	±6.0	LSb	ADCREF+ = 3.0V, ADCREF-= 0V
AD06	VADREF	ADC Reference Voltage (ADREF+ - ADREF-)	1.8	—	VDD	V	
AD07	VAIN	Full-Scale Range	ADREF-	—	ADREF+	V	
AD08	ZAIN	Recommended Impedance of Analog Voltage Source	—	1	—	kΩ	
AD09	RVREF	ADC Voltage Reference Ladder Impedance	—	50	—	kΩ	**Note 3**

* 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: Total Absolute Error is the sum of the offset, gain and integral non-linearity (INL) errors.

2: The ADC conversion result never decreases with an increase in the input and has no missing codes.

3: This is the impedance seen by the VREF pads when the external reference pads are selected.

TABLE 44-15: ANALOG-TO-DIGITAL CONVERTER (ADC) CONVERSION TIMING SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
AD20	T_{AD}	ADC Clock Period	0.5	—	9	μs	Using F_{OSC} as the ADC clock source ADCS = 1
AD21			—	2	—	μs	Using ADCRC as the ADC clock source ADCS = 0
AD22	T_{CNV}	Conversion Time	—	14 T_{AD} + 2 T_{CY}	—	—	Using F_{OSC} as the ADC clock source ADCS = 1
			—	16 T_{AD} + 2 T_{CY}	—	—	Using ADCRC as the ADC clock source ADCS = 0
AD24	T_{HCD}	Sample and Hold Capacitor Disconnect Time	—	2 T_{AD} + 1 T_{CY}	—	—	Using F_{OSC} as the ADC clock source ADCS = 1
			—	3 T_{AD} + 2 T_{CY}	—	—	Using ADCRC as the ADC clock source ADCS = 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.

FIGURE 44-10: ADC CONVERSION TIMING (ADC CLOCK F_{OSC}-BASED)

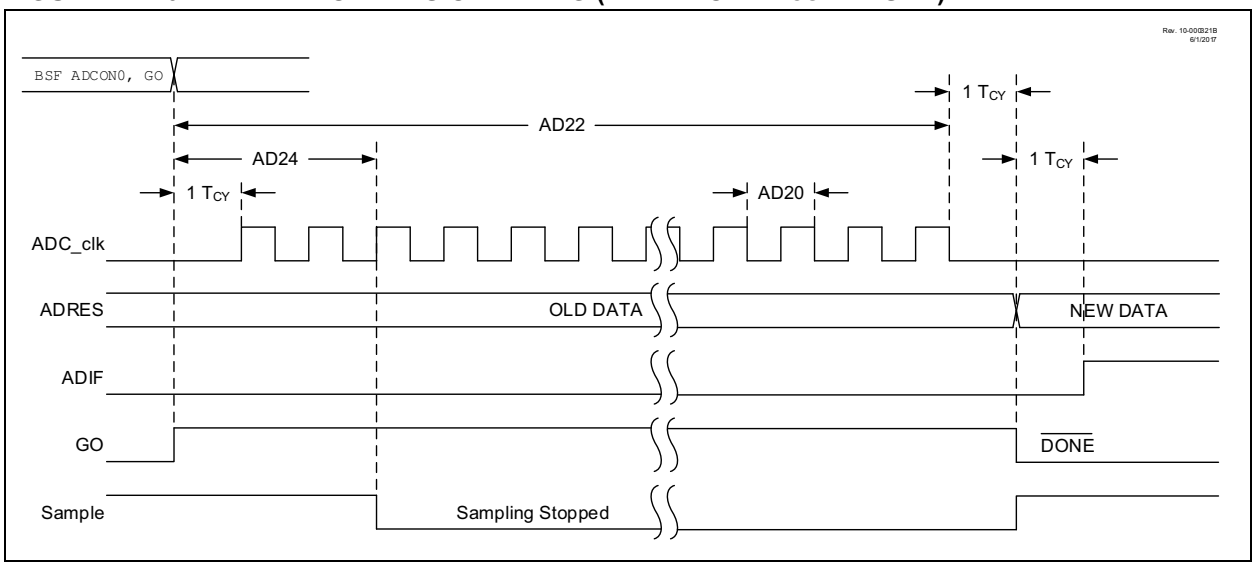
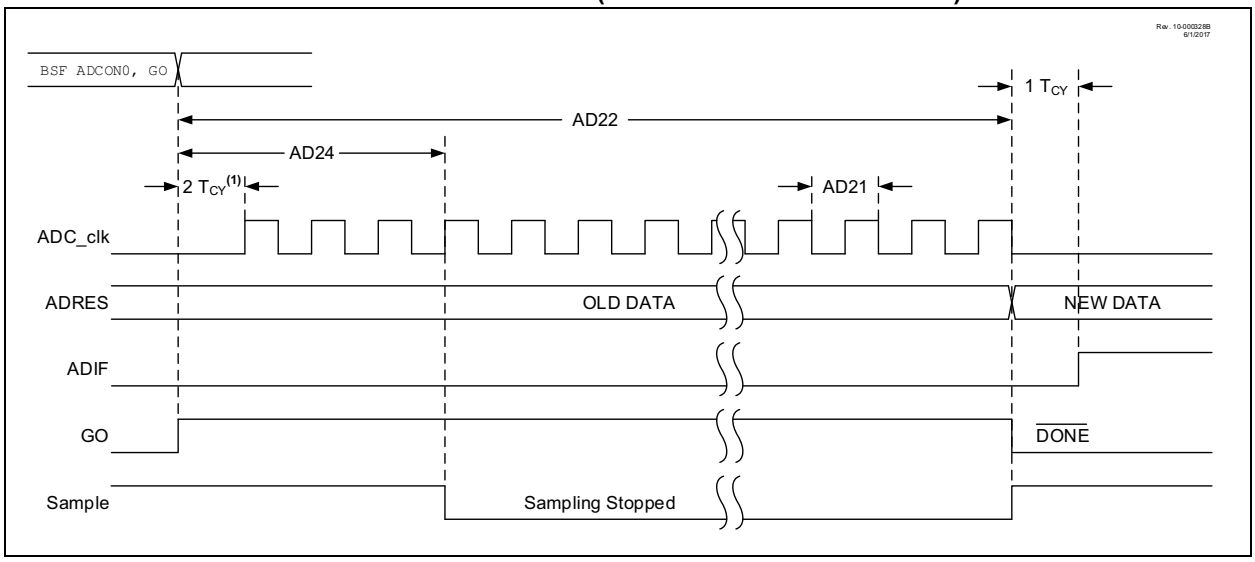


FIGURE 44-11: ADC CONVERSION TIMING (ADC CLOCK FROM ADCRC)



Note 1: If the ADC clock source is selected as ADCRC, a time of T_{CY} is added before the ADC clock starts. This allows the `SLEEP` instruction to be executed.

TABLE 44-16: COMPARATOR SPECIFICATIONS

Operating Conditions (unless otherwise stated)
VDD = 3.0V, TA = 25°C

Param No.	Sym.	Characteristics	Min.	Typ.	Max.	Units	Comments
CM01	VIOFF	Input Offset Voltage	—	—	±60	mV	VICM = VDD/2
CM02	VICM	Input Common Mode Range	GND	—	VDD	V	
CM03	CMRR	Common Mode Input Rejection Ratio	—	50	—	dB	
CM04	VHYST	Comparator Hysteresis	10	25	40	mV	
CM05	TRESP(1)	Response Time, Rising Edge	—	300	900	ns	
		Response Time, Falling Edge	—	220	500	ns	

* These parameters are characterized but not tested.

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

2: A mode change includes changing any of the control register values, including module enable.

TABLE 44-17: 5-BIT DAC SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)
VDD = 3.0V, TA = 25°C

Param No.	Sym.	Characteristics	Min.	Typ.	Max.	Units	Comments
DSB01	VLSB	Step Size	—	(VDACREF+ -VDACREF-) / 32	—	V	
DSB01	VACC	Absolute Accuracy	—	—	± 0.5	LSb	
DSB03*	RUNIT	Unit Resistor Value	—	5000	—	Ω	
DSB04*	TST	Settling Time(1)	—	—	10	µs	

* 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: Settling time measured while DACR[4:0] transitions from '00000' to '01111'.

TABLE 44-18: FIXED VOLTAGE REFERENCE (FVR) SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)

Param. No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
FVR01	VFVR1	1x Gain (1.024V)	-4	—	+4	%	VDD ≥ 2.5V, -40°C to 85°C
FVR02	VFVR2	2x Gain (2.048V)	-4	—	+4	%	VDD ≥ 2.5V, -40°C to 85°C
FVR03	VFVR4	4x Gain (4.096V)	-5	—	+5	%	VDD ≥ 4.75V, -40°C to 85°C
FVR04	TFVRST	FVR Start-up Time	—	25	—	us	

TABLE 44-19: ZERO-CROSS DETECT (ZCD) SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)
VDD = 3.0V, TA = 25°C

Param. No.	Sym.	Characteristics	Min	Typ†	Max	Units	Comments
ZC01	VPINZC	Voltage on Zero Cross Pin	—	0.75	—	V	
ZC02	IZCD_MAX	Maximum source or sink current	—	—	600	µA	
ZC03	TRESPH	Response Time, Rising Edge	—	1	—	µs	
	TRESPL	Response Time, Falling Edge	—	1	—	µs	

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

FIGURE 44-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

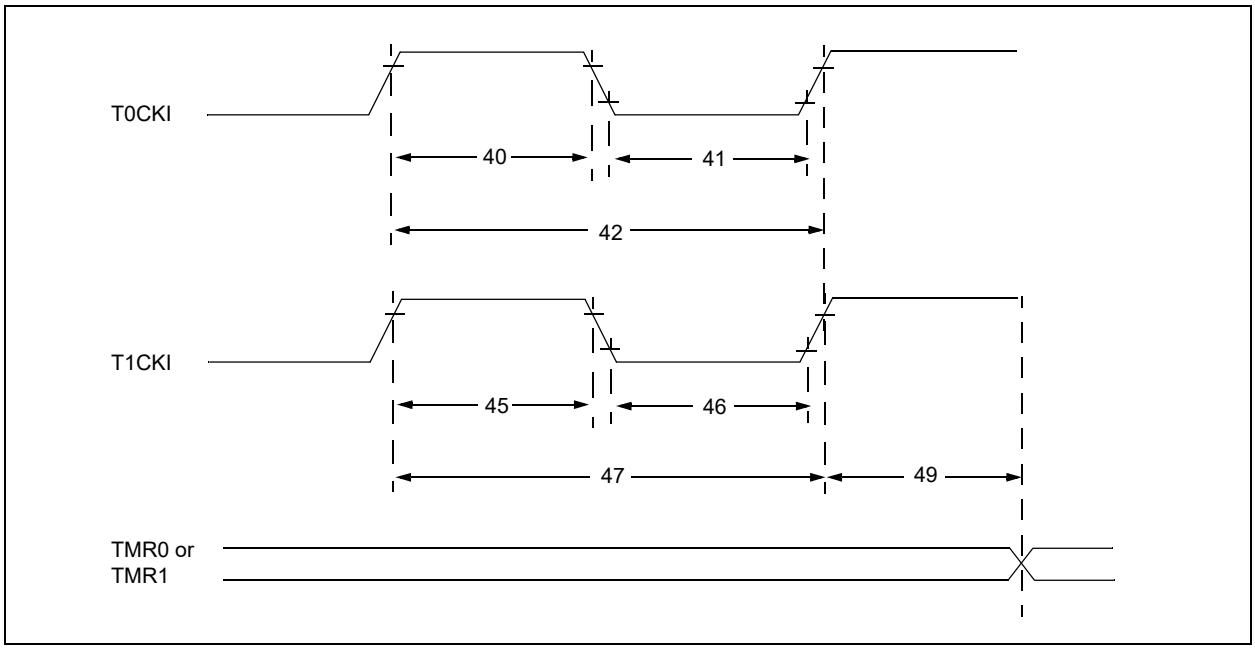


TABLE 44-20: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

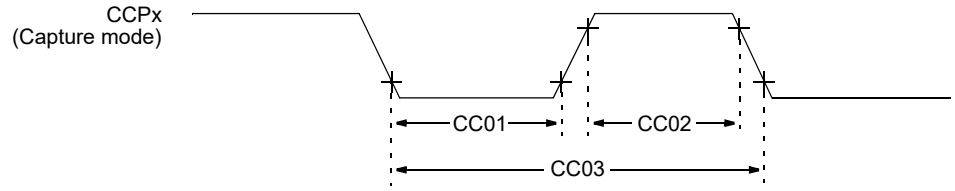
Standard Operating Conditions (unless otherwise stated)
Operating Temperature -40°C ≤ TA ≤ +125°C

Param No.	Sym.	Characteristic		Min.	Typ†	Max.	Units	Conditions
40*	TT0H	T0CKI High Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
41*	TT0L	T0CKI Low Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
42*	TT0P	T0CKI Period		Greater of: 20 or $\frac{T_{CY} + 40}{N}$	—	—	ns	N = prescale value
45*	TT1H	T1CKI High Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
46*	TT1L	T1CKI Low Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
47*	TT1P	T1CKI Input Period	Synchronous	Greater of: 30 or $\frac{T_{CY} + 40}{N}$	—	—	ns	N = prescale value
			Asynchronous	60	—	—	ns	
49*	TCKEZTMR1	Delay from External Clock Edge to Timer Increment		2 TOSC	—	7 TOSC	—	Timers in Sync mode

* 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.

FIGURE 44-13: CAPTURE/COMPARE/PWM TIMINGS (CCP)



Note: Refer to Figure 44-4 for load conditions.

TABLE 44-21: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)

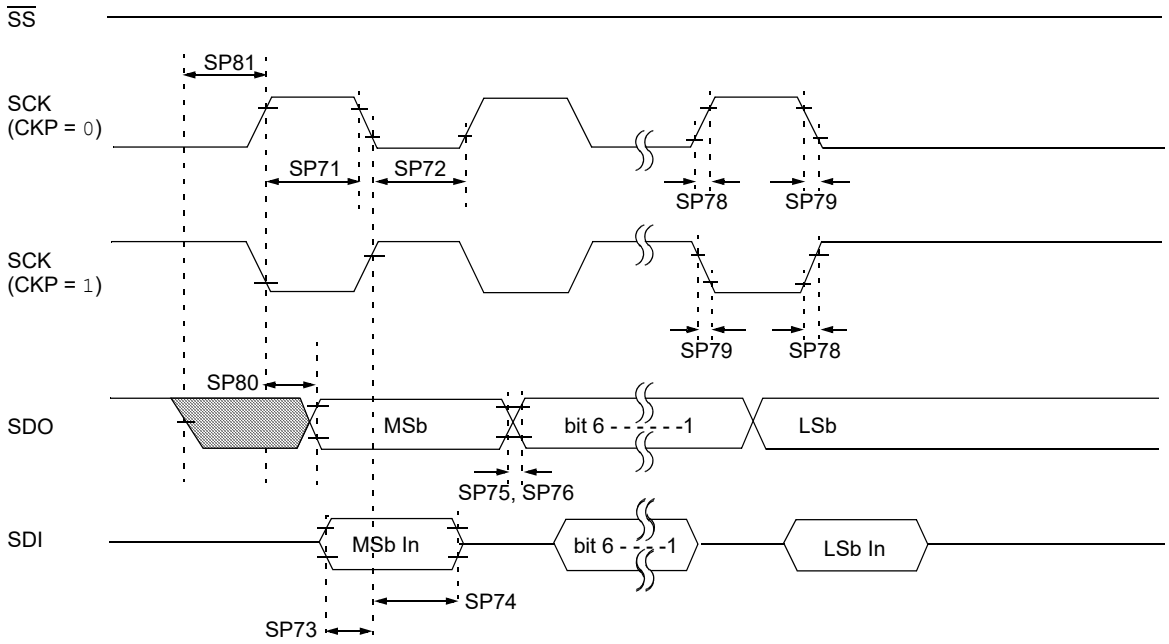
Standard Operating Conditions (unless otherwise stated)
Operating Temperature -40°C ≤ TA ≤ +125°C

Param No.	Sym.	Characteristic		Min.	Typ†	Max.	Units	Conditions
CC01*	TccL	CCPx Input Low Time	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	20	—	—	ns	
CC02*	TccH	CCPx Input High Time	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	20	—	—	ns	
CC03*	TccP	CCPx Input Period		$\frac{3T_{CY} + 40}{N}$	—	—	ns	N = prescale value

* These parameters are characterized but not tested.

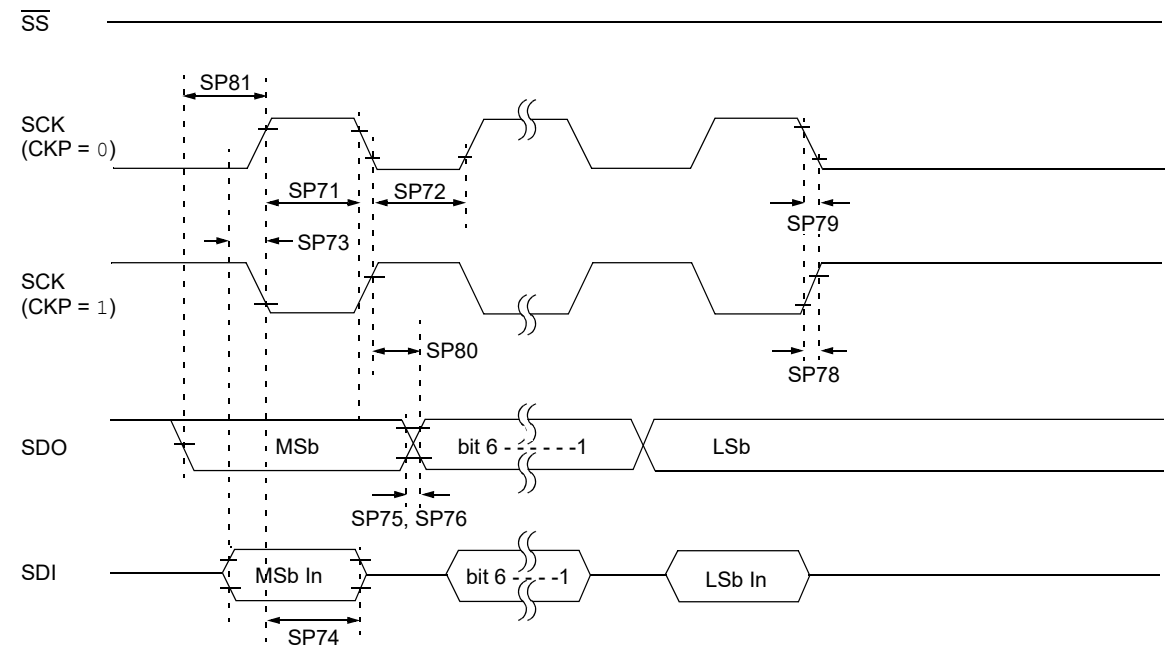
† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 44-14: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)



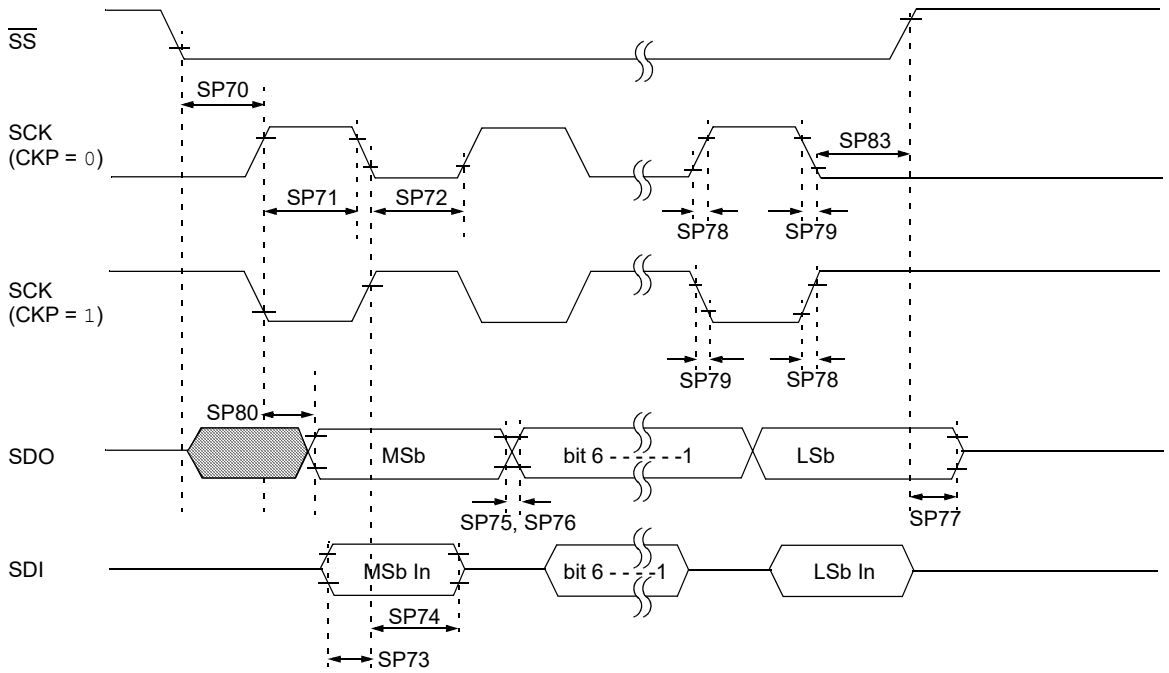
Note: Refer to Figure 44-4 for load conditions.

FIGURE 44-15: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)



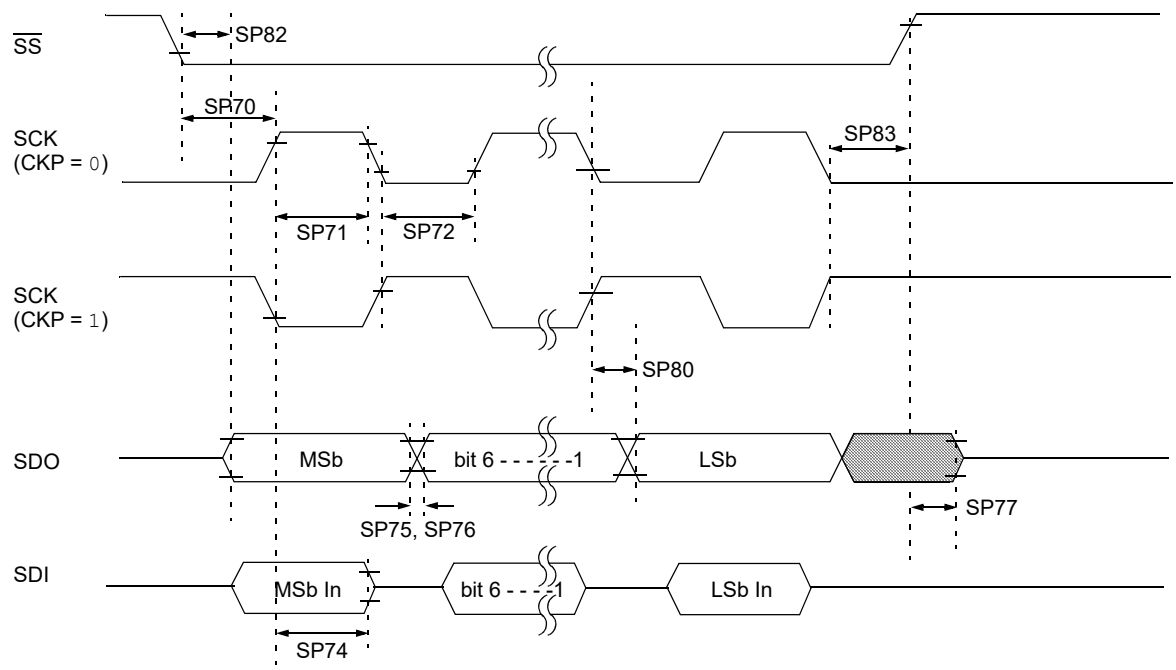
Note: Refer to Figure 44-4 for load conditions.

FIGURE 44-16: SPI SLAVE MODE TIMING (CKE = 0)



Note: Refer to Figure 44-4 for load conditions.

FIGURE 44-17: SPI SLAVE MODE TIMING (CKE = 1)



Note: Refer to Figure 44-4 for load conditions.

