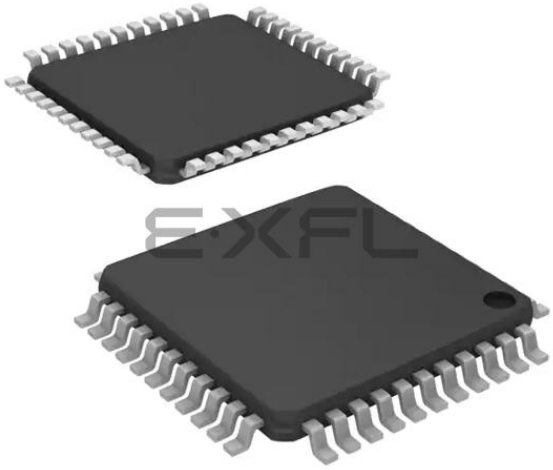


Welcome to E-XFL.COM

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	40MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f4480t-i-pt

On transitions from SEC_RUN mode to PRI_RUN mode, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switch back to the primary clock occurs (see Figure 4-2). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the clock. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run.

FIGURE 4-1: TRANSITION TIMING FOR ENTRY TO SEC_RUN MODE

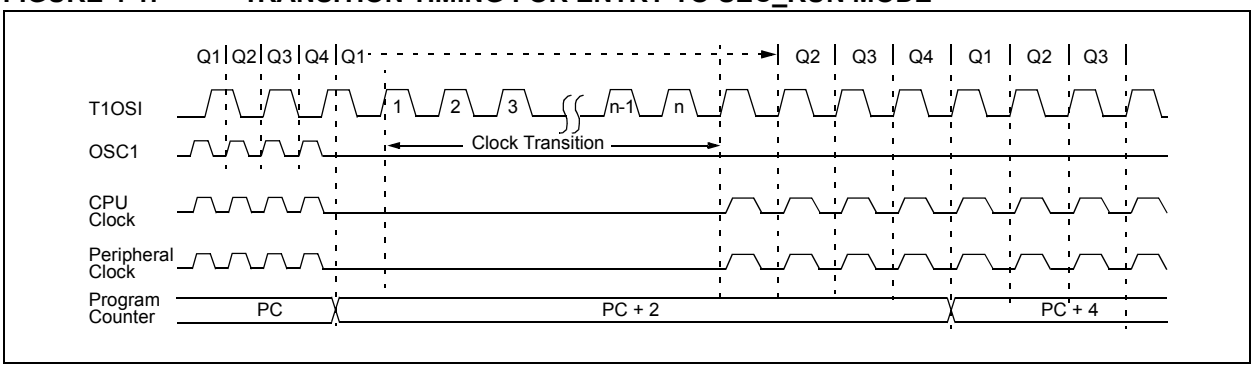
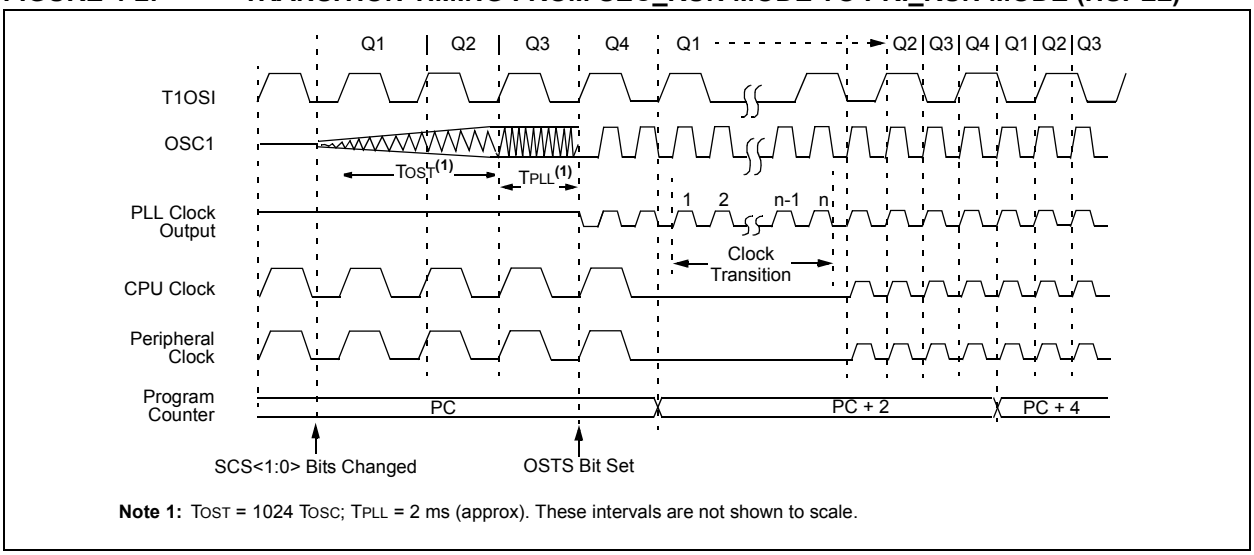


FIGURE 4-2: TRANSITION TIMING FROM SEC_RUN MODE TO PRI_RUN MODE (HSPLL)



Note 1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

4.2.3 RC_RUN MODE

In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator block using the INTOSC multiplexer; the primary clock is shut down. When using the INTRC source, this mode provides the best power conservation of all the Run modes, while still executing code. It works well for user applications which are not highly timing-sensitive or do not require high-speed clocks at all times.

If the primary clock source is the internal oscillator block (either INTRC or INTOSC), there are no distinguishable differences between PRI_RUN and RC_RUN modes during execution. However, a clock switch delay will occur during entry to, and exit from, RC_RUN mode. Therefore, if the primary clock source is the internal oscillator block, the use of RC_RUN mode is not recommended.

