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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf25j11t-i-ss

TABLE 1-3: PIC18F2XJ11 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
VSS1	8	5	P	—	Ground reference for logic and I/O pins.
VSS2	19	16	—	—	
VDD	20	17	P	—	Positive supply for peripheral digital logic and I/O pins.
VDDCORE/VCAP	6	3			Core logic power or external filter capacitor connection.
VDDCORE			P	—	Positive supply for microcontroller core logic (regulator disabled).
VCAP			P	—	External filter capacitor connection (regulator enabled).

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD); DIG = Digital output

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

TABLE 1-4: PIC18F4XJ11 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/RP3	9	8			
RB0			I/O	DIG	Digital I/O.
AN12			I	Analog	Analog input 12.
INT0			I	ST	External interrupt 0.
RP3			I/O	DIG	Remappable peripheral pin 3.
RB1/AN10/PMBE/RTCC/RP4	10	9			
RB1			I/O	DIG	Digital I/O.
AN10			I	Analog	Analog input 10.
PMBE			O	DIG	Parallel Master Port byte enable.
RTCC			O	DIG	Real Time Clock Calendar output.
RP4			I/O	DIG	Remappable peripheral pin 4.
RB2/AN8/CTED1/PMA3/REFO/RP5	11	10			
RB2			I/O	DIG	Digital I/O.
AN8			I	Analog	Analog input 8.
CTED1			I	ST	CTMU edge 1 input.
PMA3			O	DIG	Parallel Master Port address.
REFO			O	DIG	Reference output clock.
RP5			I/O	DIG	Remappable peripheral pin 5.
RB3/AN9/CTED2/PMA2/RP6	12	11			
RB3			I/O	DIG	Digital I/O.
AN9			I	Analog	Analog input 9.
CTED2			I	ST	CTMU edge 2 input.
PMA2			O	DIG	Parallel Master Port address.
RP6			I/O	DIG	Remappable peripheral pin 6.

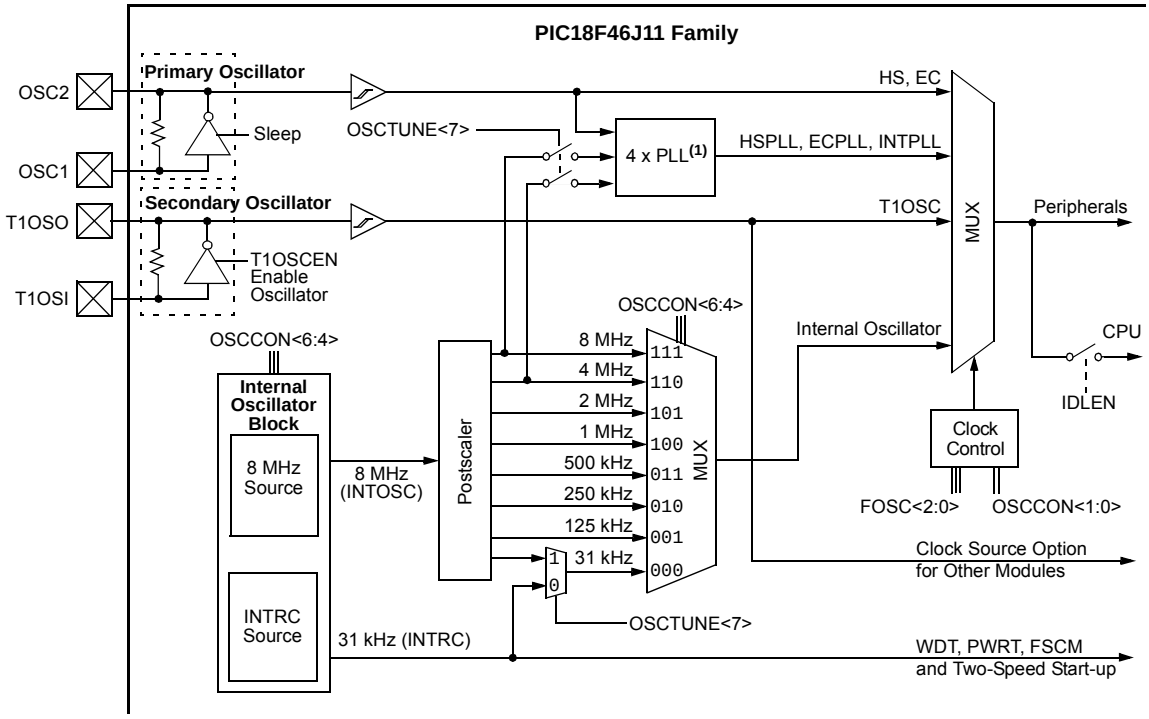
Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to V_{DD}); DIG = Digital output

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

3.2.1 OSCILLATOR MODES

Figure 3-1 helps in understanding the oscillator structure of the PIC18F46J11 family of devices.

