Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

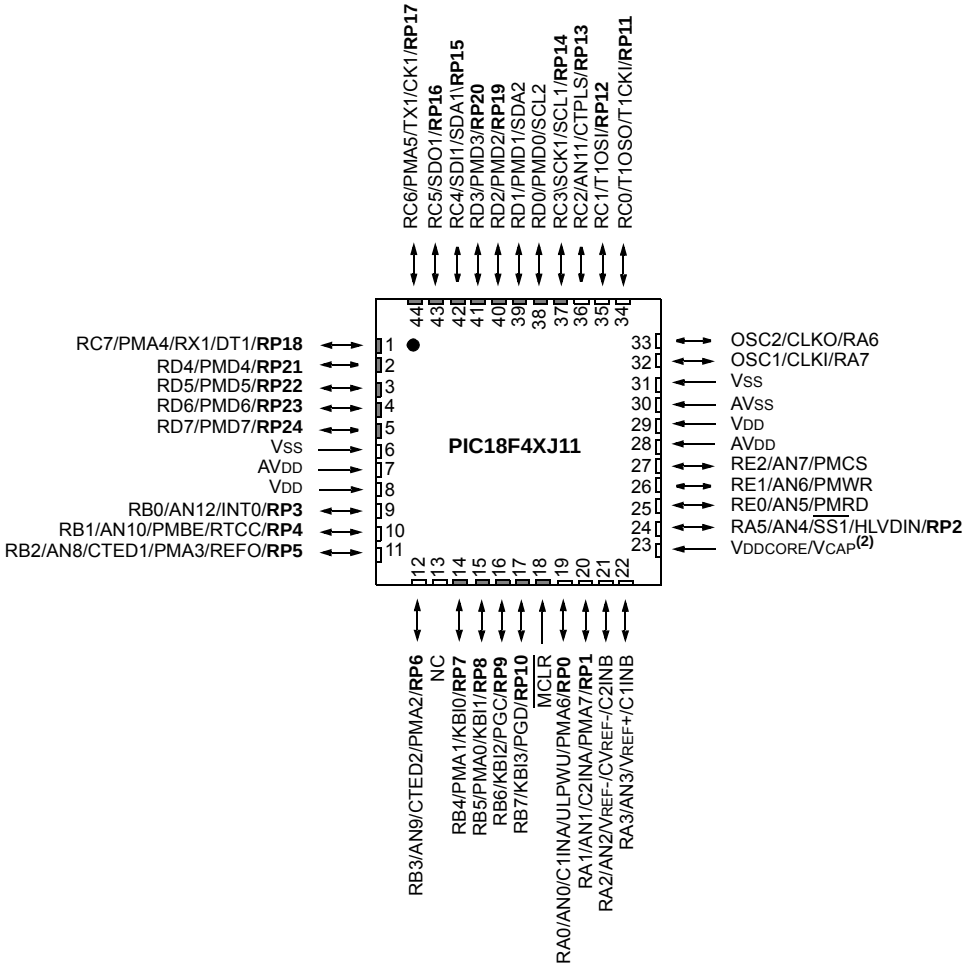
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	48MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2.15V ~ 3.6V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25j11-i-ss

Pin Diagrams (Continued)

44-Pin QFN(1,3)

■ = Pins are up to 5.5V tolerant



Legend: **RPn** represents remappable pins.

Note 1: Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see **Section 10.7 "Peripheral Pin Select (PPS)"**.

2: See **Section 26.3 "On-Chip Voltage Regulator"** for details on how to connect the VDDCORE/VCAP pin.

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

© 2011 Microchip Technology Inc.

3.0 OSCILLATOR CONFIGURATIONS

3.1 Overview

Devices in the PIC18F46J11 family incorporate a different oscillator and microcontroller clock system than general purpose PIC18F devices.

The PIC18F46J11 family has additional prescalers and postscalers, which have been added to accommodate a wide range of oscillator frequencies. Figure 3-1 provides an overview of the oscillator structure.

Other oscillator features used in PIC18 enhanced microcontrollers, such as the internal oscillator block and clock switching, remain the same. They are discussed later in this chapter.

3.1.1 OSCILLATOR CONTROL

The operation of the oscillator in PIC18F46J11 family devices is controlled through three Configuration registers and two control registers. Configuration registers, CONFIG1L, CONFIG1H and CONFIG2L, select the oscillator mode, PLL prescaler and CPU divider options. As Configuration bits, these are set when the device is programmed and left in that configuration until the device is reprogrammed.

