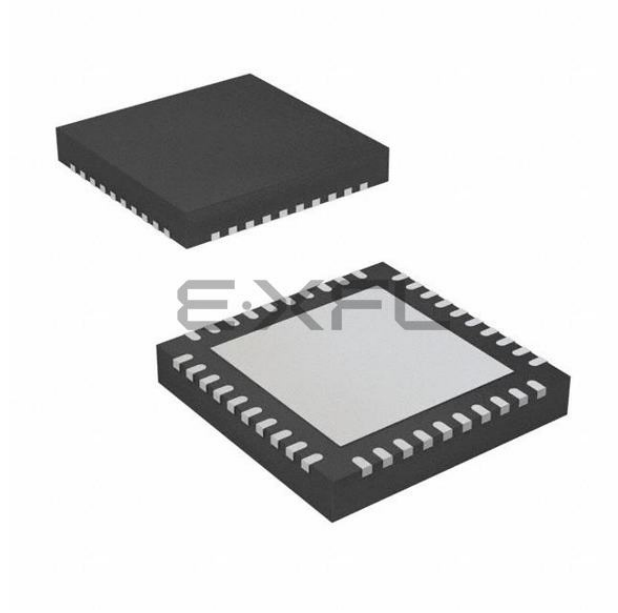


Welcome to E-XFL.COM

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	35
Program Memory Size	28KB (16K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 14x12b; D/A 1x8b, 3x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	40-UFQFN Exposed Pad
Supplier Device Package	40-UQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f1789-i-mv

PIN ALLOCATION TABLE

TABLE 1: 28-PIN ALLOCATION TABLE (PIC16(L)F1788)

I/O	28-Pin SPDIP, SOIC, SSOP	28-Pin QFN,	ADC	Reference	Comparator	Operation Amplifiers	8-bit/ 5-bit DAC	Timers	PSMC	CCP	EUSART	MSSP	Interrupt	Pull-up	Basic
RA0	2	27	AN0	—	C1IN0- C2IN0- C3IN0- C4IN0-	—	—	—	—	—	—	$\overline{SS}$(1)	IOC	Y	—
RA1	3	28	AN1	—	C1IN1- C2IN1- C3IN1- C4IN1-	OPA1OUT	—	—	—	—	—	—	IOC	Y	—
RA2	4	1	AN2	VREF- DAC1VREF-	C1IN0+ C2IN0+ C3IN0+ C4IN0+		DAC1OUT1	—	—	—	—	—	IOC	Y	—
RA3	5	2	AN3	VREF+ DAC1VREF+ DAC2VREF+ DAC3VREF+ DAC4VREF+	C1IN1+	—	—	—	—	—	—	—	IOC	Y	—
RA4	6	3	—	—	C1OUT	OPA1IN+	DAC4OUT1	T0CKI	—	—	—	—	IOC	Y	—
RA5	7	4	AN4	—	C2OUT	OPA1IN-	DAC2OUT1	—	—	—	—	$\overline{SS}$	IOC	Y	—
RA6	10	7	—	—	C2OUT(1)	—	—	—	—	—	—	—	IOC	Y	VCAP OSC2 CLKOUT
RA7	9	6	—	—	—	—	—	—	PSMC1CLK PSMC2CLK PSMC3CLK PSMC4CLK	—	—	—	IOC	Y	CLKIN OSC1
RB0	21	18	AN12	—	C2IN1+	—	—	—	PSMC1IN PSMC2IN PSMC3IN PSMC4IN	CCP1(1)	—	—	INT IOC	Y	—
RB1	22	19	AN10	—	C1IN3- C2IN3- C3IN3- C4IN3-	OPA2OUT	—	—	—	—	—	—	IOC	Y	—
RB2	23	20	AN8	—	—	OPA2IN-	DAC3OUT1	—	—	—	—	—	IOC	Y	CLKR
RB3	24	21	AN9	—	C1IN2- C2IN2- C3IN2-	OPA2IN+	—	—	—	CCP2(1)	—	—	IOC	Y	—
RB4	25	22	AN11	—	C3IN1+	—	—	—	—	—	—	$\overline{SS}$(1)	IOC	Y	—
RB5	26	23	AN13	—	C4IN2- C3OUT	—	—	T1G	—	CCP3(1)	—	SDO(1)	IOC	Y	—
RB6	27	24	—	—	C4IN1+	—	—	—	—	—	TX(1) CK(1)	SDI(1) SDA(1)	IOC	Y	ICSPCLK
RB7	28	25	—	—	—	—	DAC1OUT2 DAC2OUT2 DAC3OUT2 DAC4OUT2	—	—	—	RX(1) DT(1)	SCK(1) SCL(1)	IOC	Y	ICSPDAT
RC0	11	8	—	—	—	—	—	T1CKI T1OSO	PSMC1A	—	—	—	IOC	Y	—
RC1	12	9	—	—	—	—	—	T1OSI	PSMC1B	CCP2	—	—	IOC	Y	—
RC2	13	10	—	—	—	—	—	—	PSMC1C PSMC3B	CCP1	—	—	IOC	Y	—
RC3	14	11	—	—	—	—	—	—	PSMC1D PSMC4A	—	—	SCK SCL	IOC	Y	—
RC4	15	12	—	—	—	—	—	—	PSMC1E PSMC4B	—	—	SDI SDA	IOC	Y	—
RC5	16	13	—	—	—	—	—	—	PSMC1F PSMC3A	—	—	SDO	IOC	Y	—

Note 1: Alternate pin function selected with the APFCON1 (Register 13-1) and APFCON2 (Register 13-2) registers.

TABLE 3-6: PIC16(L)F1788/9 MEMORY MAP (BANK 10 DETAILS)

	BANK 10
50Ch	Unimplemented Read as '0'
510h	
511h	OPA1CON
512h	—
513h	OPA2CON
514h	—
515h	OPA3CON(1)
516h	—
517h	—
518h	—
519h	
51Ah	CLKRCON
51Bh	Unimplemented Read as '0'
51Fh	

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

Note 1: PIC16(L)F1789 only.

