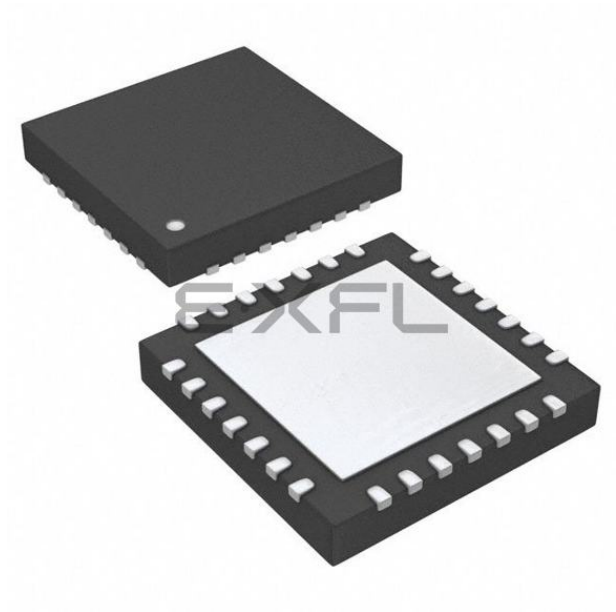
Microchip Technology - PIC18LF26J53-I/ML Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

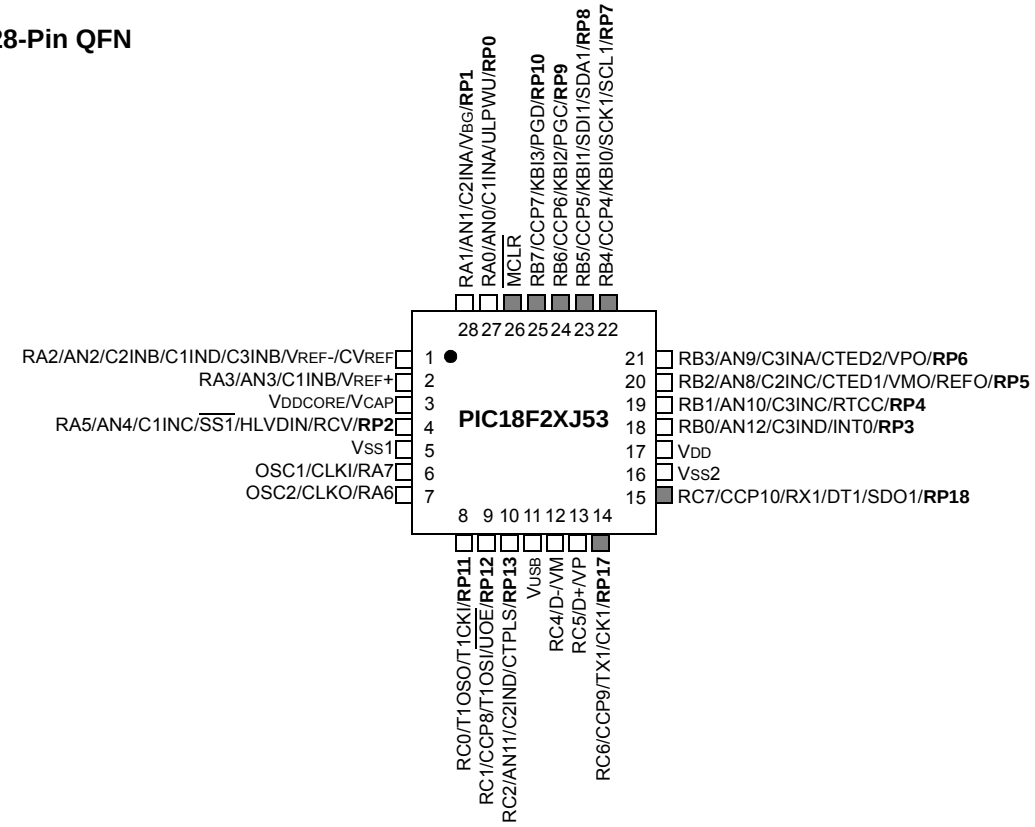
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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"

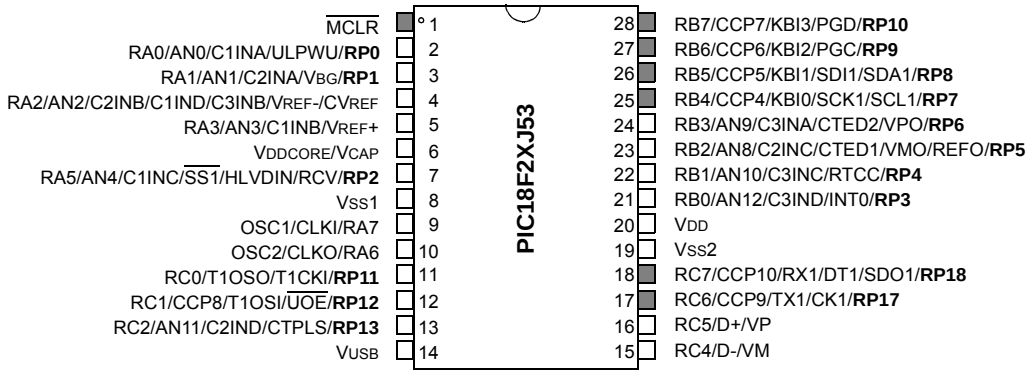
Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	48MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	22
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 2.75V
Data Converters	A/D 10x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26j53-i-ml

Pin Diagrams

28-Pin QFN



28-Pin SPDIP/SOIC/SSOP



Legend: Shaded pins are 5.5V tolerant.
RPn represents remappable pins. Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see **Section 10.7 "Peripheral Pin Select (PPS)"**.

