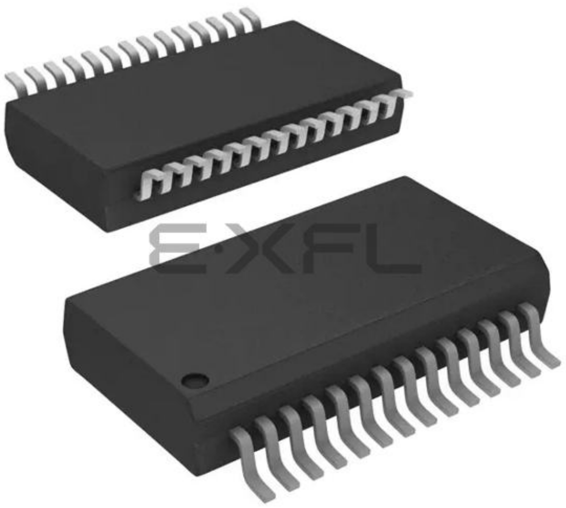


Welcome to E-XFL.COM

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	25
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
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-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf25k42-i-ss

TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH CPU

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
ISRPR	—	—	—	—	—	ISRPR2	ISRPR1	ISRPR0	21
MAINPR	—	—	—	—	—	MAINPR2	MAINPR1	MAINPR0	21
DMA1PR	—	—	—	—	—	DMA1PR2	DMA1PR1	DMA1PR0	21
DMA2PR	—	—	—	—	—	DMA2PR2	DMA2PR1	DMA2PR0	22
SCANPR	—	—	—	—	—	SCANPR2	SCANPR1	SCANPR0	22
PRLOCK	—	—	—	—	—	—	—	PRLOCKED	22

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

4.4 PIC18 Instruction Cycle

4.4.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the program counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the instruction register during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 4-2.

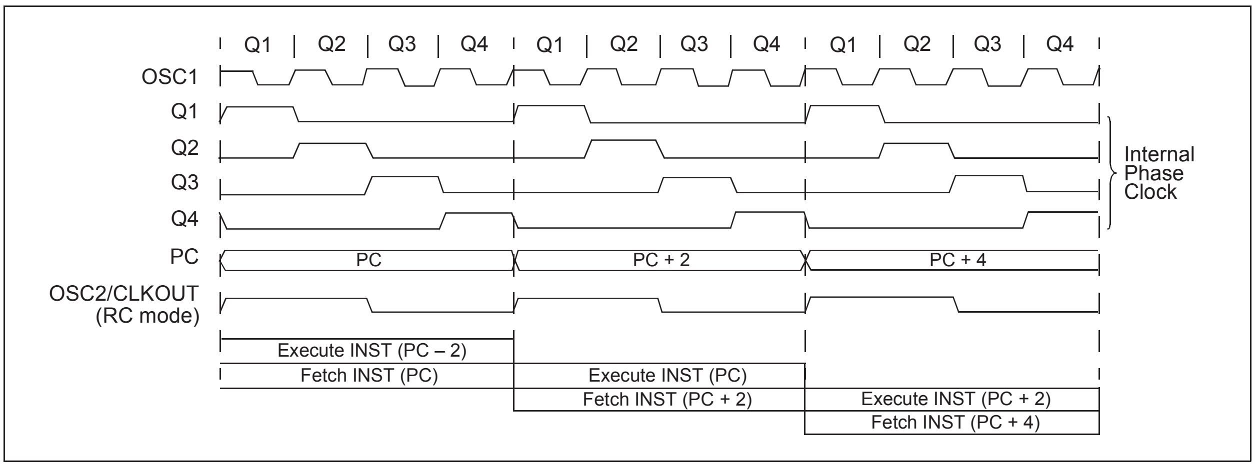
4.4.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., `GOTO`), then two cycles are required to complete the instruction (Example 4-3).

A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 4-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 4-3: INSTRUCTION PIPELINE FLOW

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

Note: There are some instructions that take multiple cycles to execute. Refer to **Section 43.0 “Instruction Set Summary”** for details.

4.5.2 ACCESS BANK

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

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

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

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

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

4.5.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM, which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 0000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

4.5.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (3FFFh) and extend downward to occupy Bank 56 through 63 (3800h to 3FFFh). A list of these registers is given in Table 4-3 to Table 4-10.

The SFRs can be classified into two sets: those associated with the “core” device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and Interrupt registers are described in their respective chapters, while the ALU’s STATUS register is described later in this section. Registers related to the operation of a peripheral feature are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as ‘0’s.

Preliminary

TABLE 4-9: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F24/25K42 DEVICES BANK 57

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

Legend: Unimplemented data memory locations and registers, read as '0'.
Note 1: Only on PIC18F24/25K42 parts.