TABLE 3-7: PIC16(L)F1788/9 MEMORY MAP (BANK 11 DETAILS)

	BANK 11
58Ch	Unimplemented Read as '0'
590h	
591h	DAC2CON0
592h	DAC2CON1
593h	DAC3CON0
594h	DAC3CON1
595h	DAC4CON0
596h	DAC4CON1
597h	Unimplemented Read as '0'
59Fh	

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

TABLE 3-8: PIC16(L)F1788/9 MEMORY MAP (BANK 31 DETAILS)

	BANK 31
F8Ch	Unimplemented Read as '0'
FE3h	
FE4h	STATUS_SHAD
FE5h	WREG_SHAD
FE6h	BSR_SHAD
FE7h	PCLATH_SHAD
FE8h	FSR0L_SHAD
FE9h	FSR0H_SHAD
FEAh	FSR1L_SHAD
FEBh	FSR1H_SHAD
FECh	—
FEDh	STKPTR
FEEh	TOSL
FEFh	TOSH

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

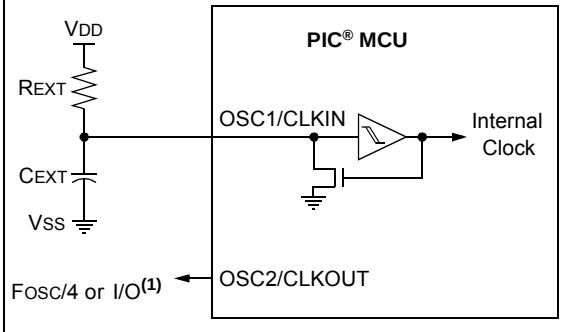
6.2.1.6 External RC Mode

The external Resistor-Capacitor (RC) modes support the use of an external RC circuit. This allows the designer maximum flexibility in frequency choice while keeping costs to a minimum when clock accuracy is not required.

The RC circuit connects to OSC1. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. The function of the OSC2/CLKOUT pin is determined by the $\overline{\text{CLKOUTEN}}$ bit in Configuration Words.

Figure 6-6 shows the external RC mode connections.

FIGURE 6-6: EXTERNAL RC MODES



Recommended values: 10 kΩ ≤ REXT ≤ 100 kΩ, <3V
3 kΩ ≤ REXT ≤ 100 kΩ, 3-5V
CEXT > 20 pF, 2-5V

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

The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. Other factors affecting the oscillator frequency are:

- threshold voltage variation
- component tolerances
- packaging variations in capacitance

The user also needs to take into account variation due to tolerance of external RC components used.

6.2.2 INTERNAL CLOCK SOURCES

The device may be configured to use the internal oscillator block as the system clock by performing one of the following actions:

- Program the FOSC<2:0> bits in Configuration Words to select the INTOSC clock source, which 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 the internal oscillator during run-time. See **Section 6.3 "Clock Switching"**for more information.

In **INTOSC** mode, OSC1/CLKIN is available for general purpose I/O. OSC2/CLKOUT is available for general purpose I/O or CLKOUT.

The function of the OSC2/CLKOUT pin is determined by the $\overline{\text{CLKOUTEN}}$ bit in Configuration Words.

The internal oscillator block has two independent oscillators and a dedicated Phase-Lock Loop, HFPLL that can produce one of three internal system clock sources.

1. The **HFINTOSC** (High-Frequency Internal Oscillator) is factory calibrated and operates at 16 MHz. The HFINTOSC source is generated from the 500 kHz MFINTOSC source and the dedicated Phase-Lock Loop, HFPLL. The frequency of the HFINTOSC can be user-adjusted via software using the OSCTUNE register (Register 6-3).
2. The **MFINTOSC** (Medium-Frequency Internal Oscillator) is factory calibrated and operates at 500 kHz. The frequency of the MFINTOSC can be user-adjusted via software using the OSCTUNE register (Register 6-3).
3. The **LFINTOSC** (Low-Frequency Internal Oscillator) is uncalibrated and operates at 31 kHz.

8.0 INTERRUPTS

The interrupt feature allows certain events to preempt normal program flow. Firmware is used to determine the source of the interrupt and act accordingly. Some interrupts can be configured to wake the MCU from Sleep mode.

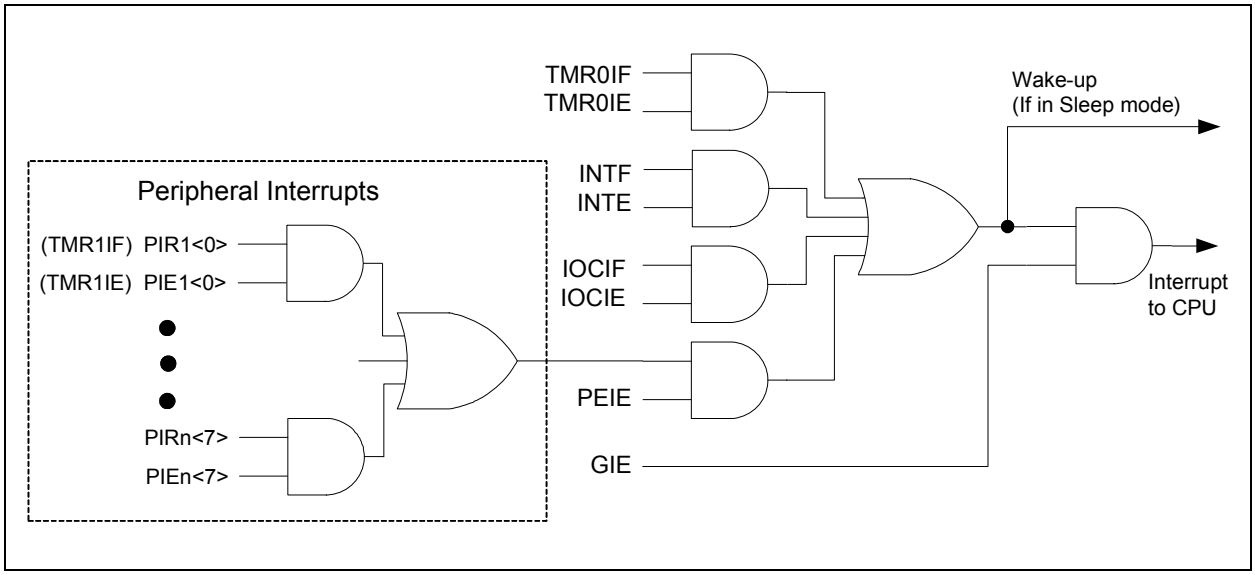
This chapter contains the following information for Interrupts:

- Operation
- Interrupt Latency
- Interrupts During Sleep
- INT Pin
- Automatic Context Saving

Many peripherals produce interrupts. Refer to the corresponding chapters for details.

A block diagram of the interrupt logic is shown in Figure 8-1.

