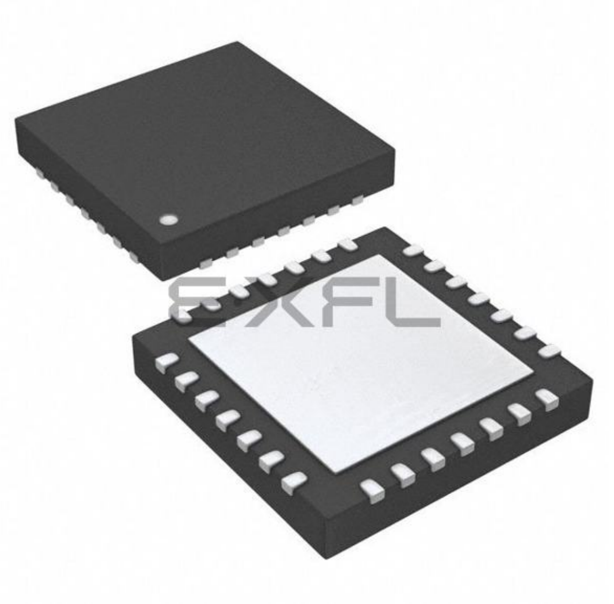


Welcome to E-XFL.COM

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	32MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PSMC, PWM, WDT
Number of I/O	24
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 11x12b; D/A 1x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-UFQFN Exposed Pad
Supplier Device Package	28-UQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1782-i-mv

TABLE 3-4: PIC16(L)F1782/3 MEMORY MAP (BANKS 8-31)

BANK 8	BANK 9	BANK 10	BANK 11	BANK 12	BANK 13	BANK 14	BANK 15
400h–40Bh: Core Registers (Table 3-2)	480h–48Bh: Core Registers (Table 3-2)	500h–50Bh: Core Registers (Table 3-2)	580h–58Bh: Core Registers (Table 3-2)	600h–60Bh: Core Registers (Table 3-2)	680h–68Bh: Core Registers (Table 3-2)	700h–70Bh: Core Registers (Table 3-2)	780h–78Bh: Core Registers (Table 3-2)
40Ch–46Fh: Unimplemented Read as '0'	48Ch–4EFh: Unimplemented Read as '0'	50Ch–510h: Unimplemented Read as '0' 511h: OPA1CON 512h: — 513h: OPA2CON 514h–519h: Unimplemented Read as '0' 51Ah: CLKRCON 51Bh–56Fh: Unimplemented Read as '0'	58Ch–5EFh: Unimplemented Read as '0'	60Ch–66Fh: Unimplemented Read as '0'	68Ch–6EFh: Unimplemented Read as '0'	70Ch–76Fh: Unimplemented Read as '0'	78Ch–7EFh: Unimplemented Read as '0'
470h–47Fh: Common RAM (Accesses 70h – 7Fh)	4F0h–4FFh: Common RAM (Accesses 70h – 7Fh)	570h–57Fh: Common RAM (Accesses 70h – 7Fh)	5F0h–5FFh: Common RAM (Accesses 70h – 7Fh)	670h–67Fh: Common RAM (Accesses 70h – 7Fh)	6F0h–6FFh: Common RAM (Accesses 70h – 7Fh)	770h–77Fh: Common RAM (Accesses 70h – 7Fh)	7F0h–7FFh: Common RAM (Accesses 70h – 7Fh)

BANK 16	BANK 17	BANK 18	BANK 19	BANK 20	BANK 21	BANK 22	BANK 23
800h–80Bh: Core Registers (Table 3-2)	880h–88Bh: Core Registers (Table 3-2)	900h–90Bh: Core Registers (Table 3-2)	980h–98Bh: Core Registers (Table 3-2)	A00h–A0Bh: Core Registers (Table 3-2)	A80h–A8Bh: Core Registers (Table 3-2)	B00h–B0Bh: Core Registers (Table 3-2)	B80h–B8Bh: Core Registers (Table 3-2)
80Ch–86Fh: See Table 3-5	88Ch–8EFh: Unimplemented Read as '0'	90Ch–96Fh: Unimplemented Read as '0'	98Ch–9EFh: Unimplemented Read as '0'	A0Ch–A6Fh: Unimplemented Read as '0'	A8Ch–AEFh: Unimplemented Read as '0'	B0Ch–B6Fh: Unimplemented Read as '0'	B8Ch–BEFh: Unimplemented Read as '0'
870h–87Fh: Common RAM (Accesses 70h – 7Fh)	8F0h–8FFh: Common RAM (Accesses 70h – 7Fh)	970h–97Fh: Common RAM (Accesses 70h – 7Fh)	9F0h–9FFh: Common RAM (Accesses 70h – 7Fh)	A70h–A7Fh: Common RAM (Accesses 70h – 7Fh)	AF0h–AFFh: Common RAM (Accesses 70h – 7Fh)	B70h–B7Fh: Common RAM (Accesses 70h – 7Fh)	BF0h–BFFh: Common RAM (Accesses 70h – 7Fh)