Note: For the QFN package, it is recommended that the bottom pad be connected to VSS.

TABLE 3-5: OSCILLATOR CONFIGURATION OPTIONS FOR USB OPERATION

Input Oscillator Frequency	PLL Division (PLLDIV<2:0>)	Clock Mode (FOSC<2:0>)	MCU Clock Division (CPDIV<1:0>)	Microcontroller Clock Frequency
48 MHz	N/A	EC	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
48 MHz	÷12 (000)	ECPLL	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
40 MHz	÷10 (001)	ECPLL	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
24 MHz	÷6 (010)	ECPLL	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
24 MHz	N/A	EC(1)	None (11)	24 MHz
			÷2 (10)	12 MHz
			÷3 (01)	8 MHz
			÷6 (00)	4 MHz
20 MHz	÷5 (011)	ECPLL	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
16 MHz	÷4 (100)	HSPLL, ECPLL	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
12 MHz	÷3 (101)	HSPLL, ECPLL	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
8 MHz	÷2 (110)	HSPLL, ECPLL, INTOSCPLL/ INTOSCPLLO	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz
4 MHz	÷1 (111)	HSPLL, ECPLL	None (11)	48 MHz
			÷2 (10)	24 MHz
			÷3 (01)	16 MHz
			÷6 (00)	8 MHz

Note 1: The 24 MHz EC mode (without PLL) is only compatible with low-speed USB. Full-speed USB requires a 48 MHz system clock.

6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as 2 bytes or 4 bytes in program memory. The Least Significant Byte (LSB) 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 6.1.3 "Program Counter"**).

Figure 6-5 provides 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 6-5 displays 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 29.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY

			LSB = 1	LSB = 0	Word Address ↓
	Program Memory Byte Locations →				000000h
					000002h
					000004h
					000006h
Instruction 1:	MOVLW	055h	0Fh	55h	000008h
Instruction 2:	GOTO	0006h	EFh	03h	00000Ah
			F0h	00h	00000Ch
Instruction 3:	MOVFF	123h, 456h	C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

6.2.4 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 (MSbs); 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 6-4 illustrates how this works.

Note:	See **Section 6.5 "Program Memory and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

EXAMPLE 6-4: 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

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J53 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F53h	CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000
F52h	CCPTMRS0	C3TSEL1	C3TSEL0	C2TSEL2	C2TSEL1	C2TSEL0	C1TSEL2	C1TSEL1	C1TSEL0	0000 0000
F51h	CCPTMRS1	C7TSEL1	C7TSEL0	—	C6TSEL0	—	C5TSEL0	C4TSEL1	C4TSEL0	00-0 -000
F50h	CCPTMRS2	—	—	—	C10TSEL0(3)	—	C9TSEL0(3)	C8TSEL1	C8TSEL0	---0 -000
F4Fh	DSGPR1	Deep Sleep Persistent General Purpose Register (contents retained even in deep sleep)								xxxx xxxx
F4Eh	DSGPR0	Deep Sleep Persistent General Purpose Register (contents retained even in deep sleep)								xxxx xxxx
F4Dh	DSCONH	DSEN	—	—	—	—	r	DSULPEN	RTCWDIS	0--- -000
F4Ch	DSCONL	—	—	—	—	—	ULPWDIS	DSBOR	RELEASE	---- -000
F4Bh	DSWAKEH	—	—	—	—	—	—	—	DSINT0	---- ---0
F4Ah	DSWAKEL	DSFLT	—	DSULP	DSWDT	DSRTC	DSMCLR	—	DSPOR	0-00 00-1
F49h	ANCON1	VBGEN	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	0--0 0000
F48h	ANCON0	PCFG7(1)	PCFG6(1)	PCFG5(1)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000
F47h	OEDCON	ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0	0000 0000
F46h	ALRMRPT	ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0	0000 0000
F45h	ALRMVALH	Alarm Value High Register Window based on ALRMPTR<1:0>								xxxx xxxx
F44h	ALRMVALL	Alarm Value Low Register Window based on ALRMPTR<1:0>								xxxx xxxx
F43h	—	—	—	—	—	—	—	—	—	---- ----
F42h	ODCON1	CCP8OD	CCP7OD	CCP6OD	CCP5OD	CCP4OD	ECCP3OD	ECCP2OD	ECCP1OD	0000 0000
F41h	ODCON2	—	—	—	—	CCP10OD	CCP9OD	U2OD	U1OD	---- 0000
F40h	ODCON3	—	—	—	—	—	—	SPI2OD	SPI1OD	---- --00
F3Fh	RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0	0-00 0000
F3Eh	RTCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	0000 0000
F3Dh	REFOCON	ROON	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0	0-00 0000
F3Ch	PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	PMPTTL(1)	---- -000
F3Bh	RTCVALH	RTCC Value High Register Window Based on RTCPTR<1:0>								0xxx xxxx
F3Ah	RTCVALL	RTCC Value Low Register Window Based on RTCPTR<1:0>								0xxx xxxx
F39h	UCFG	UTEYE	UOEMON	—	UPUEN	UTRDIS	FSEN	PPB1	PPB0	00-0 0000
F38h	UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	-000 0000
F37h	UEIE	BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE	0--0 0000
F36h	UIE	—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE	-000 0000
F35h	UEP15	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F34h	UEP14	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F33h	UEP13	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F32h	UEP12	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F31h	UEP11	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F30h	UEP10	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F2Fh	UEP9	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F2Eh	UEP8	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F2Dh	UEP7	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F2Ch	UEP6	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F2Bh	UEP5	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F2Ah	UEP4	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F29h	UEP3	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F28h	UEP2	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F27h	UEP1	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F26h	UEP0	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000
F25h	CM3CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0	0001 1111
F24h	TMR5H	Timer5 Register High Byte								xxxx xxxx