FIGURE 3-1: PIC18F46J11 FAMILY CLOCK DIAGRAM



Note 1: 8 MHz and 4 MHz are valid INTOSC postscaler settings for the PLL. Selecting other INTOSC postscaler settings will operate the PLL outside of the specification.

4.6.3 DEEP SLEEP WAKE-UP SOURCES

While in Deep Sleep mode, the device can be awakened by a MCLR, POR, RTCC, INT0 I/O pin interrupt, DSWDT or ULPWU event. After waking, the device performs a POR. When the device is released from Reset, code execution will begin at the device's Reset vector.

The software can determine if the wake-up was caused from an exit from Deep Sleep mode by reading the DS bit (WDTCON<3>). If this bit is set, the POR was caused by a Deep Sleep exit. The DS bit must be manually cleared by the software.

The software can determine the wake event source by reading the DSWAKEH and DSWAKEL registers. When the application firmware is done using the DSWAKEH and DSWAKEL status registers, individual bits do not need to be manually cleared before entering Deep Sleep again. When entering Deep Sleep mode, these registers are automatically cleared.

4.6.3.1 Wake-up Event Considerations

Deep Sleep wake-up events are only monitored while the processor is fully in Deep Sleep mode. If a wake-up event occurs before Deep Sleep mode is entered, the event status will not be reflected in the DSWAKE registers. If the wake-up source asserts prior to entering Deep Sleep, the CPU may go to the interrupt vector (if the wake source has an interrupt bit and the interrupt is fully enabled), and may abort the Deep Sleep entry sequence by executing past the SLEEP instruction. In this case, a wake-up event handler should be placed after the SLEEP instruction to process the event and re-attempt entry into Deep Sleep if desired.

When the device is in Deep Sleep with more than one wake-up source simultaneously enabled, only the first wake-up source to assert will be detected and logged in the DSWAKEH/DSWAKEL status registers.

4.6.4 DEEP SLEEP WATCHDOG TIMER (DSWDT)

Deep Sleep has its own dedicated WDT (DSWDT) with a postscaler for time-outs of 2.1 ms to 25.7 days, configurable through the bits, DSWDTPS<3:0> (CONFIG3L<7:4>).

The DSWDT can be clocked from either the INTRC or the T1OSC/T1CKI input. If the T1OSC/T1CKI source will be used with a crystal, the T1OSCEN bit in the T1CON register needs to be set prior to entering Deep Sleep. The reference clock source is configured through the DSWDTOSC bit (CONFIG3L<0>).

DSWDT is enabled through the DSWDTEN bit (CONFIG3L<3>). Entering Deep Sleep mode automatically clears the DSWDT. See **Section 26.0 "Special Features of the CPU"** for more information.

4.6.5 DEEP SLEEP BROWN OUT RESET (DSBOR)

The Deep Sleep module contains a dedicated Deep Sleep BOR (DSBOR) circuit. This circuit may be optionally enabled through the DSBOREN Configuration bit (CONFIG3L<2>).

The DSBOR circuit monitors the V_{DD} supply rail voltage. The behavior of the DSBOR circuit is described in **Section 5.4 "Brown-out Reset (BOR)"**.

4.6.6 RTCC PERIPHERAL AND DEEP SLEEP

The RTCC can operate uninterrupted during Deep Sleep mode. It can wake the device from Deep Sleep by configuring an alarm.

The RTCC clock source is configured with the RTCOSC bit (CONFIG3L<1>). The available reference clock sources are the INTRC and T1OSC/T1CKI. If the INTRC is used, the RTCC accuracy will directly depend on the INTRC tolerance. For more information on configuring the RTCC peripheral, see **Section 17.0 "Real-Time Clock and Calendar (RTCC)"**.

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 27.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY

Program Memory Byte Locations →

			LSB = 1	LSB = 0	Word Address ↓
					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

REGISTER 10-9: RPINR4: PERIPHERAL PIN SELECT INPUT REGISTER 4 (BANKED EEAh)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T0CKR4	T0CKR3	T0CKR2	T0CKR1	T0CKR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 4-0 **T0CKR<4:0>:** Timer0 External Clock Input (T0CKI) to the Corresponding RPn Pin bits

REGISTER 10-10: RPINR6: PERIPHERAL PIN SELECT INPUT REGISTER 6 (BANKED EECh)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T3CKR4	T3CKR3	T3CKR2	T3CKR1	T3CKR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 4-0 **T3CKR<4:0>:** Timer 3 External Clock Input (T3CKI) to the Corresponding RPn Pin bits

