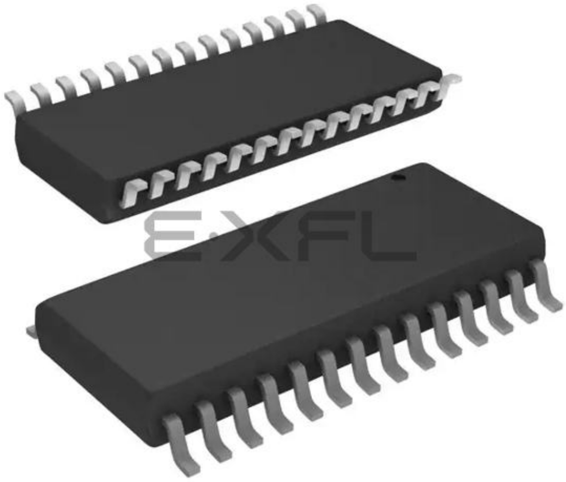


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

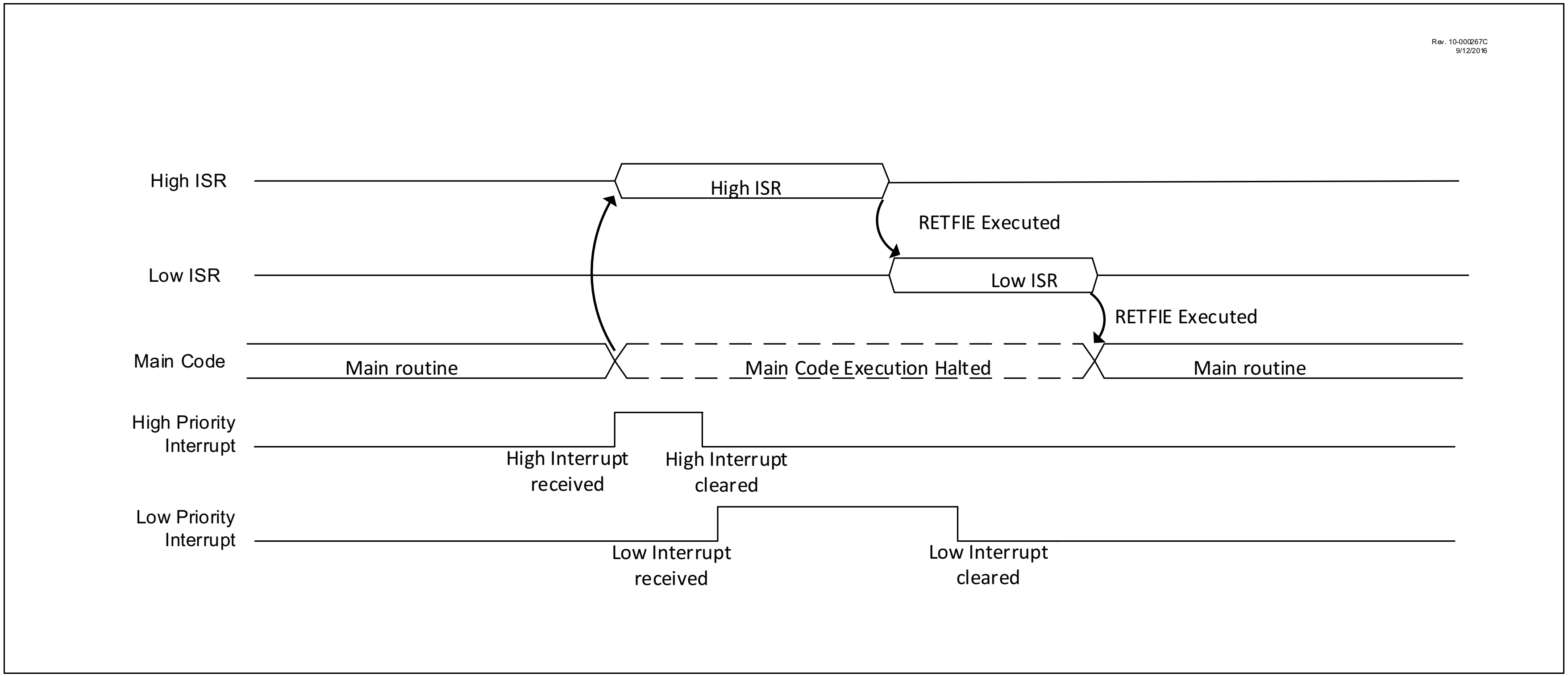
Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	CANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	25
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25k83t-i-so