This mode is entered by setting SCS1 to '1'. Although it is ignored, it is recommended that SCS0 also be cleared; this is to maintain software compatibility with future devices. When the clock source is switched to the INTOSC multiplexer (see Figure 4-3), the primary oscillator is shut down and the OSTS bit is cleared. The IRCF bits may be modified at any time to immediately change the clock speed.

Note:	Caution should be used when modifying a single IRCF bit. If VDD is less than 3V, it is possible to select a higher clock speed than is supported by the low VDD. Improper device operation may result if the VDD/FOSC specifications are violated.

6.1.2.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a full or underflow will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit but not cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

6.1.3 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers, to provide a “fast return” option for interrupts. Each stack is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers, if the `RETFIE, FAST` instruction is used to return from the interrupt.

If both low and high-priority interrupts are enabled, the stack registers cannot be used reliably to return from low-priority interrupts. If a high-priority interrupt occurs while servicing a low-priority interrupt, the stack register values stored by the low-priority interrupt will be overwritten. In these cases, users must save the key registers in software during a low-priority interrupt.

If interrupt priority is not used, all interrupts may use the Fast Register Stack for returns from interrupt. If no interrupts are used, the Fast Register Stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the Fast Register Stack for a subroutine call, a `CALL label, FAST` instruction must be executed to save the STATUS, WREG and BSR registers to the Fast Register Stack. A `RETURN, FAST` instruction is then executed to restore these registers from the Fast Register Stack.

Example 6-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

EXAMPLE 6-1: FAST REGISTER STACK CODE EXAMPLE

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

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

6.1.4 LOOK-UP TABLES IN PROGRAM MEMORY

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

- Computed `GOTO`
- Table Reads

6.1.4.1 Computed `GOTO`

A computed `GOTO` is accomplished by adding an offset to the program counter. An example is shown in Example 6-2.

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

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of 2 (LSb = `0`).

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

EXAMPLE 6-2: COMPUTED `GOTO` USING AN OFFSET VALUE

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

6.1.4.2 Table Reads and Table Writes

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

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

Table read and table write operations are discussed further in **Section 7.1 “Table Reads and Table Writes”**.

6.2 PIC18 Instruction Cycle

6.2.1 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 (PC) is incremented on every Q1; the instruction is fetched from the program memory and latched into the Instruction Register (IR) during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 6-3.

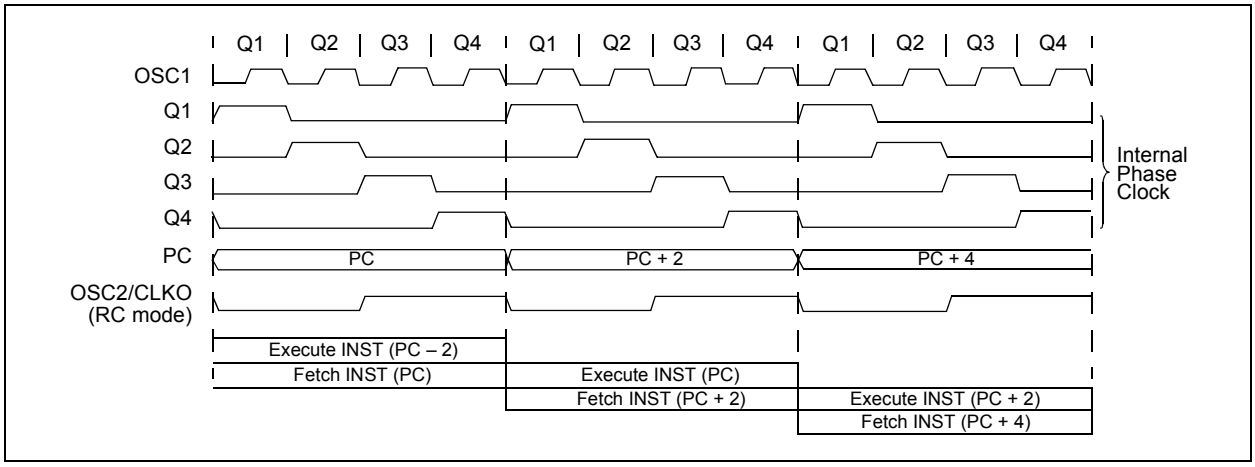
6.2.2 INSTRUCTION FLOW/PIPELINING

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 6-3).

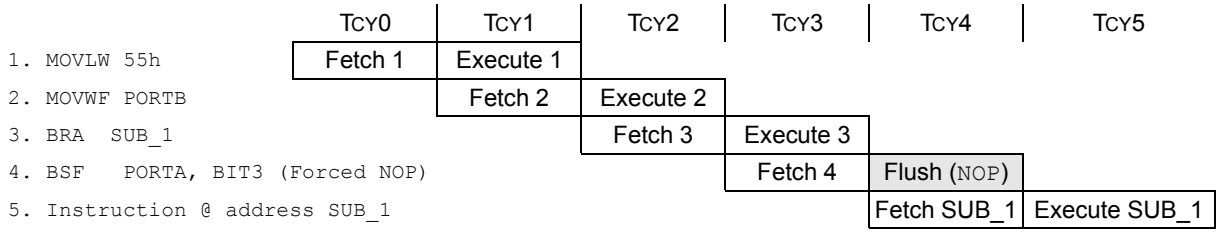
A fetch cycle begins with the program counter incrementing in Q1.

