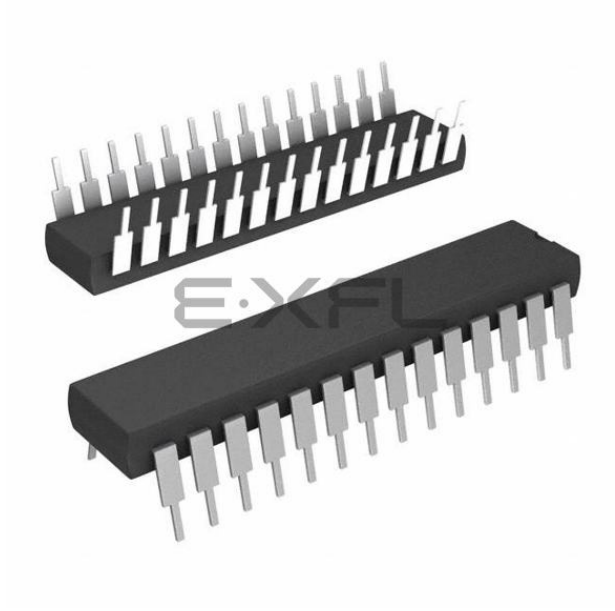


Welcome to E-XFL.COM

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	25MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	48KB (24K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.25K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f2585-e-sp

TABLE 4-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices				Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
RXF12SIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDL(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11SIDL(6)	2585	2680	4585	4680	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF11SIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF10EIDL(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF10EIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF10SIDL(6)	2585	2680	4585	4680	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF10SIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF9EIDL(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF9EIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF9SIDL(6)	2585	2680	4585	4680	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF9SIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF8EIDL(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF8EIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF8SIDL(6)	2585	2680	4585	4680	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF8SIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF7EIDL(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF7EIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF7SIDL(6)	2585	2680	4585	4680	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF7SIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF6EIDL(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF6EIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF6SIDL(6)	2585	2680	4585	4680	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF6SIDH(6)	2585	2680	4585	4680	xxxx xxxx	uuuu uuuu	-uuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 4-3 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

6: This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

5.1.2.4 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.

5.1.3 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 5-1 shows a source code example that uses the fast register stack during a subroutine call and return.

EXAMPLE 5-1: 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
```

5.1.4 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

5.1.4.1 Computed GOTO

A computed `GOTO` is accomplished by adding an offset to the program counter. An example is shown in Example 5-2.

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 5-2: COMPUTED GOTO USING AN OFFSET VALUE

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

5.1.4.2 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 6.1 “Table Reads and Table Writes”**.

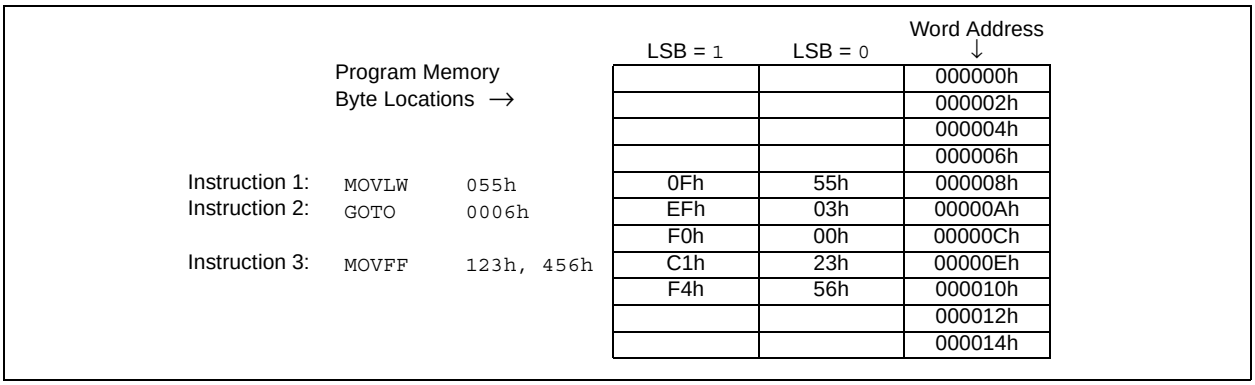
5.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = 0). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see **Section 5.1.1 "Program Counter"**).

Figure 5-4 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 5-4 shows how the instruction GOTO 0006h is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 25.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 5-4: INSTRUCTIONS IN PROGRAM MEMORY



5.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: CALL, MOVFF, GOTO and LSFR. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSbs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 5-4 shows how this works.

