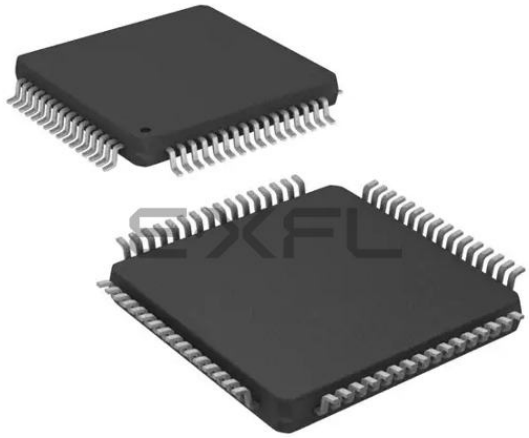
Microchip Technology - DSPIC30F5011-30I/PT Datasheet



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	dsPIC
Core Size	16-Bit
Speed	30 MIPs
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	AC'97, Brown-out Detect/Reset, I²S, LVD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	66KB (22K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 16x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f5011-30i-pt

FIGURE 1-1: dsPIC30F5011 BLOCK DIAGRAM

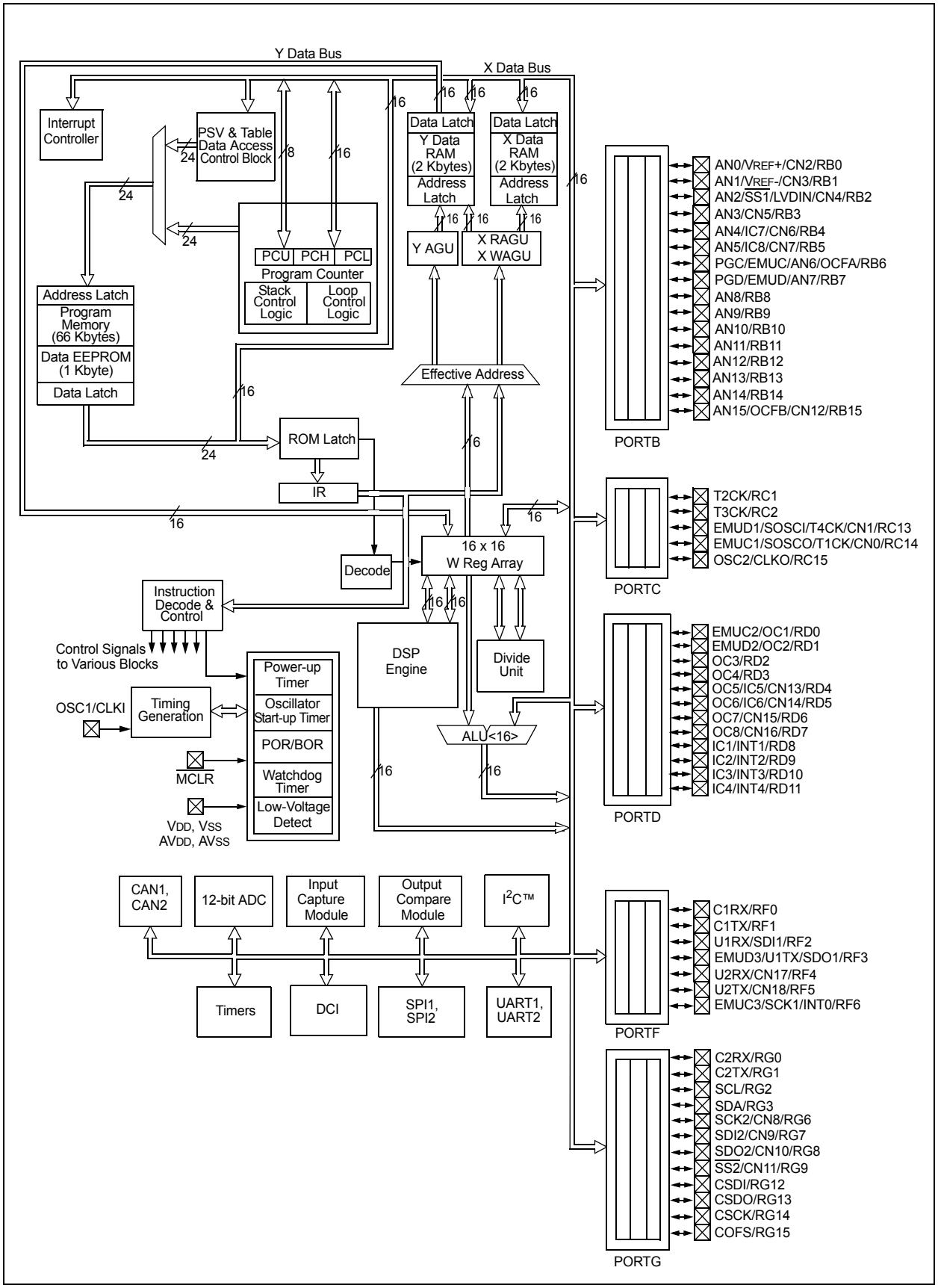
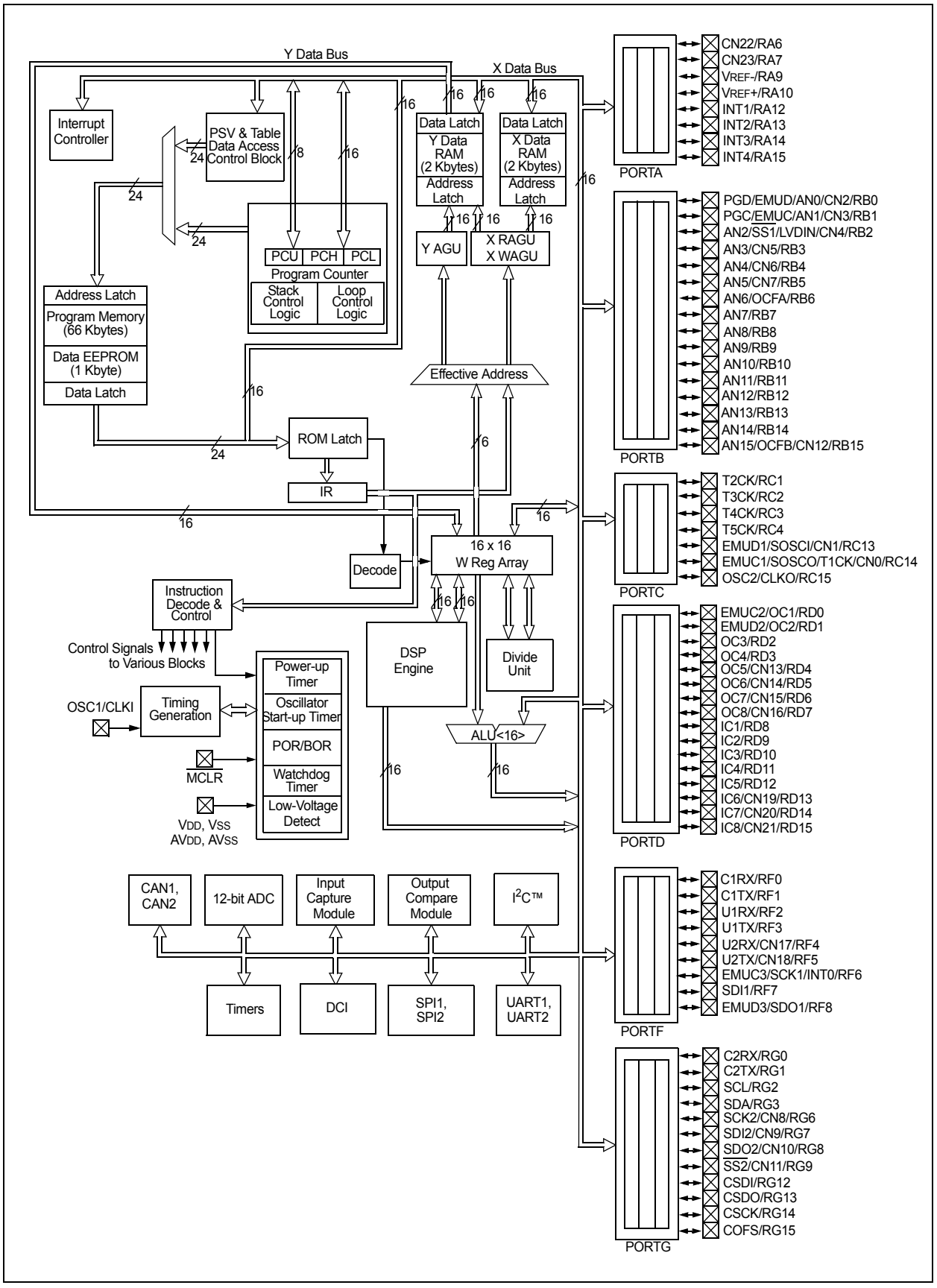


