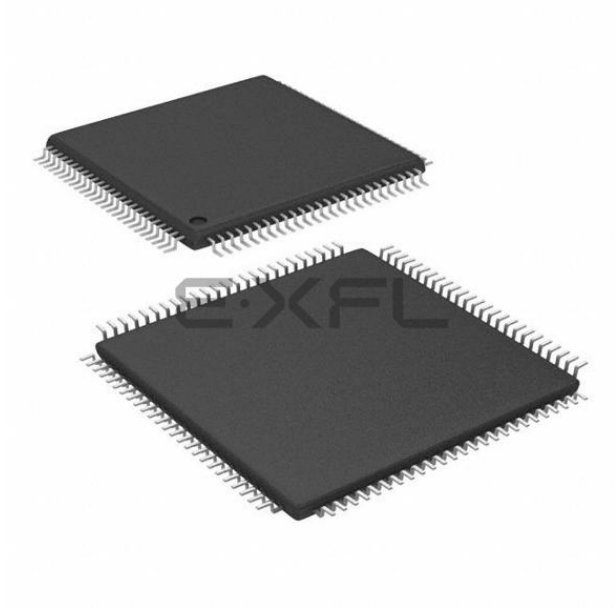


Welcome to E-XFL.COM

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	41.667MHz
Connectivity	EBI/EMI, Ethernet, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	70
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3808 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f97j60t-i-pf

1.2 Other Special Features

- **Communications:** The PIC18F97J60 family incorporates a range of serial communication peripherals, including up to two independent Enhanced USARTs and up to two Master SSP modules, capable of both SPI and I^2C™ (Master and Slave) modes of operation. In addition, one of the general purpose I/O ports can be reconfigured as an 8-bit Parallel Slave Port for direct processor-to-processor communications.
- **CCP Modules:** All devices in the family incorporate two Capture/Compare/PWM (CCP) modules and three Enhanced CCP (ECCP) modules to maximize flexibility in control applications. Up to four different time bases may be used to perform several different operations at once. Each of the three ECCP modules offers up to four PWM outputs, allowing for a total of twelve PWMs. The ECCP modules also offer many beneficial features, including polarity selection, programmable dead time, auto-shutdown and restart and Half-Bridge and Full-Bridge Output modes.
- **10-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reducing code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing an extended time-out range. See **Section 28.0 "Electrical Characteristics"** for time-out periods.

1.3 Details on Individual Family Members

Devices in the PIC18F97J60 family are available in 64-pin, 80-pin and 100-pin packages. Block diagrams for the three groups are shown in Figure 1-1, Figure 1-2 and Figure 1-3.

The devices are differentiated from each other in four ways:

1. Flash program memory (three sizes, ranging from 64 Kbytes for PIC18FX6J60 devices to 128 Kbytes for PIC18FX7J60 devices).
2. A/D channels (eleven for 64-pin devices, fifteen for 80-pin pin devices and sixteen for 100-pin devices).
3. Serial communication modules (one EUSART module and one MSSP module on 64-pin devices, two EUSART modules and one MSSP module on 80-pin devices and two EUSART modules and two MSSP modules on 100-pin devices).
4. I/O pins (39 on 64-pin devices, 55 on 80-pin devices and 70 on 100-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1, Table 1-2 and Table 1-3.

The pinouts for all devices are listed in Table 1-4, Table 1-5 and Table 1-6.

TABLE 1-3: DEVICE FEATURES FOR THE PIC18F97J60 FAMILY (100-PIN DEVICES)

Features	PIC18F96J60	PIC18F96J65	PIC18F97J60
Operating Frequency	DC – 41.667 MHz	DC – 41.667 MHz	DC – 41.667 MHz
Program Memory (Bytes)	64K	96K	128K
Program Memory (Instructions)	32764	49148	65532
Data Memory (Bytes)	3808		
Interrupt Sources	29		
I/O Ports	Ports A, B, C, D, E, F, G, H, J		
I/O Pins	70		
Timers	5		
Capture/Compare/PWM Modules	2		
Enhanced Capture/Compare/PWM Modules	3		
Serial Communications	MSSP (2), Enhanced USART (2)		
Ethernet Communications (10Base-T)	Yes		
Parallel Slave Port Communications (PSP)	Yes		
External Memory Bus	Yes		
10-Bit Analog-to-Digital Module	16 Input Channels		
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{MCLR}$, WDT (PWRT, OST)		
Instruction Set	75 Instructions, 83 with Extended Instruction Set Enabled		
Packages	100-Pin TQFP		

TABLE 1-4: PIC18F66J60/66J65/67J60 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
				PORTE is a bidirectional I/O port.
RE0/P2D	2			
RE0		I/O	ST	Digital I/O.
P2D		O	—	ECCP2 PWM Output D.
RE1/P2C	1			
RE1		I/O	ST	Digital I/O.
P2C		O	—	ECCP2 PWM Output C.
RE2/P2B	64			
RE2		I/O	ST	Digital I/O.
P2B		O	—	ECCP2 PWM Output B.
RE3/P3C	63			
RE3		I/O	ST	Digital I/O.
P3C		O	—	ECCP3 PWM Output C.
RE4/P3B	62			
RE4		I/O	ST	Digital I/O.
P3B		O	—	ECCP3 PWM Output B.
RE5/P1C	61			
RE5		I/O	ST	Digital I/O.
P1C		O	—	ECCP1 PWM Output C.

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})

3.5 Internal Oscillator Block

The PIC18F97J60 family of devices includes an internal oscillator source (INTRC) which provides a nominal 31 kHz output. The INTRC is enabled on device power-up and clocks the device during its configuration cycle until it enters operating mode. INTRC is also enabled if it is selected as the device clock source or if any of the following are enabled:

- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in greater detail in **Section 25.0 "Special Features of the CPU"**.

The INTRC can also be optionally configured as the default clock source on device start-up by setting the FOSC2 Configuration bit. This is discussed in **Section 3.7.1 "Oscillator Control Register"**.

3.6 Ethernet Operation and the Microcontroller Clock

Although devices of the PIC18F97J60 family can accept a wide range of crystals and external oscillator inputs, they must always have a 25 MHz clock source when used for Ethernet applications. No provision is made for internally generating the required Ethernet clock from a primary oscillator source of a different frequency. A frequency tolerance is specified, likely excluding the use of ceramic resonators. See **Section 28.0 "Electrical Characteristics"**, Table 28-6, Parameter 5, for more details.