Note:	See **Section 5.5 "Program Memory and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

EXAMPLE 5-4: TWO-WORD INSTRUCTIONS

CASE 1:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, skip this word
1111 0100 0101 0110			; Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code
CASE 2:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes, execute this word
1111 0100 0101 0110			; 2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3	; continue code

12.1 Timer1 Operation

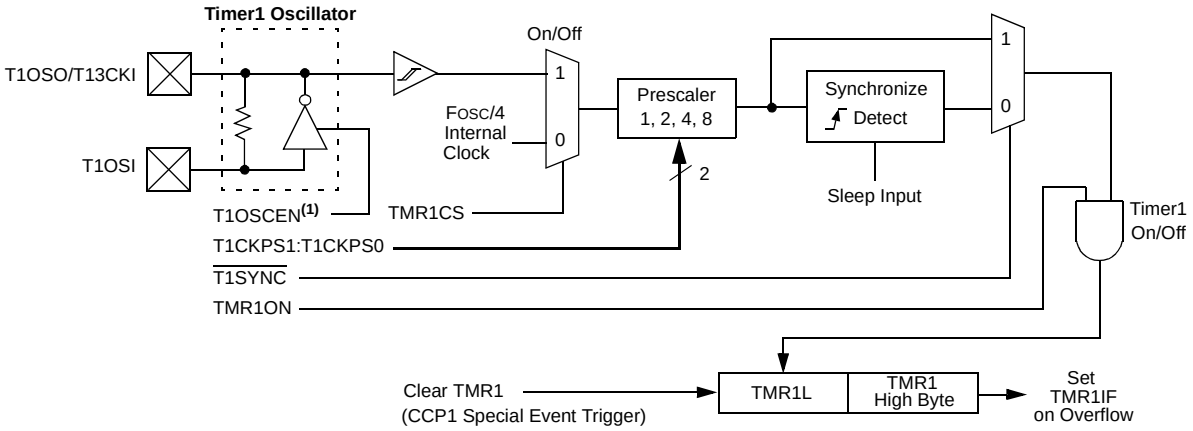
Timer1 can operate in one of these modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>). When TMR3CS is cleared (= 0), Timer3 increments on every internal instruction cycle (Fosc/4). When the bit is set, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

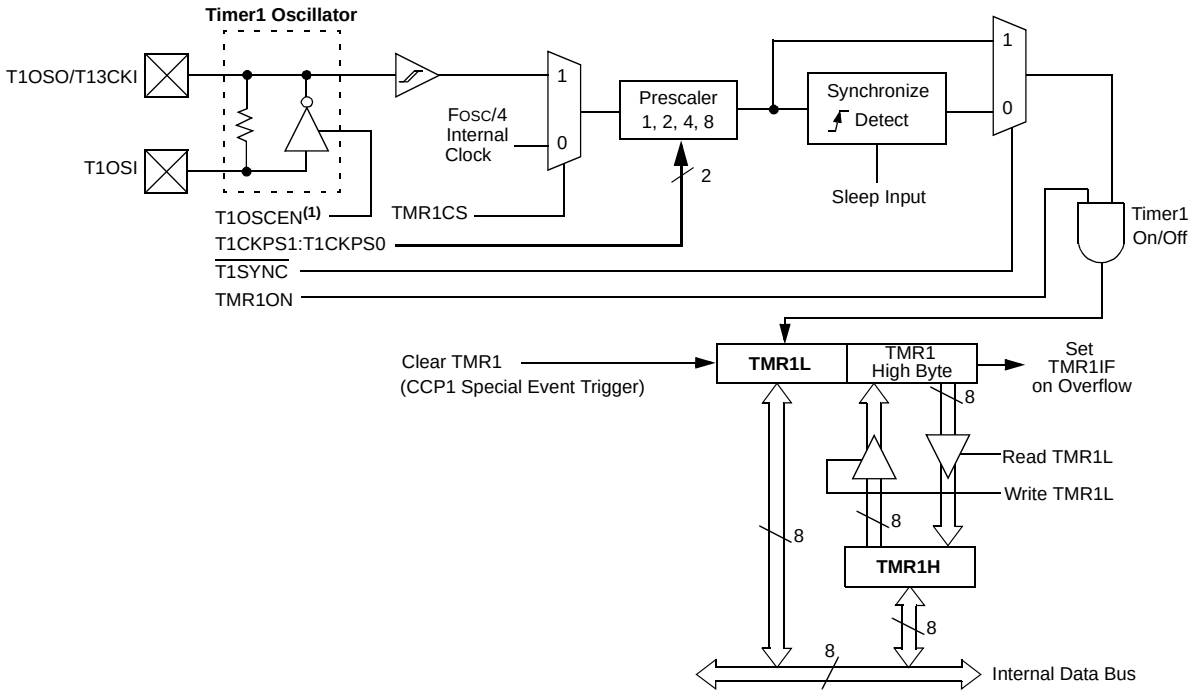
When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

FIGURE 12-1: TIMER1 BLOCK DIAGRAM



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

FIGURE 12-2: TIMER1 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

17.4.4 CLOCK STRETCHING

Both 7 and 10-bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

17.4.4.1 Clock Stretching for 7-bit Slave Receive Mode (SEN = 1)

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the ACK sequence if the BF bit is set, the CKP bit in the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 17-13).