BANK 24	BANK 25	BANK 26	BANK 27	BANK 28	BANK 29	BANK 30	BANK 31
C00h–C0Bh: Core Registers (Table 3-2)	C80h–C8Bh: Core Registers (Table 3-2)	D00h–D0Bh: Core Registers (Table 3-2)	D80h–D8Bh: Core Registers (Table 3-2)	E00h–E0Bh: Core Registers (Table 3-2)	E80h–E8Bh: Core Registers (Table 3-2)	F00h–F0Bh: Core Registers (Table 3-2)	F80h–F8Bh: Core Registers (Table 3-2)
C0Ch–C6Fh: Unimplemented Read as '0'	C8Ch–CEFh: Unimplemented Read as '0'	D0Ch–D6Fh: Unimplemented Read as '0'	D8Ch–DEFh: Unimplemented Read as '0'	E0Ch–E6Fh: Unimplemented Read as '0'	E8Ch–EEFh: Unimplemented Read as '0'	F0Ch–F6Fh: Unimplemented Read as '0'	F8Ch–FEFh: See Table 3-6
C70h–C7Fh: Common RAM (Accesses 70h – 7Fh)	CF0h–CFFh: Common RAM (Accesses 70h – 7Fh)	D70h–D7Fh: Common RAM (Accesses 70h – 7Fh)	DF0h–DFFh: Common RAM (Accesses 70h – 7Fh)	E70h–E7Fh: Common RAM (Accesses 70h – 7Fh)	EF0h–EFFh: Common RAM (Accesses 70h – 7Fh)	F70h–F7Fh: Common RAM (Accesses 70h – 7Fh)	FF0h–FFFh: Common RAM (Accesses 70h – 7Fh)

Legend: = Unimplemented data memory locations, read as '0'

TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 16 (Continued)											
831h	PSMC2CON	PSMC2EN	PSMC2LD	PSMC2DBFE	PSMC2DBRE	P2MODE<3:0>				0000 0000	0000 0000
832h	PSMC2MDL	P2MDLEN	P2MDLPOL	P2MDLBIT	—	P2MSRC<3:0>				000- 0000	000- 0000
833h	PSMC2SYNC	—	—	—	—	—	—	P2SYNC<1:0>		---- --00	---- --00
834h	PSMC2CLK	—	—	P2CPRE<1:0>		—	—	P2CSRC<1:0>		--00 --00	--00 --00
835h	PSMC2OEN	—	—	—	—	—	—	P2OEB	P2OEA	---- --00	---- --00
836h	PSMC2POL	—	P2INPOL	—	—	—	—	P2POLB	P2POLA	-0-- --00	-0-- --00
837h	PSMC2BLNK	—	—	P2FEBM<1:0>		—	—	P2REBM<1:0>		--00 --00	--00 --00
838h	PSMC2REBS	P2REBIN	—	—	—	P2REBSC3	P2REBSC2	P2REBSC1	—	0--- 000-	0--- 000-
839h	PSMC2FEBS	P2FEBIN	—	—	—	P2FEBSC3	P2FEBSC2	P2FEBSC1	—	0--- 000-	0--- 000-
83Ah	PSMC2PHS	P2PHSIN	—	—	—	P2PHSC3	P2PHSC2	P2PHSC1	P2PHST	0--- 0000	0--- 0000
83Bh	PSMC2DCS	P2DCSIN	—	—	—	P2DCSC3	P2DCSC2	P2DCSC1	P2DCST	0--- 0000	0--- 0000
83Ch	PSMC2PRS	P2PRSIN	—	—	—	P2PRSC3	P2PRSC2	P2PRSC1	P2PRST	0--- 0000	0--- 0000
83Dh	PSMC2ASDC	P2ASE	P2ASDEN	P2ARSEN	—	—	—	—	P2ASDOV	000- ---0	000- ---0
83Eh	PSMC2ASDL	—	—	P2ASDLF	P2ASDLE	P2ASDLD	P2ASDLC	P2ASDLB	P2ASDLA	--00 0000	--00 0000
83Fh	PSMC2ASDS	P2ASDSIN	—	—	—	P2ASDSC3	P2ASDSC2	P2ASDSC1	—	0--- 000-	0--- 000-
840h	PSMC2INT	P2TOVIE	P2TPHIE	P2TDCIE	P2TPRIE	P2TOVIF	P2TPHIF	P2TDCIF	P2TPRIF	0000 0000	0000 0000
841h	PSMC2PHL	Phase Low Count								0000 0000	0000 0000
842h	PSMC2PHH	Phase High Count								0000 0000	0000 0000
843h	PSMC2DCL	Duty Cycle Low Count								0000 0000	0000 0000
844h	PSMC2DCH	Duty Cycle High Count								0000 0000	0000 0000
845h	PSMC2PRL	Period Low Count								0000 0000	0000 0000
846h	PSMC2PRH	Period High Count								0000 0000	0000 0000
847h	PSMC2TMRL	Time base Low Counter								0000 0001	0000 0001
848h	PSMC2TMRH	Time base High Counter								0000 0000	0000 0000
849h	PSMC2DBR	rising Edge Dead-band Counter								0000 0000	0000 0000
84Ah	PSMC2DBF	Falling Edge Dead-band Counter								0000 0000	0000 0000
84Bh	PSMC2BLKR	rising Edge Blanking Counter								0000 0000	0000 0000
84Ch	PSMC2BLKF	Falling Edge Blanking Counter								0000 0000	0000 0000
84Dh	PSMC2FFA	—	—	—	—	Fractional Frequency Adjust Register				---- 0000	---- 0000
84Eh	PSMC2STR0	—	—	—	—	—	—	P2STRB	P2STRA	---- --01	---- --01
84Fh	PSMC2STR1	P2SYNC	—	—	—	—	—	P2LSMEN	P2HSMEN	0--- --00	0--- --00
850h — 86Fh	—	Unimplemented								—	—
Bank 17-30											
x0Ch or x8Ch to x1Fh or x9Fh	—	Unimplemented								—	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as ‘0’, r = reserved. Shaded locations are unimplemented, read as ‘0’.

