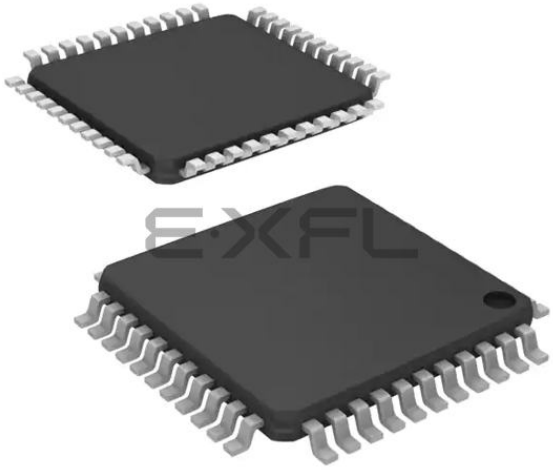


Welcome to E-XFL.COM

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 ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f1789-e-pt

3.0 MEMORY ORGANIZATION

These devices contain the following types of memory:

- Program Memory
  - Configuration Words
  - Device ID
  - User ID
  - Flash Program Memory
- Data Memory
  - Core Registers
  - Special Function Registers
  - General Purpose RAM
  - Common RAM
- Data EEPROM memory(1)

Note 1: The Data EEPROM Memory and the method to access Flash memory through the EECON registers is described in **Section 12.0 "Data EEPROM and Flash Program Memory Control"**.

The following features are associated with access and control of program memory and data memory:

- PCL and PCLATH
- Stack
- Indirect Addressing

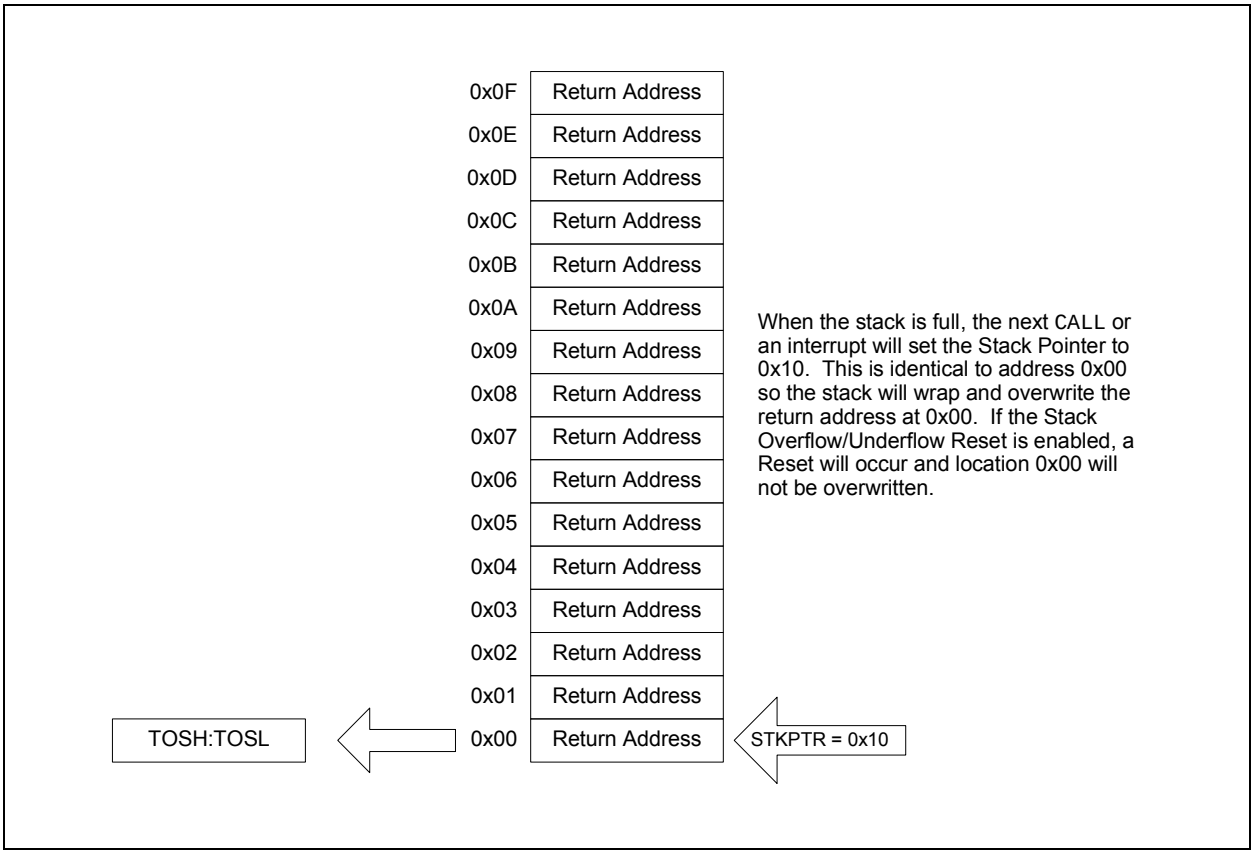
3.1 Program Memory Organization

The enhanced mid-range core has a 15-bit program counter capable of addressing a 32K x 14 program memory space. Table 3-1 shows the memory sizes implemented for the PIC16(L)F1788/9 family. Accessing a location above these boundaries will cause a wrap-around within the implemented memory space. The Reset vector is at 0000h and the interrupt vector is at 0004h (see Figure 3-1).

