

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 "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

E·XF

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	21
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 10x10b
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/pic18f25j10t-i-ml

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F24J10 PIC18LF24J10
- PIC18F25J10 PIC18LF25J10
- PIC18F44J10 PIC18LF44J10
- PIC18F45J10 PIC18LF45J10

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price. The PIC18F45J10 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 Core Features

1.1.1 LOW POWER

All of the devices in the PIC18F45J10 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- Alternate Run Modes: By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-Fly Mode Switching:** The powermanaged modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- Low Consumption in Key Modules: The power requirements for both Timer1 and the Watchdog Timer are minimized. See Section 24.0 "Electrical Characteristics" for values.

1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F45J10 family offer three different oscillator options. These include:

- Two Crystal modes, using crystals or ceramic resonators
- Two External Clock modes
- INTRC source (approximately 31 kHz)

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- Fail-Safe Clock Monitor: This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

FIGURE 1-1: PIC18F24J10/25J10 (28-PIN) BLOCK DIAGRAM



3.4 PLL Frequency Multiplier

A Phase Locked Loop (PLL) circuit is provided as an option for users who want to use a lower frequency oscillator circuit, or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals, or users who require higher clock speeds from an internal oscillator. For these reasons, the HSPLL and ECPLL modes are available.

The HSPLL and ECPLL modes provide the ability to selectively run the device at 4 times the external oscillating source to produce frequencies up to 40 MHz. The PLL is enabled by setting the PLLEN bit in the OSCTUNE register (Register 3-1).

FIGURE 3-4: PLL BLOCK DIAGRAM



REGISTER 3-1: OSCTUNE: PLL CONTROL REGISTER

U-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
—	PLLEN ⁽¹⁾	—	—	—	—	—	—
bit 7							bit 0

Legend:					
R = Readable bit	W = Writable bit	U = Unimplemented 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	Unimplemented: Read as '0'
bit 6	PLLEN: Frequency Multiplier PLL Enable bit ⁽¹⁾
	1 = PLL enabled

0 = PLL disabled

- bit 5-0 Unimplemented: Read as '0'
- Note 1: Available only for ECPLL and HSPLL oscillator configurations; otherwise, this bit is unavailable and reads as '0'.