Note 1: If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

17.4.4.2 Clock Stretching for 10-bit Slave Receive Mode (SEN = 1)

In 10-bit Slave Receive mode during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{W}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

Note: If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

17.4.4.3 Clock Stretching for 7-bit Slave Transmit Mode

7-bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 17-9).

Note 1: If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit.

17.4.4.4 Clock Stretching for 10-bit Slave Transmit Mode

In 10-bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high-order bits of the 10-bit address and the R/$\overline{W}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-bit Slave Transmit mode (see Figure 17-11).

18.4 EUSART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit, CSRC (TXSTA<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

18.4.1 EUSART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical, except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in the TXREG register.
c) Flag bit TXIF will not be set.
d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit TXIF will now be set.
e) If enable bit TXIE is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 18-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

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	49
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
TXREG	EUSART Transmit Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

Note 1: Reserved in PIC18F2X8X devices; always maintain these bits clear.

19.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) converter module has 8 inputs for the PIC18F2X8X devices and 11 for the PIC18F4X8X devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 19-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 19-2, configures the functions of the port pins. The ADCON2 register, shown in Register 19-3, configures the A/D clock source, programmed acquisition time and justification.

REGISTER 19-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits

0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)**(1,2)**
0110 = Channel 6 (AN6)**(1,2)**
0111 = Channel 7 (AN7)**(1,2)**
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Unused
1100 = Unused
1101 = Unused
1110 = Unused
1111 = Unused

Note 1: These channels are not implemented on PIC18F2X8X devices.

2: Performing a conversion on unimplemented channels will return full-scale measurements.

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit

1 = A/D converter module is enabled
0 = A/D converter module is disabled

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

23.0 ECAN™ TECHNOLOGY

PIC18F2585/2680/4585/4680 devices contain an Enhanced Controller Area Network (ECAN) module. The ECAN module is fully backward compatible with the CAN module available in PIC18CXX8 and PIC18FXX8 devices.

The Controller Area Network (CAN) module is a serial interface which is useful for communicating with other peripherals or microcontroller devices. This interface, or protocol, was designed to allow communications within noisy environments.

The ECAN module is a communication controller, implementing the CAN 2.0A or B protocol as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN 2.0B Passive and CAN 2.0B Active versions of the protocol. The module implementation is a full CAN system; however, the CAN specification is not covered within this data sheet. Refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol CAN 1.2, CAN 2.0A and CAN 2.0B
- DeviceNet™ data bytes filter support
- Standard and extended data frames
- 0-8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Fully backward compatible with PIC18XXX8 CAN module
- Three modes of operation:
  - Mode 0 – Legacy mode
  - Mode 1 – Enhanced Legacy mode with DeviceNet support
  - Mode 2 – FIFO mode with DeviceNet support
- Support for remote frames with automated handling
- Double-buffered receiver with two prioritized received message storage buffers
- Six buffers programmable as RX and TX message buffers
- 16 full (standard/extended identifier) acceptance filters that can be linked to one of four masks
- Two full acceptance filter masks that can be assigned to any filter
- One full acceptance filter that can be used as either an acceptance filter or acceptance filter mask
- Three dedicated transmit buffers with application specified prioritization and abort capability
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low-power Sleep mode

23.1 Module Overview

The CAN bus module consists of a protocol engine and message buffering and control. The CAN protocol engine automatically handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the two receive registers.

The CAN module supports the following frame types:

- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception
- Interframe Space Generation/Detection

The CAN module uses the RB2/CANTX and RB3/CANRX pins to interface with the CAN bus. In normal mode, the CAN module automatically overrides TRISB<2>. The user must ensure that TRISB<3> is set.

23.1.1 MODULE FUNCTIONALITY

The CAN bus module consists of a protocol engine, message buffering and control (see Figure 23-1). The protocol engine can best be understood by defining the types of data frames to be transmitted and received by the module.

The following sequence illustrates the necessary initialization steps before the ECAN module can be used to transmit or receive a message. Steps can be added or removed depending on the requirements of the application.