TABLE 3-1: DEVICE SIZES AND ADDRESSES

Device	Program Memory Space (Words)	Last Program Memory Address
PIC16(L)F1788/9	16,384	3FFFh

FIGURE 3-7: ACCESSING THE STACK EXAMPLE 4



3.5.2 OVERFLOW/UNDERFLOW RESET

If the STVREN bit in Configuration Words is programmed to '1', the device will be reset if the stack is PUSHed beyond the sixteenth level or POPed beyond the first level, setting the appropriate bits (STKOVF or STKUNF, respectively) in the PCON register.

3.6 Indirect Addressing

The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the File Select Registers (FSR). If the FSRn address specifies one of the two INDFn registers, the read will return '0' and the write will not occur (though Status bits may be affected). The FSRn register value is created by the pair FSRnH and FSRnL.

The FSR registers form a 16-bit address that allows an addressing space with 65536 locations. These locations are divided into three memory regions:

- Traditional Data Memory
- Linear Data Memory
- Program Flash Memory

REGISTER 6-3: OSCTUNE: OSCILLATOR TUNING REGISTER

U-0	U-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
—	—	TUN<5:0>					
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 7-6 **Unimplemented:** Read as '0'

bit 5-0 **TUN<5:0>:** Frequency Tuning bits
100000 = Minimum frequency
•
•
•
111111 =
000000 = Oscillator module is running at the factory-calibrated frequency.
000001 =
•
•
•
011110 =
011111 = Maximum frequency

TABLE 6-2: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
OSCCON	SPLLEN	IRCF<3:0>				—	SCS<1:0>		86
OSCSTAT	T1OSCR	PLLR	OSTS	HFIOFR	HFIOFL	MFIOFR	LFIOFR	HFIOFS	87
OSCTUNE	—	—	TUN<5:0>						88
PIE2	OSFIE	C2IE	C1IE	EEIE	BCL1IE	C4IE	C3IE	CCP2IE	99
PIR2	OSFIF	C2IF	C1IF	EEIF	BCL1IF	C4IF	C3IF	CCP2IF	103
T1CON	TMR1CS<1:0>		T1CKPS<1:0>		T1OSCEN	$\overline{\text{T1SYNC}}$	—	TMR1ON	217

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

TABLE 6-3: SUMMARY OF CONFIGURATION WORD WITH CLOCK SOURCES

Name	Bits	Bit -/7	Bit -/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0	Register on Page
CONFIG1	13:8	—	—	FCMEN	IESO	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		$\overline{\text{CPD}}$	58
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		FOSC<2:0>			

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by clock sources.
Note 1: PIC16F1788/9 only.

EXAMPLE 12-3: FLASH PROGRAM MEMORY READ

```
* This code block will read 1 word of program
* memory at the memory address:
    PROG_ADDR_HI : PROG_ADDR_LO
*   data will be returned in the variables;
*   PROG_DATA_HI, PROG_DATA_LO

    BANKSEL  EEADRL           ; Select Bank for EEPROM registers
    MOVLW    PROG_ADDR_LO     ;
    MOVWF    EEADRL           ; Store LSB of address
    MOVLW    PROG_ADDR_HI     ;
    MOVWL    EEADRH           ; Store MSB of address

    BCF      EECON1,CFGS      ; Do not select Configuration Space
    BSF      EECON1,EEPGD     ; Select Program Memory
    BCF      INTCON,GIE       ; Disable interrupts
    BSF      EECON1,RD        ; Initiate read
    NOP                       ; Executed (Figure 12-1)
    NOP                       ; Ignored (Figure 12-1)
    BSF      INTCON,GIE       ; Restore interrupts

    MOVF     EEDATL,W         ; Get LSB of word
    MOVWF    PROG_DATA_LO     ; Store in user location
    MOVF     EEDATH,W         ; Get MSB of word
    MOVWF    PROG_DATA_HI     ; Store in user location
```

21.0 COMPARATOR MODULE

Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. Comparators are very useful mixed signal building blocks because they provide analog functionality independent of program execution. The analog comparator module includes the following features:

- Independent comparator control
- Programmable input selection
- Comparator output is available internally/externally
- Programmable output polarity
- Interrupt-on-change
- Wake-up from Sleep
- Programmable Speed/Power optimization
- PWM shutdown
- Programmable and fixed voltage reference

