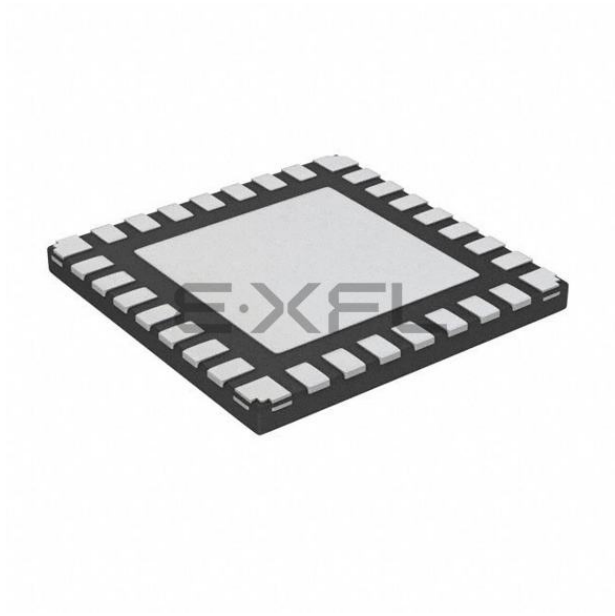


Welcome to E-XFL.COM

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	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 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-UQFN Exposed Pad
Supplier Device Package	28-UQFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f26k83-i-mx

eXtreme Low-Power (XLP) Features

- Sleep mode: 60 nA @ 1.8V, typical
- Windowed Watchdog Timer: 720 nA @ 1.8V, typical
- Secondary Oscillator: 580 nA @ 32 kHz
- Operating Current:
  - 4 uA @ 32 kHz, 1.8V, typical
  - 45 uA/MHz @ 1.8V, typical

Digital Peripherals

- Three 8-Bit Timers (TMR2/4/6) with Hardware Limit Timer (HLT):
  - Hardware monitoring and Fault detection
- Four 16-Bit Timers (TMR0/1/3/5)
- Four Configurable Logic Cell (CLC):
  - Integrated combinational and sequential logic
- Three Complementary Waveform Generators (CWGs):
  - Rising and falling edge dead-band control
  - Full-bridge, half-bridge, 1-channel drive
  - Multiple signal sources
  - Programmable dead band
  - Fault-shutdown input
- Four Capture/Compare/PWM (CCP) modules
- Four 10-bit Pulse-Width Modulators (PWMs)
- Numerically Controlled Oscillator (NCO):
  - Generates true linear frequency control
  - High resolution using 20-bit accumulator and 20-bit increment values
- DSM: Data Signal Modulator:
  - Multiplex two carrier clocks, with glitch prevention feature
  - Multiple sources for each carrier
- Programmable CRC with Memory Scan:
  - Reliable data/program memory monitoring for fail-safe operation (e.g., Class B)
  - Calculate CRC over any portion of program memory
- Two UART Modules:
  - Modules are Asynchronous, RS-232, RS-485 compatibility.
  - One of the UART modules supports LIN Master and Slave, DMX mode, DALI Gear and Device protocols
  - Automatic and user-timed BREAK period generation
  - DMA Compatible
  - Automatic checksums
  - Programmable 1, 1.5, and two Stop bits
  - Wake-up on BREAK reception
- One SPI module:
  - Configurable length bytes
  - Configurable length data packets
  - Receive-without-transmit option
  - Transmit-without-receive option
  - Transfer byte counter
  - Separate Transmit and Receive Buffers with 2-byte FIFO and DMA capabilities
- CAN module:
  - Conforms to CAN 2.0B Active Specification
  - Three operating modes: Legacy (compatible with existing PIC18CXX8/FXX8 CAN modules), Enhanced mode, and FIFO mode.
  - Message bit rates up to 1 Mbps
  - DeviceNet™ data byte filter support
  - Six programmable receive/transmit buffers
  - Three dedicated transmit buffers
  - Two dedicated receive buffers
  - 16 Full, 29-bit acceptance filters with dynamic association
  - Three full, 29-bit acceptance masks
  - Automatic remote frame handling
  - Advanced error management features.
- Two I^2C modules, SMBus, PMBus™ compatible:
  - Dedicated Address, Transmit and Receive buffers
  - Bus Collision Detection with arbitration
  - Bus time-out detection and handling
  - Multi-Master mode
  - Separate Transmit and Receive Buffers with 2-byte FIFO and DMA capabilities
  - I^2C, SMBus 2.0 and SMBus 3.0, and 1.8V input level selections
- Device I/O Port Features:
  - 25 I/O pins (PIC18(L)F25K83)
  - One input-only pin
  - Individually programmable I/O direction, open-drain, slew rate, weak pull-up control
  - Interrupt-on-change
  - Three External Interrupt Pins
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Two Signal Measurement Timer (SMT):
  - 24-bit timer/counter with prescaler