TOSU — — — Top-of-Stack Upper Byte (TOS-20:16>) 0 0000 47.53 TOSH Top-of-Stack LOW Byte (TOS-7:58>) 0000 0000 47.53 STKPTR STKVINF — Return Stack Pointer 00-0000 47.53 PCLATU — — Holding Register for PC-415.8> 0000 0000 47.53 PCLATU — — Holding Register for PC-415.8> 0000 0000 47.53 PCLATU — — Ibit 1 Program Memory Table Pointer High Byte (TBLPTR<15.8) 0000 0000 47.74 TBLPTRU Program Memory Table Pointer Low Byte (TBLPTR<15.8) 0000 0000 47.74 TBLATEN Program Memory Table Pointer Low Byte (TBLPTR<20-) 0000 0000 47.74 TBLATEN Program Memory Table Pointer Low Byte (TBLPTR 0000 0000 47.74 TBLATEN Program Memory Table Pointer Low Byte (TBLPTR<20-) 0000 0000 47.74 TBLATEN Program Memory Table Pointer Low Byte (TBLPTRC10-) 0000 0000	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
TOSH Top-of-Stack High Byte (TOS 0000 0000 47, 53 TOSL Top-of-Stack Low Byte (TOS 0000 0000 47, 53 STKPTR STKPLU	TOSU	_	—	_	Top-of-Stack	Upper Byte (T	OS<20:16>)			0 0000	47, 53
TOSL Top-of-Stack Low Byte (TOS-7:0> 0000 0000 47, 53 STKPTR STKPUL STKUHF — Return Stack Pointer 00-0 0000 47, 53 PCLATU — — Holding Register for PC<50:16> 0000 0000 47, 53 PCLATH Holding Register for PC<15:8> 0000 0000 47, 53 PCLATU — — Ibit 21 Program Memory Table Pointer High Byte (TBLPTR<15:8>) 0000 0000 47, 74 TBLPTRU Program Memory Table Pointer Low Byte (TBLPTR<7:0>) 0000 0000 47, 74 TBLATA Program Memory Table Pointer Low Byte (TBLPTR<7:0>) 0000 0000 47, 74 TROCN GE/GIEH PECIALU xxxx xxxxx 47, 81 PRODH Product Register Low Byte xxxx xxxxx 47, 81 INTCON GE/GIEH PECIALU INTEDG1 INTEDG2 — TMROIP INTIP RBIP 1000 0000 47, 67 NDFD Uses contents of FSR0 to address data memory - value of FSR0 notanged (not a physical	TOSH	Top-of-Stack	High Byte (TO	S<15:8>)	•					0000 0000	47, 53
STKFUR STKFUL STKFUL STKFUL STKFUL O 0.00 0.00 47, 54 PCLATH Holding Register for PC<15.8→	TOSL	Top-of-Stack	Low Byte (TO	S<7:0>)						0000 0000	47, 53
PCLATH — Holding Register for PC<20:16> 0 0.000 47, 53 PCLAPH Holding Register for PC<15.3> 0000 0000 47, 53 PCL PC Low Byte (PC<7:0>) 0000 0000 47, 53 TBLPTRI — it Program Memory Table Pointer Low Byte (TBLPTR<15:8>) 0000 0000 47, 74 TBLPTRI Program Memory Table Pointer Low Byte (TBLPTR<7:0>) 0000 0000 47, 74 TABLAT Program Memory Table Calct- Viscon Byte (TBLPTR<7:0-) 0000 0000 47, 74 PRODL Product Register High Byte TMROIE INTOIF RBIF 0000 0000 47, 74 NTCON GIE/GIEL PEI/GIEL TMROIE INTOIDE INTOID RBIF 1111 1-1 47, 85 INTCON2 RBFDU INTEPC0 INTEPC0 INTEPC1 INT	STKPTR	STKFUL	STKUNF	—	Return Stack	Pointer				00-0 0000	47, 54
PCLATH Holding Register for PC<15:8> 0000 047, 53 PCL PC Low Byte (PC<7:0>' 0000 0000 47, 53 SILPTRM Program Memory Table Pointer High Byte (TBLPTR<15:8>' 0000 0000 47, 74 TBLPTRH Program Memory Table Pointer Low Byte (TBLPTR<7:0>' 0000 0000 47, 74 TBLPTRH Program Memory Table Pointer Low Byte (TBLPTR<7:0>' 0000 0000 47, 74 PRODH Product Register Low Byte * × <t< td=""><td>PCLATU</td><td>—</td><td>_</td><td>_</td><td>Holding Regi</td><td>ster for PC<20</td><td>):16></td><td></td><td></td><td>0 0000</td><td>47, 53</td></t<>	PCLATU	—	_	_	Holding Regi	ster for PC<20):16>			0 0000	47, 53
PCL PC Low Byte (PC-7:0> 0000	PCLATH	Holding Regis	ster for PC<15	:8>						0000 0000	47, 53
TBL PTRU - bl 21 Program Memory Table Pointer Ugb Pyte (TBL PTR < 20:16>) 0000000 47, 74 TBL PTRL Program Memory Table Pointer Ligh Byte (TBL PTR < 15:8)	PCL	PC Low Byte	(PC<7:0>)							0000 0000	47, 53
TBLPTRH Program Memory Table Pointer Ligh Byte (TBLPTR<15.8>) 0000 0000 47, 74 TBLPTRL Program Memory Table Pointer Low Byte (TBLPTR<15.8>) 0000 0000 47, 74 TBLDTR Program Memory Table Pointer Low Byte (TBLPTR<15.8>) 0000 0000 47, 74 PRODH Product Register Ligh Byte xxxx xxxx 47, 81 INTCON GIF/GIEH PEIE/GIEL TMR0IE INT0IE RBPU INT1FDC0 INT2D0 INTEDC0 INTEDC1 INTEDC2 — TMR0IF RINT0IF RBIP 1011.1 -1 47, 85 INTCON3 INT2IP INT1FD Q INTEDC3 INTEDC2 — TMR0IF INT2IF INT1IF 1.0 -0 47, 67 POSTINC0 Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register) N/A 47, 67 POSTDEC0 Uses contents of FSR0 to address data memory - value of FSR0 opst-incremented (not a physical register) N/A 47, 67 PUSW0 Uses contents of FSR0 to address data memory - value of FSR0 opst-incremented (not a physical register) N/A 47, 67 PUSW0 Uses contents of FSR0 to address data memory - value of FSR0 opst-incremented (not a physical register) N/A 47, 6	TBLPTRU	—	—	bit 21	Program Mer	nory Table Po	inter Upper By	te (TBLPTR<2	20:16>)	00 0000	47, 74
TBLPTRL Program Memory Table Pointer Low Byte (TBLPTR<7:0>) 0000 0000 47, 74 TABLAT Program Memory Table Lath 0000 0000 47, 74 PRODH Product Register Low Byte xxxx xxxxx 47, 81 PRODL Product Register Low Byte xxxx xxxxx 47, 81 INTCON GIE/GIEH PEIC/GIEL TMR0IF INTICP RBIF 0000 0000 47, 74 INTCON GIE/GIEH PEIC/GIEL TMR0IF INTICP RBIF 0000 0000 47, 85 INTCON GIE/GIEH PEIC/GIEL TMR0IF INTICP RBIF 1111 -1 47, 86 INTCON Uses contents of FSR0 to address data memory - value of FSR0 post-decremented (not a physical register) N/A 47, 67 POSTINCO Uses contents of FSR0 to address data memory - value of FSR0 post-decremented (not a physical register) N/A 47, 67 PLUSW0 Value of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register) N/A 47, 67 PLUSW0 Value of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register) N/A 47, 67 PSR0L	TBLPTRH	Program Mer	nory Table Poi	nter High Byte	e (TBLPTR<15	5:8>)				0000 0000	47, 74
TABLAT Program Memory Table Latch 0000 47, 74 PRODI Product Register High Byte xxxx xxxx 47, 81 PRODL Product Register Low Byte xxxx xxxx 47, 81 INTCON GE/GEH PEIC/GEL TMR0IE INTDIE RBIE TMR0IF INT0IF RBIP 1111 -1-1 47, 86 INTCON2 RBPU INTEDG0 INTEDG1 INTEDC2 — TMR0IP — RBIP 1111 -1-1 47, 86 INTCON3 INT2IP INT1P — INT2IE INT1IE _ INT0IF RBIP 1111 -1-1 47, 86 INTCON3 Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register) N/A 47, 67 POSTINC0 Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory - value of FSR1 post-incremented (not a physical register) N/A 47, 67 PSR0L1 Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 PLUSW0 Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a p	TBLPTRL	Program Mer	nory Table Poi	nter Low Byte	(TBLPTR<7:0)>)				0000 0000	47, 74
PRODH Product Register High Byte xxxx 47, 81 PRODL Product Register Low Byte xxxx 47, 81 INTCON GIE/GIEN PIEIGAIEL TMROIE INTOE RBI TMROIF INTOIF RBI 111 1 -1.1 47, 85 INTCON3 INT2IP INT1P — INT0E RBI — INT0IF RBIP 1111 1 -1.1 47, 85 INTCON3 INT2IP INT1IP — INT2IE INT1IE — INT2IF INT1IF 11-0 0-00 47, 85 INDF0 Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) N/A 47, 67 POSTIDC0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 FSR0H — — — Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 VREG Working Register Vorking Register Vorking Register N/A 47, 67 INDF1	TABLAT	Program Mer	nory Table Lat	ch						0000 0000	47, 74