1. Ensure that the ECAN module is in Configuration mode.
2. Select ECAN Operational mode.
3. Set up the baud rate registers.
4. Set up the filter and mask registers.
5. Set the ECAN module to normal mode or any other mode required by the application logic.

REGISTER 23-23: BnCON: TX/RX BUFFER n CONTROL REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1](1)

R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
TXBIF	TXABT	TXLARB	TXERR	TXREQ	RTREN	TXPRI1	TXPRI0
bit 7							bit 0

bit 7 **TXBIF:** Transmit Buffer Interrupt Flag bit(3)
1 = A message is successfully transmitted
0 = No message was transmitted

bit 6 **TXABT:** Transmission Aborted Status bit(3)
1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit(3)
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit(3)
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(2,4)
1 = Requests sending a message; clears the TXABT, TXLARB and TXERR bits
0 = Automatically cleared when the message is successfully sent

bit 2 **RTREN:** Automatic Remote Transmission Request Enable bit
1 = When a remote transmission request is received, TXREQ will be automatically set
0 = When a remote transmission request is received, TXREQ will be unaffected

bit 1-0 **TXPRI1:TXPRI0:** Transmit Priority bits(5)
11 = Priority Level 3 (highest priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (lowest priority)

Note 1: These registers are available in Mode 1 and 2 only.
2: Clearing this bit in software while the bit is set will request a message abort.
3: This bit is automatically cleared when TXREQ is set.
4: While TXREQ is set or transmission is in progress, transmit buffer registers remain read-only.
5: These bits set the order in which the transmit buffer will be transferred. They do not alter the CAN message identifier.

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

REGISTER 23-59: TXBIE: TRANSMIT BUFFERS INTERRUPT ENABLE REGISTER(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0	U-0
—	—	—	TXB2IE(2)	TXB1IE(2)	TXB0IE(2)	—	—
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-2 **TXB2IE:TXB0IE:** Transmit Buffer 2-0 Interrupt Enable bit(2)

1 = Transmit buffer interrupt is enabled
0 = Transmit buffer interrupt is disabled

bit 1-0 **Unimplemented:** Read as '0'

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

2: TXBnIE in PIE3 register must be set to get an interrupt.

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

REGISTER 23-60: BIE0: BUFFER INTERRUPT ENABLE REGISTER 0(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
B5IE(2)	B4IE(2)	B3IE(2)	B2IE(2)	B1IE(2)	B0IE(2)	RXB1IE(2)	RXB0IE(2)
bit 7							bit 0

bit 7-2 **B5IE:B0IE:** Programmable Transmit/Receive Buffer 5-0 Interrupt Enable bit(2)

1 = Interrupt is enabled
0 = Interrupt is disabled

bit 1-0 **RXB1IE:RXB0IE:** Dedicated Receive Buffer 1-0 Interrupt Enable bit(2)

1 = Interrupt is enabled
0 = Interrupt is disabled

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

2: Either TXBnIE or RXBnIE in PIE3 register must be set to get an interrupt.

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

TABLE 23-1: CAN CONTROLLER REGISTER MAP (CONTINUED)

Address[1]	Name	Address	Name	Address	Name	Address	Name
EFFh	—[4]	EDFh	—[4]	EBFh	—[4]	E9Fh	—[4]
EFEh	—[4]	EDEh	—[4]	EBEh	—[4]	E9Eh	—[4]
EFDh	—[4]	EDDh	—[4]	EBDh	—[4]	E9Dh	—[4]
EFCh	—[4]	EDCh	—[4]	EBCh	—[4]	E9Ch	—[4]
EFBh	—[4]	EDBh	—[4]	EBBh	—[4]	E9Bh	—[4]
EFAh	—[4]	EDAh	—[4]	EBAh	—[4]	E9Ah	—[4]
EF9h	—[4]	ED9h	—[4]	EB9h	—[4]	E99h	—[4]
EF8h	—[4]	ED8h	—[4]	EB8h	—[4]	E98h	—[4]
EF7h	—[4]	ED7h	—[4]	EB7h	—[4]	E97h	—[4]
EF6h	—[4]	ED6h	—[4]	EB6h	—[4]	E96h	—[4]
EF5h	—[4]	ED5h	—[4]	EB5h	—[4]	E95h	—[4]
EF4h	—[4]	ED4h	—[4]	EB4h	—[4]	E94h	—[4]
EF3h	—[4]	ED3h	—[4]	EB3h	—[4]	E93h	—[4]
EF2h	—[4]	ED2h	—[4]	EB2h	—[4]	E92h	—[4]
EF1h	—[4]	ED1h	—[4]	EB1h	—[4]	E91h	—[4]
EF0h	—[4]	ED0h	—[4]	EB0h	—[4]	E90h	—[4]
EEFh	—[4]	ECFh	—[4]	EAFh	—[4]	E8Fh	—[4]
EEEh	—[4]	ECEh	—[4]	EAEh	—[4]	E8Eh	—[4]
EEDh	—[4]	ECDh	—[4]	EADh	—[4]	E8Dh	—[4]
EECh	—[4]	ECCh	—[4]	EACh	—[4]	E8Ch	—[4]
EEBh	—[4]	ECBh	—[4]	EABh	—[4]	E8Bh	—[4]
EEAh	—[4]	ECAh	—[4]	EAAh	—[4]	E8Ah	—[4]
EE9h	—[4]	EC9h	—[4]	EA9h	—[4]	E89h	—[4]
EE8h	—[4]	EC8h	—[4]	EA8h	—[4]	E88h	—[4]
EE7h	—[4]	EC7h	—[4]	EA7h	—[4]	E87h	—[4]
EE6h	—[4]	EC6h	—[4]	EA6h	—[4]	E86h	—[4]
EE5h	—[4]	EC5h	—[4]	EA5h	—[4]	E85h	—[4]
EE4h	—[4]	EC4h	—[4]	EA4h	—[4]	E84h	—[4]
EE3h	—[4]	EC3h	—[4]	EA3h	—[4]	E83h	—[4]
EE2h	—[4]	EC2h	—[4]	EA2h	—[4]	E82h	—[4]
EE1h	—[4]	EC1h	—[4]	EA1h	—[4]	E81h	—[4]
EE0h	—[4]	EC0h	—[4]	EA0h	—[4]	E80h	—[4]

Note 1: Shaded registers are available in Access Bank low area, while the rest are available in Bank 15.

