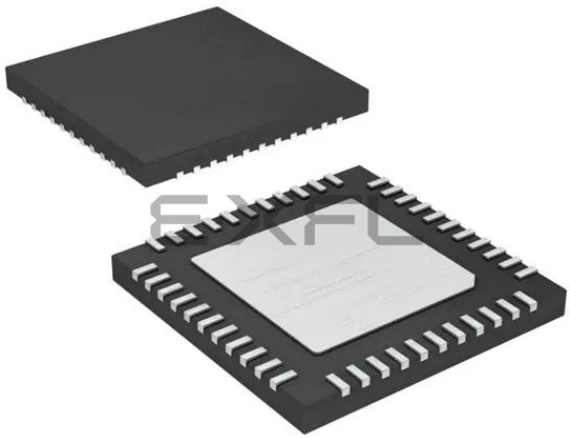


Welcome to E-XFL.COM

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	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f46k42-i-ml

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

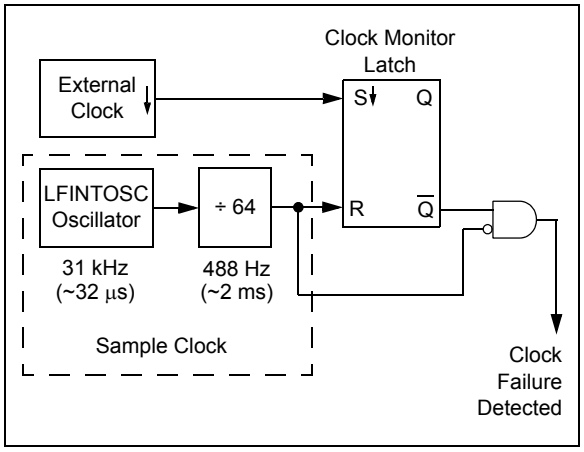
Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register (Figure 4-1). This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt, 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.

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.

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.

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.

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.

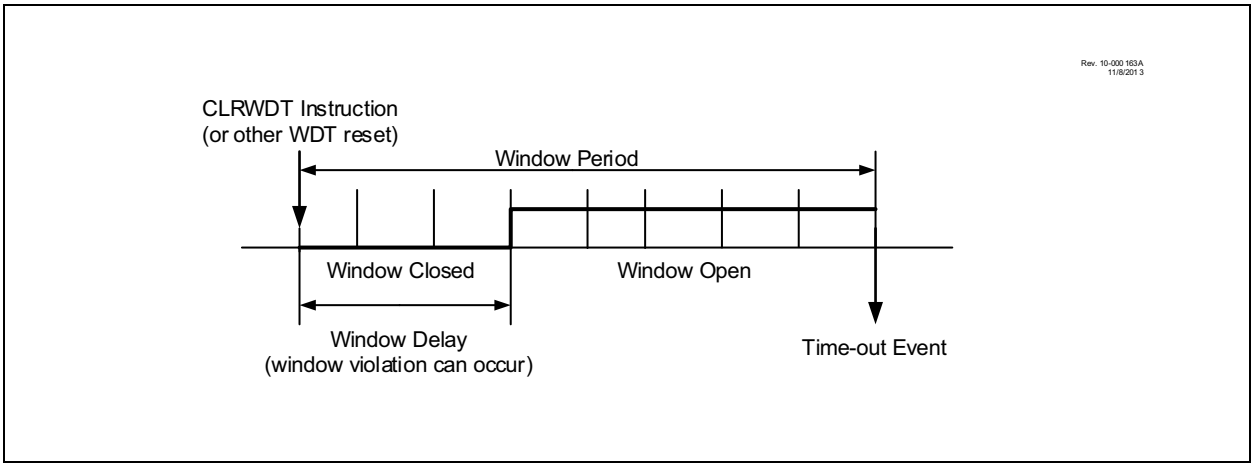
The WWDT remains clear until the Oscillator Start-up Timer (OST) completes, if enabled. See **Section 7.2.1.3 "Oscillator Start-up Timer (OST)"** for more information on the OST.