PRODL Product Register Low Byter TMROIF INTCON GIE/GIEH PEIE/GIEL TMROIF INTOIF RBIF 0.000 0.000 47, 85 INTCON2 RBFU INTEDG0 INTEDG1 INTEDG2 — TMROIP — RBIF 1111 -1 47, 85 INTCON3 INT2IP INTEDG0 INTEDG1 INTEDG2 — TMROIP — RBIF 1111 -1 47, 87 INTCON3 INT2IP INT1P INT1P INT1P INT1P N/A 47, 67 POSTINCO Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 FSR0L Indirect Data Memory Address Pointer 0 Low Byte	PRODH	Product Regi	ster High Byte							xxxx xxxx	47, 81
INTCON GIE/GIEH PEIE/GIEL TMR0IE INTOIC RBIE TMR0IF INTOIF RBIF 0000 000x 47, 85 INTCON2 RBPU INTEDG0 INTEDG0 INTEDG2 — TMR0IF — RBIP 1111 - 1-1 47, 85 INTCON3 INT2IP INT1IP — INT2IE INT1IE — INT2IF INT1IF 11.0 0 -00 47, 87 INDF0 Uses contents of FSR0 to address data memory - value of FSR0 post-decremented (not a physical register) N/A 47, 67 POSTIDC0 Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register) N/A 47, 67 FSR0H — —	PRODL	Product Regi	ster Low Byte							xxxx xxxx	47, 81
INTCON2 RBPU INTEDG0 INTEDG1 INTEDG2 — TMR0IP — RBIP 1111 -1-1 47, 86 INTCON3 INT2IP INT1IP — INT2IE INT1IF — INT2IF INT1IF 11-0 0-00 47, 87 INDF0 Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) N/A 47, 67 POSTINC0 Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) N/A 47, 67 POSTDEC0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 FSR0H — — Indirect Data Memory Address Pointer 0 Low Byte xxxxx xxxx 47, 67 VBEG Working Register xxxx xxxx 47, 67 POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 POSTINC1 Uses contents of FSR1 to address data memor	INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	47, 85
INTCON3 INT2IP INT1IP INT2IE INT1IE INT1IE INT2IF INT1IF 11-0 0-00 47, 87 INDF0 Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) N/A 47, 67 POSTIDEC0 Uses contents of FSR0 to address data memory – value of FSR0 post-faceremented (not a physical register) N/A 47, 67 POSTIDEC0 Uses contents of FSR0 to address data memory – value of FSR0 post-faceremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PSR0H —	INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	_	TMR0IP	_	RBIP	1111 -1-1	47, 86
INDF0 Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) N/A 47, 67 POSTINC0 Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) N/A 47, 67 POSTDC0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PREINC0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PREINC0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 FSR0H — — — Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 VBREG Working Register xxxx xxxx 47, 67 xxxx xxxx 47, 67 POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 prost-incremented (not a physical register) N/A 47, 67 POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 prost-incremented (not a physical register) N/A 47, 67 POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 prost-incremented (not a physical register) N/A 47, 67 POSTINC1 Uses conten	INTCON3	INT2IP	INT1IP	_	INT2IE	INT1IE	_	INT2IF	INT1IF	11-0 0-00	47, 87
POSTINC0 Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) N/A 47, 67 POSTDEC0 Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) N/A 47, 67 PREINC0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 FSR0H — — — Indirect Data Memory Address Pointer 0 Low Byte	INDF0	Uses content	s of FSR0 to a	iddress data n	nemory – valu	e of FSR0 not	changed (not	a physical reg	ister)	N/A	47, 67
POSTDEC0 Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) N/A 47, 67 PREINC0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 FSR0H — — — Indirect Data Memory Address Pointer 0 Low Byte xxxxx 47, 67 FSR0L Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 WREG Working Register xxxx xxxx 47, 67 INDF1 Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) N/A 47, 67 POSTIDC1 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 POSTDEC1 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) <t< td=""><td>POSTINC0</td><td>Uses content</td><td>s of FSR0 to a</td><td>iddress data n</td><td>nemory – valu</td><td>e of FSR0 pos</td><td>t-incremented</td><td>(not a physica</td><td>al register)</td><td>N/A</td><td>47, 67</td></t<>	POSTINC0	Uses content	s of FSR0 to a	iddress data n	nemory – valu	e of FSR0 pos	t-incremented	(not a physica	al register)	N/A	47, 67
PREINCO Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) N/A 47, 67 PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W N/A 47, 67 FSR0H — — Indirect Data Memory Address Pointer 0 Low Byte	POSTDEC0	Uses content	s of FSR0 to a	iddress data n	nemory – valu	e of FSR0 pos	t-decremented	l (not a physic	al register)	N/A	47, 67
PLUSW0 Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W N/A 47, 67 FSR0H — — — Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 FSR0L Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 WREG Working Register xxxx xxxx 47, 67 INDF1 Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) N/A 47, 67 POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 POSTDEC1 Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PLUSW1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 FSR14 — — — Indirect Data Memory Address Pointer 1 Low Byte	PREINC0	Uses content	s of FSR0 to a	iddress data n	nemory – valu	e of FSR0 pre	-incremented (not a physical	register)	N/A	47, 67
FSR0H — — Indirect Data Memory Address Pointer 0 Low Byte xxxx 47, 67 FSR0L Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 WREG Working Register xxxx xxxx 47, 67 INDF1 Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) N/A 47, 67 POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 POSTDEC1 Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PLUSV1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 FSR1H — — — Indirect Data Memory Address Pointer 1 High Byte xxxx 47, 67 FSR1L Indirect Data Memory Address Pointer 1 Low Byte	PLUSW0	Uses content value of FSR	s of FSR0 to a 0 offset by W	iddress data n	nemory – value	e of FSR0 pre	-incremented (not a physical	register) –	N/A	47, 67
FSR0L Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 WREG Working Register xxxx xxxx 47 INDF1 Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register) N/A 47, 67 POSTINC1 Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a physical register) N/A 47, 67 POSTDEC1 Uses contents of FSR1 to address data memory - value of FSR1 post-decremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PLUSW1 Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register) N/A 47, 67 FSR1H — — — Indirect Data Memory Address Pointer 1 High Byte xxxx 47, 67 FSR1 Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 44, 67 FSR1 Indirect Data Memory Address Pointer 0 Low Byte xxxx xxxx 47, 67 FSR1 Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 FSR1 Indirect Data Memory Address Pointer 1 High Byte	FSR0H	—	—		—	Indirect Data	Memory Addr	ess Pointer 0 I	High Byte	xxxx	47, 67