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved, do not modify

Note 1: Implemented only for 44-pin devices (PIC18F46J53, PIC18F47J53, PIC18LF46J53 and PIC18LF47J53).

2: Implemented only for 28-pin devices (PIC18F26J53, PIC18F27J53, PIC18LF26J53 and PIC18LF27J53).

3: Implemented only for devices with 128 Kbyte of program memory (PIC18F27J53, PIC18F47J53, PIC18LF27J53 and PIC18LF47J53).

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J53 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F23h	TMR5L	Timer5 Register Low Bytes								xxxx xxxx
F22h	T5CON	TMR5CS1	TMR5CS0	T5CKPS1	T5CKPS0	T5OSCEN	$\overline{\text{T5SYNC}}$	RD16	TMR5ON	0000 0000
F21h	T5GCON	TMR5GE	T5GPOL	T5GTM	T5GSPM	$\overline{\text{T5GGO}}$/$\overline{\text{T5DONE}}$	T5GVAL	T5GSS1	T5GSS0	0000 0x00
F20h	TMR6	Timer6 Register								0000 0000
F1Fh	PR6	Timer6 Period Register								1111 1111
F1Eh	T6CON	—	T6OUTPS3	T6OUTPS2	T6OUTPS1	T6OUTPS0	TMR6ON	T6CKPS1	T6CKPS0	-000 0000
F1Dh	TMR8	Timer8 Register								0000 0000
F1Ch	PR8	Timer8 Period Register								1111 1111
F1Bh	T8CON	—	T8OUTPS3	T8OUTPS2	T8OUTPS1	T8OUTPS0	TMR8ON	T8CKPS1	T8CKPS0	-000 0000
F1Ah	PSTR3CON	CMPL1	CMPL0	—	STRSYNC	STRD	STRC	STRB	STRA	00-0 0001
F19h	ECCP3AS	ECCP3ASE	ECCP3AS2	ECCP3AS1	ECCP3AS0	PSS3AC1	PSS3AC0	PSS3BD1	PSS3BD0	0000 0000
F18h	ECCP3DEL	P3RSEN	P3DC6	P3DC5	P3DC4	P3DC3	P3DC2	P3DC1	P3DC0	0000 0000
F17h	CCPR3H	Capture/Compare/PWM Register 3 High Byte								xxxx xxxx
F16h	CCPR3L	Capture/Compare/PWM Register 3 Low Byte								xxxx xxxx
F15h	CCP3CON	P3M1	P3M0	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	0000 0000
F14h	CCPR4H	Capture/Compare/PWM Register 4 High Byte								xxxx xxxx
F13h	CCPR4L	Capture/Compare/PWM Register 4 Low Byte								xxxx xxxx
F12h	CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0	--00 0000
F11h	CCPR5H	Capture/Compare/PWM Register 5 High Byte								xxxx xxxx
F10h	CCPR5L	Capture/Compare/PWM Register 5 Low Byte								xxxx xxxx
F0Fh	CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0	--00 0000
F0Eh	CCPR6H	Capture/Compare/PWM Register 6 High Byte								xxxx xxxx
F0Dh	CCPR6L	Capture/Compare/PWM Register 6 Low Byte								xxxx xxxx
F0Ch	CCP6CON	—	—	DC6B1	DC6B0	CCP6M3	CCP6M2	CCP6M1	CCP6M0	--00 0000
F0Bh	CCPR7H	Capture/Compare/PWM Register 7 High Byte								xxxx xxxx
F0Ah	CCPR7L	Capture/Compare/PWM Register 7 Low Byte								xxxx xxxx
F09h	CCP7CON	—	—	DC7B1	DC7B0	CCP7M3	CCP7M2	CCP7M1	CCP7M0	--00 0000
F08h	CCPR8H	Capture/Compare/PWM Register 8 High Byte								xxxx xxxx
F07h	CCPR8L	Capture/Compare/PWM Register 8 Low Byte								xxxx xxxx
F06h	CCP8CON	—	—	DC8B1	DC8B0	CCP8M3	CCP8M2	CCP8M1	CCP8M0	--00 0000
F05h	CCPR9H	Capture/Compare/PWM Register 9 High Byte								xxxx xxxx
F04h	CCPR9L	Capture/Compare/PWM Register 9 Low Byte								xxxx xxxx
F03h	CCP9CON	—	—	DC9B1	DC9B0	CCP9M3	CCP9M2	CCP9M1	CCP9M0	--00 0000
F02h	CCPR10H	Capture/Compare/PWM Register 10 High Byte								xxxx xxxx
F01h	CCPR10L	Capture/Compare/PWM Register 10 Low Byte								xxxx xxxx
F00h	CCP10CON	—	—	DC10B1	DC10B0	CCP10M3	CCP10M2	CCP10M1	CCP10M0	--00 0000
EFFh	RPINR24	—	—	—	PWM Fault Input (FLT0) to Input Pin Mapping bits					---1 1111
EFEh	RPINR23	—	—	—	SPI2 Slave Select Input (SS2) to Input Pin Mapping bits					---1 1111
EFDh	RPINR22	—	—	—	SPI2 Clock Input (SCK2) to Input Pin Mapping bits					---1 1111
EFCh	RPINR21	—	—	—	SPI2 Data Input (SDI2) to Input Pin Mapping bits					---1 1111
EFBh	—	—	—	—	—	—	—	—	—	
EFAh	—	—	—	—	—	—	—	—	—	
EF9h	—	—	—	—	—	—	—	—	—	
EF8h	RPINR17	—	—	—	EUSART2 Clock Input (CK2) to Input Pin Mapping bits					---1 1111
EF7h	RPINR16	—	—	—	EUSART2 RX2DT2 to Input Pin Mapping bits					---1 1111
EF6h	—	—	—	—	—	—	—	—	—	
EF5h	—	—	—	—	—	—	—	—	—	