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 6-3: CLOCK/INSTRUCTION CYCLE



EXAMPLE 6-3: INSTRUCTION PIPELINE FLOW



Note: All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

TABLE 11-1: PORTA I/O SUMMARY

Pin Name	Function	I/O	TRIS	Buffer	Description
RA0/AN0/CVREF	RA0	OUT	0	DIG	LATA<0> data output.
		IN	1	TTL	PORTA<0> data input.
	AN0	IN	1	ANA	A/D Input Channel 0. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	CVREF(1)	OUT	x	ANA	Comparator voltage reference analog output. Enabling this analog output overrides the digital I/O (read as clear – low level).
RA1/AN1	RA1	OUT	0	DIG	LATA<1> data output.
		IN	1	TTL	PORTA<1> data input.
	AN1	IN	1	ANA	A/D Input Channel 1. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
RA2/AN2/VREF-	RA2	OUT	0	DIG	LATA<2> data output.
		IN	1	TTL	PORTA<2> data input.
	AN2	IN	1	ANA	A/D Input Channel 2. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	VREF-	IN	1	ANA	A/D and comparator negative voltage analog input.
RA3/AN3/VREF+	RA3	OUT	0	DIG	LATA<3> data output.
		IN	1	TTL	PORTA<3> data input.
	AN3	IN	1	ANA	A/D Input Channel 3. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	VREF+	IN	1	ANA	A/D and comparator positive voltage analog input.
RA4/T0CKI	RA4	OUT	0	DIG	LATA<4> data output.
		IN	1	TTL	PORTA<4> data input.
	T0CKI	IN	1	ST	Timer0 clock input.
RA5/AN4/$\overline{SS}$/HLVDIN	RA5	OUT	0	DIG	LATA<5> data output.
		IN	1	TTL	PORTA<5> data input.
	AN4	IN	1	ANA	A/D Input Channel 4. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	$\overline{SS}$	IN	1	TTL	Slave select input for MSSP.
	HLVDIN	IN	1	ANA	High/Low-Voltage Detect external trip point input.
OSC2/CLKO/RA6	OSC2	OUT	x	ANA	Output connection; selected by FOSC<3:0> Configuration bits. Enabling OSC2 overrides digital I/O.
	CLKO	OUT	x	DIG	Output connection; selected by FOSC<3:0> Configuration bits. Enabling CLKO overrides digital I/O (FOSC/4).
	RA6	OUT	0	DIG	LATA<6> data output.
		IN	1	TTL	PORTA<6> data input.
OSC1/CLKI/RA7	OSC1	IN	x	ANA	Main oscillator input connection determined by FOSC<3:0> Configuration bits. Enabling OSC1 overrides digital I/O.
	CLKI	IN	x	ANA	Main clock input connection determined by FOSC<3:0> Configuration bits. Enabling CLKI overrides digital I/O.
	RA7	OUT	0	DIG	LATA<7> data output.
		IN	1	TTL	PORTA<7> data input.

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input

Note 1: Available on 40/44-pin devices only.

12.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable; its value is set by the PSA and T0PS<2:0> bits (T0CON<3:0>) which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When it is assigned, prescale values from 1:2 through 1:256 in power-of-2 increments are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0`, etc.) clear the prescaler count.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count but will not change the prescaler assignment.

12.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed “on-the-fly” during program execution.

12.4 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMR0IE bit (INTCON<5>). Before re-enabling the interrupt, the TMR0IF bit must be cleared in software by the Interrupt Service Routine.

Since Timer0 is shut down in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
TMR0L	Timer0 Register Low Byte								56
TMR0H	Timer0 Register High Byte								56
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	55
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	56
TRISA	TRISA7(1)	TRISA6(1)	PORTA Data Direction Register						58

Legend: — = unimplemented locations, read as ‘0’. Shaded cells are not used by Timer0.

Note 1: RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as ‘0’.

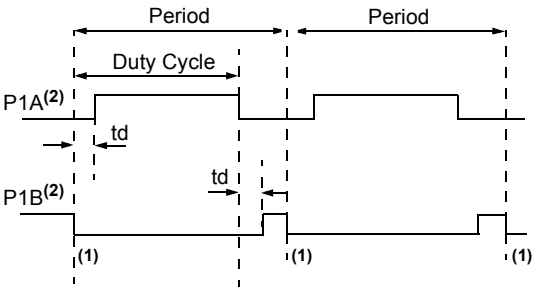
17.4.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the P1A pin, while the complementary PWM output signal is output on the P1B pin (Figure 17-4). This mode can be used for half-bridge applications, as shown in Figure 17-5, or for full-bridge applications where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits, PDC<6:0>, sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 17.4.6 "Programmable Dead-Band Delay"** for more details of the dead-band delay operations.

Since the P1A and P1B outputs are multiplexed with the PORTD<4> and PORTD<5> data latches, the TRISD<4> and TRISD<5> bits must be cleared to configure P1A and P1B as outputs.