TABLE 44-22: SPI MODE REQUIREMENTS (MASTER MODE)

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
	TSCK	SCK Cycle Time (2x Prescaled)	61	—	—	ns	Transmit only mode
			—	16(1)	—	MHz	
			95	—	—	ns	Full duplex mode
			—	10(1)	—	MHz	
SP70*	TSSL2SCH, TSSL2SCL	SDO to SCK↓ or SCK↑ input	TSCK	—	—	ns	FST = 0
			0	—	—	ns	FST = 1
SP71*	TSCH	SCK output high time	0.5 TSCK - 12	—	0.5 TSCK + 12	ns	
SP72*	TSCL	SCK output low time	0.5 TSCK - 12	—	0.5 TSCK + 12	ns	
SP73*	TDIV2SCH, TDIV2SCL	Setup time of SDI data input to SCK edge	85	—	—	ns	
SP74*	TSCH2DIL, TSCL2DIL	Hold time of SDI data input to SCK edge	0	—	—	ns	
		Hold time of SDI data input to final SCK	0.5 TSCK	—	—	ns	CKE = 0, SMP = 1
SP75*	TDOR	SDO data output rise time	—	10	25	ns	CL = 50 pF
SP76*	TDOF	SDO data output fall time	—	10	25	ns	CL = 50 pF
SP78*	TSCR	SCK output rise time	—	10	25	ns	CL = 50 pF
SP79*	TSCF	SCK output fall time	—	10	25	ns	CL = 50 pF
SP80*	TSCH2DOV, TSCL2DOV	SDO data output valid after SCK edge	- 15	—	15	ns	CL = 20 pF
SP81*	TDOV2SCH, TDOV2SCL	SDO data output valid to first SCK edge	TSCK - 10	—	—	ns	CL = 20 pF CKE = 1
SP82*	TSSL2DOV	SDO data output valid after $\overline{SS}$↓ edge	—	—	50	ns	CL = 20 pF
SP83*	TSCH2SSH, TSCL2SSH	$\overline{SS}$ ↑ after last SCK edge	0.5 TSCK - 10	—	—	ns	
SP84*	TSSH2SSL	$\overline{SS}$↑ to $\overline{SS}$↓ edge	0.5 TSCK - 10	—	—	ns	

* 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: SPIxCON1.SMP bit must be set and the slew rate control must be disabled on the clock and data pins (clear the corresponding bits in SLRCONx register) for SPI to operate over 4 MHz.

TABLE 44-23: SPI MODE REQUIREMENTS (SLAVE MODE)

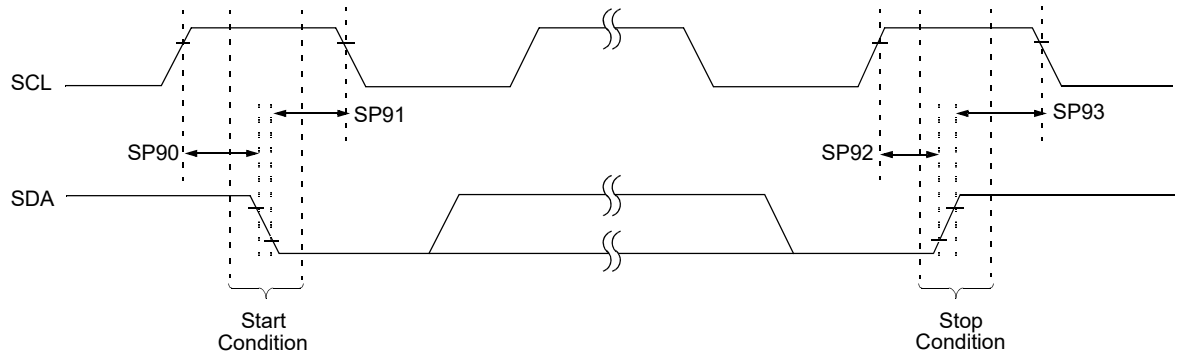
Standard Operating Conditions (unless otherwise stated)							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
	TSCK	SCK Total Cycle Time	47	—	—	ns	Receive only mode
			—	20(1)	—	MHz	
			95	—	—	ns	Full duplex mode
			—	10(1)	—	MHz	
SP70*	TSSL2SCH, TSSL2SCL	$\overline{SS}$↓ to SCK↓ or SCK↑ input	0	—	—	ns	CKE = 0
			25	—	—	ns	CKE = 1
SP71*	TSCH	SCK input high time	20	—	—	ns	
SP72*	TSCL	SCK input low time	20	—	—	ns	
SP73*	TDIV2SCH, TDIV2SCL	Setup time of SDI data input to SCK edge	10	—	—	ns	
SP74*	TSCH2DIL, TSCL2DIL	Hold time of SDI data input to SCK edge	0	—	—	ns	
SP75*	TDOR	SDO data output rise time	—	10	25	ns	CL = 50 pF
SP76*	TDOF	SDO data output fall time	—	10	25	ns	CL = 50 pF
SP77*	TSSH2DOZ	$\overline{SS}$↑ to SDO output high-impedance	—	—	85	ns	
SP80*	TSCH2DOV, TSCL2DOV	SDO data output valid after SCK edge	—	—	85	ns	
SP82*	TSSL2DOV	SDO data output valid after $\overline{SS}$↓ edge	—	—	85	ns	
SP83*	TSCH2SSH, TSCL2SSH	$\overline{SS}$ ↑ after SCK edge	20	—	—	ns	
SP84*	TSSH2SSL	$\overline{SS}$↑ to $\overline{SS}$↓ edge	47	—	—	ns	

* 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: SPIxCON1.SMP bit must be set and the slew rate control must be disabled on the clock and data pins (clear the corresponding bits in SLRCONx register) for SPI to operate over 4 MHz.

FIGURE 44-18: I²C BUS START/STOP BITS TIMING



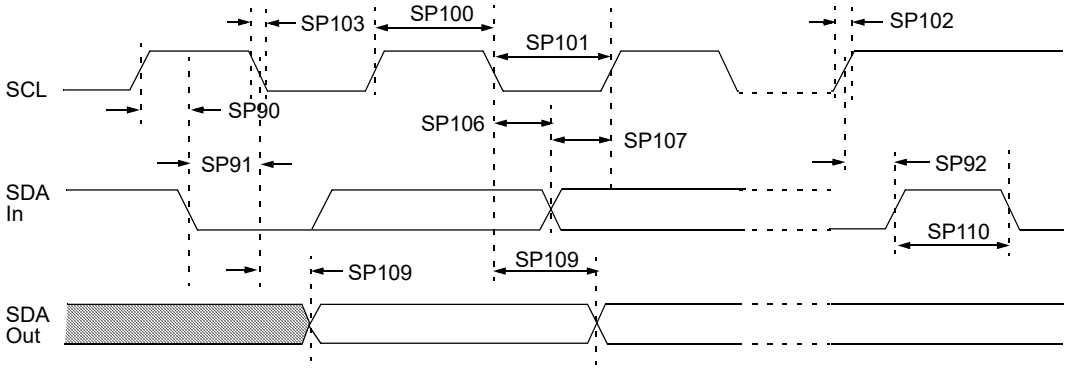
Note: Refer to Figure 44-4 for load conditions.

TABLE 44-24: I²C BUS START/STOP BITS REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)								
Param No.	**Symbol**	**Characteristic**		**Min.**	**Typ**	**Max.**	**Units**	**Conditions**
SP90*	TSU:STA	Start condition Setup time	100 kHz mode	4700	—	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	600	—	—		
			1 MHz mode	260	—	—		
SP91*	THD:STA	Start condition Hold time	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—	—		
			1 MHz mode	260	—	—		
SP92*	TSU:STO	Stop condition Setup time	100 kHz mode	4000	—	—	ns	
			400 kHz mode	600	—	—		
			1 MHz mode	260	—	—		
SP93	THD:STO	Stop condition Hold time	100 kHz mode	4700	—	—	ns	
			400 kHz mode	1300	—	—		
			1 MHz mode	500	—	—		

* These parameters are characterized but not tested.

FIGURE 44-19: I²C BUS DATA TIMING



Note: Refer to Figure 44-4 for load conditions.

TABLE 44-25: I²C BUS DATA REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Symbol**	**Characteristic**		**Min.**	**Max.**	**Units**	**Conditions**
SP100*	THIGH	Clock high time	100 kHz mode	4000	—	ns	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	600	—	ns	Device must operate at a minimum of 10 MHz
			1 MHz mode	260	—	ns	Device must operate at a minimum of 10 MHz
SP101*	TLOW	Clock low time	100 kHz mode	4700	—	ns	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	1300	—	ns	Device must operate at a minimum of 10 MHz
			1 MHz mode	500	—	—	Device must operate at a minimum of 10 MHz
SP102*	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20	300	ns	CB is specified to be from 10-400 pF
			1 MHz mode	—	120	ns	
SP103*	TF	SDA and SCL fall time	100 kHz mode	—	250	ns	
			400 kHz mode	20 X (VDD/5.5V)	250	ns	CB is specified to be from 10-400 pF
			1 MHz mode	20 X (VDD/5.5V)	120	ns	
SP106*	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	—	ns	
			1 MHz mode	0	—	ns	
SP107*	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	(2)
			400 kHz mode	100	—	ns	
			1 MHz mode	50	—	ns	
SP109*	TAA	Output valid from clock	100 kHz mode	—	3450	ns	(1)
			400 kHz mode	—	900	ns	
			1 MHz mode	—	450	ns	
SP110*	TBUF	Bus free time	100 kHz mode	4700	—	ns	Time the bus must be free before a new transmission can start
			400 kHz mode	1300	—	ns	
			1 MHz mode	500	—	ns	
SP111	CB	Bus capacitive loading		—	400	pF	

* These parameters are characterized but not tested.

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

2: A Fast mode (400 kHz) I²C bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement TSU:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification), before the SCL line is released.

TABLE 44-26: TEMPERATURE INDICATOR REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)								
Param No.	**Symbol**	**Characteristic**		**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
TS01*	TACQMIN	Minimum ADC Acquisition Time Delay		—	25	—	μs	
TS02*	MV	Voltage Sensitivity	High Range	—	-3.684	—	mV/°C	TSRNG = 1
			Low Range	—	-2.456	—	mV/°C	TSRNG = 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.

45.0 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 VDD range). This is for **information only** and devices are ensured to operate properly only within the specified range.

Unless otherwise noted, all graphs apply to both the L and LF devices.

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.

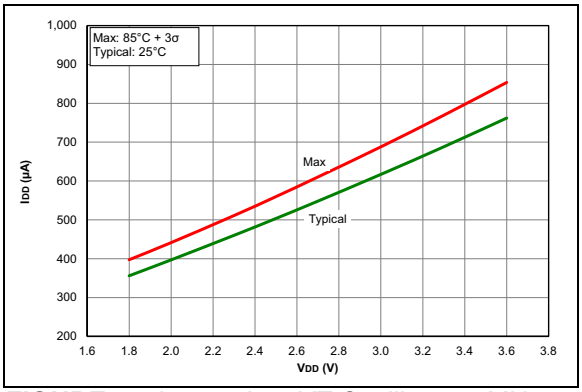


FIGURE 45-1: *IDD, XT Oscillator, 4 MHz, PIC18LF26/45/46/55/56K42 Only.*

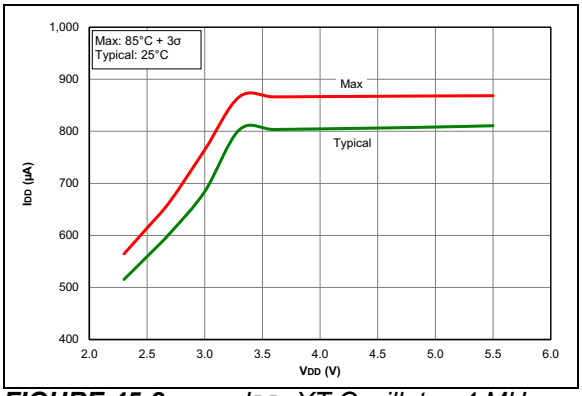


FIGURE 45-2: *IDD, XT Oscillator, 4 MHz, PIC18F26/45/46/55/56K42 Only.*

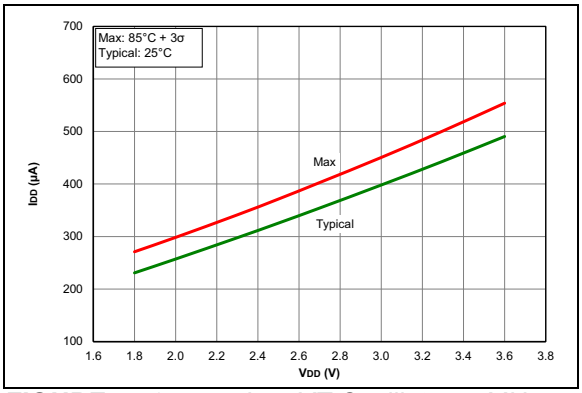


FIGURE 45-3: *IDD, XT Oscillator, 4 MHz, PMD's All '1's, PIC18LF26/45/46/55/56K42 Only.*

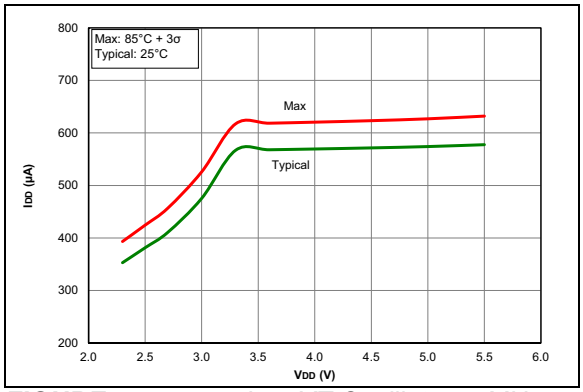


FIGURE 45-4: *IDD, XT Oscillator, 4 MHz, PMD's All '1's, PIC18F26/45/46/55/56K42 Only.*

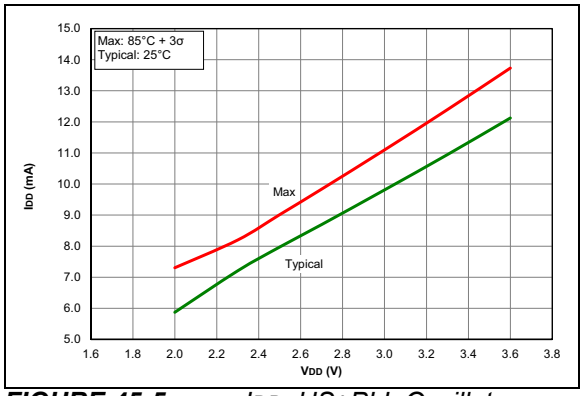


FIGURE 45-5: *IDD, HS+PLL Oscillator, 64 MHz, PIC18LF26/45/46/55/56K42 Only.*

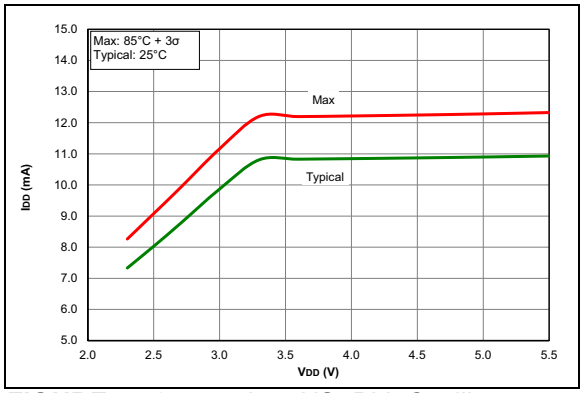


FIGURE 45-6: *IDD, HS+PLL Oscillator, 64 MHz, PIC18F26/45/46/55/56K42 Only.*

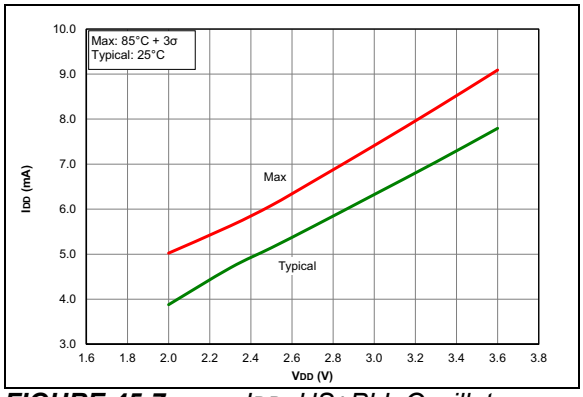


FIGURE 45-7: *IDD, HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18LF26/45/46/55/56K42 Only.*

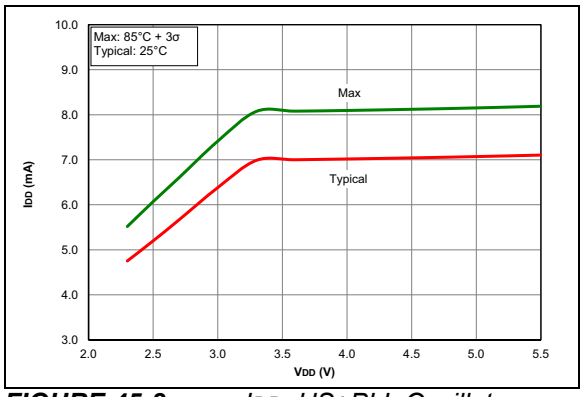


FIGURE 45-8: *IDD, HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18F26/45/46/55/56K42 Only.*

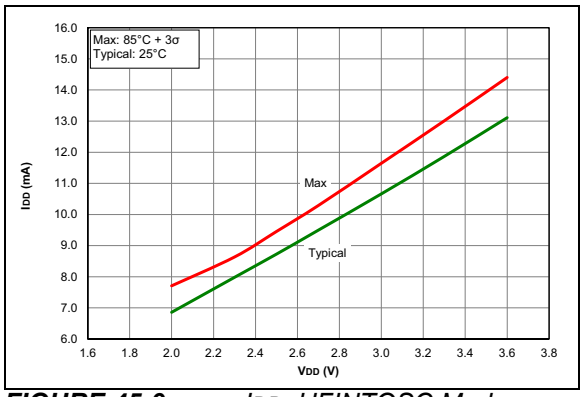


FIGURE 45-9: *IDD, HFINTOSC Mode, Fosc = 64 MHz, PIC18LF26/45/46/55/56K42 Only.*

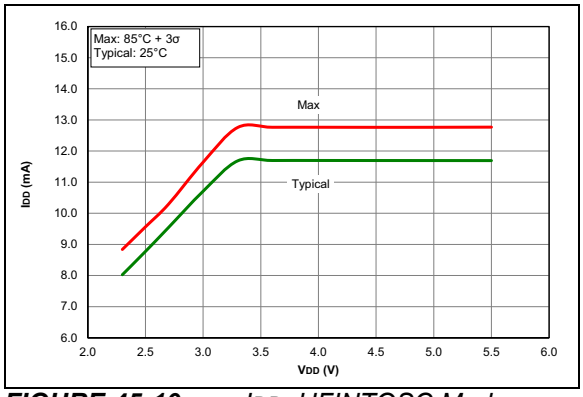


FIGURE 45-10: *IDD, HFINTOSC Mode, Fosc = 64 MHz, PIC18F26/45/46/55/56K42 Only.*

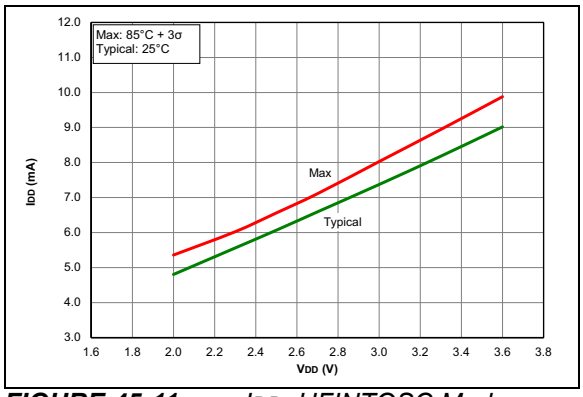


FIGURE 45-11: *IDD, HFINTOSC Mode, Fosc = 64 MHz, PMD's All '1's, PIC18LF26/45/46/55/56K42 Only.*

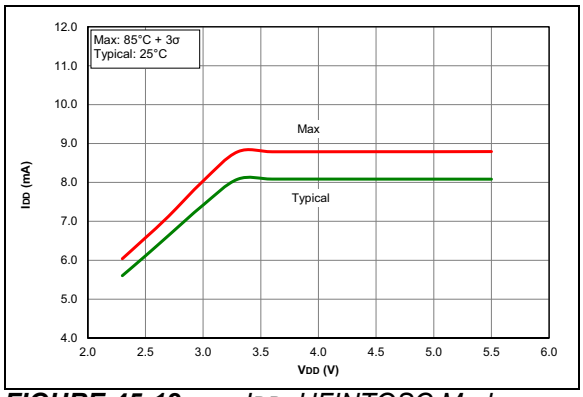


FIGURE 45-12: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PIC18F26/45/46/55/56K42 Only.*

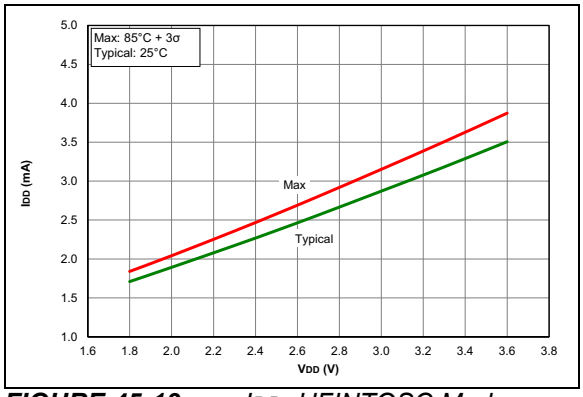


FIGURE 45-13: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PIC18LF26/45/46/55/56K42 Only.*

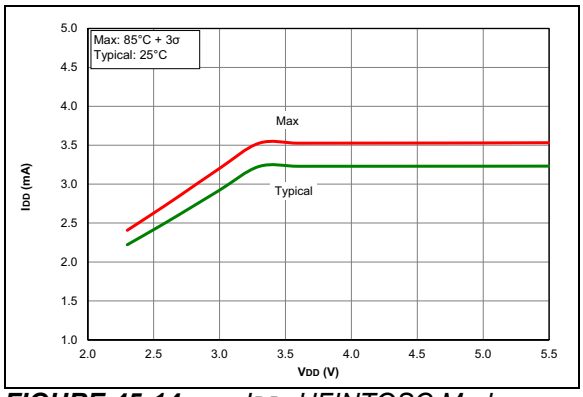


FIGURE 45-14: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PIC18F26/45/46/55/56K42 Only.*

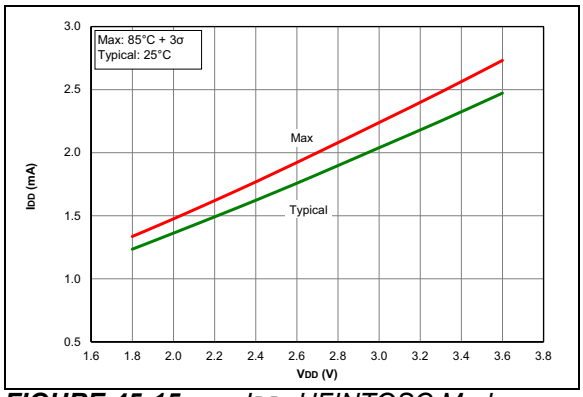


FIGURE 45-15: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PMD's All '1's, PIC18LF26/45/46/55/56K42 Only.*

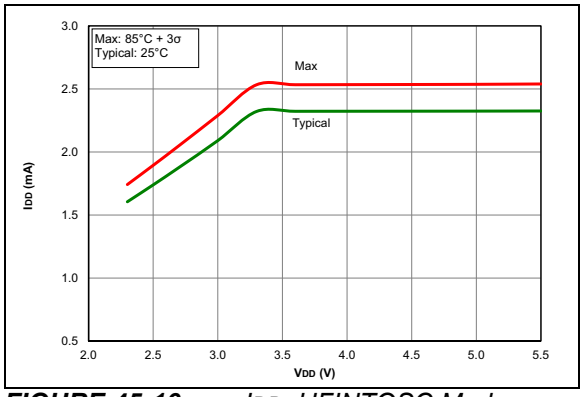


FIGURE 45-16: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PMD's All '1's, PIC18F26/45/46/55/56K42 Only.*

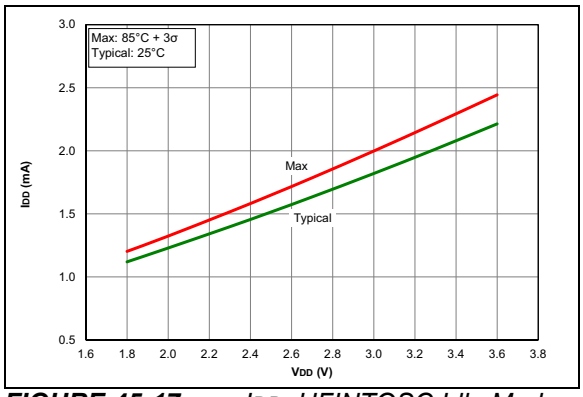


FIGURE 45-17: *IDD, HFINTOSC Idle Mode, Fosc = 16 MHz, PIC18LF26/45/46/55/56K42 Only.*

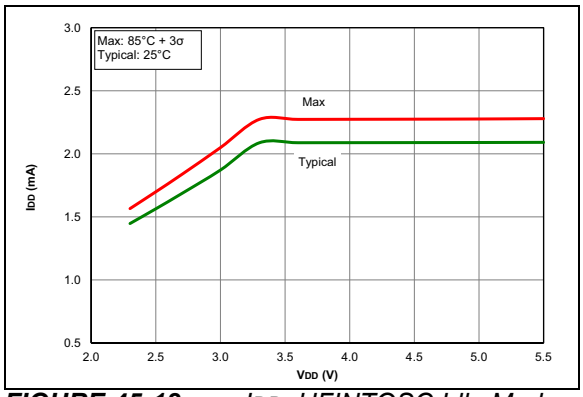


FIGURE 45-18: *IDD, HFINTOSC Idle Mode, Fosc = 16 MHz, PIC18F26/45/46/55/56K42 Only.*

