



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

Product Status	Active
Core Processor	
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	32
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 13x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf45j10-i-pt

Email: info@E-XFL.COM

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

Pin Diagrams



TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at www.microchip.com to receive the most current information on all of our products.

6.1.4.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a full or underflow will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit but not cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

6.1.5 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers, to provide a "fast return" option for interrupts. The stack for each register is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the RETFIE, FAST instruction is used to return from the interrupt.

If both low and high-priority interrupts are enabled, the stack registers cannot be used reliably to return from low-priority interrupts. If a high-priority interrupt occurs while servicing a low-priority interrupt, the stack register values stored by the low-priority interrupt will be overwritten. In these cases, users must save the key registers in software during a low-priority interrupt.

If interrupt priority is not used, all interrupts may use the Fast Register Stack for returns from interrupt. If no interrupts are used, the Fast Register Stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the Fast Register Stack for a subroutine call, a CALL label, FAST instruction must be executed to save the STATUS, WREG and BSR registers to the Fast Register Stack. A RETURN, FAST instruction is then executed to restore these registers from the Fast Register Stack.

Example 6-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

EXAMPLE 6-1: FAST REGISTER STACK CODE EXAMPLE

CALL SUB1, FAST	;STATUS, WREG, BSR ;SAVED IN FAST REGISTER ;STACK
SUB1 •	
RETURN, FAST	;RESTORE VALUES SAVED ;IN FAST REGISTER STACK

6.1.6 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

6.1.6.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 6-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW nn instructions that returns the value 'nn' to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of 2 (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

EXAMPLE 6-2: COMPUTED GOTO USING AN OFFSET VALUE

	MOVF	OFFSET,	W
	CALL	TABLE	
ORG	nn00h		
TABLE	ADDWF	PCL	
	RETLW	nnh	
	RETLW	nnh	
	RETLW	nnh	

6.1.6.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to or from program memory one byte at a time.

Table read and table write operations are discussed further in Section 7.1 "Table Reads and Table Writes".

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	Fop-of-Stack Upper Byte (TOS<20:16>)					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 (TOS<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 (TOS<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> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 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 colspan="6">Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)</td><td>N/A</td><td>47, 67</td></t<>	POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical 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	0 Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (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 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 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 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 Address Pointer 0 Low Byte xxxx xxxx 47,6							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 Register xxxx xxxx 47								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 contents of FSR2 to address data memory – value of FSR2 not changed (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 STATUS — — N OV Z DC C x xxxxx 48, 65	POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-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) – 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 contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) N						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.

Example 8-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

RES3:RES0	=	ARG1H:ARG1L • ARG2H:ARG2L
	=	$(ARG1H \bullet ARG2H \bullet 2^{16}) +$
		$(ARG1H \bullet ARG2L \bullet 2^8) +$
		$(ARG1L \bullet ARG2H \bullet 2^8) +$
		(ARG1L • ARG2L)

EXAMPLE 8-3: 1

16 x 16 UNSIGNED MULTIPLY ROUTINE

	MOVF	ARG1L, W	
	MULWF	ARG2L	; ARG1L * ARG2L->
			; PRODH:PRODL
	MOVFF	PRODH, RES1	;
	MOVFF	PRODL, RESO	;
;			
	MOVF	ARG1H, W	
	MULWF	ARG2H	; ARG1H * ARG2H->
			; PRODH:PRODL
	MOVFF	PRODH, RES3	;
	MOVFF	PRODL, RES2	;
;			
	MOVF	ARG1L, W	
	MULWF	ARG2H	; ARG1L * ARG2H->
			; PRODH:PRODL
	MOVF	PRODL, W	;
	ADDWF	RESI, F	; Add cross
	MOVE	PRODH, W	; products
	ADDWFC	RESZ, F	;
	CLRF	WREG	;
	ADDWFC	RES3, F	i
i	NOTE		
	MUTWE	ARGIH, W	/ • ADC111 * ADC21 >
	MOLWF	ARGZL	, ARGIH " ARGZL->
	MOVE		, PRODH.PRODL
		PRODL, W	, : Add grogg
	MOVE	RESI, F	, Add Cross
	ADDMEC	PECODII, W	; products
	CLRE	WREC	;
	ADDMEC	BEG3 E	,
	ADDWI C	NH03, 1	'

Example 8-4 shows the sequence to do a 16 x 16 signed multiply. Equation 8-2 shows the algorithm used. The 32-bit result is stored in four registers (RES3:RES0). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