The OSCCON register (Register 3-2) selects the Active Clock mode; it is primarily used in controlling clock switching in power-managed modes. Its use is discussed in **Section 3.3.1 "Oscillator Control Register"**.

The OSCTUNE register (Register 3-1) is used to trim the INTOSC frequency source and select the low-frequency clock source that drives several special features. The OSCTUNE register is also used to activate or disable the Phase Locked Loop (PLL). Its use is described in **Section 3.2.5.1 "OSCTUNE Register"**.

3.2 Oscillator Types

PIC18F46J11 family devices can be operated in eight distinct oscillator modes. Users can program the FOSC<2:0> Configuration bits to select one of the modes listed in Table 3-1. For oscillator modes which produce a clock output (CLKO) on pin RA6, the output frequency will be one fourth of the peripheral clock frequency. The clock output stops when in Sleep mode, but will continue during Idle mode (see Figure 3-1).

TABLE 3-1: OSCILLATOR MODES

Mode	Description
ECPLL	External Clock Input mode, the PLL can be enabled or disabled in software, CLKO on RA6, apply external clock signal to RA7.
EC	External Clock Input mode, the PLL is always disabled, CLKO on RA6, apply external clock signal to RA7.
HSPLL	High-Speed Crystal/Resonator mode, PLL can be enabled or disabled in software, crystal/resonator connected between RA6 and RA7.
HS	High-Speed Crystal/Resonator mode, PLL always disabled, crystal/resonator connected between RA6 and RA7.
INTOSCPLLO	Internal Oscillator mode, PLL can be enabled or disabled in software, CLKO on RA6, port function on RA7, the internal oscillator block is used to derive both the primary clock source and the postscaled internal clock.
INTOSCPLL	Internal Oscillator mode, PLL can be enabled or disabled in software, port function on RA6 and RA7, the internal oscillator block is used to derive both the primary clock source and the postscaled internal clock.
INTOSCO	Internal Oscillator mode, PLL is always disabled, CLKO on RA6, port function on RA7, the output of the INTOSC postscaler serves as both the postscaled internal clock and the primary clock source.
INTOSC	Internal Oscillator mode, PLL is always disabled, port function on RA6 and RA7, the output of the INTOSC postscaler serves as both the postscaled internal clock and the primary clock source.

4.6.9 DEEP SLEEP MODE REGISTERS

Deep Sleep mode registers are provided in Register 4-1 through Register 4-6.

REGISTER 4-1: DSCONH: DEEP SLEEP CONTROL HIGH BYTE REGISTER (BANKED F4Dh)

R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
DSEN(1)	—	—	—	—	(Reserved)	DSULPEN	RTCWDIS
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 **DSEN:** Deep Sleep Enable bit(1)
1 = Deep Sleep mode is entered on a SLEEP command
0 = Sleep mode is entered on a SLEEP command

bit 6-3 **Unimplemented:** Read as '0'

bit 2 **(Reserved):** Always write '0' to this bit

bit 1 **DSULPEN:** Ultra Low-Power Wake-up Module Enable bit
1 = ULPWU module is enabled in Deep Sleep
0 = ULPWU module is disabled in Deep Sleep

bit 0 **RTCWDIS:** RTCC Wake-up Disable bit
1 = Wake-up from RTCC is disabled
0 = Wake-up from RTCC is enabled

Note 1: In order to enter Deep Sleep, Sleep must be executed immediately after setting DSEN.

REGISTER 4-2: DSCONL: DEEP SLEEP CONTROL LOW BYTE REGISTER (BANKED F4Ch)

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0(1)	R/W-0(1)
—	—	—	—	—	ULPWDIS	DSBOR	RELEASE
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 7-3 **Unimplemented:** Read as '0'

bit 2 **ULPWDIS:** Ultra Low-Power Wake-up Disable bit
1 = ULPWU wake-up source is disabled
0 = ULPWU wake-up source is enabled (must also set DSULPEN = 1)

bit 1 **DSBOR:** Deep Sleep BOR Event Status bit
1 = DSBOREN was enabled and V_{DD} dropped below the DSBOR arming voltage during Deep Sleep, but did not fall below V_{DSBOR}
0 = DSBOREN was disabled or V_{DD} did not drop below the DSBOR arming voltage during Deep Sleep

bit 0 **RELEASE:** I/O Pin State Release bit
Upon waking from Deep Sleep, the I/O pins maintain their previous states. Clearing this bit will release the I/O pins and allow their respective TRIS and LAT bits to control their states.