9.5 Register Definitions: Oscillator Control

REGISTER 9-1: OSCCON1: OSCILLATOR CONTROL REGISTER1

U-0	R/W-f/f	R/W-f/f	R/W-f/f	R/W-q/q	R/W-q/q	R/W-q/q	R/W-q/q
—	NOSC<2:0>			NDIV<3:0>			
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	f = determined by Configuration bit setting
		q = Reset value is determined by hardware

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **NOSC<2:0>:** New Oscillator Source Request bits(1,2,3)
The setting requests a source oscillator and PLL combination per Table 9-1.
POR value = RSTOSC (Register 5-1).

bit 3-0 **NDIV<3:0>:** New Divider Selection Request bits(2,3)
The setting determines the new postscaler division ratio per Table 9-1.

Note 1: The default value (f/f) is determined by the RSTOSC Configuration bits. See Table 9-2below.
2: If NOSC is written with a reserved value (Table 9-1), the operation is ignored and neither NOSC nor NDIV is written.
3: When CSWEN = 0, this register is read-only and cannot be changed from the POR value.

TABLE 9-2: DEFAULT OSCILLATOR SETTINGS

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

Note 1: EXTOSC must meet the PLL specifications (Table 46-9).

11.12 Register Definitions: Interrupt Control

REGISTER 11-1: INTCON0: INTERRUPT CONTROL REGISTER 0

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

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

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

If IPEN = 0:

GIE:

1 = Enables all unmasked interrupts
0 = Disables all interrupts

If IPEN = 1:

GIEH:

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

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

If IPEN = 0:

Reserved, read as ‘0’

If IPEN = 1:

GIEL:

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

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

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

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

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

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

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

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

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

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

15.3.8 ERASING THE DATA EEPROM MEMORY

Data EEPROM Memory can be erased by writing 0xFF to all locations in the Data EEPROM Memory that needs to be erased.

EXAMPLE 15-7: DATA EEPROM REFRESH ROUTINE

```
        CLRF     NVMADRL                ; Start at address 0
        BCF      NVMCON1, CFGS          ; Set for memory
        BCF      NVMCON1, EEPGD         ; Set for Data EEPROM
        BCF      INTCON0, GIE           ; Disable interrupts
        BSF      NVMCON1, WREN          ; Enable writes
Loop                                    ; Loop to refresh array
        BSF      NVMCON1, RD            ; Read current address
        MOVLW    55h                    ;
        MOVWF    NVMCON2                ; Write 55h
        MOVLW    0AAh                   ;
        MOVWF    NVMCOM2                ; Write 0AAh
        BSF      NVMCON1, WR            ; Set WR bit to begin write
        BTFSC    NVMCON1, WR            ; Wait for write to complete
        BRA      $-2
        INCFSZ   NVMADRL, F             ; Increment address
        BRA      LOOP                   ; Not zero, do it again

        BCF      NVMCON1, WREN          ; Disable writes
        BSF      INTCON0, GIE           ; Enable interrupts
```

REGISTER 17-21: DMAxDCNTH – DMAx DESTINATION COUNT HIGH REGISTER

U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	DCNT<11:8>			
bit 7							bit 0

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

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

bit 3-0 **DCNT<11:8>:** Current Destination Byte Count

REGISTER 17-22: DMAxSIRQ – DMAx START INTERRUPT REQUEST SOURCE SELECTION REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	SIRQ6	SIRQ5	SIRQ4	SIRQ3	SIRQ2	SIRQ1	SIRQ0
bit 7							bit 0

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

bit 7 **Unimplemented**: Read as ‘0’

bit 6-0 **DMAxSIRQ<6:0>:** DMAx Start Interrupt Request Source Selection bits
Please refer to Table 17-2 for more information.

REGISTER 17-23: DMAxAIRQ – DMAx ABORT INTERRUPT REQUEST SOURCE SELECTION REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	AIRQ6	AIRQ5	AIRQ4	AIRQ3	AIRQ2	AIRQ1	AIRQ0
bit 7							bit 0

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

bit 7 **Unimplemented**: Read as ‘0’

bit 6-0 **DMAxAIRQ<6:0>:** DMAx Interrupt Request Source Selection bits
Please refer to Table 17-2 for more information.

19.8 Register Definitions: PPS Input Selection

REGISTER 19-1: xxxPPS: PERIPHERAL xxx INPUT SELECTION

U-0	U-0	U-0	R/W-m/u(1)	R/W-m/u(1)	R/W-m/u(1)	R/W-m/u(1)	R/W-m/u(1)
—	—	—	xxxPPS<4:0>				
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	-n/n = Value at POR and BOR/Value at all other Resets
u = Bit is unchanged	x = Bit is unknown	q = value depends on peripheral
‘1’ = Bit is set	U = Unimplemented bit, read as ‘0’	m = value depends on default location for that input
‘0’ = Bit is cleared		

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

bit 4-3 **xxxPPS<4:3>:** Peripheral xxx Input PORTx Pin Selection bits
See Table 19-1 for the list of available ports and default pin locations.
11 = Reserved
10 = PORTC
01 = PORTB
00 = PORTA

bit 2-0 **xxxPPS<2:0>:** Peripheral xxx Input PORTx Pin Selection bits
111 = Peripheral input is from PORTx Pin 7 (Rx7)
110 = Peripheral input is from PORTx Pin 6 (Rx6)
101 = Peripheral input is from PORTx Pin 5 (Rx5)
100 = Peripheral input is from PORTx Pin 4 (Rx4)
011 = Peripheral input is from PORTx Pin 3 (Rx3)
010 = Peripheral input is from PORTx Pin 2 (Rx2)
001 = Peripheral input is from PORTx Pin 1 (Rx1)
000 = Peripheral input is from PORTx Pin 0 (Rx0)

Note 1: The Reset value ‘m’ of this register is determined by device default locations for that input.