RES3:RES0	= ARG1H:ARG1L • ARG2H:ARG2L
	$= (ARG1H \bullet ARG2H \bullet 2^{16}) +$
	$(ARG1H \bullet ARG2L \bullet 2^8) +$
	$(ARG1L \bullet ARG2H \bullet 2^8) +$
	$(ARG1L \bullet ARG2L) +$
	$(-1 \bullet ARG2H < 7 > \bullet ARG1H: ARG1L \bullet 2^{16}) +$
	$(-1 \bullet ARG1H < 7 > \bullet ARG2H: ARG2L \bullet 2^{16})$

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

	MOVF	ARG1L, W		
	MULWF	ARG2L	;	ARG1L * ARG2L ->
			;	PRODH:PRODL
	MOVFF	PRODH, RES1	;	
	MOVFF	PRODL, RESO	;	
;				
	MOVF	ARG1H, W		
	MULWF	ARG2H	;	ARG1H * ARG2H ->
			;	PRODH:PRODL
	MOVFF	PRODH, RES3	;	
	MOVFF	PRODL, RES2	;	
;				
	MOVF	ARG1L, W		
	MULWF	ARG2H	;	ARG1L * ARG2H ->
			;	PRODH:PRODL
	MOVF	PRODL, W	;	
	ADDWF	RES1, F	;	Add cross
	MOVF	PRODH, W	;	products
	ADDWFC	RES2, F	;	
	CLRF	WREG	;	
	ADDWFC	RES3, F	;	
;				
	MOVF	ARG1H, W	;	
	MULWF	ARG2L	;	ARG1H * ARG2L ->
			;	PRODH:PRODL
	MOVF	PRODL, W	;	
	ADDWF	RES1, F	;	Add cross
	MOVF	PRODH, W	;	products
	ADDWFC	RES2, F	;	
	CLRF	WREG	;	
	ADDWFC	RES3, F	;	
;	DEFIC			
	BIFSS	ARG2H, 7	;	ARG2H: ARG2L neg?
	BRA	SIGN_ARGI	,	no, check ARGI
	MOVE	ARGIL, W		
	SUBWF	RESZ		
		ARGIN, W	'	
	SUBWFB	KE55		
, STG	N ARGI			
510	BTESS	ARG1H 7	;	ARG1H: ARG11, neg?
	BRA	CONT CODE	;	no done
	MOVE	ARG2L, W	;	no, done
	SUBWF	RES2	;	
	MOVE	ARG2H, W	;	
	SUBWFB	RES3		
;				
CON	T_CODE			
	:			

9.5 RCON Register

The RCON register contains bits used to determine the cause of the last Reset or wake-up from Idle or Sleep modes. RCON also contains the bit that enables interrupt priorities (IPEN).

REGISTER 9-13: RCON: RESET CONTROL REGISTER

R/W-0	U-0	R/W-1	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	CM	RI	TO	PD	POR	BOR
bit 7							bit 0

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

bit 7	IPEN: Interrupt Priority Enable bit
	1 = Enable priority levels on interrupts
	0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
bit 6	Unimplemented: Read as '0'
bit 5	CM: Configuration Mismatch Flag bit
	For details of bit operation, see Register 5-1.
bit 4	RI: RESET Instruction Flag bit
	For details of bit operation, see Register 5-1.
bit 3	TO: Watchdog Timer Time-out Flag bit
	For details of bit operation, see Register 5-1.
bit 2	PD: Power-Down Detection Flag bit
	For details of bit operation, see Register 5-1.
bit 1	POR: Power-on Reset Status bit
	For details of bit operation, see Register 5-1.
bit 0	BOR: Brown-out Reset Status bit
	For details of bit operation, see Register 5-1.

10.3 PORTB, TRISB and LATB Registers

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

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

EXAMPLE 10-3:	INITIALIZING PORTB
---------------	--------------------

CLRF	PORTB	; Initialize PORTB by ; clearing output
		; data latches
CLRF	LATB	; Alternate method
		; to clear output
		; data latches
MOVLW	0Fh	; Set RB<4:0> as
MOVWF	ADCON1	; digital I/O pins
MOVLW	OCFh	; Value used to
		; initialize data
		; direction
MOVWF	TRISB	; Set RB<3:0> as inputs
		; RB<5:4> as outputs
		; RB<7:6> as inputs

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, RBPU (INTCON2<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Note: On a Power-on Reset, RB<4:0> are configured as analog inputs by default and read as '0'; RB<7:5> are configured as digital inputs. Four of the PORTB pins (RB<7:4>) have an interrupton-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB<7:4> pin configured as an output is excluded from the interrupton-change comparison). The input pins (of RB<7:4>) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB<7:4> are ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from Sleep mode or any of the Idle modes. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- a) Any read or write of PORTB (except with the MOVFF (ANY), PORTB instruction).
- b) Clear flag bit, RBIF.