2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

3: These registers are not CAN registers.

4: Unimplemented registers are read as '0'.

REGISTER 24-8: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	WRT3(1)	WRT2	WRT1	WRT0
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **WRT3:** Write Protection bit(1)
1 = Block 3 (00C000-00FFFFh) not write-protected
0 = Block 3 (00C000-00FFFFh) write-protected

Note 1: Unimplemented in PIC18FX585 devices; maintain this bit set.

bit 2 **WRT2:** Write Protection bit
1 = Block 2 (008000-00BFFFh) not write-protected
0 = Block 2 (008000-00BFFFh) write-protected

bit 1 **WRT1:** Write Protection bit
1 = Block 1 (004000-007FFFh) not write-protected
0 = Block 1 (004000-007FFFh) write-protected

bit 0 **WRT0:** Write Protection bit
1 = Block 0 (000800-003FFFh) not write-protected
0 = Block 0 (000800-003FFFh) write-protected

Legend:		
R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-9: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)

R/C-1	R/C-1	R-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC(1)	—	—	—	—	—
bit 7							bit 0

bit 7 **WRTD:** Data EEPROM Write Protection bit
1 = Data EEPROM not write-protected
0 = Data EEPROM write-protected

bit 6 **WRTB:** Boot Block Write Protection bit
1 = Boot block (000000-0007FFh) not write-protected
0 = Boot block (000000-0007FFh) write-protected

bit 5 **WRTC:** Configuration Register Write Protection bit(1)
1 = Configuration registers (300000-3000FFh) not write-protected
0 = Configuration registers (300000-3000FFh) write-protected

Note 1: This bit is read-only in normal execution mode; it can be written only in Program mode.

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

Legend:		
R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed		u = Unchanged from programmed state

24.5 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PIC® devices.

The user program memory is divided into five blocks. One of these is a boot block of 2 Kbytes. The remainder of the memory is divided into four blocks on binary boundaries.

Each of the five blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 24-5 shows the program memory organization for 48 and 64-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 24-3.

FIGURE 24-5: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F2585/2680/4585/4680

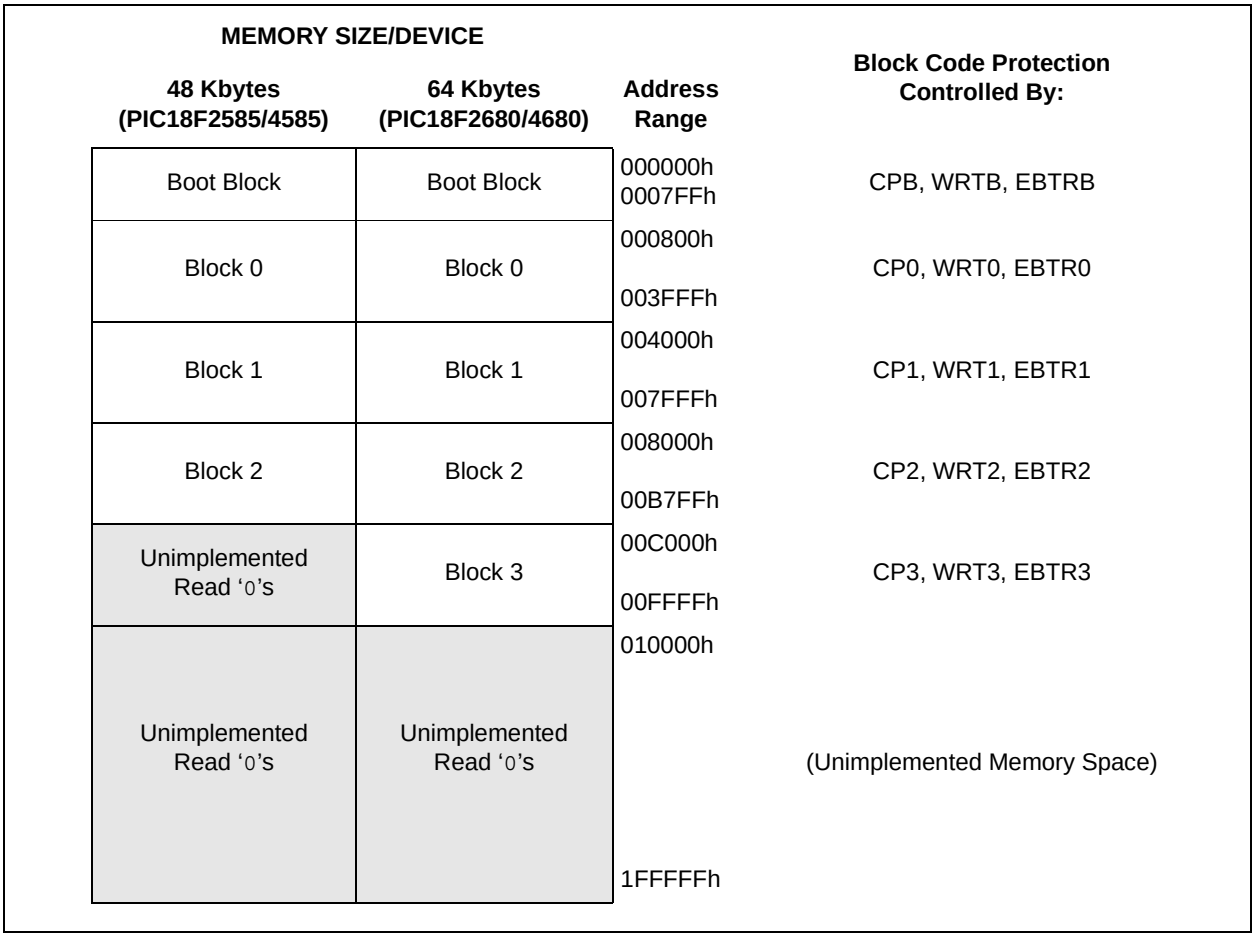


TABLE 24-3: SUMMARY OF CODE PROTECTION REGISTERS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	—	CP3*	CP2	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	—	WRT3*	WRT2	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch	CONFIG7L	—	—	—	—	EBTR3*	EBTR2	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—

Legend: Shaded cells are unimplemented.