2.0 GUIDELINES FOR GETTING STARTED WITH PIC18(L)F25/ 26K83 MICROCONTROLLERS

2.1 Basic Connection Requirements

Getting started with the PIC18(L)F25/26K83 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins (see **Section 2.2 "Power Supply Pins"**)
- $\overline{\text{MCLR}}$ pin (see **Section 2.3 "Master Clear (MCLR) Pin"**)

These pins must also be connected if they are being used in the end application:

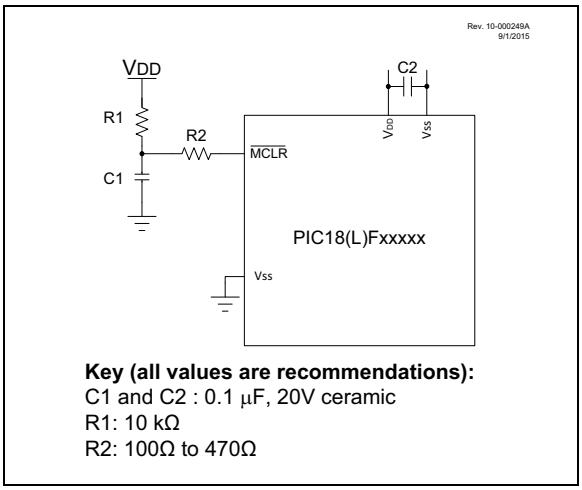
- ICSPCLK/ICSPDAT pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see **Section 2.4 "ICSP™ Pins"**)
- OSCI and OSCO pins when an external oscillator source is used (see **Section 2.5 "External Oscillator Pins"**)

Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

The minimum mandatory connections are shown in Figure 2-1.

FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS



Key (all values are recommendations):
C1 and C2 : 0.1 μF, 20V ceramic
R1: 10 kΩ
R2: 100Ω to 470Ω

2.2 Power Supply Pins

2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins (VDD and VSS) is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1 μF (100 nF), 10-20V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01 μF to 0.001 μF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 μF in parallel with 0.001 μF).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

2.2.2 TANK CAPACITORS

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7 μF to 47 μF.

TABLE 4-6: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 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'.

© 2017 Microchip Technology Inc.

TABLE 4-10: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 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'.

© 2017 Microchip Technology Inc.

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.