FIGURE 1-2: dsPIC30F5013 BLOCK DIAGRAM



NOTES:

7.3.2 WRITING A BLOCK OF DATA EEPROM

To write a block of data EEPROM, write to all sixteen latches first, and then set the NVMCON register and program the block.

EXAMPLE 7-5: DATA EEPROM BLOCK WRITE

```
MOV        #LOW_ADDR_WORD,W0  ; Init pointer
MOV        #HIGH_ADDR_WORD,W1
MOV        W1,TBLPAG
MOV        #data1,W2          ; Get 1st data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data2,W2          ; Get 2nd data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data3,W2          ; Get 3rd data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data4,W2          ; Get 4th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data5,W2          ; Get 5th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data6,W2          ; Get 6th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data7,W2          ; Get 7th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data8,W2          ; Get 8th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data9,W2          ; Get 9th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data10,W2         ; Get 10th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data11,W2         ; Get 11th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data12,W2         ; Get 12th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data13,W2         ; Get 13th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data14,W2         ; Get 14th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data15,W2         ; Get 15th data
TBLWTL     W2,[ W0]++         ; write data
MOV        #data16,W2         ; Get 16th data
TBLWTL     W2,[ W0]++         ; write data. The NVMADR captures last table access address.
MOV        #0x400A,W0         ; Select data EEPROM for multi word op
MOV        W0,NVMCON          ; Operate Key to allow program operation
DISI       #5                 ; Block all interrupts with priority <7 for next 5 instructions
MOV        #0x55,W0
MOV        W0,NVMKEY          ; Write the 0x55 key
MOV        #0xAA,W1
MOV        W1,NVMKEY          ; Write the 0xAA key
BSET       NVMCON,#WR         ; Start write cycle
NOP
NOP
```

7.4 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

7.5 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, the WREN bit is cleared, and the Power-up Timer prevents EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

TABLE 8-5: PORTD REGISTER MAP FOR dsPIC30F5011(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISD	02D2	—	—	—	—	TRISD11	TRISD10	TRISD9	TRISD8	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	0000 1111 1111 1111
PORTD	02D4	—	—	—	—	RD11	RD10	RD9	RD8	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	0000 0000 0000 0000
LATD	02D6	—	—	—	—	LATD11	LATD10	LATD9	LATD8	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	0000 0000 0000 0000

Legend: — = unimplemented, read as '0'
Note 1: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

TABLE 8-6: PORTD REGISTER MAP FOR dsPIC30F5013(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISD	02D2	TRISD15	TRISD14	TRISD13	TRISD12	TRISD11	TRISD10	TRISD9	TRISD8	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111 1111 1111
PORTD	02D4	RD15	RD14	RD13	RD12	RD11	RD10	RD9	RD8	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	0000 0000 0000 0000
LATD	02D6	LATD15	LATD14	LATD13	LATD12	LATD11	LATD10	LATD9	LATD8	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	0000 0000 0000 0000

Note 1: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

TABLE 8-7: PORTF REGISTER MAP FOR dsPIC30F5011(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISF	02DE	—	—	—	—	—	—	—	—	—	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	0000 0000 0111 1111
PORTF	02E0	—	—	—	—	—	—	—	—	—	RF6	RF5	RF4	RF3	RF2	RF1	RF0	0000 0000 0000 0000
LATF	02E2	—	—	—	—	—	—	—	—	—	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	0000 0000 0000 0000

Legend: — = unimplemented, read as '0'
Note 1: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

TABLE 8-8: PORTF REGISTER MAP FOR dsPIC30F5013(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISF	02DE	—	—	—	—	—	—	—	TRISF8	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	0000 0001 1111 1111
PORTF	02E0	—	—	—	—	—	—	—	RF8	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	0000 0000 0000 0000
LATF	02E2	—	—	—	—	—	—	—	LATF8	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	0000 0000 0000 0000

Legend: — = unimplemented, read as '0'

Note 1: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

TABLE 8-9: PORTG REGISTER MAP FOR dsPIC30F5011/5013(1)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISG	02E4	TRISG15	TRISG14	TRISG13	TRISG12	—	—	TRISG9	TRISG8	TRISG7	TRISG6	—	—	TRISG3	TRISG2	TRISG1	TRISG0	1111 0011 1100 1111
PORTG	02E6	RG15	RG14	RG13	RG12	—	—	RG9	RG8	RG7	RG6	—	—	RG3	RG2	RG1	RG0	0000 0000 0000 0000
LATG	02E8	LATG15	LATG14	LATG13	LATG12	—	—	LATG9	LATG8	LATG7	LATG6	—	—	LATG3	LATG2	LATG1	LATG0	0000 0000 0000 0000

Legend: — = unimplemented, read as '0'

Note 1: Refer to the "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

15.4.1 10-BIT MODE SLAVE TRANSMISSION

Once a slave is addressed in this fashion with the full 10-bit address (we will refer to this state as "PRIOR_ADDR_MATCH"), the master can begin sending data bytes for a slave reception operation.