Note 1: These registers can be addressed from any bank.

2: Unimplemented, read as ‘1’.

3: PIC16F1782/3 only.

TABLE 5-5: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BORCON	SBOREN	BORFS	—	—	—	—	—	BORRDY	47
PCON	STKOVF	STKUNF	—	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	51
STATUS	—	—	—	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	18
WDTCON	—	—	WDTPS<4:0>					SWDTEN	94

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by Resets.

6.2 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (EC mode), quartz crystal resonators or ceramic resonators (LP, XT and HS modes) and Resistor-Capacitor (RC) mode circuits.

Internal clock sources are contained within the oscillator module. The internal oscillator block has two internal oscillators and a dedicated Phase-Lock Loop (HFPLL) that are used to generate three internal system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC), 500 kHz (MFINTOSC) and the 31 kHz Low-Frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS) bits in the OSCCON register. See **Section 6.3 "Clock Switching"** for additional information.

6.2.1 EXTERNAL CLOCK SOURCES

An external clock source can be used as the device system clock by performing one of the following actions:

- Program the FOSC<2:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Write the SCS<1:0> bits in the OSCCON register to switch the system clock source to:
  - Timer1 oscillator during run-time, or
  - An external clock source determined by the value of the FOSC bits.

See **Section 6.3 "Clock Switching"**for more information.

6.2.1.1 EC Mode

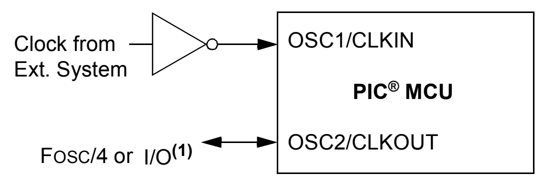
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. Figure 6-2 shows the pin connections for EC mode.

EC mode has three power modes to select from through Configuration Words:

- High power, 4-32 MHz (FOSC = `111`)
- Medium power, 0.5-4 MHz (FOSC = `110`)
- Low power, 0-0.5 MHz (FOSC = `101`)

The Oscillator Start-up Timer (OST) is disabled when EC mode is selected. Therefore, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

FIGURE 6-2: EXTERNAL CLOCK (EC) MODE OPERATION



Note 1: Output depends upon $\overline{\text{CLKOUTEN}}$ bit of the Configuration Words.

6.2.1.2 LP, XT, HS Modes

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 6-3). The three modes select a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification.

HS Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require a high drive setting.

Figure 6-3 and Figure 6-4 show typical circuits for quartz crystal and ceramic resonators, respectively.

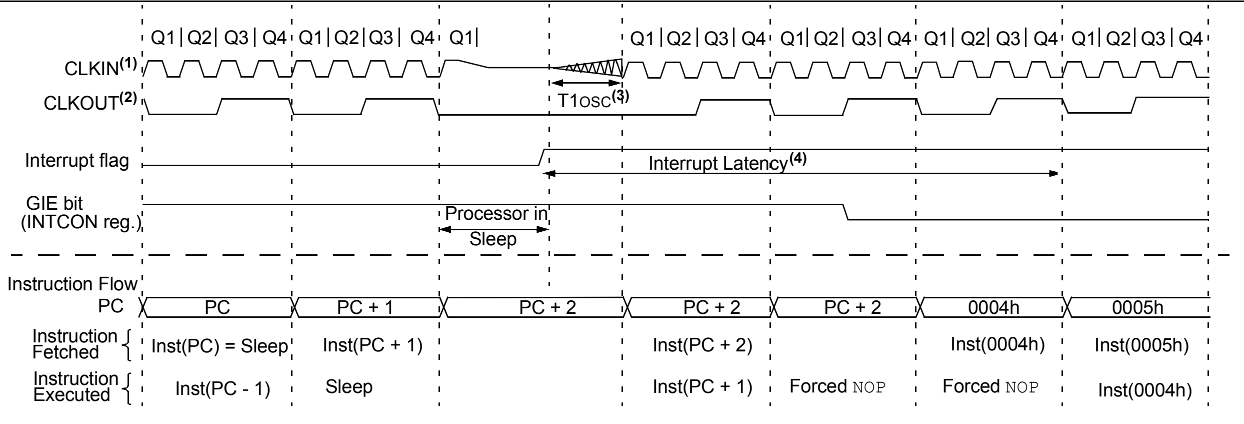
9.1.1 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a SLEEP instruction
  - SLEEP instruction will execute as a NOP.
  - WDT and WDT prescaler will not be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will not be set
  - $\overline{\text{PD}}$ bit of the STATUS register will not be cleared.
- If the interrupt occurs **during or after** the execution of a SLEEP instruction
  - SLEEP instruction will be completely executed
  - Device will immediately wake-up from Sleep
  - WDT and WDT prescaler will be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will be set
  - $\overline{\text{PD}}$ bit of the STATUS register will be cleared.

Even if the flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{\text{PD}}$ bit. If the $\overline{\text{PD}}$ bit is set, the SLEEP instruction was executed as a NOP.

FIGURE 9-1: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: External clock. High, Medium, Low mode assumed.
2: CLKOUT is shown here for timing reference.
3: T1OSC; See **Section 30.0 "Electrical Specifications"**.
4: GIE = 1 assumed. In this case after wake-up, the processor calls the ISR at 0004h. If GIE = 0, execution will continue in-line.