FIGURE 17-4: HALF-BRIDGE PWM OUTPUT

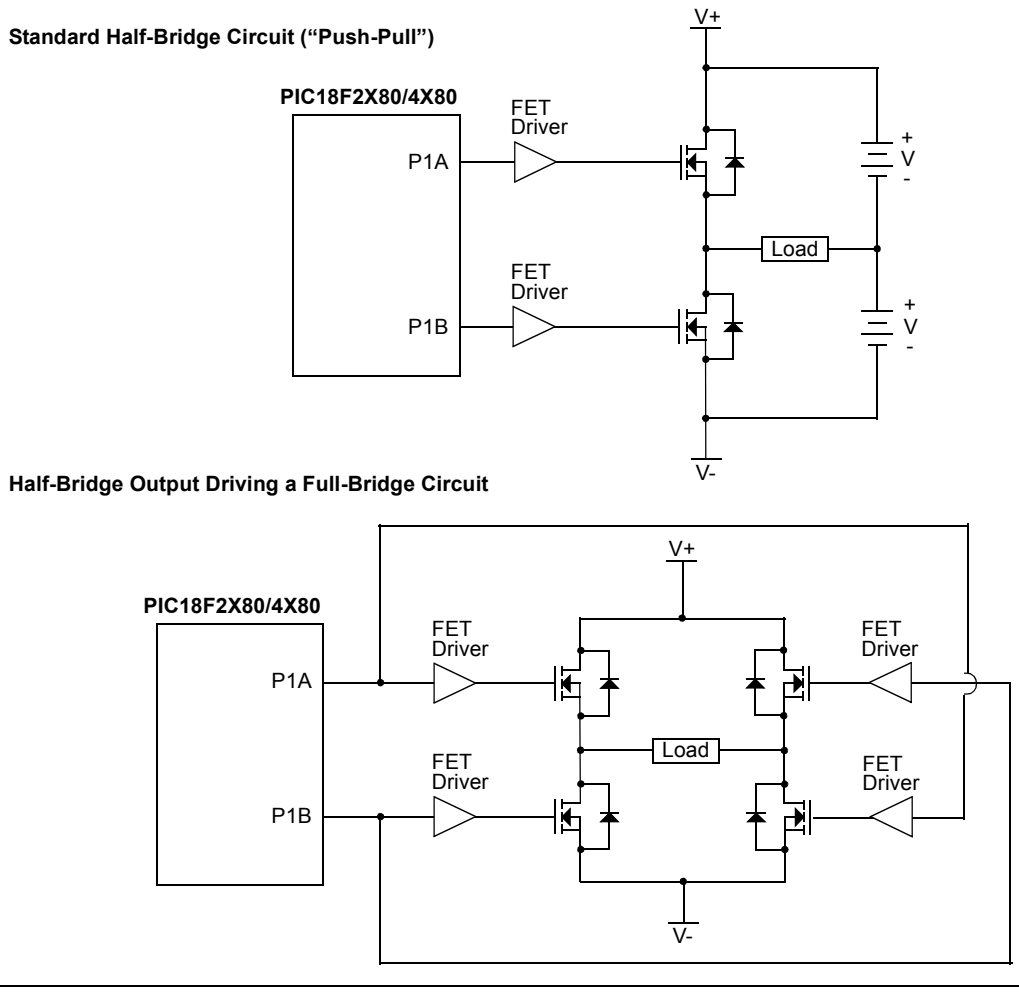


td = Dead-Band Delay

Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 17-5: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS



18.3.3 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- $\overline{SS}$ must have TRISF<7> bit set

Any serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value.

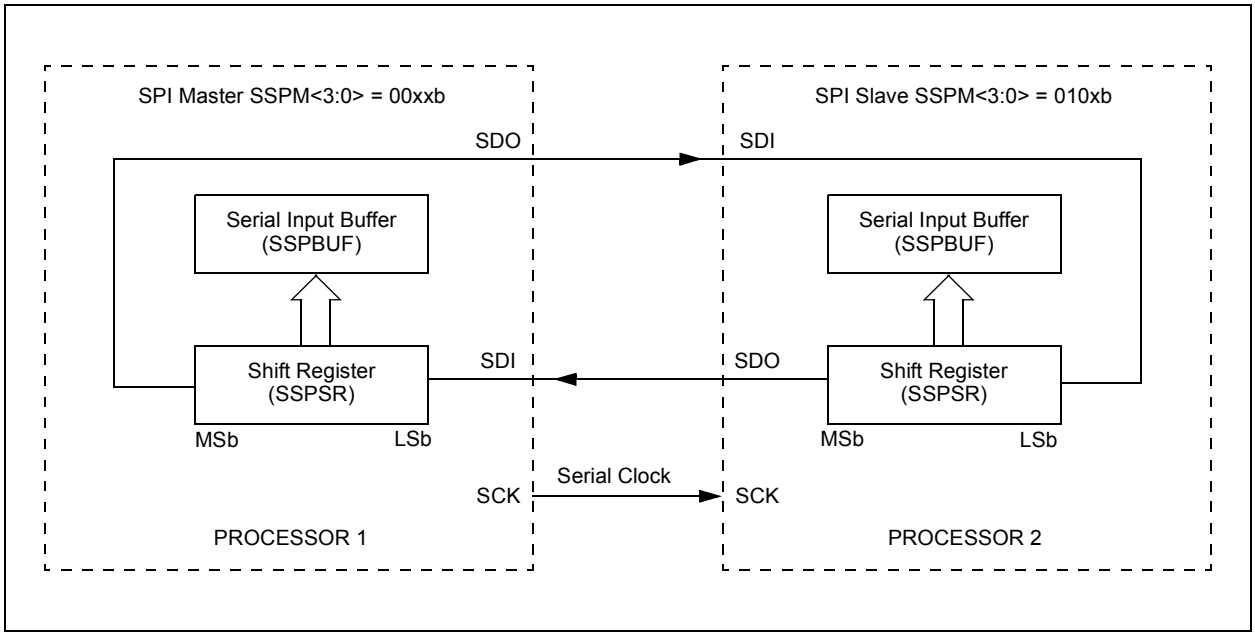
18.3.4 TYPICAL CONNECTION

Figure 18-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

Note:	When the module is enabled and in Master mode (CKE, SSPSTAT<6> = 1), a small glitch of approximately half a TCY may be seen on the SCK pin. To resolve this, keep the SCK pin as an input while setting SPEN. Then, configure the SCK pin as an output (TRISC<3> = 0).

FIGURE 18-2: SPI MASTER/SLAVE CONNECTION



REGISTER 19-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
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 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be an address/data bit or a parity bit and must be calculated by user firmware.

EXAMPLE 19-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

Desired Baud Rate = FOSC/(64 ([SPBRGH:SPBRG] + 1)

Solving for SPBRGH:SPBRG:

X = ((FOSC/Desired Baud Rate)/64) – 1

= ((16000000/9600)/64) – 1

= [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))

= 9615

Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate

= (9615 – 9600)/9600 = 0.16%

TABLE 19-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	57
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	57
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	57
SPBRGH	EUSART Baud Rate Generator Register High Byte								57
SPBRG	EUSART Baud Rate Generator Register Low Byte								57

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the BRG.