15.4.2 10-BIT MODE SLAVE RECEPTION

Once addressed, the master can generate a Repeated Start, Reset the high byte of the address and set the R_W bit without generating a Stop bit, thus initiating a slave transmit operation.

15.5 Automatic Clock Stretch

In the Slave modes, the module can synchronize buffer reads and write to the master device by clock stretching.

15.5.1 TRANSMIT CLOCK STRETCHING

Both 10-bit and 7-bit Transmit modes implement clock stretching by asserting the SCLREL bit after the falling edge of the ninth clock, if the TBF bit is cleared, indicating the buffer is empty.

In Slave Transmit modes, clock stretching is always performed irrespective of the STREN bit.

Clock synchronization takes place following the ninth clock of the transmit sequence. If the device samples an $\overline{ACK}$ on the falling edge of the ninth clock and if the TBF bit is still clear, then the SCLREL bit is automatically cleared. The SCLREL being cleared to '0' will assert the SCL line low. The user's ISR must set the SCLREL 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 I2CTRN before the master device can initiate another transmit sequence.

Note 1: If the user loads the contents of I2CTRN, setting the TBF bit before the falling edge of the ninth clock, the SCLREL bit will not be cleared and clock stretching will not occur.

2: The SCLREL bit can be set in software, regardless of the state of the TBF bit.

15.5.2 RECEIVE CLOCK STRETCHING

The STREN bit in the I2CCON register can be used to enable clock stretching in Slave Receive mode. When the STREN bit is set, the SCL pin will be held low at the end of each data receive sequence.

15.5.3 CLOCK STRETCHING DURING 7-BIT ADDRESSING (STREN = 1)

When the STREN bit is set in Slave Receive mode, the SCL line is held low when the buffer register is full. The method for stretching the SCL output is the same for both 7 and 10-bit Addressing modes.

Clock stretching takes place following the ninth clock of the receive sequence. On the falling edge of the ninth clock at the end of the ACK sequence, if the RBF bit is set, the SCLREL bit is automatically cleared, forcing the SCL output to be held low. The user's ISR must set the SCLREL bit 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 I^2CRCV before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring.

Note 1: If the user reads the contents of the I2CRCV, clearing the RBF bit before the falling edge of the ninth clock, the SCLREL bit will not be cleared and clock stretching will not occur.

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

15.5.4 CLOCK STRETCHING DURING 10-BIT ADDRESSING (STREN = 1)

Clock stretching takes place automatically during the addressing sequence. Because this module has a register for the entire address, it is not necessary for the protocol to wait for the address to be updated.

After the address phase is complete, clock stretching will occur on each data receive or transmit sequence as was described earlier.

15.6 Software Controlled Clock Stretching (STREN = 1)

When the STREN bit is '1', the SCLREL bit may be cleared by software to allow software to control the clock stretching. The logic will synchronize writes to the SCLREL bit with the SCL clock. Clearing the SCLREL bit will not assert the SCL output until the module detects a falling edge on the SCL output and SCL is sampled low. If the SCLREL bit is cleared by the user while the SCL line has been sampled low, the SCL output will be asserted (held low). The SCL output will remain low until the SCLREL bit is set, and all other devices on the I^2C bus have de-asserted SCL. This ensures that a write to the SCLREL bit will not violate the minimum high time requirement for SCL.

If the STREN bit is '0', a software write to the SCLREL bit will be disregarded and have no effect on the SCLREL bit.

16.3.4 TRANSMIT INTERRUPT

The transmit interrupt flag (U1TXIF or U2TXIF) is located in the corresponding interrupt flag register.

The transmitter generates an edge to set the UxTXIF bit. The condition for generating the interrupt depends on the UTXISEL control bit:

a) If UTXISEL = 0, an interrupt is generated when a word is transferred from the transmit buffer to the Transmit Shift register (UxTSR). This implies that the transmit buffer has at least one empty word.
b) If UTXISEL = 1, an interrupt is generated when a word is transferred from the transmit buffer to the Transmit Shift register (UxTSR) and the transmit buffer is empty.

Switching between the two Interrupt modes during operation is possible and sometimes offers more flexibility.

16.3.5 TRANSMIT BREAK

Setting the UTXBRK bit (UxSTA<11>) will cause the UxTX line to be driven to logic '0'. The UTXBRK bit overrides all transmission activity. Therefore, the user should generally wait for the transmitter to be Idle before setting UTXBRK.

To send a break character, the UTXBRK bit must be set by software and must remain set for a minimum of 13 baud clock cycles. The UTXBRK bit is then cleared by software to generate Stop bits. The user must wait for a duration of at least one or two baud clock cycles in order to ensure a valid Stop bit(s) before reloading the UxTXB, or starting other transmitter activity. Transmission of a break character does not generate a transmit interrupt.

16.4 Receiving Data

16.4.1 RECEIVING IN 8-BIT OR 9-BIT DATA MODE

The following steps must be performed while receiving 8-bit or 9-bit data:

1. Set up the UART (see **Section 16.3.1 "Transmitting in 8-bit data mode"**).
2. Enable the UART (see **Section 16.3.1 "Transmitting in 8-bit data mode"**).
3. A receive interrupt will be generated when one or more data words have been received, depending on the receive interrupt settings specified by the URXISEL bits (UxSTA<7:6>).
4. Read the OERR bit to determine if an overrun error has occurred. The OERR bit must be reset in software.
5. Read the received data from UxRXREG. The act of reading UxRXREG will move the next word to the top of the receive FIFO, and the PERR and FERR values will be updated.

16.4.2 RECEIVE BUFFER (UxRXB)

The receive buffer is 4 words deep. Including the Receive Shift register (UxRSR), the user effectively has a 5-word deep FIFO buffer.

URXDA (UxSTA<0>) = 1 indicates that the receive buffer has data available. URXDA = 0 implies that the buffer is empty. If a user attempts to read an empty buffer, the old values in the buffer will be read and no data shift will occur within the FIFO.

The FIFO is reset during any device Reset. It is not affected when the device enters or wakes up from a Power-Saving mode.

16.4.3 RECEIVE INTERRUPT

The receive interrupt flag (U1RXIF or U2RXIF) can be read from the corresponding interrupt flag register. The interrupt flag is set by an edge generated by the receiver. The condition for setting the receive interrupt flag depends on the settings specified by the URXISEL<1:0> (UxSTA<7:6>) control bits.

a) If URXISEL<1:0> = 00 or 01, an interrupt is generated every time a data word is transferred from the Receive Shift register (UxRSR) to the receive buffer. There may be one or more characters in the receive buffer.
b) If URXISEL<1:0> = 10, an interrupt is generated when a word is transferred from the Receive Shift register (UxRSR) to the receive buffer, which as a result of the transfer, contains 3 characters.
c) If URXISEL<1:0> = 11, an interrupt is set when a word is transferred from the Receive Shift register (UxRSR) to the receive buffer, which as a result of the transfer, contains 4 characters (i.e., becomes full).