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

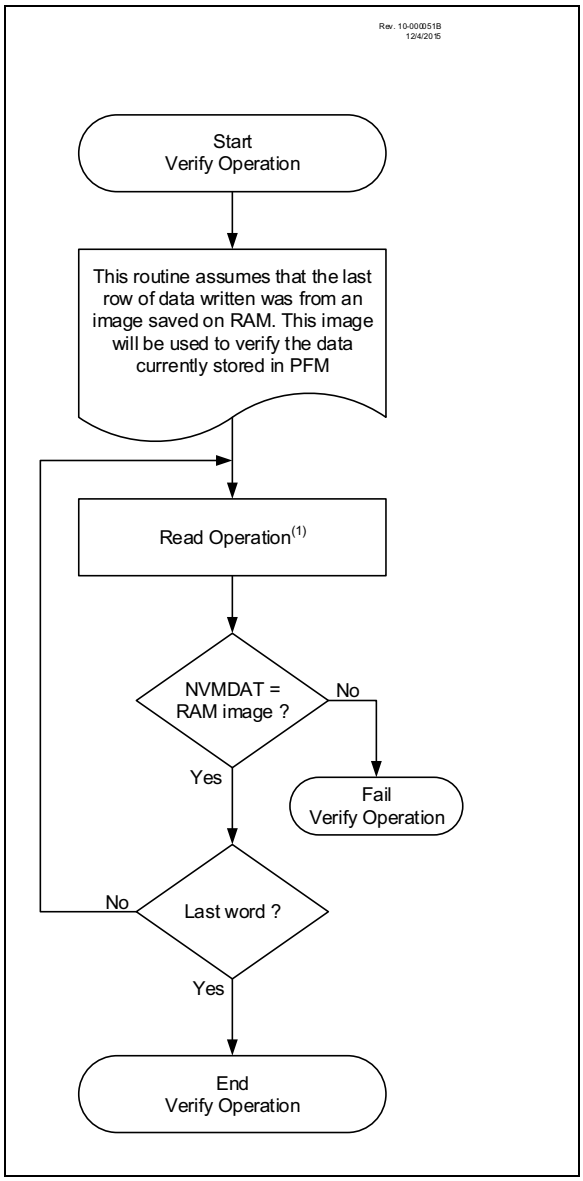
FIGURE 11-2: WINDOW PERIOD AND DELAY



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 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.

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	270
INLVLD(6)	INLVLD7	INLVLD6	INLVLD5	INLVLD4	INLVLD3	INLVLD2	INLVLD1(5)	INLVLD0(5)	270
INLVLF(7)	INLVLF7	INLVLF6	INLVLF5	INLVLF4	INLVLF3	INLVLF2	INLVLF1	INLVLF0	270
INLVLE	—	—	—	—	INLVLE3	—	—	—	270
RB1I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>		271
RB2I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>		271
RC3I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>		271
RC4I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>		271
RD0I2C(6)	—	SLEW	PU<1:0>		—	—	TH<1:0>		271
RD1I2C(6)	—	SLEW	PU<1:0>		—	—	TH<1:0>		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.

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

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 to 32 input signals and, through the use of configurable gates, reduces the 32 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

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.

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.

The CERIF checksum interrupt flag is not active in any mode other than LIN.

31.14 Collision Detection

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)

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.