Note 1: This is the value when V_{DD} is initially applied.

© 2011 Microchip Technology Inc.

TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
PSTR1CON	PIC18F2XJ11	PIC18F4XJ11	00-0 0001	00-0 0001	uu-u uuuu
ECCP1AS	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
ECCP1DEL	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
CCPR1H	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
PSTR2CON	PIC18F2XJ11	PIC18F4XJ11	00-0 0001	00-0 0001	uu-u uuuu
ECCP2AS	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
ECCP2DEL	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
CCPR2H	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
CTMUCONH	PIC18F2XJ11	PIC18F4XJ11	0-00 000-	0-00 000-	u-uu uuu-
CTMUCONL	PIC18F2XJ11	PIC18F4XJ11	0000 00xx	0000 00xx	uuuu uuuu
CTMUICON	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
SPBRG1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
RCREG1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
TXREG1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
TXSTA1	PIC18F2XJ11	PIC18F4XJ11	0000 0010	0000 0010	uuuu uuuu
RCSTA1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
SPBRG2	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
RCREG2	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
TXREG2	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
TXSTA2	PIC18F2XJ11	PIC18F4XJ11	0000 0010	0000 0010	uuuu uuuu
EECON2	PIC18F2XJ11	PIC18F4XJ11	---- ----	---- ----	---- ----
EECON1	PIC18F2XJ11	PIC18F4XJ11	--00 x00-	--00 q00-	--00 u00-
IPR3	PIC18F2XJ11	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
PIR3	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu(3)
PIE3	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
IPR2	PIC18F2XJ11	PIC18F4XJ11	111- 1111	111- 1111	uuu- uuuu
PIR2	PIC18F2XJ11	PIC18F4XJ11	000- 0000	000- 0000	uuu- uuuu(3)
PIE2	PIC18F2XJ11	PIC18F4XJ11	000- 0000	000- 0000	uuu- uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition.

Note 1: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 5-1 for Reset value for specific condition.

5: Not implemented for PIC18F2XJ11 devices.

6: Not implemented on "LF" devices.

TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
IPR1	PIC18F2XJ11	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
PIR1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu(3)
PIE1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
RCSTA2	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
OSCTUNE	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
T1GCON	PIC18F2XJ11	PIC18F4XJ11	0000 0x00	0000 0x00	uuuu uxuu
RTCVALH	PIC18F2XJ11	PIC18F4XJ11	0xxx xxxx	0uuu uuuu	0uuu uuuu
RTCVALL	PIC18F2XJ11	PIC18F4XJ11	0xxx xxx	0uuu uuuu	0uuu uuuu
T3GCON	PIC18F2XJ11	PIC18F4XJ11	0000 0x00	uuuu uxuu	uuuu uxuu
TRISE(5)	—	PIC18F4XJ11	---- -111	---- -111	---- -uuu
TRISD(5)	—	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
TRISC	PIC18F2XJ11	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
TRISB	PIC18F2XJ11	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
TRISA	PIC18F2XJ11	PIC18F4XJ11	111- 1111	111- 1111	uuu- uuuu
ALRMCFG	PIC18F2XJ11	PIC18F4XJ11	0000 0000	uuuu uuuu	uuuu uuuu
ALRMRPT	PIC18F2XJ11	PIC18F4XJ11	0000 0000	uuuu uuuu	uuuu uuuu
ALRMVALH	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
ALRMVALL	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATE(5)	—	PIC18F4XJ11	---- -xxx	---- -uuu	---- -uuu
LATD(5)	—	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA	PIC18F2XJ11	PIC18F4XJ11	xxx- xxxx	uuu- uuuu	uuu- uuuu
DMACON1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
DMACON2	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
HLVDCON	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
PORTE(5)	—	PIC18F4XJ11	00-- -xxx	uu-- -uuu	uu-- -uuu
PORTD(5)	—	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	PIC18F2XJ11	PIC18F4XJ11	xxx- xxxx	uuu- uuuu	uuu- uuuu
SPBRGH1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition.

Note 1: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 5-1 for Reset value for specific condition.