3.6.1 PLL BLOCK

To accommodate a range of applications and microcontroller clock speeds, a separate PLL block is incorporated into the clock system. It consists of three components:

- A configurable prescaler (1:2 or 1:3)
- A 5x PLL frequency multiplier
- A configurable postscaler (1:1, 1:2, or 1:3)

The operation of the PLL block's components is controlled by the OSCTUNE register (Register 3-1). The use of the PLL block's prescaler and postscaler, with or without the PLL itself, provides a range of system clock frequencies to choose from, including the unaltered 25 MHz of the primary oscillator. The full range of possible oscillator configurations compatible with Ethernet operation is shown in Table 3-2.

REGISTER 3-1: OSCTUNE: PLL BLOCK CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
PPST1	PLLEN(1)	PPST0	PPRE	—	—	—	—
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 **PPST1:** PLL Postscaler Configuration bit
1 = Divide-by-2
0 = Divide-by-3

bit 6 **PLLEN:** 5x Frequency Multiplier PLL Enable bit(1)
1 = PLL is enabled
0 = PLL is disabled

bit 5 **PPST0:** PLL Postscaler Enable bit
1 = Postscaler is enabled
0 = Postscaler is disabled

bit 4 **PPRE:** PLL Prescaler Configuration bit
1 = Divide-by-2
0 = Divide-by-3

bit 3-0 **Unimplemented:** Read as '0'

Note 1: Available only for ECPLL and HSPLL oscillator configurations; otherwise, this bit is unavailable and is read as '0'.

4.4.1 PRI_IDLE MODE

This mode is unique among the three low-power Idle modes in that it does not disable the primary device clock. For timing-sensitive applications, this allows for the fastest resumption of device operation with its more accurate primary clock source, since the clock source does not have to “warm up” or transition from another oscillator.

PRI_IDLE mode is entered from PRI_RUN mode by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then set the SCS<1:0> bits to ‘10’ and execute SLEEP. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC<1:0> Configuration bits. The OSTS bit remains set (see Figure 4-7).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval, TCSD, is required between the wake event and when code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 4-8).

4.4.2 SEC_IDLE MODE

In SEC_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the Timer1 oscillator. This mode is entered from SEC_RUN by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then set SCS<1:0> to ‘01’ and execute SLEEP. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the T1RUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of TCSD, following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 4-8).

Note:	The Timer1 oscillator should already be running prior to entering SEC_IDLE mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the SLEEP instruction will be ignored and entry to SEC_IDLE mode will not occur. If the Timer1 oscillator is enabled, but not yet running, peripheral clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result.

FIGURE 4-7: TRANSITION TIMING FOR ENTRY TO IDLE MODE

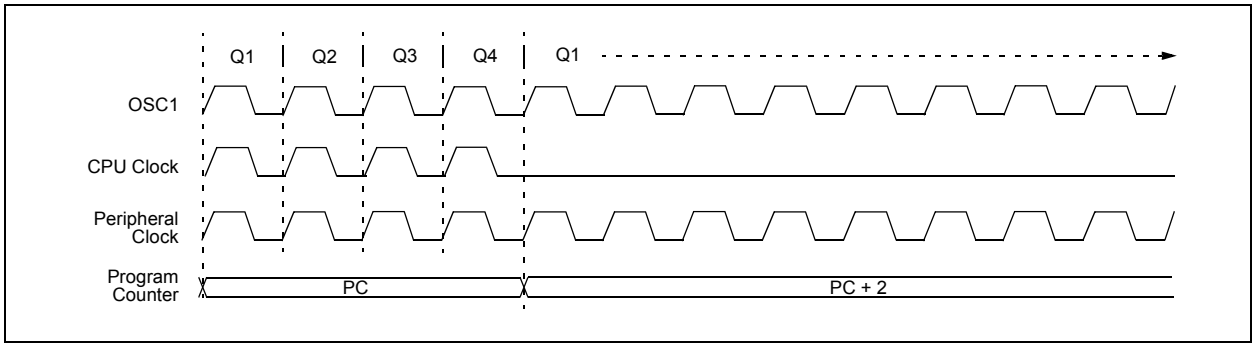
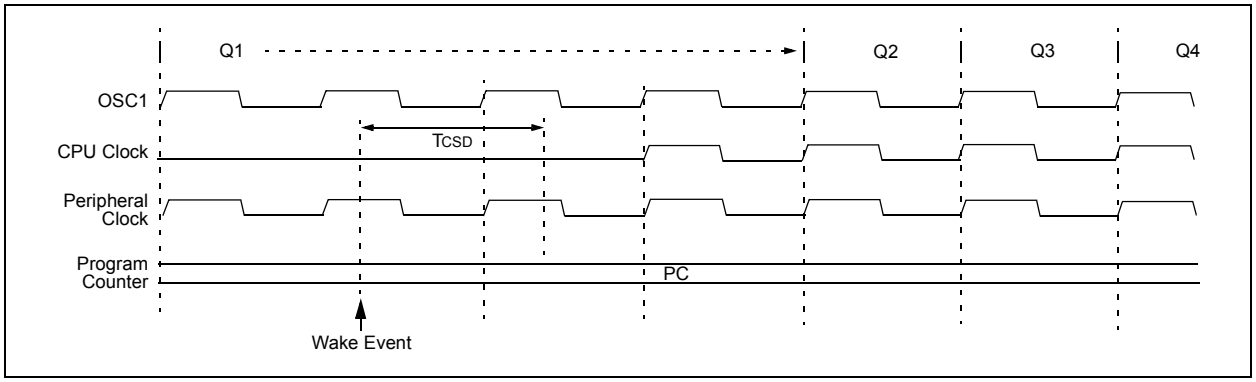