TABLE 19-1: PPS INPUT REGISTER DETAILS

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	Input Available from Selected PORTx		
Interrupt 0	INT0PPS	RB0	5’b0 1000	A	B	—
Interrupt 1	INT1PPS	RB1	5’b0 1001	A	B	—
Interrupt 2	INT2PPS	RB2	5’b0 1010	A	B	—
Timer0 Clock	T0CKIPPS	RA4	5’b0 0100	A	B	—
Timer1 Clock	T1CKIPPS	RC0	5’b1 0000	A	—	C
Timer1 Gate	T1GPPS	RB5	5’b0 1101	—	B	C
Timer3 Clock	T3CKIPPS	RC0	5’b1 0000	—	B	C
Timer3 Gate	T3GPPS	RC0	5’b1 0000	A	—	C
Timer5 Clock	T5CKIPPS	RC2	5’b1 0010	A	—	C
Timer5 Gate	T5GPPS	RB4	5’b0 1100	—	B	C
Timer2 Clock	T2INPPS	RC3	5’b1 0011	A	—	C
Timer4 Clock	T4INPPS	RC5	5’b1 0101	—	B	C

22.0 TIMER0 MODULE

Timer0 module is an 8/16-bit timer/counter with the following features:

- 16-bit timer/counter
- 8-bit timer/counter with programmable period
- Synchronous or asynchronous operation
- Selectable clock sources
- Programmable prescaler
- Programmable postscaler
- Operation during Sleep mode
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals

FIGURE 22-1: BLOCK DIAGRAM OF TIMER0

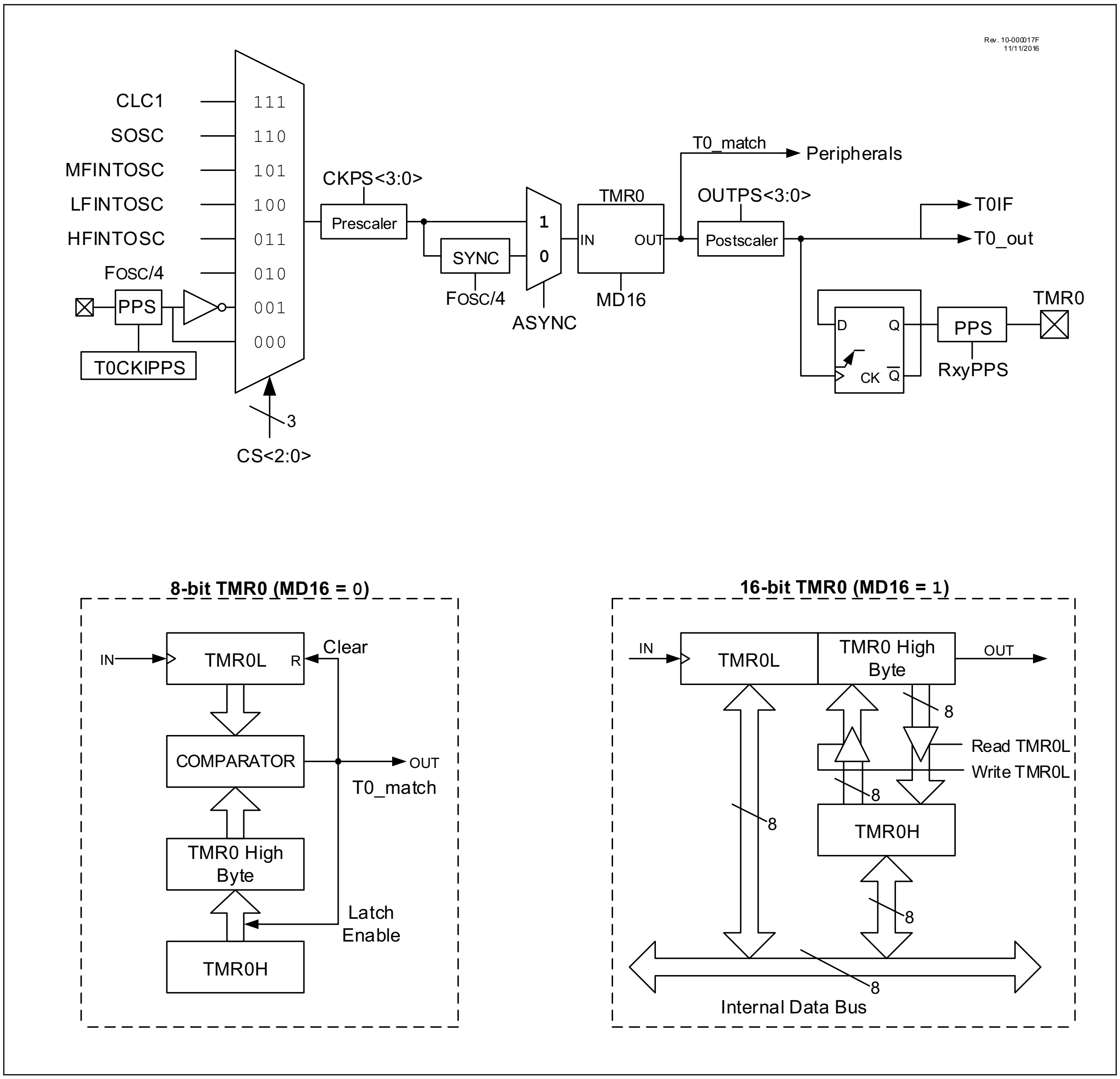


FIGURE 28-14: CWG SHUTDOWN BLOCK DIAGRAM

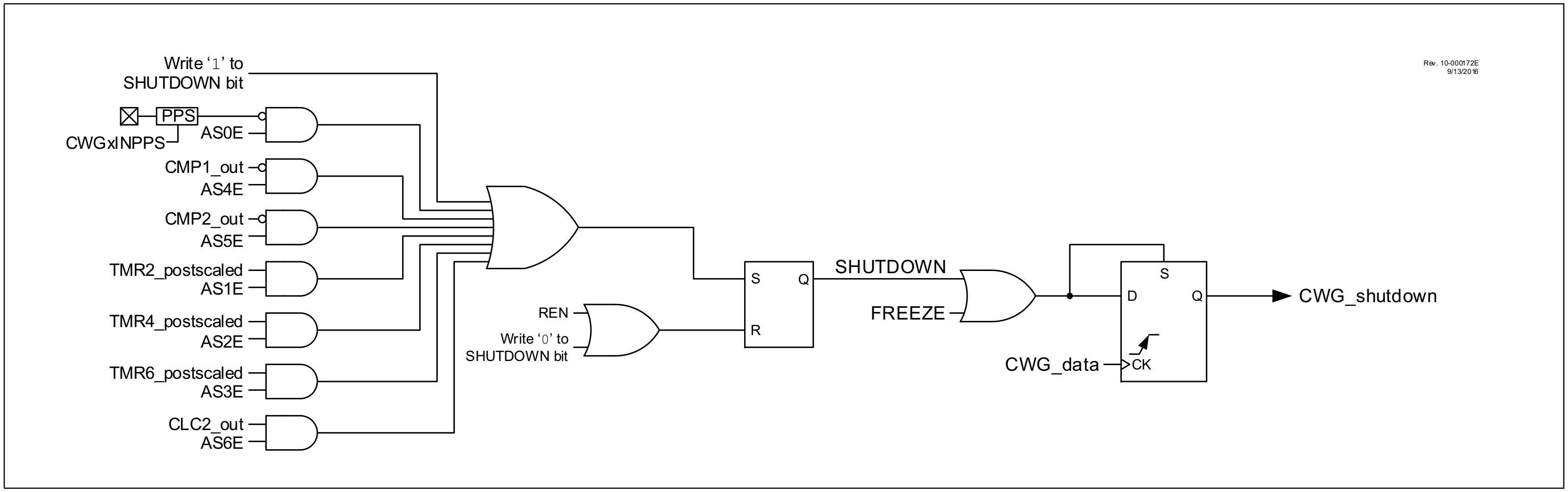
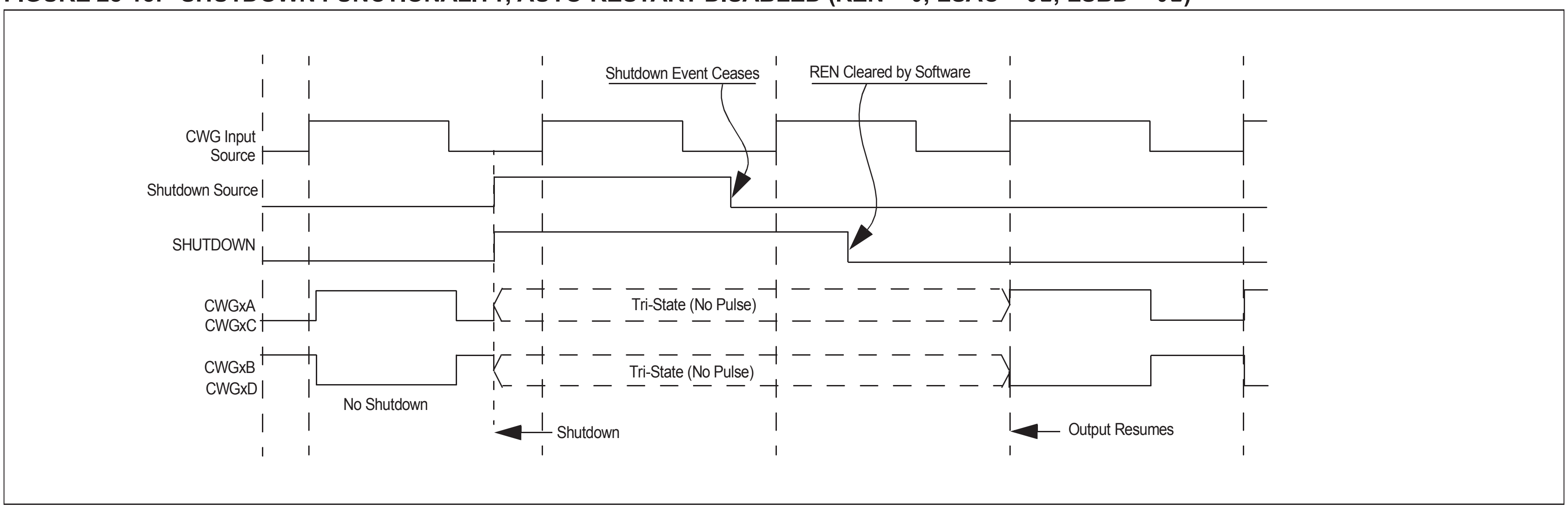