Switching between the Interrupt modes during operation is possible, though generally not advisable during normal operation.

16.5 Reception Error Handling

16.5.1 RECEIVE BUFFER OVERRUN ERROR (OERR BIT)

The OERR bit (UxSTA<1>) is set if all of the following conditions occur:

a) The receive buffer is full.
b) The Receive Shift register is full, but unable to transfer the character to the receive buffer.
c) The Stop bit of the character in the UxRSR is detected, indicating that the UxRSR needs to transfer the character to the buffer.

Once OERR is set, no further data is shifted in UxRSR (until the OERR bit is cleared in software or a Reset occurs). The data held in UxRSR and UxRXREG remains valid.

16.5.2 FRAMING ERROR (FERR)

The FERR bit (UxSTA<2>) is set if a '0' is detected instead of a Stop bit. If two Stop bits are selected, both Stop bits must be '1', otherwise FERR will be set. The read-only FERR bit is buffered along with the received data. It is cleared on any Reset.

16.5.3 PARITY ERROR (PERR)

The PERR bit (UxSTA<3>) is set if the parity of the received word is incorrect. This error bit is applicable only if a Parity mode (odd or even) is selected. The read-only PERR bit is buffered along with the received data bytes. It is cleared on any Reset.

16.5.4 IDLE STATUS

When the receiver is active (i.e., between the initial detection of the Start bit and the completion of the Stop bit), the RIDLE bit (UxSTA<4>) is '0'. Between the completion of the Stop bit and detection of the next Start bit, the RIDLE bit is '1', indicating that the UART is Idle.

16.5.5 RECEIVE BREAK

The receiver will count and expect a certain number of bit times based on the values programmed in the PDSEL (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.

If the break is longer than 13 bit times, the reception is considered complete after the number of bit times specified by PDSEL and STSEL. The URXDA bit is set, FERR is set, zeros are loaded into the receive FIFO, interrupts are generated if appropriate and the RIDLE bit is set.

When the module receives a long break signal and the receiver has detected the Start bit, the data bits and the invalid Stop bit (which sets the FERR), the receiver must wait for a valid Stop bit before looking for the next Start bit. It cannot assume that the break condition on the line is the next Start bit.

Break is regarded as a character containing all '0's with the FERR bit set. The break character is loaded into the buffer. No further reception can occur until a Stop bit is received. Note that RIDLE goes high when the Stop bit has not yet been received.

16.6 Address Detect Mode

Setting the ADDEN bit (UxSTA<5>) enables this special mode in which a 9th bit (URX8) value of '1' identifies the received word as an address, rather than data. This mode is only applicable for 9-bit data communication. The URXISEL control bit does not have any impact on interrupt generation in this mode since an interrupt (if enabled) will be generated every time the received word has the 9th bit set.

16.7 Loopback Mode

Setting the LPBACK bit enables this special mode in which the UxTX pin is internally connected to the UxRX pin. When configured for the Loopback mode, the UxRX pin is disconnected from the internal UART receive logic. However, the UxTX pin still functions as in a normal operation.

To select this mode:

1. Configure UART for desired mode of operation.
2. Set LPBACK = 1 to enable Loopback mode.
3. Enable transmission as defined in **Section 16.3 "Transmitting Data"**.

16.8 Baud Rate Generator

The UART has a 16-bit Baud Rate Generator to allow maximum flexibility in baud rate generation. The Baud Rate Generator register (UxBRG) is readable and writable. The baud rate is computed as follows:

BRG = 16-bit value held in UxBRG register (0 through 65535)

FCY = Instruction Clock Rate (1/TCY)

The Baud Rate is given by Equation 16-1.

EQUATION 16-1: BAUD RATE

$$Baud\ Rate = F_{CY}/(16*(BRG+1))$$

Therefore, the maximum baud rate possible is

FCY/16 (if BRG = 0),

and the minimum baud rate possible is

FCY/(16* 65536).

With a full 16-bit Baud Rate Generator at 30 MIPS operation, the minimum baud rate achievable is 28.5 bps.

17.3 Modes of Operation

The CAN module can operate in one of several operation modes selected by the user. These modes include:

- Initialization Mode
- Disable Mode
- Normal Operation Mode
- Listen Only Mode
- Loopback Mode
- Error Recognition Mode

Modes are requested by setting the REQOP<2:0> bits (CiCTRL<10:8>). Entry into a mode is Acknowledged by monitoring the OPMODE<2:0> bits (CiCTRL<7:5>). The module will not change the mode and the OPMODE bits until a change in mode is acceptable, generally during bus Idle time which is defined as at least 11 consecutive recessive bits.

17.3.1 INITIALIZATION MODE

In the Initialization mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to configuration registers that are access restricted in other modes. The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is on-line. The CAN module will not be allowed to enter the Configuration mode while a transmission is taking place. The Configuration mode serves as a lock to protect the following registers.

- All Module Control Registers
- Baud Rate and Interrupt Configuration Registers
- Bus Timing Registers
- Identifier Acceptance Filter Registers
- Identifier Acceptance Mask Registers

17.3.2 DISABLE MODE

In Disable mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity, however, any pending interrupts will remain and the error counters will retain their value.

If the REQOP<2:0> bits (CiCTRL<10:8>) = 001, the module will enter the Module Disable mode. If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an Idle bus, then accept the module disable command. When the OPMODE<2:0> bits (CiCTRL<7:5>) = 001, that indicates whether the module successfully went into Module Disable mode. The I/O pins will revert to normal I/O function when the module is in the Module Disable mode.

The module can be programmed to apply a low-pass filter function to the CiRX input line while the module or the CPU is in Sleep mode. The WAKFIL bit (CiCFG2<14>) enables or disables the filter.

Note:	Typically, if the CAN module is allowed to transmit in a particular mode of operation and a transmission is requested immediately after the CAN module has been placed in that mode of operation, the module waits for 11 consecutive recessive bits on the bus before starting transmission. If the user switches to Disable mode within this 11-bit period, then this transmission is aborted and the corresponding TXABT bit is set and TXREQ bit is cleared.

17.3.3 NORMAL OPERATION MODE

Normal operating mode is selected when REQOP<2:0> = 000. In this mode, the module is activated and the I/O pins assume the CAN bus functions. The module transmits and receives CAN bus messages via the CxTX and CxRX pins.

17.3.4 LISTEN ONLY MODE

If the Listen Only mode is activated, the module on the CAN bus is passive. The transmitter buffers revert to the port I/O function. The receive pins remain inputs. For the receiver, no error flags or Acknowledge signals are sent. The error counters are deactivated in this state. The Listen Only mode can be used for detecting the baud rate on the CAN bus. To use this, it is necessary that there are at least two further nodes that communicate with each other.