FIGURE 4-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE

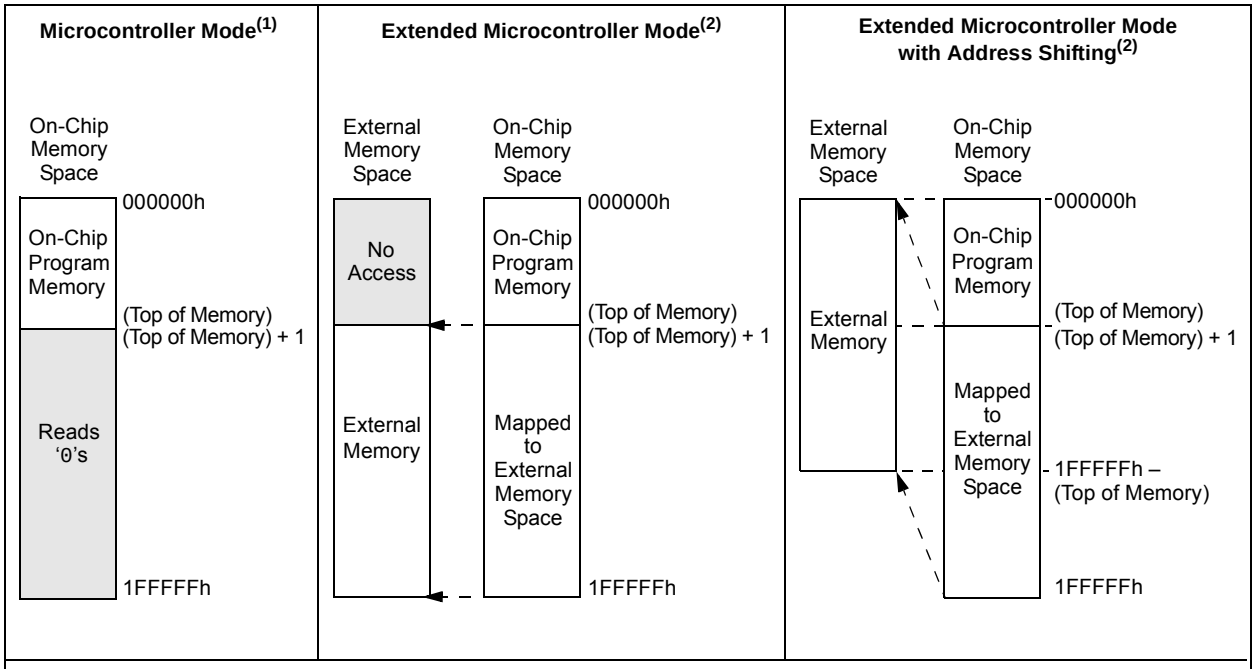


6.1.4 EXTENDED MICROCONTROLLER MODE AND ADDRESS SHIFTING

By default, devices in Extended Microcontroller mode directly present the program counter value on the external address bus for those addresses in the range of the external memory space. In practical terms, this means addresses in the external memory device below the top of on-chip memory are unavailable.

To avoid this, the Extended Microcontroller mode implements an address shifting option to enable automatic address translation. In this mode, addresses presented on the external bus are shifted down by the size of the on-chip program memory and are remapped to start at 0000h. This allows the complete use of the external memory device's memory space.

FIGURE 6-3: MEMORY MAPS FOR PIC18F97J60 FAMILY PROGRAM MEMORY MODES



Legend: (Top of Memory) represents upper boundary of on-chip program memory space (see Figure 6-1 for device-specific values). Shaded areas represent unimplemented or inaccessible areas depending on the mode.

Note 1: This mode is the only available mode on 64-pin and 80-pin devices and the default on 100-pin devices.

2: These modes are only available in 100-pin devices.

TABLE 6-2: MEMORY ACCESS FOR PIC18F9XJ60/9XJ65 PROGRAM MEMORY MODES

Operating Mode	Internal Program Memory			External Program Memory		
	Execution From	Table Read From	Table Write To	Execution From	Table Read From	Table Write To
Microcontroller	Yes	Yes	Yes	No Access	No Access	No Access
Extended Microcontroller	Yes	Yes	Yes	Yes	Yes	Yes

6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte (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 6.1.5 "Program Counter"**).

Figure 6-6 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1> which accesses the desired byte address in program memory. Instruction #2 in Figure 6-6 shows how the instruction, GOTO 0006h, is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 26.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 6-6: 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 shows 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

7.2.2 TABLE LATCH REGISTER (TABLAT)

The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The Table Latch register is used to hold 8-bit data during data transfers between program memory and data RAM.

7.2.3 TABLE POINTER REGISTER (TBLPTR)

The Table Pointer (TBLPTR) register addresses a byte within the program memory. The TBLPTR is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers join to form a 22-bit wide pointer. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. The 22nd bit allows access to the Device ID and Configuration bits.

The Table Pointer register, TBLPTR, is used by the TBLRD and TBLWT instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 7-1. The table operations on the TBLPTR only affect the low-order 21 bits.

7.2.4 TABLE POINTER BOUNDARIES

TBLPTR is used in reads, writes and erases of the Flash program memory.

When a TBLRD is executed, all 22 bits of the TBLPTR determine which byte is read from program memory into TABLAT.

When a TBLWT is executed, the six LSbs of the Table Pointer register (TBLPTR<5:0>) determine which of the 64 program memory holding registers is written to. When the timed write to program memory begins (via the WR bit), the 15 MSbs of the TBLPTR (TBLPTR<20:6>) determine which program memory block of 64 bytes is written to. For more detail, see **Section 7.5 “Writing to Flash Program Memory”**.

When an erase of program memory is executed, the 11 MSbs of the Table Pointer register (TBLPTR<20:10>) point to the 1024-byte block that will be erased. The Least Significant bits (TBLPTR<9:0>) are ignored.