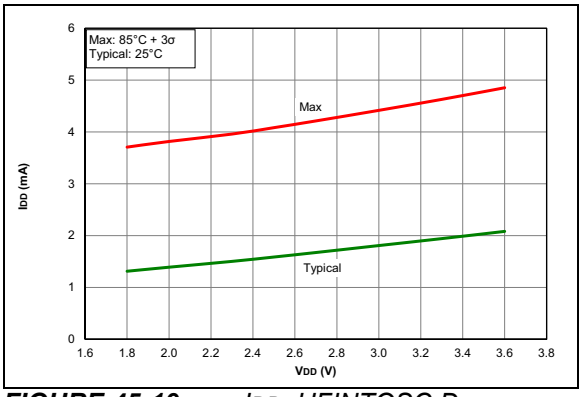


FIGURE 45-19: *IDD, HFINTOSC Doze Mode, Fosc = 16 MHz, PIC18LF26/45/46/55/56K42 Only.*

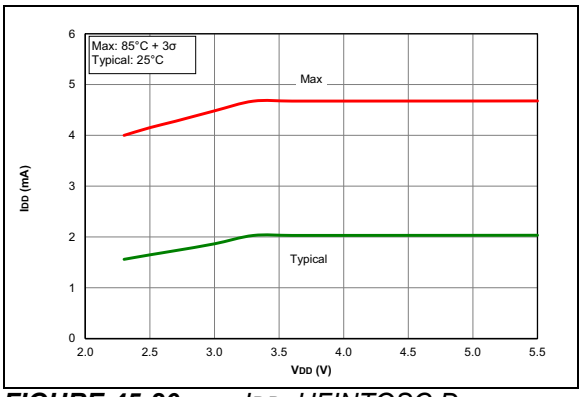


FIGURE 45-20: *IDD, HFINTOSC Doze Mode, Fosc = 16 MHz, PIC18F26/45/46/55/56K42 Only.*

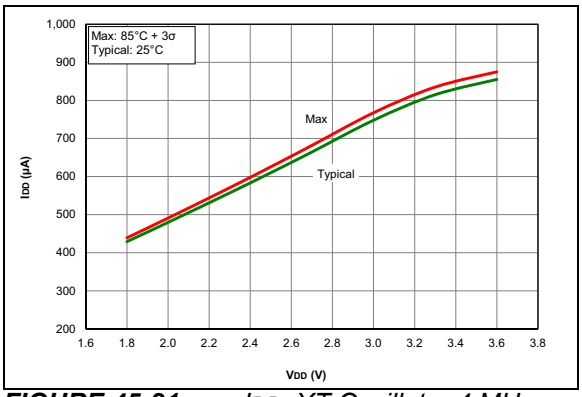


FIGURE 45-21: *IDD, XT Oscillator 4 MHz, PIC18LF27/47/57K42 Only*

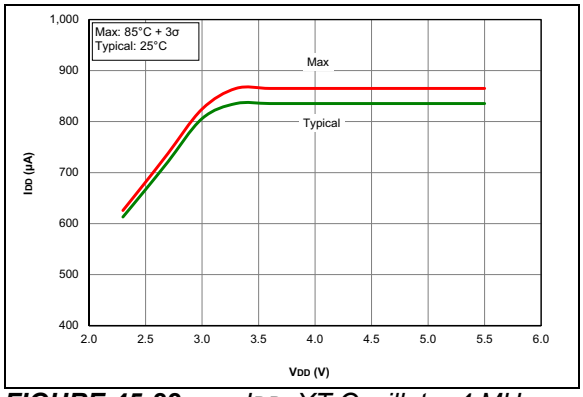


FIGURE 45-22: *IDD, XT Oscillator 4 MHz, PIC18F27/47/57K42 Only*

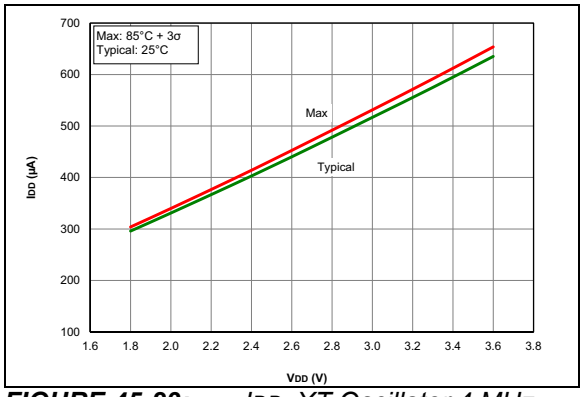


FIGURE 45-23: *IDD, XT Oscillator 4 MHz, PMD's All '1's, PIC18LF27/47/57K42 Only*

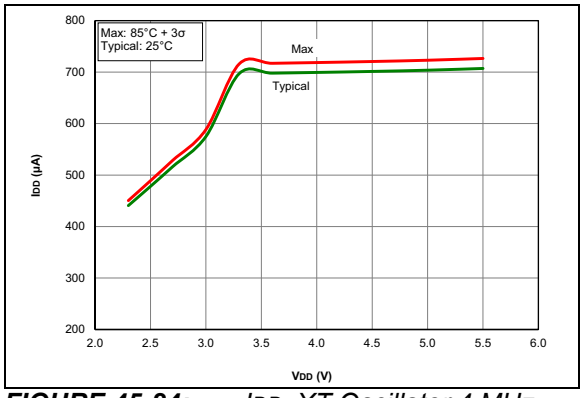


FIGURE 45-24: *IDD, XT Oscillator 4 MHz, PMD's All '1's, PIC18F27/47/57K42 Only*

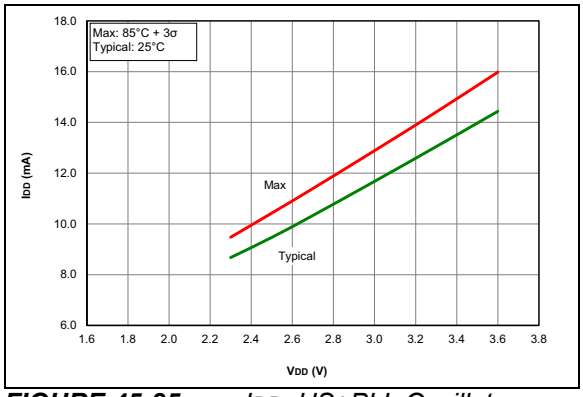


FIGURE 45-25: *IDD, HS+PLL Oscillator, 64 MHz, PIC18LF27/47/57K42 Only.*

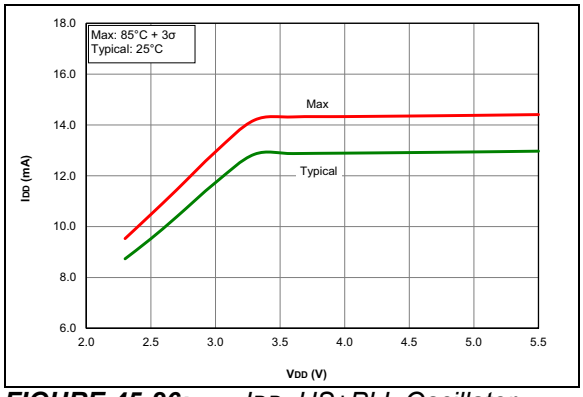


FIGURE 45-26: *IDD, HS+PLL Oscillator, 64 MHz, PIC18F27/47/57K42 Only.*

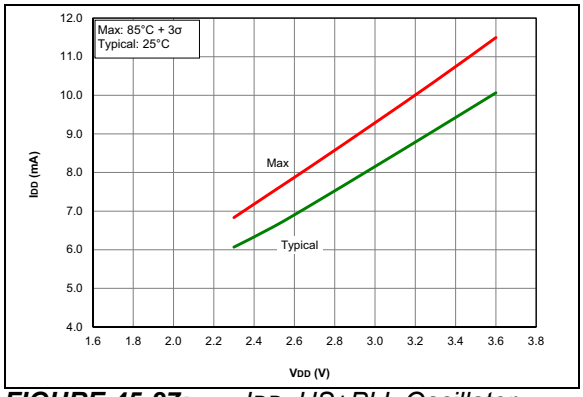


FIGURE 45-27: *IDD, HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18LF27/47/57K42 Only.*

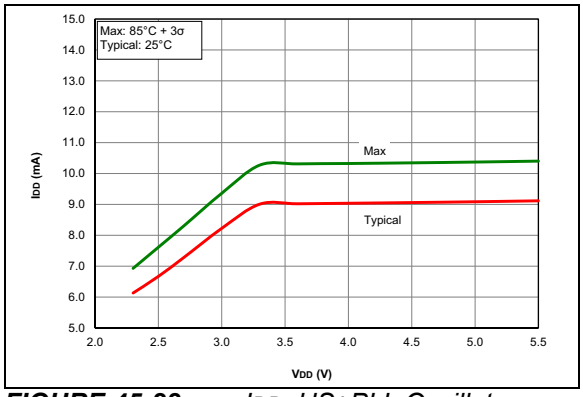


FIGURE 45-28: *IDD, HS+PLL Oscillator, 64 MHz, PMD's All '1's, PIC18F27/47/57K42 Only.*

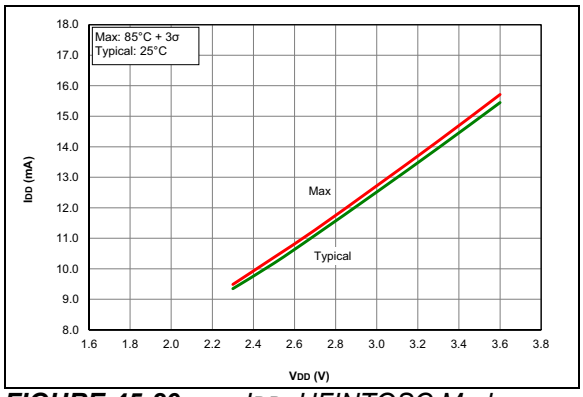


FIGURE 45-29: *IDD, HFINTOSC Mode, FOSC = 64 MHz, PIC18LF27/47/57K42 Only.*

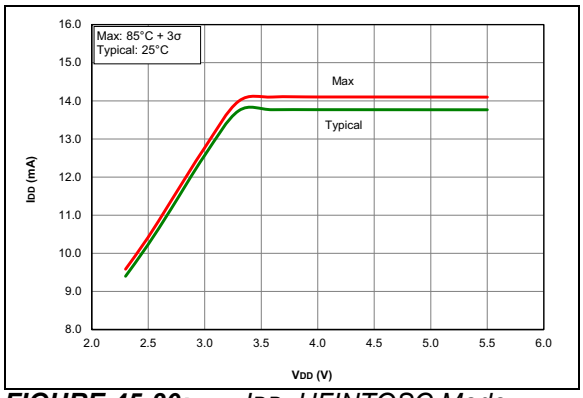


FIGURE 45-30: *IDD, HFINTOSC Mode, FOSC = 64 MHz, PIC18F27/47/57K42 Only.*

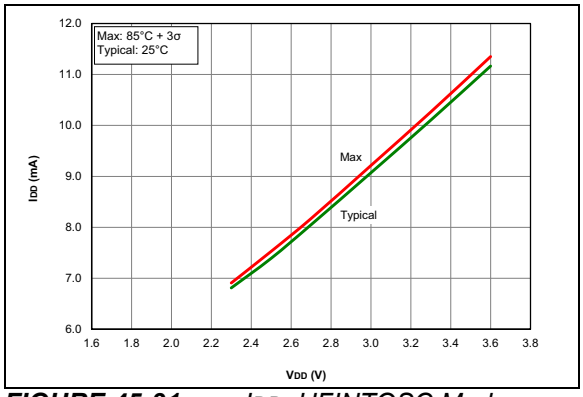


FIGURE 45-31: *IDD, HFINTOSC Mode, Fosc = 64 MHz, PMD's All '1's, PIC18LF27/47/57K42 Only.*

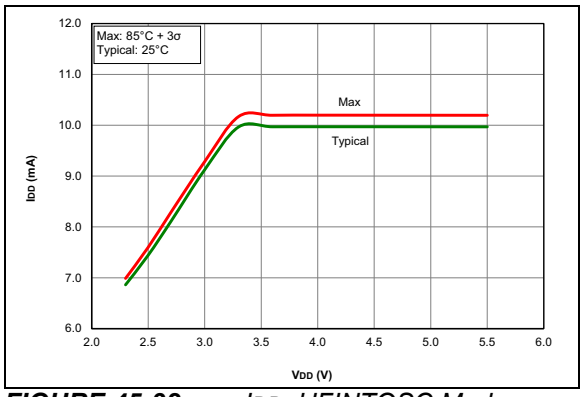


FIGURE 45-32: *IDD, HFINTOSC Mode, Fosc = 64 MHz, PMD's All '1's, PIC18F27/47/57K42 Only.*

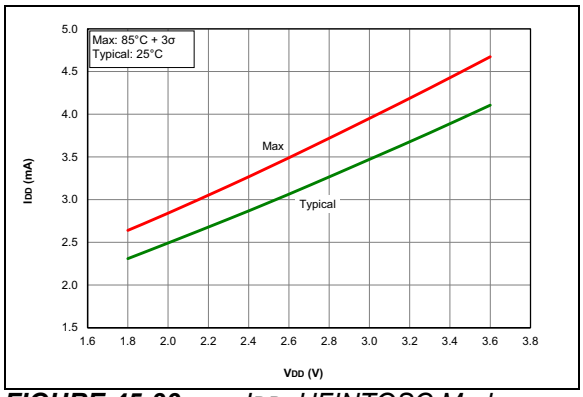


FIGURE 45-33: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PIC18LF27/47/57K42 Only.*

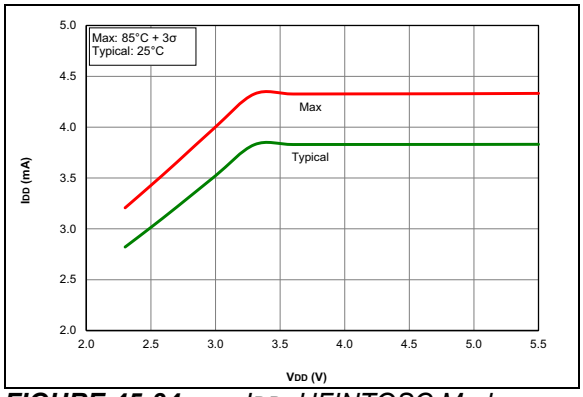


FIGURE 45-34: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PIC18F27/47/57K42 Only.*

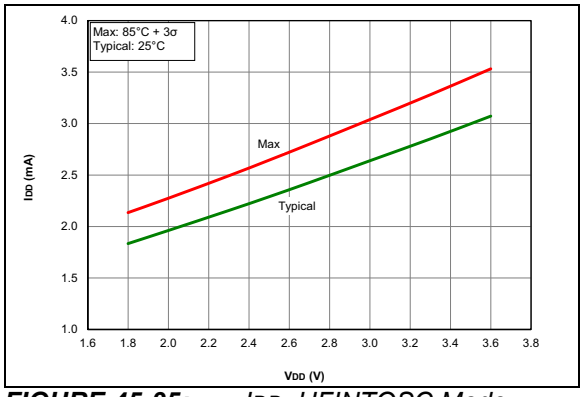


FIGURE 45-35: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PMD's All '1's, PIC18LF27/47/57K42 Only.*

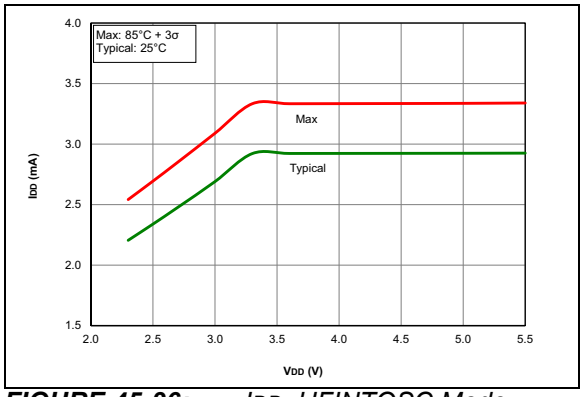


FIGURE 45-36: *IDD, HFINTOSC Mode, Fosc = 16 MHz, PMD's All '1's, PIC18F27/47/57K42 Only.*

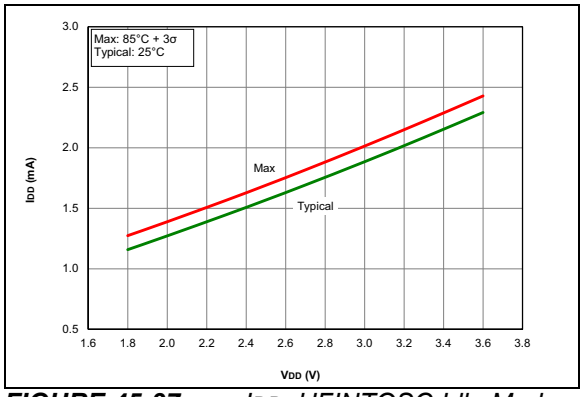


FIGURE 45-37: *IDD, HFINTOSC Idle Mode, Fosc = 16 MHz, PIC18LF27/47/57K42 Only.*

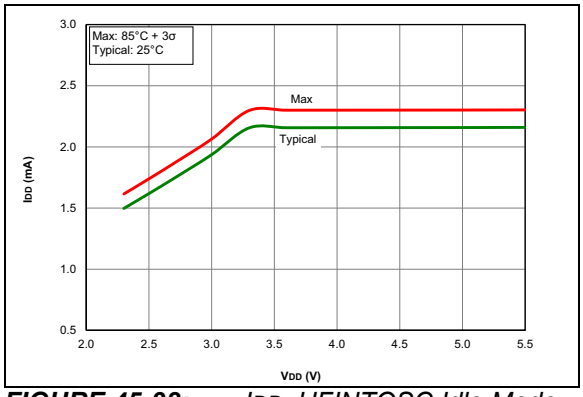


FIGURE 45-38: *IDD, HFINTOSC Idle Mode, Fosc = 16 MHz, PIC18F27/47/57K42 Only.*

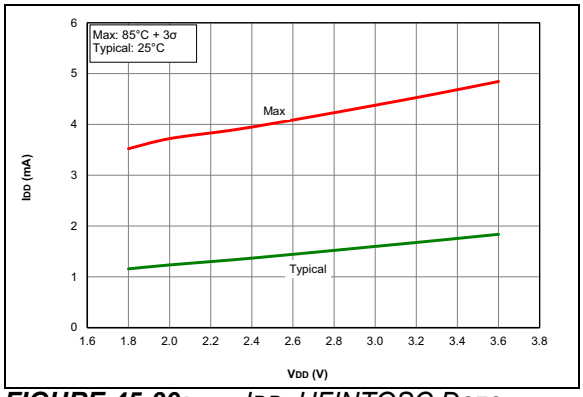


FIGURE 45-39: *IDD, HFINTOSC Doze Mode, Fosc = 16 MHz, PIC18LF27/47/57K42 Only.*

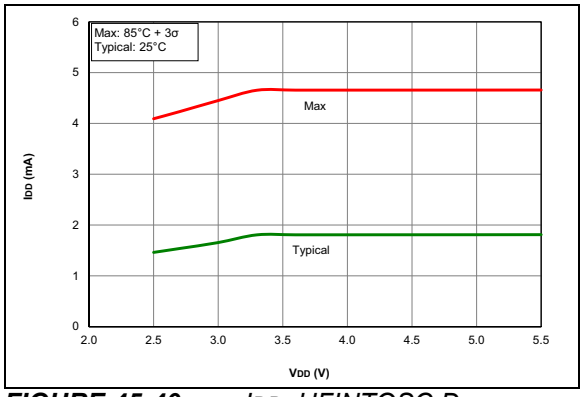


FIGURE 45-40: *IDD, HFINTOSC Doze Mode, Fosc = 16 MHz, PIC18F27/47/57K42 Only.*

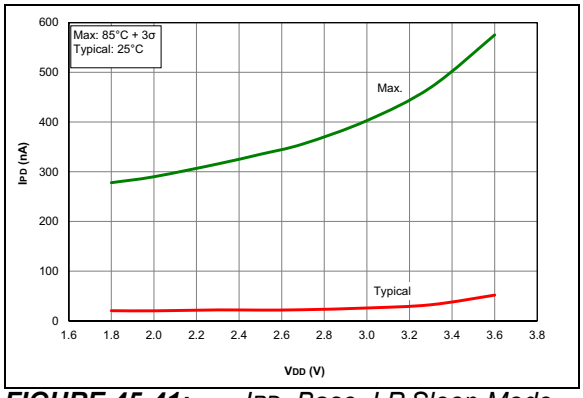


FIGURE 45-41: *IPD, Base, LP Sleep Mode, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

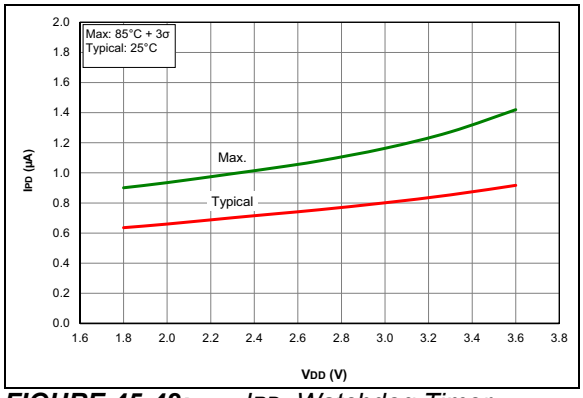


FIGURE 45-42: *IPD, Watchdog Timer (WDT), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

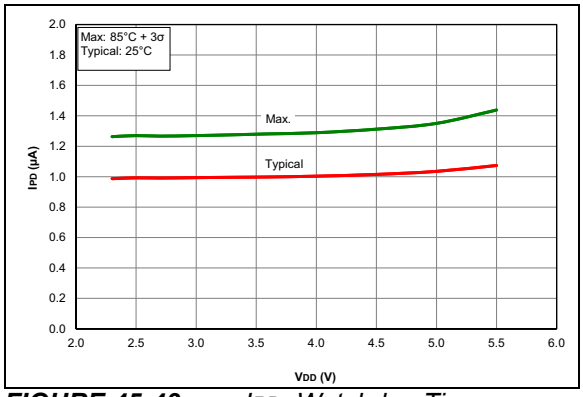


FIGURE 45-43: *IPD, Watchdog Timer (WDT), PIC18F26/27/45/46/47/55/56/57K42 Only.*

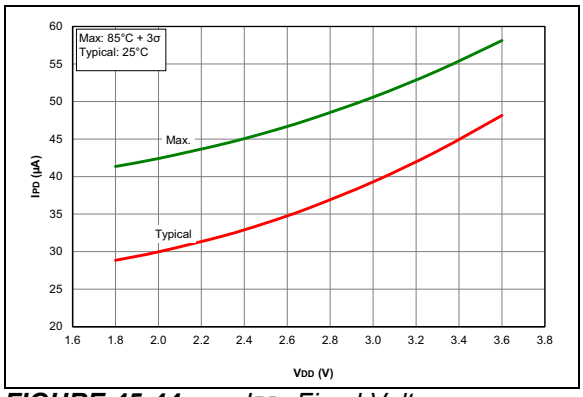


FIGURE 45-44: *IPD, Fixed Voltage Reference (FVR), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

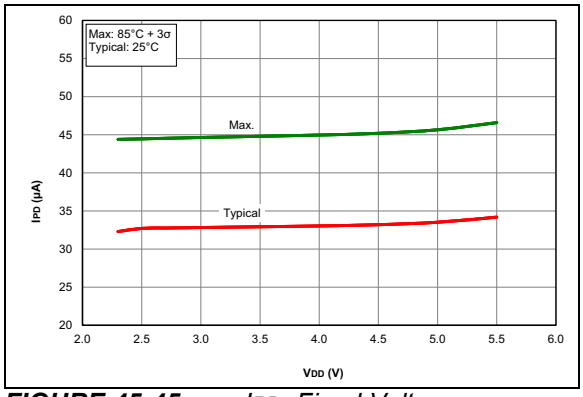


FIGURE 45-45: *IPD, Fixed Voltage Reference (FVR), PIC18F26/27/45/46/47/55/56/57K42 Only.*

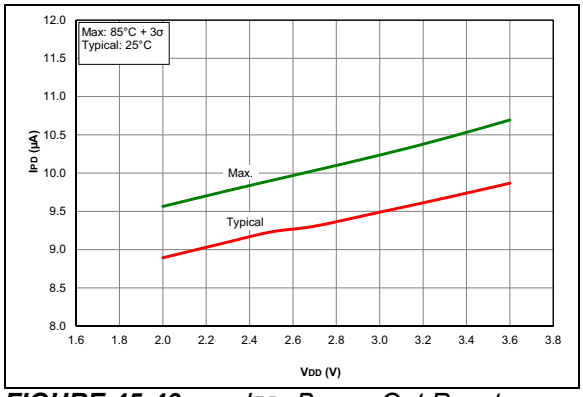


FIGURE 45-46: *IPD, Brown-Out Reset (BOR), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

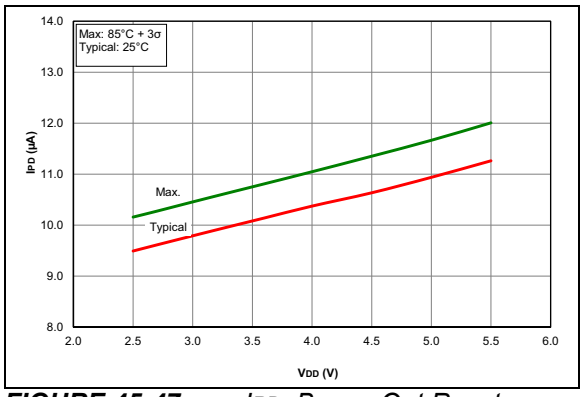


FIGURE 45-47: *IPD, Brown-Out Reset (BOR), PIC18F26/27/45/46/47/55/56/57K42 Only.*