12.6 Write Verify

Depending on the application, good programming practice may dictate that the value written to the data EEPROM or program memory should be verified (see Example 12-6) to the desired value to be written. Example 12-6 shows how to verify a write to EEPROM.

EXAMPLE 12-6: EEPROM WRITE VERIFY

```
BANKSEL EEDATL      ;
MOVF    EEDATL, W   ;EEDATL not changed
                    ;from previous write
BSF     EECON1, RD  ;YES, Read the
                    ;value written
XORWF   EEDATL, W   ;
BTFSS   STATUS, Z   ;Is data the same
GOTO    WRITE_ERR   ;No, handle error
:                   ;Yes, continue
```


13.5 PORTB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB (Register 13-11). 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., enable the output driver and put the contents of the output latch on the selected pin). Example 13-1 shows how to initialize an I/O port.

Reading the PORTB register (Register 13-10) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATB).

13.5.1 DIRECTION CONTROL

The TRISB register (Register 13-11) controls the PORTB pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISB register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

13.5.2 OPEN DRAIN CONTROL

The ODCONB register (Register 13-15) controls the open-drain feature of the port. Open drain operation is independently selected for each pin. When an ODCONB bit is set, the corresponding port output becomes an open drain driver capable of sinking current only. When an ODCONB bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

13.5.3 SLEW RATE CONTROL

The SLRCONB register (Register 13-16) controls the slew rate option for each port pin. Slew rate control is independently selectable for each port pin. When an SLRCONB bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONB bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

13.5.4 INPUT THRESHOLD CONTROL

The INLVLB register (Register 13-17) controls the input voltage threshold for each of the available PORTB input pins. A selection between the Schmitt Trigger CMOS or the TTL Compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTB register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See **Section TABLE 30-1: "Supply Voltage"** for more information on threshold levels.

Note:	Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

13.5.5 ANALOG CONTROL

The ANSELB register (Register 13-13) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELB bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELB bits has no effect on digital output functions. A pin with TRIS clear and ANSELB set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

Note:	The ANSELB bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.

TABLE 13-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	121
INLVLB	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0	122
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	120
ODCONB	ODB7	ODB6	ODB5	ODB4	ODB3	ODB2	ODB1	ODB0	122
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	120
SLRCONB	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0	122
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	120
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	121

Legend: x = unknown, u = unchanged, - = unimplemented locations read as ‘0’. Shaded cells are not used by PORTB.

14.6 Register Definitions: Interrupt-on-Change Control

REGISTER 14-1: IOCxP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
IOCxP7	IOCxP6	IOCxP5	IOCxP4	IOCxP3	IOCxP2	IOCxP1	IOCxP0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **IOCxP<7:0>:** Interrupt-on-Change Positive Edge Enable bits(1)
1 = Interrupt-on-Change enabled on the pin for a positive going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin.

Note 1: For IOCEP register, bit 3 (IOCEP3) is the only implemented bit in the register.

REGISTER 14-2: IOCxN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
IOCxN7	IOCxN6	IOCxN5	IOCxN4	IOCxN3	IOCxN2	IOCxN1	IOCxN0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **IOCxN<7:0>:** Interrupt-on-Change Negative Edge Enable bits(1)
1 = Interrupt-on-Change enabled on the pin for a negative going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin.

Note 1: For IOCEN register, bit 3 (IOCEN3) is the only implemented bit in the register.

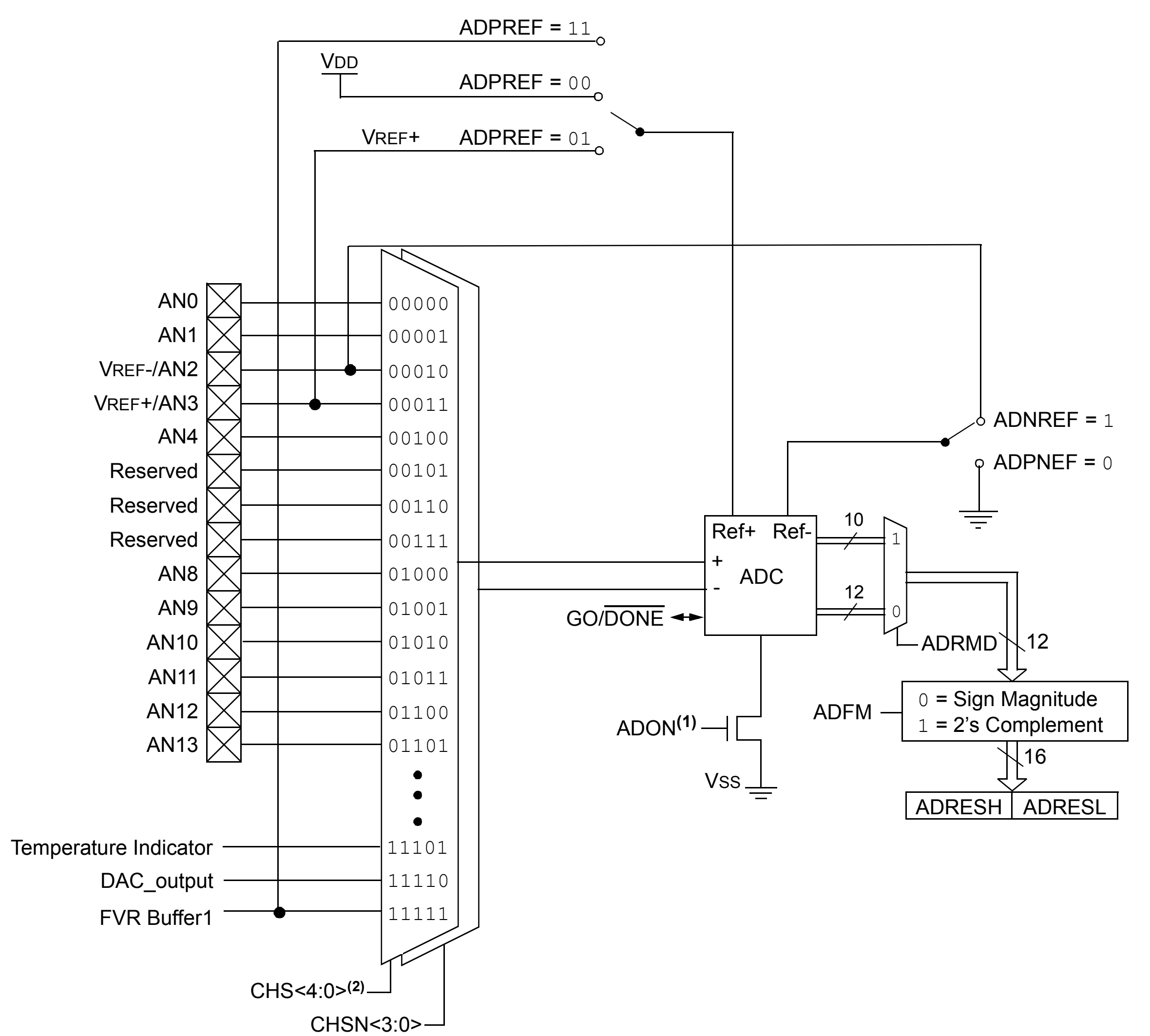
17.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