Figure 7-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

TABLE 7-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

Example	Operation on Table Pointer
TBLRD* TBLWT*	TBLPTR is not modified
TBLRD*+ TBLWT*+	TBLPTR is incremented after the read/write
TBLRD*- TBLWT*-	TBLPTR is decremented after the read/write
TBLRD+* TBLWT+*	TBLPTR is incremented before the read/write

FIGURE 7-3: TABLE POINTER BOUNDARIES BASED ON OPERATION

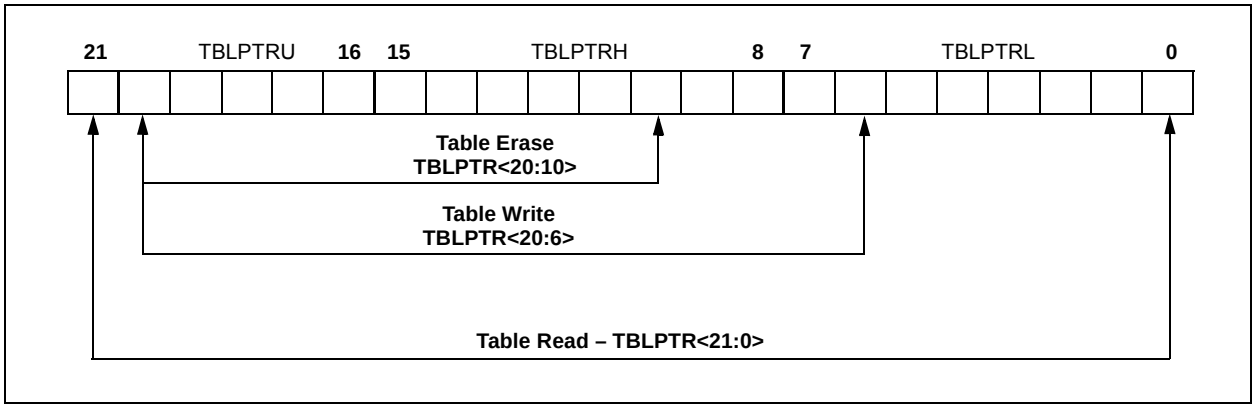


TABLE 11-9: PORTD FUNCTIONS

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RD0/AD0/PSP0 (RD0/P1B)	RD0	0	O	DIG	LATD<0> data output.
		1	I	ST	PORTD<0> data input; weak pull-up when RDPU bit is set.
	AD0(1)	x	O	DIG	External memory interface, Address/Data Bit 0 output.(2)
		x	I	TTL	External memory interface, Data Bit 0 input.(2)
	PSP0(1)	x	O	DIG	PSP read output data (LATD<0>); takes priority over port data.
		x	I	TTL	PSP write data input.
	P1B(3)	0	O	DIG	ECCP1 Enhanced PWM output, Channel B; takes priority over port and PSP data. May be configured for tri-state during Enhanced PWM shutdown events.
RD1/AD1/PSP1 (RD1/ECCP3/ P3A)	RD1	0	O	DIG	LATD<1> data output.
		1	I	ST	PORTD<1> data input; weak pull-up when RDPU bit is set.
	AD1(1)	x	O	DIG	External memory interface, Address/Data Bit 1 output.(2)
		x	I	TTL	External memory interface, Data Bit 1 input.(2)
	PSP1(1)	x	O	DIG	PSP read output data (LATD<1>); takes priority over port data.
		x	I	TTL	PSP write data input.
	ECCP3(3)	0	O	DIG	ECCP3 compare and PWM output; takes priority over port data.
		1	I	ST	ECCP3 capture input.
	P3A(3)	0	O	DIG	ECCP3 Enhanced PWM output, Channel A; takes priority over port and PSP data. May be configured for tri-state during Enhanced PWM shutdown events.
RD2/AD2/PSP2 (RD2/CCP4/ P3D)	RD2	0	O	DIG	LATD<2> data output.
		1	I	ST	PORTD<2> data input; weak pull-up when RDPU bit is set.
	AD2(1)	x	O	DIG	External memory interface, Address/Data Bit 2 output.(2)
		x	I	TTL	External memory interface, Data Bit 2 input.(2)
	PSP2(1)	x	O	DIG	PSP read output data (LATD<2>); takes priority over port data.
		x	I	TTL	PSP write data input.
	CCP4(3)	0	O	DIG	CCP4 compare output and PWM output; takes priority over port data.
		1	I	ST	CCP4 capture input.
	P3D(3)	0	O	DIG	ECCP3 Enhanced PWM output, Channel D; takes priority over port and PSP data. May be configured for tri-state during Enhanced PWM shutdown events.
RD3/AD3/ PSP3(1)	RD3(1)	0	O	DIG	LATD<3> data output.
		1	I	ST	PORTD<3> data input; weak pull-up when RDPU bit is set.
	AD3(1)	x	O	DIG	External memory interface, Address/Data Bit 3 output.(2)
		x	I	TTL	External memory interface, Data Bit 3 input.(2)
	PSP3(1)	x	O	DIG	PSP read output data (LATD<3>); takes priority over port data.
		x	I	TTL	PSP write data input.
RD4/AD4/ PSP4/SDO2(1)	RD4(1)	0	O	DIG	LATD<4> data output.
		1	I	ST	PORTD<4> data input; weak pull-up when RDPU bit is set.
	AD4(1)	x	O	DIG	External memory interface, Address/Data Bit 4 output.(2)
		x	I	TTL	External memory interface, Data Bit 4 input.(2)
	PSP4(1)	x	O	DIG	PSP read output data (LATD<4>); takes priority over port data.
		x	I	TTL	PSP write data input.
	SDO2(1)	0	O	DIG	SPI data output (MSSP2 module); takes priority over port data.

Legend: O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

Note 1: These features or port pins are implemented only on 100-pin devices.

2: External memory interface I/O takes priority over all other digital and PSP I/O.

3: These features are implemented on this pin only on 64-pin devices; for all other devices, they are multiplexed with RE6/RH7 (P1B), RG0 (ECCP3/P3A) or RG3 (CCP4/P3D).

TABLE 17-4: REGISTERS ASSOCIATED WITH PWM, TIMER2 AND TIMER4

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	69
RCON	IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	70
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	71
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	71
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	71
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CCP5IF	CCP4IF	CCP3IF	71
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CCP5IE	CCP4IE	CCP3IE	71
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CCP5IP	CCP4IP	CCP3IP	71
TRISG	TRISG7	TRISG6	TRISG5	TRISG4	TRISG3(1)	TRISG2	TRISG1	TRISG0	71
TMR2	Timer2 Register								70
PR2	Timer2 Period Register								70
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	70
TMR4	Timer4 Register								72
PR4	Timer4 Period Register								72
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0	72
CCPR4L	Capture/Compare/PWM Register 4 Low Byte								72
CCPR4H	Capture/Compare/PWM Register 4 High Byte								72
CCPR5L	Capture/Compare/PWM Register 5 Low Byte								73
CCPR5H	Capture/Compare/PWM Register 5 High Byte								73
CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0	73
CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0	73

Legend: — = unimplemented, read as '0'. Shaded cells are not used by PWM, Timer2 or Timer4.