5: Not implemented for PIC18F2XJ11 devices.

6: Not implemented on "LF" devices.

© 2011 Microchip Technology Inc.

6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as 2 bytes or 4 bytes in program memory. The Least Significant Byte (LSB) of an instruction word is always stored in a program memory location with an even address (LSB = 0). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see **Section 6.1.3 "Program Counter"**).

Figure 6-5 provides an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 6-5 displays how the instruction, GOTO 0006h, is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 27.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY

Program Memory Byte Locations →

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

6.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: CALL, MOVFF, GOTO and LSFR. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits (MSbs); the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSbs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence immediately after the first word, the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason, and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 6-4 illustrates how this works.

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

EXAMPLE 6-4: TWO-WORD INSTRUCTIONS

CASE 1:

Object Code	Source Code		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, skip this word
1111 0100 0101 0110			; Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code

CASE 2:

Object Code	Source Code		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes, execute this word
1111 0100 0101 0110			; 2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3	; continue code

18.5.8 OPERATION IN POWER-MANAGED MODES

In Sleep mode, all clock sources are disabled. Timer2 will not increment and the state of the module will not change. If the ECCPx pin is driving a value, it will continue to drive that value. When the device wakes up, it will continue from this state. If Two-Speed Start-ups are enabled, the initial start-up frequency from HFINTOSC and the postscaler may not be stable immediately.

In PRI_IDLE mode, the primary clock will continue to clock the ECCPx module without change.

18.5.8.1 Operation with Fail-Safe Clock Monitor (FSCM)

If the Fail-Safe Clock Monitor (FSCM) is enabled, a clock failure will force the device into the power-managed RC_RUN mode and the OSCFIF bit of the PIR2 register will be set. The ECCPx will then be clocked from the internal oscillator clock source, which may have a different clock frequency than the primary clock.

18.5.9 EFFECTS OF A RESET

Both Power-on Reset and subsequent Resets will force all ports to Input mode and the ECCP registers to their Reset states.

This forces the ECCP module to reset to a state compatible with previous, non-enhanced ECCP modules used on other PIC18 and PIC16 devices.

TABLE 18-5: REGISTERS ASSOCIATED WITH ECCP1 MODULE AND TIMER1 TO TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	69
RCON	IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	70
PIR1	PMPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	72
PIE1	PMPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	72
IPR1	PMPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	72
PIR2	OSCFIF	CM2IF	CM1IF	—	BCL1IF	LVDIF	TMR3IF	CCP2IF	72
PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	LVDIE	TMR3IE	CCP2IE	72
IPR2	OSCFIP	CM2IP	CM1IP	—	BCL1IP	LVDIP	TMR3IP	CCP2IP	72
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	72
TMR1L	Timer1 Register Low Byte								70
TMR1H	Timer1 Register High Byte								70
TCLKCON	—	—	—	T1RUN	—	—	T3CCP2	T3CCP1	94
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	RD16	TMR1ON	70
TMR2	Timer2 Register								70
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	70
PR2	Timer2 Period Register								70
TMR3L	Timer3 Register Low Byte								73
TMR3H	Timer3 Register High Byte								73
T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	—	$\overline{T3SYNC}$	RD16	TMR3ON	73
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								72
CCPR1H	Capture/Compare/PWM Register 1 High Byte								72
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	72
ECCP1AS	ECCP1ASE	ECCP1AS2	ECCP1AS1	ECCP1AS0	PSS1AC1	PSS1AC0	PSS1BD1	PSS1BD0	70
ECCP1DEL	P1RSEN	P1DC6	P1DC5	P1DC4	P1DC3	P1DC2	P1DC1	P1DC0	72

Legend: — = unimplemented, read as '0'. Shaded cells are not used during ECCP operation.

Note 1: These bits are only available on 44-pin devices.

19.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices include serial EEPROMs, shift registers, display drivers and A/D Converters.

19.1 Master SSP (MSSP) Module Overview

The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C™)
  - Full Master mode
  - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode with 5-bit and 7-bit address masking (with address masking for both 10-bit and 7-bit addressing)

All members of the PIC18F46J11 family have two MSSP modules, designated as MSSP1 and MSSP2. The modules operate independently:

- PIC18F4XJ11 devices – Both modules can be configured for either I^2C or SPI communication
- PIC18F2XJ11 devices:
  - MSSP1 can be used for either I^2C or SPI communication
  - MSSP2 can be used only for SPI communication

All of the MSSP1 module-related SPI and I^2C I/O functions are hard-mapped to specific I/O pins.