The Analog-to-Digital Converter (ADC) allows conversion of a single-ended and differential analog input signals to a 12-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 12-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair). Figure 17-1 shows the block diagram of the ADC.

The ADC voltage reference is software selectable to be either internally generated or externally supplied.

The ADC can generate an interrupt upon completion of a conversion. This interrupt can be used to wake-up the device from Sleep.

FIGURE 17-1: ADC BLOCK DIAGRAM



Note 1: When ADON = 0, all multiplexer inputs are disconnected.

2: See ADCON0 register (Register 17-1) and ADCON2 register (Register 17-3) for detailed analog channel selection per device.

19.4 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the DACCON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

19.5 Effects of a Reset

A device Reset affects the following:

- DAC is disabled.
- DAC output voltage is removed from the DACOUT pin.
- The DACR<7:0> range select bits are cleared.

24.3.3 PUSH-PULL PWM

The push-pull PWM is used to drive transistor bridge circuits. It uses at least two outputs and generates PWM signals that alternate between the two outputs in even and odd cycles.

Variations of the push-pull waveform include four outputs with two outputs being complementary or two sets of two identical outputs. Refer to Sections 24.3.4 through 24.3.6 for the other Push-Pull modes.

24.3.3.1 Mode Features

- No dead-band control available
- No steering control available
- Output is on the following two pins only:
  - PSMCxA
  - PSMCxB

Note:	This is a subset of the 6-pin output of the push-pull PWM output, which is why pin functions are fixed in these positions, so they are compatible with that mode. See **Section 24.3.6 "Push-Pull PWM with Four Full-Bridge and Complementary Outputs"**

24.3.3.2 Waveform Generation

Odd numbered period rising edge event:

- PSMCxA is set active

Odd numbered period falling edge event:

- PSMCxA is set inactive

Even numbered period rising edge event:

- PSMCxB is set active

Even numbered period falling edge event:

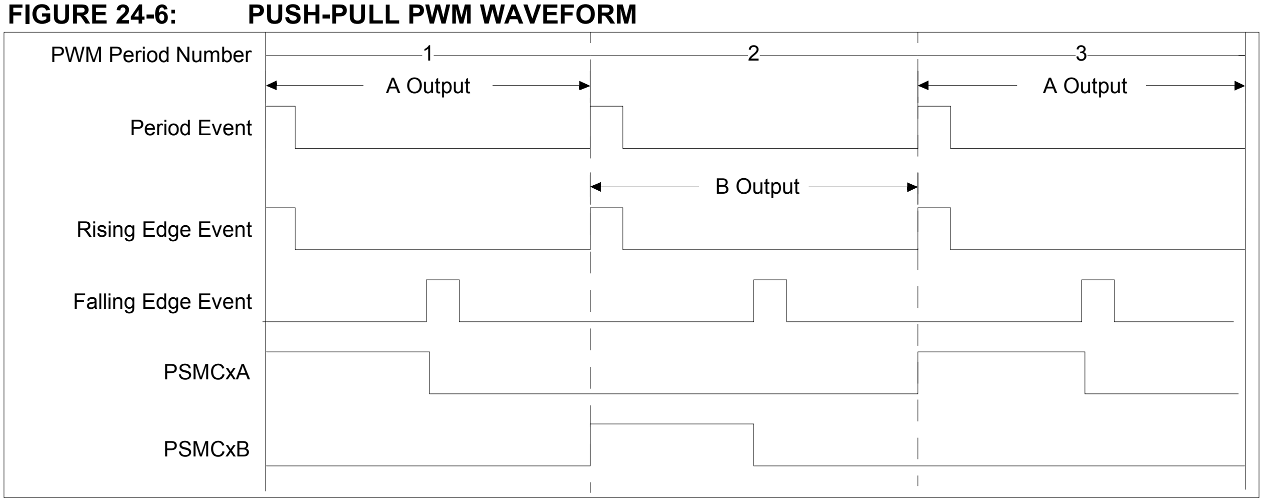
- PSMCxB is set inactive

Code for setting up the PSMC generate the complementary single-phase waveform shown in Figure 24-6, and given in Example 24-3.