WREG Working Register xxxx xxx	FSR0L	Indirect Data	Memory Addr	ess Pointer 0 I	Low Byte					xxxx xxxx	47, 67
INDF1 Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) N/A 47, 67 POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 POSTDEC1 Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PLUSW1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 FSR1H — — — Indirect Data Memory Address Pointer 1 High Byte xxxx 47, 67 FSR1L Indirect Data Memory Address Pointer 1 Low Byte Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 BSR — — — Bank Select Register 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 POSTINC2	WREG	Working Reg	ister							xxxx xxxx	47
POSTINC1 Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) N/A 47, 67 POSTDEC1 Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PLUSW1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 FSR1H — — — Indirect Data Memory Address Pointer 1 High Byte xxxxx 47, 67 FSR1L Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 88 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67	INDF1	Uses content	s of FSR1 to a	iddress data n	nemory – valu	e of FSR1 not	changed (not	a physical reg	ister)	N/A	47, 67
POSTDEC1 Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) N/A 47, 67 PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PLUSW1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W N/A 47, 67 FSR1H — — — Indirect Data Memory Address Pointer 1 High Byte xxxx 47, 67 FSR1L Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 BSR — — — Bank Select Register 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to add	POSTINC1	Uses content	s of FSR1 to a	iddress data n	nemory – value	e of FSR1 pos	t-incremented	(not a physica	al register)	N/A	47, 67
PREINC1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) N/A 47, 67 PLUSW1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W N/A 47, 67 FSR1H — — — Indirect Data Memory Address Pointer 1 High Byte xxxx 47, 67 FSR1L Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 47, 67 BSR — — — Bank Select Register 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents o	POSTDEC1	Uses content	s of FSR1 to a	iddress data n	nemory – value	e of FSR1 pos	t-decremented	l (not a physic	al register)	N/A	47, 67
PLUSW1 Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – N/A 47, 67 FSR1H — — — Indirect Data Memory Address Pointer 1 High Byte xxxxx 47, 67 FSR1L Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 BSR — — — Bark Select Register 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre	PREINC1	Uses content	s of FSR1 to a	iddress data n	nemory – value	e of FSR1 pre-	-incremented (not a physical	register)	N/A	47, 67
FSR1H - - - Indirect Data Memory Address Pointer 1 High Byte xxxxx 47, 67 FSR1L Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxx 47, 67 BSR - - - Bank Select Register 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory - value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory - value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register) N/A 48, 67 FSR2H - - - - Indirect Data Memory Address Pointer 2 High Byte xxxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxxx 48, 67 Xxxx xxxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte<	PLUSW1	Uses content value of FSR	s of FSR1 to a 1 offset by W	iddress data n	nemory – value	e of FSR1 pre	-incremented (not a physical	register) –	N/A	47, 67
FSR1L Indirect Data Memory Address Pointer 1 Low Byte xxxx xxxxx 47, 67 BSR — — — Bank Select Register 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 FSR2H — — — — Indirect Data Memory Address Pointer 2 High Byte xxxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxxx 48, 67 48, 67 STATUS — — N OV Z DC C	FSR1H	—	—		—	Indirect Data	Memory Addr	ess Pointer 1 I	High Byte	xxxx	47, 67
BSR — — — Bank Select Register 0000 47, 58 INDF2 Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 FSR2H — — — Indirect Data Memory Address Pointer 2 High Byte xxxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxxx 48, 67 xxxx xxxxx 48, 67 STATUS — — — N OV Z DC C xxxxxx 48, 67	FSR1L	Indirect Data	Memory Addr	ess Pointer 1 I	Low Byte					xxxx xxxx	47, 67
INDF2 Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) N/A 48, 67 POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – N/A 48, 67 FSR2H — — — Indirect Data Memory Address Pointer 2 High Byte xxxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxx 48, 67 xxxx xxxx 48, 67 STATUS — — N OV Z DC C xxxxx 48, 65	BSR	—	—	—	—	Bank Select	Register			0000	47, 58
POSTINC2 Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) N/A 48, 67 POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 FSR2H — — — Indirect Data Memory Address Pointer 2 High Byte xxxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxx 48, 67 Xxxx xxxx 48, 67 STATUS — — N OV Z DC C xxxxx 48, 65	INDF2	Uses content	s of FSR2 to a	iddress data n	nemory – value	e of FSR2 not	changed (not	a physical reg	ister)	N/A	48, 67
POSTDEC2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 67 PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 FSR2H — — — Indirect Data Memory Address Pointer 2 High Byte xxxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxxx 48, 67 STATUS — — N OV Z DC C x xxxxx 48, 65	POSTINC2	Uses content	s of FSR2 to a	iddress data n	nemory – value	e of FSR2 pos	t-incremented	(not a physica	al register)	N/A	48, 67
PREINC2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N/A 48, 67 PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W N/A 48, 67 FSR2H — — — Indirect Data Memory Address Pointer 2 High Byte xxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxx 48, 67 STATUS — — N OV Z DC C x xxxx 48, 65	POSTDEC2	2 Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) N/A 48, 6								48, 67	
PLUSW2 Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – N/A 48, 67 FSR2H — — — Indirect Data Memory Address Pointer 2 High Byte xxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxx 48, 67 STATUS — — N OV Z DC C xxxx 48, 65	PREINC2	Uses content	Jses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)						N/A	48, 67	
FSR2H — — — Indirect Data Memory Address Pointer 2 High Byte xxxx 48, 67 FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx x48, 67 STATUS — — N OV Z DC C	PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – N/A 48, 67 value of FSR2 offset by W								48, 67	
FSR2L Indirect Data Memory Address Pointer 2 Low Byte xxxx xxxx 48, 67 STATUS — — N OV Z DC C	FSR2H	_	—	—		Indirect Data	Memory Addr	ess Pointer 2 I	High Byte	xxxx	48, 67
STATUS – – – N OV Z DC Cx xxxx 48,65	FSR2L	Indirect Data	Memory Addr	ess Pointer 2 I	Low Byte	•				xxxx xxxx	48, 67
	STATUS	—	—	—	N	OV	Z	DC	С	x xxxx	48, 65