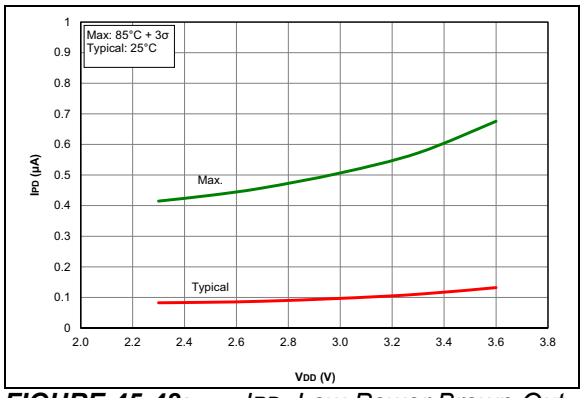


FIGURE 45-48: *IPD, Low-Power Brown-Out Reset (LPBOR), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

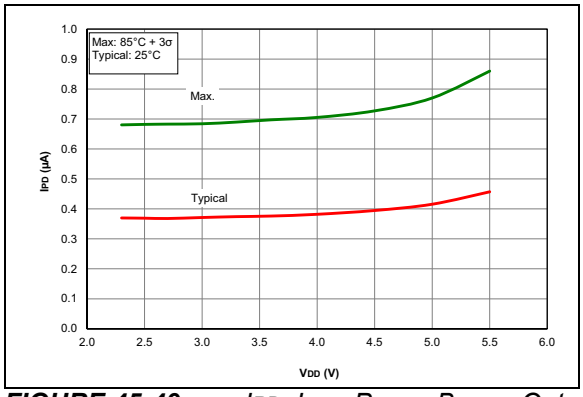


FIGURE 45-49: *IPD, Low-Power Brown-Out Reset (LPBOR), PIC18F26/27/45/46/47/55/56/57K42 Only.*

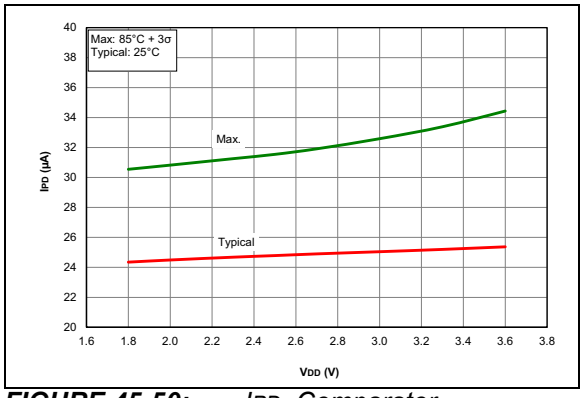


FIGURE 45-50: *IPD, Comparator, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

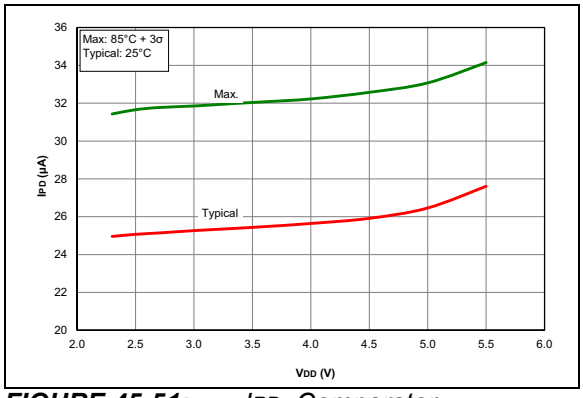


FIGURE 45-51: *IPD, Comparator, PIC18F26/27/45/46/47/55/56/57K42 Only.*

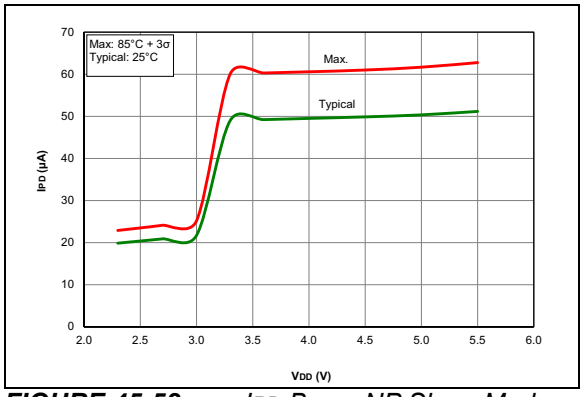


FIGURE 45-52: *IPD Base, NP Sleep Mode, PIC18F26/27/45/46/47/55/56/57K42 Only.*

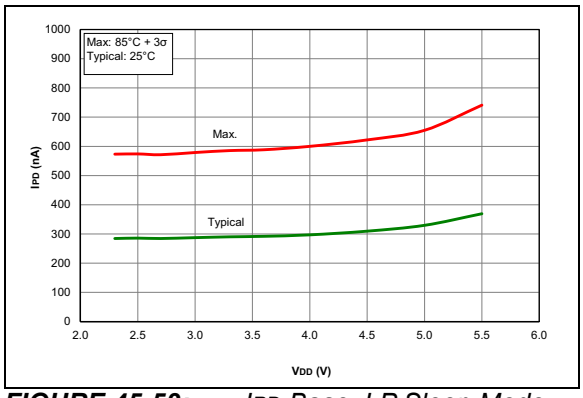


FIGURE 45-53: *IPD Base, LP Sleep Mode, PIC18F26/27/45/46/47/55/56/57K42 Only*

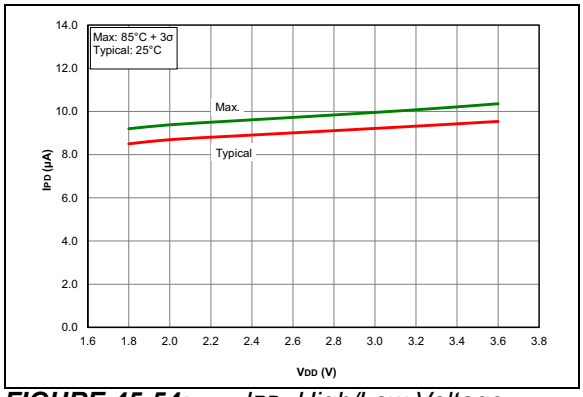


FIGURE 45-54: *IPD, High/Low Voltage detect (HLVD), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

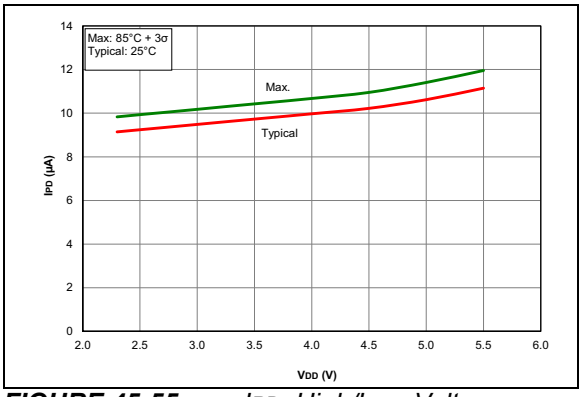


FIGURE 45-55: *IPD, High/Low Voltage detect (HLVD), PIC18F26/27/45/46/47/55/56/57K42 Only.*

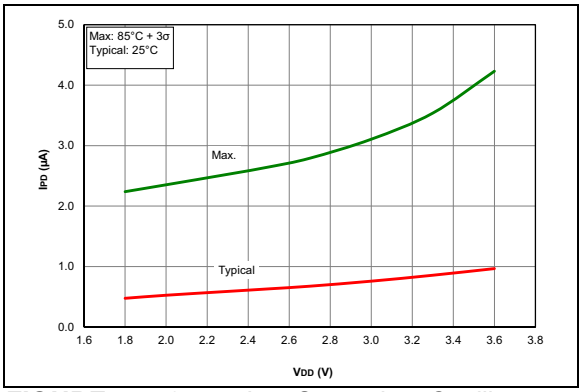


FIGURE 45-56: *IPD, Secondary Oscillator (SOSC), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

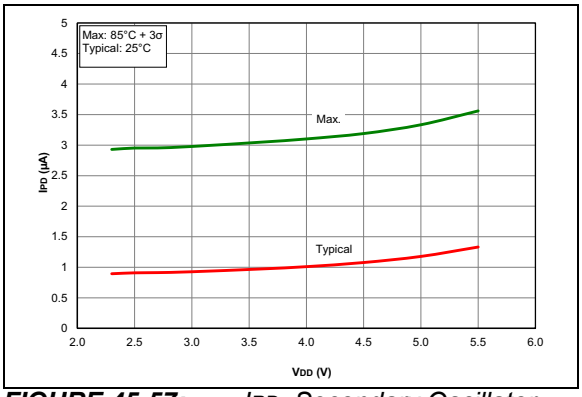


FIGURE 45-57: *IPD, Secondary Oscillator (SOSC), PIC18F26/27/45/46/47/55/56/57K42 Only.*

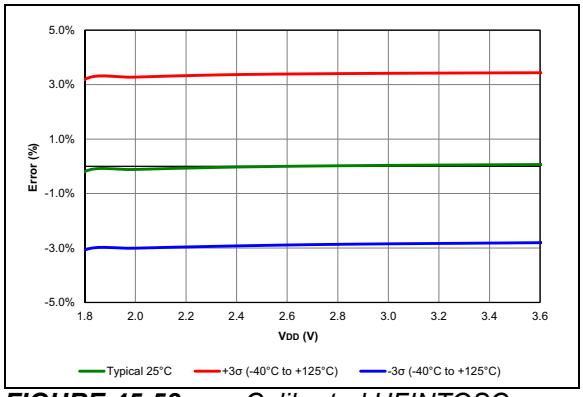


FIGURE 45-58: *Calibrated HFINTOSC, Typical Frequency Error, PIC18LF26/27/45/46/47/55/56/57K42 only.*

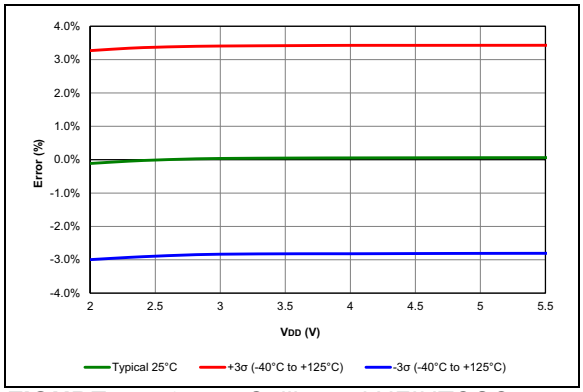


FIGURE 45-59: *Calibrated HFINTOSC, Typical Frequency Error, PIC18F26/27/45/46/47/55/56/57K42 only.*

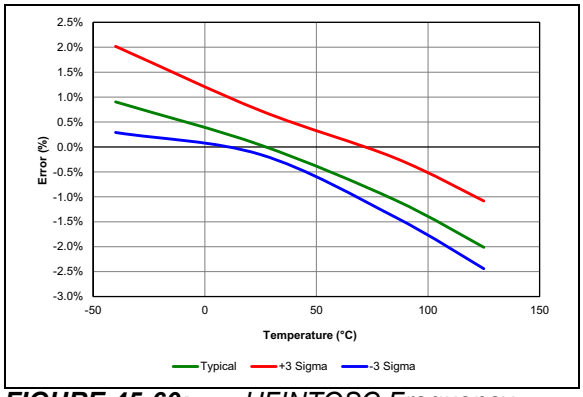


FIGURE 45-60: *HFINTOSC Frequency Error, VDD = 3.0V.*

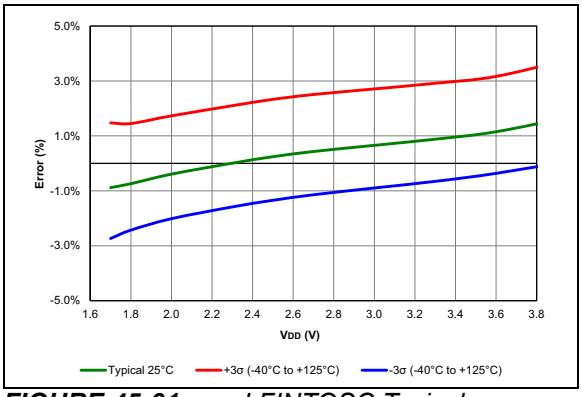


FIGURE 45-61: *LFINTOSC Typical Frequency Error, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

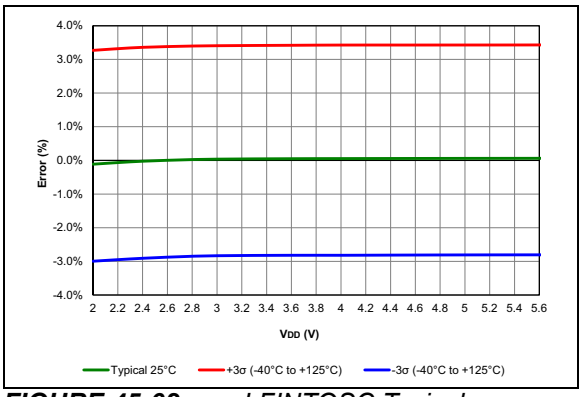


FIGURE 45-62: *LFINTOSC Typical Frequency Error, PIC18F26/27/45/46/47/55/56/57K42 Only.*

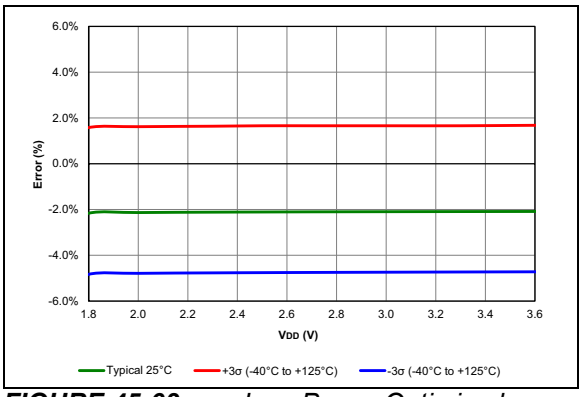


FIGURE 45-63: *Low-Power Optimized HFINTOSC Typical Frequency Error, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

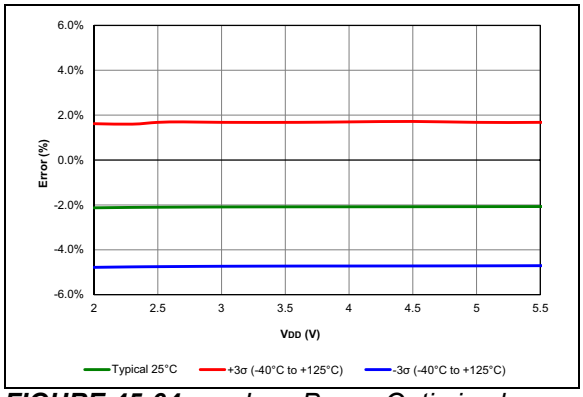


FIGURE 45-64: *Low-Power Optimized HFINTOSC Typical Frequency Error, PIC18F26/27/45/46/47/55/56/57K42 Only.*

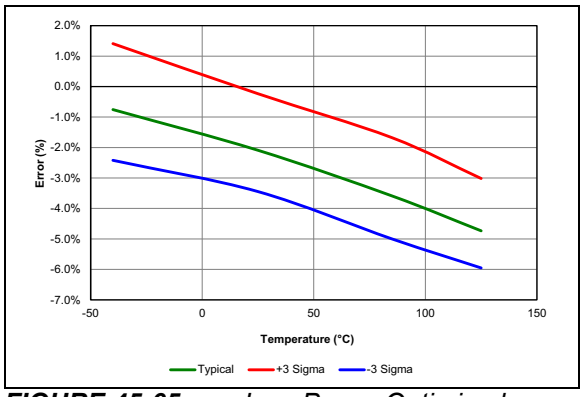


FIGURE 45-65: *Low-Power Optimized HFINTOSC Frequency Error, VDD = 3.0V.*

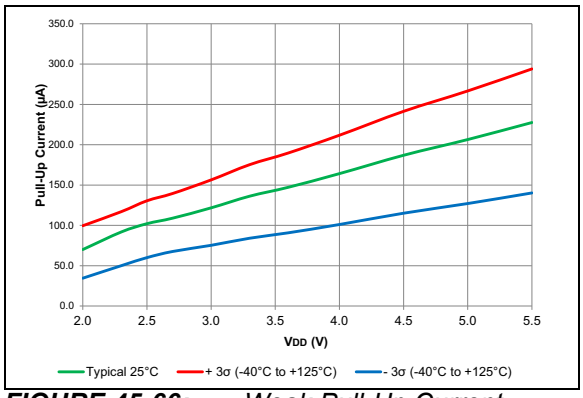


FIGURE 45-66: *Weak Pull-Up Current, PIC18F26/27/45/46/47/55/56/57K42 Only.*

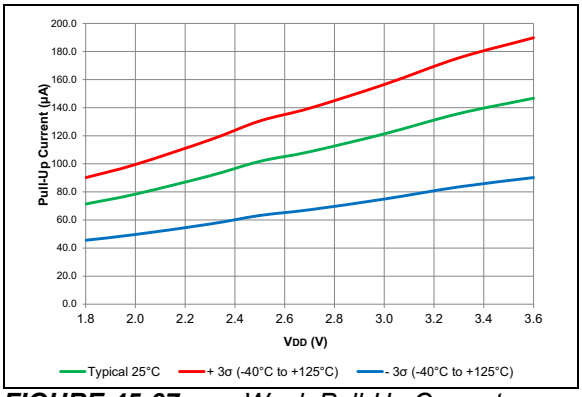


FIGURE 45-67: *Weak Pull-Up Current, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

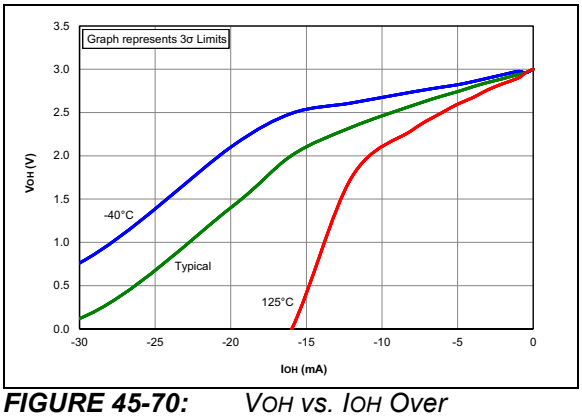


FIGURE 45-70: *VOH vs. IOH Over Temperature, VDD = 3.0V.*

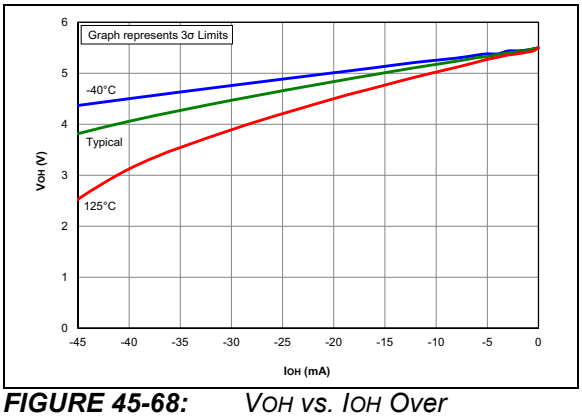


FIGURE 45-68: *VOH vs. IOH Over Temperature, VDD = 5.5V, PIC18F26/27/45/46/47/55/56/57K42 Only.*

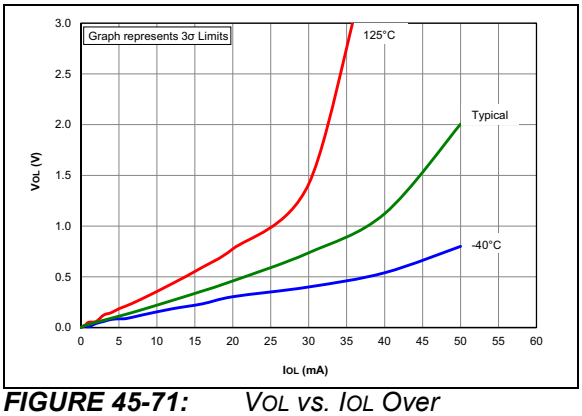


FIGURE 45-71: *VOL vs. IOL Over Temperature, VDD = 3.0V.*

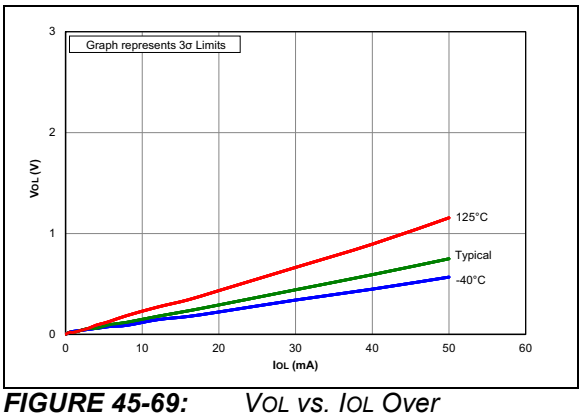


FIGURE 45-69: *VOL vs. IOL Over Temperature, VDD = 5.5V, PIC18F26/27/45/46/47/55/56/57K42 Only.*

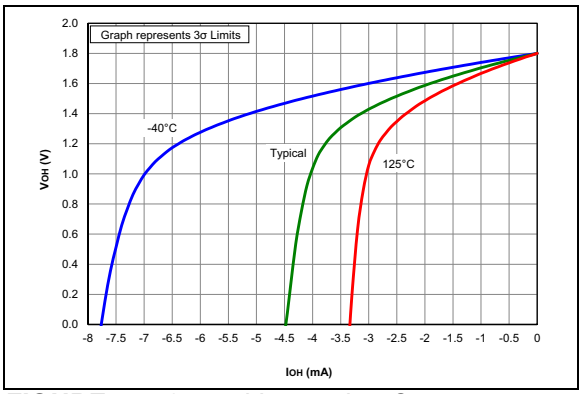


FIGURE 45-72: *VOH vs. IOH Over Temperature, VDD = 1.8V, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

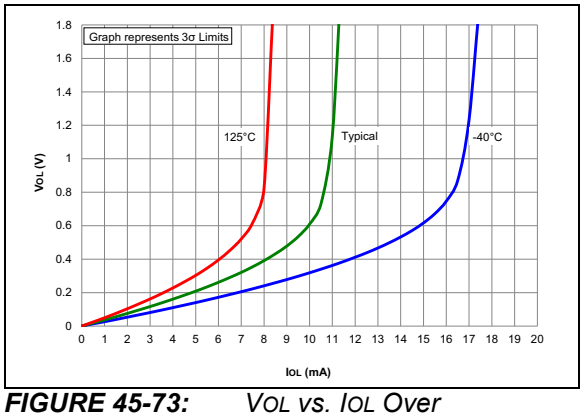


FIGURE 45-73: *VOL vs. IOL Over Temperature, VDD = 1.8V, PIC18LF26/27/45/46/47/55/56/57K42 Only*