FIGURE 28-15: SHUTDOWN FUNCTIONALITY, AUTO-RESTART DISABLED (REN = 0, LSAC = 01, LSBD = 01)



33.2.2 UART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 33-2. The data is received on the RX pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 4 or 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the UART receiver. The FIFO registers and RSR are not directly accessible by software. Access to the received data is via the UxRXB register.

33.2.2.1 Enabling the Receiver

The UART receiver is enabled for asynchronous operation by configuring the following control bits:

- RXEN = `1`
- MODE<3:0> = 0h through 3h
- UxBRGH:L = desired baud rate
- RXPPS = code for desired input pin
- Input pin ANSEL bit = `0`
- ON = `1`

All other UART control bits are assumed to be in their default state.

Setting the RXEN bit in the UxCON0 register enables the receiver circuitry of the UART. Setting the MODE<3:0> bits in the UxCON0 register configures the UART for the desired asynchronous mode. Setting the ON bit in the UxCON1 register enables the UART. The TRIS bit corresponding to the selected RX I/O pin must be set to configure the pin as an input.

Note:	If the RX function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

33.2.2.2 Receiving Data

Data is recovered from the bit stream by timing to the center of the bits and sampling the input level. In High-Speed mode, there are four BRG clocks per bit and only one sample is taken per bit. In Normal-Speed mode, there are 16 BRG clocks per bit and three samples are taken per bit.

The receiver data recovery circuit initiates character reception on the falling edge of the Start bit. The Start bit, is always a ‘`0`’. The Start bit is qualified in the middle of the bit. In Normal-Speed mode only, the Start bit is also qualified at the leading edge of the bit. The following paragraphs describe the majority detect sampling of Normal-Speed mode.

The falling edge starts the baud rate generator (BRG) clock. The input is sampled at the first and second BRG clocks.

If both samples are high then the falling edge is deemed a glitch and the UART returns to the Start bit detection state without generating an error.

If either sample is low, the data recovery circuit continues counting BRG clocks and takes samples at clock counts 7, 8, and 9. When less than two samples are low, the Start bit is deemed invalid and the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit.

When two or more samples are low, the Start bit is deemed valid and the data recovery continues. After a valid Start bit is detected, the BRG clock counter continues and resets at count 16. This is the beginning of the first data bit.

The data recovery circuit counts BRG clocks from the beginning of the bit and takes samples at clocks 7, 8, and 9. The bit value is determined from the majority of the samples. The resulting ‘`0`’ or ‘`1`’ is shifted into the RSR.The BRG clock counter continues and resets at count 16. This sequence repeats until all data bits have been sampled and shifted into the RSR.

After all data bits have been shifted in, the first Stop bit is sampled. Stop bits are always a ‘`1`’. If the bit sampling determines that a ‘`0`’ is in the Stop bit position, the framing error is set for this character. Otherwise, the framing error is cleared for this character. See **Section 33.2.2.4 “Receive Framing Error”** for more information on framing errors.

33.2.2.3 Receive Interrupts

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the UART receive FIFO. The UxRXIF interrupt flag in the respective PIR register is set at this time, provided it is not being suppressed.

The UxRXIF is suppressed by any of the following:

- FERIF if FERIE is set
- PERIF if PERIE is set

This suspends DMA transfer of data until software processes the error and reads UxRXB to advance the FIFO beyond the error.

