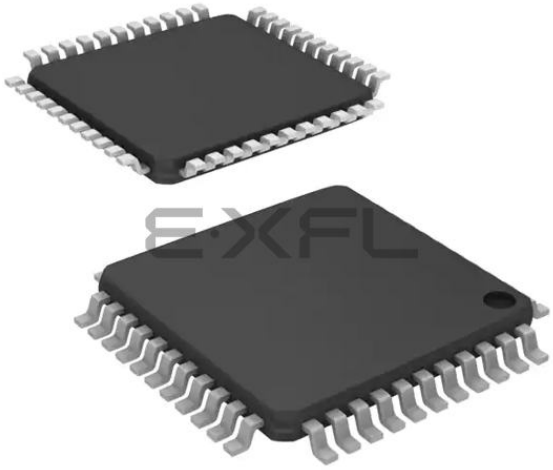


Welcome to E-XFL.COM

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	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	30
Program Memory Size	24KB (8K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 13x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f3014-30i-pt

3.1.1 DATA ACCESS FROM PROGRAM MEMORY USING TABLE INSTRUCTIONS

This architecture fetches 24-bit wide program memory. Consequently, instructions are always aligned. However, as the architecture is modified Harvard, data can also be present in program space.

There are two methods by which program space can be accessed: via special table instructions, or through the remapping of a 16K word program space page into the upper half of data space (see **Section 3.1.2 "Data Access from Program Memory Using Program Space Visibility"**). The TBLRDL and TBLWTL instructions offer a direct method of reading or writing the lsw of any address within program space, without going through data space. The TBLRDH and TBLWTH instructions are the only method whereby the upper 8 bits of a program space word can be accessed as data.

The PC is incremented by two for each successive 24-bit program word. This allows program memory addresses to directly map to data space addresses. Program memory can thus be regarded as two 16-bit word-wide address spaces, residing side by side, each with the same address range. TBLRDL and TBLWTL access the space which contains the least significant data word, and TBLRDH and TBLWTH access the space which contains the MS Data Byte.

Figure 3-3 shows how the EA is created for table operations and data space accesses (PSV = 1). Here, P<23:0> refers to a program space word, whereas D<15:0> refers to a data space word.

A set of table instructions are provided to move byte or word-sized data to and from program space. (See Figure 3-4 and Figure 3-5.)

1. TBLRDL: Table Read Low
   Word: Read the lsw of the program address; P<15:0> maps to D<15:0>.
   Byte: Read one of the LSBs of the program address;
   P<7:0> maps to the destination byte when byte select = 0;
   P<15:8> maps to the destination byte when byte select = 1.
2. TBLWTL: Table Write Low (refer to **Section 5.0 "Flash Program Memory"** for details on Flash programming)
3. TBLRDH: Table Read High
   Word: Read the most significant word (msw) of the program address; P<23:16> maps to D<7:0>; D<15:8> will always be = 0.
   Byte: Read one of the MSBs of the program address;
   P<23:16> maps to the destination byte when byte select = 0;
   The destination byte will always be = 0 when byte select = 1.
4. TBLWTH: Table Write High (refer to **Section 5.0 "Flash Program Memory"** for details on Flash Programming)

FIGURE 3-4: PROGRAM DATA TABLE ACCESS (LEAST SIGNIFICANT WORD)