REGISTER 10-11: RPINR7: PERIPHERAL PIN SELECT INPUT REGISTER 7 (BANKED EEDh)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	IC1R4	IC1R3	IC1R2	IC1R1	IC1R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 4-0 **IC1R<4:0>:** Assign Input Capture 1 (ECCP1) to the Corresponding RPn Pin bits

REGISTER 10-21: RPOR0: PERIPHERAL PIN SELECT OUTPUT REGISTER 0 (BANKED EC6h)(1)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP0R4	RP0R3	RP0R2	RP0R1	RP0R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 4-0 **RP0R<4:0>:** Peripheral Output Function is Assigned to RP0 Output Pin bits (see Table 10-14 for peripheral function numbers)

Note 1: Register values can be changed only if PPSCON<IOLOCK> = 0.

REGISTER 10-22: RPOR1: PERIPHERAL PIN SELECT OUTPUT REGISTER 1 (BANKED EC7h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP1R4	RP1R3	RP1R2	RP1R1	RP1R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 4-0 **RP1R<4:0>:** Peripheral Output Function is Assigned to RP1 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-23: RPOR2: PERIPHERAL PIN SELECT OUTPUT REGISTER 2 (BANKED EC8h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP2R4	RP2R3	RP2R2	RP2R1	RP2R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 4-0 **RP2R<4:0>:** Peripheral Output Function is Assigned to RP2 Output Pin bits (see Table 10-14 for peripheral function numbers)

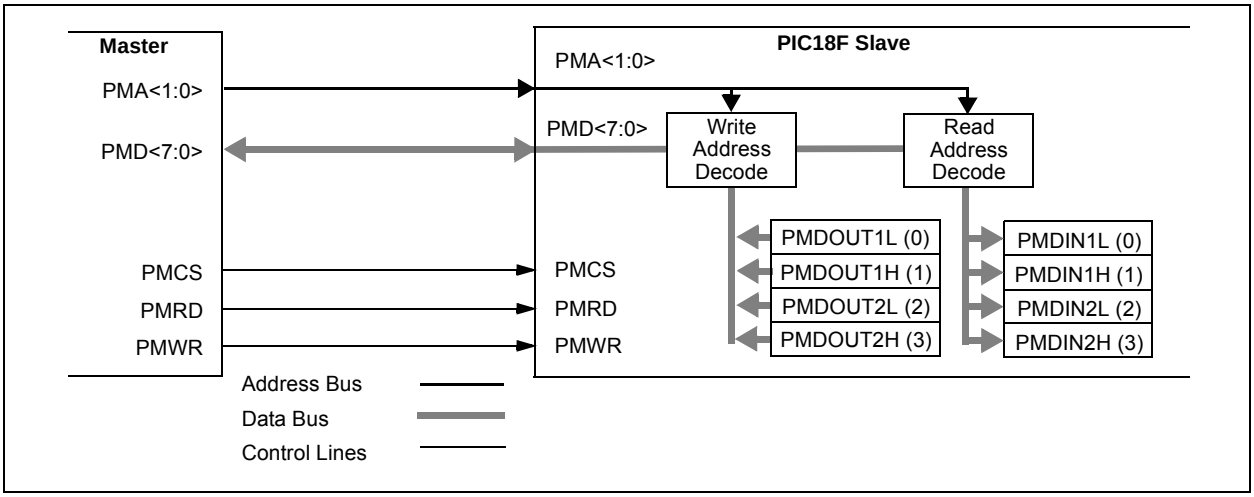
11.2.5 ADDRESSABLE PARALLEL SLAVE PORT MODE

In the Addressable Parallel Slave Port mode (PMMODEH<1:0> = 01), the module is configured with two extra inputs, PMA<1:0>, which are the address lines 1 and 0. This makes the 4-byte buffer space directly addressable as fixed pairs of read and write buffers. As with Legacy Buffered mode, data is output from PMDOUT1L, PMDOUT1H, PMDOUT2L and PMDOUT2H, and is read in on PMDIN1L, PMDIN1H, PMDIN2L and PMDIN2H. Table 11-1 provides the buffer addressing for the incoming address to the input and output registers.