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

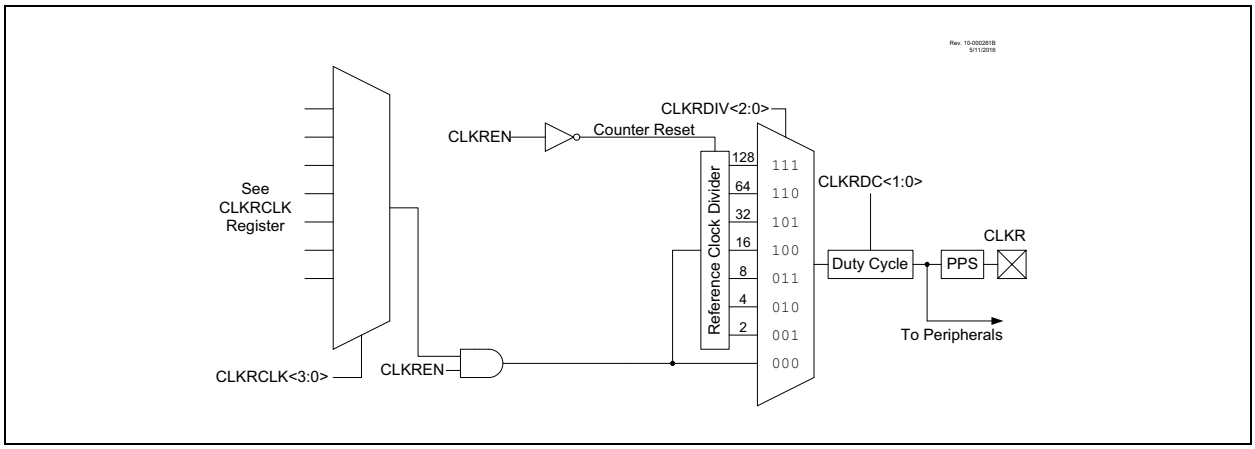
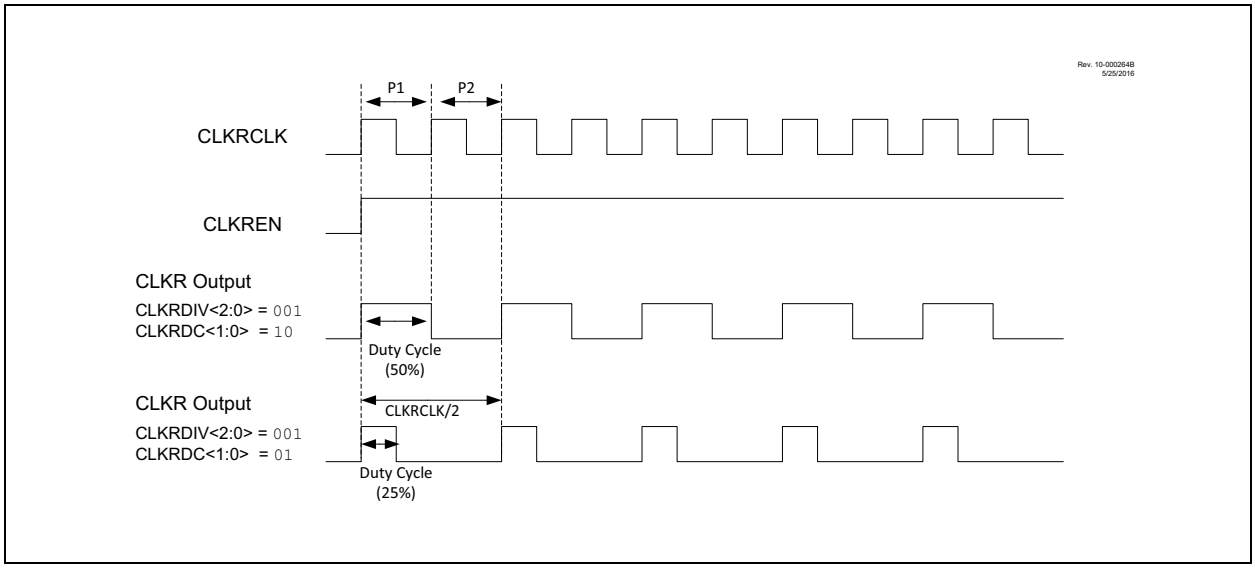


FIGURE 8-2: CLOCK REFERENCE TIMING



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.

- When 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.

9.6 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 illustrates 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.

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

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	272
IOCxN	IOCxN7	IOCxN6	IOCxN5	IOCxN4	IOCxN3	IOCxN2	IOCxN1	IOCxN0	272
IOCxP	IOCxP7	IOCxP6	IOCxP5	IOCxP4	IOCxP3	IOCxP2	IOCxP1	IOCxP0	272

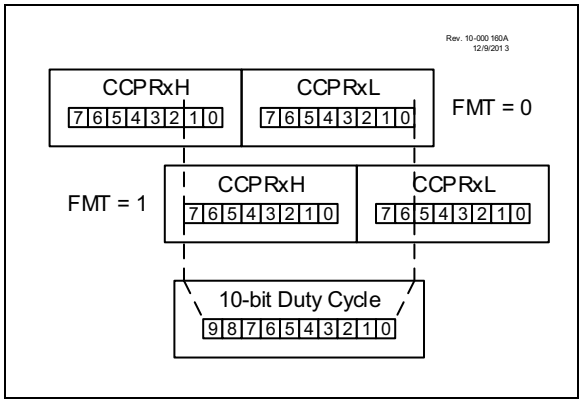
Legend: — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

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.