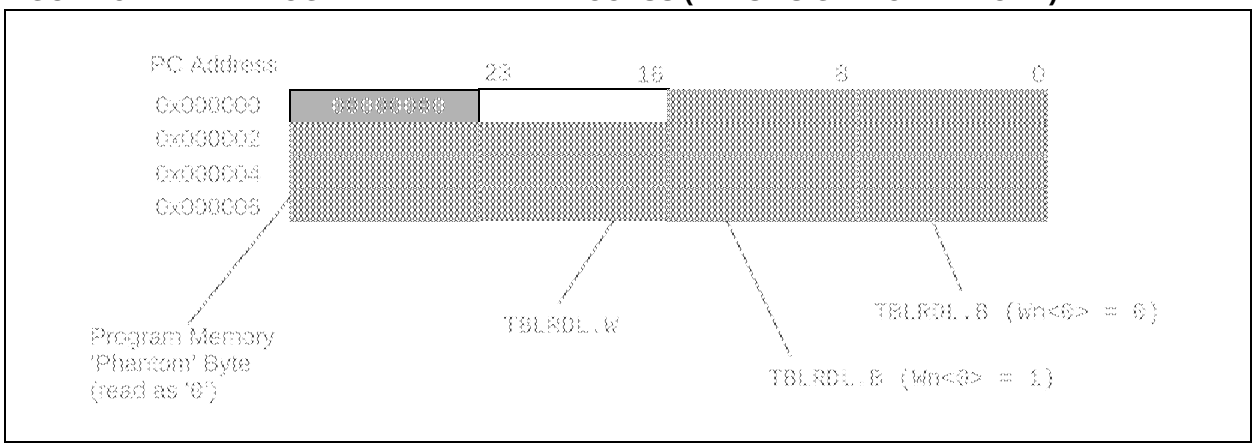
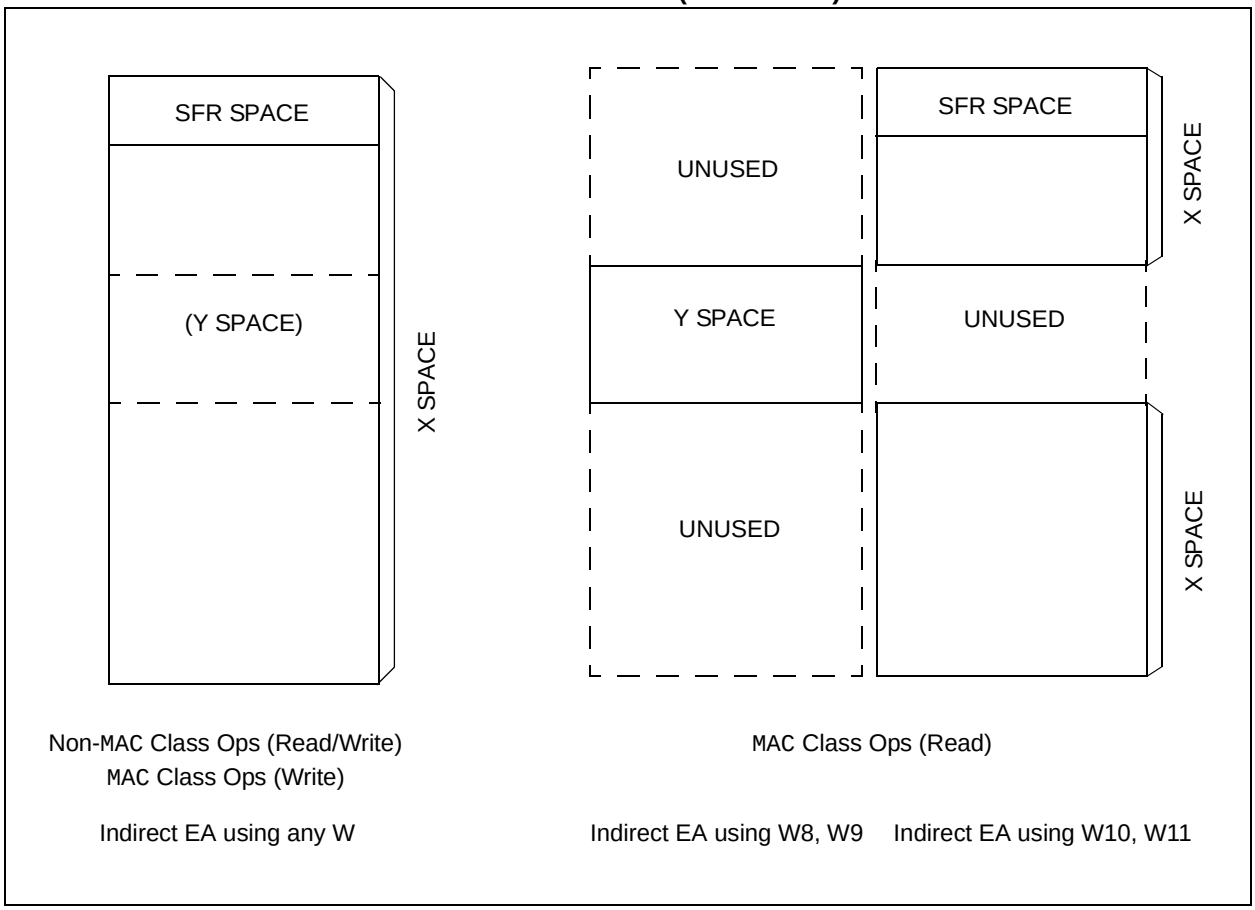


FIGURE 3-8: DATA SPACE FOR MCU AND DSP (MAC CLASS) INSTRUCTIONS EXAMPLE



All byte loads into any W register are loaded into the LSB. The MSB is not modified.

A Sign-Extend (SE) instruction is provided to allow users to translate 8-bit signed data to 16-bit signed values. Alternatively, for 16-bit unsigned data, users can clear the MSB of any W register by executing a Zero-Extend (ZE) instruction on the appropriate address.

Although most instructions are capable of operating on word or byte data sizes, it should be noted that some instructions, including the DSP instructions, operate only on words.

3.2.5 NEAR DATA SPACE

An 8-Kbyte 'near' data space is reserved in X address memory space between 0x0000 and 0x1FFF, which is directly addressable via a 13-bit absolute address field within all memory direct instructions. The remaining X address space and all of the Y address space is addressable indirectly. Additionally, the whole of X data space is addressable using MOV instructions, which support memory direct addressing with a 16-bit address field.

3.2.6 SOFTWARE STACK

The dsPIC DSC devices contain a software stack. W15 is used as the Stack Pointer.

The Stack Pointer always points to the first available free word and grows from lower addresses towards higher addresses. It pre-decrements for stack pops and post-increments for stack pushes as shown in Figure 3-10. Note that for a PC push during any CALL instruction, the MSB of the PC is zero-extended before the push, ensuring that the MSB is always clear.

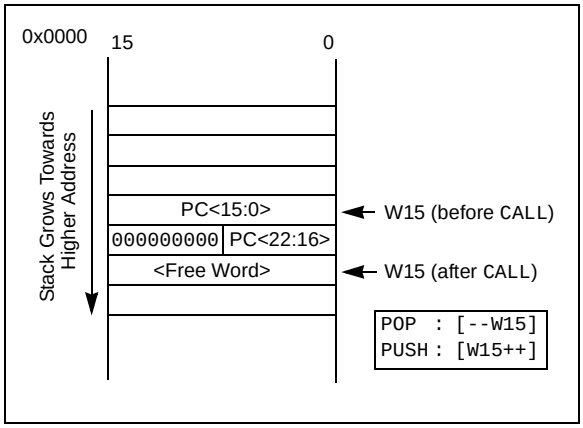
Note:	A PC push during exception processing concatenates the SRL register to the MSB of the PC prior to the push.

There is a Stack Pointer Limit register (SPLIM) associated with the Stack Pointer. SPLIM is uninitialized at Reset. As is the case for the Stack Pointer, SPLIM<0> is forced to '0' because all stack operations must be word-aligned. Whenever an Effective Address (EA) is generated, using W15 as a source or destination pointer, the address thus generated is compared with the value in SPLIM. If the contents of the Stack Pointer (W15) and the SPLIM register are equal and a push operation is performed, a stack error trap does not occur. The stack error trap occurs on a subsequent push operation. Thus, for example, if it is desirable to cause a stack error trap when the stack grows beyond address 0x2000 in RAM, initialize the SPLIM with the value, 0x1FFE.