TABLE 11-1: SLAVE MODE BUFFER ADDRESSING

PMA<1:0>	Output Register (Buffer)	Input Register (Buffer)
00	PMDOUT1L (0)	PMDIN1L (0)
01	PMDOUT1H (1)	PMDIN1H (1)
10	PMDOUT2L (2)	PMDIN2L (2)
11	PMDOUT2H((3)	PMDIN2H (3)

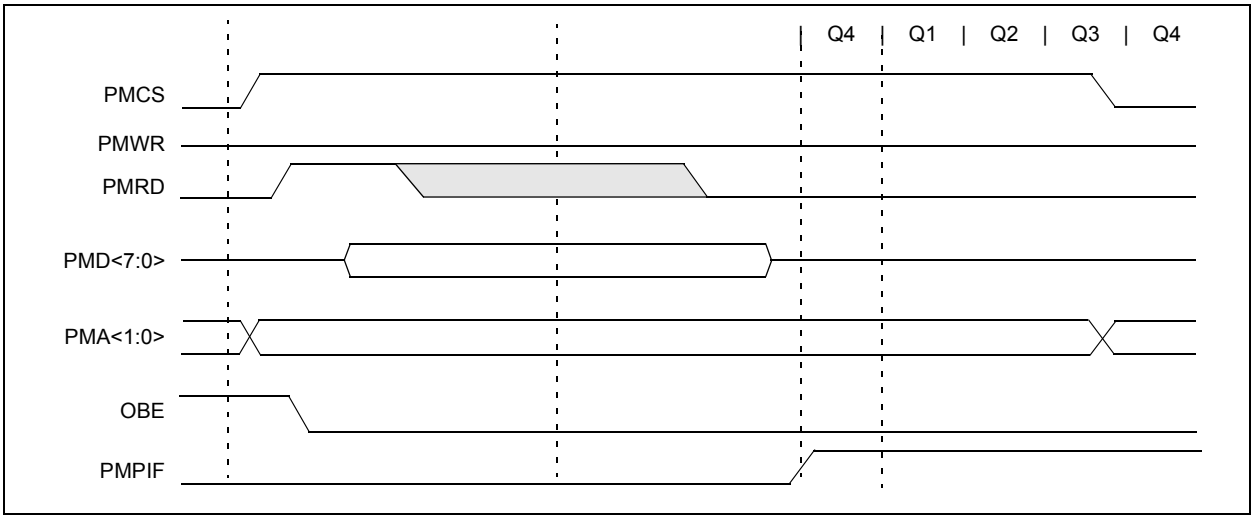
FIGURE 11-6: PARALLEL MASTER/SLAVE CONNECTION ADDRESSED BUFFER EXAMPLE



11.2.5.1 READ FROM SLAVE PORT

When chip select is active and a read strobe occurs (PMCS = 1 and PMRD = 1), the data from one of the four output bytes is presented onto PMD<7:0>. Which byte is read depends on the 2-bit address placed on ADDR<1:0>. Table 11-1 provides the corresponding output registers and their associated address. When an output buffer is read, the corresponding OBxE bit is set. The OBxE flag bit is set when all the buffers are empty. If any buffer is already empty, OBxE = 1, the next read to that buffer will generate an OBUF event.

FIGURE 11-7: PARALLEL SLAVE PORT READ WAVEFORMS



19.2 Control Registers

Each MSSP module has three associated control registers. These include a status register (SSPxSTAT) and two control registers (SSPxCON1 and SSPxCON2). The use of these registers and their individual Configuration bits differ significantly depending on whether the MSSP module is operated in SPI or I^2C mode.

Additional details are provided under the individual sections.

Note:	In devices with more than one MSSP module, it is very important to pay close attention to the SSPxCON register names. SSP1CON1 and SSP1CON2 control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

19.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported.

When MSSP2 is used in SPI mode, it can optionally be configured to work with the SPI DMA submodule described in **Section 19.4 "SPI DMA Module"**.

To accomplish communication, typically three pins are used:

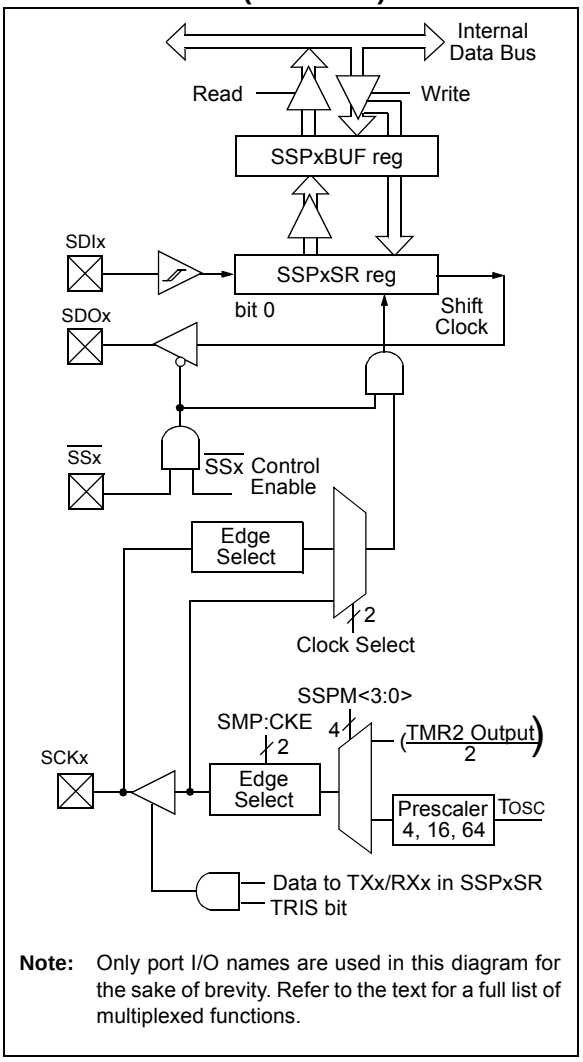
- Serial Data Out (SDOx) – RC5/SDO1/RP16 or SDO2/Remappable
- Serial Data In (SDIx) – RC4/SDI1/SDA1/RP15 or SDI2/Remappable
- Serial Clock (SCKx) – RC3/SCK1/SCL1/RP14 or SCK2/Remappable