For MSSP2 functions:

- SPI I/O functions (SDO2, SDI2, SCK2 and $\overline{SS2}$) are all routed through the Peripheral Pin Select (PPS) module.

  These functions may be configured to use any of the RPn remappable pins, as described in **Section 10.7 “Peripheral Pin Select (PPS)”**.
- I^2C functions (SCL2 and SDA2) have fixed pin locations.

On all PIC18F46J11 family devices, the SPI DMA capability can only be used in conjunction with MSSP2. The SPI DMA feature is described in **Section 19.4 “SPI DMA Module”**.

Note:	Throughout this section, generic references to an MSSP module in any of its operating modes may be interpreted as being equally applicable to MSSP1 or MSSP2. Register names and module I/O signals use the generic designator ‘x’ to indicate the use of a numeral to distinguish a particular module when required. Control bit names are not individuated.

TABLE 19-2: REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	69
PIR1	PMPIF(2)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	72
PIE1	PMPIE(2)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	72
IPR1	PMPIP(2)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	72
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF	72
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE	72
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP	72
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	72
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	72
SSP1BUF	MSSP1 Receive Buffer/Transmit Register								70
SSPxCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	70
SSPxSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	70
SSP2BUF	MSSP2 Receive Buffer/Transmit Register								73
ODCON3(1)	—	—	—	—	—	—	SPI2OD	SPI1OD	74

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

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

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

19.5.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPxCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCLx pin to be held low at the end of each data receive sequence.

19.5.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{ACK}$ sequence, if the BF bit is set, the CKP bit in the SSPxCON1 register is automatically cleared, forcing the SCLx output to be held low. The CKP bit being cleared to '0' will assert the SCLx line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and read the contents of the SSPxBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 19-15).

Note 1: If the user reads the contents of the SSPxBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

19.5.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = 1)

In 10-Bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{W}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPxADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

Note: If the user polls the UA bit and clears it by updating the SSPxADD register before the falling edge of the ninth clock occurs, and if the user has not cleared the BF bit by reading the SSPxBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

19.5.4.3 Clock Stretching for 7-Bit Slave Transmit Mode

The 7-Bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's Interrupt Service Routine (ISR) must set the CKP bit before transmission is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and load the contents of the SSPxBUF before the master device can initiate another transmit sequence (see Figure 19-10).

Note 1: If the user loads the contents of SSPxBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit.

19.5.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high-order bits of the 10-bit address and the R/$\overline{W}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 19-13).

© 2011 Microchip Technology Inc.

19.5.9 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPxCON2<1>) is programmed high and the I²C logic module is in the Idle state. When the RSEN bit is set, the SCLx pin is asserted low. When the SCLx pin is sampled low, the BRG is loaded with the contents of SSPxADD<5:0> and begins counting. The SDAx pin is released (brought high) for one BRG count (TBRG). When the BRG times out, and if SDAx is sampled high, the SCLx pin will be deasserted (brought high). When SCLx is sampled high, the BRG is reloaded with the contents of SSPxADD<6:0> and begins counting. SDAx and SCLx must be sampled high for one TBRG. This action is then followed by assertion of the SDAx pin (SDAx = 0) for one TBRG while SCLx is high. Following this, the RSEN bit (SSPxCON2<1>) will be automatically cleared and the BRG will not be reloaded, leaving the SDAx pin held low. As soon as a Start condition is detected on the SDAx and SCLx pins, the Start bit (SSPxSTAT<3>) will be set. The SSPxIF bit will not be set until the BRG has timed out.

Note 1: If RSEN is programmed while any other event is in progress, it will not take effect.

2: A bus collision during the Repeated Start condition occurs if:

- SDAx is sampled low when SCLx goes from low-to-high.
- SCLx goes low before SDAx is asserted low. This may indicate that another master is attempting to transmit a data '1'.