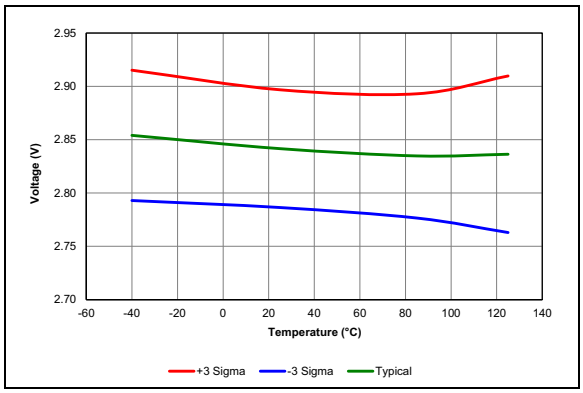


FIGURE 45-74: *Brown-Out Reset Voltage, Trip Point (BORV = 00).*

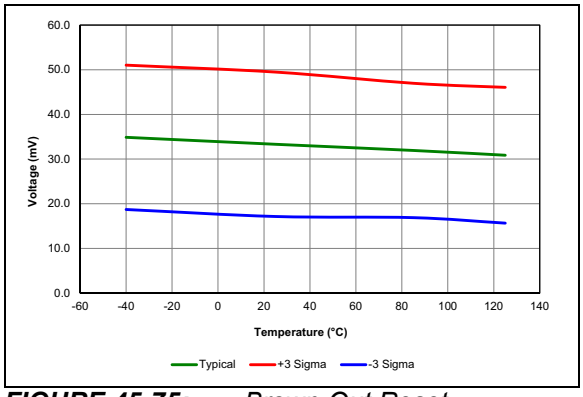


FIGURE 45-75: *Brown-Out Reset, Hysteresis, Trip Point (BORV = 00).*

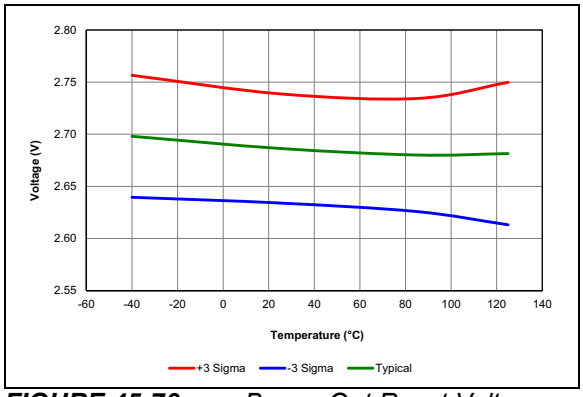


FIGURE 45-76: *Brown-Out Reset Voltage, Trip Point (BORV = 01).*

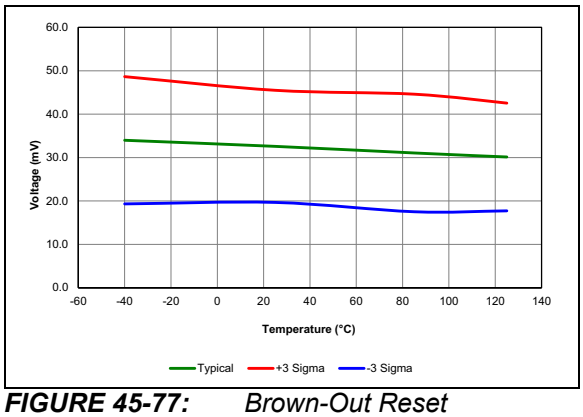


FIGURE 45-77: *Brown-Out Reset Hysteresis, Trip Point (BORV = 01).*

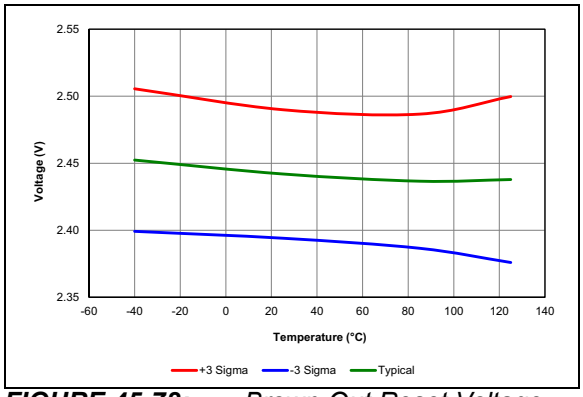


FIGURE 45-78: *Brown-Out Reset Voltage, Trip Point (BORV = 10).*

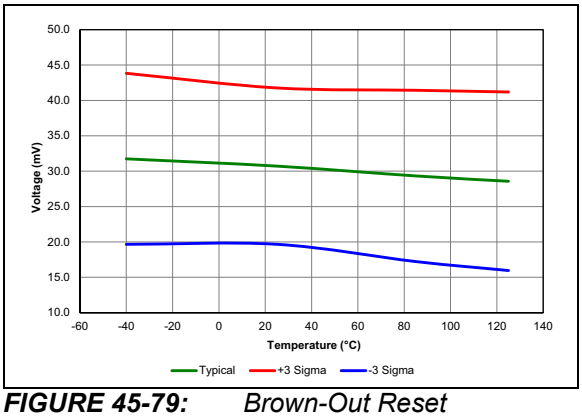


FIGURE 45-79: *Brown-Out Reset Hysteresis, Trip Point (BORV = 10).*

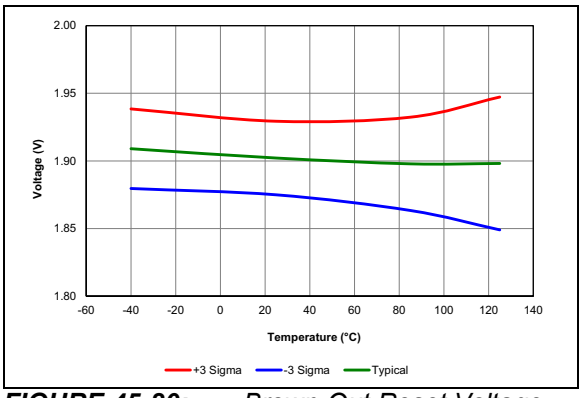


FIGURE 45-80: *Brown-Out Reset Voltage, Trip Point (BORV = 11), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

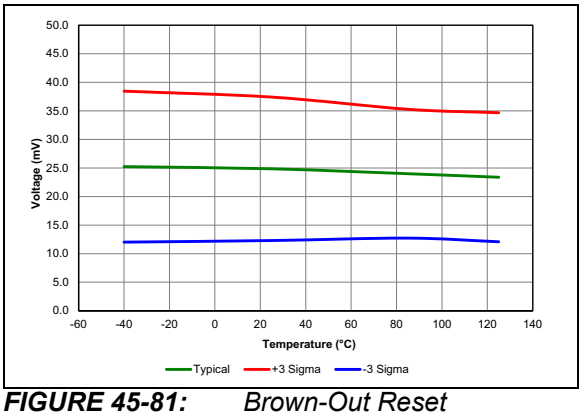


FIGURE 45-81: *Brown-Out Reset Hysteresis, Trip Point (BORV = 11), PIC18LF26/27/45/46/47/55/56/57K42 Only.*

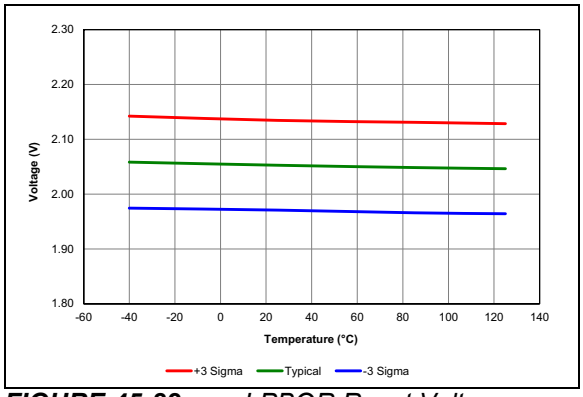


FIGURE 45-82: *LPBOR Reset Voltage.*

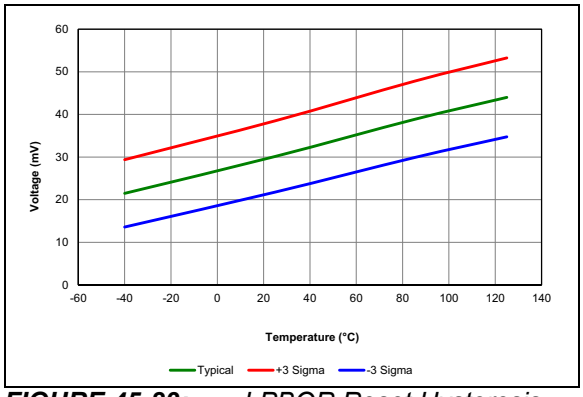


FIGURE 45-83: *LPBOR Reset Hysteresis.*

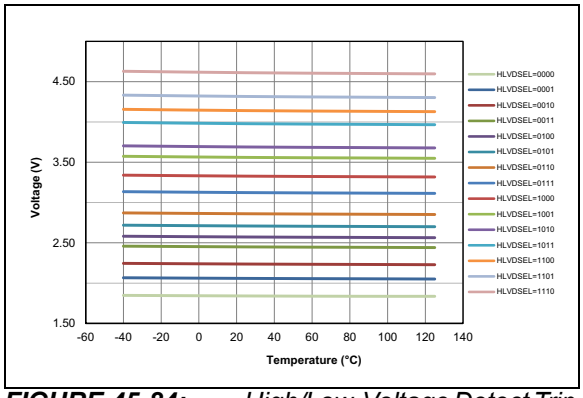


FIGURE 45-84: *High/Low-Voltage Detect Trip Voltage.*

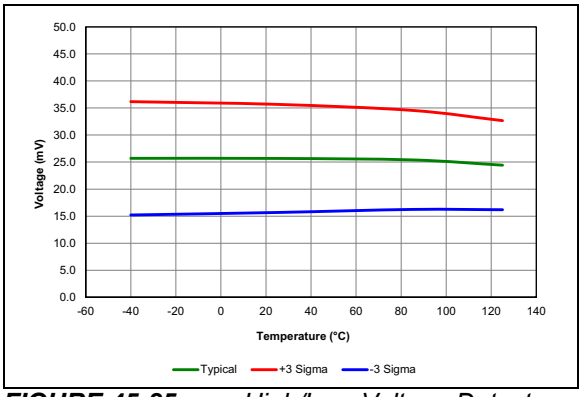


FIGURE 45-85: *High/Low-Voltage Detect Hysteresis.*

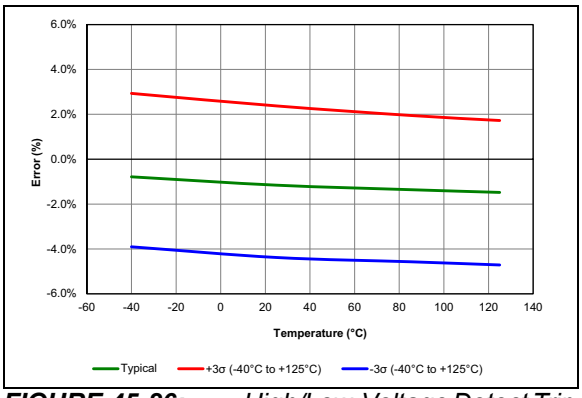


FIGURE 45-86: *High/Low-Voltage Detect Trip Voltage, Typical Error (HLVDSEL[3:0] = `0001`).*

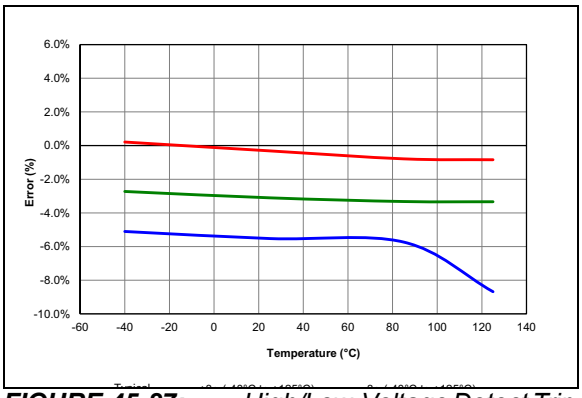


FIGURE 45-87: *High/Low-Voltage Detect Trip Voltage, Typical Error (HLVDSEL[3:0] = `0000`).*

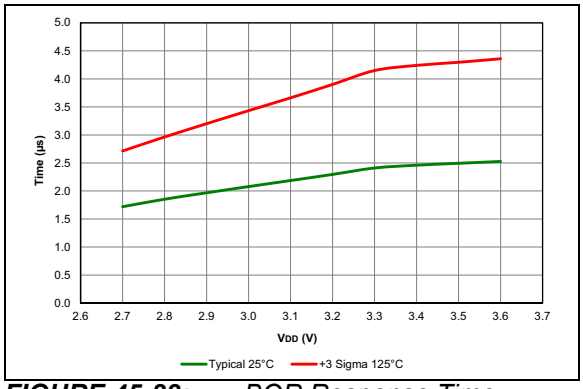


FIGURE 45-88: *BOR Response Time, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

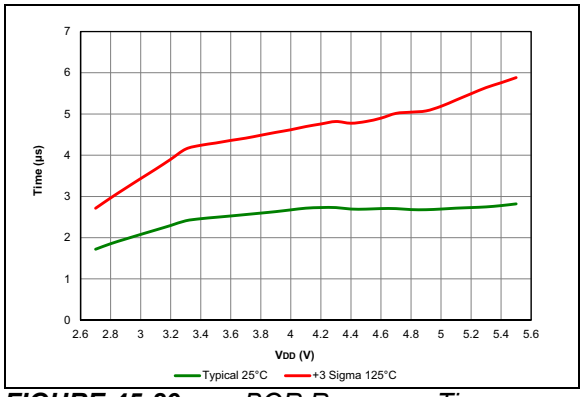


FIGURE 45-89: *BOR Response Time, PIC18F26/27/45/46/47/55/56/57K42 Only.*

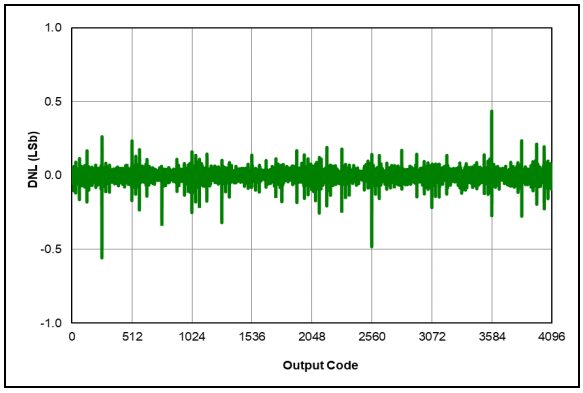


FIGURE 45-90: *ADC 12-Bit Mode, Single-Ended, Typical DNL, V_{DD} = 3.0V, V_{REF} = 3.0V, T_{AD} = 0.5 µS, 25°C, All devices.*

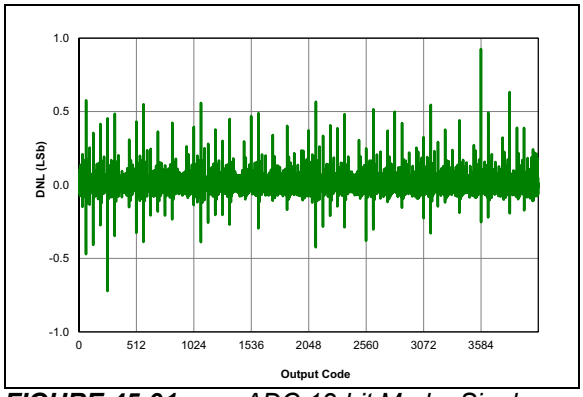


FIGURE 45-91: *ADC 12-bit Mode, Single-Ended DNL, V_{DD} = 3.0V, V_{REF} = 3.0V, T_{AD} = 1 μS, CP OFF, 25°C.*

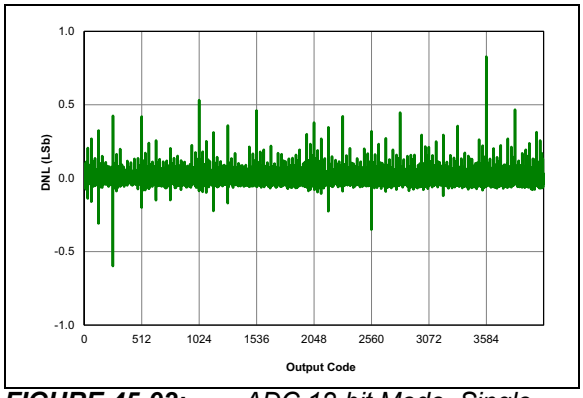


FIGURE 45-92: *ADC 12-bit Mode, Single-Ended DNL, V_{DD} = 3.0V, V_{REF} = 3.0V, T_{AD} = 1 μS, CP ON, 25°C.*

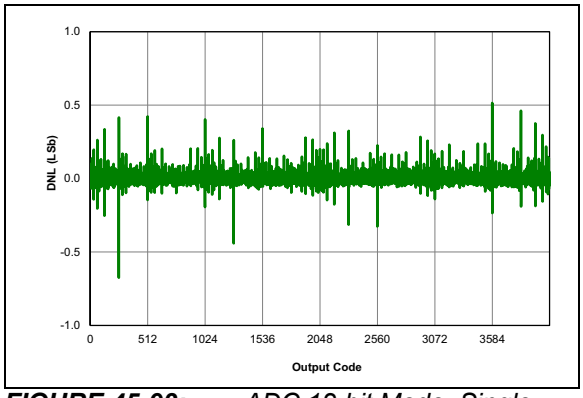


FIGURE 45-93: *ADC 12-bit Mode, Single-Ended DNL, V_{DD} = 2.3V, V_{REF} = 2.3V, T_{AD} = 1 μS, CP ON, 25°C.*

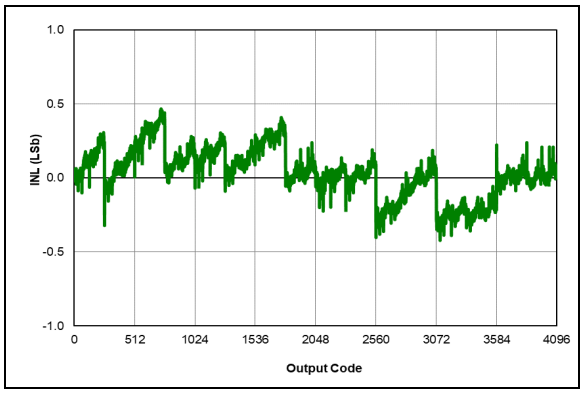


FIGURE 45-94: *ADC 12-bit Mode, Single-Ended, Typical INL, V_{DD} = 3.0V, V_{REF} = 3.0V, T_{AD} = 0.5 μS, 25°C, All devices.*

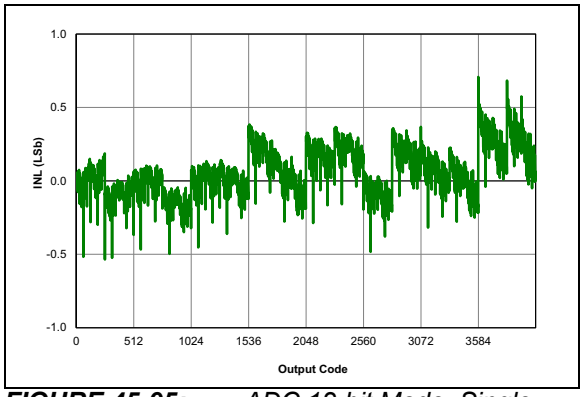


FIGURE 45-95: *ADC 12-bit Mode, Single-Ended INL, V_{DD} = 3.0V, V_{REF} = 3.0V, T_{AD} = 1 μS, CP OFF, 25°C.*

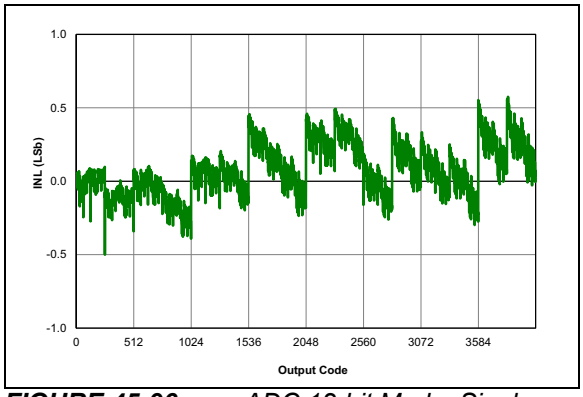


FIGURE 45-96: *ADC 12-bit Mode, Single-Ended INL, V_{DD} = 3.0V, V_{REF} = 3.0V, T_{AD} = 1 μS, CP ON, 25°C.*

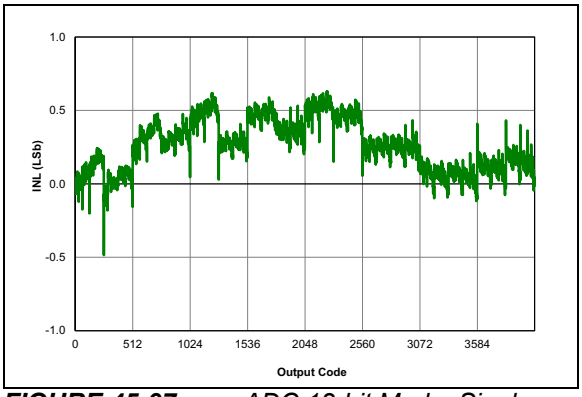


FIGURE 45-97: *ADC 12-bit Mode, Single-Ended INL, V_{DD} = 2.3V, V_{REF} = 2.3V, T_{AD} = 1 μS, CP ON, 25°C.*

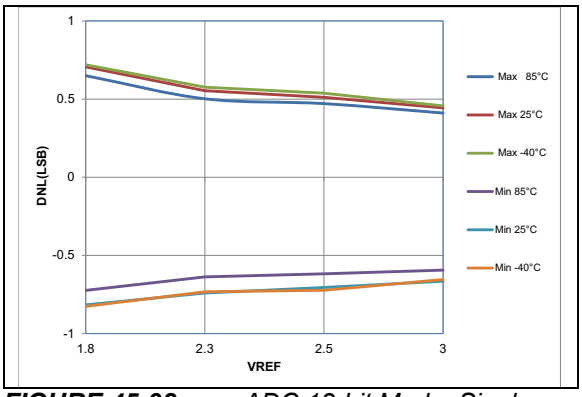


FIGURE 45-98: *ADC 12-bit Mode, Single-Ended Typical DNL, V_{DD} = 3.0V, T_{AD} = 1 μS, CP ON*

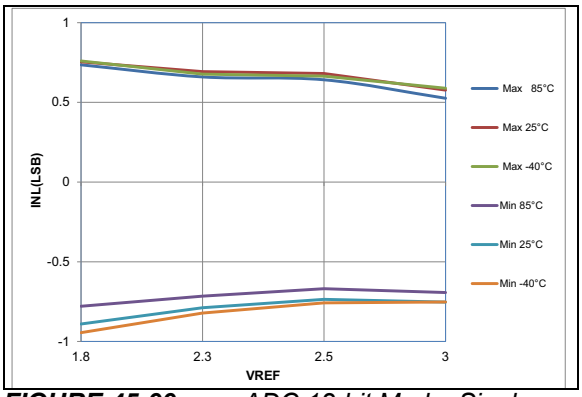


FIGURE 45-99: *ADC 12-bit Mode, Single-Ended Typical INL, V_{DD} = 3.0V, T_{AD} = 1 μS, CP ON.*

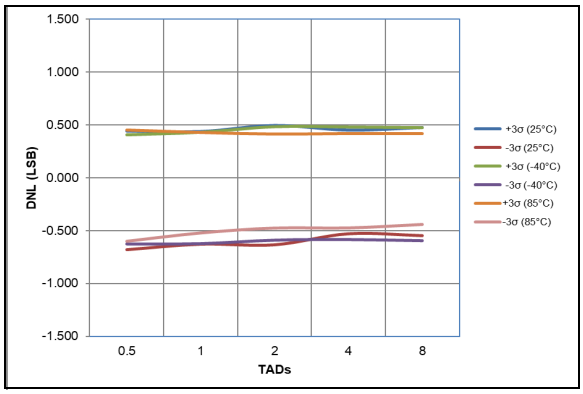


FIGURE 45-100: *ADC 12-bit Mode, Single-Ended DNL, V_{DD} = 3.0V, V_{REF} = 3.0V, All devices.*

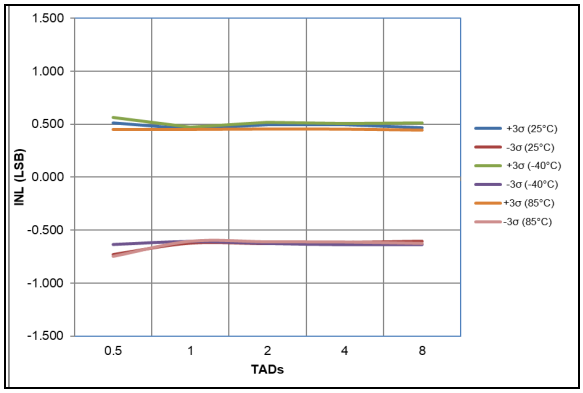


FIGURE 45-101: *ADC 12-bit Mode, Single-Ended INL, V_{DD} = 3.0V, V_{REF} = 3.0V, All devices.*

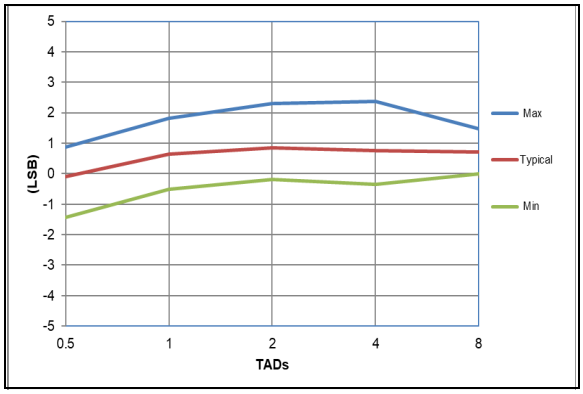


FIGURE 45-102: *ADC 12-bit Mode, Single-Ended Gain, Error, V_{DD} = 3.0V, V_{REF} = 3.0V, -40°C to 85°C, All devices.*

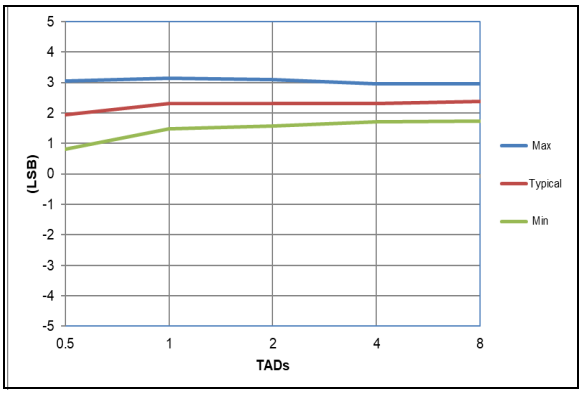


FIGURE 45-103: *ADC 12-bit Mode, Single-Ended Offset, Error, VDD = 3.0V, VREF = 3.0V, -40°C to 85°C, All devices.*