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SSx}$) – RA5/AN4/$\overline{SS1}$/ HLVDIN/RP2 or $\overline{SS2}$/Remappable

Figure 19-1 depicts the block diagram of the MSSP module when operating in SPI mode.

FIGURE 19-1: MSSPx BLOCK DIAGRAM (SPI MODE)



Note: Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

19.5.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address, is accomplished by simply writing a value to the SSPxBUF register. This action will set the Buffer Full flag bit, BF, and allow the BRG to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDAx pin after the falling edge of SCLx is asserted (see data hold time specification parameter 106). SCLx is held low for one BRG rollover count (TBRG). Data should be valid before SCLx is released high (see data setup time specification parameter 107). When the SCLx pin is released high, it is held that way for TBRG.

The data on the SDAx pin must remain stable for that duration and some hold time after the next falling edge of SCLx. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDAx. This allows the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ is written into the ACKDT bit on the falling edge of the ninth clock.

If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared; if not, the bit is set. After the ninth clock, the SSPxIF bit is set and the master clock (BRG) is suspended until the next data byte is loaded into the SSPxBUF, leaving SCLx low and SDAx unchanged (Figure 19-23).

After the write to the SSPxBUF, each bit of the address will be shifted out on the falling edge of SCLx until all seven address bits and the R/$\overline{W}$ bit are completed. On the falling edge of the eighth clock, the master will deassert the SDAx pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDAx pin to see if the address was recognized by a slave. The status of the $\overline{ACK}$ bit is loaded into the ACKSTAT status bit (SSPxCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPxIF flag is set, the BF flag is cleared and the BRG is turned off until another write to the SSPxBUF takes place, holding SCLx low and allowing SDAx to float.

19.5.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPxSTAT<0>) is set when the CPU writes to SSPxBUF and is cleared when all eight bits are shifted out.

19.5.10.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPxSR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur) after 2 TCY after the SSPxBUF write. If SSPxBUF is rewritten within 2 TCY, the WCOL bit is set and SSPxBUF is updated. This may result in a corrupted transfer.

The user should verify that the WCOL bit is clear after each write to SSPxBUF to ensure the transfer is correct. In all cases, WCOL must be cleared in software.

19.5.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPxCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

19.5.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPxCON2<3>).

Note:	The MSSP module must be in an inactive state before the RCEN bit is set or the RCEN bit will be disregarded.

The BRG begins counting and on each rollover, the state of the SCLx pin changes (high-to-low/low-to-high) and data is shifted into the SSPxSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPxSR are loaded into the SSPxBUF, the BF flag bit is set, the SSPxIF flag bit is set and the BRG is suspended from counting, holding SCLx low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPxCON2<4>).

19.5.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPxBUF from SSPxSR. It is cleared when the SSPxBUF register is read.

19.5.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPxSR and the BF flag bit is already set from a previous reception.

19.5.11.3 WCOL Status Flag

If users write the SSPxBUF when a receive is already in progress (i.e., SSPxSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

The ANCON0 and ANCON1 registers are used to configure the operation of the I/O pin associated with each analog channel. Setting any one of the PCFG bits configures the corresponding pin to operate as a digital only I/O. Clearing a bit configures the pin to operate as an analog input for either the A/D Converter or the comparator module; all digital peripherals are disabled and digital inputs read as '0'. As a rule, I/O pins that are multiplexed with analog inputs default to analog operation on device Resets.

In order to correctly perform A/D conversions on the VBG band gap reference (ADCON0<5:2> = 1111), the reference circuit must be powered on first. The VBGEN bit in the ANCON1 register allows the firmware to manually request that the band gap reference circuit should be enabled. For best accuracy, firmware should allow a settling time of at least 10 ms prior to performing the first acquisition on this channel after enabling the band gap reference.