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



REGISTER 9-12: PIR9: PERIPHERAL INTERRUPT REGISTER 9[1]

U-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
—	CLC4IF	CCP4IF	CLC3IF	CWG3IF	CCP3IF	TMR6IF	TMR5GIF
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 **CLC4IF:** CLC4 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5 **CCP4IF:** CCP4 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 4 **CLC3IF:** CLC3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 3 **CWG3IF:** CWG3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **CCP3IF:** CCP3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

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

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

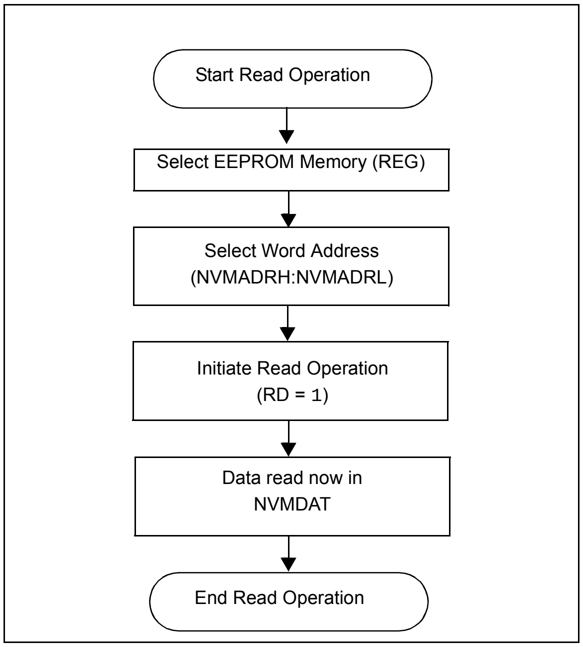
Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

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.

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 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”**.

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 23-4) 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.

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

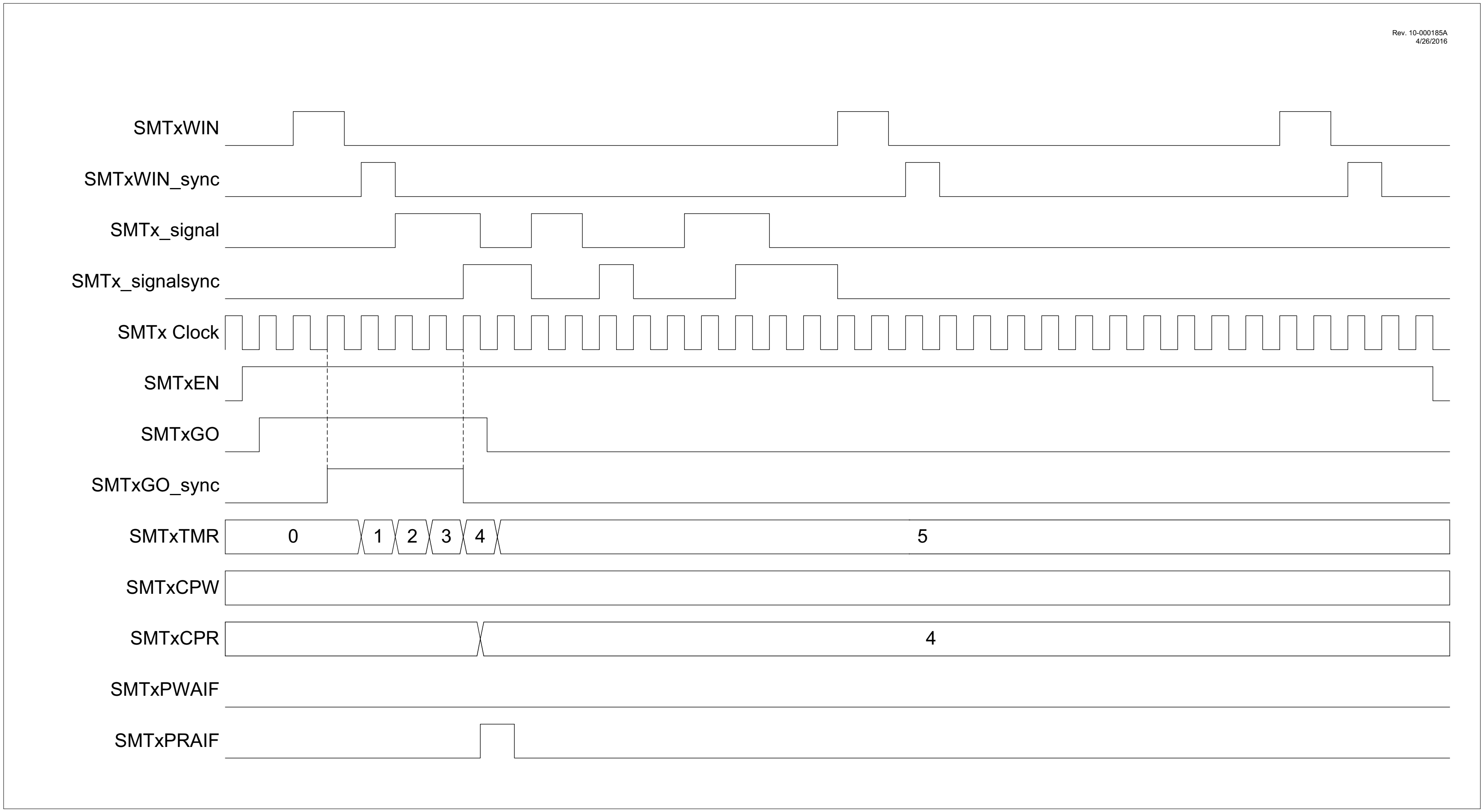
bit 7-6 **P8TSEL<1:0>:** PWM8 Timer Selection bits
`11` = PWM8 based on TMR6
`10` = PWM8 based on TMR4
`01` = PWM8 based on TMR2
`00` = Reserved