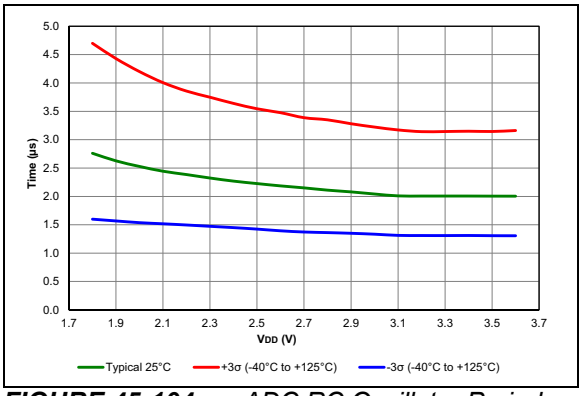


FIGURE 45-104: *ADC RC Oscillator Period, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

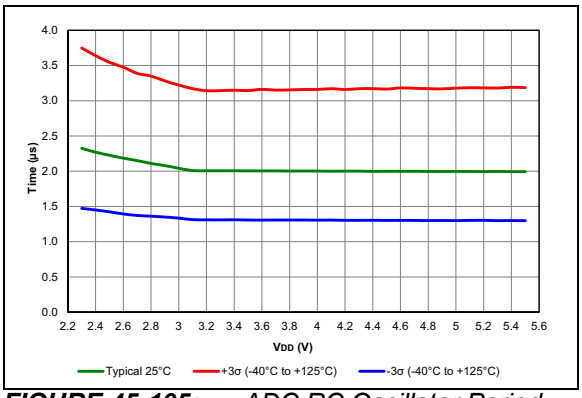


FIGURE 45-105: *ADC RC Oscillator Period, PIC18F26/27/45/46/47/55/56/57K42 Only.*

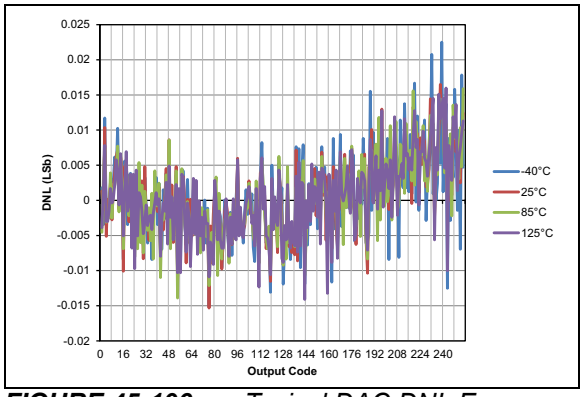


FIGURE 45-106: *Typical DAC DNL Error, VDD = 3.0V, VREF = External 3.0V.*

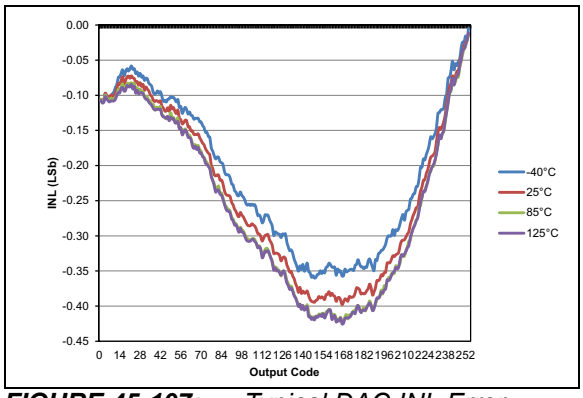


FIGURE 45-107: *Typical DAC INL Error, VDD = 3.0V, VREF = External 3.0V.*

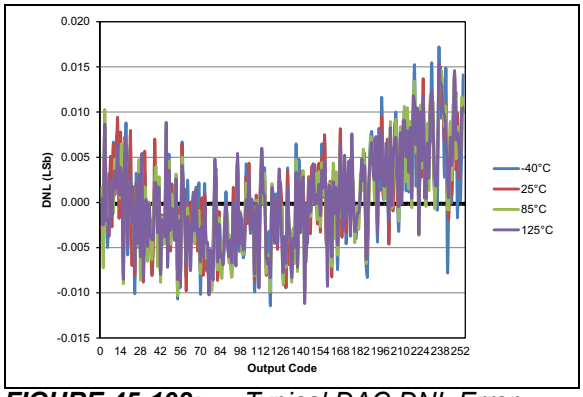


FIGURE 45-108: *Typical DAC DNL Error, VDD = 5.0V, VREF = External 5.0V, PIC18F26/27/45/46/47/55/56/57K42 Only.*

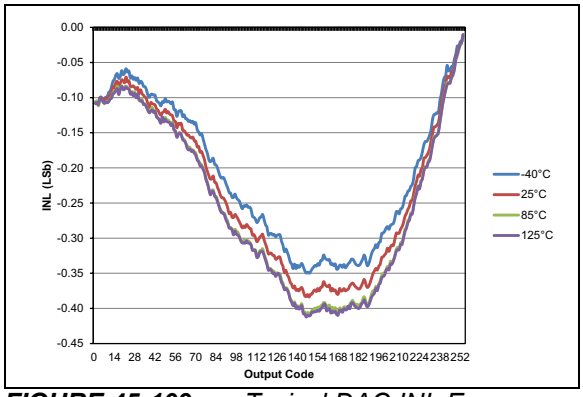


FIGURE 45-109: *Typical DAC INL Error, VDD = 5.0V, VREF = External 5.0V PIC18F26/27/45/46/47/55/56/57K42 Only.*

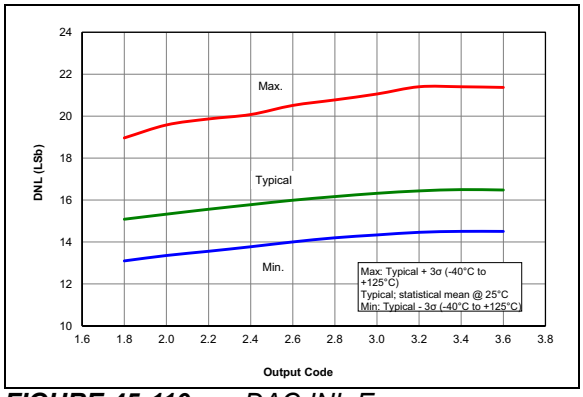


FIGURE 45-110: *DAC INL Error, VDD = 3.0V, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

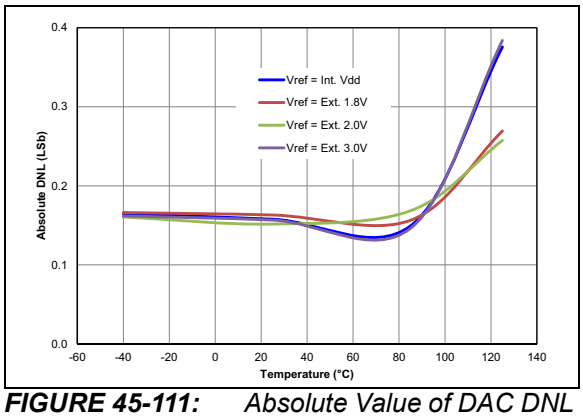


FIGURE 45-111: *Absolute Value of DAC DNL Error, VDD = 3.0V, VREF = VDD.*

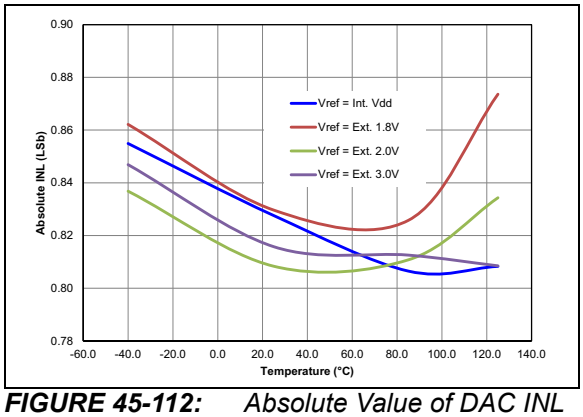


FIGURE 45-112: *Absolute Value of DAC INL Error, VDD = 3.0V, VREF = VDD.*

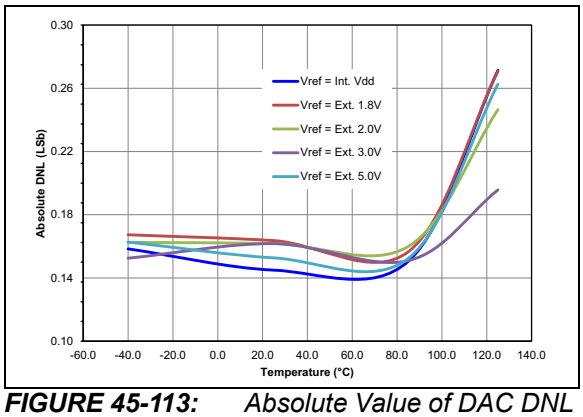


FIGURE 45-113: *Absolute Value of DAC DNL Error, VDD = 5.0V, VREF = VDD, PIC18F26/27/45/46/47/55/56/57K42 Only*

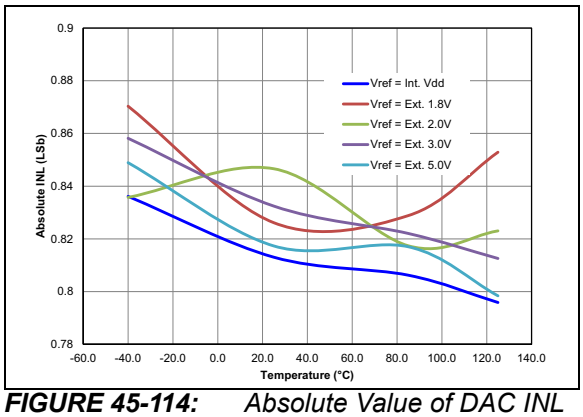


FIGURE 45-114: *Absolute Value of DAC INL Error, VDD = 5.0V, VREF = VDD, PIC18F26/27/45/46/47/55/56/57K42 Only.*

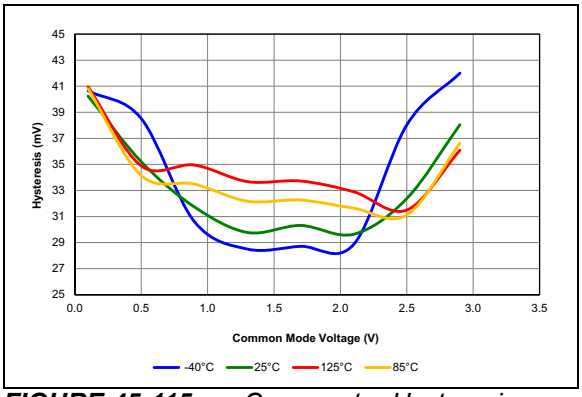


FIGURE 45-115: *Comparator Hysteresis, NP Mode (CxSP = 1), VDD = 3.0V, Typical Measured Values.*

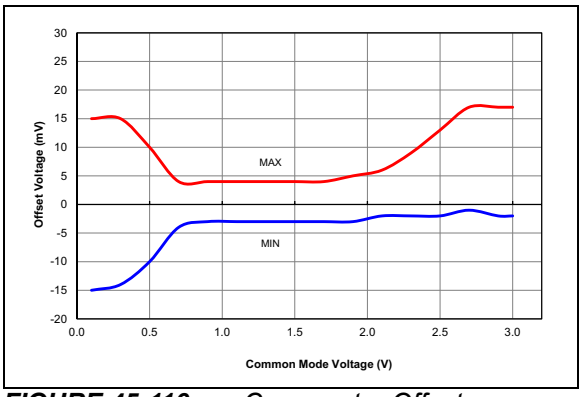


FIGURE 45-116: *Comparator Offset, NP Mode (CxSP = 1), VDD = 3.0V, Typical Measured Values at 25°C.*

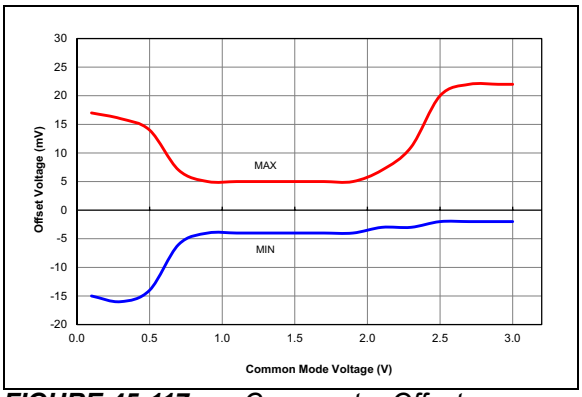


FIGURE 45-117: *Comparator Offset, NP Mode (CxSP = 1), VDD = 3.0V, Typical Measured Values from -40°C to 125°C.*

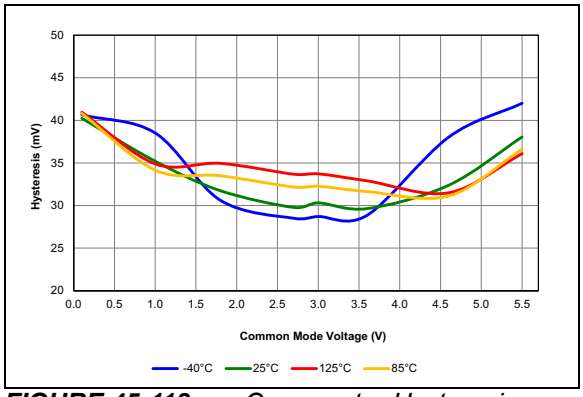


FIGURE 45-118: *Comparator Hysteresis, NP Mode (CxSP = 1), VDD = 5.5V, Typical Measured Values,PIC18F26/27/45/46/47/55/56/57K42 Only.*

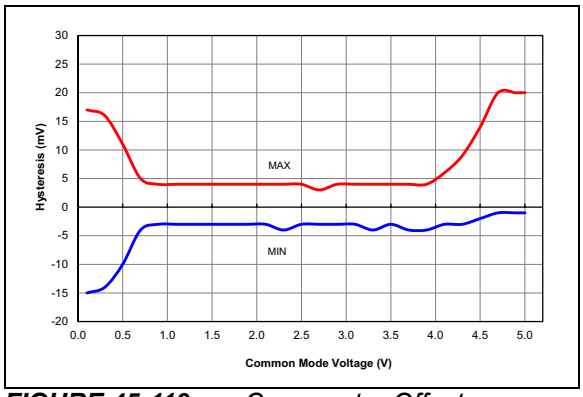


FIGURE 45-119: *Comparator Offset, NP Mode (CxSP = 1), VDD = 5.0V, Typical Measured Values at 25°C, PIC18F26/27/45/46/47/55/56/57K42 Only*

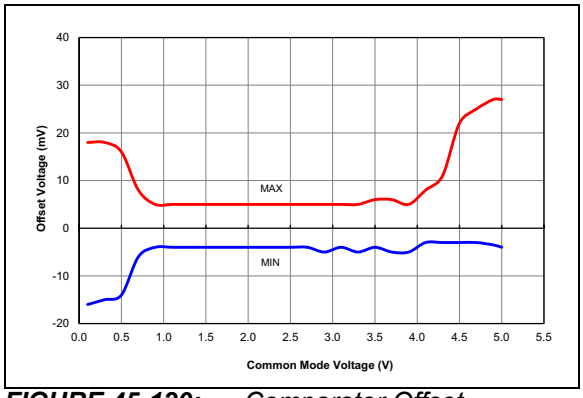


FIGURE 45-120: *Comparator Offset, NP Mode (CxSP = 1), VDD = 5.5V, Typical Measured Values from -40°C to 125°C, PIC18F26/27/45/46/47/55/56/57K42 Only.*

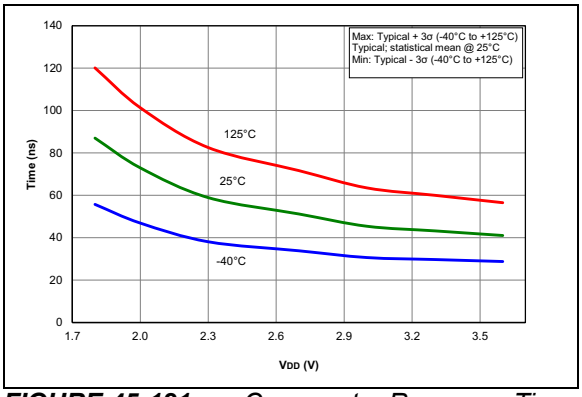


FIGURE 45-121: *Comparator Response Time Over Voltage, NP Mode (CxSP = 1), Typical Measured Values, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

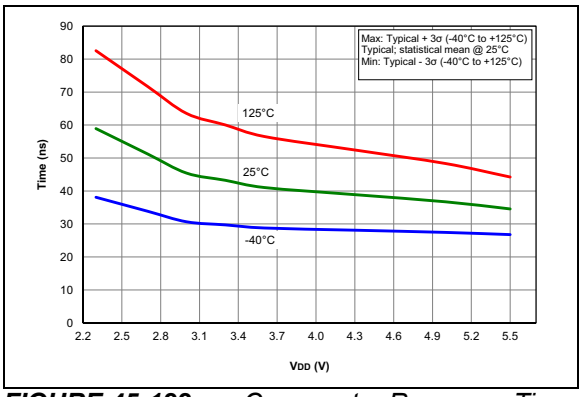


FIGURE 45-122: *Comparator Response Time Over Voltage, NP Mode (CxSP = 1), Typical Measured Values, PIC18F26/27/45/46/47/55/56/57K42 Only*

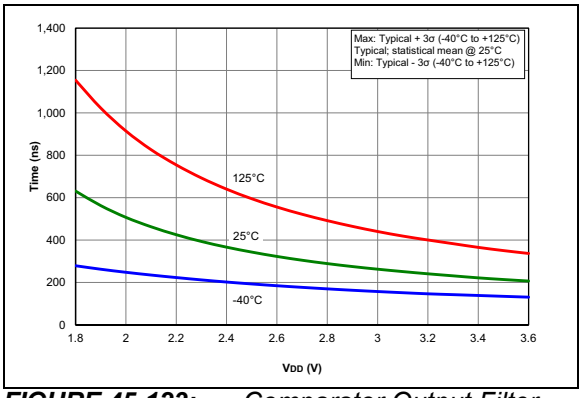


FIGURE 45-123: *Comparator Output Filter Delay Time Over Temp., NP Mode (CxSP = 1), Typical Measured Values, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

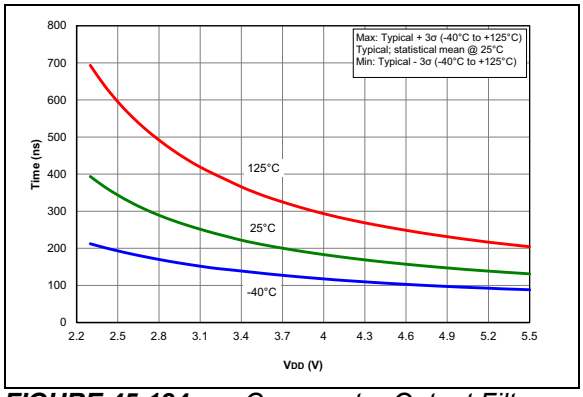


FIGURE 45-124: *Comparator Output Filter Delay Time Over Temp., NP Mode (CxSP = 1), Typical Measured Values, PIC18F26/27/45/46/47/55/56/57K42 Only.*

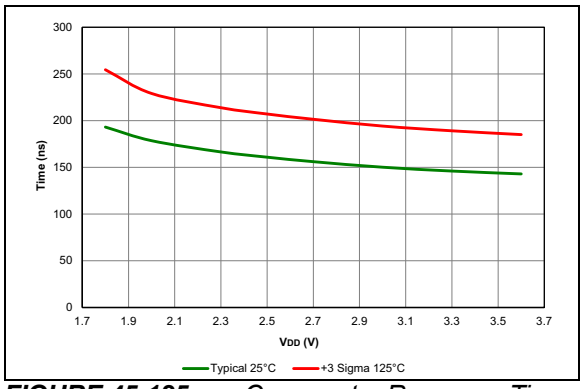


FIGURE 45-125: *Comparator Response Time Falling Edge, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

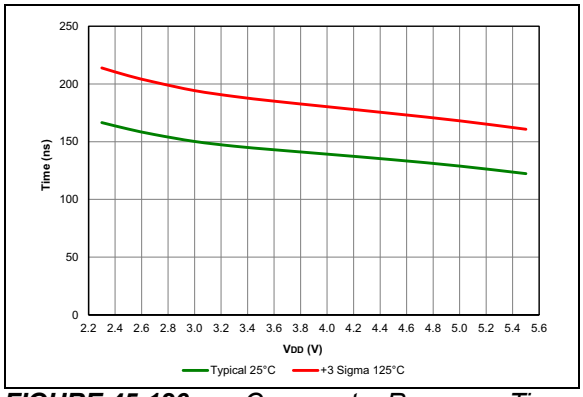


FIGURE 45-126: *Comparator Response Time Falling Edge, PIC18F26/27/45/46/47/55/56/57K42 Only.*

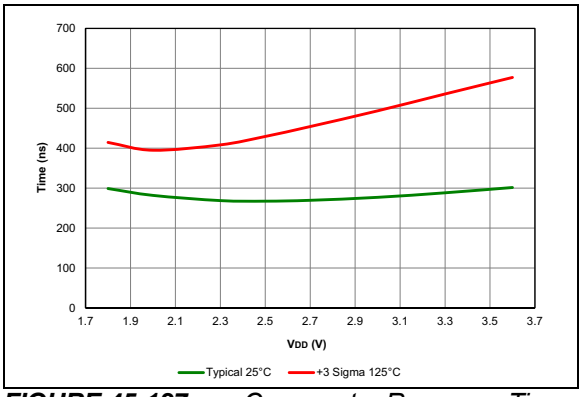


FIGURE 45-127: *Comparator Response Time Rising Edge, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

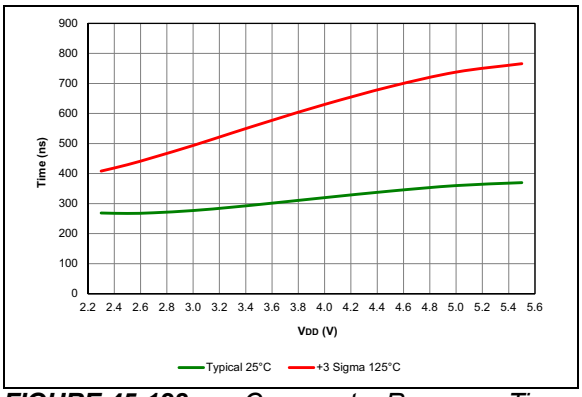


FIGURE 45-128: *Comparator Response Time Rising Edge, PIC18F26/27/45/46/47/55/56/57K42 Only.*

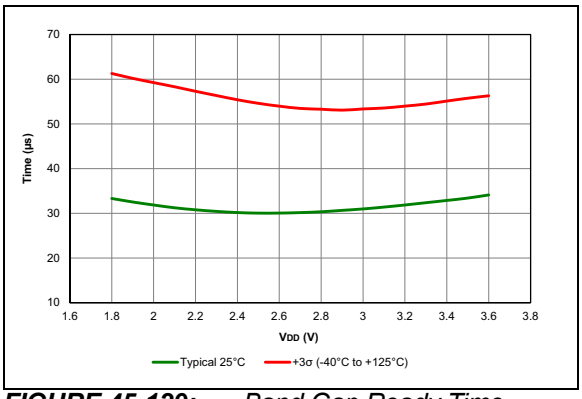


FIGURE 45-129: *Band Gap Ready Time, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

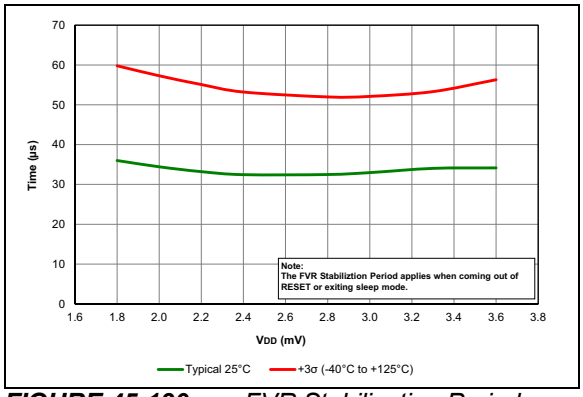


FIGURE 45-130: *FVR Stabilization Period, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

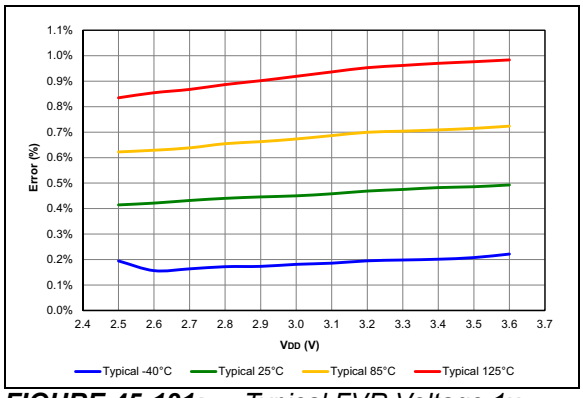


FIGURE 45-131: *Typical FVR Voltage 1x, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