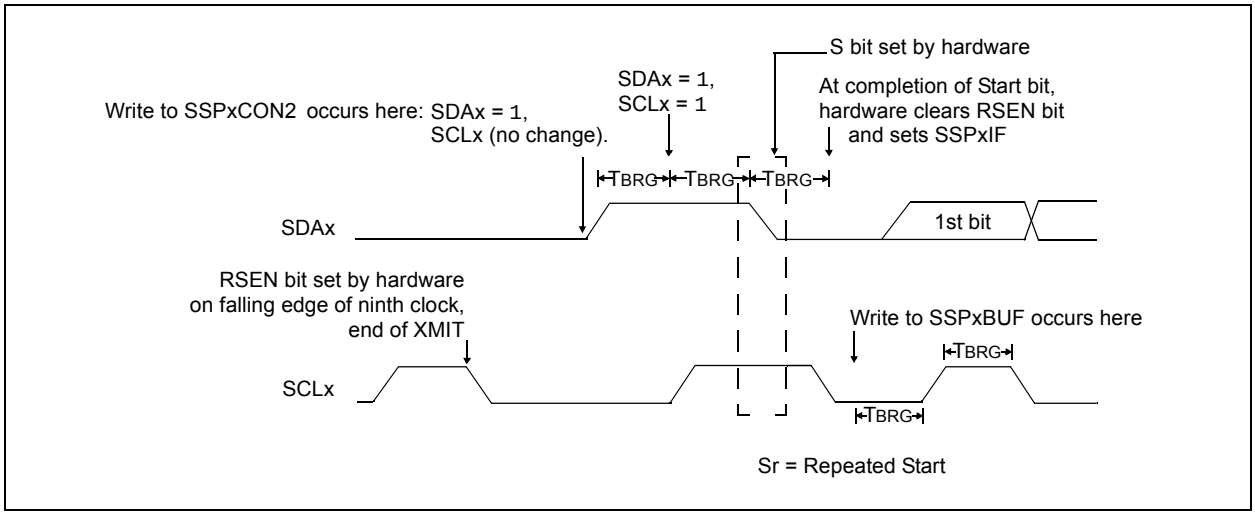
Immediately following the SSPxIF bit getting set, the user may write the SSPxBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional 8 bits of address (10-bit mode) or 8 bits of data (7-bit mode).

19.5.9.1 WCOL Status Flag

If the user writes the SSPxBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write does not occur).

Note: Because queueing of events is not allowed, writing of the lower five bits of SSPxCON2 is disabled until the Repeated Start condition is complete.

FIGURE 19-22: REPEATED START CONDITION WAVEFORM



19.5.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDAx when SCLx goes from a low level to a high level.

b) SCLx goes low before SDAx is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDAx and the pin is allowed to float high, the BRG is loaded with SSPxADD<6:0> and counts down to 0. The SCLx pin is then deasserted and when sampled high, the SDAx pin is sampled.

If SDAx is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', see Figure 19-31). If SDAx is sampled high, the BRG is reloaded and begins counting. If SDAx goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDAx at exactly the same time.

If SCLx goes from high-to-low before the BRG times out and SDAx has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (see Figure 19-32).

If, at the end of the BRG time-out, both SCLx and SDAx are still high, the SDAx pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCLx pin, the SCLx pin is driven low and the Repeated Start condition is complete.