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

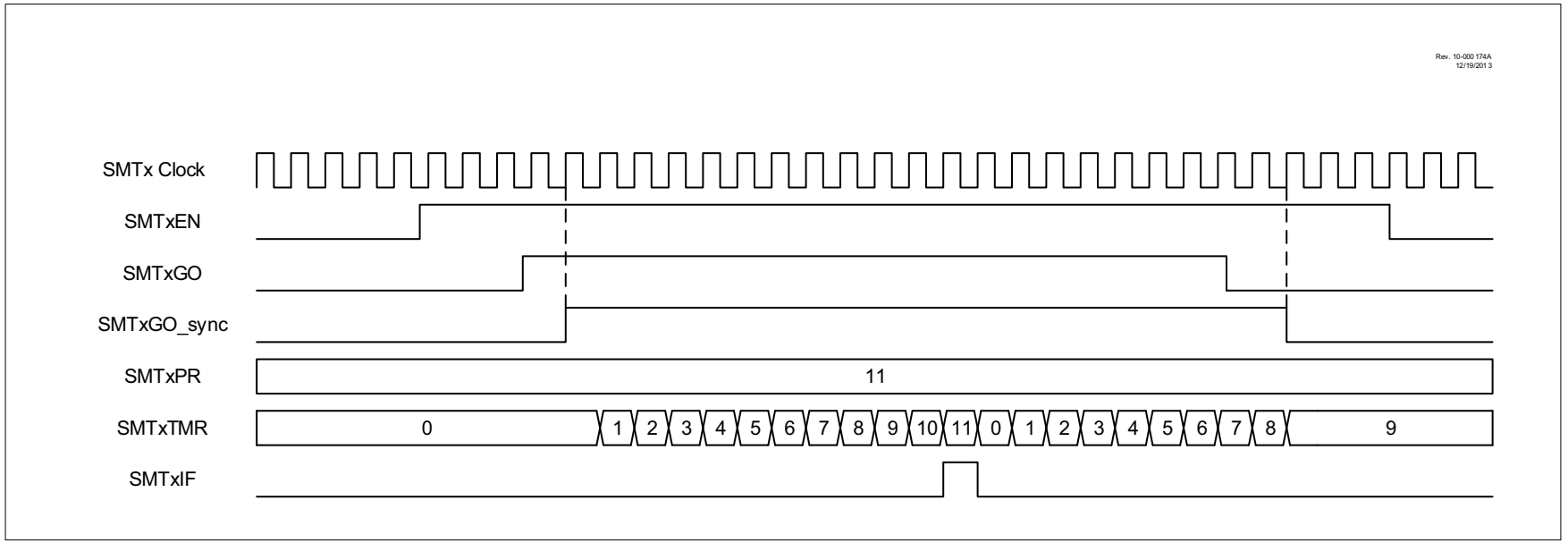
bit 7-6 **P8TSEL<1:0>:** PWM8 Timer Selection bits
11 = PWM8 based on TMR6
10 = PWM8 based on TMR4
01 = PWM8 based on TMR2
00 = Reserved