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

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

RB3 can be configured by the Configuration bit, CCP2MX, as the alternate peripheral pin for the CCP2 module (CCP2MX = 0).

The RB5 pin is multiplexed with the Timer0 module clock input and one of the comparator outputs to become the RB5/KBI1/T0CKI/C1OUT pin.

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	_	TRISE2	TRISE1	TRISE0
bit 7							bit 0
Legend:							
R = Readable	e bit	W = Writable	bit	U = Unimplen	nented bit, read	d as '0'	
-n = Value at	POR	'1' = Bit is set		'0' = Bit is clea	ared	x = Bit is unkr	nown
bit 7	IBF: Input But	ffer Full Status	bit				
	1 = A word ha	as been receiv	ed and is waitir	ng to be read b	y the CPU		
h.H.C		nas been recei					
DIT 6		Butter Full Stat	US DIT				
	1 = The output = 0 = The output = 0	ut buffer still no	een read	ly written word			
bit 5	IBOV: Input B	Suffer Overflow	Detect bit (in N	/licroprocessor	mode)		
	1 = A write o	ccurred when a	a previously inp	out word has no	ot been read (m	ust be cleared	in software)
	0 = No overfloor	ow occurred					
bit 4	PSPMODE: F	Parallel Slave F	Port Mode Sele	ct bit			
	1 = Parallel S	Slave Port mod	e				
hit 3		ted: Read as '	o,				
bit 2		Direction Con	trol bit				
5112		Direction Con					
	0 = Output						
bit 1	TRISE1: RE1	Direction Con	trol bit				
	1 = Input						
	0 = Output						
bit 0	TRISE0: RE0	Direction Con	trol bit				
	1 = Input						

REGISTER 10-1: TRISE REGISTER (40/44-PIN DEVICES ONLY)

13.0 TIMER2 MODULE

The Timer2 timer module incorporates the following features:

- 8-bit Timer and Period registers (TMR2 and PR2, respectively)
- · Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscaler (1:1 through 1:16)
- · Interrupt on TMR2 to PR2 match
- Optional use as the shift clock for the MSSP module

The module is controlled through the T2CON register (Register 13-1) which enables or disables the timer and configures the prescaler and postscaler. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in Figure 13-1.

13.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 4-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options; these are selected by the prescaler control bits, T2CKPS<1:>0 (T2CON<1:0>). The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscaler (see **Section 13.2 "Timer2 Interrupt"**).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, MCLR Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7	•			·		•	bit 0
Legend:							
R = Readable	bit	W = Writable	bit	U = Unimplem	ented bit, read	1 as '0'	
-n = Value at I	POR	'1' = Bit is set		'0' = Bit is clea	ared	x = Bit is unkn	Iown
bit 7	Unimplement	ted: Read as 'd)'				
bit 6-3	T2OUTPS<3:	0>: Timer2 Ou	tput Postscale	Select bits			
	0000 = 1:1 Po	ostscale					
	0001 = 1:2 Po	ostscale					
	•						
	•						
	1111 = 1:16 F	Postscale					
bit 2	TMR2ON: Tin	ner2 On bit					
	1 = Timer2 is	on					
	0 = Timer2 is	off					
bit 1-0	T2CKPS<1:0	>: Timer2 Cloc	k Prescale Sel	ect bits			
	00 = Prescale	er is 1					
	01 = Prescale	eris 4					
	$\perp x = Prescale$	er is 16					