17.3.5 LISTEN ALL MESSAGES MODE

The module can be set to ignore all errors and receive any message. The Listen All Messages mode is activated by setting the REQOP<2:0> bits to '111'. In this mode, the data which is in the message assembly buffer until the time an error occurred, is copied in the receive buffer and can be read via the CPU interface.

17.3.6 LOOPBACK MODE

If the Loopback mode is activated, the module connects the internal transmit signal to the internal receive signal at the module boundary. The transmit and receive pins revert to their port I/O function.

- Receive Error Interrupts:

  A receive error interrupt will be indicated by the ERRIF bit. This bit shows that an error condition occurred. The source of the error can be determined by checking the bits in the CAN Interrupt Status register, CiINTF.

  - Invalid Message Received:

    If any type of error occurred during reception of the last message, an error will be indicated by the IVRIF bit.

  - Receiver Overrun:

    The RXnOVR bit indicates that an overrun condition occurred.

  - Receiver Warning:

    The RXWAR bit indicates that the receive error counter (RERRCNT<7:0>) has reached the warning limit of 96.

  - Receiver Error Passive:

    The RXEP bit indicates that the receive error counter has exceeded the error passive limit of 127 and the module has gone into error passive state.

17.5 Message Transmission

17.5.1 TRANSMIT BUFFERS

The CAN module has three transmit buffers. Each of the three buffers occupies 14 bytes of data. Eight of the bytes are the maximum 8 bytes of the transmitted message. Five bytes hold the standard and extended identifiers and other message arbitration information.

17.5.2 TRANSMIT MESSAGE PRIORITY

Transmit priority is a prioritization within each node of the pending transmittable messages. There are 4 levels of transmit priority. If TXPRI<1:0> (CiTXnCON<1:0>, where n = 0, 1 or 2 represents a particular transmit buffer) for a particular message buffer is set to '11', that buffer has the highest priority. If TXPRI<1:0> for a particular message buffer is set to '10' or '01', that buffer has an intermediate priority. If TXPRI<1:0> for a particular message buffer is '00', that buffer has the lowest priority.

17.5.3 TRANSMISSION SEQUENCE

To initiate transmission of the message, the TXREQ bit (CiTXnCON<3>) must be set. The CAN bus module resolves any timing conflicts between setting of the TXREQ bit and the Start-of-Frame (SOF), ensuring that if the priority was changed, it is resolved correctly before the SOF occurs. When TXREQ is set, the TXABT (CiTXnCON<6>), TXLARB (CiTXnCON<5>) and TXERR (CiTXnCON<4>) flag bits are automatically cleared.

Setting TXREQ bit simply flags a message buffer as enqueued for transmission. When the module detects an available bus, it begins transmitting the message which has been determined to have the highest priority.

If the transmission completes successfully on the first attempt, the TXREQ bit is cleared automatically, and an interrupt is generated if TXIE was set.

If the message transmission fails, one of the error condition flags will be set, and the TXREQ bit will remain set indicating that the message is still pending for transmission. If the message encountered an error condition during the transmission attempt, the TXERR bit will be set, and the error condition may cause an interrupt. If the message loses arbitration during the transmission attempt, the TXLARB bit is set. No interrupt is generated to signal the loss of arbitration.

17.5.4 ABORTING MESSAGE TRANSMISSION

The system can also abort a message by clearing the TXREQ bit associated with each message buffer. Setting the ABAT bit (CiCTRL<12>) will request an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets the TXABT bit and the TXnIF flag is not automatically set.

17.5.5 TRANSMISSION ERRORS

The CAN module will detect the following transmission errors:

- Acknowledge Error
- Form Error
- Bit Error

These transmission errors will not necessarily generate an interrupt but are indicated by the transmission error counter. However, each of these errors will cause the transmission error counter to be incremented by one. Once the value of the error counter exceeds the value of 96, the ERRIF bit (CiINTF<5>) and the TXWAR bit (CiINTF<10>) are set. Once the value of the error counter exceeds the value of 96, an interrupt is generated and the TXWAR bit in the Error Flag register is set.

21.0 INSTRUCTION SET SUMMARY

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the "*dsPIC30F Family Reference Manual*" (DS70046). For more information on the device instruction set and programming, refer to the *"16-bit MCU and DSC Programmer's Reference Manual"* (DS70157).

The dsPIC30F instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from PIC MCU instruction sets.

Most instructions are a single program memory word (24 bits). Only three instructions require two program memory locations.

Each single-word instruction is a 24-bit word divided into an 8-bit opcode which specifies the instruction type, and one or more operands which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into five basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- DSP operations
- Control operations

Table 21-1 shows the general symbols used in describing the instructions.

The dsPIC30F instruction set summary in Table 21-2 lists all the instructions, along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand which is typically a register 'Wb' without any address modifier
- The second source operand which is typically a register 'Ws' with or without an address modifier
- The destination of the result which is typically a register 'Wd' with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value 'f'
- The destination, which could either be the file register 'f' or the W0 register, which is denoted as 'WREG'

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of 'Ws' or 'f')
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register 'Wb')

The literal instructions that involve data movement may use some of the following operands:

- A literal value to be loaded into a W register or file register (specified by the value of 'k')
- The W register or file register where the literal value is to be loaded (specified by 'Wb' or 'f')

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand which is a register 'Wb' without any address modifier
- The second source operand which is a literal value
- The destination of the result (only if not the same as the first source operand) which is typically a register 'Wd' with or without an address modifier

The MAC class of DSP instructions may use some of the following operands:

- The accumulator (A or B) to be used (required operand)
- The W registers to be used as the two operands
- The X and Y address space prefetch operations
- The X and Y address space prefetch destinations
- The accumulator write back destination

The other DSP instructions do not involve any multiplication, and may include:

- The accumulator to be used (required)
- The source or destination operand (designated as Wso or Wdo, respectively) with or without an address modifier
- The amount of shift specified by a W register 'Wn' or a literal value

The control instructions may use some of the following operands:

- A program memory address
- The mode of the table read and table write instructions