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved, do not modify

Note 1: Implemented only for 44-pin devices (PIC18F46J53, PIC18F47J53, PIC18LF46J53 and PIC18LF47J53).

2: Implemented only for 28-pin devices (PIC18F26J53, PIC18F27J53, PIC18LF26J53 and PIC18LF27J53).

3: Implemented only for devices with 128 Kbyte of program memory (PIC18F27J53, PIC18F47J53, PIC18LF27J53 and PIC18LF47J53).

6.6 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different. This is due to the introduction of a new addressing mode for the data memory space. This mode also alters the behavior of Indirect Addressing using FSR2 and its associated operands.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect Addressing with FSR0 and FSR1 also remains unchanged.

6.6.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair and its associated file operands. Under proper conditions, instructions that use the Access Bank, that is, most bit and byte-oriented instructions, can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced ('a' = 0); and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in Direct Addressing) or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

6.6.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte and bit-oriented instructions are not affected if they use the Access Bank (Access RAM bit is '1') or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is provided in Figure 6-9.

Those who desire to use byte or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 29.2.1 "Extended Instruction Syntax"**.

7.5.3 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.

7.5.4 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. If the write operation is interrupted by a MCLR Reset, or a WDT time-out Reset during normal operation, the user can check the WRERR bit and rewrite the location(s) as needed.

7.6 Flash Program Operation During Code Protection

See **Section 28.6 "Program Verification and Code Protection"** for details on code protection of Flash program memory.

TABLE 7-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)				
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)							
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)							
TABLAT	Program Memory Table Latch							
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
EECON2	Program Memory Control Register 2 (not a physical register)							
EECON1	—	—	WPROG	FREE	WRERR	WREN	WR	—

Legend: — = unimplemented, read as '0'. Shaded cells are not used during Flash program memory access.

REGISTER 9-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3 (ACCESS FA4h)

R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
bit 7							bit 0

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

bit 7 **SSP2IF:** Master Synchronous Serial Port 2 Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 6 **BCL2IF:** Bus Collision Interrupt Flag bit (MSSP2 module)
1 = A bus collision occurred (must be cleared in software)
0 = No bus collision occurred

bit 5 **RC2IF:** EUSART2 Receive Interrupt Flag bit
1 = The EUSART2 receive buffer, RCREG2, is full (cleared when RCREG2 is read)
0 = The EUSART2 receive buffer is empty

bit 4 **TX2IF:** EUSART2 Transmit Interrupt Flag bit
1 = The EUSART2 transmit buffer, TXREG2, is empty (cleared when TXREG2 is written)
0 = The EUSART2 transmit buffer is full

bit 3 **TMR4IF:** TMR4 to PR4 Match Interrupt Flag bit
1 = A TMR4 to PR4 match occurred (must be cleared in software)
0 = No TMR4 to PR4 match occurred

bit 2 **CTMUIF:** Charge Time Measurement Unit Interrupt Flag bit
1 = A CTMU event has occurred (must be cleared in software)
0 = A CTMU event has not occurred

bit 1 **TMR3GIF:** Timer3 Gate Event Interrupt Flag bit
1 = A Timer3 gate event completed (must be cleared in software)
0 = No Timer3 gate event completed

bit 0 **RTCCIF:** RTCC Interrupt Flag bit
1 = An RTCC interrupt occurred (must be cleared in software)
0 = No RTCC interrupt occurred

REGISTER 9-15: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2 (ACCESS FA2h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	CM2IP	CM1IP	USBIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP
bit 7							bit 0

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

bit 7 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **CM2IP:** Comparator 2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **C12IP:** Comparator 1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **USBIP:** USB Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **BCL1IP:** Bus Collision Interrupt Priority bit (MSSP1 module)
1 = High priority
0 = Low priority

bit 2 **HLVDIP:** High/Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP:** ECCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

16.0 TIMER4/6/8 MODULE

The Timer4/6/8 timer modules have the following features:

- Eight-bit Timer register (TMRx)
- Eight-bit Period register (PRx)
- Readable and writable (all registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMRx match of PRx

Note: Throughout this section, generic references are used for register and bit names that are the same – except for an 'x' variable that indicates the item's association with the Timer4, Timer6 or Timer8 module. For example, the control register is named TxCON and refers to T4CON, T6CON and T8CON.