TABLE 6-3: REGISTER FILE SUMMARY (PIC18F24J10/25J10/44J10/45J10)

These registers and/or bits are not implemented on 28-pin devices and are read as '0'. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as '-'.

Alternate names and definitions for these bits when the MSSP module is operating in I²C™ Slave mode. See Section 16.4.3.2 "Address Masking" for details.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	50
LATC	PORTC Data Latch Register (Read and Write to Data Latch)								50
TRISC	PORTC Data Direction Control Register								50

TABLE 10-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

10.5 PORTD, TRISD and LATD Registers

Note:	PORTD	is only		available	in	40/44-pin
	devices.					

PORTD is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISD. Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISD bit (= 0) will make the corresponding PORTD pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATD) is also memory mapped. Read-modify-write operations on the LATD register read and write the latched output value for PORTD.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Three of the PORTD pins are multiplexed with outputs P1B, P1C and P1D of the Enhanced CCP module. The operation of these additional PWM output pins is covered in greater detail in Section 15.0 "Enhanced Capture/Compare/PWM (ECCP) Module".

Note: On a Power-on Reset, these pins are configured as digital inputs.

PORTD can also be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. See **Section 10.7 "Parallel Slave Port**" for additional information on the Parallel Slave Port (PSP).

Note:	When the Enhanced PWM mode is used						
	with either dual or quad outputs, the PSP						
	functions of PORTD are automatically						
	disabled.						

EXAMPLE 10-5: INITIALIZING PORTD

CLRF	PORTD	; Initialize PORTD by ; clearing output
CLRF	LATD	; data latches ; Alternate method ; to clear output
MOVLW	0CFh	; data latches ; Value used to ; initialize data
MOVWF	TRISD	; direction ; Set RD<3:0> as inputs ; RD<5:4> as outputs ; RD<7:6> as inputs

15.0 ENHANCED CAPTURE/ COMPARE/PWM (ECCP) MODULE

Note:	The ECCP module is implemented only in
	40/44-pin devices.

In PIC18F44J10/45J10 devices, ECCP1 is implemented as a standard CCP module with Enhanced PWM capabilities. These include the provisions for 2 or 4 output channels, user-selectable polarity, dead-band control and automatic shutdown

and restart. The Enhanced features are discussed in detail in **Section 15.4 "Enhanced PWM Mode**". Capture, Compare and single output PWM functions of the ECCP module are the same as described for the standard CCP module.

The control register for the Enhanced CCP module is shown in Register 15-1. It differs from the CCP1CON register in PIC18F24J10/25J10 devices in that the two Most Significant bits are implemented to control PWM functionality.

REGISTER 15-1: CCP1CON: ECCP1 CONTROL REGISTER (40/44-PIN DEVICES)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

Legend:	l egend:								
R = Readable	bit	W = Writable bit	U = Unimplemented bit, read	d as '0'					
-n = Value at F	POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown					
bit 7-6	P1M<1:0>:	Enhanced PWM Output Config	uration bits						
	$\frac{\text{If CCP1M<3}}{\text{vv} = P1A as}$	<u>:2> = 00, 01, 10:</u> signed as Capture/Compare ir	nut/output: P1B_P1C_P1D_a	ssigned as port nins					
	If CCP1M<3	:2> = 11:		ssigned as port pins					
	00 = Single	output: P1A modulated; P1B, F	P1C, P1D assigned as port pi	ns					
	01 = Full-bri	dge output forward: P1D modu	lated; P1A active; P1B, P1C	inactive					
	10 = Half-brid11 = Full-brid	dge output: P1A, P1B modulat	ed with dead-band control; P1 lated; P1C active; P1A, P1D i	inactive					
bit 5-4	DC1B<1:0>:	: PWM Duty Cycle bit 1 and bit	0						
	Capture mod Unused.	<u>le:</u>							
	Compare mo	ode:							
	Unused.								
	PWM mode:	re the two I Che of the 10 bit D	MA duty avala The sight MC	as of the duty quals are found					
	in CCPR1L.	re the two LSbs of the 10-bit Pr	www.auty.cycle. The eight wist	is of the duty cycle are found					
bit 3-0	CCP1M<3:0	>: CCP1 Module Mode Select	bits						
	0000 = Cap	oture/Compare/PWM off (resets	s ECCP module)						
	0001 = Res	erved	match						
	0010 - Cor	oture mode	naton						
	0100 = Cap	oture mode, every falling edge							
	0101 = Cap	ture mode, every rising edge							
	0110 = Cap 0111 = Cap	oture mode, every 4th hsing ea	ge dae						
	1000 = Con	npare mode, initialize CCP1 pi	n low, set output on compare	match (set CCP1IF)					
	1001 = Con	npare mode, initialize CCP1 pi	n high, clear output on compa	re match (set CCP1IF)					
	1010 = Con	npare mode, generate software	e interrupt only, CCP1 pin reve ant (ECCP resets TMR1, sets	erts to I/O state					
	1100 = PW	M mode; P1A, P1C active-high	; P1B, P1D active-high						
	1101 = PW	M mode; P1A, P1C active-high	; P1B, P1D active-low						
	1110 = PW	M mode; P1A, P1C active-low;	P1B, P1D active-high						
	1111 = PW	ivi mode; P1A, P1C active-low;	PIB, PID active-low						

16.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCKx. The master determines when the slave (Processor 2, Figure 16-2) will broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDOx output could be disabled (programmed as an input). The SSPxSR register will continue to shift in the signal present on the SDIx pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPxBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode. The clock polarity is selected by appropriately programming the CKP bit (SSPxCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 16-3, Figure 16-5 and Figure 16-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- · Fosc/4 (or Tcy)
- Fosc/16 (or 4 Tcy)
- Fosc/64 (or 16 Tcy)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 16-3 shows the waveforms for Master mode. When the CKE bit is set, the SDOx data is valid before there is a clock edge on SCKx. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPxBUF is loaded with the received data is shown.