TABLE 21-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of Words	# of Cycles	Status Flags Affected
48	MPY	MPY	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wm by Wn to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		MPY	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square Wm to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
49	MPY.N	MPY.N	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wm by Wn) to Accumulator	1	1	None
50	MSC	MSC	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd, AWB	Multiply and Subtract from Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
51	MUL	MUL.SS	Wb,Ws,Wnd	{Wnd+1, Wnd} = signed(Wb) * signed(Ws)	1	1	None
		MUL.SU	Wb,Ws,Wnd	{Wnd+1, Wnd} = signed(Wb) * unsigned(Ws)	1	1	None
		MUL.US	Wb,Ws,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * signed(Ws)	1	1	None
		MUL.UU	Wb,Ws,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * unsigned(Ws)	1	1	None
		MUL.SU	Wb,#lit5,Wnd	{Wnd+1, Wnd} = signed(Wb) * unsigned(lit5)	1	1	None
		MUL.UU	Wb,#lit5,Wnd	{Wnd+1, Wnd} = unsigned(Wb) * unsigned(lit5)	1	1	None
		MUL	f	W3:W2 = f * WREG	1	1	None
52	NEG	NEG	Acc	Negate Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		NEG	f	f = $\bar{f}$ + 1	1	1	C,DC,N,OV,Z
		NEG	f,WREG	WREG = $\bar{f}$ + 1	1	1	C,DC,N,OV,Z
		NEG	Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1	C,DC,N,OV,Z
53	NOP	NOP		No Operation	1	1	None
		NOPR		No Operation	1	1	None
54	POP	POP	f	Pop f from Top-of-Stack (TOS)	1	1	None
		POP	Wdo	Pop from Top-of-Stack (TOS) to Wdo	1	1	None
		POP.D	Wnd	Pop from Top-of-Stack (TOS) to W(nd):W(nd+1)	1	2	None
		POP.S		Pop Shadow Registers	1	1	All
55	PUSH	PUSH	f	Push f to Top-of-Stack (TOS)	1	1	None
		PUSH	Wso	Push Wso to Top-of-Stack (TOS)	1	1	None
		PUSH.D	Wns	Push W(ns):W(ns+1) to Top-of-Stack (TOS)	1	2	None
		PUSH.S		Push Shadow Registers	1	1	None
56	PWRSAV	PWRSAV	#lit1	Go into Sleep or Idle mode	1	1	WDTO,Sleep
57	RCALL	RCALL	Expr	Relative Call	1	2	None
		RCALL	Wn	Computed Call	1	2	None
58	REPEAT	REPEAT	#lit14	Repeat Next Instruction lit14+1 times	1	1	None
		REPEAT	Wn	Repeat Next Instruction (Wn)+1 times	1	1	None
59	RESET	RESET		Software device Reset	1	1	None
60	RETFIE	RETFIE		Return from interrupt	1	3 (2)	None
61	RETLW	RETLW	#lit10,Wn	Return with literal in Wn	1	3 (2)	None
62	RETURN	RETURN		Return from Subroutine	1	3 (2)	None
63	RLC	RLC	f	f = Rotate Left through Carry f	1	1	C,N,Z
		RLC	f,WREG	WREG = Rotate Left through Carry f	1	1	C,N,Z
		RLC	Ws,Wd	Wd = Rotate Left through Carry Ws	1	1	C,N,Z
64	RLNC	RLNC	f	f = Rotate Left (No Carry) f	1	1	N,Z
		RLNC	f,WREG	WREG = Rotate Left (No Carry) f	1	1	N,Z
		RLNC	Ws,Wd	Wd = Rotate Left (No Carry) Ws	1	1	N,Z
65	RRC	RRC	f	f = Rotate Right through Carry f	1	1	C,N,Z
		RRC	f,WREG	WREG = Rotate Right through Carry f	1	1	C,N,Z
		RRC	Ws,Wd	Wd = Rotate Right through Carry Ws	1	1	C,N,Z
66	RRNC	RRNC	f	f = Rotate Right (No Carry) f	1	1	N,Z
		RRNC	f,WREG	WREG = Rotate Right (No Carry) f	1	1	N,Z
		RRNC	Ws,Wd	Wd = Rotate Right (No Carry) Ws	1	1	N,Z

22.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers and dsPIC® digital signal controllers are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

22.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - In-Circuit Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

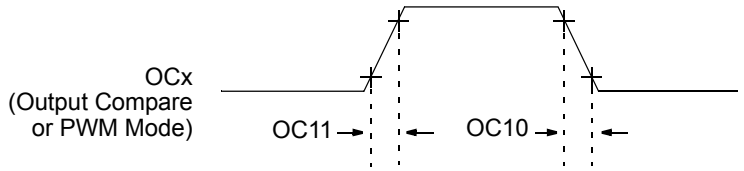
TABLE 23-5: DC CHARACTERISTICS: OPERATING CURRENT (IDD) (CONTINUED)

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 5.5V (unless otherwise stated) Operating temperature -40°C ≤TA ≤+85°C for Industrial -40°C ≤TA ≤+125°C for Extended			
Parameter No.	**Typical(1)**	**Max**	**Units**	**Conditions**		
Operating Current (IDD)(2)						
DC23a	13.5	20	mA	25°C	3.3V	4 MIPS
DC23b	14	21	mA	85°C		
DC23c	15	22.5	mA	125°C		
DC23e	23	34.5	mA	25°C	5V	
DC23f	23.5	35	mA	85°C		
DC23g	24	36	mA	125°C		
DC24a	32	48	mA	25°C	3.3V	10 MIPS
DC24b	32.5	49	mA	85°C		
DC24c	33	49.5	mA	125°C		
DC24e	53.5	80	mA	25°C	5V	
DC24f	54	81	mA	85°C		
DC24g	54	81	mA	125°C		
DC27d	101	152	mA	25°C	5V	20 MIPS
DC27e	100	150	mA	85°C		
DC27f	100	150	mA	125°C		
DC29a	145	217	mA	25°C	5V	30 MIPS
DC29b	144	216	mA	85°C		

Note 1: Data in "Typical" column is at 5V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature also have an impact on the current consumption. The test conditions for all IDD measurements are as follows: OSC1 driven with external square wave from rail to rail. All I/O pins are configured as Inputs and pulled to VDD. MCLR = VDD, WDT, FSCM, LVD and BOR are disabled. CPU, SRAM, Program Memory and Data Memory are operational. No peripheral modules are operating.

FIGURE 23-10: OUTPUT COMPARE MODULE (OCx) TIMING CHARACTERISTICS



Note: Refer to Figure 23-3 for load conditions.

TABLE 23-27: OUTPUT COMPARE MODULE TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 5.5V (unless otherwise stated) Operating temperature -40°C ≤TA ≤+85°C for Industrial -40°C ≤TA ≤+125°C for Extended				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
OC10	TccF	OCx Output Fall Time	—	—	—	ns	See Parameter DO32
OC11	TccR	OCx Output Rise Time	—	—	—	ns	See Parameter DO31

Note 1: These parameters are characterized but not tested in manufacturing.