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

FIGURE 25-15: TIME OF FLIGHT MODE SINGLE ACQUISITION TIMING DIAGRAM



Preliminary

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 SMTxCPR register on each rising edge of SMTWINx, and updates the value of the CPW register on each falling edge of the SMTWINx. The timer is not reset by any hardware conditions in this mode and must be reset by software, if desired. See Figure 25-16 and Figure 25-17.

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

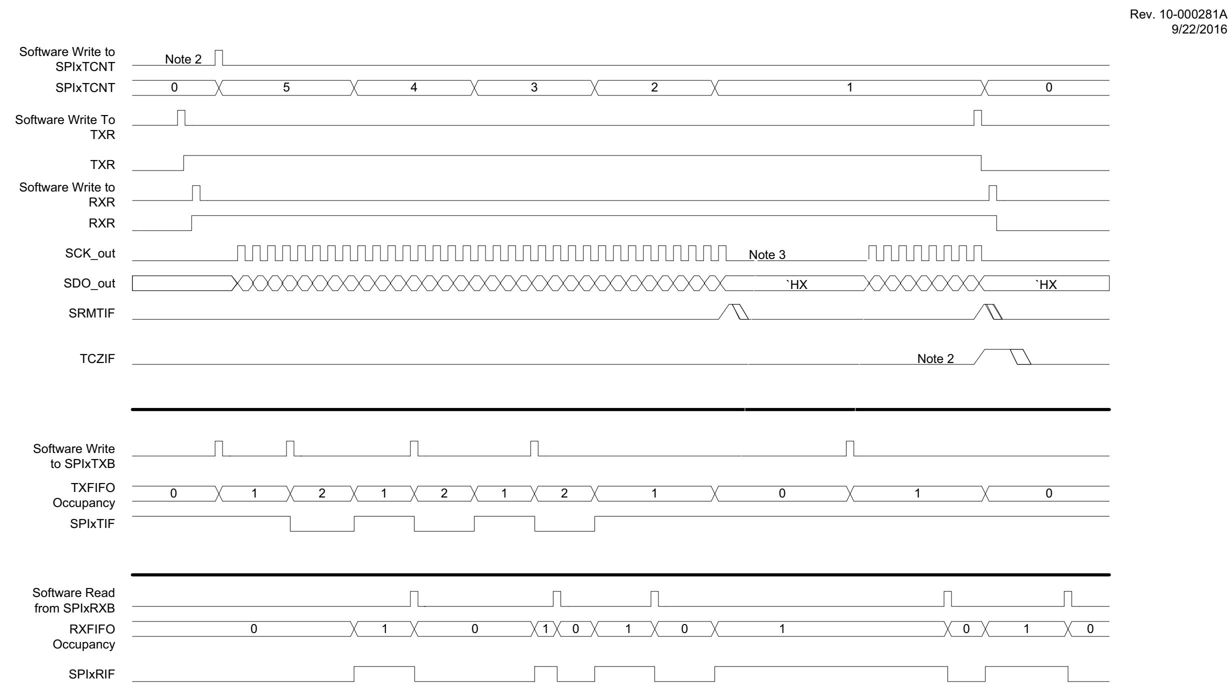
bit 5 **TXEN:** Transmit Enable Control bit(2)
- `1` = Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is idle.
- `0` = Transmit is disabled. TX output pin drive is controlled by PORT TRIS control