EXAMPLE 24-3: PUSH-PULL SETUP

```
; Push-Pull PWM PSMC setup
; Fully synchronous operation
; Period = 10 us
; Duty cycle = 50% (25% each phase)
   BANKSEL  PSMC1CON
   MOVLW    0x02       ; set period
   MOVWF    PSMC1PRH
   MOVLW    0x7F
   MOVWF    PSMC1PRL
   MOVLW    0x01       ; set duty cycle
   MOVWF    PSMC1DCH
   MOVLW    0x3F
   MOVWF    PSMC1DCL
   CLRF     PSMC1PHH   ; no phase offset
   CLRF     PSMC1PHL
   MOVLW    0x01       ; PSMC clock=64 MHz
   MOVWF    PSMC1CLK
; output on A and B, normal polarity
   MOVLW    B'00000011'
   MOVWF    PSMC1OEN
   CLRF     PSMC1POL
; set time base as source for all events
   BSF      PSMC1PRS, P1PRST
   BSF      PSMC1PHS, P1PHST
   BSF      PSMC1DCS, P1DCST
; enable PSMC in Push-Pull Mode
; this also loads steering and time buffers
   MOVLW    B'11000010'
   MOVWF    PSMC1CON
   BANKSEL  TRISC
   BCF      TRISC, 0   ; enable pin drivers
   BCF      TRISC, 1
```

FIGURE 24-6: PUSH-PULL PWM WAVEFORM



24.3.7 PULSE-SKIPPING PWM

The pulse-skipping PWM is used to generate a series of fixed-length pulses that can be triggered at each period event. A rising edge event will be generated when any enabled asynchronous rising edge input is active when the period event occurs, otherwise no event will be generated.

The rising edge event occurs based upon the value in the PSMCxPH register pair.

The falling edge event always occurs according to the enabled event inputs without qualification between any two inputs.

24.3.7.1 Mode Features

- No dead-band control available
- No steering control available
- PWM is output to only one pin:
  - PSMCxA

24.3.7.2 Waveform Generation

Rising Edge Event

If any enabled asynchronous rising edge event = `1` when there is a period event, then upon the next synchronous rising edge event:

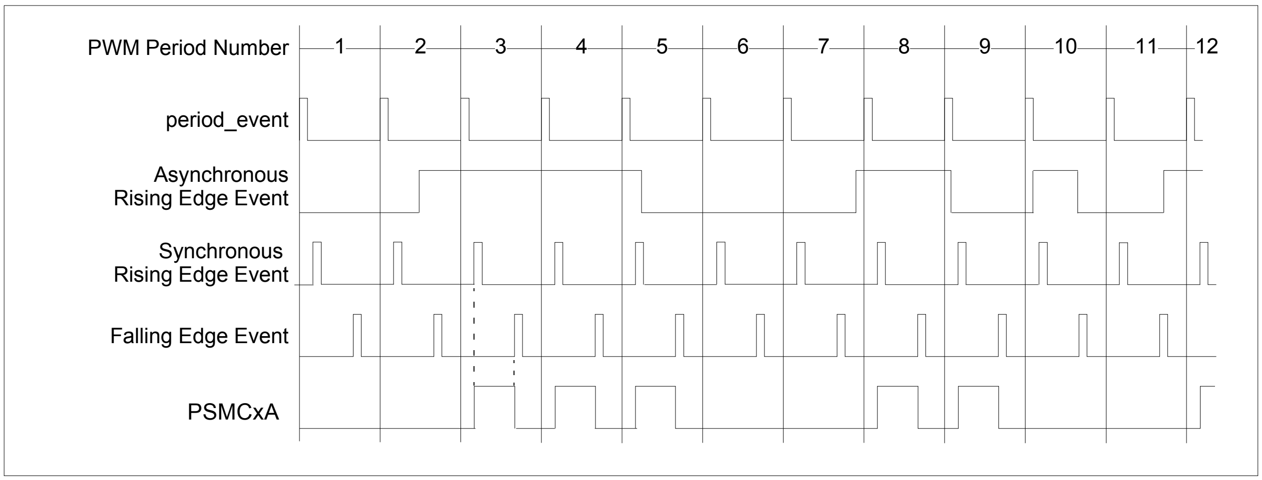
- PSMCxA is set active

Falling Edge Event

- PSMCxA is set inactive

Note:	To use this mode, an external source must be used for the determination of whether or not to generate the set pulse. If the phase time base is used, it will either always generate a pulse or never generate a pulse based on the PSMCxPH value.

FIGURE 24-10: PULSE-SKIPPING PWM WAVEFORM



REGISTER 24-19: PSMCxPHL: PSMC PHASE COUNT LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
PSMCxPHL<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **PSMCxPHL<7:0>:** 16-bit Phase Count Least Significant bits
= PSMCxPH<7:0>

REGISTER 24-20: PSMCxPHH: PSMC PHASE COUNT HIGH BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
PSMCxPHH<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **PSMCxPHH<7:0>:** 16-bit Phase Count Most Significant bits
= PSMCxPH<15:8>

25.3.2 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for standard PWM operation:

1. Disable the CCPx pin output driver by setting the associated TRIS bit.
2. Load the PR2 register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register and the DCxBx bits of the CCPxCON register, with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the PIRx register. See Note below.
   - Configure the T2CKPS bits of the T2CON register with the Timer prescale value.
   - Enable the Timer by setting the TMR2ON bit of the T2CON register.
6. Enable PWM output pin:
   - Wait until the Timer overflows and the TMR2IF bit of the PIR1 register is set. See Note below.
   - Enable the CCPx pin output driver by clearing the associated TRIS bit.