Note 1: This bit is only available in 80-pin and 100-pin devices; otherwise, it is unimplemented and reads as '0'.

18.2 Capture and Compare Modes

Except for the operation of the Special Event Trigger discussed below, the Capture and Compare modes of the ECCPx modules are identical in operation to that of CCP4. These are discussed in detail in **Section 17.2 "Capture Mode"** and **Section 17.3 "Compare Mode"**.

18.2.1 SPECIAL EVENT TRIGGER

ECCP1 and ECCP2 incorporate an internal hardware trigger that is generated in Compare mode on a match between the CCPRx register pair and the selected timer. This can be used, in turn, to initiate an action. This mode is selected by setting CCPxCON<3:0> to '1011'.

The Special Event Trigger output of either ECCP1 or ECCP2 resets the TMR1 or TMR3 register pair, depending on which timer resource is currently selected. This allows the CCPRx register to effectively be a 16-Bit Programmable Period register for Timer1 or Timer3. In addition, the ECCP2 Special Event Trigger will also start an A/D conversion if the A/D module is enabled.

Special Event Triggers are not implemented for ECCP3, CCP4 or CCP5. Selecting the Special Event Trigger mode for these modules has the same effect as selecting the Compare with Software Interrupt mode (CCPxM<3:0> = 1010).

Note:	The Special Event Trigger from ECCP2 will not set the Timer1 or Timer3 interrupt flag bits.

18.3 Standard PWM Mode

When configured in Single Output mode, the ECCPx modules function identically to the standard CCPx modules in PWM mode, as described in **Section 17.4 "PWM Mode"**. Sometimes this is also referred to as "Compatible CCP" mode, as in Tables 18-1 through 18-3.

Note:	When setting up single output PWM operations, users are free to use either of the processes described in **Section 17.4.3 "Setup for PWM Operation"** or **Section 18.4.9 "Setup for PWM Operation"**. The latter is more generic but will work for either single or multi-output PWM.

19.2.2 SFRs AND THE ETHERNET MODULE

Like other peripherals, direct control of the Ethernet module is accomplished through a set of SFRs. Because of their large number, the majority of these registers is located in the bottom half of Bank 14 of the microcontroller's data memory space.

Five key SFRs for the Ethernet module are located in the microcontroller's regular SFR area in Bank 15, where fast access is possible. They are:

- ECON1
- EDATA
- EIR
- The Ethernet Buffer Read Pointer Pair (ERDPTH and ERDPTL)

ECON1 is described along with other Ethernet control registers in the following section. EDATA and ERDPTH:ERDPTL are the Ethernet Data Buffer registers and its pointers during read operations (see **Section 19.2.1 "Ethernet Buffer and Buffer Pointer Registers"**). EIR is part of the Ethernet interrupt structure and is described in **Section 19.3 "Ethernet Interrupts"**.

Many of the Ethernet SFRs in Bank 14 serve as pointer registers to indicate addresses within the dedicated Ethernet buffer for storage and retrieval of packet data. Others store information for packet pattern masks or checksum operations. Several are used for controlling overall module operations, as well as specific MAC and PHY functions.

19.2.3 ETHERNET CONTROL REGISTERS

The ECON1 register (Register 19-1) is used to control the main functions of the module. Receive enable, transmit request and DMA control bits are all located here. The ECON2 register (Register 19-2) is used to control other top level functions of the module. The ESTAT register (Register 19-3) is used to report the high-level status of the module and Ethernet communications.

The Ethernet SFRs with the 'E' prefix are always accessible, regardless of whether or not the module is enabled.

REGISTER 19-1: ECON1: ETHERNET CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
TXRST	RXRST	DMAST	CSUMEN	TXRTS	RXEN	—	—
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 **TXRST:** Transmit Logic Reset bit
1 = Transmit logic is held in Reset
0 = Normal operation

bit 6 **RXRST:** Receive Logic Reset bit
1 = Receive logic is held in Reset
0 = Normal operation

bit 5 **DMAST:** DMA Start and Busy Status bit
1 = DMA copy or checksum operation is in progress (set by software, cleared by hardware or software)
0 = DMA hardware is Idle

bit 4 **CSUMEN:** DMA Checksum Enable bit
1 = DMA hardware calculates checksums
0 = DMA hardware copies buffer memory

bit 3 **TXRTS:** Transmit Request to Send bit
1 = The transmit logic is attempting to transmit a packet (set by software, cleared by hardware or software)
0 = The transmit logic is Idle

bit 2 **RXEN:** Receive Enable bit
1 = Packets which pass the current filter configuration will be written into the receive buffer
0 = All packets received will be discarded by hardware

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

19.5.3 RECEIVING PACKETS

Assuming that the receive buffer has been initialized, the MAC has been properly configured and the receive filters have been configured, the application should perform these steps to receive Ethernet packets:

1. Set the PKTIE and ETHIE bits to generate an Ethernet interrupt whenever a packet is received (if desired).
2. Clear the RXERIF flag and set both RXERIE and ETHIE to generate an interrupt whenever a packet is dropped, due to insufficient buffer space or memory access bandwidth (if desired).
3. Enable reception by setting the RXEN bit (ECON1<2>).

After setting RXEN, the Duplex mode and the Receive Buffer Start and End Pointers should not be modified. Additionally, to prevent unexpected packets from arriving, it is recommended that RXEN be cleared before altering the receive filter configuration (ERXFCON) and MAC address.