2: Data in “Typ” column is at 5V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

TABLE 23-29: DCI MODULE (MULTICHANNEL, I^2S MODES) TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 5.5V (unless otherwise stated) Operating temperature -40°C ≤TA ≤+85°C for Industrial -40°C ≤TA ≤+125°C for Extended				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
CS10	TCSCKL	CSCK Input Low Time (CSCK pin is an input)	TCY / 2 + 20	—	—	ns	
		CSCK Output Low Time(3) (CSCK pin is an output)	30	—	—	ns	
CS11	TCSCKH	CSCK Input High Time (CSCK pin is an input)	TCY / 2 + 20	—	—	ns	
		CSCK Output High Time(3) (CSCK pin is an output)	30	—	—	ns	
CS20	TCSCKF	CSCK Output Fall Time(4) (CSCK pin is an output)	—	10	25	ns	
CS21	TCSCKR	CSCK Output Rise Time(4) (CSCK pin is an output)	—	10	25	ns	
CS30	TCSDOF	CSDO Data Output Fall Time(4)	—	10	25	ns	
CS31	TCSDOR	CSDO Data Output Rise Time(4)	—	10	25	ns	
CS35	TDV	Clock edge to CSDO data valid	—	—	10	ns	
CS36	TDIV	Clock edge to CSDO tri-stated	10	—	20	ns	
CS40	TCSDI	Setup time of CSDI data input to CSCK edge (CSCK pin is input or output)	20	—	—	ns	
CS41	THCSDI	Hold time of CSDI data input to CSCK edge (CSCK pin is input or output)	20	—	—	ns	
CS50	TCOFSF	COFS Fall Time (COFS pin is output)	—	10	25	ns	**Note 1**
CS51	TCOFSR	COFS Rise Time (COFS pin is output)	—	10	25	ns	**Note 1**
CS55	TSCOFS	Setup time of COFS data input to CSCK edge (COFS pin is input)	20	—	—	ns	
CS56	THCOFS	Hold time of COFS data input to CSCK edge (COFS pin is input)	20	—	—	ns	

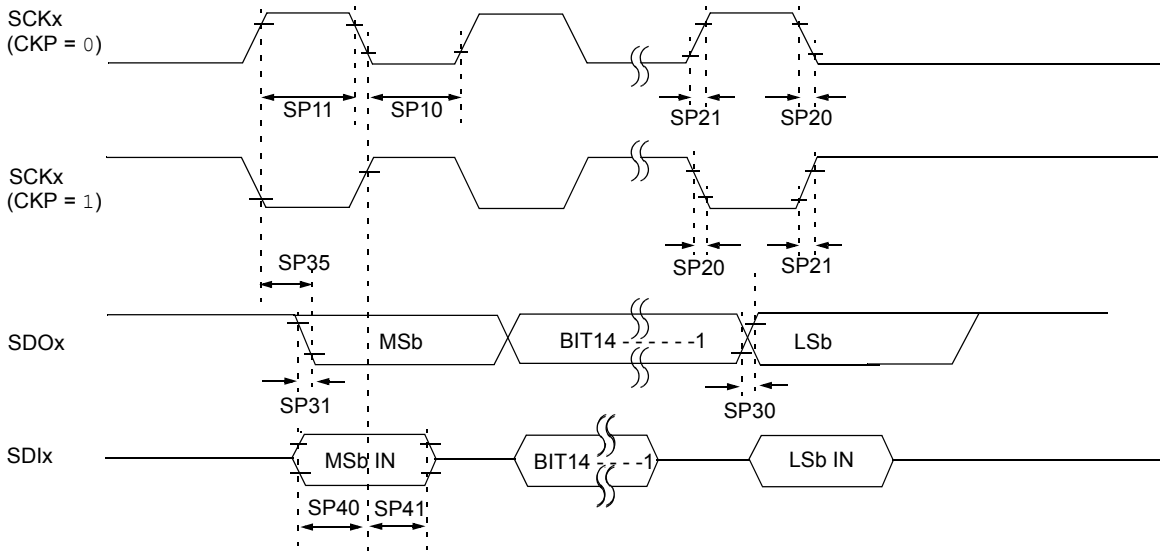
Note 1: These parameters are characterized but not tested in manufacturing.

2: Data in "Typ" column is at 5V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

3: The minimum clock period for CSCK is 100 ns. Therefore, the clock generated in Master mode must not violate this specification.

4: Assumes 50 pF load on all DCI pins.

FIGURE 23-14: SPI MODULE MASTER MODE (CKE = 0) TIMING CHARACTERISTICS



Note: Refer to Figure 23-3 for load conditions.

TABLE 23-31: SPI MASTER MODE (CKE = 0) TIMING REQUIREMENTS

AC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 5.5V (unless otherwise stated) Operating temperature -40°C ≤TA ≤+85°C for Industrial -40°C ≤TA ≤+125°C for Extended				
Param No.	**Symbol**	**Characteristic(1)**	**Min**	**Typ(2)**	**Max**	**Units**	**Conditions**
SP10	TscL	SCKx Output Low Time(3)	TCY / 2	—	—	ns	
SP11	TscH	SCKx Output High Time(3)	TCY/2	—	—	ns	
SP20	TscF	SCKx Output Fall Time(4	—	—	—	ns	See parameter DO32
SP21	TscR	SCKx Output Rise Time(4)	—	—	—	ns	See parameter DO31
SP30	TdoF	SDOx Data Output Fall Time(4)	—	—	—	ns	See parameter DO32
SP31	TdoR	SDOx Data Output Rise Time(4)	—	—	—	ns	See parameter DO31
SP35	TscH2doV, TscL2doV	SDOx Data Output Valid after SCKx Edge	—	—	30	ns	
SP40	TdiV2scH, TdiV2scL	Setup Time of SDIx Data Input to SCKx Edge	20	—	—	ns	
SP41	TscH2diL, TscL2diL	Hold Time of SDIx Data Input to SCKx Edge	20	—	—	ns	

Note 1: These parameters are characterized but not tested in manufacturing.

2: Data in "Typ" column is at 5V, 25°C unless otherwise stated. Parameters are for design guidance only and are not tested.

3: The minimum clock period for SCK is 100 ns. Therefore, the clock generated in Master mode must not violate this specification.

4: Assumes 50 pF load on all SPI pins.