FIGURE 8-1: INTERRUPT LOGIC



12.3.2 ERASING FLASH PROGRAM MEMORY

While executing code, program memory can only be erased by rows. To erase a row:

1. Load the EEADRH:EEADRL register pair with the address of new row to be erased.
2. Clear the CFGS bit of the EECON1 register.
3. Set the EEPGD, FREE, and WREN bits of the EECON1 register.
4. Write 55h, then AAh, to EECON2 (Flash programming unlock sequence).
5. Set control bit WR of the EECON1 register to begin the erase operation.
6. Poll the FREE bit in the EECON1 register to determine when the row erase has completed.

See Example 12-4.

After the "BSF EECON1,WR" instruction, the processor requires two cycles to set up the erase operation. The user must place two NOP instructions after the WR bit is set. The processor will halt internal operations for the typical 2 ms erase time. This is not Sleep mode as the clocks and peripherals will continue to run. After the erase cycle, the processor will resume operation with the third instruction after the EECON1 write instruction.

12.3.3 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the starting address of the word(s) to be programmed.
2. Load the write latches with data.
3. Initiate a programming operation.
4. Repeat steps 1 through 3 until all data is written.

Before writing to program memory, the word(s) to be written must be erased or previously unwritten. Program memory can only be erased one row at a time. No automatic erase occurs upon the initiation of the write.

Program memory can be written one or more words at a time. The maximum number of words written at one time is equal to the number of write latches. See Figure 12-2 (block writes to program memory with 32 write latches) for more details. The write latches are aligned to the address boundary defined by EEADRL as shown in Table 12-1. Write operations do not cross these boundaries. At the completion of a program memory write operation, the write latches are reset to contain 0x3FFF.

The following steps should be completed to load the write latches and program a block of program memory. These steps are divided into two parts. First, all write latches are loaded with data except for the last program memory location. Then, the last write latch is loaded and the programming sequence is initiated. A special unlock sequence is required to load a write latch with data or initiate a Flash programming operation. This unlock sequence should not be interrupted.

1. Set the EEPGD and WREN bits of the EECON1 register.
2. Clear the CFGS bit of the EECON1 register.
3. Set the LWLO bit of the EECON1 register. When the LWLO bit of the EECON1 register is '1', the write sequence will only load the write latches and will not initiate the write to Flash program memory.
4. Load the EEADRH:EEADRL register pair with the address of the location to be written.
5. Load the EEDATH:EEDATL register pair with the program memory data to be written.
6. Write 55h, then AAh, to EECON2, then set the WR bit of the EECON1 register (Flash programming unlock sequence). The write latch is now loaded.
7. Increment the EEADRH:EEADRL register pair to point to the next location.
8. Repeat steps 5 through 7 until all but the last write latch has been loaded.
9. Clear the LWLO bit of the EECON1 register. When the LWLO bit of the EECON1 register is '0', the write sequence will initiate the write to Flash program memory.
10. Load the EEDATH:EEDATL register pair with the program memory data to be written.
11. Write 55h, then AAh, to EECON2, then set the WR bit of the EECON1 register (Flash programming unlock sequence). The entire latch block is now written to Flash program memory.

It is not necessary to load the entire write latch block with user program data. However, the entire write latch block will be written to program memory.

An example of the complete write sequence for eight words is shown in Example 12-5. The initial address is loaded into the EEADRH:EEADRL register pair; the eight words of data are loaded using indirect addressing.

After the "BSF EECON1,WR" instruction, the processor requires two cycles to set up the write operation. The user must place two NOP instructions after the WR bit is set. The processor will halt internal operations for the typical 2 ms, only during the cycle in which the write takes place (i.e., the last word of the block write). This is not Sleep mode as the clocks and peripherals will continue to run. The processor does not stall when LWLO = 1, loading the write latches. After the write cycle, the processor will resume operation with the third instruction after the EECON1 WRITE instruction.

NOTES:

19.6 Register Definitions: DAC Control

REGISTER 19-1: DAC1CON0: VOLTAGE REFERENCE CONTROL REGISTER 0

R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0
DAC1EN	—	DAC1OE1	DAC1OE2	DAC1PSS<1:0>		—	DAC1NSS
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 **DAC1EN:** DAC1 Enable bit
1 = DAC1 is enabled
0 = DAC1 is disabled

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **DAC1OE1:** DAC1 Voltage Output 1 Enable bit
1 = DAC1 voltage level is also an output on the DAC1OUT1 pin
0 = DAC1 voltage level is disconnected from the DAC1OUT1 pin

bit 4 **DAC1OE2:** DAC1 Voltage Output 2 Enable bit
1 = DAC1 voltage level is also an output on the DAC1OUT2 pin
0 = DAC1 voltage level is disconnected from the DAC1OUT2 pin

bit 3-2 **DAC1PSS<1:0>:** DAC1 Positive Source Select bits
11 = Reserved, do not use
10 = FVR Buffer2 output
01 = V_{REF+} pin
00 = V_{DD}

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **DAC1NSS:** DAC1 Negative Source Select bits
1 = V_{REF-} pin
0 = V_{SS}

REGISTER 19-2: DAC1CON1: VOLTAGE REFERENCE CONTROL REGISTER 1

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
DAC1R<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 **DAC1R<7:0>:** DAC1 Voltage Output Select bits

TABLE 19-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
FVRCON	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR<1:0>		ADFVR<1:0>		**167**
DAC1CON0	DAC1EN	—	DAC1OE1	DAC1OE2	DAC1PSS<1:0>		—	DAC1NSS	**192**
DAC1CON1	DAC1R<7:0>								**192**

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used with the DAC module.

25.3 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully on and fully off states. The PWM signal resembles a square wave where the high portion of the signal is considered the on state and the low portion of the signal is considered the off state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 25-3 shows a typical waveform of the PWM signal.

25.3.1 STANDARD PWM OPERATION

The standard PWM function described in this section is available and identical for all CCP modules.

The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- PR2 registers
- T2CON registers
- CCPRxL registers
- CCPxCON registers

Figure 25-4 shows a simplified block diagram of PWM operation.

Note 1:	The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.
2:	Clearing the CCPxCON register will relinquish control of the CCPx pin.