* Unimplemented in PIC18FX585 devices; maintain this bit set.

24.5.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits internal and external writes to data EEPROM. The CPU can continue to read and write data EEPROM regardless of the protection bit settings.

24.5.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In normal execution mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

24.6 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are both readable and writable during normal execution through the TBLRD and TBLWT instructions or during program/verify. The ID locations can be read when the device is code-protected.

24.7 In-Circuit Serial Programming

PIC18F2585/2680/4585/4680 microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

24.8 In-Circuit Debugger

When the $\overline{\text{DEBUG}}$ Configuration bit is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use. Table 24-4 shows which resources are required by the background debugger.

TABLE 24-4: DEBUGGER RESOURCES

I/O pins:	RB6, RB7
Stack:	2 levels

Note: Memory resources listed in MPLAB® IDE.

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to $\overline{\text{MCLR}}$/VPP/RE3, VDD, VSS, RB7 and RB6. This will interface to the In-Circuit debugger module available from Microchip or one of the third party development tool companies.

24.9 Single-Supply ICSP Programming

The LVP Configuration bit enables Single-Supply ICSP programming (formerly known as *Low-Voltage ICSP Programming* or *LVP*). When Single-Supply Programming is enabled, the microcontroller can be programmed without requiring high voltage being applied to the $\overline{\text{MCLR}}$/VPP/RE3 pin, but the RB5/KBI1/PGM pin is then dedicated to controlling Program mode entry and is not available as a general purpose I/O pin.

While programming using Single-Supply Programming, VDD is applied to the $\overline{\text{MCLR}}$/VPP/RE3 pin as in normal execution mode. To enter Programming mode, VDD is applied to the PGM pin.

Note 1: High-voltage programming is always available, regardless of the state of the LVP bit, by applying VIHH to the $\overline{\text{MCLR}}$ pin.

2: While in Low-Voltage ICSP Programming mode, the RB5 pin can no longer be used as a general purpose I/O pin and should be held low during normal operation.

3: When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 5 in the TRISB register must be cleared to disable the pull-up on RB5 and ensure the proper operation of the device.

4: If the device Master Clear is disabled, verify that either of the following is done to ensure proper entry into ICSP mode:

a) disable Low-Voltage Programming (CONFIG4l<2> = 0); or

b) make certain that RB5/PGM is held low during entry into ICSP.