Similarly, a Stack Pointer underflow (stack error) trap is generated when the Stack Pointer address is found to be less than 0x0800, thus preventing the stack from interfering with the Special Function Register (SFR) space.

A write to the SPLIM register should not be immediately followed by an indirect read operation using W15.

FIGURE 3-10: CALL STACK FRAME



4.2.3 MODULO ADDRESSING APPLICABILITY

Modulo Addressing can be applied to the Effective Address (EA) calculation associated with any W register. It is important to realize that the address boundaries check for addresses less than or greater than the upper (for incrementing buffers) and lower (for decrementing buffers) boundary addresses (not just equal to). Address changes may, therefore, jump beyond boundaries and still be adjusted correctly.

Note:	The modulo corrected effective address is written back to the register only when Pre-Modify or Post-Modify Addressing mode is used to compute the effective address. When an address offset (e.g., [W7+W2]) is used, Modulo Addressing correction is performed but the contents of the register remain unchanged.

4.3 Bit-Reversed Addressing

Bit-Reversed Addressing is intended to simplify data re-ordering for radix-2 FFT algorithms. It is supported by the X AGU for data writes only.

The modifier, which may be a constant value or register contents, is regarded as having its bit order reversed. The address source and destination are kept in normal order. Thus, the only operand requiring reversal is the modifier.

4.3.1 BIT-REVERSED ADDRESSING IMPLEMENTATION

Bit-Reversed Addressing is enabled when:

1. BWM (W register selection) in the MODCON register is any value other than ‘15’ (the stack cannot be accessed using Bit-Reversed Addressing) **and**
2. the BREN bit is set in the XBREV register **and**
3. the addressing mode used is Register Indirect with Pre-Increment or Post-Increment.

If the length of a bit-reversed buffer is $M = 2^N$ bytes, then the last ‘N’ bits of the data buffer start address must be zeros.

XB<14:0> is the bit-reversed address modifier or ‘pivot point’ which is typically a constant. In the case of an FFT computation, its value is equal to half of the FFT data buffer size.

Note:	All bit-reversed EA calculations assume word-sized data (LSb of every EA is always clear). The XB value is scaled accordingly to generate compatible (byte) addresses.

When enabled, Bit-Reversed Addressing is only executed for Register Indirect with Pre-Increment or Post-Increment Addressing and word-sized data writes. It does not function for any other addressing mode or for byte sized data. Normal addresses are generated instead. When Bit-Reversed Addressing is active, the W Address Pointer is always added to the address modifier (XB) and the offset associated with the Register Indirect Addressing mode is ignored. In addition, as word-sized data is a requirement, the LSb of the EA is ignored (and always clear).

Note:	Modulo Addressing and Bit-Reversed Addressing should not be enabled together. In the event that the user attempts to do this, Bit-Reversed Addressing assumes priority when active for the X WAGU, and X WAGU Modulo Addressing is disabled. However, Modulo Addressing continues to function in the X RAGU.

If Bit-Reversed Addressing has already been enabled by setting the BREN (XBREV<15>) bit, then a write to the XBREV register should not be immediately followed by an indirect read operation using the W register that has been designated as the Bit-Reversed Pointer.

FIGURE 4-2: BIT-REVERSED ADDRESS EXAMPLE

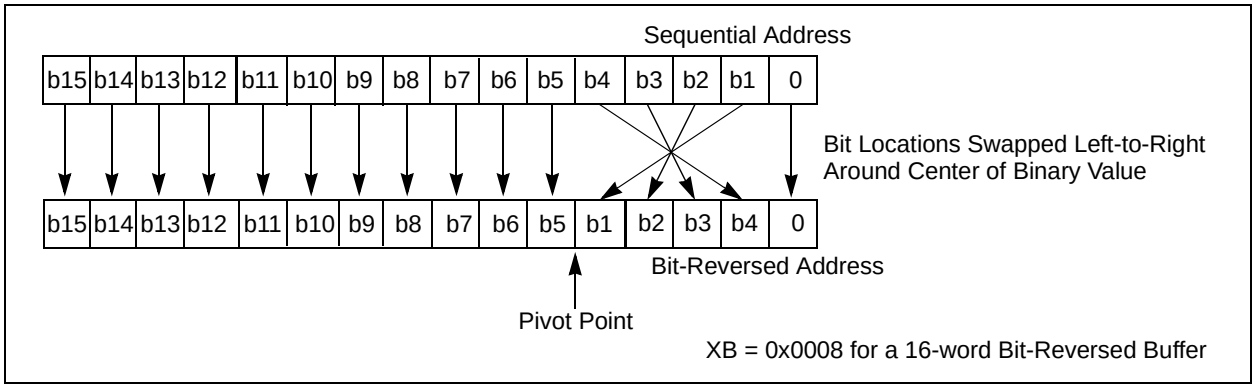


TABLE 4-2: BIT-REVERSED ADDRESS SEQUENCE (16-ENTRY)

Normal Address					Bit-Reversed Address				
A3	A2	A1	A0	Decimal	A3	A2	A1	A0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

TABLE 4-3: BIT-REVERSED ADDRESS MODIFIER VALUES FOR XBREV REGISTER

Buffer Size (Words)	XB<14:0> Bit-Reversed Address Modifier Value
1024	0x0200
512	0x0100
256	0x0080
128	0x0040
64	0x0020
32	0x0010
16	0x0008
8	0x0004
4	0x0002
2	0x0001

5.4 RTSP Operation

The dsPIC30F Flash program memory is organized into rows and panels. Each row consists of 32 instructions or 96 bytes. Each panel consists of 128 rows or 4K x 24 instructions. RTSP allows the user to erase one row (32 instructions) at a time and to program four instructions at one time. RTSP may be used to program multiple program memory panels, but the Table Pointer must be changed at each panel boundary.

Each panel of program memory contains write latches that hold 32 instructions of programming data. Prior to the actual programming operation, the write data must be loaded into the panel write latches. The data to be programmed into the panel is loaded in sequential order into the write latches; instruction 0, instruction 1, etc. The instruction words loaded must always be from a 32 address boundary.