REGISTER 1	I6-6: SSPx	CON2: MSSF	Px CONTRO	L REGISTER	2 (I ² C™ SLA	VE MODE)		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
GCEN	ACKSTAT	ADMSK5	ADMSK4	ADMSK3	ADMSK2	ADMSK1	SEN ⁽¹⁾	
bit 7				·			bit 0	
Legend:								
R = Readable	e bit	W = Writable	bit	U = Unimplem	nented bit, read	d as '0'		
-n = Value at	POR	'1' = Bit is set		'0' = Bit is clea	ared	x = Bit is unkn	own	
bit 7	GCEN: General 1 = Enable int 0 = General c	ral Call Enable terrupt when a all address dis	bit general call ac abled	ldress (0000h)	is received in t	he SSPxSR		
bit 6	ACKSTAT: Ac Unused in Sla	cknowledge Sta ive mode.	atus bit					
bit 5-2	ADMSK<5:2> 1 = Masking c 0 = Masking c	Slave Addres of correspondin of correspondin	ss Mask Select g bits of SSPx g bits of SSPx	t bits ADD enabled ADD disabled				
bit 1	 a Masking of corresponding bits of SSPXADD disabled a ADMSK1: Slave Address Least Significant bit(s) Mask Select bit a T-Bit Addressing mode: a Masking of SSPxADD<1> only enabled b Masking of SSPxADD<1> only disabled b In 10-Bit Addressing mode: c Masking of SSPxADD<1> enabled 							
bit 0	SEN: Stretch 1 = Clock stre 0 = Clock stre	Enable bit ⁽¹⁾ etching is enabl etching is disab	ed for both sla led	ve transmit and	I slave receive	(stretch enable	d)	

Note 1: If the I²C module is active, this bit may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

16.4.3.2 Address Masking

Masking an address bit causes that bit to become a "don't care". When one address bit is masked, two addresses will be Acknowledged and cause an interrupt. It is possible to mask more than one address bit at a time, which makes it possible to Acknowledge up to 31 addresses in 7-Bit Addressing mode and up to 63 addresses in 10-Bit Addressing mode (see Example 16-2).

The I²C Slave behaves the same way, whether address masking is used or not. However, when address masking is used, the I²C slave can Acknowledge multiple addresses and cause interrupts. When this occurs, it is necessary to determine which address caused the interrupt by checking SSPxBUF.

In 7-Bit Addressing mode, Address Mask bits, ADMSK<5:1> (SSPxCON2<5:1>), mask the corresponding address bits in the SSPxADD register. For any ADMSK bits that are set (ADMSK<n> = 1), the corresponding address bit is ignored (SSPxADD<n> = x). For the module to issue an address Acknowledge, it is sufficient to match only on addresses that do not have an active address mask.

In 10-Bit Addressing mode, ADMSK<5:2> bits mask the corresponding address bits in the SSPxADD register. In addition, ADMSK1 simultaneously masks the two LSbs of the address (SSPxADD<1:0>). For any ADMSK bits that are active (ADMSK<n> = 1), the corresponding address bit is ignored (SSPxADD<n> = x). Also note that although in 10-Bit Addressing mode, the upper address bits reuse part of the SSPxADD register bits, the address mask bits do not interact with those bits. They only affect the lower address bits.

Note 1: ADMSK1 masks the two Least Significant bits of the address.

 The two Most Significant bits of the address are not affected by address masking.

EXAMPLE 16-2: ADDRESS MASKING EXAMPLES

7-Bit Addressing:

SSPxADD<7:1>= A0h (1010000) (SSPxADD<0> is assumed to be '0')

ADMSK<5:1> = 00111

Addresses Acknowledged: A0h, A2h, A4h, A6h, A8h, AAh, ACh, AEh

10-Bit Addressing:

SSPxADD<7:0>= A0h (10100000) (the two MSbs of the address are ignored in this example, since they are not affected by masking)

ADMSK<5:1> = 00111

Addresses Acknowledged: A0h, A1h, A2h, A3h, A4h, A5h, A6h, A7h, A8h, A9h, AAh, ABh, ACh, ADh, AEh, AFh

16.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

- a) After the SDAx pin has been deasserted and allowed to float high, SDAx is sampled low after the BRG has timed out.
- b) After the SCLx pin is deasserted, SCLx is sampled low before SDAx goes high.

The Stop condition begins with SDAx asserted low. When SDAx is sampled low, the SCLx pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPxADD<6:0> and counts down to 0. After the BRG times out, SDAx is sampled. If SDAx is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 16-31). If the SCLx pin is sampled low before SDAx is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 16-32).