FIGURE 25-3: CCP PWM OUTPUT SIGNAL

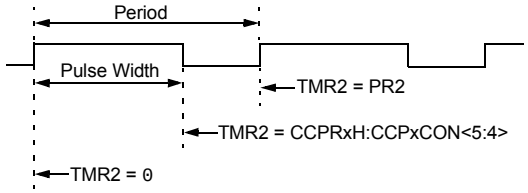
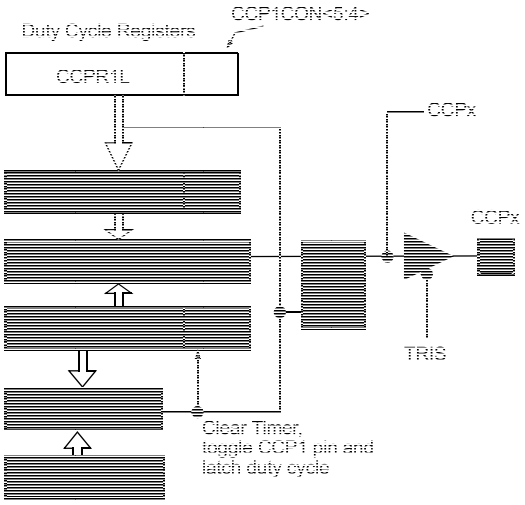


FIGURE 25-4: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: The 8-bit timer TMR2 register is concatenated with the 2-bit internal system clock (FOSC), or two bits of the prescaler, to create the 10-bit time base.

2: In PWM mode, CCPR1H is a read-only register.

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: $T_{OSC} = 1/F_{OSC}$

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 24.1 "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

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

EQUATION 25-3: DUTY CYCLE RATIO

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

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.

The 8-bit timer TMR2 register is concatenated with either the 2-bit internal system clock (F_{OSC}), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRxH and 2-bit latch, then the CCPx pin is cleared (see Figure 25-4).

26.3.9 ECCP COMPATIBLE FULL-BRIDGE PWM

This mode of operation is designed to match the Full-Bridge mode from the ECCP module. It is called ECCP compatible as the term “full-bridge” alone has different connotations in regards to the output waveforms.

Full-Bridge Compatible mode uses the same waveform events as the single PWM mode to generate the output waveforms.

There are both Forward and Reverse modes available for this operation, again to match the ECCP implementation. Direction is selected with the mode control bits.

26.3.9.1 Mode Features

- Dead-band control available on direction switch
  - Changing from forward to reverse uses the falling edge dead-band counters.
  - Changing from reverse to forward uses the rising edge dead-band counters.
- No steering control available
- PWM is output on the following four pins only:
  - PSMCxA
  - PSMCxB
  - PSMCxC
  - PSMCxD

26.3.9.2 Waveform Generation - Forward

In this mode of operation, three of the four pins are static. PSMCxA is the only output that changes based on rising edge and falling edge events.

Static Signal Assignment

- Outputs set to active state
  - PSMCxD
- Outputs set to inactive state
  - PSMCxB
  - PSMCxC

Rising Edge Event

- PSMCxA is set active

Falling Edge Event

- PSMCxA is set inactive

26.3.9.3 Waveform Generation – Reverse

In this mode of operation, three of the four pins are static. Only PSMCxB toggles based on rising edge and falling edge events.

Static Signal Assignment

- Outputs set to active state
  - PSMCxC
- Outputs set to inactive state
  - PSMCxA
  - PSMCxD

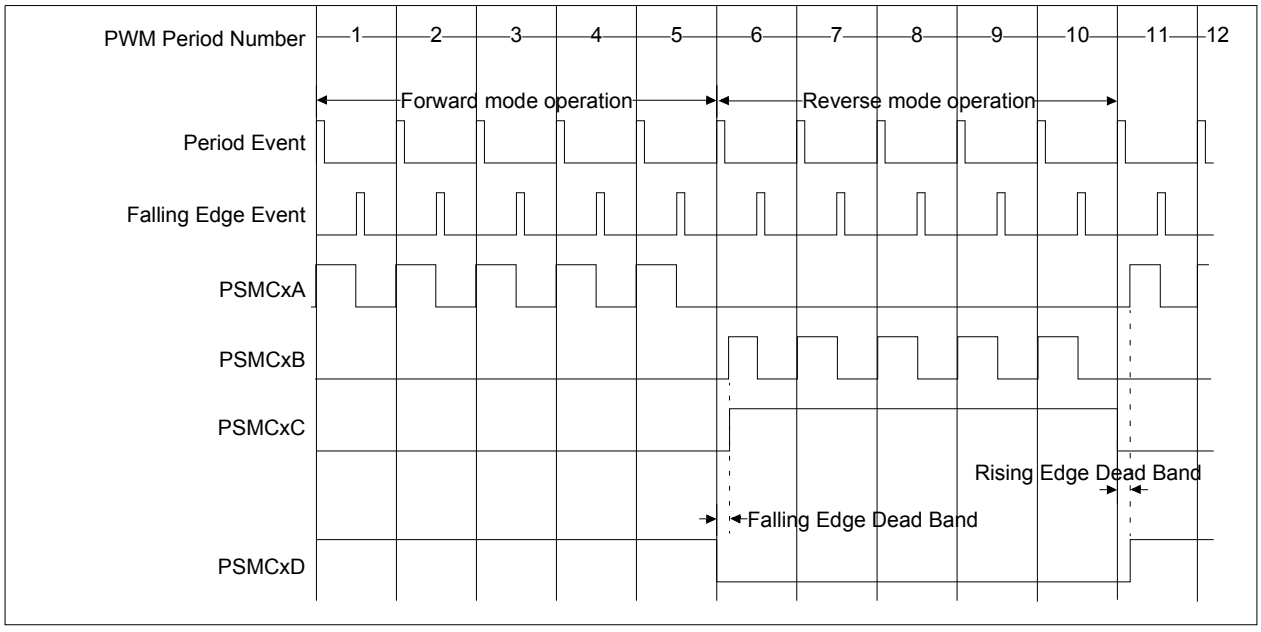
Rising Edge Event

- PSMCxB is set active

Falling Edge Event

- PSMCxB is set inactive

FIGURE 26-12: ECCP COMPATIBLE FULL-BRIDGE PWM WAVEFORM – PSMCXSTR0 = 0FH



26.5 Output Steering

Output steering allows for PWM signals generated by the PSMC module to be placed on different pins under software control. Synchronized steering will hold steering changes until the first period event after the PSMCxLD bit is set. Unsynchronized steering changes will take place immediately.