I²C 10-bit Slave Mode Operation 93
Reception 94
Transmission 94
I²C 7-bit Slave Mode Operation 93
Reception 93
Transmission 93
I²C Master Mode Operation 95
Baud Rate Generator 96
Clock Arbitration 96
Multi-Master Communication, Bus Collision and Bus Arbitration 96
Reception 95
Transmission 95
I²C Master Mode Support 95
I²C Module 91
Addresses 93
Bus Data Timing Characteristics
Master Mode 194
Slave Mode 196
Bus Data Timing Requirements
Master Mode 195
Slave Mode 197
Bus Start/Stop Bits Timing Characteristics
Master Mode 194
Slave Mode 196
General Call Address Support 95
Interrupts 95
IPMI Support 95
Operating Function Description 91
Operation During CPU Sleep and Idle Modes 96
Pin Configuration 91
Programmer's Model 91
Register Map 97
Registers 91
Slope Control 95
Software Controlled Clock Stretching (STREN = 1) 94
Various Modes 91
I²S Mode Operation 125
Data Justification 125
Frame and Data Word Length Selection 125
Idle Current (IIDLE) 167
In-Circuit Serial Programming (ICSP) 47, 137
Input Capture (CAPX) Timing Characteristics 183
Input Capture Module 77
Interrupts 78
Register Map 79
Input Capture Operation During Sleep and Idle Modes 78
CPU Idle Mode 78
CPU Sleep Mode 78
Input Capture Timing Requirements 183
Input Change Notification Module 62
dsPIC30F5011 Register Map (Bits 15-8) 62
dsPIC30F5011 Register Map (Bits 7-0) 62
dsPIC30F5013 Register Map (Bits 15-8) 62
dsPIC30F5013 Register Map (Bits 7-0) 62
Instruction Addressing Modes 41
File Register Instructions 41
Fundamental Modes Supported 41
MAC Instructions 42
MCU Instructions 41
Move and Accumulator Instructions 42
Other Instructions 42
Instruction Set
Overview 154
Summary 151
Internal Clock Timing Examples 175
Internet Address 215
Interrupt Controller
Register Map 40
Interrupt Priority 36
Traps 37
Interrupt Sequence 39
Interrupt Stack Frame 39
Interrupts 35

L

Load Conditions 173
Low Voltage Detect (LVD) 147
Low-Voltage Detect Characteristics 170
LVDL Characteristics 171

M

Memory Organization 23
Core Register Map 32
Microchip Internet Web Site 215
Modes of Operation
Disable 109
Initialization 109
Listen All Messages 109
Listen Only 109
Loopback 109
Normal Operation 109
Modulo Addressing 42
Applicability 44
Incrementing Buffer Operation Example 43
Start and End Address 43
W Address Register Selection 43
MPLAB ASM30 Assembler, Linker, Librarian 160
MPLAB Integrated Development Environment Software 159
MPLAB PM3 Device Programmer 162
MPLAB REAL ICE In-Circuit Emulator System 161
MPLINK Object Linker/MPLIB Object Librarian 160

N

NVM
Register Map 51

O

OC/PWM Module Timing Characteristics 185
Operating Current (IDD) 165
Oscillator
Configurations 140
Fail-Safe Clock Monitor 142
Fast RC (FRC) 141
Initial Clock Source Selection 140
Low Power RC (LPRC) 141
LP Oscillator Control 140
Phase Locked Loop (PLL) 141
Start-up Timer (OST) 140
Operating Modes (Table) 138
System Overview 137
Oscillator Selection 137
Oscillator Start-up Timer
Timing Characteristics 178
Timing Requirements 179
Output Compare Interrupts 84
Output Compare Module 81
Register Map 85
Timing Characteristics 184
Timing Requirements 184
Output Compare Operation During CPU Idle Mode 84

Output Compare Sleep Mode Operation 84

P

Packaging Information 203
  Marking 203
Peripheral Module Disable (PMD) Registers 149
Pinout Descriptions 12
PLL Clock Timing Specifications 175
POR. *See* Power-on Reset.
Port Write/Read Example 58
PORTA
  Register Map for dsPIC30F5013 59
PORTB
  Register Map for dsPIC30F5011/5013 59
PORTC
  Register Map for dsPIC30F5011 59
  Register Map for dsPIC30F5013 59
PORTD
  Register Map for dsPIC30F5011 60
  Register Map for dsPIC30F5013 60
PORTF
  Register Map for dsPIC30F5011 60
  Register Map for dsPIC30F5013 61
PORTG
  Register Map for dsPIC30F5011/5013 61
Power Saving Modes 147
  Idle 148
  Sleep 147
  Sleep and Idle 137
Power-Down Current (I_{PD}) 168
Power-up Timer
  Timing Characteristics 178
  Timing Requirements 179
Program Address Space 23
  Construction 24
  Data Access from Program Memory Using Program Space Visibility 26
  Data Access From Program Memory Using Table Instructions 25
  Data Access from, Address Generation 24
  Data Space Window into Operation 27
  Data Table Access (LS Word) 25
  Data Table Access (MS Byte) 26
  Memory Map 23
  Table Instructions
    TBLRDH 25
    TBLRDL 25
    TBLWTH 25
    TBLWTL 25
Program and EEPROM Characteristics 172
Program Counter 16
Programmable 137
Programmer's Model 16
  Diagram 17
Programming Operations 49
  Algorithm for Program Flash 49
  Erasing a Row of Program Memory 49
  Initiating the Programming Sequence 50
  Loading Write Latches 50
Protection Against Accidental Writes to OSCCON 142

R

Reader Response 216
Reset 137, 143
  BOR, Programmable 145
  Brown-out Reset (BOR) 137
  Oscillator Start-up Timer (OST) 137
  POR
    Operating without FSCM and PWRT 145
    With Long Crystal Start-up Time 145
  POR (Power-on Reset) 143
  Power-on Reset (POR) 137
  Power-up Timer (PWRT) 137
Reset Sequence 37
  Reset Sources 37
Reset Sources
  Brown-out Reset (BOR) 37
  Illegal Instruction Trap 37
  Trap Lockout 37
  Uninitialized W Register Trap 37
  Watchdog Time-out 37
Reset Timing Characteristics 178
Reset Timing Requirements 179
Run-Time Self-Programming (RTSP) 47

S

Simple Capture Event Mode 77
  Buffer Operation 78
  Hall Sensor Mode 78
  Prescaler 77
  Timer2 and Timer3 Selection Mode 78
Simple OC/PWM Mode Timing Requirements 185
Simple Output Compare Match Mode 82
Simple PWM Mode 82
  Input Pin Fault Protection 82
  Period 83
Software Simulator (MPLAB SIM) 161
Software Stack Pointer, Frame Pointer 16
  CALL Stack Frame 31
SPI Module 87
  Framed SPI Support 87
  Operating Function Description 87
  Operation During CPU Idle Mode 89
  Operation During CPU Sleep Mode 89
  SDOx Disable 87
  Slave Select Synchronization 89
  SPI1 Register Map 90
  SPI2 Register Map 90
  Timing Characteristics
    Master Mode (CKE = 0) 189
    Master Mode (CKE = 1) 190
    Slave Mode (CKE = 1) 191, 192
  Timing Requirements
    Master Mode (CKE = 0) 189
    Master Mode (CKE = 1) 190
    Slave Mode (CKE = 0) 191
    Slave Mode (CKE = 1) 193
  Word and Byte Communication 87
Status Bits, Their Significance and the Initialization Condition for RCON Register, Case 1 146
Status Bits, Their Significance and the Initialization Condition for RCON Register, Case 2 146
Status Register 16
Symbols Used in Opcode Descriptions 152
System Integration 137
  Register Map 150

T

Table Instruction Operation Summary 47
Temperature and Voltage Specifications
  AC 173
Timer1 Module 63