FIGURE 16-3: SPI MODE WAVEFORM (MASTER MODE)

16.4 I²C Mode

The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial clock (SCLx) RC3/SCK1/SCL1 or RD6/SCK2/SCL2
- Serial data (SDAx) RC4/SDI1/SDA1 or RD5/SDI2/SDA2

The user must configure these pins as inputs by setting the associated TRIS bits.



16.4.1 REGISTERS

The MSSP module has six registers for $\mathsf{I}^2\mathsf{C}$ operation. These are:

- MSSP Control Register 1 (SSPxCON1)
- MSSP Control Register 2 (SSPxCON2)
- MSSP Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSP Shift Register (SSPxSR) Not directly accessible
- MSSP Address Register (SSPxADD)

SSPxCON1, SSPxCON2 and SSPxSTAT are the control and status registers in I²C mode operation. The SSPxCON1 and SSPxCON2 registers are readable and writable. The lower 6 bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

Many of the bits in SSPxCON2 assume different functions, depending on whether the module is operating in Master or Slave mode; bits<5:2> also assume different names in Slave mode. The different aspects of SSPxCON2 are shown in Register 16-5 (for Master mode) and Register 16-6 (Slave mode).

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from.

SSPxADD register holds the slave device address when the MSSP is configured in I²C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPxADD act as the Baud Rate Generator reload value.

In receive operations, SSPxSR and SSPxBUF together create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

Note: Disabling the MSSP module by clearing the SSPEN (SSPxCON1<5>) bit may not reset the module. It is recommended to clear the SSPxSTAT, SSPxCON1 and SSPxCON2 registers and select the mode prior to setting the SSPEN bit to enable the MSSP module.







FIGURE 16-24: STOP CONDITION RECEIVE OR TRANSMIT MODE

16.4.14 SLEEP OPERATION

While in Sleep mode, the I²C module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

16.4.15 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

16.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit (SSPxSTAT<4>) is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the MSSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed in hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- · A Start Condition
- · A Repeated Start Condition
- An Acknowledge Condition

16.4.17 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '1' on SDAx, by letting SDAx float high, and another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin = 0, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF and reset the l^2C port to its Idle state (Figure 16-25).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine, and if the I²C bus is free, the user can resume communication by asserting a Start condition.

If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the I²C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set.

A write to the SSPxBUF will start the transmission of data at the first data bit regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the l^2C bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is Idle and the S and P bits are cleared.

FIGURE 19-3: COMPARATOR OUTPUT BLOCK DIAGRAM



19.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that occurred. The CMIF bit (PIR2<6>) is the Comparator Interrupt Flag. The CMIF bit must be reset by clearing it. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

Both the CMIE bit (PIE2<6>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.



The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- a) Any read or write of CMCON will end the mismatch condition.
- b) Clear flag bit CMIF.

A mismatch condition will continue to set flag bit, CMIF. Reading CMCON will end the mismatch condition and allow flag bit, CMIF, to be cleared.

19.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional, if enabled. This interrupt will wake-up the device from Sleep mode, when enabled. Each operational comparator will consume additional current, as shown in the comparator specifications. To minimize power consumption while in Sleep mode, turn off the comparators (CM<2:0> = 111) before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

19.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state, causing the comparator modules to be turned off (CM<2:0> = 111). However, the input pins (RA0 through RA3) are configured as analog inputs by default on device Reset. The I/O configuration for these pins is determined by the setting of the PCFG<3:0> bits (ADCON1<3:0>). Therefore, device current is minimized when analog inputs are present at Reset time.

NOTES:

CLRF	Clear f	CLRWDT	Clear Watchdog Timer
Syntax:	CLRF f {,a}	Syntax:	CLRWDT
Operands:	$0 \leq f \leq 255$	Operands:	None
	a ∈ [0, 1]	Operation:	000h \rightarrow WDT,
Operation:	$\begin{array}{l} 000h \rightarrow f, \\ 1 \rightarrow Z \end{array}$		$000h \rightarrow WDT \text{ postscaler,} \\ 1 \rightarrow \overline{TO},$
Status Affected:	Z		$1 \rightarrow PD$
Encoding:	0110 101a ffff ffff	Status Affected:	TO, PD
Description:	Clears the contents of the specified	Encoding:	0000 0000 0000 0100
	register. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).	Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{\text{TO}}$ and $\overline{\text{PD}}$, are set.
	If 'a' is '0' and the extended instruction	Words:	1
	in Indexed Literal Offset Addressing	Cycles:	1
	mode whenever $f \le 95$ (5Fh). See	Q Cycle Activity:	
	Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed	Q1	Q2 Q3 Q4
	Literal Offset Mode" for details.	Decode	No Process No
Words:	1		operation Data operation
Cycles:	1	Example:	CT.RWDT
Q Cycle Activity:		Before Instruct	tion
Q1	Q2 Q3 Q4	WDT Cou	unter = ?
Decode	ReadProcessWriteregister 'f'Dataregister 'f'	After Instructio WDT Cou <u>WD</u> T Pos	n unter = 00h tscaler = 0 = 1
Example:	CLRF FLAG_REG, 1	PD	= 1
Before Instruc FLAG_RI	tion EG = 5Ah vo		
FLAG_R	EG = 00h		