The reference circuit may already have been turned on if some other hardware module (such as comparators or HLVD) has already requested it. In this case, the initial turn-on settling time may have already elapsed and firmware does not need to wait as long before measuring VBG. Once the acquisition is complete, firmware may clear the VBGEN bit, which will save a small amount of power if no other modules are still requesting the VBG reference.

REGISTER 21-3: ANCON0: A/D PORT CONFIGURATION REGISTER 2 (BANKED F48h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7(1)	PCFG6(1)	PCFG5(1)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
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 **PCFG<7:0>:** Analog Port Configuration bits (AN<7:0>)
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads '0'

Note 1: These bits are not implemented on 28-pin devices.

REGISTER 21-4: ANCON1: A/D PORT CONFIGURATION REGISTER 1 (BANKED F49h)

R/W-0	r	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VBGEN	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 7							bit 0

Legend:	r = Reserved		
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 **VBGEN:** 1.2V Band Gap Reference Enable bit
1 = 1.2V band gap reference is powered on
0 = 1.2V band gap reference is turned off to save power (if no other modules are requesting it)

bit 6 **Reserved:** Always maintain as '0' for lowest power consumption

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

bit 4-0 **PCFG<12:8>:** Analog Port Configuration bits (AN<12:8>)
1 = Pin configured as a digital port
0 = Pin configured as an analog channel – digital input disabled and reads '0'

25.9 Registers

There are three control registers for the CTMU:

- CTMUCONH
- CTMUCONL
- CTMUICON

The CTMUCONH and CTMUCONL registers (Register 25-1 and Register 25-2) contain control bits for configuring the CTMU module edge source selection, edge source polarity selection, edge sequencing, A/D trigger, analog circuit capacitor discharge and enables. The CTMUICON register (Register 25-3) has bits for selecting the current source range and current source trim.

REGISTER 25-1: CTMUCONH: CTMU CONTROL REGISTER HIGH (ACCESS FB3h)

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	—
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 **CTMUEN:** CTMU Enable bit
1 = Module is enabled
0 = Module is disabled

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **CTMUSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 4 **TGEN:** Time Generation Enable bit
1 = Enables edge delay generation
0 = Disables edge delay generation

bit 3 **EDGEN:** Edge Enable bit
1 = Edges are not blocked
0 = Edges are blocked

bit 2 **EDGSEQEN:** Edge Sequence Enable bit
1 = Edge 1 event must occur before Edge 2 event can occur
0 = No edge sequence is needed

bit 1 **IDISSEN:** Analog Current Source Control bit
1 = Analog current source output is grounded
0 = Analog current source output is not grounded

bit 0 **Reserved:** Write as ‘0’

REGISTER 25-2: CTMUCONL: CTMU CONTROL REGISTER LOW (ACCESS FB2h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x	R/W-x
EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT
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 **EDG2POL:** Edge 2 Polarity Select bit
1 = Edge 2 programmed for a positive edge response
0 = Edge 2 programmed for a negative edge response

bit 6-5 **EDG2SEL<1:0>:** Edge 2 Source Select bits
11 = CTED1 pin
10 = CTED2 pin
01 = ECCP1 Special Event Trigger
00 = ECCP2 Special Event Trigger

bit 4 **EDG1POL:** Edge 1 Polarity Select bit
1 = Edge 1 programmed for a positive edge response
0 = Edge 1 programmed for a negative edge response

bit 3-2 **EDG1SEL<1:0>:** Edge 1 Source Select bits
11 = CTED1 pin
10 = CTED2 pin
01 = ECCP1 Special Event Trigger
00 = ECCP2 Special Event Trigger

bit 1 **EDG2STAT:** Edge 2 Status bit
1 = Edge 2 event has occurred
0 = Edge 2 event has not occurred

bit 0 **EDG1STAT:** Edge 1 Status bit
1 = Edge 1 event has occurred
0 = Edge 1 event has not occurred

DECFSZ **Decrement f, Skip if 0**

Syntax: DECFSZ f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0010	11da	ffff	ffff

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If the result is '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    DECFSZ  CNT, 1, 1
        GOTO    LOOP
CONTINUE
```
Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT – 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE + 2)

DCFSNZ **Decrement f, Skip if not 0**

Syntax: DCFSNZ f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

0100	11da	ffff	ffff

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If the result is not '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    DCFSNZ  TEMP, 1, 0
ZERO    :
NZERO   :
```
Before Instruction
TEMP = ?
After Instruction
TEMP = TEMP – 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP ≠ 0;
PC = Address (NZERO)

MOVLW **Move Literal to W**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk

Description: The eight-bit literal ‘k’ is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example: `MOVLW 0x5A`

After Instruction
W = 5Ah

MOVWF **Move W to f**

Syntax: MOVWF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank.