If Single-Supply ICSP Programming mode will not be used, the LVP bit can be cleared. RB5/KBI1/PGM then becomes available as the digital I/O pin, RB5. The LVP bit may be set or cleared only when using standard high-voltage programming (VIHH applied to the $\overline{\text{MCLR}}$/VPP/RE3 pin). Once LVP has been disabled, only the standard high-voltage programming is available and must be used to program the device.

Memory that is not code-protected can be erased using either a block erase, or erased row by row, then written at any specified VDD. If code-protected memory is to be erased, a block erase is required. If a block erase is to be performed when using Low-Voltage Programming, the device must be supplied with VDD of 4.5V to 5.5V.

27.2 DC Characteristics: Power-Down and Supply Current PIC18F2585/2680/4585/4680 (Industrial) PIC18LF2585/2680/4585/4680 (Industrial) (Continued)

PIC18LF2585/2680/4585/4680 (Industrial)	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial
PIC18F2585/2680/4585/4680 (Industrial, Extended)	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended

Param No.	Device	Typ	Max	Units	Conditions		
	Supply Current (IDD)(2,3)						
	PIC18FX585/X680	15.00	23.00	mA	+125°C	VDD = 4.2V	FOSC = 25 MHz (**PRI_RUN**, EC oscillator)
		20.00	31.00	mA	+125°C	VDD = 5.0V	
	All devices	30.00	38.00	mA	-40°C	VDD = 4.2V	FOSC = 40 MHz (**PRI_RUN**, EC oscillator)
		31.00	38.00	mA	+25°C		
		31.00	38.00	mA	+85°C		
	All devices	37.00	44.00	mA	-40°C	VDD = 5.0V	
		38.00	44.00	mA	+25°C		
		39.00	44.00	mA	+85°C		
	All devices	7.50	16.00	mA	-40°C	VDD = 4.2V	FOSC = 4 MHz (**PRI_RUN HS+PLL**)
		7.40	15.00	mA	+25°C		
		7.30	14.00	mA	+85°C		
	All devices	10.00	21.00	mA	-40°C	VDD = 5.0V	FOSC = 4 MHz (**PRI_RUN HS+PLL**)
		10.00	20.00	mA	+25°C		
		9.70	19.00	mA	+85°C		
	All devices	17.00	35.00	mA	-40°C	VDD = 4.2V	FOSC = 10 MHz (**PRI_RUN HS+PLL**)
		17.00	35.00	mA	+25°C		
		17.00	35.00	mA	+85°C		
	All devices	23.00	40.00	mA	-40°C	VDD = 5.0V	FOSC = 10 MHz (**PRI_RUN HS+PLL**)
		23.00	40.00	mA	+25°C		
		23.00	40.00	mA	+85°C		

Legend: Shading of rows is to assist in readability of the table.

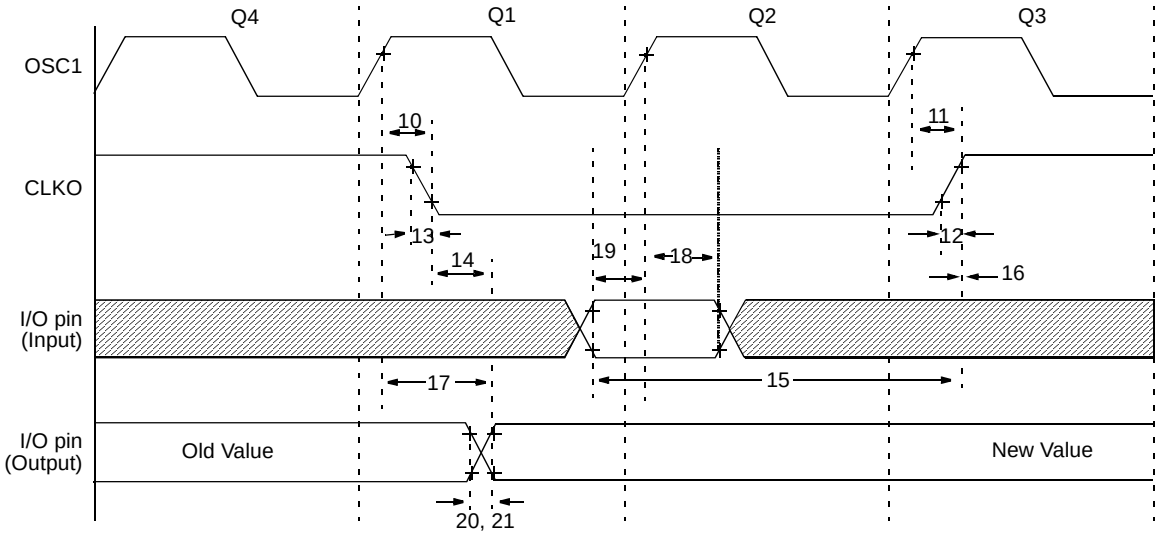
Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption. The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

4: Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

FIGURE 27-6: CLKO AND I/O TIMING



Note: Refer to Figure 27-4 for load conditions.