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
- `1011` = LIN Slave-Only mode
- `1010` = DMX mode
- `1001` = DALI Control Gear mode
- `1000` = DALI Control Device mode
- `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: ABDEN is read-only when MODE = `1001`. When MODE = `100x` and ABDEN = `1`, then auto-baud is determined from Start bit.

FIGURE 32-3: SPI MASTER OPERATION – DATA EXCHANGE, TXR/RXR = 1/1



Note: 1. SS(out) is not shown on this diagram
2. SPIxTCNT write is optional TXR/RXR = 1/1 and BMODE=1. If BMODE=0, a write to SPIxTCNT is required to start transmission; TCZIF signals the end of the transmission.
3. Transmission gap occurs while waiting for transmitter data.

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 SPIxTCTL is written with ‘3’, the transfer will start with the SPIxTCTL 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.

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

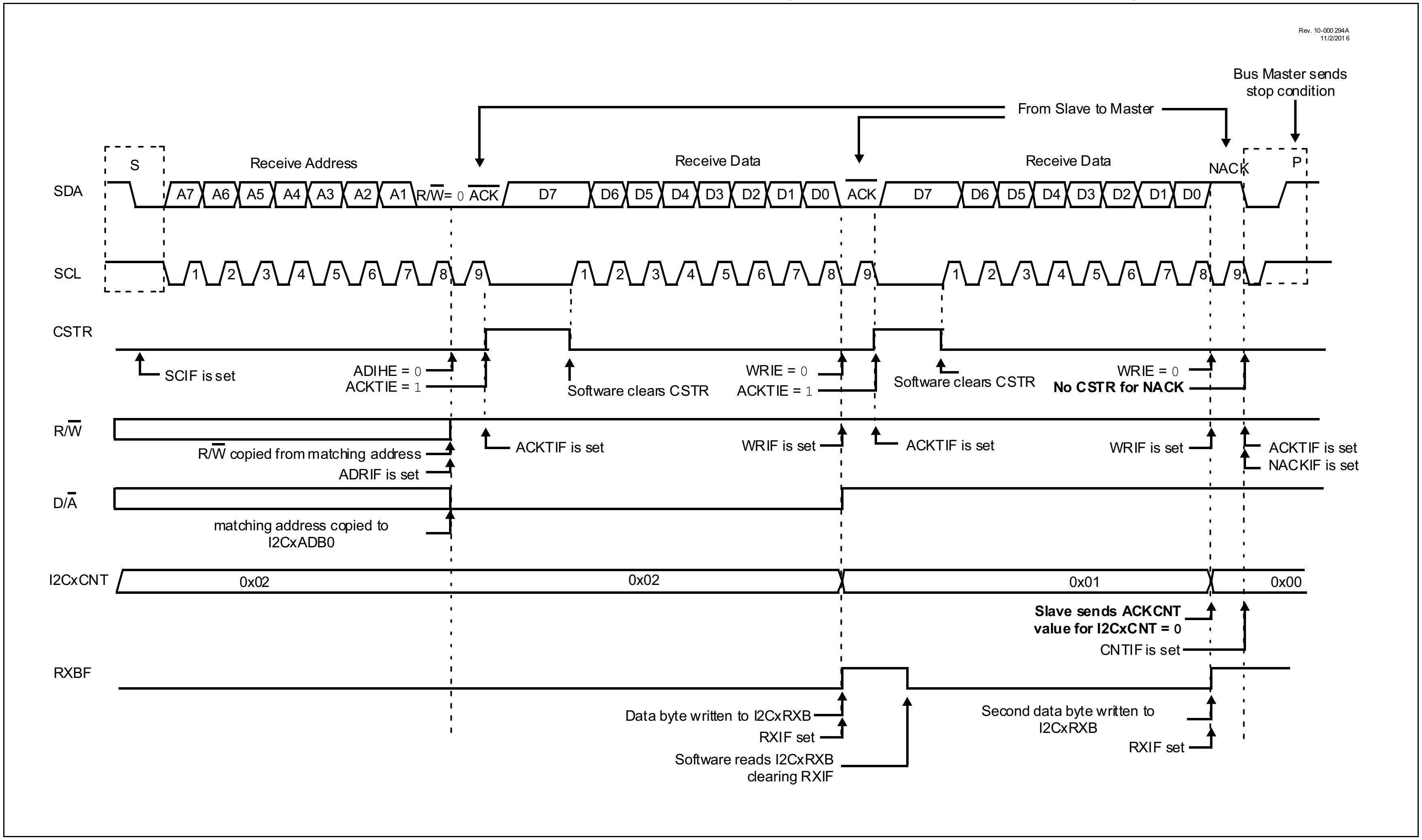
bit 1 **TXUIF**: Transmitter Underflow Interrupt Flag bit