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default).

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write register ‘f’

Example: `MOVWF REG, 0`

Before Instruction
W = 4Fh
REG = FFh
After Instruction
W = 4Fh
REG = 4Fh

RCALL **Relative Call**

Syntax: RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1101	1nnn	nnnn	nnnn

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' 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 a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

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

Example: HERE RCALL Jump

Before Instruction
PC = Address (HERE)
After Instruction
PC = Address (Jump)
TOS = Address (HERE + 2)

RESET **Reset**

Syntax: RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset in software.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start reset	No operation	No operation

Example: RESET

After Instruction
Registers = Reset Value
Flags* = Reset Value

TBLWT **Table Write**

Syntax: TBLWT (*; *+; *-; +*)

Operands: None

Operation: if TBLWT*,
(TABLAT) → Holding Register,
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register,
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register,
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR,
(TABLAT) → Holding Register

Status Affected: None

Encoding:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 6.0 "Memory Organization"** for additional details on programming Flash memory.)

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR<0> = 0: Least Significant Byte of Program Memory Word

TBLPTR<0> = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT **Table Write (Continued)**

Example 1: TBLWT *+

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
HOLDING REGISTER (00A356h) = FFh
After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
HOLDING REGISTER (00A356h) = 55h

Example 2: TBLWT +*

Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = FFh
After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = 34h

27.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

Note: Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing (**Section 6.6.1 "Indexed Addressing with Literal Offset"**). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank (a = 0) or in a GPR bank designated by the BSR (a = 1). When the extended instruction set is enabled and a = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward-compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see **Section 27.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**).

Although the Indexed Literal Offset mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset mode are provided on the following page to show how execution is affected. The operand conditions provided in the examples are applicable to all instructions of these types.

27.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument 'f' in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within the brackets, will generate an error in the MPASM Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled) when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM Assembler.

The destination argument 'd' functions as before.

In the latest versions of the MPASM Assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

27.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F46J11 family, it is very important to consider the type of code. A large, re-entrant application that is written in C and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

29.3 DC Characteristics: PIC18F46J11 Family (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial			
Param No.	**Symbol**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**				
		All I/O ports:				
D030		with TTL Buffer	VSS	0.15 VDD	V	VDD < 3.3V
D030A			—	0.8	V	3.3V ≤ VDD ≤ 3.6V
D031		with Schmitt Trigger Buffer	VSS	0.2 VDD	V	
D031A		SDAx/SCLx	—	0.3 VDD	V	I²C™ enabled
D031B			—	0.8	V	SMBus enabled
D032		$\overline{MCLR}$	VSS	0.2 VDD	V	
D033		OSC1	VSS	0.3 VDD	V	HS, HSPLL modes
D033A		OSC1	VSS	0.2 VDD	V	EC, ECPLL modes
D034		T1OSI	VSS	0.3	V	T1OSCEN = 1
	VIH	**Input High Voltage**				
		I/O Ports with non 5.5V Tolerance:(4)				
D040		with TTL Buffer	0.25 VDD + 0.8V	VDD	V	VDD < 3.3V
D040A			2.0	VDD	V	3.3V ≤ VDD ≤ 3.6V
D041		with Schmitt Trigger Buffer	0.8 VDD	VDD	V	
		I/O Ports with 5.5V Tolerance:(4)			V	
Dxxx		with TTL Buffer	0.25 VDD + 0.8V	5.5	V	VDD < 3.3V
DxxxA			2.0	5.5	V	3.3V ≤ VDD ≤ 3.6V
Dxxx		with Schmitt Trigger Buffer	0.8 VDD	5.5	V	
D041A		SDAx/SCLx	0.7 VDD	—	V	I²C™ enabled
D041B			2.1	—		SMBus enabled, VDD ≥ 3V
D042		$\overline{MCLR}$	0.8 VDD	5.5	V	
D043		OSC1	0.7 VDD	VDD	V	HS, HSPLL modes
D043A		OSC1	0.8 VDD	VDD	V	EC, ECPLL modes
D044		T1OSI	1.6	VDD	V	T1OSCEN = 1
	IIL	**Input Leakage Current(1,2)**				
D060		I/O Ports	—	±0.2	µA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance
D061		$\overline{MCLR}$	—	±0.2	µA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	±0.2	µA	Vss ≤ VPIN ≤ VDD
	IPU	**Weak Pull-up Current**				
D070	IPURB	PORTB, PORTD(3) and PORTE(3) Weak Pull-up Current	80	400	µA	VDD = 3.3V, VPIN = VSS

Note 1: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

2: Negative current is defined as current sourced by the pin.

3: Only available in 44-pin devices.

4: Refer to Table 10-2 for the pins that have corresponding tolerance limits.