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

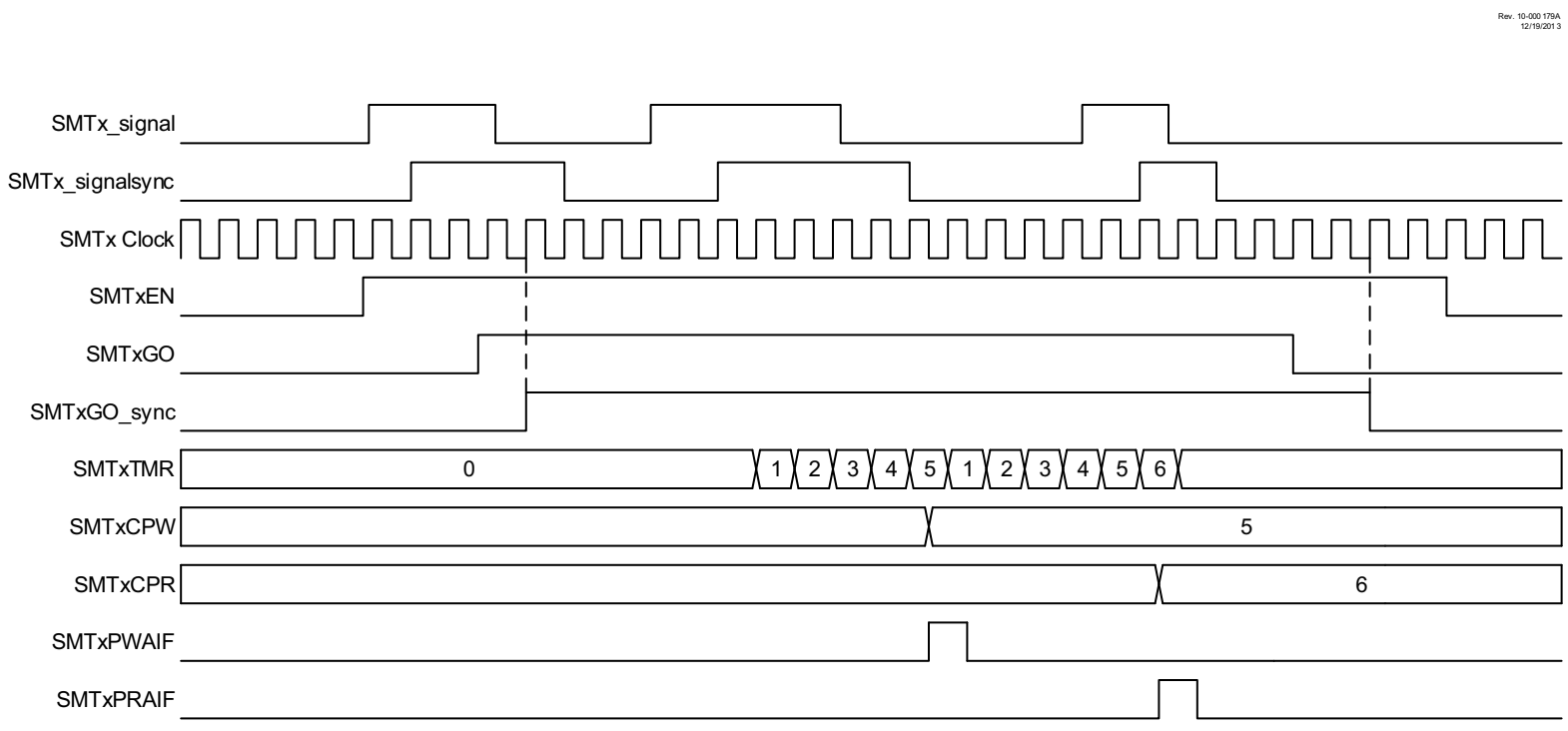
bit 1-0 **P5TSEL<1:0>:** PWM5 Timer Selection bits
11 = PWM5 based on TMR6
10 = PWM5 based on TMR4
01 = PWM5 based on TMR2
00 = Reserved

FIGURE 25-3: TIMER MODE TIMING DIAGRAM



© 2017 Microchip Technology Inc.

FIGURE 25-9: HIGH AND LOW MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM



© 2017 Microchip Technology Inc.

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.