Note:	In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

25.3.3 TIMER2 TIMER RESOURCE

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

25.3.4 PWM PERIOD

The PWM period is specified by the PR2 register of Timer2. The PWM period can be calculated using the formula of Equation 25-1.

EQUATION 25-1: PWM PERIOD

$$PWM\ Period = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

Note 1: TOSC = 1/FOSC

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is latched from CCPRxL into CCPRxH.

Note:	The Timer postscaler (see **Section 23.1 "Timer2 Operation"**) is not used in the determination of the PWM frequency.

25.3.5 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to multiple registers: CCPRxL register and DCxB<1:0> bits of the CCPxCON register. The CCPRxL contains the eight MSbs and the DCxB<1:0> bits of the CCPxCON register contain the two LSbs. CCPRxL and DCxB<1:0> bits of the CCPxCON register can be written to at any time. The duty cycle value is not latched into CCPRxH until after the period completes (i.e., a match between PR2 and TMR2 registers occurs). While using the PWM, the CCPRxH register is read-only.

Equation 25-2 is used to calculate the PWM pulse width.

Equation 25-3 is used to calculate the PWM duty cycle ratio.

EQUATION 25-2: PULSE WIDTH

$$Pulse\ Width = (CCPRxL{:}CCPxCON{<}5{:}4{>}) \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

EQUATION 25-3: DUTY CYCLE RATIO

$$Duty\ Cycle\ Ratio = \frac{(CCPRxL{:}CCPxCON{<}5{:}4{>})}{4(PR2 + 1)}$$

The CCPRxH register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

26.5.2 SLAVE RECEPTION

When the R/$\overline{\text{W}}$ bit of a matching received address byte is clear, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and acknowledged.

When the overflow condition exists for a received address, then not Acknowledge is given. An overflow condition is defined as either bit BF of the SSPSTAT register is set, or bit SSPOV of the SSPCON1 register is set. The BOEN bit of the SSPCON3 register modifies this operation. For more information see Register 26-4.

An MSSP interrupt is generated for each transferred data byte. Flag bit, SSP1IF, must be cleared by software.

When the SEN bit of the SSPCON2 register is set, SCL will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit of the SSPCON1 register, except sometimes in 10-bit mode. See **Section 26.2.3 “SPI Master Mode”** for more detail.

26.5.2.1 7-bit Addressing Reception

This section describes a standard sequence of events for the MSSP module configured as an I^{2}C Slave in 7-bit Addressing mode. All decisions made by hardware or software and their effect on reception. Figure 26-13 and Figure 26-14 is used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish I^{2}C communication.

1. Start bit detected.
2. S bit of SSPSTAT is set; SSP1IF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{\text{W}}$ bit clear is received.
4. The slave pulls SDA low sending an $\overline{\text{ACK}}$ to the master, and sets SSP1IF bit.
5. Software clears the SSP1IF bit.
6. Software reads received address from SSPBUF clearing the BF flag.
7. If SEN = `1`; Slave software sets CKP bit to release the SCL line.
8. The master clocks out a data byte.
9. Slave drives SDA low sending an $\overline{\text{ACK}}$ to the master, and sets SSP1IF bit.
10. Software clears SSP1IF.
11. Software reads the received byte from SSPBUF clearing BF.
12. Steps 8-12 are repeated for all received bytes from the master.
13. Master sends Stop condition, setting P bit of SSPSTAT, and the bus goes idle.

26.5.2.2 7-bit Reception with AHEN and DHEN

Slave device reception with AHEN and DHEN set operate the same as without these options with extra interrupts and clock stretching added after the 8th falling edge of SCL. These additional interrupts allow the slave software to decide whether it wants to $\overline{\text{ACK}}$ the receive address or data byte, rather than the hardware. This functionality adds support for PMBus™ that was not present on previous versions of this module.

This list describes the steps that need to be taken by slave software to use these options for I^{2}C communication. Figure 26-15 displays a module using both address and data holding. Figure 26-16 includes the operation with the SEN bit of the SSPCON2 register set.

1. S bit of SSPSTAT is set; SSP1IF is set if interrupt on Start detect is enabled.
2. Matching address with R/$\overline{\text{W}}$ bit clear is clocked in. SSP1IF is set and CKP cleared after the 8th falling edge of SCL.
3. Slave clears the SSP1IF.
4. Slave can look at the ACKTIM bit of the SSPCON3 register to determine if the SSP1IF was after or before the $\overline{\text{ACK}}$.
5. Slave reads the address value from SSPBUF, clearing the BF flag.
6. Slave sets $\overline{\text{ACK}}$ value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSP1IF is set after an $\overline{\text{ACK}}$, not after a NACK.
9. If SEN = `1` the slave hardware will stretch the clock after the $\overline{\text{ACK}}$.
10. Slave clears SSP1IF.