FIGURE 19-31: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

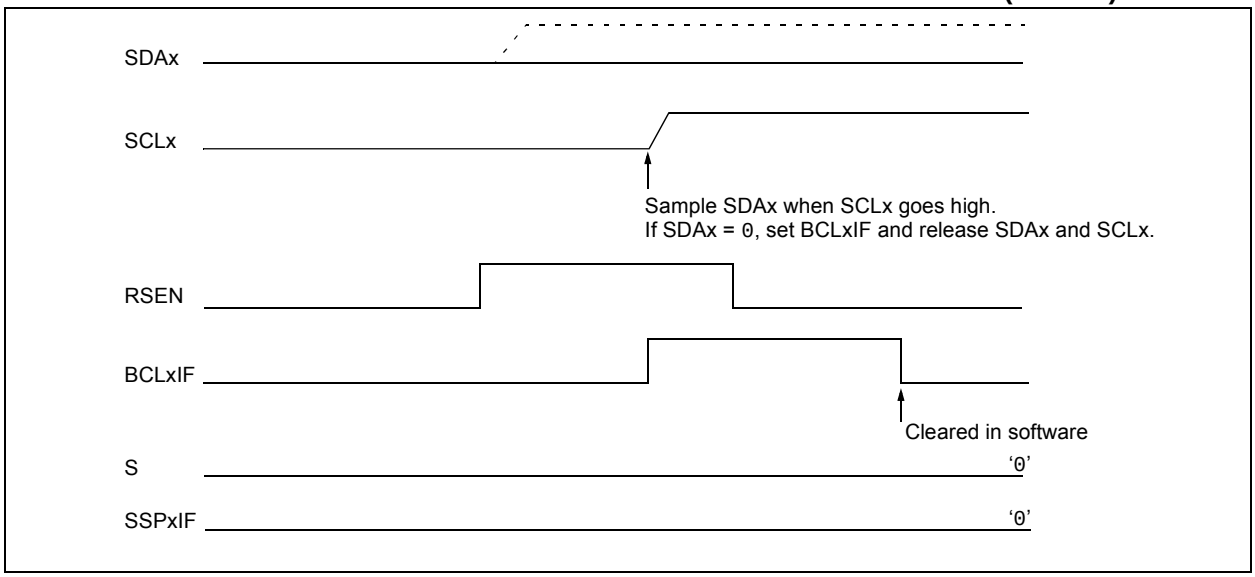
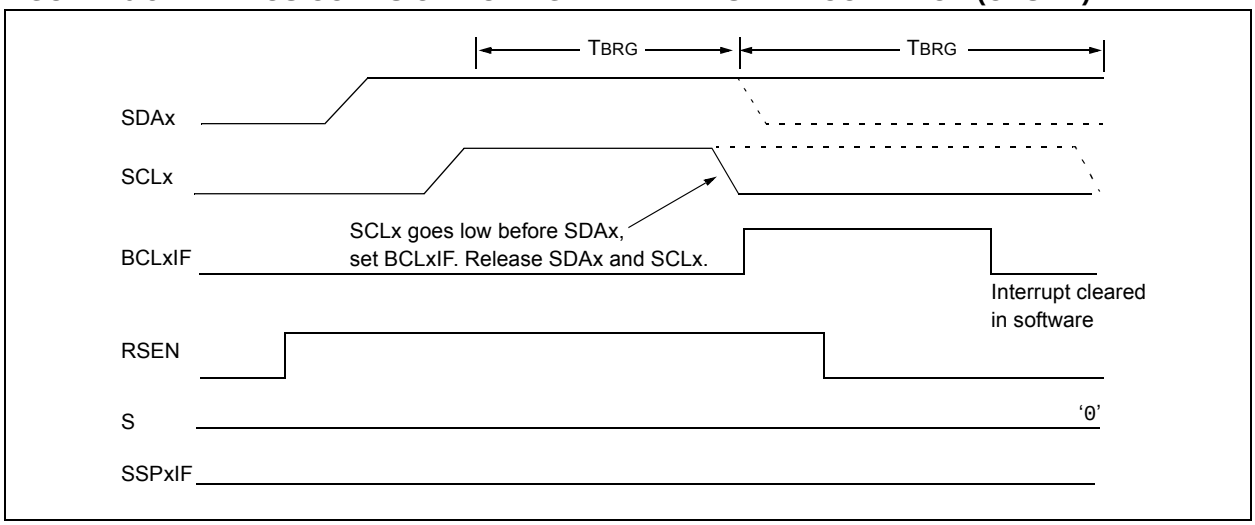


FIGURE 19-32: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



REGISTER 20-2: RCSTAx: RECEIVE STATUS AND CONTROL REGISTER (ACCESS FACh/F9Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
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 **SPEN:** Serial Port Enable bit
1 = Serial port enabled
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-Bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-Bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be cleared by reading RCREGx register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN). UART reception will be discarded until the overun error is cleared.
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

23.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register (Register 23-1). 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:

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 29-4 in **Section 29.0 "Electrical Characteristics"**).

EQUATION 23-1: CALCULATING OUTPUT OF THE COMPARATOR VOLTAGE REFERENCE

When CVRR = 1 and CVRSS = 0:
$CV_{REF} = ((CVR<3:0>)/24) \times (AV_{DD} - AV_{SS})$
When CVRR = 0 and CVRSS = 0:
$CV_{REF} = ((AV_{DD} - AV_{SS})/4) + ((CVR<3:0>)/32) \times (AV_{DD} - AV_{SS})$
When CVRR = 1 and CVRSS = 1:
$CV_{REF} = ((CVR<3:0>)/24) \times ((V_{REF}+) - V_{REF}-)$
When CVRR = 0 and CVRSS = 1:
$CV_{REF} = (((V_{REF}+) - V_{REF}-)/4) + ((CVR<3:0>)/32) \times ((V_{REF}+) - V_{REF}-)$