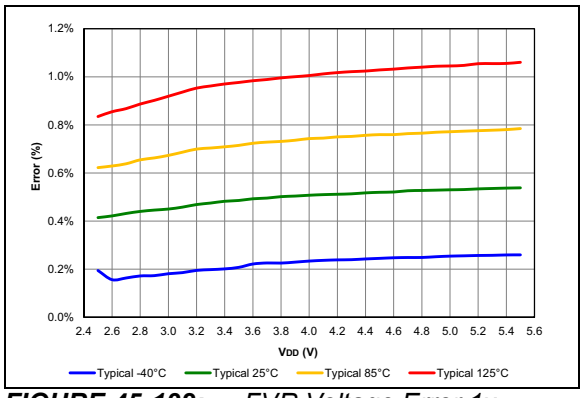


FIGURE 45-132: *FVR Voltage Error 1x, PIC18F26/27/45/46/47/55/56/57K42 Only.*

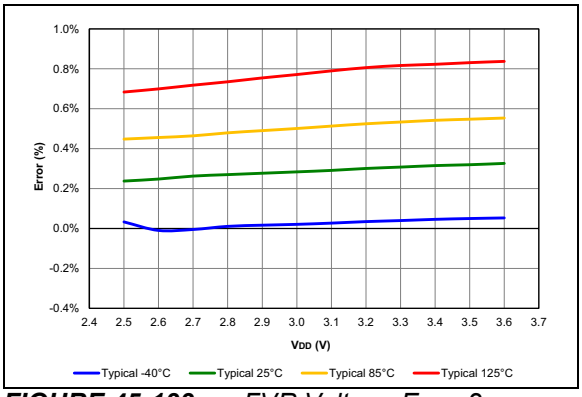


FIGURE 45-133: *FVR Voltage Error 2x, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

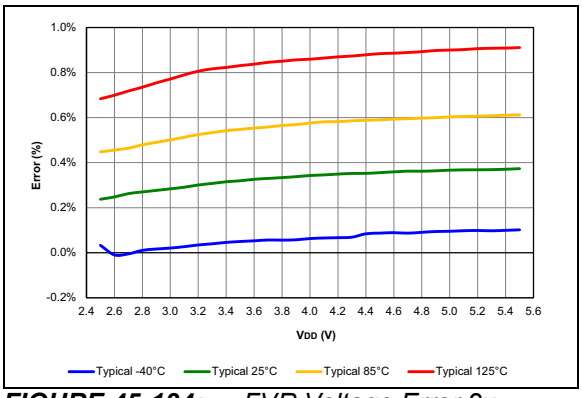


FIGURE 45-134: *FVR Voltage Error 2x, PIC18F26/27/45/46/47/55/56/57K42 Only.*

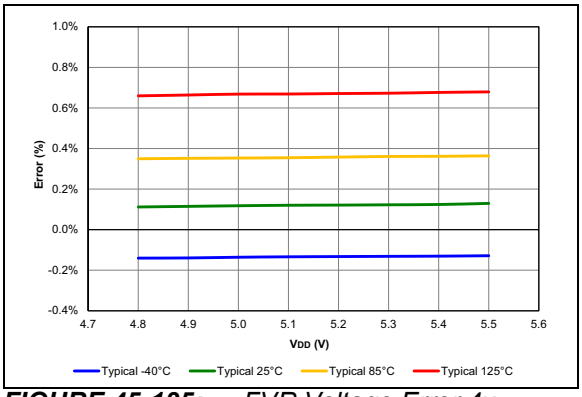


FIGURE 45-135: *FVR Voltage Error 4x, PIC18F26/27/45/46/47/55/56/57K42 Only.*

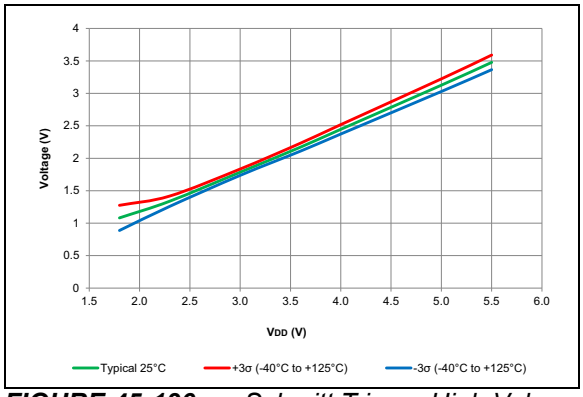


FIGURE 45-136: *Schmitt Trigger High Values.*

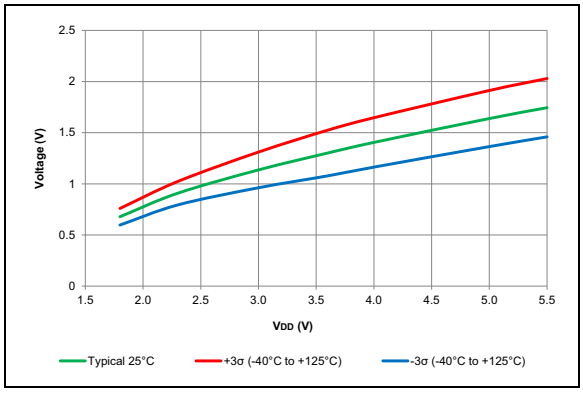


FIGURE 45-137: *Schmitt Trigger Low Values.*

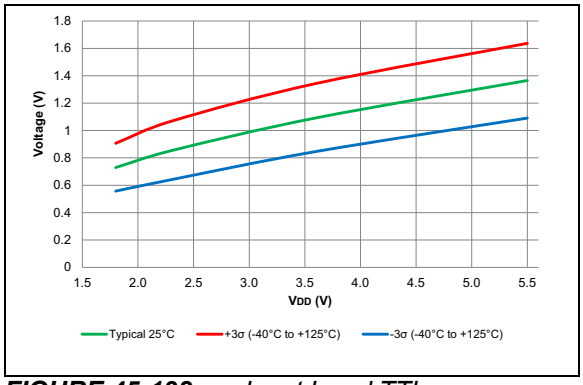


FIGURE 45-138: *Input Level TTL.*

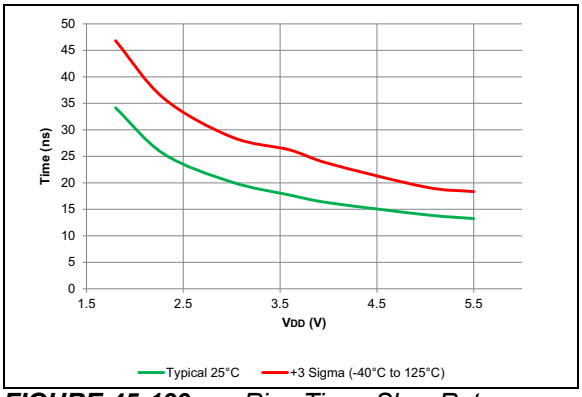


FIGURE 45-139: *Rise Time, Slew Rate Control Enabled.*

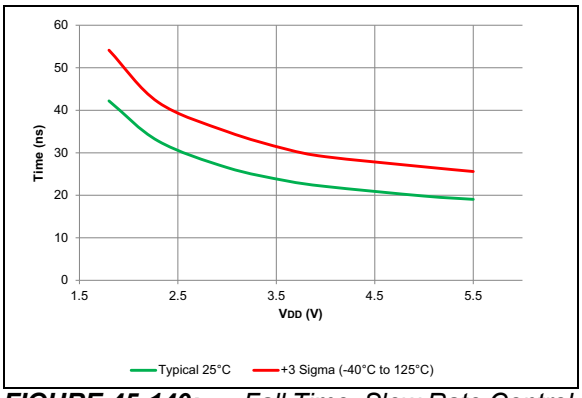


FIGURE 45-140: *Fall Time, Slew Rate Control Enabled.*

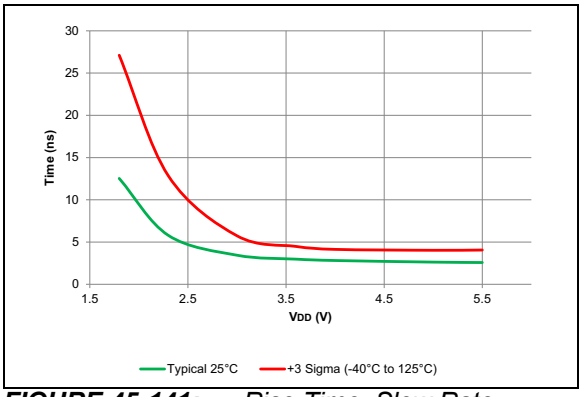


FIGURE 45-141: *Rise Time, Slew Rate Control Disabled.*

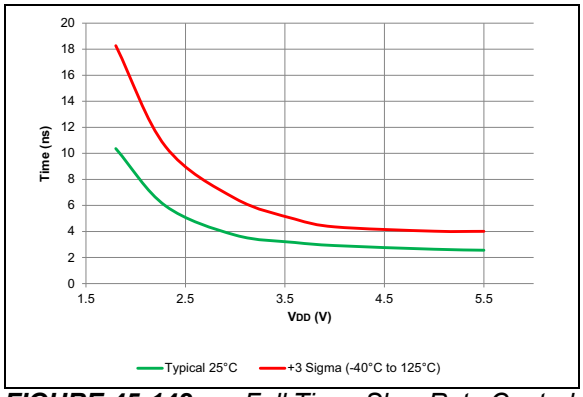


FIGURE 45-142: *Fall Time, Slew Rate Control Disabled.*

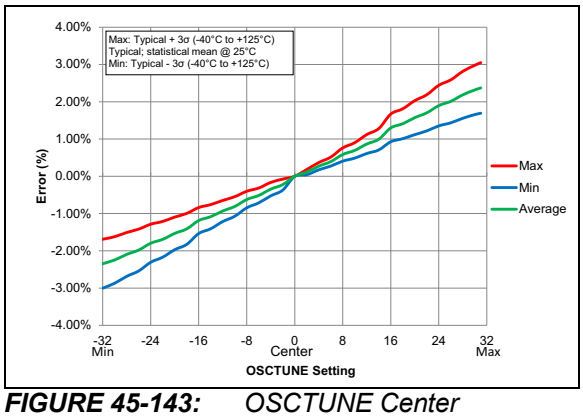


FIGURE 45-143: *OSCTUNE Center Frequency, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

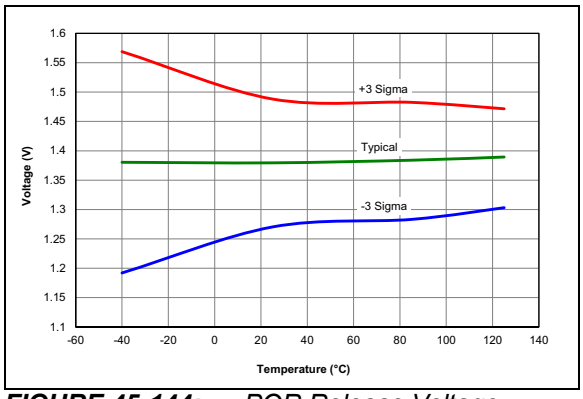


FIGURE 45-144: *POR Release Voltage.*

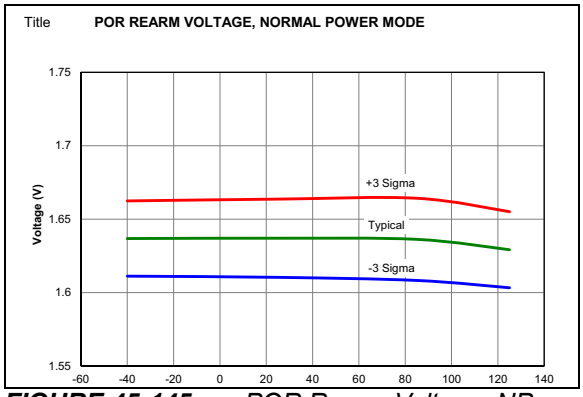


FIGURE 45-145: *POR Rearm Voltage, NP Mode, PIC18F26/27/45/46/47/55/56/57K42 Only.*

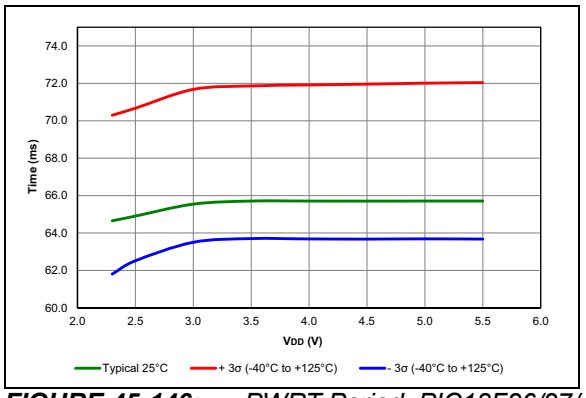


FIGURE 45-146: *PWRT Period, PIC18F26/27/45/46/47/55/56/57K42 Only.*

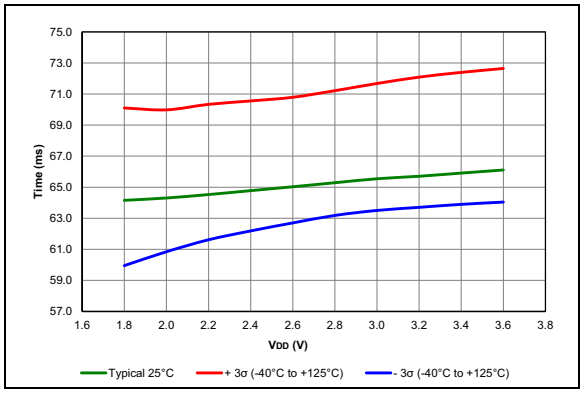


FIGURE 45-147: *PWRT Period, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

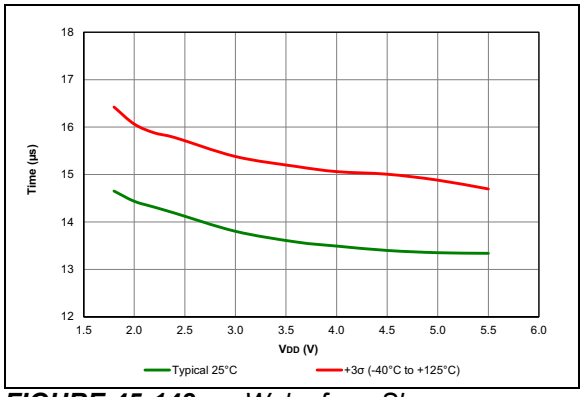


FIGURE 45-148: *Wake from Sleep, VREGPM = 0, HFINTOSC = 4 MHz, PIC18F26/27/45/46/47/55/56/57K42 Only.*

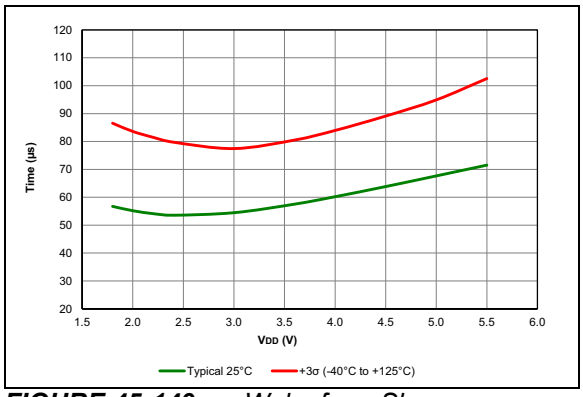


FIGURE 45-149: *Wake from Sleep, VREGPM = 1, HFINTOSC = 4 MHz, PIC18F26/27/45/46/47/55/56/57K42 Only.*

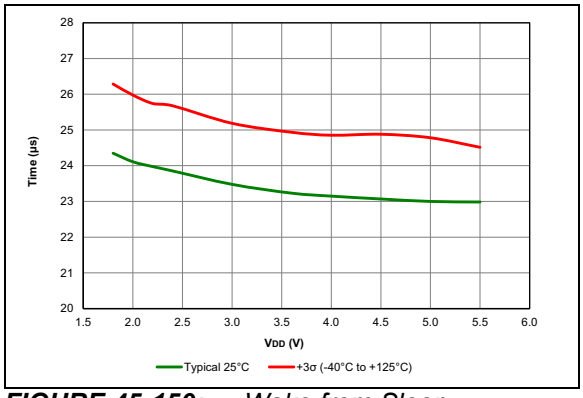


FIGURE 45-150: *Wake from Sleep, VREGPM = 0, HFINTOSC = 16 MHz, PIC18F26/27/45/46/47/55/56/57K42 Only.*

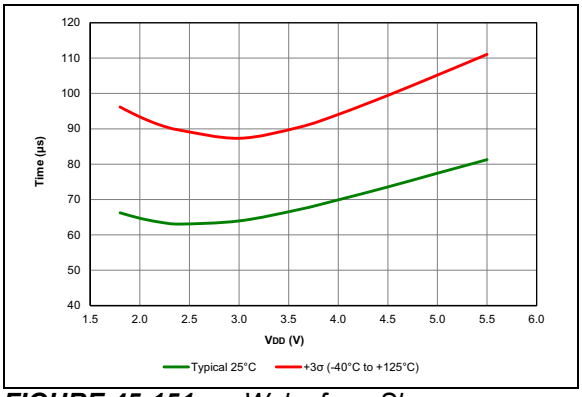


FIGURE 45-151: *Wake from Sleep, VREGPM = 1, HFINTOSC = 16 MHz, PIC18F26/27/45/46/47/55/56/57K42 Only.*

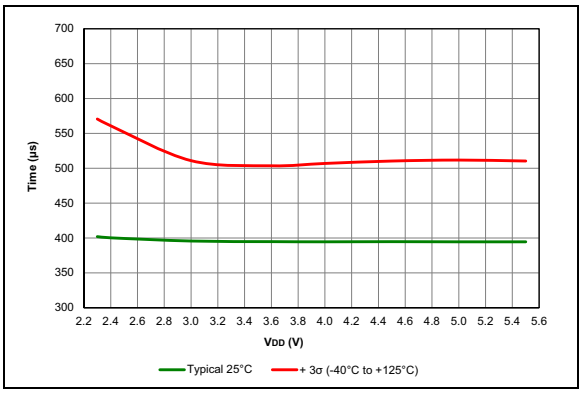


FIGURE 45-152: *Wake from Sleep, VREGPM = 1, PIC18F26/27/45/46/47/55/56/57K42 Only.*

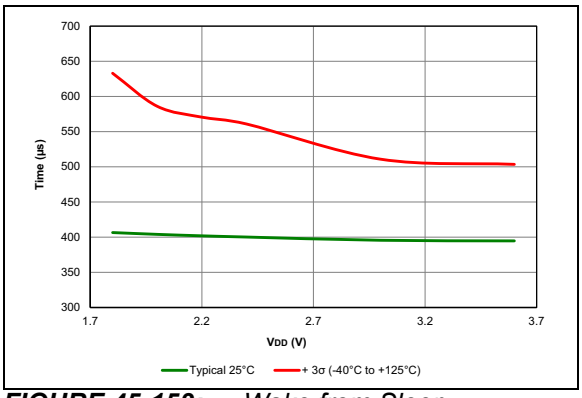


FIGURE 45-153: *Wake from Sleep, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

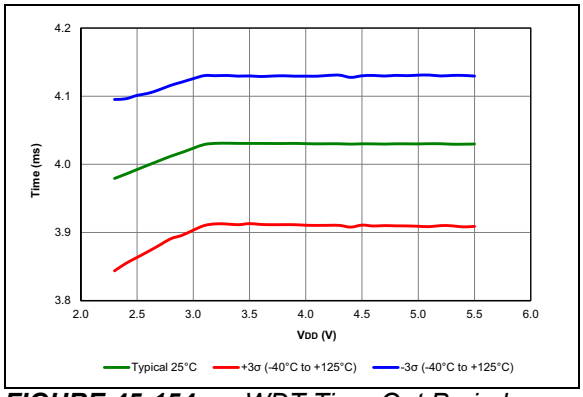


FIGURE 45-154: *WDT Time-Out Period, PIC18F26/27/45/46/47/55/56/57K42 Only.*

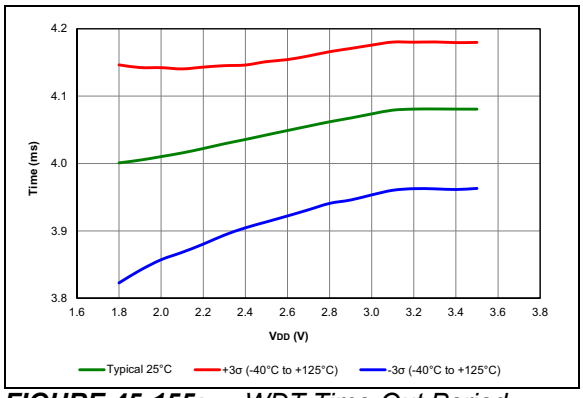


FIGURE 45-155: *WDT Time-Out Period, PIC18LF26/27/45/46/47/55/56/57K42 Only.*

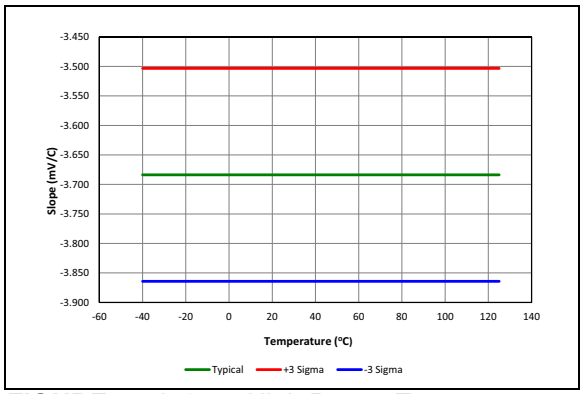


FIGURE 45-156: *High Range Temperature Indicator Voltage Sensitivity Across Temperature.*

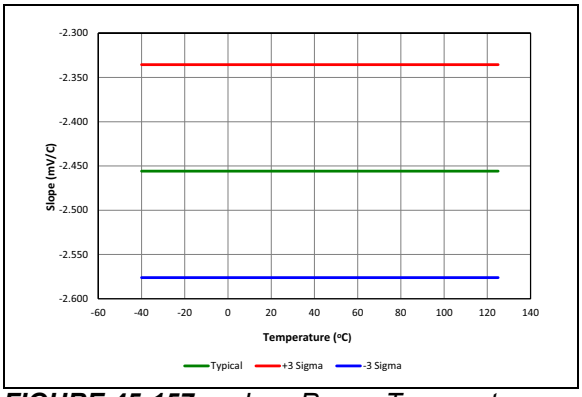


FIGURE 45-157: *Low Range Temperature Indicator Voltage Sensitivity Across Temperature*

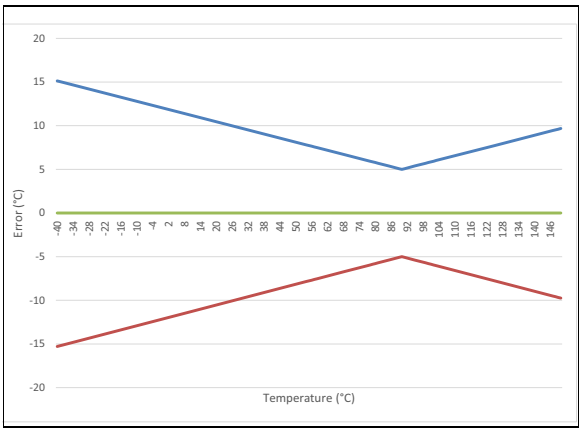
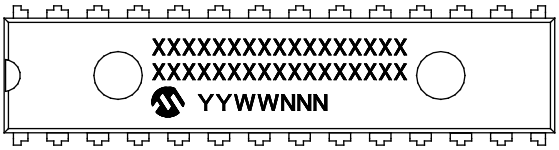


FIGURE 45-158: *Temperature Indicator Performance Over Temperature*

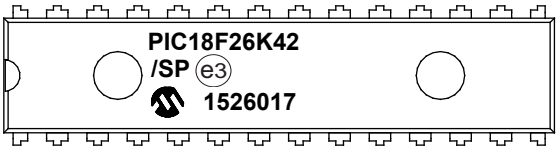
46.0 PACKAGING INFORMATION

Package Marking Information

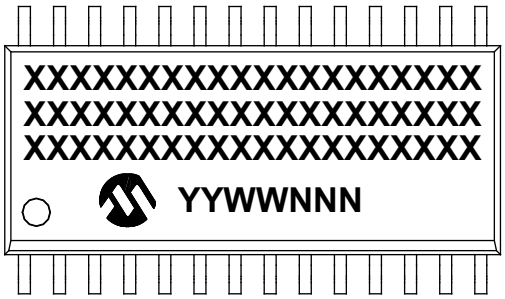
28-Lead SPDIP (.300”)



Example

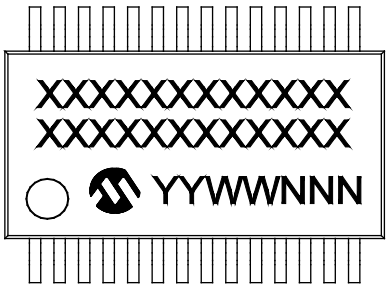


28-Lead SOIC (7.50 mm)

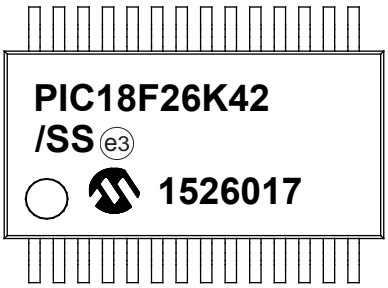


Example

28-Lead SSOP (5.30 mm)



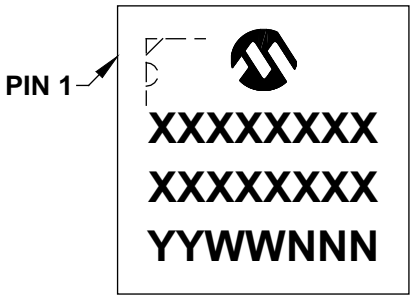
Example



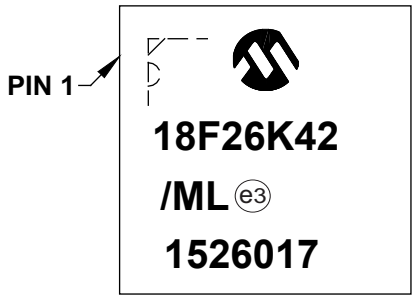
Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	e3	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.
Note:	In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.	