19.4.2 EUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep or any Idle mode and bit, SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREG register. If the RCIE enable bit is set, the interrupt generated will wake the chip from the low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RCIE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RCIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCIE, was set.
6. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

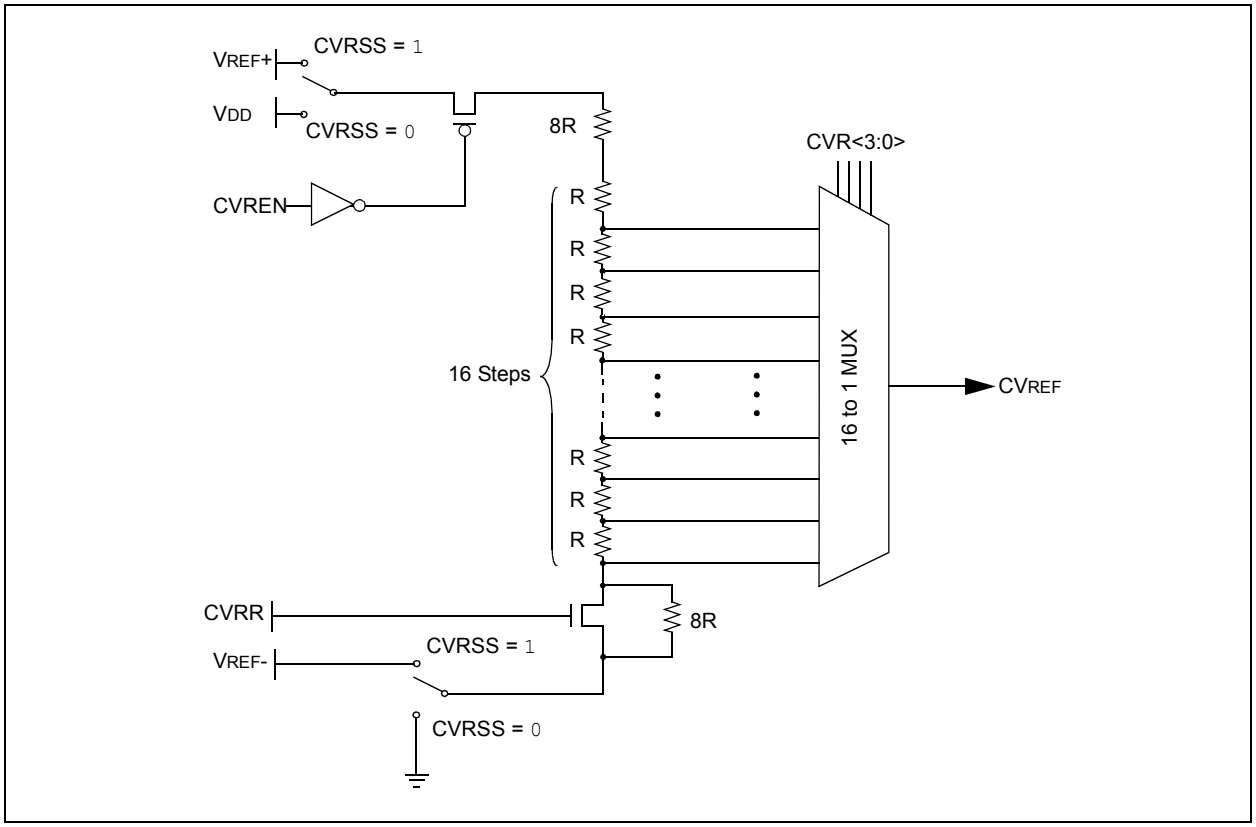
TABLE 19-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	55
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	58
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	58
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	58
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	57
RCREG	EUSART Receive Register								57
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	57
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	57
SPBRGH	EUSART Baud Rate Generator Register High Byte								57
SPBRG	EUSART Baud Rate Generator Register Low Byte								57

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

Note 1: Reserved in PIC18F2X80 devices; always maintain these bits clear.

FIGURE 22-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM



22.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 22-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in **Section 28.0 "Electrical Characteristics"**.

22.3 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the CVRCON register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

22.4 Effects of a Reset

A device Reset disables the voltage reference by clearing bit, CVREN (CVRCON<7>). This Reset also disconnects the reference from the RA0 pin by clearing bit, CVROE (CVRCON<6>), and selects the high-voltage range by clearing bit, CVRR (CVRCON<5>). The CVR value select bits are also cleared.

22.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RA0 pin if the TRISA<0> bit and the CVROE bit are both set. Enabling the voltage reference output onto the RA0 pin, with an input signal present, will increase current consumption. Connecting RA0 as a digital output with CVRSS enabled will also increase current consumption.

The RA0 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 22-2 shows an example buffering technique.

24.2.2 DEDICATED CAN TRANSMIT BUFFER REGISTERS

This section describes the dedicated CAN Transmit Buffer registers and their associated control registers.

REGISTER 24-5: TXBnCON: TRANSMIT BUFFER n CONTROL REGISTERS [0 ≤ n ≤ 2]

Mode 0	U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
	—	TXABT(1)	TXLARB(1)	TXERR(1)	TXREQ(2)	—	TXPRI1(3)	TXPRI0(3)

Mode 1,2	R/C-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
	TXBIF	TXABT(1)	TXLARB(1)	TXERR(1)	TXREQ(2)	—	TXPRI1(3)	TXPRI0(3)
	bit 7							bit 0

Legend:	C = Clearable bit		
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 Mode 0:
Unimplemented: Read as '0'
Mode 1, 2:
TXBIF: Transmit Buffer Interrupt Flag bit
1 = Transmit buffer has completed transmission of message and may be reloaded
0 = Transmit buffer has not completed transmission of a message

bit 6 **TXABT:** Transmission Aborted Status bit(1)
1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit(1)
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit(1)
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQ:** Transmit Request Status bit(2)
1 = Requests sending a message. Clears the TXABT, TXLARB and TXERR bits.
0 = Automatically cleared when the message is successfully sent

bit 2 **Unimplemented:** Read as '0'

bit 1-0 **TXPRI<1:0>:** Transmit Priority bits(3)
11 = Priority Level 3 (highest priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (lowest priority)

Note 1: This bit is automatically cleared when TXREQ is set.
2: While TXREQ is set, Transmit Buffer registers remain read-only. Clearing this bit in software while the bit is set will request a message abort.
3: These bits define the order in which transmit buffers will be transferred. They do not alter the CAN message identifier.