The basic sequence for RTSP programming is to set up a Table Pointer, then do a series of TBLWT instructions to load the write latches. Programming is performed by setting the special bits in the NVMCON register. 32 TBLWTL and four TBLWTH instructions are required to load the 32 instructions. If multiple panel programming is required, the Table Pointer needs to be changed and the next set of multiple write latches written.

All of the table write operations are single-word writes (2 instruction cycles), because only the table latches are written. A programming cycle is required for programming each row.

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

5.5 Control Registers

The four SFRs used to read and write the program Flash memory are:

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

5.5.1 NVMCON REGISTER

The NVMCON register controls which blocks are to be erased, which memory type is to be programmed and the start of the programming cycle.

5.5.2 NVMADR REGISTER

The NVMADR register is used to hold the lower two bytes of the Effective Address. The NVMADR register captures the EA<15:0> of the last table instruction that has been executed and selects the row to write.

5.5.3 NVMADRU REGISTER

The NVMADRU register is used to hold the upper byte of the Effective Address. The NVMADRU register captures the EA<23:16> of the last table instruction that has been executed.

5.5.4 NVMKEY REGISTER

NVMKEY is a write-only register that is used for write protection. To start a programming or erase sequence, the user must consecutively write 0x55 and 0xAA to the NVMKEY register. Refer to **Section 5.6 “Programming Operations”** for further details.

Note:	The user can also directly write to the NVMADR and NVMADRU registers to specify a program memory address for erasing or programming.

6.0 DATA EEPROM MEMORY

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 data EEPROM memory is readable and writable during normal operation over the entire VDD range. The data EEPROM memory is directly mapped in the program memory address space.

The four SFRs used to read and write the program Flash memory are used to access data EEPROM memory as well. As described in **Section 5.5 "Control Registers"**, these registers are:

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

The EEPROM data memory allows read and write of single words and 16-word blocks. When interfacing to data memory, NVMADR, in conjunction with the NVMADRU register, are used to address the EEPROM location being accessed. TBLRDL and TBLWTL instructions are used to read and write data EEPROM. The dsPIC30F devices have up to 8 Kbytes (4K words) of data EEPROM with an address range from 0x7FF000 to 0x7FFFFE.

A word write operation should be preceded by an erase of the corresponding memory location(s). The write typically requires 2 ms to complete, but the write time varies with voltage and temperature.

A program or erase operation on the data EEPROM does not stop the instruction flow. The user is responsible for waiting for the appropriate duration of time before initiating another data EEPROM write/erase operation. Attempting to read the data EEPROM while a programming or erase operation is in progress results in unspecified data.

Control bit, WR, initiates write operations similar to program Flash writes. This bit cannot be cleared, only set, in software. They are cleared in hardware at the completion of the write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

The WREN bit, when set, allows a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a MCLR Reset or a WDT Time-out Reset during normal operation. In these situations, following Reset, the user can check the WRERR bit and rewrite the location. The address register, NVMADR, remains unchanged.

Note: Interrupt flag bit, NVMIF in the IFS0 register, is set when the write is complete. It must be cleared in software.

6.1 Reading the Data EEPROM

A TBLRD instruction reads a word at the current program word address. This example uses W0 as a pointer to data EEPROM. The result is placed in register W4 as shown in Example 6-1.

EXAMPLE 6-1: DATA EEPROM READ

```
MOV     #LOW_ADDR_WORD,W0  ; Init Pointer
MOV     #HIGH_ADDR_WORD,W1
MOV     W1,TBLPAG
TBLRDL  [ W0 ], W4         ; read data EEPROM
```

6.2 Erasing Data EEPROM

6.2.1 ERASING A BLOCK OF DATA EEPROM

In order to erase a block of data EEPROM, the NVMADRU and NVMADR registers must initially point to the block of memory to be erased. Configure NVMCON for erasing a block of data EEPROM and set the WR and WREN bits in the NVMCON register. Setting the WR bit initiates the erase, as shown in Example 6-2.

6.2.2 ERASING A WORD OF DATA EEPROM

The NVMADRU and NVMADR registers must point to the block. Select a block of data Flash and set the WR and WREN bits in the NVMCON register. Setting the WR bit initiates the erase, as shown in Example 6-3.

EXAMPLE 6-2: DATA EEPROM BLOCK ERASE

```
; Select data EEPROM block, WR, WREN bits
    MOV     #4045,W0
    MOV     W0,NVMCON                 ; Initialize NVMCON SFR

; Start erase cycle by setting WR after writing key sequence
    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                ; Initiate erase sequence
    NOP
    NOP
; Erase cycle will complete in 2mS. CPU is not stalled for the Data Erase Cycle
; User can poll WR bit, use NVMIF or Timer IRQ to determine erasure complete
```

EXAMPLE 6-3: DATA EEPROM WORD ERASE

```
; Select data EEPROM word, WR, WREN bits
    MOV     #4044,W0
    MOV     W0,NVMCON

; Start erase cycle by setting WR after writing key sequence
    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                  ; Initiate erase sequence
    NOP
    NOP
; Erase cycle will complete in 2mS. CPU is not stalled for the Data Erase Cycle
; User can poll WR bit, use NVMIF or Timer IRQ to determine erasure complete
```

TABLE 8-3: dsPIC30F3014 INTERRUPT CONTROLLER REGISTER MAP(1)