Output steering is available in the following modes:

- 3-phase PWM
- Single PWM
- Complementary PWM

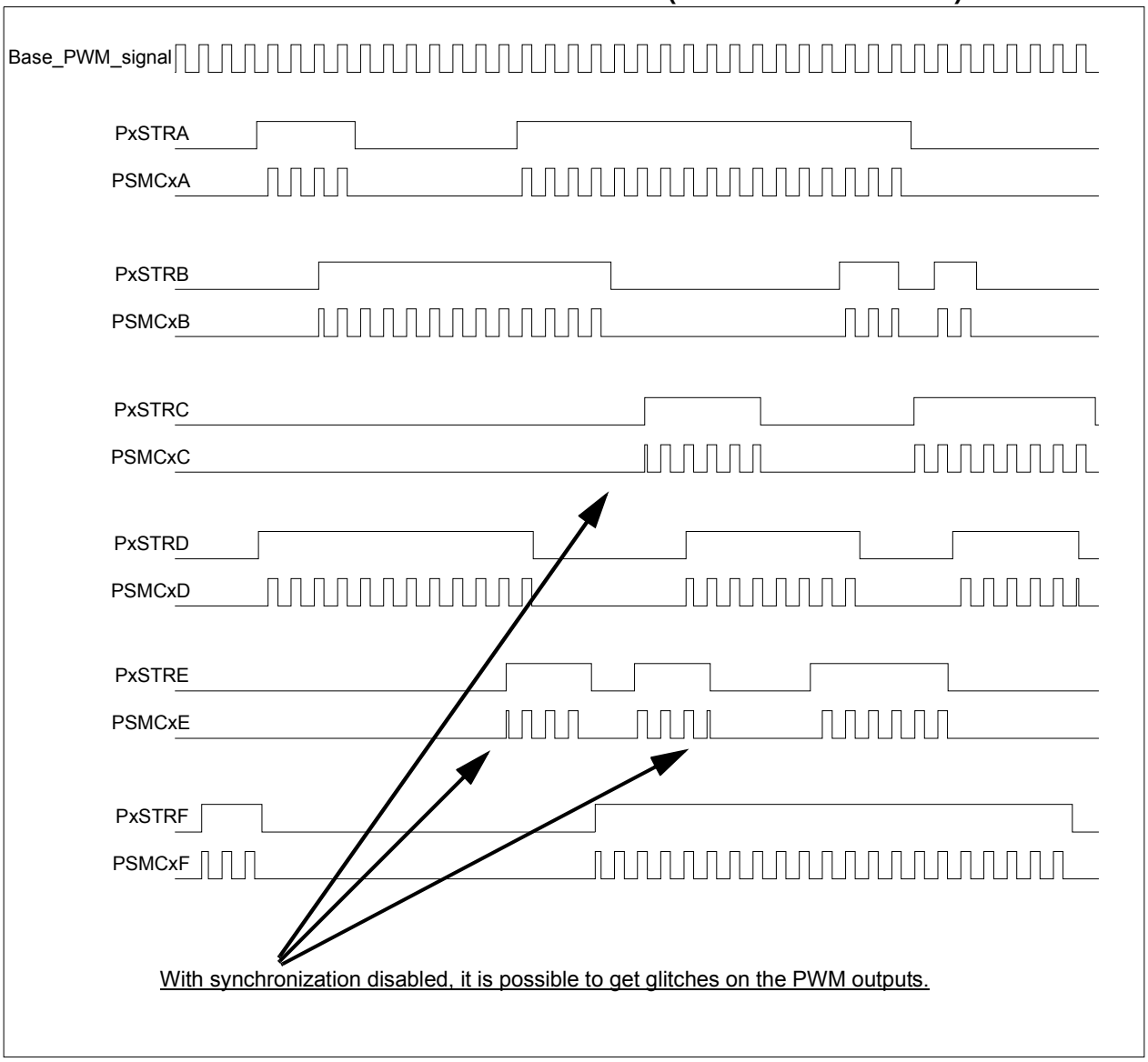
26.5.1 3-PHASE STEERING

3-phase steering is available in the 3-Phase Modulation mode only. For more details on 3-phase steering refer to **Section 26.3.12 "3-Phase PWM"**.

26.5.2 SINGLE PWM STEERING

In Single PWM Steering mode, the single PWM signal can be routed to any combination of the PSMC output pins. Examples of unsynchronized single PWM steering are shown in Figure 26-16.

FIGURE 26-16: SINGLE PWM STEERING WAVEFORM (NO SYNCHRONIZATION)



27.4 I²C MODE OPERATION

All MSSP I²C communication is byte oriented and shifted out MSb first. Six SFR registers and two interrupt flags interface the module with the PIC® microcontroller and user software. Two pins, SDA and SCL, are exercised by the module to communicate with other external I²C devices.

27.4.1 BYTE FORMAT

All communication in I²C is done in 9-bit segments. A byte is sent from a master to a slave or vice-versa, followed by an Acknowledge bit sent back. After the 8th falling edge of the SCL line, the device outputting data on the SDA changes that pin to an input and reads in an acknowledge value on the next clock pulse.

The clock signal, SCL, is provided by the master. Data is valid to change while the SCL signal is low, and sampled on the rising edge of the clock. Changes on the SDA line while the SCL line is high define special conditions on the bus, explained below.

27.4.2 DEFINITION OF I²C TERMINOLOGY

There is language and terminology in the description of I²C communication that have definitions specific to I²C. That word usage is defined below and may be used in the rest of this document without explanation. This table was adapted from the Philips I²C specification.

27.4.3 SDA AND SCL PINS

Selection of any I²C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain. These pins should be set by the user to inputs by setting the appropriate TRIS bits.

Note: Data is tied to output zero when an I²C mode is enabled.

27.4.4 SDA HOLD TIME

The hold time of the SDA pin is selected by the SDAHT bit of the SSPCON3 register. Hold time is the time SDA is held valid after the falling edge of SCL. Setting the SDAHT bit selects a longer 300 ns minimum hold time and may help on buses with large capacitance.

TABLE 27-2: I²C BUS TERMS

TERM	Description
Transmitter	The device which shifts data out onto the bus.
Receiver	The device which shifts data in from the bus.
Master	The device that initiates a transfer, generates clock signals and terminates a transfer.
Slave	The device addressed by the master.
Multi-master	A bus with more than one device that can initiate data transfers.
Arbitration	Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted.
Synchronization	Procedure to synchronize the clocks of two or more devices on the bus.
Idle	No master is controlling the bus, and both SDA and SCL lines are high.
Active	Any time one or more master devices are controlling the bus.
Addressed Slave	Slave device that has received a matching address and is actively being clocked by a master.
Matching Address	Address byte that is clocked into a slave that matches the value stored in SSPADD.
Write Request	Slave receives a matching address with R/$\overline{W}$ bit clear, and is ready to clock in data.
Read Request	Master sends an address byte with the R/$\overline{W}$ bit set, indicating that it wishes to clock data out of the Slave. This data is the next and all following bytes until a Restart or Stop.
Clock Stretching	When a device on the bus hold SCL low to stall communication.
Bus Collision	Any time the SDA line is sampled low by the module while it is out-putting and expected high state.