EXAMPLE 24-3: TRANSMITTING A CAN MESSAGE USING BANKED METHOD

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. And since we want banked method, we need to make sure
; that correct bank is selected.
BANKSEL  TXB0CON                    ; One BANKSEL in beginning will make sure that we are
                                    ; in correct bank for rest of the buffer access.
; Now load transmit data into TXB0 buffer.
MOVLW    MY_DATA_BYTE1              ; Load first data byte into buffer
MOVWF    TXB0D0                     ; Compiler will automatically set "BANKED" bit
; Load rest of data bytes - up to 8 bytes into TXB0 buffer.
...
; Load message identifier
MOVLW    60H                        ; Load SID2:SID0, EXIDE = 0
MOVWF    TXB0SIDL
MOVLW    24H                        ; Load SID10:SID3
MOVWF    TXB0SIDH
; No need to load TXB0EIDL:TXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW    B'00001000'                ; Normal priority; Request transmission
MOVWF    TXB0CON

; If required, wait for message to get transmitted
BTFSC    TXB0CON, TXREQ             ; Is it transmitted?
BRA      $-2                        ; No. Continue to wait...

; Message is transmitted.
```

EXAMPLE 24-4: TRANSMITTING A CAN MESSAGE USING WIN BITS

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. Use WIN bits to map it to RXB0 area.
MOVF    CANCON, W                   ; WIN bits are in lower 4 bits only. Read CANCON
                                    ; register to preserve all other bits. If operation
                                    ; mode is already known, there is no need to preserve
                                    ; other bits.
ANDLW   B'11110000'                 ; Clear WIN bits.
IORLW   B'00001000'                 ; Select Transmit Buffer 0
MOVWF   CANCON                      ; Apply the changes.
; Now TXB0 is mapped in place of RXB0. All future access to RXB0 registers will actually
; yield TXB0 register values.

; Load transmit data into TXB0 buffer.
MOVLW   MY_DATA_BYTE1               ; Load first data byte into buffer
MOVWF   RXB0D0                      ; Access TXB0D0 via RXB0D0 address.
; Load rest of the data bytes - up to 8 bytes into "TXB0" buffer using RXB0 registers.
...
; Load message identifier
MOVLW   60H                         ; Load SID2:SID0, EXIDE = 0
MOVWF   RXB0SIDL
MOVLW   24H                         ; Load SID10:SID3
MOVWF   RXB0SIDH
; No need to load RXB0EIDL:RXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW   B'00001000'                 ; Normal priority; Request transmission
MOVWF   RXB0CON

; If required, wait for message to get transmitted
BTFSC   RXB0CON, TXREQ              ; Is it transmitted?
BRA     $-2                         ; No. Continue to wait...

; Message is transmitted.
; If required, reset the WIN bits to default state.
```

REGISTER 24-42: RXMnSIDL: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	U-0	R/W-0	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDEN(1)	—	EID17	EID16
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 7-5 **SID<2:0>:** Standard Identifier Mask bits or Extended Identifier Mask bits (EID<20:18>)

bit 4 **Unimplemented:** Read as '0'

bit 3 Mode 0:
Unimplemented: Read as '0'
Mode 1, 2:
EXIDEN: Extended Identifier Filter Enable Mask bit(1)
1 = Messages selected by the EXIDEN bit in RXFnSIDL will be accepted
0 = Both standard and extended identifier messages will be accepted

bit 2 **Unimplemented:** Read as '0'

bit 1-0 **EID<17:16>:** Extended Identifier Mask bits

Note 1: This bit is available in Mode 1 and 2 only.

REGISTER 24-43: RXMnEIDH: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

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

bit 7-0 **EID<15:8>:** Extended Identifier Mask bits

REGISTER 24-44: RXMnEIDL: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

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

bit 7-0 **EID<7:0>:** Extended Identifier Mask bits

REGISTER 24-48: MSEL0: MASK SELECT REGISTER 0(1)

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_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 **FIL3_<1:0>:** Filter 3 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL2_<1:0>:** Filter 2 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL1_<1:0>:** Filter 1 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL0_<1:0>:** Filter 0 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

24.2.6 CAN INTERRUPT REGISTERS

The registers in this section are the same as described in **Section 10.0 "Interrupts"**. They are duplicated here for convenience.

REGISTER 24-56: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3

Mode 0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF(1)	TXB0IF(1)	RXB1IF	RXB0IF

Mode 1,2	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IRXIF	WAKIF	ERRIF	TXBnIF	TXB1IF(1)	TXB0IF(1)	RXBnIF	FIFOWMIF
	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 **IRXIF:** CAN Bus Error Message Received Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus

bit 6 **WAKIF:** CAN Bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus

bit 5 **ERRIF:** CAN Module Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources; refer to **Section 24.15.6 "Error Interrupt"**)
0 = No CAN module errors

bit 4 When CAN is in Mode 0:
TXB2IF: CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 2 has not completed transmission of a message
When CAN is in Mode 1 or 2:
TXBnIF: Any Transmit Buffer Interrupt Flag bit
1 = One or more transmit buffers have completed transmission of a message and may be reloaded
0 = No transmit buffer is ready for reload

bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit(1)
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 1 has not completed transmission of a message

bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit(1)
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 0 has not completed transmission of a message

bit 1 When CAN is in Mode 0:
RXB1IF: CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message
When CAN is in Mode 1 or 2:
RXBnIF: Any Receive Buffer Interrupt Flag bit
1 = One or more receive buffers has received a new message
0 = No receive buffer has received a new message

bit 0 When CAN is in Mode 0:
RXB0IF: CAN Receive Buffer 0 Interrupt Flag bit
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message
When CAN is in Mode 1:
Unimplemented: Read as '0'
When CAN is in Mode 2:
FIFOWMIF: FIFO Watermark Interrupt Flag bit
1 = FIFO high watermark is reached
0 = FIFO high watermark is not reached

Note 1: In CAN Mode 1 and 2, these bits are forced to '0'.