LFS	R	Load FSF	र						
Synta	ax:	LFSR f, k							
Oper	ands:	$\begin{array}{l} 0 \leq f \leq 2 \\ 0 \leq k \leq 409 \end{array}$	$\begin{array}{l} 0 \leq f \leq 2 \\ 0 \leq k \leq 4095 \end{array}$						
Oper	ation:	$k \to FSRf$							
Statu	s Affected:	None							
Enco	ding:	1110 1111	1110 0000	00ff k ₇ kk	k kkkk				
Desc	ription:	The 12-bit File Select	literal 'k' Register	is loade pointee	ed into the d to by 'f'.				
Word	ls:	2							
Cycle	es:	2							
QC	ycle Activity:								
	Q1	Q2	Q3		Q4				
	Decode	Read literal 'k' MSB	Proce Data	ess a	Write literal 'k' MSB to FSRfH				
	Decode	Read literal 'k' LSB	Proce Data	ess a	Write literal k' to FSRfL				
<u>Exan</u>	n <u>ple:</u> After Instructio FSR2H FSR2L	LFSR 2, on = 03 = AE	3ABh h Bh						

Move f					
MOVF f{,	d {,a}}				
$0 \leq f \leq 255$					
d ∈ [0,1]					
a ∈ [0,1]					
$f \to dest$					
N, Z					
0101	00da	fff	f	ffff	
placed in W. If 'd' is '1', the result is placed back in register 'f' (default). 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 22.2.3 "Byte-Oriented and					
	et moue	101 1	ucia	15.	
1					
02	Q3			Q4	
Read	Proce	SS	V	/rite W	
register 'f'	Data	1			
MOVF RI	EG, 0,	0			
MOVF RI	EG, 0,	0			
MOVF RE tion = 221	EG, 0, h	0			
MOVF RE tion = 220 = FF	∑G, 0, h h	0			
MOVF RE tion = 221 = FF n = 221	EG, 0, h h	0			
	Move f MOVF f {, $0 \le f \le 255$ $d \in [0, 1]$ $a \in [0, 1]$ $f \rightarrow dest$ N, Z 0101 The content a destination status of 'd' placed in W placed back Location 'f' 256-byte back If 'a' is '0', tf If 'a' is '1', tf GPR bank (If 'a' is '0' at set is enable in Indexed I mode whene Section 22 Bit-Oriente Literal Offs 1 1 Q2 Read register 'f'	Move fMOVFf {,d {,a}} $0 \le f \le 255$ $d \in [0, 1]$ $a \in [0, 1]$ $a \in [0, 1]$ $a \in [0, 1]$ $f \rightarrow$ destN, Z010100daThe contents of regis a destination dependsstatus of 'd'. If 'd' is 'placed in W. If 'd' is 'placed back in registLocation 'f' can be at256-byte bank.If 'a' is '0', the AccessIf 'a' is '0', the AccessIf 'a' is '0', the AscessIf 'a' is '0' and the exset is enabled, this irin Indexed Literal Offmode whenever $f \le S$ Section 22.2.3 "BytBit-Oriented InstructLiteral Offset Mode111Q2Q3ReadProceregister 'f'Data	Move f $MOVF$ $f \langle .d \langle .a \rangle \}$ $0 \leq f \leq 255$ $d \in [0, 1]$ $a \in [0, 1]$ $a \in [0, 1]$ $f \rightarrow$ dest N, Z $\boxed{0101}$ $00da$ \boxed{ff} The contents of register 'f' a destination dependent usstatus of 'd'. If 'd' is '0', theplaced in W. If 'd' is '1', theplaced back in register 'f'Location 'f' can be anywhat256-byte bank.If 'a' is '0', the Access BarIf 'a' is '1', the BSR is usedGPR bank (default).If 'a' is '0' and the extenderset is enabled, this instructionin Indexed Literal Offset Amode whenever $f \leq 95$ (5FSection 22.2.3 "Byte-OriBit-Oriented InstructionsLiteral Offset Mode" for of11Q2Q3ReadProcessregister 'f'Data	Move f MOVF f {,d {,a}} $0 \le f \le 255$ d ∈ [0, 1] $a \in [0, 1]$ $a \in [0, 1]$ $f \rightarrow dest$ N, Z 0101 00da ffff The contents of register if are a destination dependent upon status of 'd'. If 'd' is '0', the resplaced in W. If 'd' is '1', the resplaced back in register 'f' (defa Location 'f' can be anywhere in 256-byte bank. If 'a' is '0', the Access Bank is If 'a' is '1', the BSR is used to start is enabled, this instruction in Indexed Literal Offset Addrem mode whenever f ≤ 95 (5Fh). Start Section 22.2.3 "Byte-Oriented Bit-Oriented Instructions in Literal Offset Mode" for detait 1 1 Q2 Q3 Read Process W register 'f' Data V	

MOVFF	Move f to	o f						
Syntax:	MOVFF f	s,f _d						
Operands:	$\begin{array}{l} 0 \leq f_s \leq 4095 \\ 0 \leq f_d \leq 4095 \end{array}$							
Operation:	$(f_s) \rightarrow f_d$							
Status Affected:	None							
Encoding: 1st word (source) 2nd word (destin.)	1100 ffff ffff ffff _s 1111 ffff ffff ffff _d							
	The contents of source register ' f_s ' are moved to destination register ' f_d '. Location of source ' f_s ' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination ' f_d ' can also be anywhere from 000h to FFFh. Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.							
words.	2							
Q Cycle Activity:	∠ (3)							

cle Activity:			
Q1	Q2	Q3	Q4
Decode	Read	Process	No
	register 'f'	Data	operation
	(src)		
Decode	No	No	Write
	operation	operation	(deet)
	No dummy		(dest)
	read		

Example:	MOVFF	REG1,	REG2	

Before Instruction		
REG1	=	33h
REG2	=	11h
After Instruction		
REG1	=	33h
REG2	=	33h