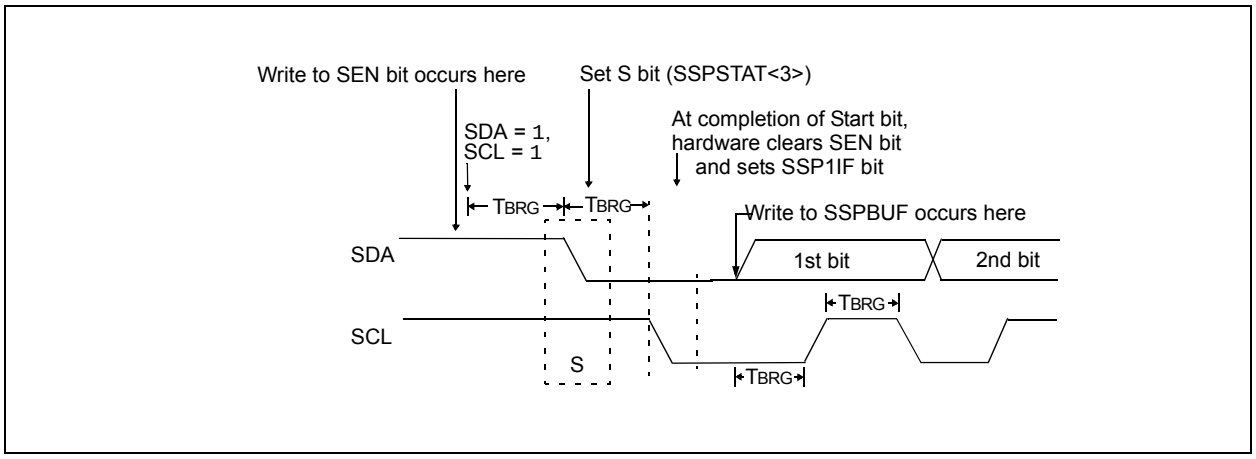
27.6.4 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN bit of the SSPCON2 register. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit of the SSPSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

Note 1: If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCL1IF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.

2: The Philips I²C specification states that a bus collision cannot occur on a Start.

FIGURE 27-26: FIRST START BIT TIMING



27.6.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 27-37). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 27-38).

FIGURE 27-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)

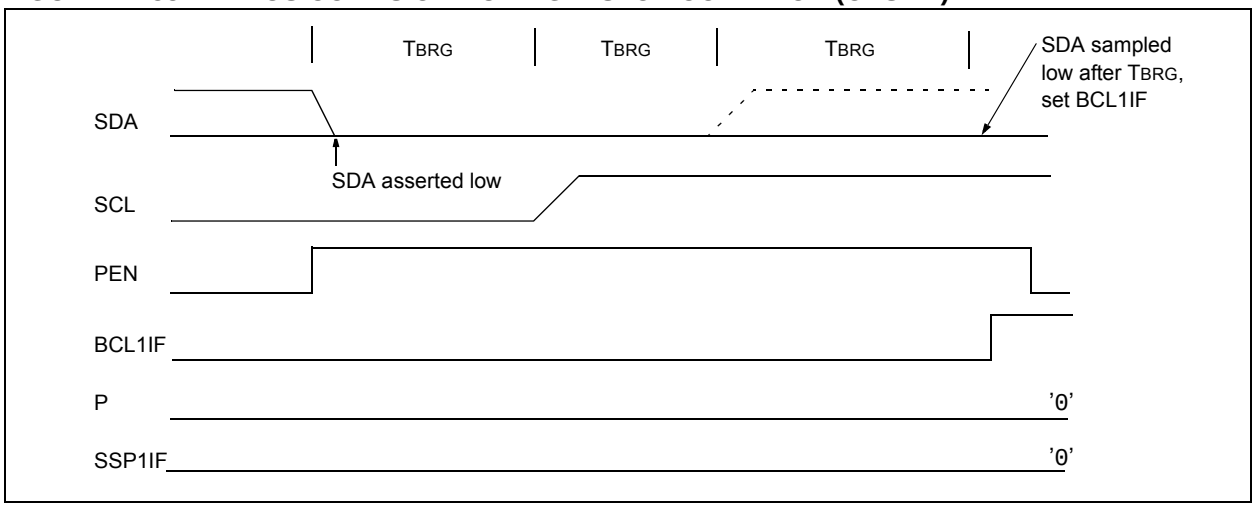
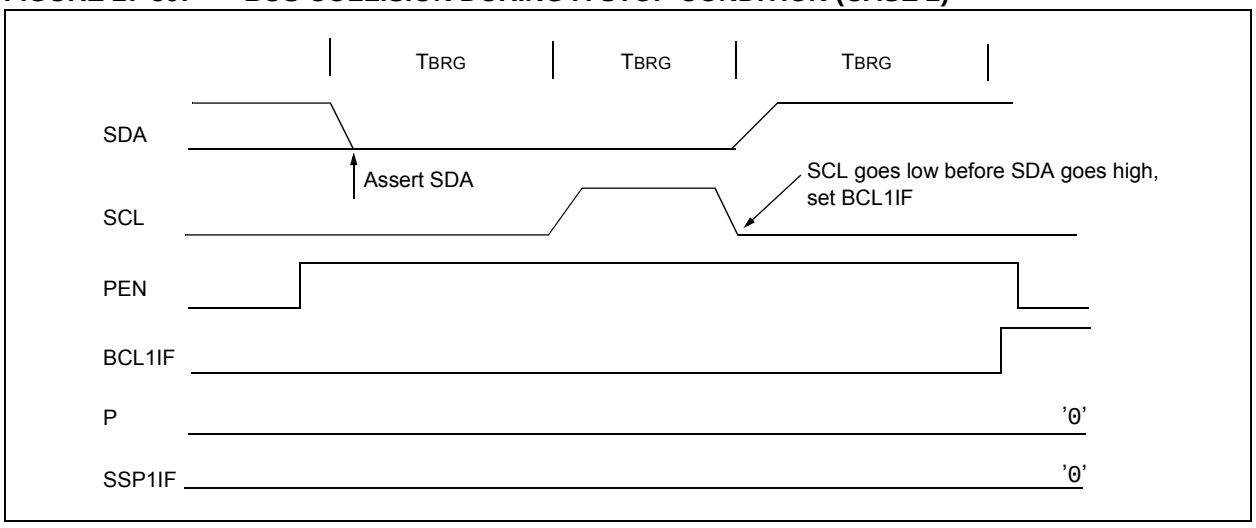