1 = Slave Data transfer started when TXBE = 1 and TXR = 1

0 = No interrupt pending

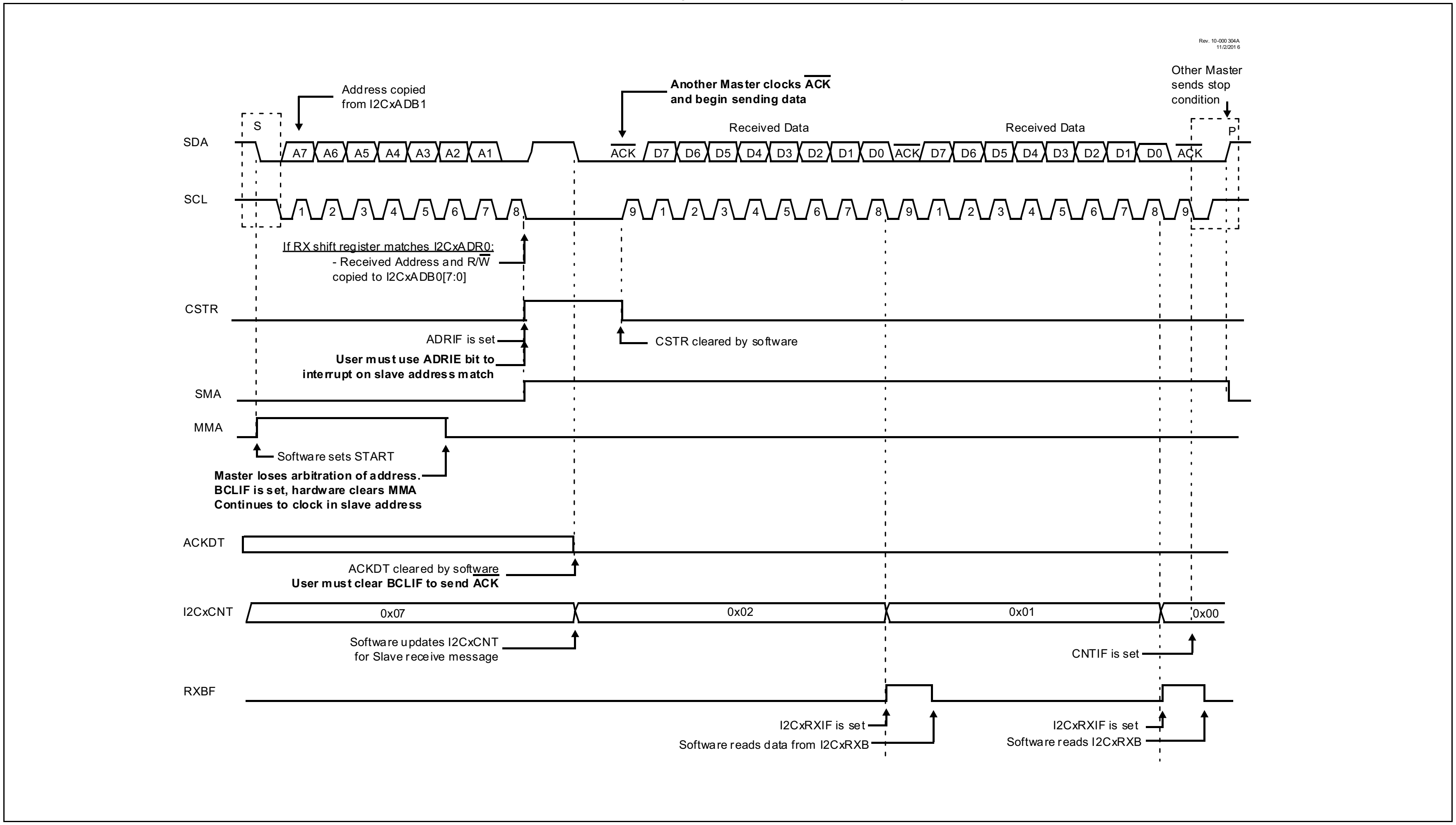
bit 0 **Unimplemented**: Read as ‘0’