After reception is enabled, packets which are not filtered out will be written into the circular receive buffer. Any packet which does not meet the necessary filter criteria will be discarded and the application will not have any means of identifying that a packet was thrown away. When a packet is accepted and completely written into the buffer:

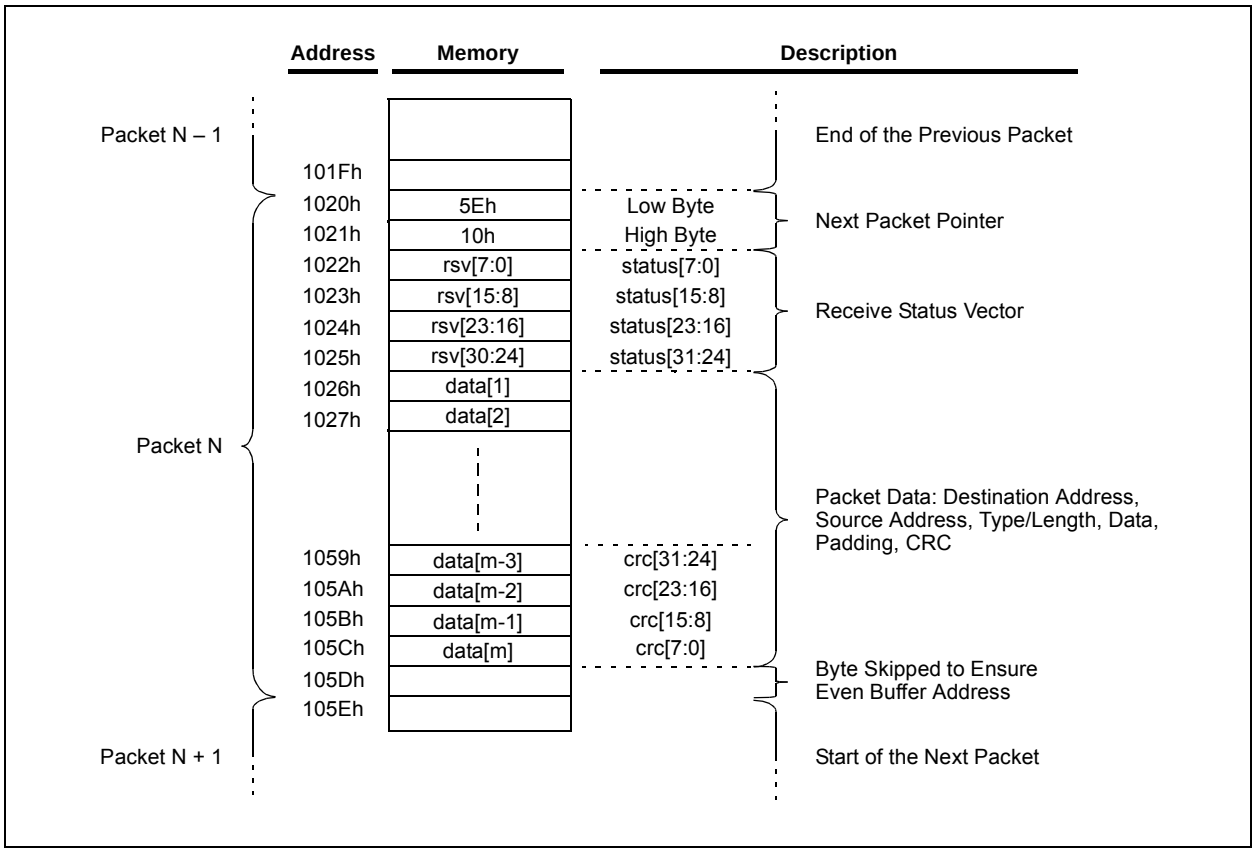
- The EPKTCNT register is incremented
- The PKTIF bit is set
- An interrupt is generated (if enabled)
- The Hardware Write Pointers, ERXWRPT, are automatically advanced

19.5.3.1 Receive Packet Layout

Figure 19-11 shows the layout of a received packet. The packets are preceded by a 6-byte header which contains a Next Packet Pointer in addition to a receive status vector which contains receive statistics, including the packet's size. The receive status vectors are shown in Table 19-5.

If the last byte in the packet ends on an odd value address, the hardware will automatically add a padding byte when advancing the Hardware Write Pointer. As such, all packets will start on an even boundary.

FIGURE 19-11: SAMPLE RECEIVE PACKET LAYOUT



19.5.3.3 Freeing Receive Buffer Space

After the user application has processed a packet (or part of the packet) and needs to free the occupied buffer space used by the processed data, it must advance the Receive Buffer Read Pointer pair, ERXRDPT. The module always writes up to, but not over, the memory pointed to by the ERXRDPT registers. If an attempt to overwrite the Receive Buffer Read Pointer location occurs, the packet in progress is aborted, the RXERIF flag is set and an interrupt is generated (if enabled). In this manner, the hardware will never overwrite unprocessed packets. Normally, the ERXRDPT pair is advanced close to a value pointed to by the Next Packet Pointer, which precedes the receive status vector for the current packet.

The Receive Buffer Read Pointer Low Byte (ERXRDPTL register) is internally buffered to prevent the pointer from moving when only one byte is updated. To move the ERXRDPT pair, the application must write to ERXRDPTL first. The write will update the internal buffer but will not affect the register. When the application writes to ERXRDPTH, the internally buffered low byte will be loaded into the ERXRDPTL register at the same time. The ERXRDPT bytes can be read in any order. When they are read, the actual value of the registers will be returned. As a result, the buffered low byte is not readable.

In addition to advancing the Receive Buffer Read Pointer, after each packet is fully processed, the application must set the PKTDEC bit (ECON2<6>). This causes the EPKTCNT register to decrement by 1. After decrementing, if EPKTCNT is '0', the PKTIF flag bit is automatically cleared. Otherwise, it remains set, indicating that additional packets are in the receive buffer and are waiting to be processed. Attempting to decrement EPKTCNT below 0 does not cause an underflow to 255, but may cause an unintentional interrupt. The application should avoid decrementing EPKTCNT in this situation.

Additionally, if the EPKTCNT register ever maximizes at 255, all new packets which are received will be aborted, even if buffer space is available. To indicate the error, the RXERIF is set and an interrupt is generated (if enabled). To prevent this condition, the user application must properly decrement the counter whenever a packet is processed.

Because only one pointer is available to control buffer area ownership, the application must process packets in the order they are received. If a packet is to be saved and processed later, the application should copy the packet to an unused location in memory. This can be done efficiently using the integrated DMA controller (see **Section 19.9 "Direct Memory Access Controller"**).