25.0 SPECIAL FEATURES OF THE CPU

PIC18F2480/2580/4480/4580 devices include several features intended to maximize reliability and minimize cost through elimination of external components. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- ID Locations
- In-Circuit Serial Programming

The oscillator can be configured for the application depending on frequency, power, accuracy and cost. All of the options are discussed in detail in **Section 3.0 "Oscillator Configurations"**.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet.

In addition to their Power-up and Oscillator Start-up Timers provided for Resets, PIC18F2480/2580/4480/4580 devices have a Watchdog Timer, which is either permanently enabled via the Configuration bits or software controlled (if configured as disabled).

The inclusion of an internal RC oscillator also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up, while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

25.1 Configuration Bits

The Configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction with the TBLPTR pointing to the Configuration register sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a '1' or a '0' into the cell. For additional details on Flash programming, refer to **Section 7.5 "Writing to Flash Program Memory"**.

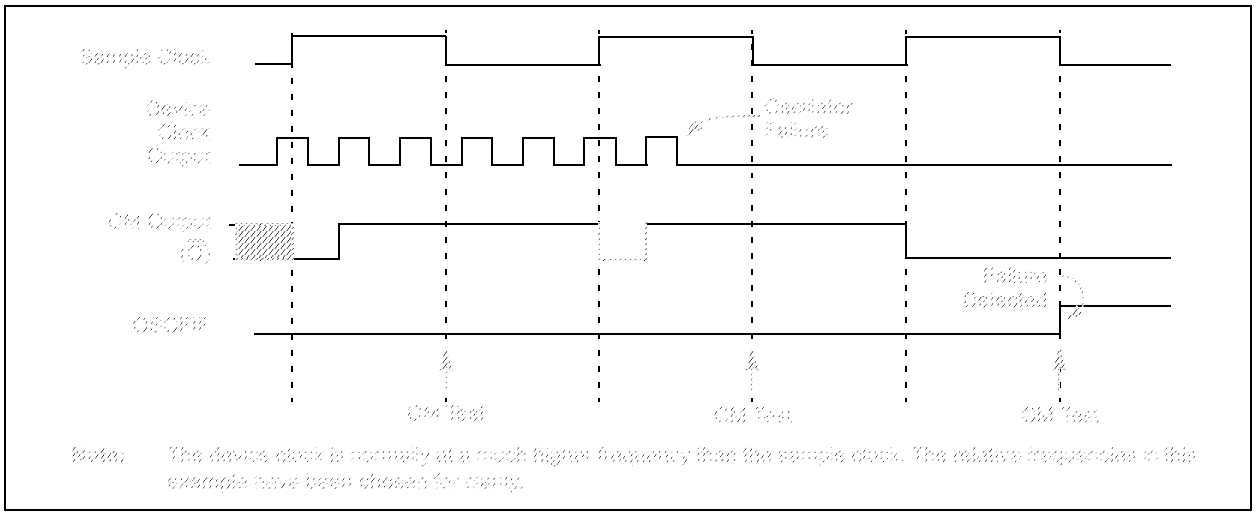
TABLE 25-1: CONFIGURATION BITS AND DEVICE IDs

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300001h	CONFIG1H	IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0	00-- 0111
300002h	CONFIG2L	—	—	—	BORV1	BORV0	BOREN1	BOREN0	$\overline{\text{PWRTEN}}$	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300005h	CONFIG3H	MCLRE	—	—	—	—	LPT1OSC	PBADEN	—	1--- -01-
300006h	CONFIG4L	$\overline{\text{DEBUG}}$	XINST	—	BBSIZ	—	LVP	—	STVREN	10-0 -1-1
300008h	CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0	---- 1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0	---- 1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0	---- 1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	xxxx xxxx(1)
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 1100

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition.
Shaded cells are unimplemented, read as '0'.

Note 1: See Register 25-12 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

FIGURE 25-4: FSCM TIMING DIAGRAM



25.4.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. Fail-Safe Clock Monitoring of the power-managed clock source resumes in the power-managed mode.

If an oscillator failure occurs during power-managed operation, the subsequent events depend on whether or not the oscillator failure interrupt is enabled. If enabled (OSCFIF = 1), code execution will be clocked by the INTOSC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTOSC source.

25.4.4 POR OR WAKE-UP FROM SLEEP

The FSCM is designed to detect oscillator failure at any point after the device has exited Power-on Reset (POR) or low-power Sleep mode. When the primary device clock is EC, RC or INTRC modes, monitoring can begin immediately following these events.

For oscillator modes involving a crystal or resonator (HS, HSPLL, LP or XT), the situation is somewhat different. Since the oscillator may require a start-up time considerably longer than the FCSM sample clock time, a false clock failure may be detected. To prevent this, the internal oscillator block is automatically configured as the device clock and functions until the primary clock is stable (the OST and PLL timers have timed out). This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTRC returns to its role as the FSCM source.