FIGURE 33-7: I²C SLAVE, 7-BIT ADDRESS, RECEPTION WITH I2CxCNT (ACKTIE = 1, ADRIE = 0, WRIE = 0)



Preliminary

FIGURE 33-24: I²C MULTI-MASTER, 7-BIT ADDRESS, WRITE (ADRIE = 1, WRIE = 0)



REGISTER 33-2: I2CxCON1: I^{2}C 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 I2CCNT = 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 I2CCNT! = 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 I^{2}C 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 CSTRDIS = 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 I^{2}C hardware when bus errors are indicated in the I2CxSTAT1 or I2CxERR registers.

37.6.1 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 accumulator is a 16-bit wide register which can be accessed through the ADACCH:ADACCL register pair.

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 $2^{(accumulator\_width)}-1 = 18 = 262143$, 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 ADACLR bit in the ADCON2 register. Setting the ADACLR bit will also clear the ADAOV (Accumulator overflow) bit in the ADSTAT register, as well as the ADCNT register. The ADACLR bit is cleared by the hardware when accumulator clearing action is complete.

Note:	When ADC is operating from FRC, five FRC clock cycles are required to execute the ACC clearing operation.

The ADCRS <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. 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 37-3 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 37-3: 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
7	0.0078	-48.1

37.6.2 BASIC MODE

Basic mode (ADMD = `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.

37.6.3 ACCUMULATE MODE

In Accumulate mode (ADMD = `001`), after every conversion, the ADC result is added to the ADACC register. The ADACC register is right-shifted by the value of the ADCRS bits in the ADCON2 register. This right-shifted value is copied in to the ADFLT 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 37.6.7 “Threshold Comparison”**) and the ADTIF interrupt may trigger.

37.6.4 AVERAGE MODE

In Average mode (ADMD = `010`), the ADACC registers accumulate with each ADC sample, much as in Accumulate mode, and the ADCNT register increments with each sample. The ADFLT register is also updated with the right-shifted value of the ADACC register. The value of the ADCRS 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.

REGISTER 37-4: 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 37-2.
2: When PSIS = 0
3: When PSIS = 1.

39.7 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 45-15 and Table 45-17 for more details.

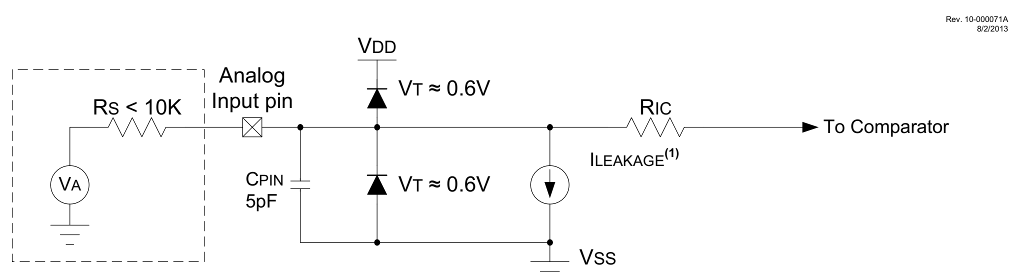
39.8 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 39-3. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to V_{DD} and V_{SS}. The analog input, therefore, must be between V_{SS} and V_{DD}. 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.