The Timer4/6/8 modules have a control register shown in Register 16-1. Timer4/6/8 can be shut off by clearing control bit, TMRxON (TxCON<2>), to minimize power consumption. The prescaler and postscaler selection of Timer4/6/8 are also controlled by this register. Figure 16-1 is a simplified block diagram of the Timer4/6/8 modules.

16.1 Timer4/6/8 Operation

Timer4/6/8 can be used as the PWM time base for the PWM mode of the ECCP modules. The TMRx registers are readable and writable, and are cleared on any device Reset. The input clock (FOSC/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits, TxCKPS<1:0> (TxCON<1:0>). The match output of TMRx goes through a four-bit postscaler (that gives a 1:1 to 1:16 inclusive scaling) to generate a TMRx interrupt, latched in the flag bit, TMRxIF. Table 16-1 gives each module's flag bit.

TABLE 16-1: TIMER4/6/8 FLAG BITS

Timer Module	Flag Bit
4	PIR3<3>
6	PIR5<3>
8	PIR5<4>

The interrupt can be enabled or disabled by setting or clearing the Timerx Interrupt Enable bit (TMRxIE), shown in Table 16-2.

TABLE 16-2: TIMER4/6/8 INTERRUPT ENABLE BITS

Timer Module	Flag Bit
4	PIE3<3>
6	PIE5<3>
8	PIE5<4>

The prescaler and postscaler counters are cleared when any of the following occurs:

- A write to the TMRx register
- A write to the TxCON register
- Any device Reset (Power-on Reset (POR), $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset (WDTR) or Brown-out Reset (BOR))

A TMRx is not cleared when a TxCON is written.

Note: The CCP and ECCP modules use Timers 1 through 8 for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRSx registers. For more details, see Register 19-2, Register 18-2 and Register 18-3.

17.1.2 RTCVALH AND RTCVALL REGISTER MAPPINGS

REGISTER 17-6: RESERVED REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

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

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

REGISTER 17-7: YEAR: YEAR VALUE REGISTER(1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
YRTEN3	YRTEN2	YRTEN1	YRTEN0	YRONE3	YRONE2	YRONE1	YRONE0
bit 7							bit 0

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

bit 7-4 **YRTEN<3:0>:** Binary Coded Decimal Value of Year's Tens Digit bits
Contains a value from 0 to 9.

bit 3-0 **YRONE<3:0>:** Binary Coded Decimal Value of Year's Ones Digit bits
Contains a value from 0 to 9.

Note 1: A write to the YEAR register is only allowed when RTCWREN = 1.

REGISTER 17-8: MONTH: MONTH VALUE REGISTER(1)

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	MTHTEN0	MTHONE3	MTHONE2	MTHONE1	MTHONE0
bit 7							bit 0

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

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

bit 4 **MTHTEN0:** Binary Coded Decimal Value of Month's Tens Digit bit
Contains a value of 0 or 1.

bit 3-0 **MTHONE<3:0>:** Binary Coded Decimal Value of Month's Ones Digit bits
Contains a value from 0 to 9.

Note 1: A write to this register is only allowed when RTCWREN = 1.

19.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

PIC18F47J53 family devices have three Enhanced Capture/Compare/PWM (ECCP) modules: ECCP1, ECCP2 and ECCP3. These modules contain a 16-bit register, which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register. These ECCP modules are upwardly compatible with CCP.

Note:	Throughout this section, generic references are used for register and bit names that are the same – except for an 'x' variable that indicates the item's association with the ECCP1, ECCP2 or ECCP3 module. For example, the control register is named CCPxCON and refers to CCP1CON, CCP2CON and CCP3CON.

The ECCP modules are implemented as standard CCP modules with enhanced PWM capabilities. These include:

- Provision for two or four output channels
- Output Steering modes
- Programmable polarity
- Programmable dead-band control
- Automatic shutdown and restart