FIGURE 16-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)



FIGURE 16-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)



R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7						-	bit 0
Legend:							
R = Readable	e bit	W = Writable	bit	U = Unimplen	nented bit, read	d as '0'	
-n = Value at	POR	'1' = Bit is set		'0' = Bit is clea	ared	x = Bit is unkr	nown
bit 7	ADFM: A/D 1 = Right jus 0 = Left justi	Result Format S stified fied	elect bit				
bit 6	Unimpleme	nted: Read as '	o'				
bit 5-3	ACQT<2:0>	: A/D Acquisition	n Time Select	bits			
	111 = 20 TAI 110 = 16 TAI 101 = 12 TAI 100 = 8 TAD 011 = 6 TAD 010 = 4 TAD 001 = 2 TAD 000 = 0 TAD	D D D (1)					
bit 2-0	ADCS<2:0> 111 = FRC (c 110 = Fosc/ 101 = Fosc/ 001 = Fosc/ 011 = FRC (c 010 = Fosc/ 001 = Fosc/ 000 = Fosc/	: A/D Conversio clock derived fro /64 /16 /4 clock derived fro /32 /2	n Clock Select m A/D RC osc m A/D RC osc	t bits cillator) ⁽¹⁾ cillator) ⁽¹⁾			

REGISTER 18-3: ADCON2: A/D CONTROL REGISTER 2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

18.5 A/D Conversions

Figure 18-3 shows the operation of the A/D converter after the GO/DONE bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 18-4 shows the operation of the A/D converter after the GO/DONE bit has been set, the ACQT<2:0> bits are set to '010' and selecting a 4 TAD acquisition time before the conversion starts.

Clearing the GO/DONE bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. This means the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

Note:	The GO/DONE bit should NOT be set in
	the same instruction that turns on the A/D.

18.6 Use of the ECCP2 Trigger

An A/D conversion can be started by the "Special Event Trigger" of the ECCP2 module. This requires that the CCP2M<3:0> bits (CCP2CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/DONE bit will be set, starting the A/D acquisition and conversion and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition period is either timed by the user, or an appropriate TACQ time is selected before the Special Event Trigger sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the Special Event Trigger will be ignored by the A/D module but will still reset the Timer1 (or Timer3) counter.

FIGURE 18-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)



FIGURE 18-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)



20.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 16-tap resistor ladder network that provides a selectable reference voltage. Although its primary purpose is to provide a reference for the analog comparators, it may also be used independently of them.

A block diagram of the module is shown in Figure 20-1. The resistor ladder is segmented to provide two ranges of CVREF values and has a power-down function to conserve power when the reference is not being used. The module's supply reference can be provided from either device VDD/VSS or an external voltage reference.

20.1 Configuring the Comparator Voltage Reference

The voltage reference module is controlled through the CVRCON register (Register 20-1). The comparator voltage reference provides two ranges of output voltage, each with 16 distinct levels. The range to be

used is selected by the CVRR bit (CVRCON<5>). The primary difference between the ranges is the size of the steps selected by the CVREF Selection bits (CVR<3:0>), with one range offering finer resolution. The equations used to calculate the output of the comparator voltage reference are as follows:

<u>If CVRR = 1:</u> CVREF = ((CVR<3:0>)/24) x CVRSRC <u>If CVRR = 0:</u> CVREF = (CVRSRC x 1/4) + (((CVR<3:0>)/32) x CVRSRC)

The comparator reference supply voltage can come from either VDD and VSS, or the external VREF+ and VREF- that are multiplexed with RA2 and RA3. The voltage source is selected by the CVRSS bit (CVRCON<4>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see Table 24-3 in **Section 24.0 "Electrical Characteristics"**).

REGISTER 20-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE ⁽¹⁾	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

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

bit /	CVREN : Comparator Voltage Reference Enable bit
	1 = CVREF circuit powered on
	0 = CVREF circuit powered down
bit 6	CVROE: Comparator VREF Output Enable bit ⁽¹⁾
	 1 = CVREF voltage level is also output on the RA2/AN2/VREF-/CVREF pin 0 = CVREF voltage is disconnected from the RA2/AN2/VREF-/CVREF pin
bit 5	CVRR: Comparator VREF Range Selection bit
	 1 = 0 to 0.667 CVRSRC, with CVRSRC/24 step size (low range) 0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size (high range)
bit 4	CVRSS: Comparator VREF Source Selection bit
	 1 = Comparator reference source, CVRSRC = (VREF+) – (VREF-) 0 = Comparator reference source, CVRSRC = VDD – VSS
bit 3-0	CVR<3:0>: Comparator VREF Value Selection bits ($0 \le (CVR<3:0>) \le 15$)
	When CVRR = 1:
	$\overline{\text{CVREF}} = ((\text{CVR} < 3:0 >)/24) \bullet (\text{CVRSRC})$
	When CVRR = 0:
	CVREF = (CVRSRC/4) + ((CVR<3:0>)/32) • (CVRSRC)

Note 1: CVROE overrides the TRISA<2> bit setting.