A maximum source impedance of 10 kΩ is recommended for the analog sources. 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 39-3: ANALOG INPUT MODEL



Legend:

C_{PIN}	= Input Capacitance
$I_{LEAKAGE}$	= Leakage Current at the pin due to various junctions
R_{IC}	= Interconnect Resistance
R_S	= Source Impedance
V_A	= Analog Voltage
V_T	= Threshold Voltage

Note 1: See **Section 45.0 "Electrical Specifications"**.

BTG — Bit Toggle f

Syntax:	BTG f, b {,a}
Operands:	$0 \le f \le 255$ $0 \le b < 7$ $a \in [0,1]$
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 ≤ 95 (5Fh). See **Section 42.2.3 "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 → 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)`

TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3980h	IPR0	IOCIP	CRCIP	SCANIP	NVMIP	CSWIP	OSFIP	HLVDIP	SWIP	147
397Fh - 397Eh	—	Unimplemented								—
397Dh	SCANTRIG	—	—	—	—	TSEL				216
397Ch	SCANCON0	EN	TRIGEN	SGO	—	—	MREG	BURSTMD	BUSY	212
397Bh	SCANHADRU	—	—	HADR						214
397Ah	SCANHADRH	HADR								215
3979h	SCANHADRL	HADR								215
3978h	SCANLADRU	—	—	LADR						213
3977h	SCANLADRH	LADR								213
3976h	SCANLADRL	LADR								214
3975h - 396Ah	—	Unimplemented								—
3969h	CRCCON1	DLEN				PLEN				208
3968h	CRCCON0	EN	CRCGO	BUSY	ACCM	—	—	SHIFTM	FULL	208
3967h	CRCXORH	X15	X14	X13	X12	X11	X10	X9	X8	211
3966h	CRCXORL	X7	X6	X5	X4	X3	X2	X1	—	211
3965h	CRCSHIFTH	SHFT15	SHFT14	SHFT13	SHFT12	SHFT11	SHFT10	SHFT9	SHFT8	210
3964h	CRCSHIFTL	SHFT7	SHFT6	SHFT5	SHFT4	SHFT3	SHFT2	SHFT1	SHFT0	210
3963h	CRCACCH	ACC15	ACC14	ACC13	ACC12	ACC11	ACC10	ACC9	ACC8	209
3962h	CRCACCL	ACC7	ACC6	ACC5	ACC4	ACC3	ACC2	ACC1	ACC0	210
3961h	CRCDATH	DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8	209
3960h	CRCDATL	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	209
395Fh	WDTTMR	WDTTMR					STATE	PSCNT		175
395Eh	WDTPSH	PSCNT								174
395Dh	WDTPSL	PSCNT								174
395Ch	WDTCON1	—	WDTCS			—	WINDOW			173
395Bh	WDTCON0	—	—	WDTPS					SEN	172
395Ah - 38A0h	—	Unimplemented								—
389Fh	IVTADU	AD								158
389Eh	IVTADH	AD								158
389Dh	IVTADL	AD								158
389Ch - 3891h	—	Unimplemented								—
3890h	PRODH_SHAD	PRODH								115
388Fh	PRODL_SHAD	PRODL								115
388Eh	FSR2H_SHAD	—	—	FSR2H						115
388Dh	FSR2L_SHAD	FSR2L								115
388Ch	FSR1H_SHAD	—	—	FSR1H						115
388Bh	FSR1L_SHAD	FSR1L								115
388Ah	FSR0H_SHAD	—	—	FSR0H						115
3889h	FSR0L_SHAD	FSR0L								115
3888h	PCLATU_SHAD	—	—	—	PCU					115
3887h	PCLATH_SHAD	PCH								115
3886h	BSR_SHAD	—	—	BSR						115
3885h	WREG_SHAD	WREG								115
3884h	STATUS_SHAD	—	$\overline{TO}$	$\overline{PD}$	N	OV	Z	DC	C	115
3883h	SHADCON	—	—	—	—	—	—	—	SHADLO	159
3882h	BSR_CSHAD	—	—	BSR						47
3881h	WREG_CSHAD	WREG								47

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not present in LF devices.