19.5.3.4 Receive Buffer Free Space

At any time the application needs to know how much receive buffer space is remaining, it should read the Hardware Write Pointers (ERXWRPT registers) and compare it with the ERXRDPT registers. Combined with the known size of the receive buffer, the free space can be derived.

Note:	The ERXWRPT registers only update when a packet has been successfully received. If the application reads it just before another packet is to be successfully completed, the value returned could be stale and off by the maximum frame length permitted (MAMXFLH:MAMXFLL) plus 8. Furthermore, as the application reads one byte of the ERXWRPT registers, a new packet may arrive and update the 13-bit pointer before the application has an opportunity to read the other byte of the ERXWRPT registers.

When reading the ERXWRPT registers with the receive hardware enabled, special care must be taken to ensure the low and high bytes are read as a matching set.

To be assured that a matching set is obtained:

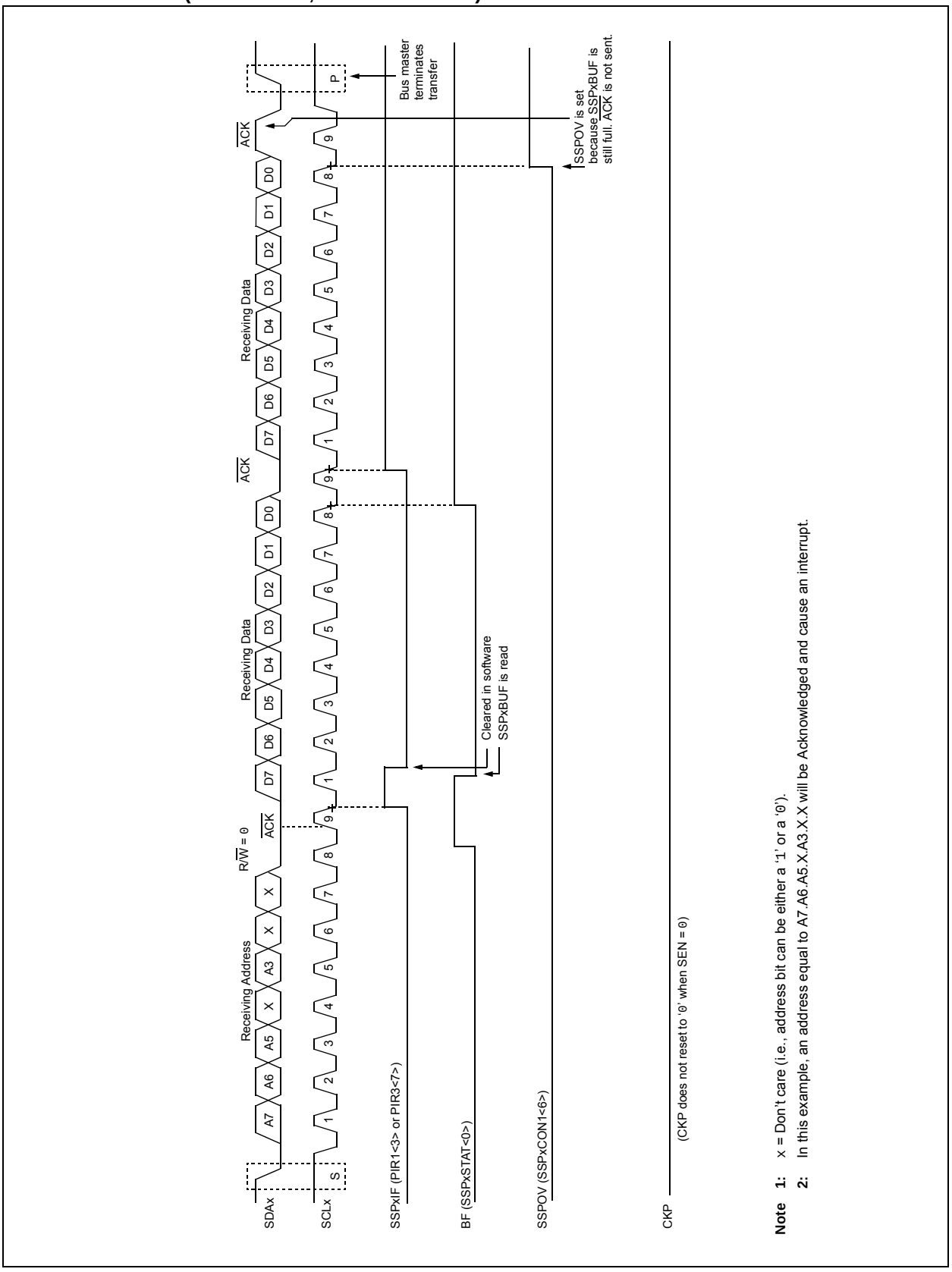
1. Read the EPKTCNT register and save its contents.
2. Read ERXWRPTL and ERXWRPTH.
3. Read the EPKTCNT register again.
4. Compare the two packet counts. If they are not the same, go back to Step 2.

With the Hardware Write Pointers obtained, the free space can be calculated as shown in Equation 19-2. The hardware prohibits moving the Write Pointer to the same value occupied by the ERXRDPT registers, so at least one byte will always go unused in the buffer. The Equation 19-2 calculation reflects the lost byte.

EQUATION 19-2: RECEIVE BUFFER FREE SPACE CALCULATION

```
If ERXWRPT > ERXRDPT, then
        Free Space = (ERXND – ERXST) – (ERXWRPT – ERXRDPT)
else:
if ERXWRPT = ERXRDPT, then
        Free Space = (ERXND – ERXST)
else:
        Free Space = ERXRDPT – ERXWRPT – 1
```

FIGURE 20-9: I²C™ SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01011 (RECEPTION, 7-BIT ADDRESS)



Note 1: x = Don't care (i.e., address bit can be either a '1' or a '0').

2: In this example, an address equal to A7.A6.A5.X.A3.X.X will be Acknowledged and cause an interrupt.

25.4 Two-Speed Start-up

The Two-Speed Start-up feature helps to minimize the latency period, from oscillator start-up to code execution, by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is HS or HSPLL (Crystal-Based) modes. Since the EC and ECPLL modes do not require an Oscillator Start-up Timer delay, Two-Speed Start-up should be disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI_RUN mode.