UxRXIF interrupts are enabled by setting all of the following bits:

- UxRXIE, Interrupt Enable bit in the PIE register
- GIE, Global Interrupt Enable bits in the INTCON0 register

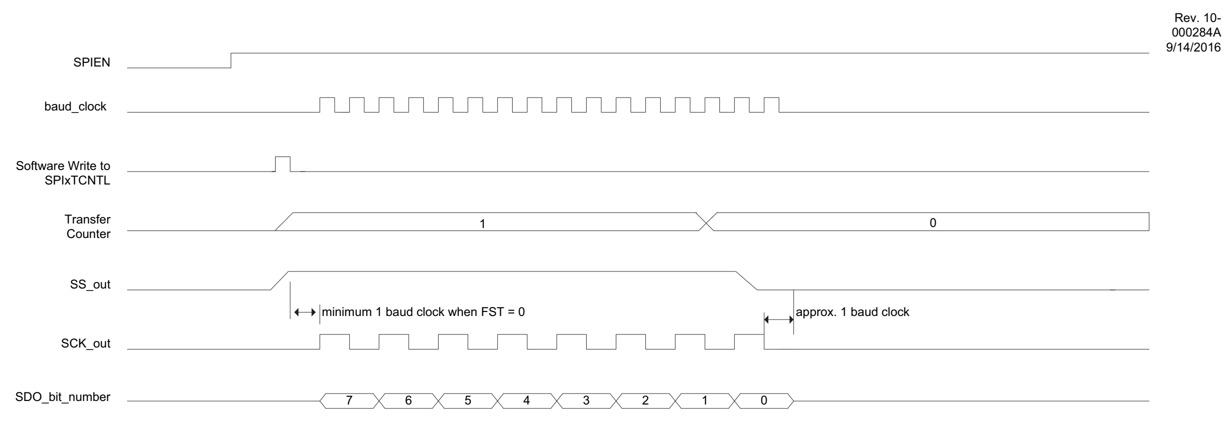
The UxRXIF interrupt flag bit will be set when not suppressed and there is an unread character in the FIFO, regardless of the state of interrupt enable bits. Reading the UxRXB register will transfer the top character out of the FIFO and reduce the FIFO contents by one. The UxRXIF interrupt flag bit is read-only, it cannot be set or cleared by software.

34.5.5 MASTER MODE SLAVE SELECT CONTROL

34.5.5.1 Hardware Slave Select Control

This SPI module allows for direct hardware control of a Slave Select output. The Slave Select output SS(out) is controlled both directly, through the SSET bit of SPIxCON2, as well indirectly by the hardware while the transfer counter is non-zero (see **Section 34.4 "Transfer Counter"**). SS(out) is steered by the PPS registers to pins (see **Section 19.2 "PPS Outputs"**) and its polarity is controlled by the SSP bit of SPIxCON1. Setting the SSET bit will also assert SS(out). Clearing the SSET bit will leave SS(out) to be controlled by the Transfer Counter. When the Transfer Counter is loaded, the SPI module will automatically assert the SS. When the Transfer Counter decrements to zero, the SPI module will deassert SS either one baud period after the final SCK pulse of the final transfer (if CKE/SMP = `0/1`) or one half baud period otherwise (see Figure 34-6).

FIGURE 34-6: SPI MASTER SS OPERATION- CKE = `0`, BMODE = `1`, TCWIDTH = `0`, SSP = `0`



Note: 1. *SDO bit number* illustrates the transmitted bit number, and is not intended to imply SDO (out) tristate operation.
2. Assumes SPIxTXB holds data when SPIxTCNTL is written.

34.5.5.2 Software Slave Select Control

Slave Select can also be controlled through software via a general purpose I/O pin. In this case, ensure that the pin in question is configured as a GPIO through PPS (see **Section 19.2 "PPS Outputs"**), and ensure that the pin is set as an output (clear the appropriate bit in the appropriate TRIS register). In this case, SSET will not affect the slave select, the Transfer Counter will not automatically control the slave select output, and all setting and clearing of the slave select output line must be directly controlled by software.

35.5.11 MASTER TRANSMISSION IN 10-BIT ADDRESSING MODE

This section describes the sequence of events for the I^2C module configured as an I^2C master in 10-bit Addressing mode and is transmitting data. Figure 35-21 is used as a visual reference for this description

1. If ABD = 0; i.e. Address buffers are enabled

Master software loads number of bytes to be transmitted in one sequence in I2CxCNT, high address byte of slave address in I2CxADB1 with R/W = 0, low address byte in I2CxADB0 and the first byte of data in I2CxTXB. Master software has to set the Start (S) bit to initiate communication.

If ABD = 1; i.e. Address buffers are disabled

Master software loads the number of bytes to be transmitted in one sequence in I2CxCNT and the high address byte of the slave address with R/W = 0 into the I2CxTXB register. Writing to the I2CxTXB will assert the start condition on the bus and sets the S bit. Software writes to the S bit are ignored in this case.

2. Master hardware waits for BFRE bit to be set; then shifts out the start and high address and waits for acknowledge.
3. If NACK, master hardware sends Stop.
4. If ABD = 0; i.e. Address buffer are enabled

If ACK, master hardware sends the low address byte from I2CxADB0.