MOVLB	Move Lite	eral to Lo	ow Ni	bbl	e in BSF			
Syntax:	MOVLW k	(
Operands:	$0 \le k \le 255$							
Operation:	$k \to BSR$							
Status Affected:	None							
Encoding:	0000	0001	kkk	k	kkkk			
Description:	The eight-b Bank Select of BSR<7:4 regardless	The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of $k_7:k_4$.						
Words:	1	1						
Cycles:	1							
Q Cycle Activity:								
Q1	Q2	Q3			Q4			
Decode	Read literal 'k'	Proce Data	ess a	Wr 'k'	ite literal to BSR			
Example:	MOVLB	5						
Before Instruc BSR Reg	tion jister = 02	'h						

After Instruction BSR Register = 05h

DC CH	CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated)Operating temperature $-40^{\circ}C \le TA \le +85^{\circ}C$ for industrial				
Param No.	Sym	Characteristic	Min	Тур†	Max	Units	Conditions	
		Program Flash Memory						
D130	Ер	Cell Endurance	100	1K	—	E/W	-40°C to +85°C	
D131	Vpr	VDD for Read	VMIN	_	3.6	V	Vмın = Minimum operating voltage	
D132B	Vpew	Voltage for Self-Timed Erase or Write:						
		VDD	2.7	—	3.6	V	PIC18FXXJ10	
		VDDCORE	2.25	—	2.7	V	PIC18LFXXJ10	
D133A	Tiw	Self-Timed Write Cycle Time	_	2.8	—	ms		
D133B	TIE	Self-Timed Page Erased Cycle Time	—	33.0	—	ms		
D134	Tretd	Characteristic Retention	20	—	—	Year	Provided no other specifications are violated	
D135	IDDP	Supply Current during Programming	_	10	—	mA		

TABLE 24-1: MEMORY PROGRAMMING REQUIREMENTS

† Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

25.2 Package Details

The following sections give the technical details of the packages.

28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



	Units		INCHES	
Dimensi	on Limits	MIN	NOM	MAX
Number of Pins	Ν		28	
Pitch	е		.100 BSC	
Top to Seating Plane	Α	-	-	.200
Molded Package Thickness	A2	.120	.135	.150
Base to Seating Plane	A1	.015	-	-
Shoulder to Shoulder Width	E	.290	.310	.335
Molded Package Width	E1	.240	.285	.295
Overall Length	D	1.345	1.365	1.400
Tip to Seating Plane	L	.110	.130	.150
Lead Thickness	С	.008	.010	.015
Upper Lead Width	b1	.040	.050	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	_	_	.430

Notes:

- 1. Pin 1 visual index feature may vary, but must be located within the hatched area.
- 2. § Significant Characteristic.
- 3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
- 4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

Calibration (A/D Converter)
CALL 264
CALLW 293
Capture (CCP Module) 129
Associated Registers 131
CCP Pin Configuration 129
CCPRxH CCPRxI Registers 129
Prescaler 129
Software Interrunt 120
Capture (ECCP Module) 136
Capture (Compare/PWM (CCP)
Capture Mode, See Capture
CCP Modules and Timer Resources 128
CCPDyL Degister 129
CCPDvl Degister
COPRIL REGISTER
Compare Mode. See Compare.
Interactions Between ECCPT/CCPT and
CCP2 for Timer Resources
Module Configuration
Clock Sources
Default System Clock on Reset
Selection Using OSCCON Register
CLRF
CLRWD1
Code Examples
16 x 16 Signed Multiply Routine
16 x 16 Unsigned Multiply Routine
8 x 8 Signed Multiply Routine
8 x 8 Unsigned Multiply Routine81
Changing Between Capture Prescalers
Computed GOTO Using an Offset Value
Erasing a Flash Program Memory Row
Fast Register Stack55
How to Clear RAM (Bank 1) Using
Indirect Addressing66
Implementing a Real-Time Clock Using
a Timer1 Interrupt Service
Initializing PORTA98
Initializing PORTB101
Initializing PORTC104
Initializing PORTD107
Initializing PORTE110
Loading the SSP1BUF (SSP1SR) Register152
Reading a Flash Program Memory Word75
Saving STATUS, WREG and
BSR Registers in RAM95
Writing to Flash Program Memory78
Code Protection
COMF
Comparator
Analog Input Connection Considerations
Associated Registers229
Configuration226
Effects of a Reset
Interrupts228
Operation
Operation During Sleep
Outputs
Reference
External Signal227
Internal Signal
Response Time
Comparator Specifications 316

Comparator Voltage Reference	231
Accuracy and Error	232
Associated Registers	233
Configuring	231
Connection Considerations	232
Effects of a Reset	232
Operation During Sleep	232
Compare (CCP Module)	130
Associated Registers	131
CCPRx Register	130
Pin Configuration	130
Software Interrupt	130
Special Event Trigger	130
Timer1 Mode Selection	130
Compare (ECCP Module)	136
Special Event Trigger	136, 222
Computed GOTO	55
Configuration Bits	235
Configuration Register Protection	
Context Saving During Interrupts	95
CPFSEQ	266
CPFSGT	
CPFSLT	
Crystal Oscillator/Ceramic Resonator	27
Customer Change Notification Service	363
Customer Notification Service	363
Customer Support	
••	

D

Data Addressing Modes	66
Comparing Addressing Modes with the	
Extended Instruction Set Enabled	69
Direct	66
Indexed Literal Offset	68
Instructions Affected	68
Indirect	66
Inherent and Literal	66
Data Memory	58
Access Bank	60
and the Extended Instruction Set	68
Bank Select Register (BSR)	58
General Purpose Registers	60
Map for PIC18F45J10 Family	59
Special Function Registers	61
DAW	268
DC Characteristics	313
Power-Down and Supply Current	306
Supply Voltage	305
DCFSNZ	269
DECF	268
DECFSZ	269
Default System Clock	31
Development Support	299
Device Overview	7
Core Features	7
Details on Individual Family Members	8
Features (table)	9
Other Special Features	8
Direct Addressing	67
5	