FIGURE 20-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM

20.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 20-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in **Section 24.0 "Electrical Characteristics"**.

20.3 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the CVRCON register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

20.4 Effects of a Reset

A device Reset disables the voltage reference by clearing bit, CVREN (CVRCON<7>). This Reset also disconnects the reference from the RA2 pin by clearing bit, CVROE (CVRCON<6>) and selects the high-voltage range by clearing bit, CVRR (CVRCON<5>). The CVR value select bits are also cleared.

20.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RA2 pin if the CVROE bit is set. Enabling the voltage reference output onto RA2 when it is configured as a digital input will increase current consumption. Connecting RA2 as a digital output with CVRSS enabled will also increase current consumption.

The RA2 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 20-2 shows an example buffering technique.

21.2 Watchdog Timer (WDT)

For PIC18F45J10 family devices, the WDT is driven by the INTRC oscillator. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexor, controlled by the WDTPS bits in Configuration Register 2H. Available periods range from about 4 ms to 135 seconds (2.25 minutes) depending on voltage, temperature and Watchdog postscaler. The WDT and postscaler are cleared whenever a SLEEP or CLRWDT instruction is executed, or a clock failure (primary or Timer1 oscillator) has occurred.

FIGURE 21-1: WDT BLOCK DIAGRAM



2: When a CLRWDT instruction is executed, the postscaler count will be cleared.

21.2.1 CONTROL REGISTER

The WDTCON register (Register 21-9) is a readable and writable register. The SWDTEN bit enables or disables WDT operation.



REGISTER 21-9: WDTCON: WATCHDOG TIMER CONTROL REGISTER

u-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN ⁽¹⁾
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 7-1 Unimplemented: Read as '0'

bit 0 **SWDTEN:** Software Controlled Watchdog Timer Enable bit⁽¹⁾

1 = Watchdog Timer is on

0 = Watchdog Timer is off

Note 1: This bit has no effect if the Configuration bit, WDTEN, is enabled.

TABLE 21-2:SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
RCON	IPEN	_	CM	RI	TO	PD	POR	BOR	48
WDTCON		_	—	—	—	_	_	SWDTEN	48

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Watchdog Timer.

25.0 PACKAGING INFORMATION

25.1 Package Marking Information

28-Lead SPDIP



28-Lead SOIC



28-Lead SSOP



28-Lead QFN





Example



Example



Example



Leger	nd: XXX Y YY WW NNN @3 *	Customer-specific information Year code (last digit of calendar year) Year code (last 2 digits of calendar year) Week code (week of January 1 is week '01') Alphanumeric traceability code Pb-free JEDEC designator for Matte Tin (Sn) This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package.
Note:	In the eve be carrie characters	nt the full Microchip part number cannot be marked on one line, it will d over to the next line, thus limiting the number of available s for customer-specific information.