If ABD = 1; i.e., Address buffer are disabled

If ACK, master hardware sets TXIF and MDR bits and the software has to write the low address byte into I2CxTXB. Writing to I2CxTXB sends the low address on the bus.

5. If TXBE = 1 and I2CxCNT! = 0, I2CxTXIF and MDR bits are set. Clock is stretched on 8th falling SCL edge till master software writes next data byte to I2CxTXB.
6. Master hardware sends ninth SCL pulse for ACK from slave and loads the shift register from I2CxTXB. I2CxCNT is decremented.
7. If slave sends a NACK, master hardware sends Stop and ends transmission.
8. If slave sends an ACK, master hardware outputs data in the shift register on SDA. I2CxCNT value is checked on the 8th falling SCL edge. If I2CxCNT = 0; master hardware sends 9th SCL pulse for ACK and CNTIF is set.
9. If I2CxCNT != 0; go to step 5.

REGISTER 35-14: I2CxADR2 – I²C ADDRESS 2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
bit 7							bit 0

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

bit 7-0 **ADR<7-0>:** Address 2 bits

MODE<2:0> = 000 | 110 - 7-bit Slave/Multi-Master Modes
ADR<7:1>:7-bit Slave Address

MODE<2:0> = 001 | 111 - 7-bit Slave/Multi-Master Modes with Masking
ADR<7:1>:7-bit Slave Address

MODE<2:0> = 010 - 10-Bit Slave Mode
ADR<7:0>: Eight Least Significant bits of second 10-bit address

MODE<2:0> = 011 - 10-Bit Slave Mode with Masking
MSK0<7-0>: The received address byte is masked, then compared to I2CxADR0

37.4 Minimum Operating VDD

When the temperature circuit is operated in Low range, the device may be operated at any operating voltage that is within specifications. When the temperature circuit is operated in High range, the device operating voltage, VDD, must be high enough to ensure that the temperature circuit is correctly biased.

Table 37-1 shows the recommended minimum VDD vs. Range setting.

TABLE 37-1: RECOMMENDED VDD vs. RANGE

Min.VDD, TSRNG = 1 (High Range)	Min. VDD, TSRNG = 0 (Low Range)
≥ 2.5	≥ 1.8

37.5 Temperature Indicator Range

The temperature indicator circuit operates in either High or Low range. The High range, selected by setting the TSRNG bit of the FVRCON register, provides a wider output voltage. This provides more resolution over the temperature range. High range requires a higher-bias voltage to operate and thus, a higher VDD is needed. The Low range is selected by clearing the TSRNG bit of the FVRCON register. The Low range generates a lower sensor voltage and thus, a lower VDD voltage is needed to operate the circuit.

The output voltage of the sensor is the highest value at -40°C and the lowest value at +125°C.

- **High Range:** The High range is used in applications with the reference for the ADC, VREF = 2.048V. This range may not be suitable for battery-powered applications.
- **Low Range:** This mode is useful in applications in which the VDD is too low for high-range operation. The VDD in this mode can be as low as 1.8V. VDD must, however, be at least 0.5V higher than the maximum sensor voltage depending on the expected low operating temperature.

37.6 DIA Information

DIA data provide ADC readings at two operating temperatures. DIA data is taken during factory testing and stored within the device. The 90°C reading alone allows single-point calibration as described in Section 37.2.1, Calibration, by solving Equation 37-1 for TOFFSET.

Refer to **Section 6.0 "Device Information Area"** for more information on the data stored in the DIA and how to access them.