FIGURE 29-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)

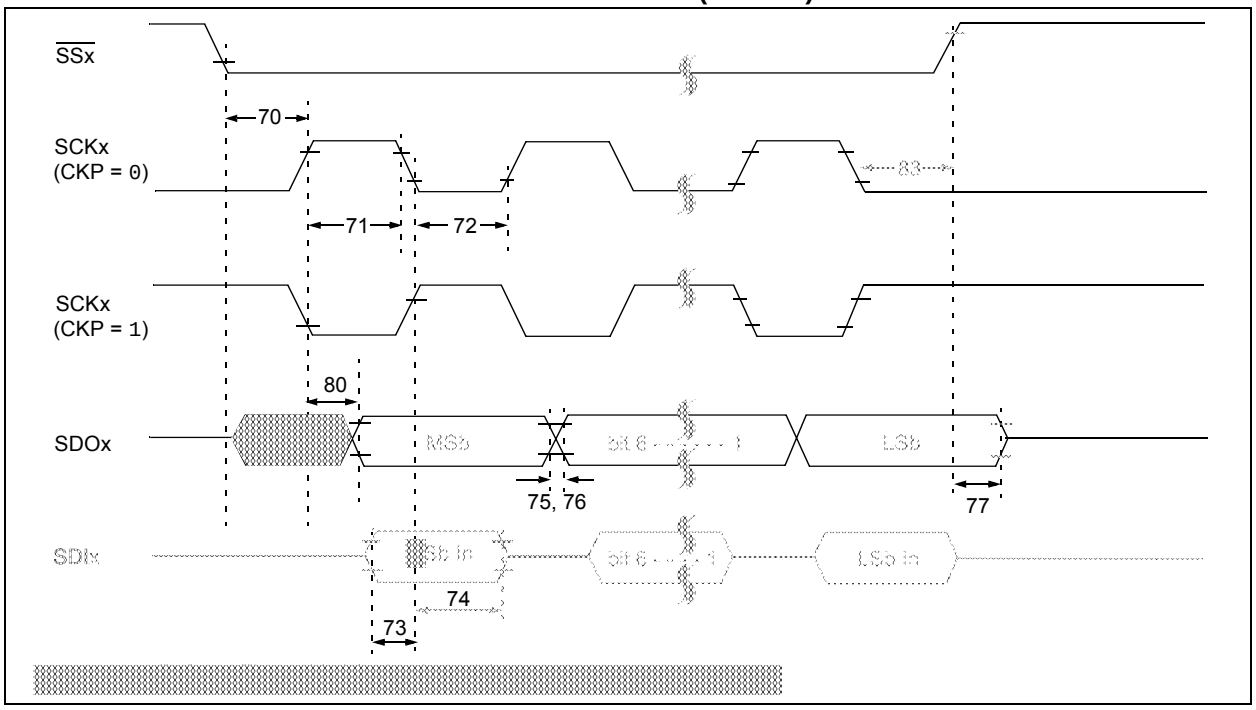


TABLE 29-22: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SSx}$ ↓ to SCKx ↓ or SCKx ↑ Input		3 Tcy	—	ns	
70A	TssL2WB	$\overline{SSx}$ ↓ to Write to SSPxBUF		3 Tcy	—	ns	
71	TscH	SCKx Input High Time (Slave mode)	Continuous	1.25 Tcy + 30	—	ns	
71A			Single byte	40	—	ns	**(Note 1)**
72	TscL	SCKx Input Low Time (Slave mode)	Continuous	1.25 Tcy + 30	—	ns	
72A			Single byte	40	—	ns	**(Note 1)**
73	TdiV2scH, TdiV2scL	Setup Time of SDIx Data Input to SCKx Edge		25	—	ns	
73A	Tb2b	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2		1.5 Tcy + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold Time of SDIx Data Input to SCKx Edge		35	—	ns	VDD = 3.3V, VDDCORE = 2.5V
				100	—	ns	VDD = 2.15V
75	TdoR	SDOx Data Output Rise Time		—	25	ns	PORTB or PORTC
76	TdoF	SDOx Data Output Fall Time		—	25	ns	PORTB or PORTC
77	TssH2doZ	$\overline{SSx}$ ↑ to SDOx Output High-Impedance		10	70	ns	
80	TscH2doV, TscL2doV	SDOx Data Output Valid after SCKx Edge		—	50	ns	VDD = 3.3V, VDDCORE = 2.5V
					100	ns	VDD = 2.15V
83	TscH2ssH, TscL2ssH	$\overline{SSx}$ ↑ after SCKx Edge		1.5 Tcy + 40	—	ns	

Note 1: Requires the use of Parameter #73A.
2: Only if Parameter #71A and #72A are used.