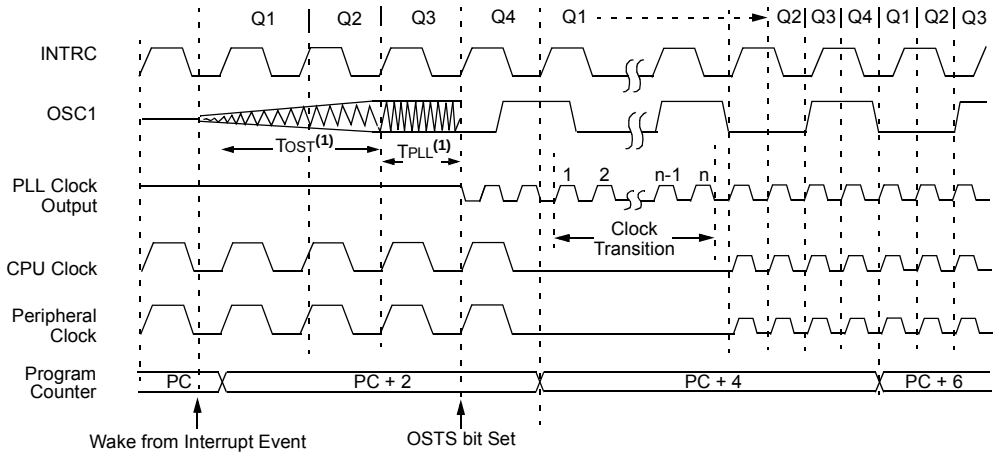
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

25.4.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTRC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including serial SLEEP instructions (refer to **Section 4.1.4 “Multiple Sleep Commands”**). In practice, this means that user code can change the SCS<1:0> bit settings, or issue SLEEP instructions, before the OST times out. This would allow an application to briefly wake-up, perform routine “housekeeping” tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

FIGURE 25-3: TIMING TRANSITION FOR TWO-SPEED START-UP (INTRC TO HSPLL)



Note 1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

BTG **Bit Toggle f**

Syntax: BTG f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff

Description: Bit 'b' in data memory location 'f' is inverted.

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 26.2.3 "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: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

BOV **Branch if Overflow**

Syntax: BOV n

Operands: $-128 \le n \le 127$

Operation: if Overflow bit is '1',
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn

Description: If the Overflow bit is '1', 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 two-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    BOV  Jump
```
Before Instruction
PC = address (HERE)
After Instruction
If Overflow = 1;
PC = address (Jump)
If Overflow = 0;
PC = address (HERE + 2)

INCFSZ **Increment f, Skip if 0**

Syntax: INCFSZ 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:

0011	11da	ffff	ffff

Description: The contents of register 'f' are incremented. 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 26.2.3 "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    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

INFSNZ **Increment f, Skip if Not 0**

Syntax: INFSNZ 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	10da	ffff	ffff

Description: The contents of register 'f' are incremented. 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 26.2.3 "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    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

26.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set for the PIC18F97J60 family. This includes the MPLAB C18 C Compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

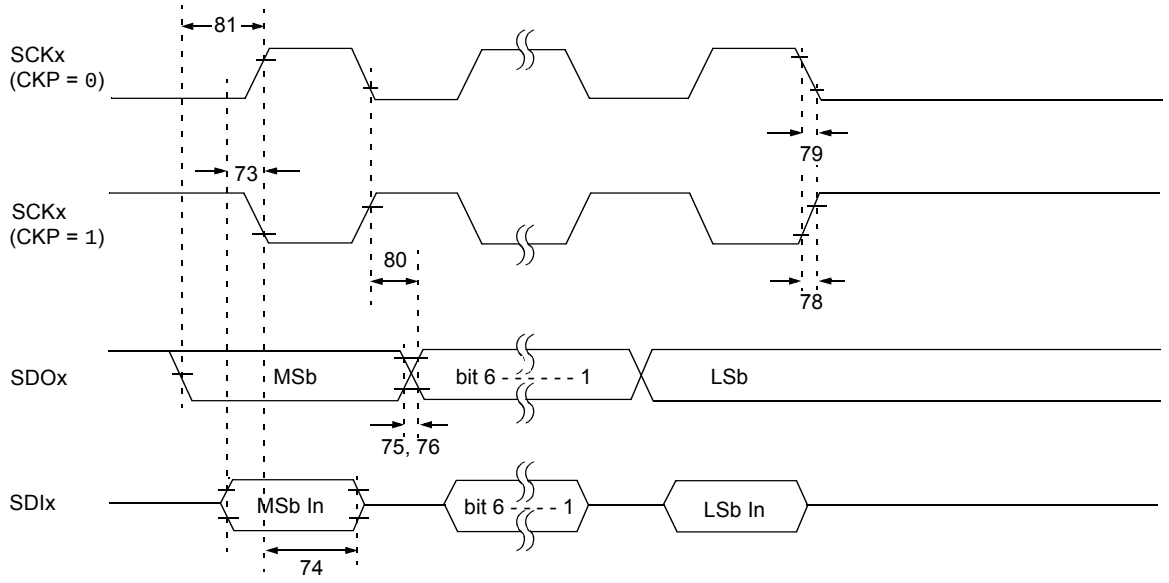
When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option or dialog box within the environment that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

FIGURE 28-12: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)



Note: Refer to Figure 28-3 for load conditions.

TABLE 28-17: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
73	TDIV2SCH, TDIV2SCL	Setup Time of SDIx Data Input to SCKx Edge	100	—	ns	
74	TSCH2DIL, TSCL2DIL	Hold Time of SDIx Data Input to SCKx Edge	100	—	ns	
75	TDOR	SDOx Data Output Rise Time	—	25	ns	
76	TDOF	SDOx Data Output Fall Time	—	25	ns	
78	TSCR	SCKx Output Rise Time	—	25	ns	
79	TSCF	SCKx Output Fall Time	—	25	ns	
80	TSCH2DOV, TSCL2DOV	SDOx Data Output Valid after SCKx Edge	—	50	ns	
81	TDOV2SCH, TDOV2SCL	SDOx Data Output Setup to SCKx Edge	TCY	—	ns	