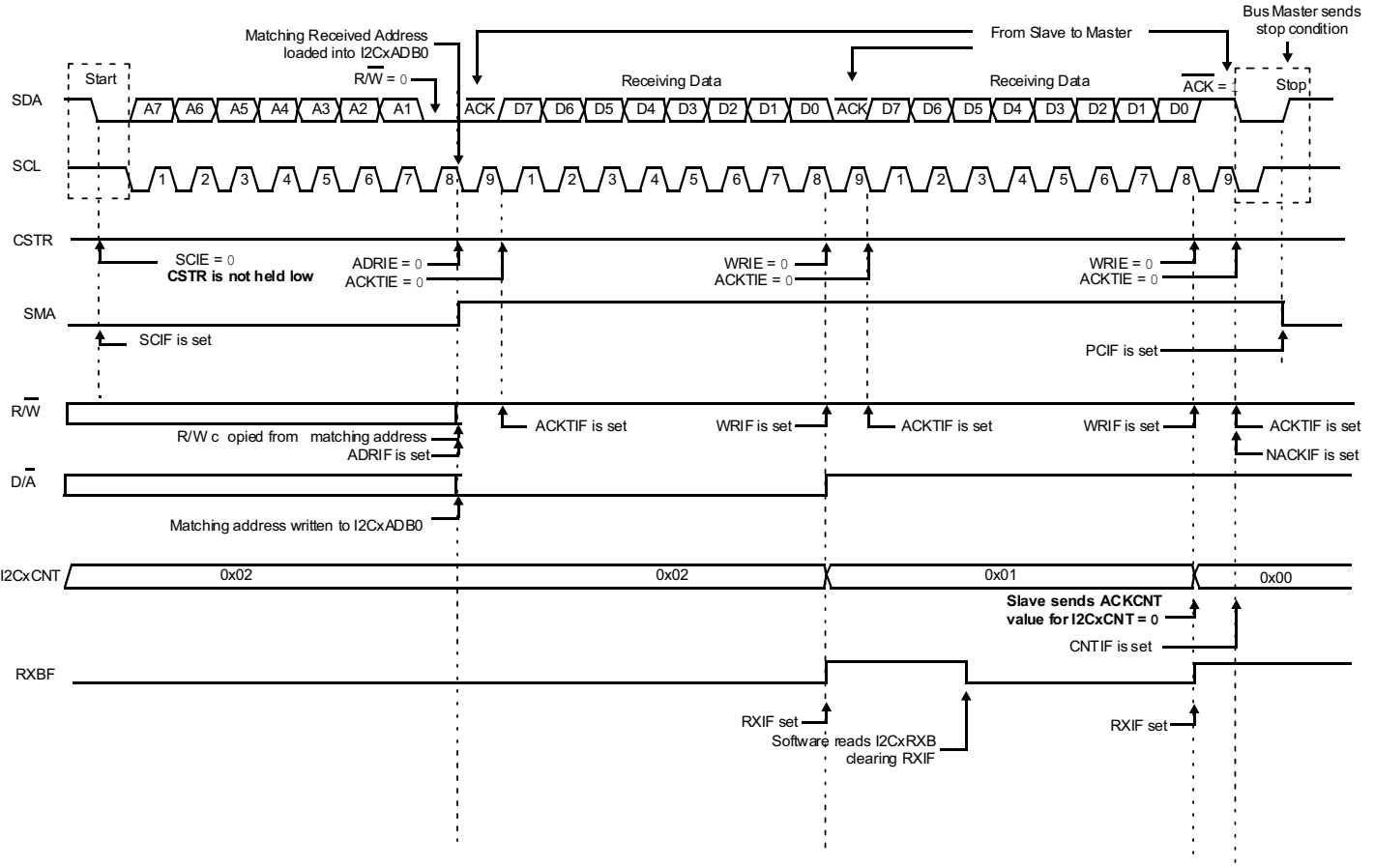
32.8.3.4 Receiver Overflow and Transmitter Underflow Interrupts

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.

FIGURE 33-6: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (ACKTIE = 0, ADRIE = 0, WRIE = 0)



© 2017 Microchip Technology Inc.

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.

42.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F25/26K83 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDULNK and SUBULNK) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 42-3. Detailed descriptions are provided in **Section 42.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 42-1 apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

42.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets ("[ ]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 42.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }").

MOVSS **Move Indexed to Indexed**

Syntax: MOVSS [z_s], [z_d]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: ((FSR2) + z_s) → ((FSR2) + z_d)

Status Affected: None

Encoding:

1st word (source)	1110	1011	1zzz	$zzzz_s$
2nd word (dest.)	1111	xxxx	xzzz	$zzzz_d$

Description: The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets 'z_s' or 'z_d', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).
The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	Determine dest addr	Determine dest addr	Write to dest reg

Example: MOVSS [05h], [06h]

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h
After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

PUSHL **Store Literal at FSR2, Decrement FSR2**

Syntax: PUSHL k

Operands: $0 \le k \le 255$

Operation: k → (FSR2),
FSR2 – 1 → FSR2

Status Affected: None

Encoding:

1111	1010	kkkk	kkkk

Description: The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process data	Write to destination

Example: PUSHL 08h

Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h