Package Marking Information (Continued)

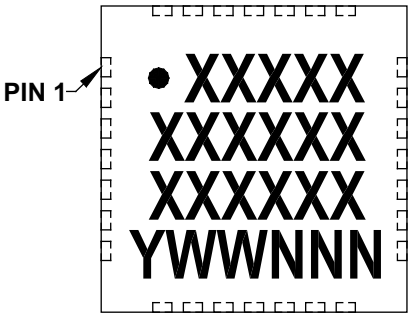
28-Lead QFN (6x6 mm)



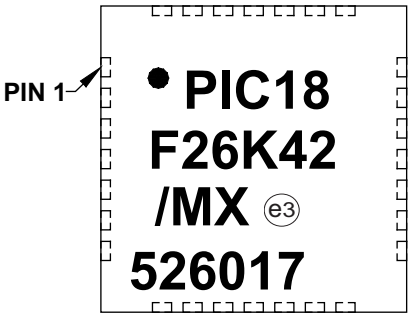
Example



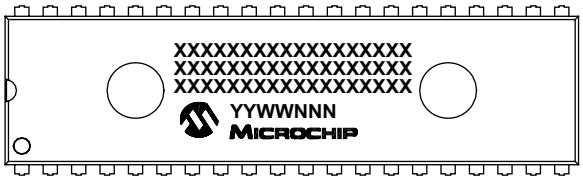
28-Lead UQFN (6x6x0.5 mm)



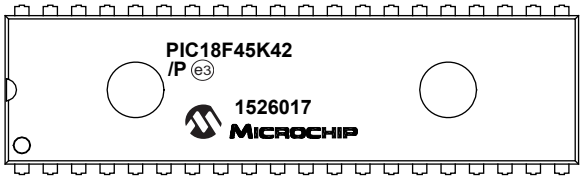
Example



40-Lead PDIP (600 mil)



Example

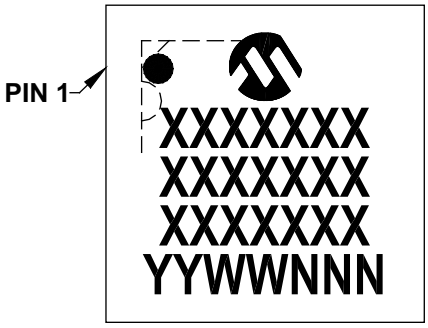


Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	e3	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

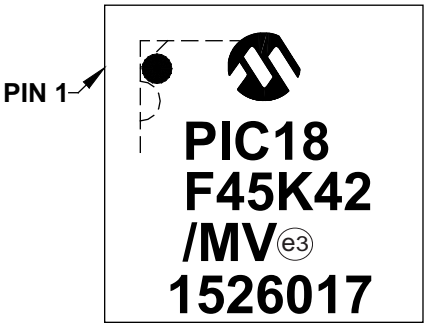
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

Package Marking Information (Continued)

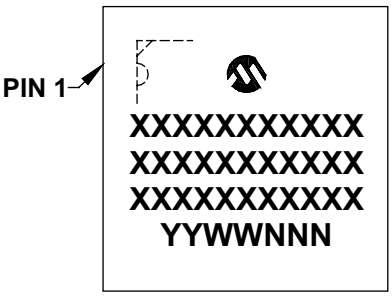
40-Lead UQFN (5x5x0.5 mm)



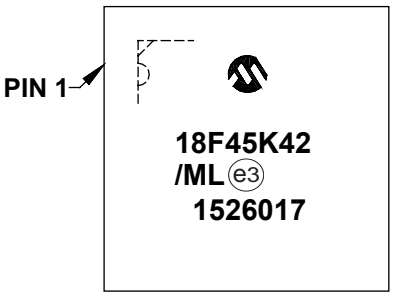
Example



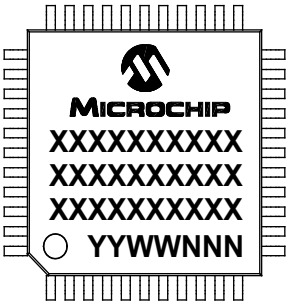
44-Lead QFN (8x8x0.9 mm)



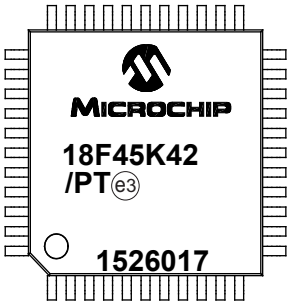
Example



44-Lead TQFP (10x10x1 mm)



Example

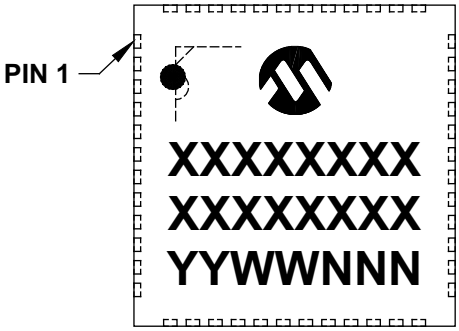


Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	e3	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

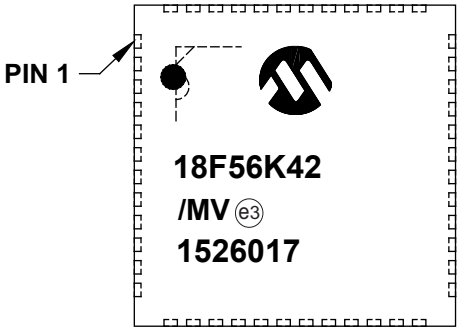
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

Package Marking Information (Continued)

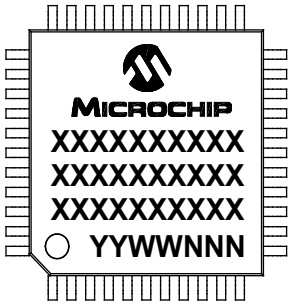
48-Lead UQFN (6x6x0.5 mm)



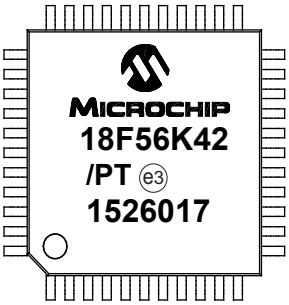
Example



48-Lead TQFP (7x7x1 mm)



Example



Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	e3	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

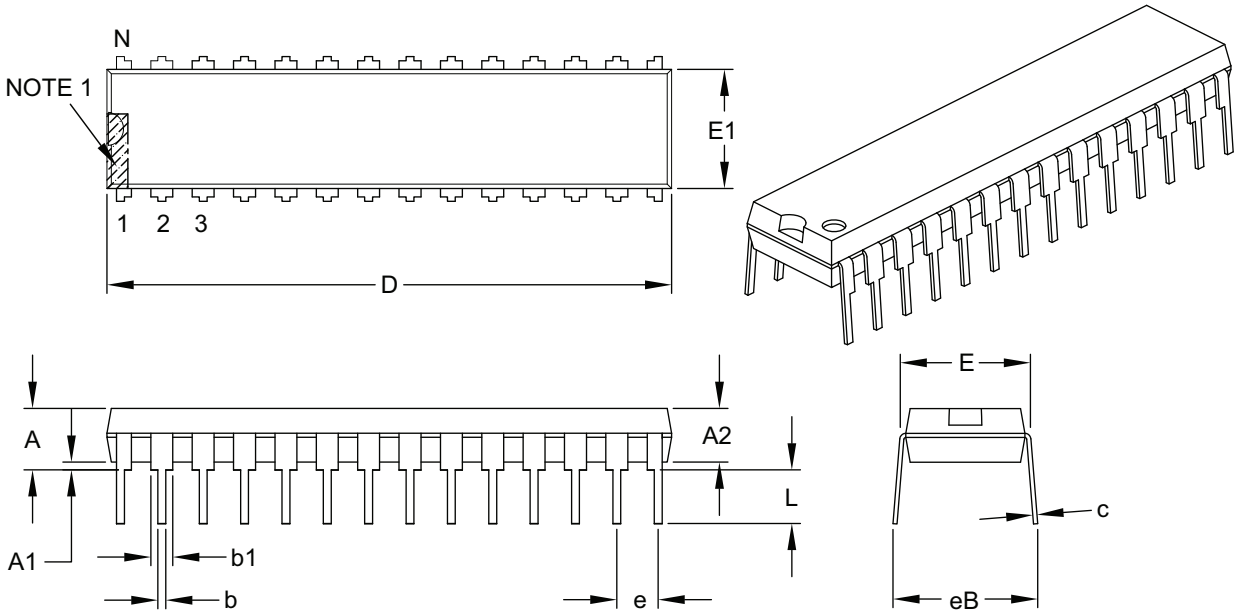
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

46.1 Package Details

The following sections give the technical details of the packages.

28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

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



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	.100 BSC		
Top to Seating Plane	A	–	–	.200
Molded Package Thickness	A2	.120	.135	.150
Base to Seating Plane	A1	.015	–	–
Shoulder to Shoulder Width	E	.290	.310	.335
Molded Package Width	E1	.240	.285	.295
Overall Length	D	1.345	1.365	1.400
Tip to Seating Plane	L	.110	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.040	.050	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	–	–	.430

Notes:

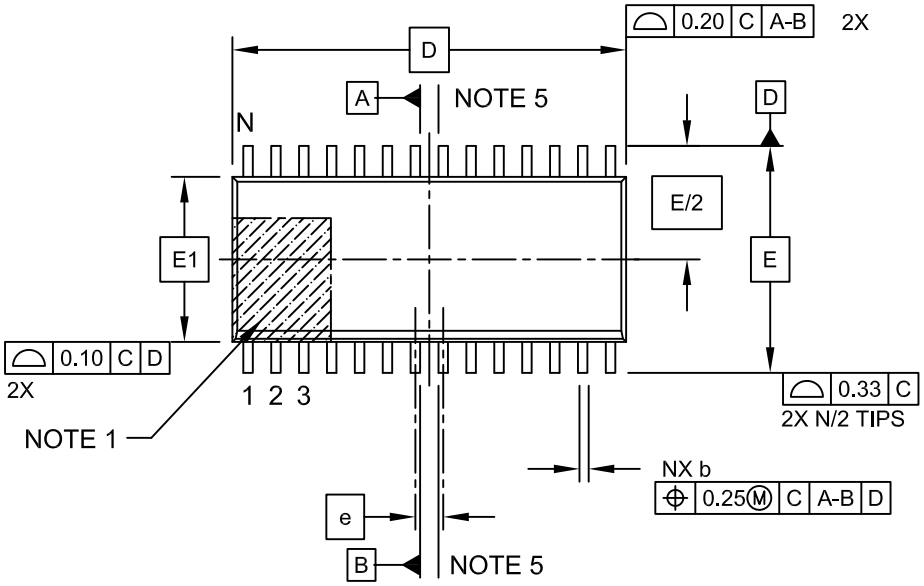
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

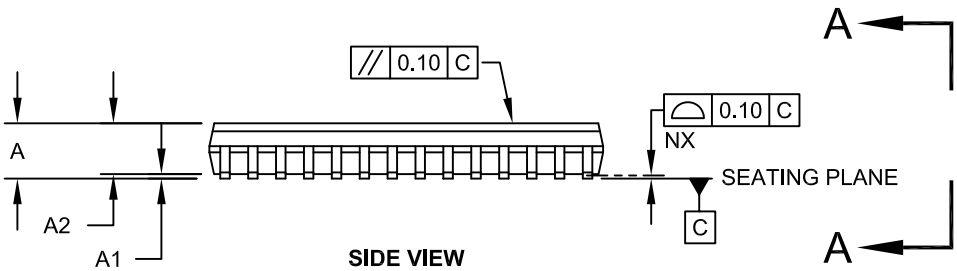
Microchip Technology Drawing C04-070B

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

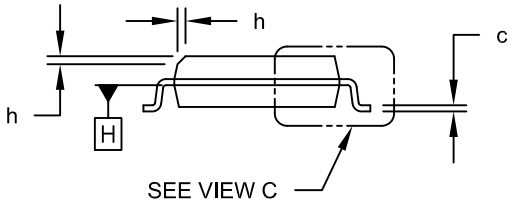
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW

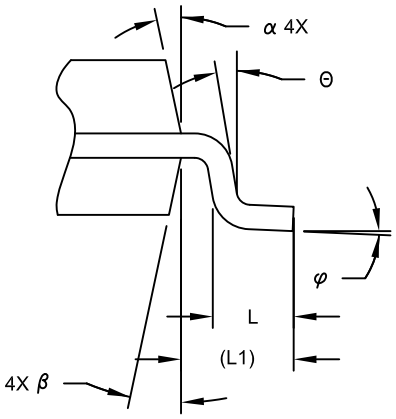


VIEW A-A

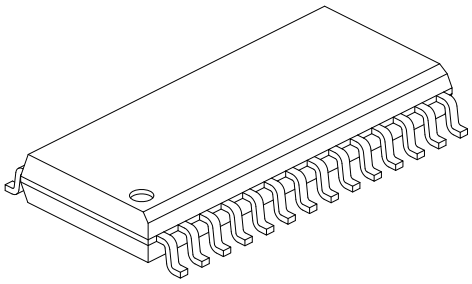
Microchip Technology Drawing C04-052C Sheet 1 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

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



VIEW C



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	1.27 BSC		
Overall Height	A	-	-	2.65
Molded Package Thickness	A2	2.05	-	-
Standoff §	A1	0.10	-	0.30
Overall Width	E	10.30 BSC		
Molded Package Width	E1	7.50 BSC		
Overall Length	D	17.90 BSC		
Chamfer (Optional)	h	0.25	-	0.75
Foot Length	L	0.40	-	1.27
Footprint	L1	1.40 REF		
Lead Angle	Θ	0°	-	-
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.18	-	0.33
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

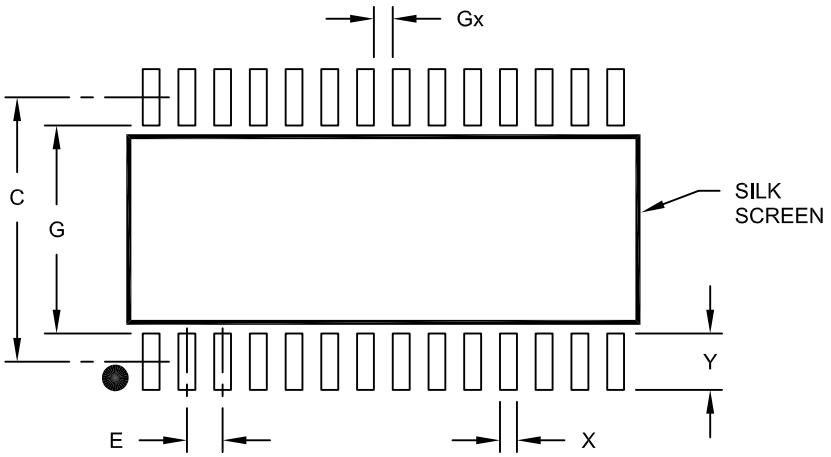
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		1.27 BSC	
Contact Pad Spacing	C		9.40	
Contact Pad Width (X28)	X			0.60
Contact Pad Length (X28)	Y			2.00
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.40		

Notes:

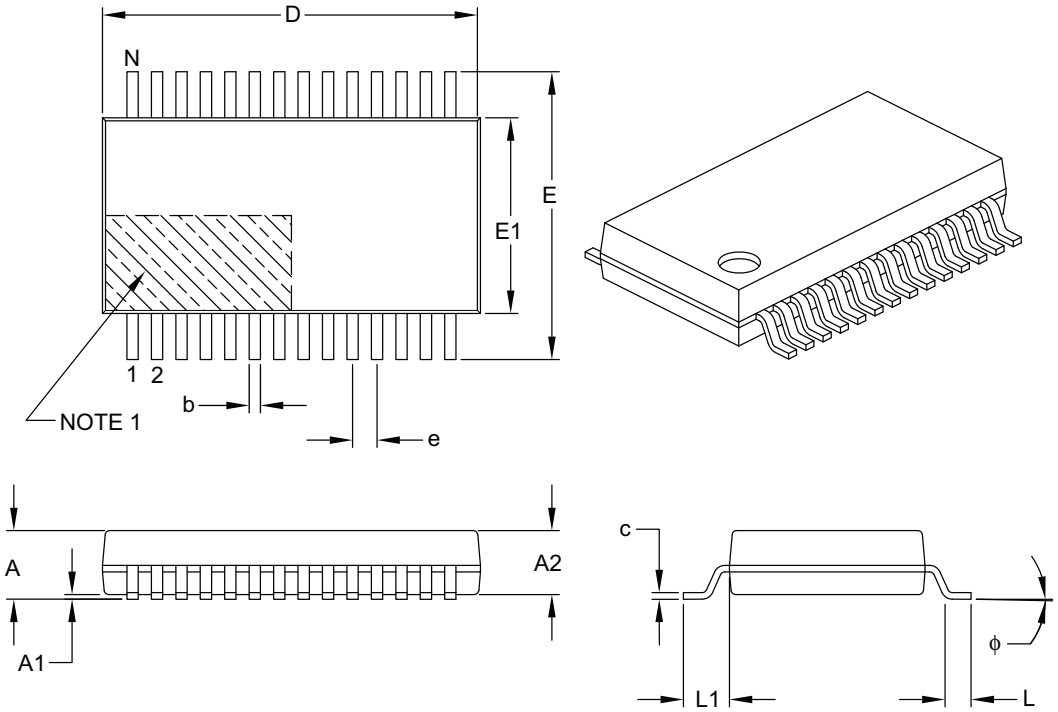
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

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



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	0.65 BSC		
Overall Height	A	–	–	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	–	–
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1	1.25 REF		
Lead Thickness	c	0.09	–	0.25
Foot Angle	ϕ	0°	4°	8°
Lead Width	b	0.22	–	0.38

Notes:

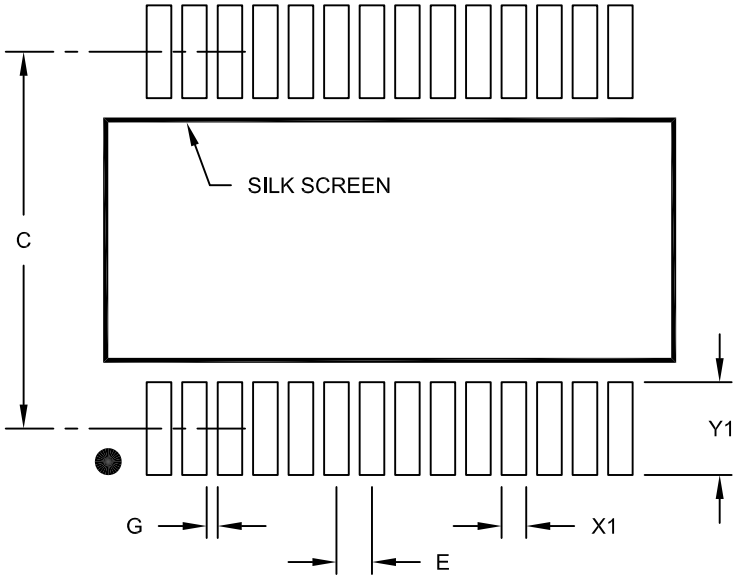
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Contact Pad Spacing	C		7.20	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.75
Distance Between Pads	G	0.20		

Notes:

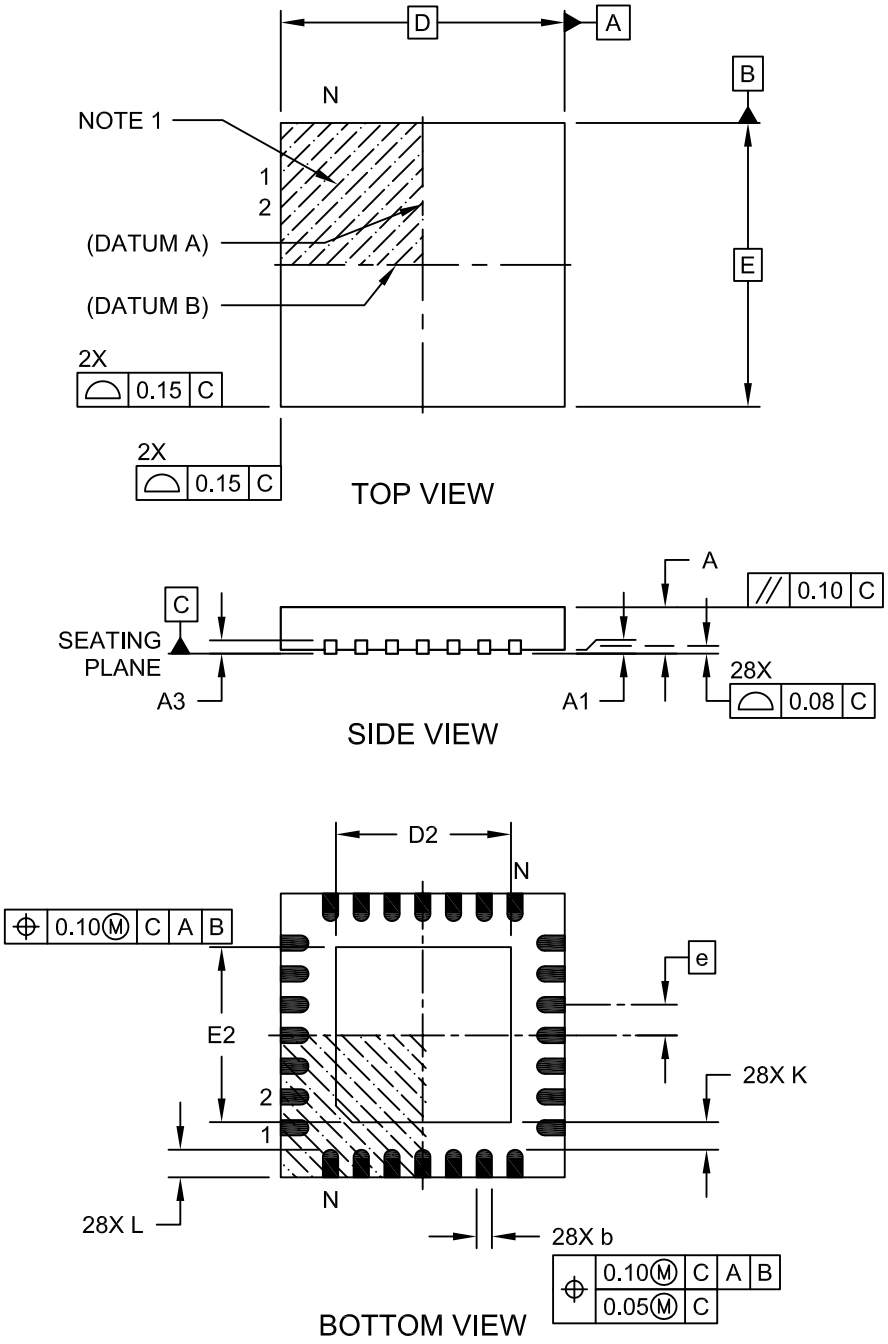
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

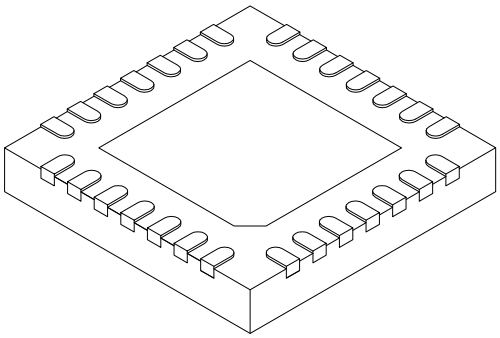
28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN]
With 0.55 mm Terminal Length

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



Microchip Technology Drawing C04-105C Sheet 1 of 2

28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN]
With 0.55 mm Terminal Length



	Units	MILLIMETERS		
Dimension	Limits	MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	0.65 BSC		
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3	0.20 REF		
Overall Width	E	6.00 BSC		
Exposed Pad Width	E2	3.65	3.70	4.20
Overall Length	D	6.00 BSC		
Exposed Pad Length	D2	3.65	3.70	4.20
Terminal Width	b	0.23	0.30	0.35
Terminal Length	L	0.50	0.55	0.70
Terminal-to-Exposed Pad	K	0.20	-	-

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Mic