FIGURE 27-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)



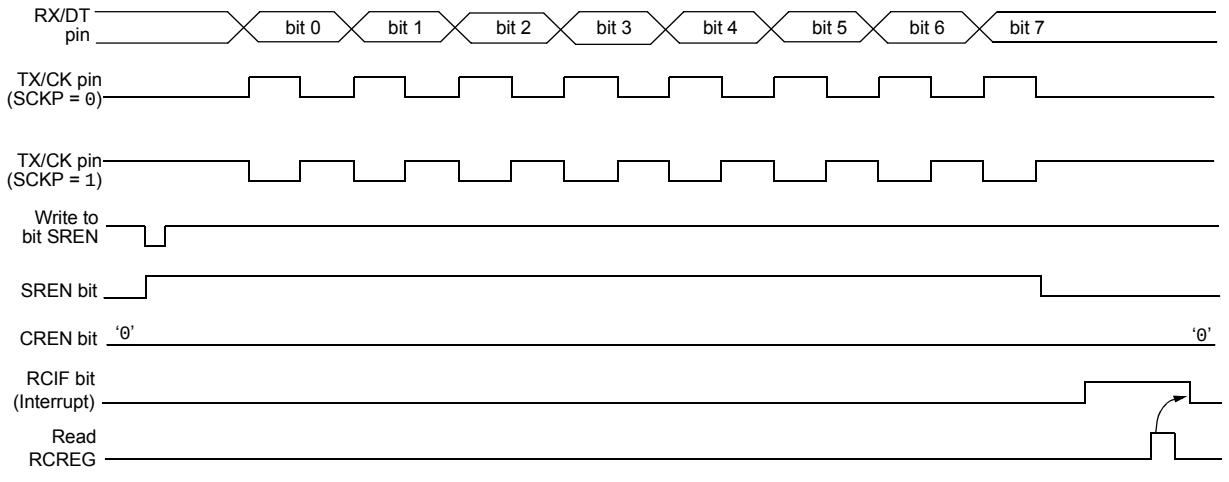
REGISTER 27-1: SSPSTAT: SSP STATUS REGISTER (CONTINUED)

bit 0 **BF:** Buffer Full Status bit

Receive (SPI and I^2C modes):
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

Transmit (I^2C mode only):
1 = Data transmit in progress (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{ACK}$ and Stop bits), SSPBUF is empty

FIGURE 28-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.

TABLE 28-8: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
APFCON1	C2OUTSEL	CC1PSEL	SDOSEL	SCKSEL	SDISEL	TXSEL	RXSEL	CCP2SEL	132
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	356
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	97
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	98
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	102
RCREG	EUSART Receive Data Register								349*
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	355
SPBRGL	BRG<7:0>								357
SPBRGH	BRG<15:8>								357
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	147
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	354

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous master reception.

* Page provides register information.

MOVWF **Move W to f**

Syntax: [*label*] MOVWF f

Operands: $0 \leq f \leq 127$

Operation: (W) → (f)

Status Affected: None

Description: Move data from W register to register 'f'.

Words: 1

Cycles: 1

Example:

```
MOVWF   OPTION_REG
```

Before Instruction
OPTION_REG = 0xFF
W = 0x4F
After Instruction
OPTION_REG = 0x4F
W = 0x4F

MOVWI **Move W to INDFn**

Syntax:
[*label*] MOVWI ++FSRn
[*label*] MOVWI --FSRn
[*label*] MOVWI FSRn++
[*label*] MOVWI FSRn--
[*label*] MOVWI k[FSRn]

Operands:
$n \in [0,1]$
$mm \in [00,01, 10, 11]$
$-32 \leq k \leq 31$

Operation: W → INDFn
Effective address is determined by
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be either:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)

Unchanged

Status Affected: None

Mode	Syntax	mm
Preincrement	++FSRn	00
Predecrement	--FSRn	01
Postincrement	FSRn++	10
Postdecrement	FSRn--	11

Description: This instruction is used to move data between W and one of the indirect registers (INDFn). Before/ after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.

Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.

FSRn is limited to the range 0000h - FFFFh. Incrementing/ decrementing it beyond these bounds will cause it to wrap-around.

The increment/decrement operation on FSRn WILL NOT affect any Status bits.

TABLE 31-4: I/O PORTS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
	VIL	**Input Low Voltage**					
		I/O PORT:					
D034		with TTL buffer	—	—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D034A			—	—	0.15 VDD	V	1.8V ≤ VDD ≤ 4.5V
D035		with Schmitt Trigger buffer	—	—	0.2 VDD	V	2.0V ≤ VDD ≤ 5.5V
		with I^2C levels	—	—	0.3 VDD	V	
		with SMBus levels	—	—	0.8	V	2.7V ≤ VDD ≤ 5.5V
D036		$\overline{MCLR}$, OSC1 (RC mode)(1)	—	—	0.2 VDD	V	
D036A		OSC1 (HS mode)	—	—	0.3 VDD	V	
	VIH	**Input High Voltage**					
		I/O ports:					
D040		with TTL buffer	2.0	—	—	V	4.5V ≤ VDD ≤ 5.5V
D040A			0.25 VDD + 0.8	—	—	V	1.8V ≤ VDD ≤ 4.5V
D041		with Schmitt Trigger buffer	0.8 VDD	—	—	V	2.0V ≤ VDD ≤ 5.5V
		with I^2C levels	0.7 VDD	—	—	V	
		with SMBus levels	2.1	—	—	V	2.7V ≤ VDD ≤ 5.5V
D042		$\overline{MCLR}$	0.8 VDD	—	—	V	
D043A		OSC1 (HS mode)	0.7 VDD	—	—	V	
D043B		OSC1 (RC mode)	0.9 VDD	—	—	V	**(Note 1)**
	IIL	**Input Leakage Current(2)**					
D060		I/O ports	—	± 5	± 125	nA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance @ 85°C
				± 5	± 1000	nA	125°C
D061		$\overline{MCLR}$(3)	—	± 50	± 200	nA	VSS ≤ VPIN ≤ VDD @ 85°C
	IPUR	**Weak Pull-up Current**					
D070*			25 25	100 140	200 300	μA	VDD = 3.3V, VPIN = VSS VDD = 5.0V, VPIN = VSS
	VOL	**Output Low Voltage(4)**					
D080		I/O ports	—	—	0.6	V	IOL = 8mA, VDD = 5V IOL = 6mA, VDD = 3.3V IOL = 1.8mA, VDD = 1.8V
	VOH	**Output High Voltage(4)**					
D090		I/O ports	VDD - 0.7	—	—	V	IOH = 3.5mA, VDD = 5V IOH = 3mA, VDD = 3.3V IOH = 1mA, VDD = 1.8V