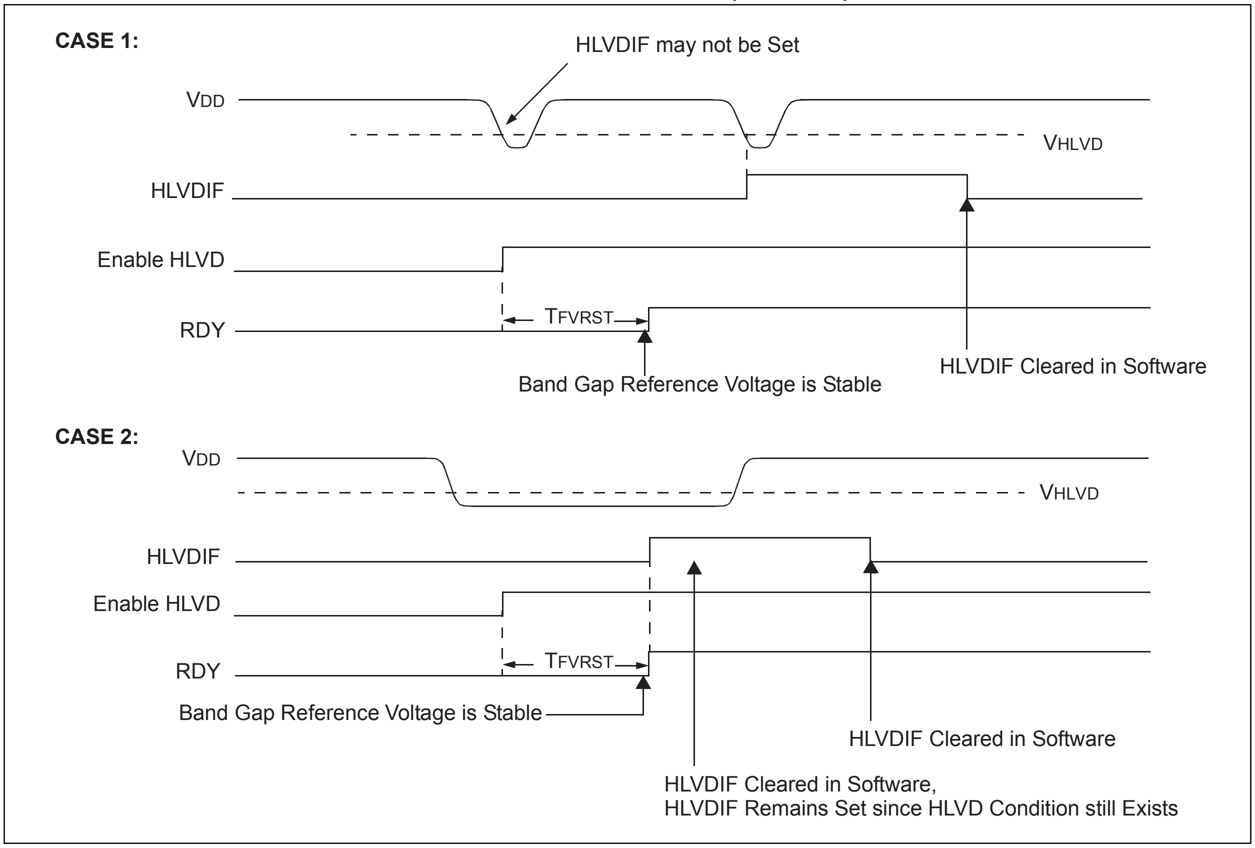
Note:	The lower temperature range (e.g., - 40°C) will suffer in accuracy because temperature conversion must extrapolate below the reference points, amplifying any measurement errors.

TABLE 37-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
FVRCON	EN	RDY	TSEN	TSRNG	CDAFVR<1:0>		ADFVR<1:0>		600

Legend: — = Unimplemented location, read as '0'. Shaded cells are unused by the temperature indicator module.

FIGURE 41-2: LOW-VOLTAGE DETECT OPERATION (INTL = 1)



BZ — Branch if Zero

Syntax:	BZ n
Operands:	$-128 \le n \le 127$
Operation:	if ZERO bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 0000 nnnn nnnn
Description:	If the ZERO bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BZ Jump`

Before Instruction
- PC = address (HERE)

After Instruction
- If ZERO = 1;
- PC = address (Jump)
- If ZERO = 0;
- PC = address (HERE + 2)

CALL — Subroutine Call

Syntax:	CALL k {,s}
Operands:	$0 \le k \le 1048575$ $s \in [0,1]$
Operation:	(PC) + 4 → TOS, k → PC<20:1>, if s = 1 (W) → WS, (Status) → STATUSS, (BSR) → BSRS
Status Affected:	None
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 110s $k_7$kkk kkkk_0 1111 k_{19}kkk kkkk kkkk_8
Description:	Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a 2-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE, 1`

Before Instruction
- PC = address (HERE)

After Instruction
- PC = address (THERE)
- TOS = address (HERE + 4)
- WS = W
- BSRS = BSR
- STATUSS = Status

SLEEP **Enter Sleep mode**

Syntax:	SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT postscaler, 1 → $\overline{\text{TO}}$, 0 → $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Encoding:	`0000` `0000` `0000` `0011`
Description:	The Power-down Status bit ($\overline{\text{PD}}$) is cleared. The Time-out Status bit ($\overline{\text{TO}}$) is set. Watchdog Timer and its postscaler are cleared. The processor is put into Sleep mode with the oscillator stopped.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to Sleep

Example: `SLEEP`

Before Instruction
$\overline{\text{TO}}$ = ?
$\overline{\text{PD}}$ = ?
After Instruction
$\overline{\text{TO}}$ = 1 †
$\overline{\text{PD}}$ = 0

† If WDT causes wake-up, this bit is cleared.

SUBFSR **Subtract Literal from FSR**

Syntax:	SUBFSR f, k
Operands:	0 ≤ k ≤ 63 f ∈ [0, 1, 2]
Operation:	FSR(f) – k → FSRf
Status Affected:	None
Encoding:	`1110` `1001` `ffkk` `kkkk`
Description:	The 6-bit literal ‘k’ is subtracted from the contents of the FSR specified by ‘f’.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

Example: `SUBFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 03DCh

45.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

45.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

45.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

45.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

45.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.