FIGURE 32-9: CLOCKING DETAIL-MASTER MODE, CKE = 0, SMP = 1

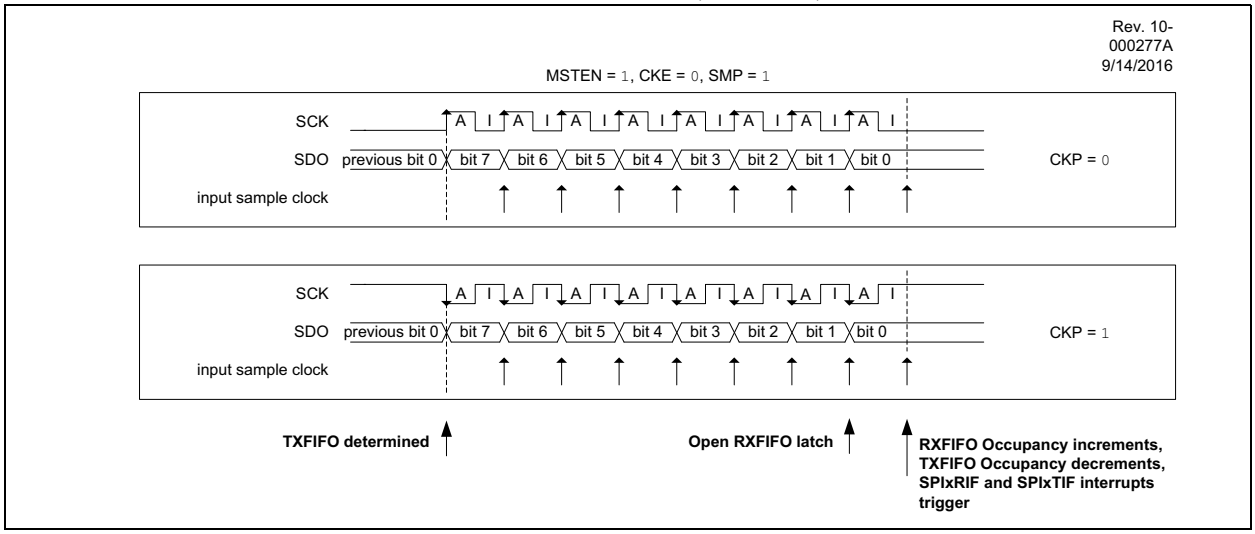
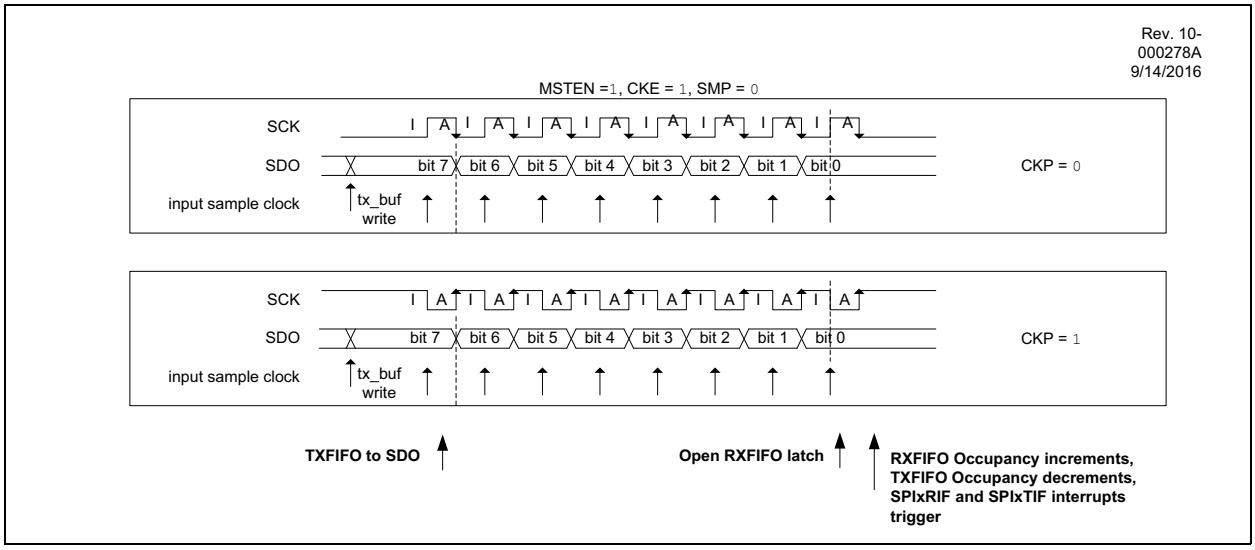


FIGURE 32-10: CLOCKING DETAIL-MASTER MODE, CKE = 1, SMP = 0



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.