> **Note:** SSP1IF is still set after the 9th falling edge of SCL even if there is no clock stretching and BF has been cleared. Only if NACK is sent to master is SSP1IF not set

11. SSP1IF set and CKP cleared after 8th falling edge of SCL for a received data byte.
12. Slave looks at ACKTIM bit of SSPCON3 to determine the source of the interrupt.
13. Slave reads the received data from SSPBUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an $\overline{\text{ACK}}$ = `1`, or the master sending a Stop condition. If a Stop is sent and Interrupt on Stop Detect is disabled, the slave will only know by polling the P bit of the SSTSTAT register.

26.8 Register Definitions: MSSP Control

REGISTER 26-1: SSPSTAT: SSP STATUS REGISTER

R/W-0/0	R/W-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7 **SMP:** SPI Data Input Sample bit

SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time

SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode

In I^{2}C Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high speed mode (400 kHz)

bit 6 **CKE:** SPI Clock Edge Select bit (SPI mode only)

In SPI Master or Slave mode:
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

In I^{2}C™ mode only:
1 = Enable input logic so that thresholds are compliant with SMBus specification
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{\text{A}}$:** Data/$\overline{\text{Address}}$ bit (I^{2}C mode only)
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit
(I^{2}C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a Stop bit has been detected last (this bit is ‘0’ on Reset)
0 = Stop bit was not detected last

bit 3 **S:** Start bit
(I^{2}C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a Start bit has been detected last (this bit is ‘0’ on Reset)
0 = Start bit was not detected last

bit 2 **R/$\overline{\text{W}}$:** Read/$\overline{\text{Write}}$ bit information (I^{2}C mode only)
This bit holds the R/$\overline{\text{W}}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit, or not ACK bit.

In I^{2}C Slave mode:
1 = Read
0 = Write

In I^{2}C Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress
OR-ing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode.

bit 1 **UA:** Update Address bit (10-bit I^{2}C mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

REGISTER 26-3: SSPCON2: SSP CONTROL REGISTER 2

R/W-0/0	R-0/0	R/W-0/0	R/S/HS-0/0	R/S/HS-0/0	R/S/HS-0/0	R/S/HS-0/0	R/W/HS-0/0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HC = Cleared by hardware S = User set

bit 7 **GCEN:** General Call Enable bit (in I²C Slave mode only)
1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (in I²C mode only)
1 = Acknowledge was not received
0 = Acknowledge was received

bit 5 **ACKDT:** Acknowledge Data bit (in I²C mode only)
In Receive mode:
Value transmitted when the user initiates an Acknowledge sequence at the end of a receive
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (in I²C Master mode only)
In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

bit 3 **RCEN:** Receive Enable bit (in I²C Master mode only)
1 = Enables Receive mode for I²C
0 = Receive idle

bit 2 **PEN:** Stop Condition Enable bit (in I²C Master mode only)
SCKMSSP Release Control:
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit (in I²C Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enable/Stretch Enable bit
In Master mode:
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is disabled

Note 1: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I²C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	`0` → (f<b>)
Status Affected:	None
Description:	Bit ‘b’ in register ‘f’ is cleared.

BRA	Relative Branch
Syntax:	[*label*] BRA label [*label*] BRA $+k
Operands:	$-256 \leq \text{label} - PC + 1 \leq 255$ $-256 \leq k \leq 255$
Operation:	(PC) + 1 + k → PC
Status Affected:	None
Description:	Add the signed 9-bit literal ‘k’ to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + k. This instruction is a 2-cycle instruction. This branch has a limited range.

BRW	Relative Branch with W
Syntax:	[*label*] BRW
Operands:	None
Operation:	(PC) + (W) → PC
Status Affected:	None
Description:	Add the contents of W (unsigned) to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + (W). This instruction is a 2-cycle instruction.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	`1` → (f<b>)
Status Affected:	None
Description:	Bit ‘b’ in register ‘f’ is set.

BTFSC	Bit Test f, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	skip if (f<b>) = `0`
Status Affected:	None
Description:	If bit ‘b’ in register ‘f’ is ‘`1`’, the next instruction is executed. If bit ‘b’, in register ‘f’, is ‘`0`’, the next instruction is discarded, and a `NOP` is executed instead, making this a 2-cycle instruction.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b < 7$
Operation:	skip if (f<b>) = `1`
Status Affected:	None
Description:	If bit ‘b’ in register ‘f’ is ‘`0`’, the next instruction is executed. If bit ‘b’ is ‘`1`’, then the next instruction is discarded and a `NOP` is executed instead, making this a 2-cycle instruction.

LSLF	**Logical Left Shift**
Syntax:	[*label*] LSLF f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<7>) → C (f<6:0>) → dest<7:1> 0 → dest<0>
Status Affected:	C, Z
Description:	The contents of register ‘f’ are shifted one bit to the left through the Carry flag. A ‘`0`’ is shifted into the LSb. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is stored back in register ‘f’.

LSRF	**Logical Right Shift**
Syntax:	[*label*] LSRF f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	0 → dest<7> (f<7:1>) → dest<6:0>, (f<0>) → C,
Status Affected:	C, Z
Description:	The contents of register ‘f’ are shifted one bit to the right through the Carry flag. A ‘`0`’ is shifted into the MSb. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘`1`’, the result is stored back in register ‘f’.

MOVF	**Move f**
Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [`0`,`1`]
Operation:	(f) → (dest)
Status Affected:	Z
Description:	The contents of register f is moved to a destination dependent upon the status of d. If d = `0`, destination is W register. If d = `1`, the destination is file register f itself. d = `1` is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1
Example:	`MOVF FSR, 0`
	After Instruction W = value in FSR register Z = `1`