SFR Name	ADR	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
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF	0000 0000 0000 0000
IFS1	0086	—	—	—	—	C1IF	—	U2TXIF	U2RXIF	INT2IF	—	—	—	—	—	—	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	—	—	LVDIF	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	—	—	—	—	C1IE	—	U2TXIE	U2RXIE	INT2IE	—	—	—	—	—	—	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	—	—	LVDIE	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC3	009A	—	CNIP<2:0>			—	MI2CIP<2:0>			—	SI2CIP<2:0>			—	NVMIP<2:0>			0100 0100 0100 0100
IPC4	009C	—	—	—	—	—	—	—	—	—	—	—	—	—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100
IPC6	00A0	—	C1IP<2:0>			—	—	—	—	—	U2TXIP<2:0>			—	U2RXIP<2:0>			0100 0100 0100 0100
IPC7	00A2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100
IPC8	00A4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100
IPC9	00A6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0100 0100 0100
IPC10	00A8	—	—	—	—	—	LVDIP<2:0>			—	DCIIP<2:0>			—	—	—	—	0000 0100 0100 0000

Legend: — = unimplemented bit, read as '0'

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

NOTES:

14.7 Interrupts

The I²C module generates two interrupt flags, MI2CIF (I²C Master Interrupt Flag) and SI2CIF (I²C Slave Interrupt Flag). The MI2CIF interrupt flag is activated on completion of a master message event. The SI2CIF interrupt flag is activated on detection of a message directed to the slave.

14.8 Slope Control

The I²C standard requires slope control on the SDA and SCL signals for Fast mode (400 kHz). The control bit, DISSLW, enables the user to disable slew rate control if desired. It is necessary to disable the slew rate control for 1 MHz mode.

14.9 IPMI Support

The control bit, IPMIEN, enables the module to support Intelligent Peripheral Management Interface (IPMI). When this bit is set, the module accepts and acts upon all addresses.

14.10 General Call Address Support

The general call address can address all devices. When this address is used, all devices should, in theory, respond with an Acknowledgement.

The general call address is one of eight addresses reserved for specific purposes by the I²C protocol. It consists of all '0's with R_W = 0.

The general call address is recognized when the General Call Enable (GCEN) bit is set (I2CCON<7> = 1). Following a Start bit detection, 8 bits are shifted into I2CRSR and the address is compared with I2CADD, and is also compared with the general call address which is fixed in hardware.

If a general call address match occurs, the I2CRSR is transferred to the I2CRCV after the eighth clock, the RBF flag is set and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the Master Event Interrupt Flag (MI2CIF) is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the I2CRCV to determine if the address was device-specific or a general call address.

14.11 I²C Master Support

As a master device, six operations are supported:

- Assert a Start condition on SDA and SCL.
- Assert a Restart condition on SDA and SCL.
- Write to the I2CTRN register initiating transmission of data/address.
- Generate a Stop condition on SDA and SCL.
- Configure the I²C port to receive data.
- Generate an $\overline{ACK}$ condition at the end of a received byte of data.

14.12 I²C Master Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus is not released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this case, the data direction bit (R_W) is logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an $\overline{ACK}$ bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the data direction bit. In this case, the data direction bit (R_W) is logic '1'. Thus, the first byte transmitted is a 7-bit slave address, followed by a '1' to indicate the receive bit. Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an $\overline{ACK}$ bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

14.12.1 I²C MASTER TRANSMISSION

Transmission of a data byte, a 7-bit address or the second half of a 10-bit address, is accomplished by simply writing a value to I2CTRN register. The user should only write to I2CTRN when the module is in a Wait state. This action sets the Buffer Full Flag (TBF) and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data is shifted out onto the SDA pin after the falling edge of SCL is asserted. The Transmit Status Flag, TRSTAT (I2CSTAT<14>), indicates that a master transmit is in progress.

NOTES:

- Receive Error Interrupts:

  A receive error interrupt is 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 register, CiINTF.

  - Invalid Message Received:

    If any type of error occurred during reception of the last message, an error is 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 TX1IE was set.

If the message transmission fails, one of the error condition flags is set, and the TXREQ bit remains set, indicating that the message is still pending for transmission. If the message encountered an error condition during the transmission attempt, the TXERR bit is 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>) requests 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 is 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 detects the following transmission errors:

- Acknowledge error
- Form error
- Bit error

These transmission errors do not necessarily generate an interrupt but are indicated by the transmission error counter. However, each of these errors causes the transmission error counter to be incremented by one. Once the value of the error counter exceeds the value of 96, the ERRIF (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.

size and Frame Sync generator control bits. A new I^2S data transfer boundary is marked by a high-to-low or a low-to-high transition edge on the COFS pin.

18.3.6 SLAVE FRAME SYNC OPERATION

When the DCI module is operating as a Frame Sync slave (COFSD = 1), data transfers are controlled by the Codec device attached to the DCI module. The COFSM control bits control how the DCI module responds to incoming COFS signals.

In the Multichannel mode, a new data frame transfer begins one CSCK cycle after the COFS pin is sampled high (see Figure 18-2). The pulse on the COFS pin resets the Frame Sync generator logic.

In the I^2S mode, a new data word is transferred one CSCK cycle after a low-to-high or a high-to-low transition is sampled on the COFS pin. A rising or falling edge on the COFS pin resets the Frame Sync generator logic.

In the AC-Link mode, the tag slot and subsequent data slots for the next frame is transferred one CSCK cycle after the COFS pin is sampled high.

The COFSG and WS bits must be configured to provide the proper frame length when the module is operating in the Slave mode. Once a valid Frame Sync pulse has been sampled by the module on the COFS pin, an entire data frame transfer takes place. The module will not respond to further Frame Sync pulses until the data frame transfer has completed.