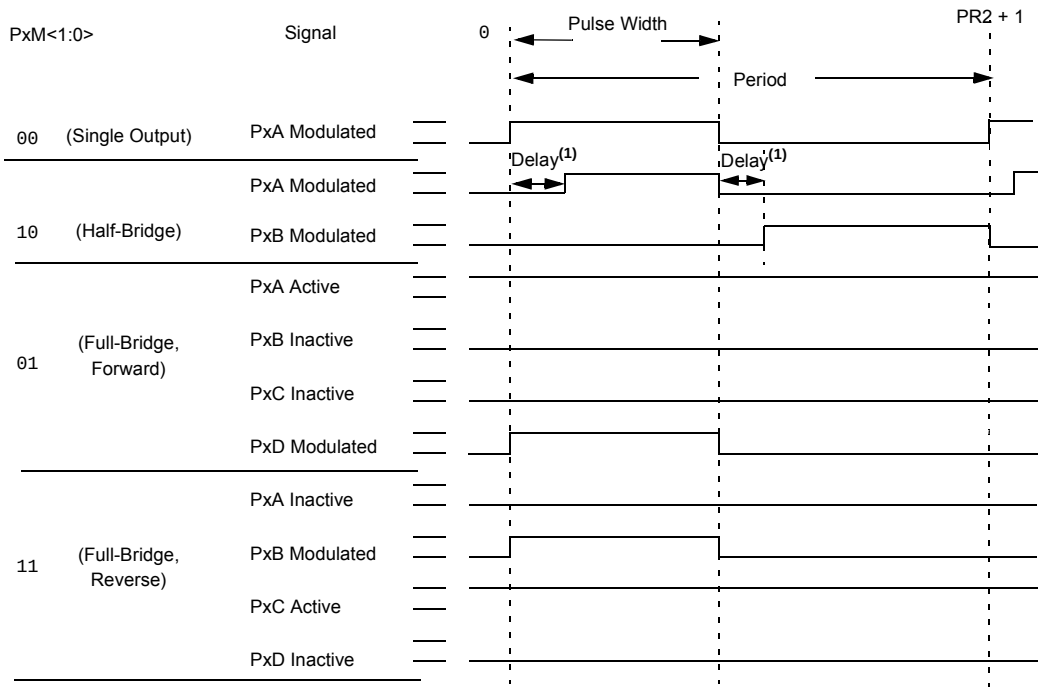
The enhanced features are discussed in detail in **Section 19.4 "PWM (Enhanced Mode)"**.

TABLE 19-3: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES
TABLE 19-3:

ECCP Mode	PxM<1:0>	PxA	PxB	PxC	PxD
Single	00	Yes(1)	Yes(1)	Yes(1)	Yes(1)
Half-Bridge	10	Yes	Yes	No	No
Full-Bridge, Forward	01	Yes	Yes	Yes	Yes
Full-Bridge, Reverse	11	Yes	Yes	Yes	Yes

Note 1: Outputs are enabled by pulse steering in Single mode (see Register 19-6).

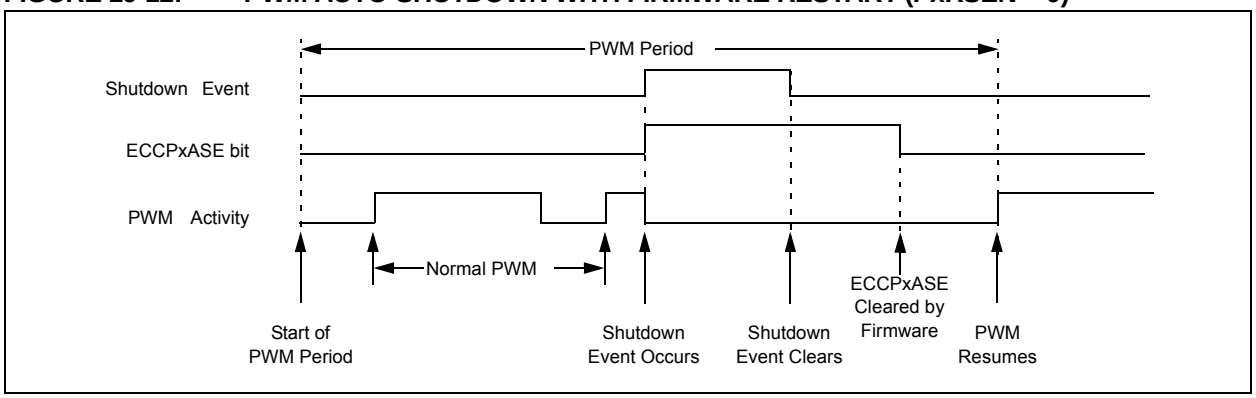
FIGURE 19-4: PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE) EXAMPLE



Relationships:
- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Pulse Width = TOSC * (CCPRxL<7:0>:CCPxCON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * (ECCPxDEL<6:0>)

Note 1: Dead-band delay is programmed using the ECCPxDEL register (see **Section 19.4.6 "Programmable Dead-Band Delay Mode"**).

FIGURE 19-12: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (PxRSEN = 0)



19.4.5 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the PxRSEN bit (ECCPxDEL<7>).

If auto-restart is enabled, the ECCPxASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the ECCPxASE bit will be cleared via hardware and normal operation will resume.

The module will wait until the next PWM period begins, however, before re-enabling the output pin. This behavior allows the auto-shutdown with auto-restart features to be used in applications based on the current mode of PWM control.

FIGURE 19-13: PWM AUTO-SHUTDOWN WITH AUTO-RESTART ENABLED (PxRSEN = 1)