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/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/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/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 (TXRE) is set and the new data is discarded. When TXRE 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.

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/W bit = d 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/W = 1.
3. Master sends out the 9th SCL pulse for ACK, master hardware clocks in 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 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 ACK value to slave. If I2CxCNT = 0, master hardware clocks out ACKCNT as 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.

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"**.

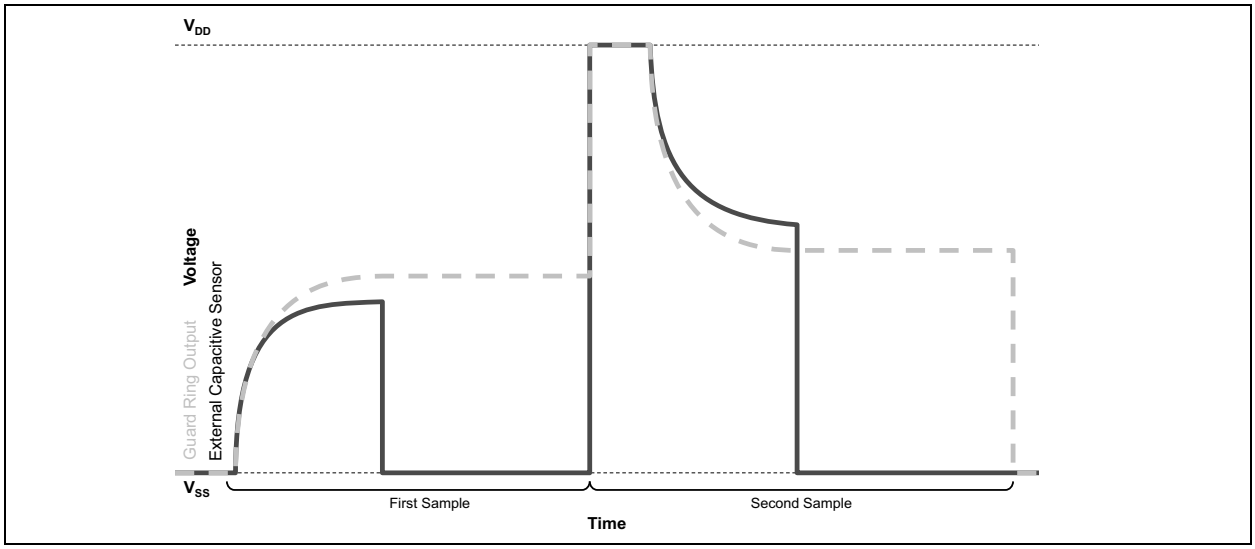
EXAMPLE 36-1: ADC CONVERSION

```
/*This code block configures the ADC
for polling, VDD and VSS references, FRC
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; //FRC 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
    }
}
```

FIGURE 36-9: DIFFERENTIAL CVD WITH GUARD RING OUTPUT WAVEFORM



36.5.5 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 36-10.

FIGURE 39-2: LOW-VOLTAGE DETECT OPERATION (INTL = 1)