REGISTER 23-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER (BANKED F53h)

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 as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **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/C2INB pin
0 = CVREF voltage is disconnected from the RA2/AN2/VREF-/CVREF/C2INB 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 = AVDD – AVSS

bit 3-0 **CVR<3:0>:** Comparator VREF Value Selection bits ($0 \le (CVR<3:0>) \le 15$)
When CVRR = 1:
CVREF = ((CVR<3:0>)/24) • (CVRSRC)
When CVRR = 0:
CVREF = (CVRSRC/4) + ((CVR<3:0>)/32) • (CVRSRC)

Note 1: CVROE overrides the TRIS bit setting.

© 2011 Microchip Technology Inc.

27.1.1 STANDARD INSTRUCTION SET

ADDLW **ADD Literal to W**

Syntax: ADDLW k

Operands: $0 \le k \le 255$

Operation: $(W) + k \rightarrow W$

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

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of W are added to the 8-bit literal ‘k’ and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: ADDLW 0x15

Before Instruction
W = 10h
After Instruction
W = 25h

ADDWF **ADD W to f**

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

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

Operation: $(W) + (f) \rightarrow$ dest

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

Encoding:

0010	01da	ffff	ffff

Description: Add W to register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default).

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: ADDWF REG, 0, 0

Before Instruction
W = 17h
REG = 0C2h
After Instruction
W = 0D9h
REG = 0C2h

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

BNOV **Branch if Not Overflow**

Syntax: BNOV n

Operands: -128 ≤ n ≤ 127

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

Status Affected: None

Encoding:

1110	0101	nnnn	nnnn

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

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

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

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

If No Jump:

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

Example: HERE BNOV Jump

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

BNZ **Branch if Not Zero**

Syntax: BNZ n

Operands: -128 ≤ n ≤ 127

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

Status Affected: None

Encoding:

1110	0001	nnnn	nnnn

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

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

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

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

If No Jump:

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

Example: HERE BNZ Jump

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

© 2011 Microchip Technology Inc.

COMF **Complement f**

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

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

Operation: $\bar{f} \rightarrow$ dest

Status Affected: N, Z

Encoding:

0001	11da	ffff	ffff

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

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `COMF REG, 0, 0`

Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

CPFSEQ **Compare f with W, Skip if f = W**

Syntax: CPFSEQ f {,a}

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

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

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

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

Words: 1

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

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

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

If skip and followed by 2-word instruction:

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

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

© 2011 Microchip Technology Inc.

CPFSGT **Compare f with W, Skip if f > W**

Syntax: CPFSGT f {,a}

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

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	010a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.

If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

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

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

Words: 1

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

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

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

If skip and followed by 2-word instruction:

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

Example:
```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

CPFSLT **Compare f with W, Skip if f < W**

Syntax: CPFSLT f {,a}

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

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	000a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

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

Words: 1

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

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

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

If skip and followed by 2-word instruction:

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

Example:
```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

TBLRD **Table Read**

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

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT,
TBLPTR – No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT,
(TBLPTR) + 1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT,
(TBLPTR) – 1 → TBLPTR;
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR,
(Prog Mem (TBLPTR)) → TABLAT

Status Affected: None

Encoding:

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

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

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

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

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

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD **Table Read (Continued)**

Example 1: TBLRD *+

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example 2: TBLRD +*

Before Instruction
TABLAT = AAh
TBLPTR = 01A357h
MEMORY(01A357h) = 12h
MEMORY(01A358h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 01A358h

TSTFSZ **Test f, Skip if 0**

Syntax: TSTFSZ f {,a}

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

Operation: skip if f = 0

Status Affected: None

Encoding:

0110	011a	ffff	ffff

Description: If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction.

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

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

Words: 1

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

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

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

If skip and followed by 2-word instruction:

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

Example:
```
HERE    TSTFSZ  CNT, 1
NZERO   :
ZERO    :
```
Before Instruction
PC = Address (HERE)
After Instruction
If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)

XORLW **Exclusive OR Literal with W**

Syntax: XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → W

Status Affected: N, Z

Encoding:

0000	1010	kkkk	kkkk

Description: The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
XORLW   0xAF
```
Before Instruction
W = B5h
After Instruction
W = 1Ah