FIGURE 18-2: FRAME SYNC TIMING, MULTICHANNEL MODE

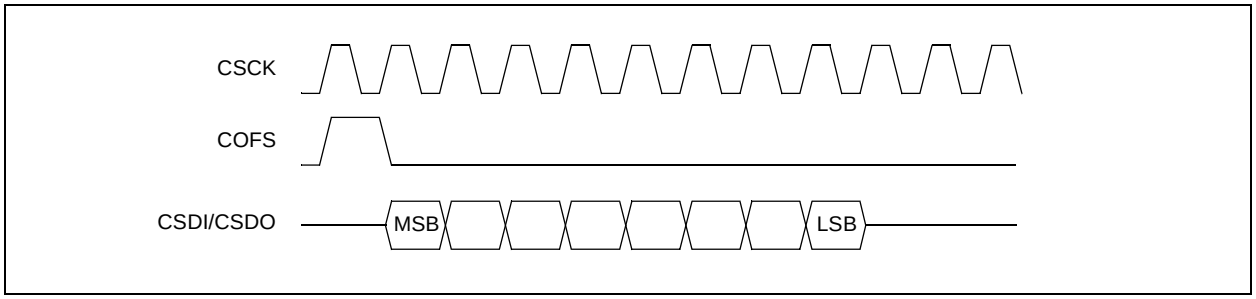


FIGURE 18-3: FRAME SYNC TIMING, AC-LINK START-OF-FRAME

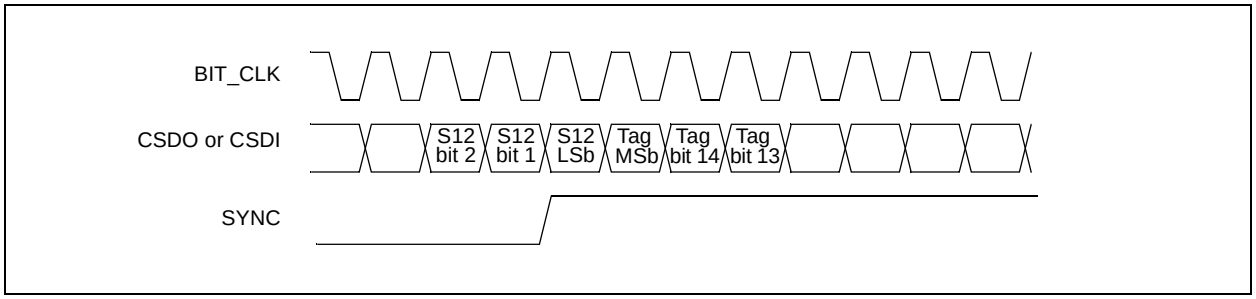
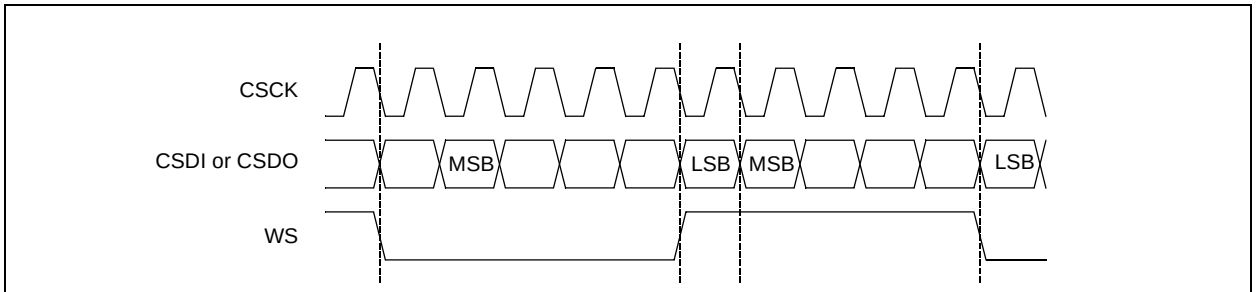
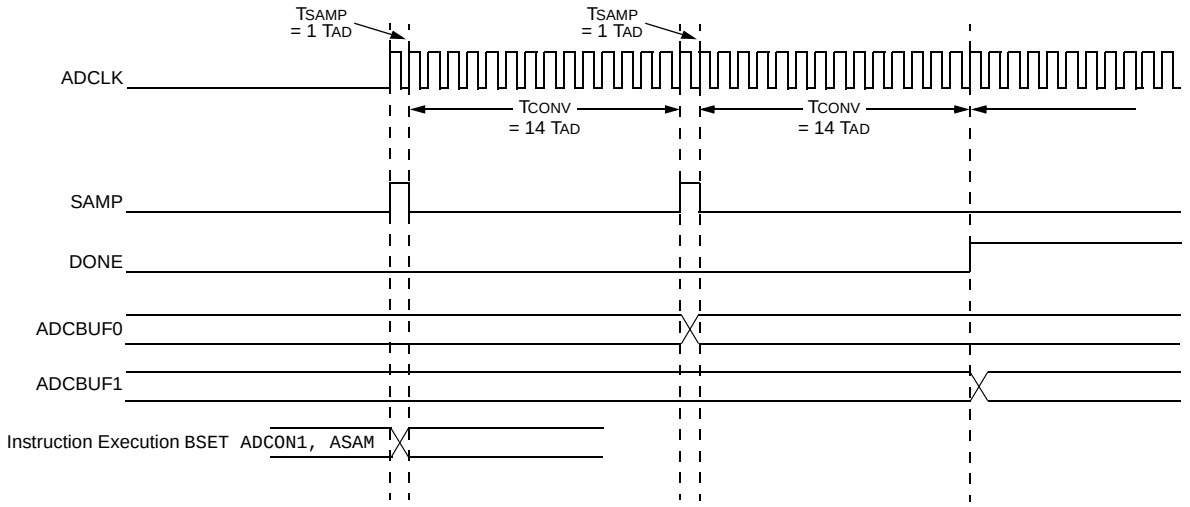


FIGURE 18-4: I^2S INTERFACE FRAME SYNC TIMING



Note: A 5-bit transfer is shown here for illustration purposes. The I^2S protocol does not specify word length – this will be system dependent.