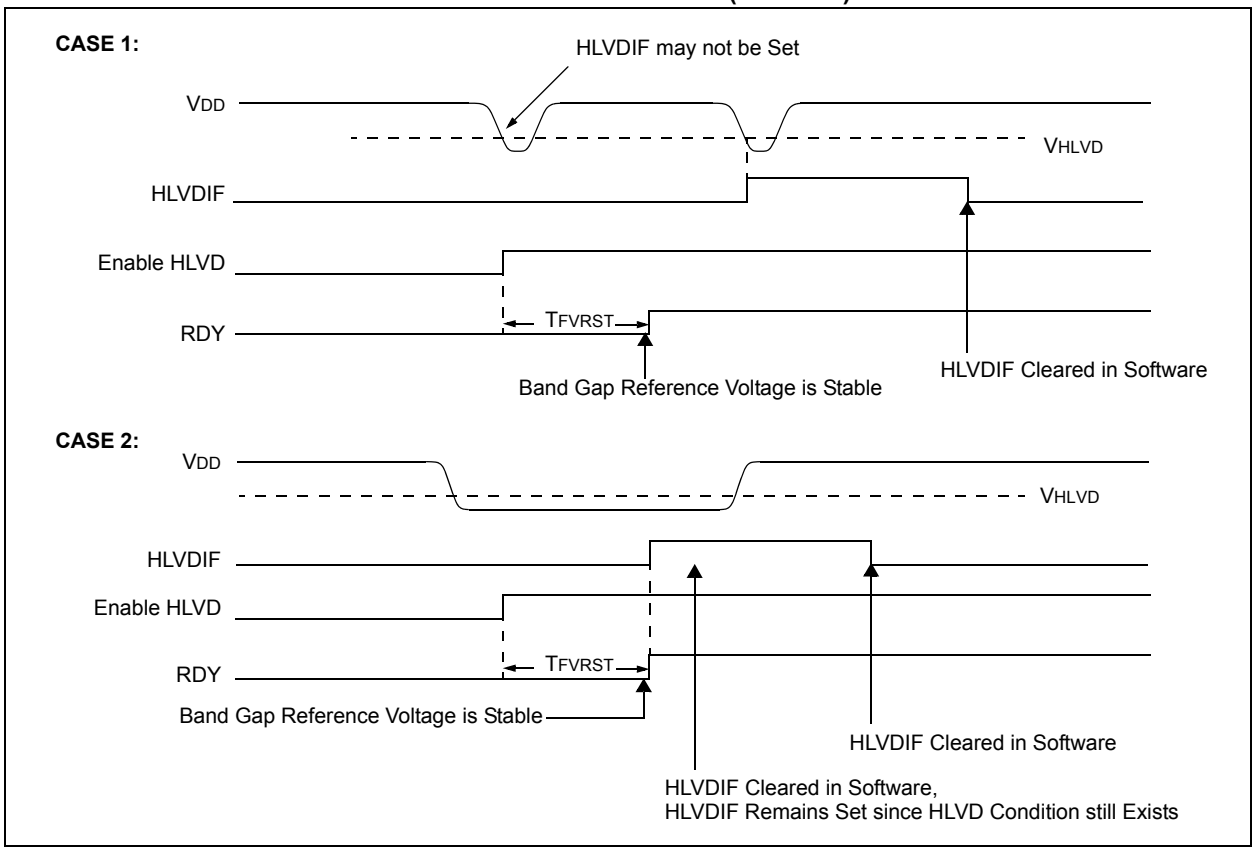
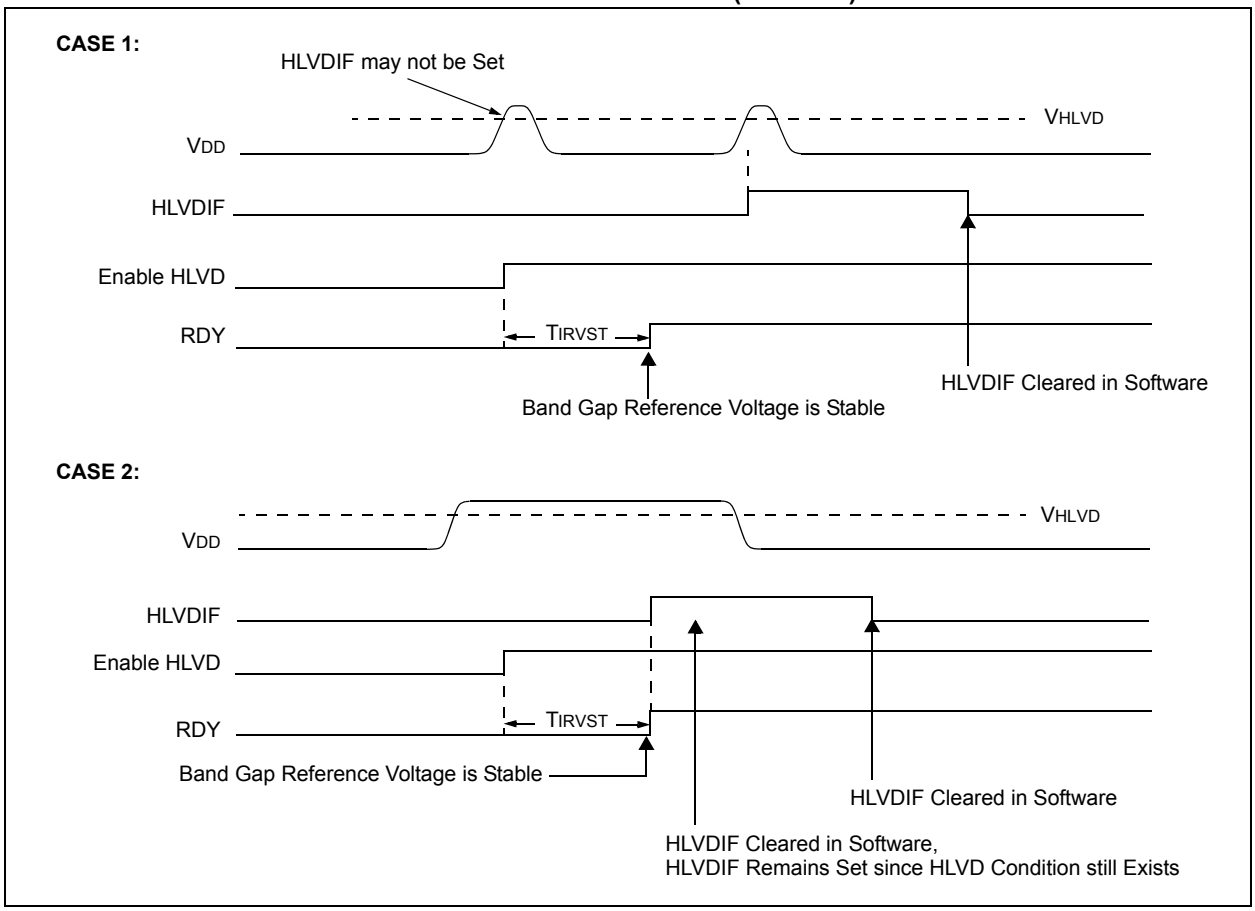