INDEX

Α

A/D	215
A/D Converter Interrupt Configuring	219
Acquisition Requirements	220
ADCAL Bit	
ADCON0 Register	215
ADCON1 Register	215
ADCON2 Register	215
ADRESH Register	5, 218
ADRESL Register	215
Analog Port Pins, Configuring	221
Associated Registers	223
Automatic Acquisition Time	221
Calculating the Minimum Required	
Acquisition Time	220
Calibration	223
Configuring the Module	219
Conversion Clock (TAD)	221
Conversion Status (GO/DONE Bit)	218
Conversions	222
Converter Characteristics	334
Operation in Power-Managed Modes	223
Special Event Trigger (ECCP)13	6, 222
Use of the ECCP2 Trigger	222
Absolute Maximum Ratings	303
AC (Timing) Characteristics	317
Load Conditions for Device	
Liming Specifications	318
Parameter Symbology	317
Temperature and Voltage Specifications	318
	310
A access Depl	
Access Bank Manning with Indexed Literal Offect Mede	70
Access Bank Mapping with Indexed Literal Offset Mode	70
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT	70 182
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag	70 182 182 223
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register	70 182 182 223 215
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit	70 182 182 223 215 218
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register	70 182 182 223 215 218 215
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register	70 182 223 215 218 215 215 215
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register	70 182 223 215 218 215 215 215 292
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW	70 182 223 215 215 215 215 292 255
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK	70 182 213 215 215 215 215 292 255 292
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDLW ADDULNK	70 182 223 215 215 215 215 292 255 292 255
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDWF ADDWF	70 182 213 215 215 215 292 255 292 255 256
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADDWF ADRESH Register	70 182 213 215 215 215 292 255 292 255 256 256 215
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADDWF ADRESH Register ADRESL Register 21	70 182 213 215 215 215 292 255 292 255 256 215 5, 218
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADDWF ADRESH Register ADRESL Register Analog-to-Digital Converter. See A/D.	70 182 213 215 215 215 292 255 292 255 256 215 5, 218
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADDWF ADRESH Register ADRESL Register Analog-to-Digital Converter. See A/D. ANDLW	70 182 213 215 215 215 292 255 292 255 256 215 5, 218 256
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADDWF ADRESH Register ADRESL Register ADRESL Register ADDLW ANDLW ANDLW	70 182 215 215 215 215 292 255 292 255 256 215 5, 218 256 257
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADDWF ADRESH Register ADRESH Register ADRESL Register ADRESL Register ADDLW ANDLW ANDLW ANDLW ANDWF ASsembler	70 182 182 215 215 215 215 292 255 292 255 256 215 5, 218 256 257
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADDWF ADRESH Register ADRESH Register ADRESL REGISTER ADR	70 182 215 215 215 215 292 255 292 255 256 215 5, 218 256 257 257
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDUWF ADDWF ADDWF ADRESH Register ADRESH Register ADRESL REGISTER ADRE	70 182 215 215 215 215 292 255 292 255 256 215 5, 218 256 257 257
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDLW ADDULNK ADDWF ADDWF ADRESH Register ADRESH Register ADRESH Register ADRESL REGISTER	70 182 213 215 215 215 292 255 292 256 215 5, 218 256 256 257 256 257 300 206
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDLW ADDULNK ADDWF ADRESH Register ADRESH Register ADRESH Register ADRESL REGIS	70 182 213 215 215 215 292 255 292 256 215 5, 218 256 256 257 300 300 58
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDLW ADDULNK ADDWF ADRESH Register ADRESH Register ADRESL REGISTER ANDLW ANDWF Assembler MPASM Assembler Auto-Wake-up on Sync Break Character B Bank Select Register (BSR) Baud Rate Generator	70 182 213 215 215 215 292 255 292 256 256 256 257 256 257 300 58 58 58
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDLW ADDULNK ADDWF ADBWF ADRESH Register ADRESH Register ADRESL REGISTER ANDLW ANDWF Assembler MPASM Assembler Auto-Wake-up on Sync Break Character B Bank Select Register (BSR) Baud Rate Generator BC	70 182 215 215 215 215 292 255 292 256 256 256 257 256 257 300 58 58 58 57
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDLW ADDULNK ADDWF ADDWF ADRESH Register ADRESH Register ADRESH Register ADRESL REGISTER ANDLW ANDWF Assembler MPASM Assembler Auto-Wake-up on Sync Break Character BC Baud Rate Generator BC BC MEAST	70 182 215 215 215 215 292 255 292 256 256 256 256 257 300 58 58 58 58
Access Bank Mapping with Indexed Literal Offset Mode ACKSTAT ACKSTAT Status Flag ADCAL Bit ADCON0 Register GO/DONE Bit ADCON1 Register ADCON2 Register ADCON2 Register ADDFSR ADDLW ADDULNK ADDULNK ADDWF ADRESH Register ADRESH Register ADRESL REG	70 182 215 215 215 215 292 255 292 256 256 256 256 256 257 300 58 58 58 58 58