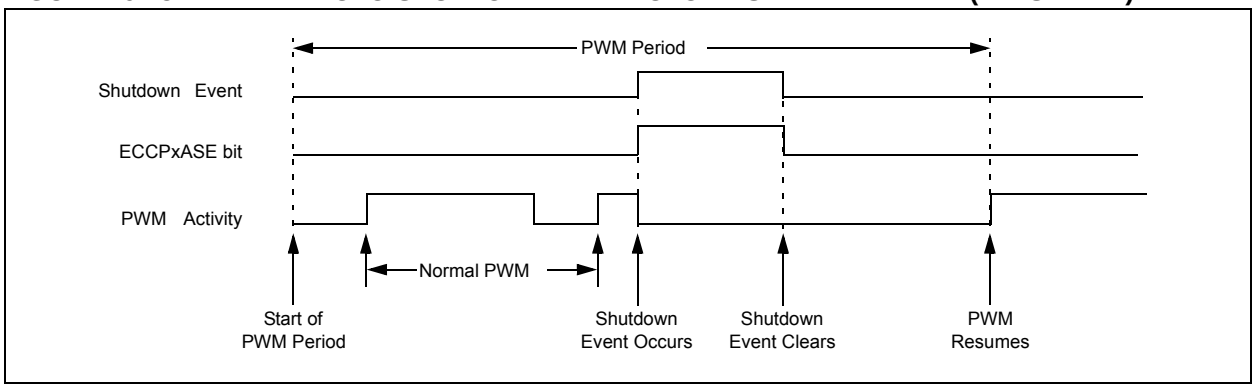


TABLE 20-2: REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(2)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(2)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(2)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
SSP1BUF	MSSP1 Receive Buffer/Transmit Register							
SSPxCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
SSPxSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
SSP2BUF	MSSP2 Receive Buffer/Transmit Register							
ODCON3(1)	—	—	—	—	—	—	SPI2OD	SPI1OD

Legend: Shaded cells are not used by the MSSPx module in SPI mode.

Note 1: Configuration SFR overlaps with default SFR at this address; available only when WDTCON<4> = 1.

2: These bits are only available on 44-pin devices.

20.5.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDAx or SCLx is sampled low at the beginning of the Start condition (Figure 20-28).

b) SCLx is sampled low before SDAx is asserted low (Figure 20-29).

During a Start condition, both the SDAx and the SCLx pins are monitored.

If the SDAx pin is already low, or the SCLx pin is already low, then all of the following occur:

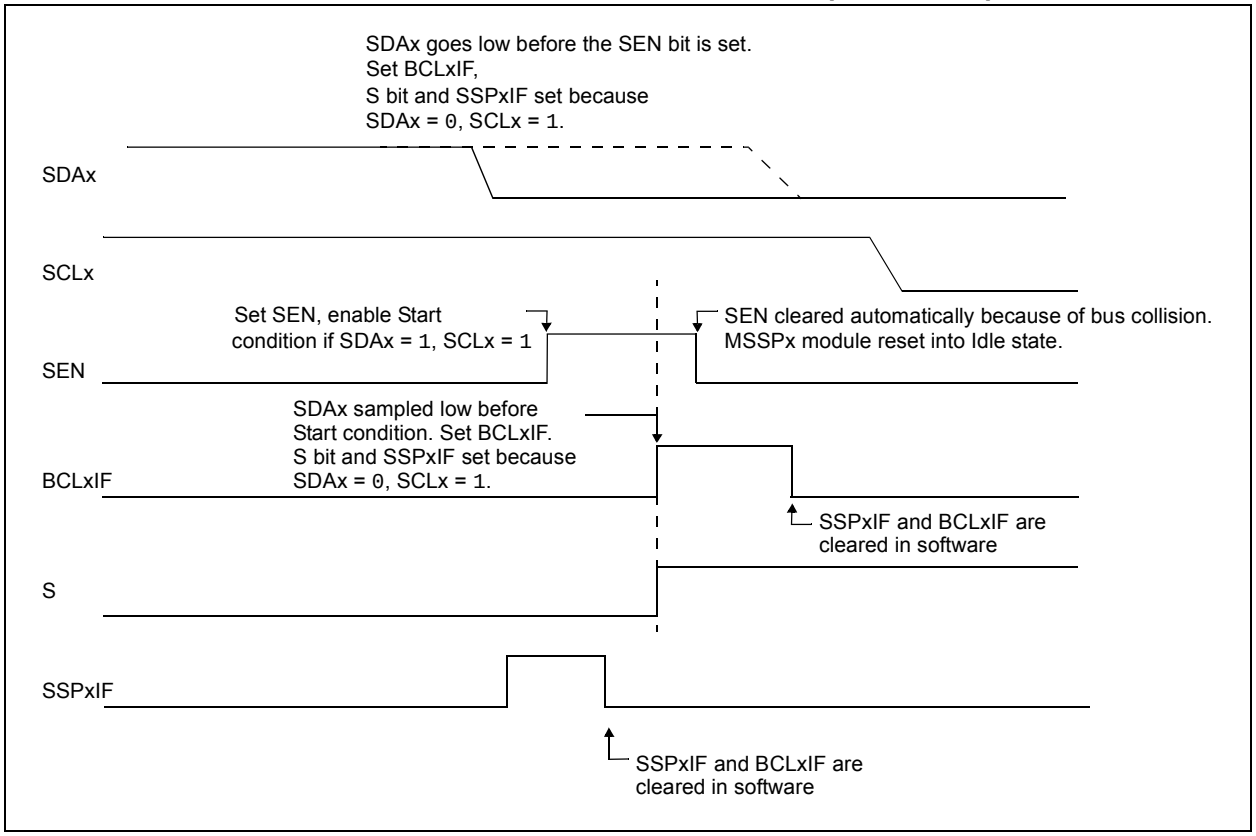
- The Start condition is aborted
- The BCLxIF flag is set
- The MSSP module is reset to its inactive state (Figure 20-28)

The Start condition begins with the SDAx and SCLx pins deasserted. When the SDAx pin is sampled high, the BRG is loaded from SSPxADD<6:0> and counts down to 0. If the SCLx pin is sampled low while SDAx is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDAx pin is sampled low during this count, the BRG is reset and the SDAx line is asserted early (Figure 20-30). If, however, a '1' is sampled on the SDAx pin, the SDAx pin is asserted low at the end of the BRG count. The BRG is then reloaded and counts down to 0. If the SCLx pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCLx pin is asserted low.