FIGURE 39-3: HIGH-VOLTAGE DETECT OPERATION (INTH = 1)



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, VA, the HLVD logic generates an interrupt at time, TA. 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 TB. This would give the application a time window, represented by the difference between TA and TB, to safely exit.

FIGURE 39-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION

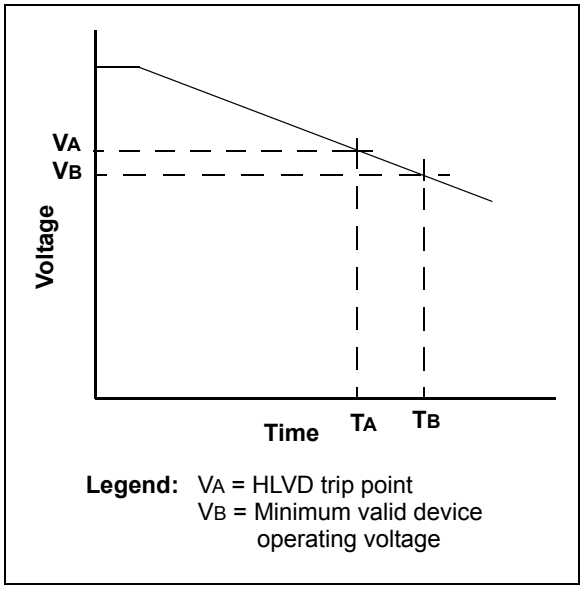


TABLE 44-24: SPI MODE REQUIREMENTS (SLAVE MODE)

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	SS↑ to 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.

PRELIMINARY