FIGURE 19-3: CONVERTING 1 CHANNEL AT 200 ksps, AUTO-SAMPLE START, 1 TAD SAMPLING TIME

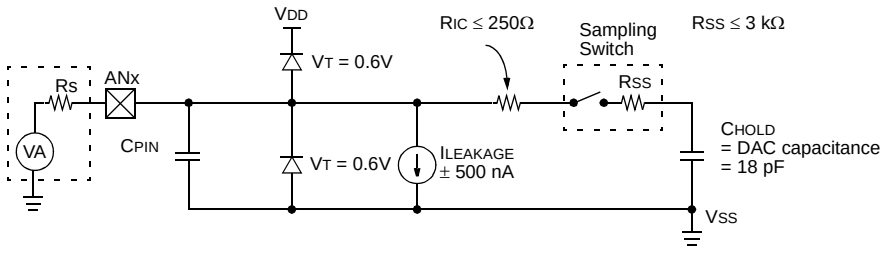


19.8 A/D Acquisition Requirements

The analog input model of the 12-bit A/D converter is shown in Figure 19-4. The total sampling time for the A/D is a function of the internal amplifier settling time and the holding capacitor charge time.

For the A/D converter to meet its specified accuracy, the Charge Holding Capacitor (CHOLD) must be allowed to fully charge to the voltage level on the analog input pin. The Source Impedance (RS), the Interconnect Impedance (RIC) and the Internal Sampling Switch (RSS) Impedance combine to directly affect the time required to charge the capacitor, CHOLD. The combined impedance of the analog sources must therefore be small enough to fully charge the holding capacitor within the chosen sample time. To minimize the effects of pin leakage currents on the accuracy of the A/D converter, the maximum recommended source impedance, RS, is 2.5 kΩ. After the analog input channel is selected (changed), this sampling function must be completed prior to starting the conversion. The internal holding capacitor will be in a discharged state prior to each sample operation.

FIGURE 19-4: 12-BIT A/D CONVERTER ANALOG INPUT MODEL



Legend:	CPIN	= Input Capacitance
	VT	= Threshold Voltage
	ILEAKAGE	= Leakage Current at the pin due to various junctions
	RIC	= Interconnect Resistance
	RSS	= Sampling Switch Resistance
	CHOLD	= Sample/Hold Capacitance (from DAC)

Note: CPIN value depends on device package and is not tested. Effect of CPIN negligible if Rs ≤ 2.5 kΩ.

Most single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles with the additional instruction cycle(s) executed as a NOP. Notable exceptions are the BRA (unconditional/computed branch), indirect CALL/GOTO, all table reads and writes, and RETURN/RETFIE instructions, which are single-word instructions but take two or three cycles. Certain instructions that involve skipping over the subsequent instruction require either two or three cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction. Moreover, double-word moves require two cycles. The double-word instructions execute in two instruction cycles.

Note: For more details on the instruction set, refer to the *"16-bit DSC and MCU Programmer's Reference Manual"* (DS70157).

TABLE 21-1: SYMBOLS USED IN OPCODE DESCRIPTIONS

Field	Description
#text	Means literal defined by "text"
(text)	Means "content of text"
[text]	Means "the location addressed by text"
{ }	Optional field or operation
<n:m>	Register bit field
.b	Byte mode selection
.d	Double-Word mode selection
.S	Shadow register select
.w	Word mode selection (default)
Acc	One of two accumulators {A, B}
AWB	Accumulator Write-Back Destination Address register ∈ {W13, [W13]+=2}
bit4	4-bit bit selection field (used in word addressed instructions) ∈ {0...15}
C, DC, N, OV, Z	MCU Status bits: Carry, Digit Carry, Negative, Overflow, Sticky Zero
Expr	Absolute address, label or expression (resolved by the linker)
f	File register address ∈ {0x0000...0x1FFF}
lit1	1-bit unsigned literal ∈ {0,1}
lit4	4-bit unsigned literal ∈ {0...15}
lit5	5-bit unsigned literal ∈ {0...31}
lit8	8-bit unsigned literal ∈ {0...255}
lit10	10-bit unsigned literal ∈ {0...255} for Byte mode, {0:1023} for Word mode
lit14	14-bit unsigned literal ∈ {0...16384}
lit16	16-bit unsigned literal ∈ {0...65535}
lit23	23-bit unsigned literal ∈ {0...8388608}; LSB must be 0
None	Field does not require an entry, may be blank
OA, OB, SA, SB	DSP Status bits: AccA Overflow, AccB Overflow, AccA Saturate, AccB Saturate
PC	Program Counter
Slit10	10-bit signed literal ∈ {-512...511}
Slit16	16-bit signed literal ∈ {-32768...32767}
Slit6	6-bit signed literal ∈ {-16...16}

TABLE 21-1: SYMBOLS USED IN OPCODE DESCRIPTIONS (CONTINUED)

Field	Description
Wb	Base W register ∈ {W0..W15}
Wd	Destination W register ∈ { Wd, [Wd], [Wd++], [Wd--], [++Wd], [--Wd] }
Wdo	Destination W register ∈ { Wnd, [Wnd], [Wnd++], [Wnd--], [++Wnd], [--Wnd], [Wnd+Wb] }
Wm,Wn	Dividend, Divisor Working register pair (direct addressing)
Wm*Wm	Multiplicand and Multiplier working register pair for Square instructions ∈ {W4*W4,W5*W5,W6*W6,W7*W7}
Wm*Wn	Multiplicand and Multiplier working register pair for DSP instructions ∈ {W4*W5,W4*W6,W4*W7,W5*W6,W5*W7,W6*W7}
Wn	One of 16 working registers ∈ {W0..W15}
Wnd	One of 16 destination working registers ∈ {W0..W15}
Wns	One of 16 source working registers ∈ {W0..W15}
WREG	W0 (working register used in file register instructions)
Ws	Source W register ∈ { Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws] }
Wso	Source W register ∈ { Wns, [Wns], [Wns++], [Wns--], [++Wns], [--Wns], [Wns+Wb] }
Wx	X data space prefetch address register for DSP instructions ∈ {[W8]+=6, [W8]+=4, [W8]+=2, [W8], [W8]-=6, [W8]-=4, [W8]-=2, [W9]+=6, [W9]+=4, [W9]+=2, [W9], [W9]-=6, [W9]-=4, [W9]-=2, [W9+W12],none}
Wxd	X data space prefetch destination register for DSP instructions ∈ {W4..W7}
Wy	Y data space prefetch address register for DSP instructions ∈ {[W10]+=6, [W10]+=4, [W10]+=2, [W10], [W10]-=6, [W10]-=4, [W10]-=2, [W11]+=6, [W11]+=4, [W11]+=2, [W11], [W11]-=6, [W11]-=4, [W11]-=2, [W11+W12], none}
Wyd	Y data space prefetch destination register for DSP instructions ∈ {W4..W7}