Block	Diagrams

Block Diagrams	
A/D	218
Analog Input Model	219
Baud Rate Generator	178
Capture Mode Operation	129
Comparator Analog Input Model	229
Comparator I/O Operating Modes	226
Comparator Output	228
Comparator Voltage Reference	232
Comparator Voltage Reference Output	202
Buffer Example	233
Compare Mede Operation	120
Device Cleck	20
	107
	137
	205
	203
External Power-on Reset Circuit	
(Slow VDD Power-up)	43
Fail-Safe Clock Monitor	245
Generic I/O Port Operation	97
Interrupt Logic	84
MSSP (I ² C Master Mode)	176
MSSP (I ² C Mode)	159
MSSP (SPI Mode)	149
On-Chip Reset Circuit	41
PIC18F24J10/25J10	10
PIC18F44J10/45J10	11
PII	29
PORTD and PORTE (Parallel Slave Port)	113
PWM Operation (Simplified)	132
Poads from Elash Program Momory	75
Single Compositor	
	221
Table Read Operation	/1
	72
Table Writes to Flash Program Memory	//
Timer0 in 16-Bit Mode	116
Timer0 in 8-Bit Mode	116
Timer1	120
Timer1 (16-Bit Read/Write Mode)	121
Timer2	126
Watchdog Timer	242
BN	258
BNC	259
BNN	259
BNOV	260
BNZ	260
BOR See Brown-out Reset	
BOV	263
BDA	261
Brook Character (12 Bit) Transmit and Bossiva	201
BPC See Paud Date Consister	200
BRG. See Bauu Rale Generator.	40
BIOWII-OUL RESEL (BOR)	43
and On-Chip Voltage Regulator	243
Disabling in Sleep Mode	43
BSF	261
BIFSC	262
BTFSS	262
BTG	263
BZ	264
C	
U	

C Compilers	
MPLAB C18	 300
MPLAB C30	 300

L
LFSR
Μ
Master Clear (MCLR)
Master Synchronous Serial Port (MSSP). See MSSP.
Memory Organization51
Data Memory58
Program Memory51
Memory Programming Requirements
Microchip Internet Web Site
MOVF
MOVFF
MOVLB
MOVLW
MOVSF
MOVSS
MOVWF
MPLAB ASM30 Assembler, Linker, Librarian
MPLAB ICD 2 In-Circuit Debugger
MPLAB ICE 2000 High-Performance
Universal In-Circuit Emulator
MPLAB Integrated Development
Environment Software
MPLAB PM3 Device Programmer
MPLAB REAL ICE In-Circuit Emulator System
MPLINK Object Linker/MPLIB Object Librarian
MSSP
ACK Pulse164, 166
Control Registers (general)149
I ² C Mode. See I ² C Mode.
Module Overview149
SPI Master/Slave Connection153
SSPxBUF Register154
SSPxSR Register
MULLW
MULWF
N

Ν

NEGF	
NOP	
Notable Differences Between PIC18F4520	
and PIC18F45J10 Families	
Oscillator Options	
Peripherals	
Pinouts	
Power Requirements	

0

Oscillator Configuration	27
EC	27
ECPLL	
HS	
HS Modes	27
HSPLL	27
Internal Oscillator Block	
INTRC	
Oscillator Selection	
Oscillator Start-up Timer (OST)	
Oscillator Switching	
Oscillator Transitions	
Oscillator, Timer1	

Ρ

Packaging Information	337
Marking	337
Derellel Slove Dert (DSD)	107 112
Associated Poristors	107, 113
Associated Registers	112
	113
PD (Peed Input)	113
RD (Reau IIIpul)	107 113
	107, 113
DICETART Dive Development Drogrommer	202
DIE Degistere	302
PIE Reyisters	
	12 16
	12,10
	12,10
	12, 10
RA0/AN0	13, 17
	13, 17
RAZ/ANZ/VREF-/GVREF	12 17
	13, 17
	1/ 10
DD1/INT1/AN10	1/ 10
	1/ 10
	14, 10
RB3/AN9/CCF2	1/ 18
	14, 10
	10 1/
	1/ 18
	1/ 19
	14, 10
PC1/T1OSU/CCP2	15, 19
PC2/CCP1	15, 15
	10
RC3/SCK1/SCL1	15 10
	15 10
RC5/SDO1	15 10
RC6/TX/CK	15 19
RC7/RX/DT	15 19
RD0/PSP0/SCK2/SCL2	20
RD1/PSP1/SDI2/SDA2	20
RD2/PSP2/SDO2	20
RD3/PSP3/SS2	20
RD4/PSP4	20
RD5/PSP5/P1B	
RD6/PSP6/P1C	
RD7/PSP7/P1D	
RE0/RD/AN5	
RE1/WR/AN6	
RE2/CS/AN7	
VDD	15, 21
VDDCORE/VCAP	15, 21
Vss	15, 21
Pinout I/O Descriptions	
PIC18F24J10/25J10	12
PIC18F44J10/45J10	16
PIR Registers	
PLL Frequency Multiplier	29
ECPLL Oscillator Mode	
HSPLL Oscillator Mode	
POP	278
POR. See Power-on Reset.	