Note:	The reason that a bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDAx before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 20-28: BUS COLLISION DURING START CONDITION (SDAx ONLY)



REGISTER 28-10: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F47J53 FAMILY DEVICES (BYTE ADDRESS 3FFFFFh)

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

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

bit 7-0 **DEV<10:3>:** Device ID bits
These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.

DEV<10:3> (DEVID2<7:0>)	DEV<2:0> (DEVID2<7:5>)	Device
0101 1000	111	PIC18F47J53
0101 1000	101	PIC18F46J53
0101 1000	011	PIC18F27J53
0101 1000	001	PIC18F26J53
0101 1010	111	PIC18LF47J53
0101 1010	101	PIC18LF46J53
0101 1010	011	PIC18LF27J53
0101 1010	001	PIC18LF26J53

BNC **Branch if Not Carry**

Syntax: BNC n

Operands: -128 ≤ n ≤ 127

Operation: if Carry bit is '0',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0011	nnnn	nnnn

Description: If the Carry bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

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

If No Jump:

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

Example: HERE BNC Jump

Before Instruction
PC = address (HERE)
After Instruction
If Carry = 0;
PC = address (Jump)
If Carry = 1;
PC = address (HERE + 2)

BNN **Branch if Not Negative**

Syntax: BNN n

Operands: -128 ≤ n ≤ 127

Operation: if Negative bit is '0',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0111	nnnn	nnnn

Description: If the Negative bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

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

If No Jump:

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

Example: HERE BNN Jump

Before Instruction
PC = address (HERE)
After Instruction
If Negative = 0;
PC = address (Jump)
If Negative = 1;
PC = address (HERE + 2)

ADDWF **ADD W to Indexed (Indexed Literal Offset mode)**

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$

Operation: (W) + ((FSR2) + k) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0010	01d0	kkkk	kkkk

Description: The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'.

If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process Data	Write to destination

Example: ADDWF [OFST],0

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

BSF **Bit Set Indexed (Indexed Literal Offset mode)**

Syntax: BSF [k], b

Operands: $0 \le f \le 95$
$0 \le b \le 7$

Operation: 1 → ((FSR2) + k)<b>

Status Affected: None

Encoding:

1000	bbb0	kkkk	kkkk

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: BSF [FLAG_OFST], 7

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h
After Instruction
Contents of 0A0Ah = D5h

SETF **Set Indexed (Indexed Literal Offset mode)**

Syntax: SETF [k]

Operands: $0 \le k \le 95$

Operation: FFh → ((FSR2) + k)

Status Affected: None

Encoding:

0110	1000	kkkk	kkkk

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process Data	Write register

Example: SETF [OFST]

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh

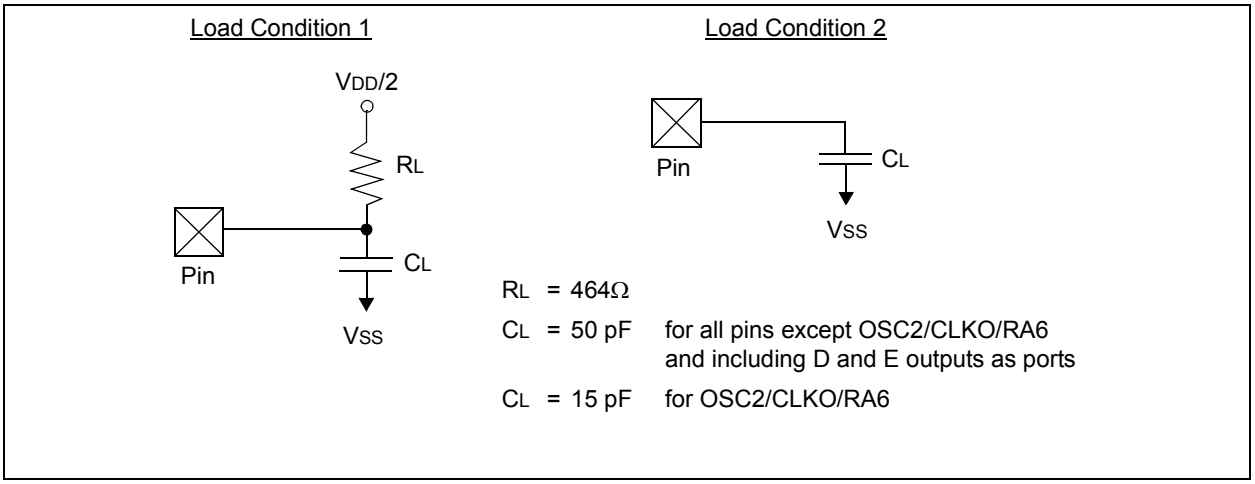
31.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 31-9 apply to all timing specifications unless otherwise noted. Figure 31-4 specifies the load conditions for the timing specifications.

TABLE 31-9: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial Operating voltage VDD range as described in **Section 31.1** and **Section 31.3**.

FIGURE 31-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



31.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 31-5: EXTERNAL CLOCK TIMING