TABLE 21-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of Words	# of Cycles	Status Flags Affected
9	BTG	BTG	f,#bit4	Bit Toggle f	1	1	None
		BTG	Ws,#bit4	Bit Toggle Ws	1	1	None
10	BTSC	BTSC	f,#bit4	Bit Test f, Skip if Clear	1	1 (2 or 3)	None
		BTSC	Ws,#bit4	Bit Test Ws, Skip if Clear	1	1 (2 or 3)	None
11	BTSS	BTSS	f,#bit4	Bit Test f, Skip if Set	1	1 (2 or 3)	None
		BTSS	Ws,#bit4	Bit Test Ws, Skip if Set	1	1 (2 or 3)	None
12	BTST	BTST	f,#bit4	Bit Test f	1	1	Z
		BTST.C	Ws,#bit4	Bit Test Ws to C	1	1	C
		BTST.Z	Ws,#bit4	Bit Test Ws to Z	1	1	Z
		BTST.C	Ws,Wb	Bit Test Ws<Wb> to C	1	1	C
		BTST.Z	Ws,Wb	Bit Test Ws<Wb> to Z	1	1	Z
13	BTSTS	BTSTS	f,#bit4	Bit Test then Set f	1	1	Z
		BTSTS.C	Ws,#bit4	Bit Test Ws to C, then Set	1	1	C
		BTSTS.Z	Ws,#bit4	Bit Test Ws to Z, then Set	1	1	Z
14	CALL	CALL	lit23	Call Subroutine	2	2	None
		CALL	Wn	Call Indirect Subroutine	1	2	None
15	CLR	CLR	f	f = 0x0000	1	1	None
		CLR	WREG	WREG = 0x0000	1	1	None
		CLR	Ws	Ws = 0x0000	1	1	None
		CLR	Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Accumulator	1	1	OA,OB,SA,SB
16	CLRWDT	CLRWDT		Clear Watchdog Timer	1	1	WDTO, Sleep
17	COM	COM	f	$f = \overline{f}$	1	1	N,Z
		COM	f,WREG	$WREG = \overline{f}$	1	1	N,Z
		COM	Ws,Wd	$Wd = \overline{Ws}$	1	1	N,Z
18	CP	CP	f	Compare f with WREG	1	1	C,DC,N,OV,Z
		CP	Wb,#lit5	Compare Wb with lit5	1	1	C,DC,N,OV,Z
		CP	Wb,Ws	Compare Wb with Ws (Wb – Ws)	1	1	C,DC,N,OV,Z
19	CP0	CP0	f	Compare f with 0x0000	1	1	C,DC,N,OV,Z
		CP0	Ws	Compare Ws with 0x0000	1	1	C,DC,N,OV,Z
20	CPB	CPB	f	Compare f with WREG, with Borrow	1	1	C,DC,N,OV,Z
		CPB	Wb,#lit5	Compare Wb with lit5, with Borrow	1	1	C,DC,N,OV,Z
		CPB	Wb,Ws	Compare Wb with Ws, with Borrow ($Wb – Ws – \overline{C}$)	1	1	C,DC,N,OV,Z
21	CPSEQ	CPSEQ	Wb, Wn	Compare Wb with Wn, Skip if =	1	1 (2 or 3)	None
22	CPSGT	CPSGT	Wb, Wn	Compare Wb with Wn, Skip if >	1	1 (2 or 3)	None
23	CPSLT	CPSLT	Wb, Wn	Compare Wb with Wn, Skip if <	1	1 (2 or 3)	None
24	CPSNE	CPSNE	Wb, Wn	Compare Wb with Wn, Skip if ≠	1	1 (2 or 3)	None
25	DAW	DAW	Wn	Wn = Decimal Adjust Wn	1	1	C
26	DEC	DEC	f	f = f – 1	1	1	C,DC,N,OV,Z
		DEC	f,WREG	WREG = f – 1	1	1	C,DC,N,OV,Z
		DEC	Ws,Wd	Wd = Ws – 1	1	1	C,DC,N,OV,Z
27	DEC2	DEC2	f	f = f – 2	1	1	C,DC,N,OV,Z
		DEC2	f,WREG	WREG = f – 2	1	1	C,DC,N,OV,Z
		DEC2	Ws,Wd	Wd = Ws – 2	1	1	C,DC,N,OV,Z
28	DISI	DISI	#lit14	Disable Interrupts for k Instruction Cycles	1	1	None

TABLE 23-32: SPI MODULE SLAVE 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**
SP70	TscL	SCKx Input Low Time	30	—	—	ns	
SP71	TscH	SCKx Input High Time	30	—	—	ns	
SP72	TscF	SCKx Input Fall Time(3)	—	—	25	ns	
SP73	TscR	SCKx Input Rise Time(3)	—	—	25	ns	
SP30	TdoF	SDOx Data Output Fall Time(3)	—	—	—	ns	See parameter DO32
SP31	TdoR	SDOx Data Output Rise Time(3)	—	—	—	ns	See parameter DO31
SP35	TscH2do, 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	
SP50	TssL2scH, TssL2scL	$\overline{SSx}$↓ to SCKx↑ or SCKx↓ Input	120	—	—	ns	
SP51	TssH2doZ	$\overline{SSx}$↑ to SDOx Output High-impedance(3)	10	—	50	ns	
SP52	TscH2ssH TscL2ssH	$\overline{SSx}$ after SCKx Edge	1.5 TCY + 40	—	—	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: Assumes 50 pF load on all SPI pins.