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

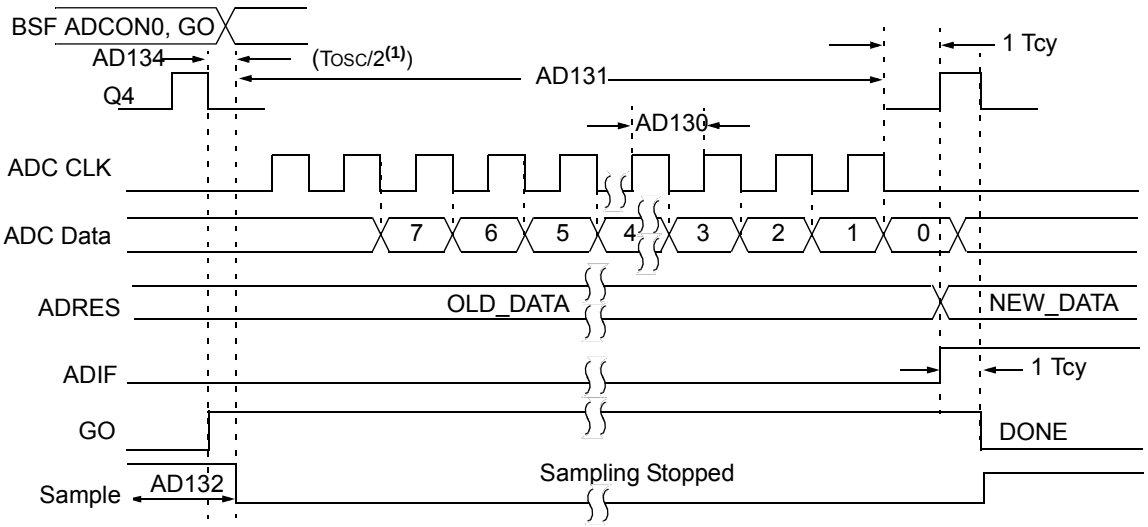
Note 1: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended to use an external clock in RC mode.

2: Negative current is defined as current sourced by the pin.

3: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

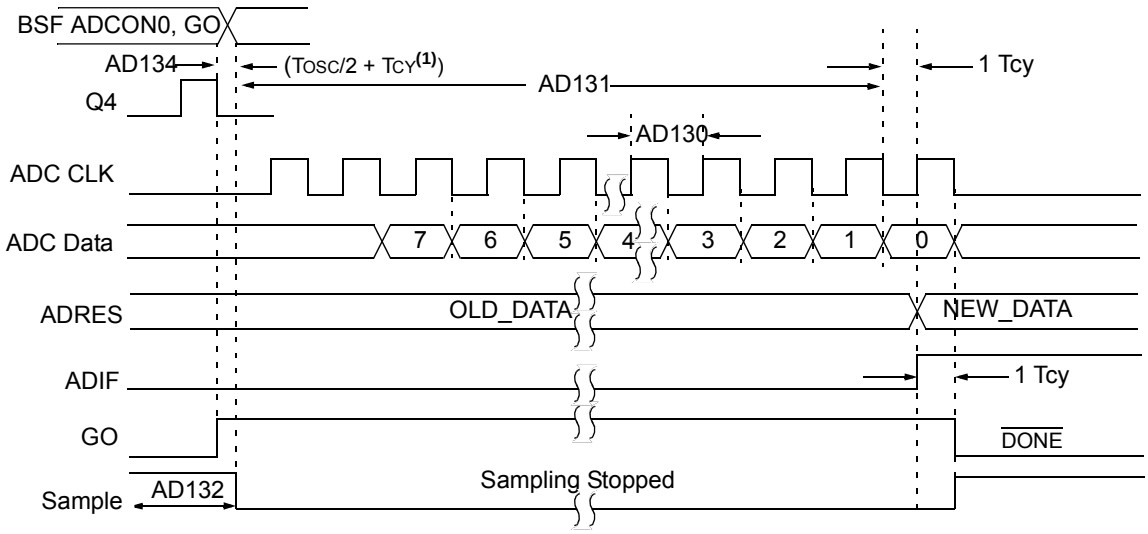
4: Including OSC2 in CLKOUT mode.

FIGURE 31-12: ADC CONVERSION TIMING (NORMAL MODE)



Note 1: If the ADC clock source is selected as RC, a time of TCY is added before the ADC clock starts. This allows the SLEEP instruction to be executed.

FIGURE 31-13: ADC CONVERSION TIMING (SLEEP MODE)



Note 1: If the ADC clock source is selected as RC, a time of TCY is added before the ADC clock starts. This allows the SLEEP instruction to be executed.

APPENDIX A: DATA SHEET REVISION HISTORY

Revision A (02/2013)

Initial release.

Revision B (09/2014)

Change from Preliminary to Final data sheet.

Corrected the following Tables: Family Types Table on page 3, Table 3-3, Table 3-8, Table 20-3, Table 22-2, Table 22-3, Table 23-1, Table 25-3, Table 30-1, Table 30-2, Table 30-3, Table 30-6, Table 30-7, Table 30-13, Table 30-14, Table 30-15, Table 30-16, Table 30-20.

Corrected the following Sections: Section 3.2, Section 9.2, Section 13.3, Section 17.1.6, Section 15.1, Section 15.3, Section 17.2.5, Section 18.2, Section 18.3, Section 19.0, Section 22.6.5, Section 22.9, Section 23.0, Section 23.1, Section 24.2.4, Section 24.2.5, Section 24.2.7, Section 24.8, Section 25.0, Section 26.6.7.4, Section 30.3.

Corrected the following Registers: Register 4-2, Register 8-2, Register 8-5, Register 17-3, Register 18-1, Register 24-3, Register 24-4.

Corrected Equation 17-1.

Corrected Figure 30-9. Removed Figure 24-21.

Revision C (12/2015)

Updated the following Tables: Table 1-1, Table 30-3, Table 31-17, Table 31-18. Updated the following Figures: Figure 18-1, Figure 19-1 and Figure 32-128. Updated Register 18-1 and Register 21-2. Updated the following Sections: Section 26.3.10.2, Section 28.4.2 and Section 31.1; Other minor corrections.