Note:	The same logic that prevents false oscillator failure interrupts on POR, or wake from Sleep, will also prevent the detection of the oscillator's failure to start at all following these events. This can be avoided by monitoring the OSTS bit and using a timing routine to determine if the oscillator is taking too long to start. Even so, no oscillator failure interrupt will be flagged.

As noted in **Section 25.3.1 "Special Considerations for Using Two-Speed Start-up"**, it is also possible to select another clock configuration and enter an alternate power-managed mode while waiting for the primary clock to become stable. When the new power-managed mode is selected, the primary clock is disabled.

TABLE 28-1: MEMORY PROGRAMMING REQUIREMENTS

DC Characteristics			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Internal Program Memory Programming Specifications(1)					
D110	VPP	Voltage on $\overline{\text{MCLR}}$/VPP/RE3 Pin	9.00	—	13.25	V	**(Note 3)**
D113	IDDP	Supply Current during Programming	—	—	10	mA	
		Data EEPROM Memory					
D120	ED	Byte Endurance	100K	1M	—	E/W	-40°C to +85°C
D121	VDRW	VDD for Read/Write	VMIN	—	5.5	V	Using EECON to read/write VMIN = Minimum operating voltage
D122	TDEW	Erase/Write Cycle Time	—	4	—	ms	
D123	TRETD	Characteristic Retention	40	—	—	Year	Provided no other specifications are violated
D124	TREF	Number of Total Erase/Write Cycles before Refresh(2)	1M	10M	—	E/W	-40°C to +85°C
		Program Flash Memory					
D130	EP	Cell Endurance	10K	100K	—	E/W	-40°C to +85°C
D131	VPR	VDD for Read	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D132	VIE	VDD for Block Erase	4.5	—	5.5	V	Using ICSP™ port
D132A	VIW	VDD for Externally Timed Erase or Write	4.5	—	5.5	V	Using ICSP port
D132B	VPEW	VDD for Self-Timed Write	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D133	TIE	ICSP Block Erase Cycle Time	—	4	—	ms	VDD > 4.5V
D133A	TIW	ICSP Erase or Write Cycle Time (externally timed)	1	—	—	ms	VDD > 4.5V
D133A	TIW	Self-Timed Write Cycle Time	—	2	—	ms	
D134	TRETD	Characteristic Retention	40	100	—	Year	Provided no other specifications are violated

† Data in “Typ” column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: These specifications are for programming the on-chip program memory through the use of table write instructions.

2: Refer to **Section 8.8 “Using the Data EEPROM”** for a more detailed discussion on data EEPROM endurance.

3: Required only if Single-Supply Programming is disabled.

FIGURE 28-3: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

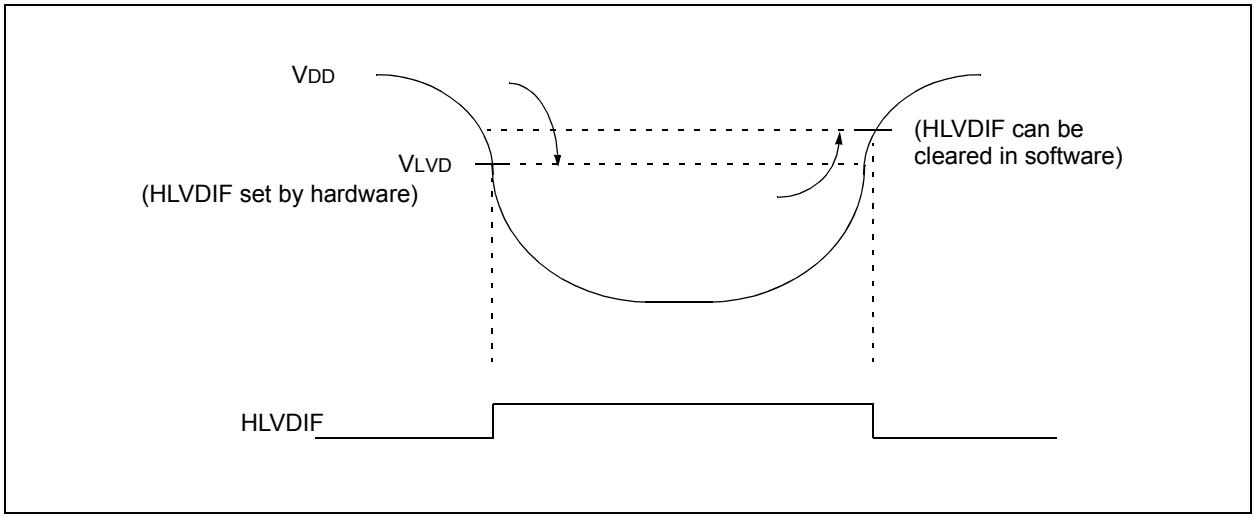


TABLE 28-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

				Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C < TA < +125°C for extended				
Param No.	**Symbol**	**Characteristic**		**Min**	**Typ**	**Max**	**Units**	**Conditions**
D420		HLVD Voltage on VDD Transition High-to-Low	LVV = 0000	2.12	2.17	2.22	V	
			LVV = 0001	2.18	2.23	2.28	V	
			LVV = 0010	2.31	2.36	2.42	V	
			LVV = 0011	2.38	2.44	2.49	V	
			LVV = 0100	2.54	2.60	2.66	V	
			LVV = 0101	2.72	2.79	2.85	V	
			LVV = 0110	2.82	2.89	2.95	V	
			LVV = 0111	3.05	3.12	3.19	V	
			LVV = 1000	3.31	3.39	3.47	V	
			LVV = 1001	3.46	3.55	3.63	V	
			LVV = 1010	3.63	3.71	3.80	V	
			LVV = 1011	3.81	3.90	3.99	V	
			LVV = 1100	4.01	4.11	4.20	V	
			LVV = 1101	4.23	4.33	4.43	V	
			LVV = 1110	4.48	4.59	4.69	V	
			LVV = 1111	1.14	1.2	1.26	V	