TABLE 27-9: CLKO AND I/O TIMING REQUIREMENTS

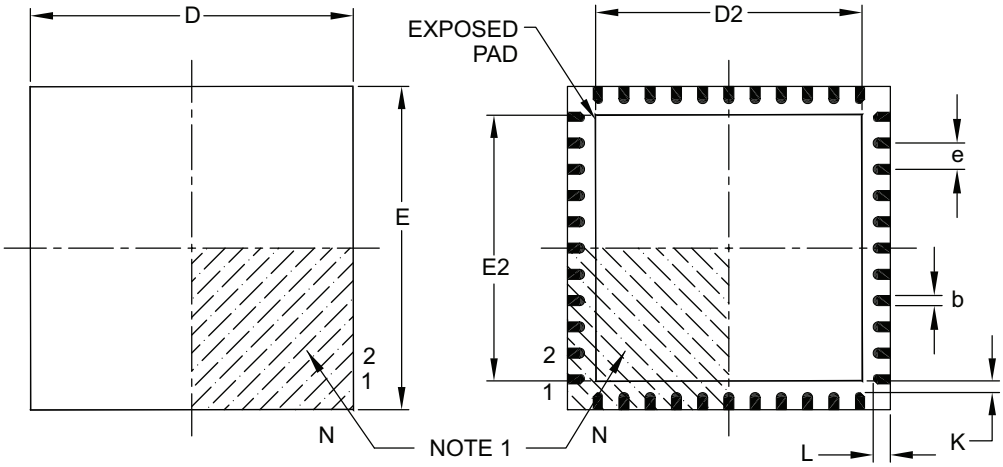
Param No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
10	TosH2ckL	OSC1 ↑ to CLKO ↓		—	75	200	ns	**(Note 1)**
11	TosH2ckH	OSC1 ↑ to CLKO ↑		—	75	200	ns	**(Note 1)**
12	TckR	CLKO Rise Time		—	35	100	ns	**(Note 1)**
13	TckF	CLKO Fall Time		—	35	100	ns	**(Note 1)**
14	TckL2ioV	CLKO ↓ to Port Out Valid		—	—	0.5 TCY + 20	ns	**(Note 1)**
15	TioV2ckH	Port In Valid before CLKO ↑		0.25 TCY + 25	—	—	ns	**(Note 1)**
16	TckH2ioI	Port In Hold after CLKO ↑		0	—	—	ns	**(Note 1)**
17	TosH2ioV	OSC1 ↑ (Q1 cycle) to Port Out Valid		—	50	150	ns	
18	TosH2ioI	OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time)	PIC18**F**XXXX	100	—	—	ns	
18A			PIC18**LF**XXXX	200	—	—	ns	VDD = 2.0V
19	TioV2osH	Port Input Valid to OSC1 ↑ (I/O in setup time)		0	—	—	ns	
20	TioR	Port Output Rise Time	PIC18**F**XXXX	—	10	25	ns	
20A			PIC18**LF**XXXX	—	—	60	ns	VDD = 2.0V
21	TioF	Port Output Fall Time	PIC18**F**XXXX	—	10	25	ns	
21A			PIC18**LF**XXXX	—	—	60	ns	VDD = 2.0V
22†	TinP	INT pin High or Low Time		TCY	—	—	ns	
23†	TrbP	RB7:RB4 Change INT High or Low Time		TCY	—	—	ns	
24†	TrcP	RC7:RC4 Change INT High or Low Time		20			ns	

† These parameters are asynchronous events not related to any internal clock edges.

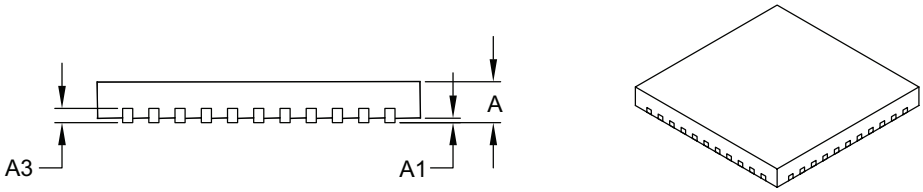
Note 1: Measurements are taken in RC mode, where CLKO output is 4 x TOSC.

44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW **BOTTOM VIEW**



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		44	
Pitch	e		0.65 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Contact Thickness	A3		0.20 REF	
Overall Width	E		8.00 BSC	
Exposed Pad Width	E2	6.30	6.45	6.80
Overall Length	D		8.00 BSC	
Exposed Pad Length	D2	6.30	6.45	6.80
Contact Width	b	0.25	0.30	0.38
Contact Length	L	0.30	0.40	0.50
Contact-to-Exposed Pad	K	0.20	–	–

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103B