21.1 Comparator Overview

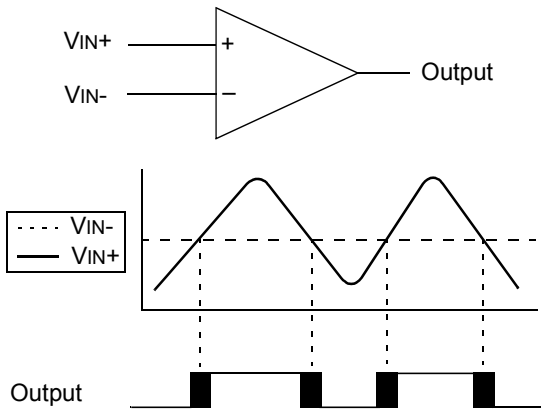
A single comparator is shown in Figure 21-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at VIN+ is less than the analog voltage at VIN-, the output of the comparator is a digital low level. When the analog voltage at VIN+ is greater than the analog voltage at VIN-, the output of the comparator is a digital high level.

The comparators available for this device are located in Table 21-1.

TABLE 21-1: COMPARATOR AVAILABILITY PER DEVICE

Device	C1	C2	C3	C4
PIC16(L)F1788/9	●	●	●	●

FIGURE 21-1: SINGLE COMPARATOR



Note: The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.

21.11 Register Definitions: Comparator Control

REGISTER 21-1: CMxCON0: COMPARATOR Cx CONTROL REGISTER 0

R/W-0/0	R-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-1/1	R/W-0/0	R/W-0/0
CxON	CxOUT	CxOE	CxPOL	CxZLF	CxSP	CxHYS	CxSYNC
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 **CxON:** Comparator Enable bit
1 = Comparator is enabled
0 = Comparator is disabled and consumes no active power

bit 6 **CxOUT:** Comparator Output bit
If CxPOL = 1 (inverted polarity):
1 = CxVP < CxVN
0 = CxVP > CxVN
If CxPOL = 0 (non-inverted polarity):
1 = CxVP > CxVN
0 = CxVP < CxVN

bit 5 **CxOE:** Comparator Output Enable bit
1 = CxOUT is present on the CxOUT pin. Requires that the associated TRIS bit be cleared to actually drive the pin. Not affected by CxON.
0 = CxOUT is internal only

bit 4 **CxPOL:** Comparator Output Polarity Select bit
1 = Comparator output is inverted
0 = Comparator output is not inverted

bit 3 **CxZLF:** Comparator Zero Latency Filter Enable bit
1 = Comparator output is filtered
0 = Comparator output is unfiltered

bit 2 **CxSP:** Comparator Speed/Power Select bit
1 = Comparator operates in normal power, higher speed mode
0 = Comparator operates in low-power, low-speed mode

bit 1 **CxHYS:** Comparator Hysteresis Enable bit
1 = Comparator hysteresis enabled
0 = Comparator hysteresis disabled

bit 0 **CxSYNC:** Comparator Output Synchronous Mode bit
1 = Comparator output to Timer1 and I/O pin is synchronous to changes on Timer1 clock source. Output updated on the falling edge of Timer1 clock source.
0 = Comparator output to Timer1 and I/O pin is asynchronous.

24.1 Timer2 Operation

The clock input to the Timer2 modules is the system instruction clock (FOSC/4).

A 4-bit counter/prescaler on the clock input allows direct input, divide-by-4 and divide-by-16 prescale options. These options are selected by the prescaler control bits, T2CKPS<1:0> of the T2CON register. The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscaler (see **Section 24.2 "Timer2 Interrupt"**).

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

- a write to the TMR2 register
- a write to the T2CON register
- Power-on Reset (POR)
- Brown-out Reset (BOR)
- $\overline{\text{MCLR}}$ Reset
- Watchdog Timer (WDT) Reset
- Stack Overflow Reset
- Stack Underflow Reset
- RESET Instruction

Note:	TMR2 is not cleared when T2CON is written.

24.2 Timer2 Interrupt

Timer2 can also generate an optional device interrupt. The Timer2 output signal (TMR2-to-PR2 match) provides the input for the 4-bit counter/postscaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF of the PIR1 register. The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE, of the PIE1 register.

A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS<3:0>, of the T2CON register.

24.3 Timer2 Output

The unscaled output of TMR2 is available primarily to the CCP modules, where it is used as a time base for operations in PWM mode.

Timer2 can be optionally used as the shift clock source for the MSSP module operating in SPI mode. Additional information is provided in **Section 27.0 "Master Synchronous Serial Port (MSSP) Module"**

24.4 Timer2 Operation During Sleep

The Timer2 timers cannot be operated while the processor is in Sleep mode. The contents of the TMR2 and PR2 registers will remain unchanged while the processor is in Sleep mode.

26.2.1.4 16-bit Duty Cycle Register

The PSMCxDC Duty Cycle register is used to determine a synchronous falling edge event referenced to the 16-bit PSMCxTMR digital counter. A match between the PSMCxTMR and PSMCxDC register values will generate a falling edge event.

The match will generate a duty cycle match interrupt, thereby setting the PxTDCIF bit of the PSMC Time Base Interrupt Control (PSMCxINT) register (Register 26-34).

The 16-bit duty cycle value is accessible to software as two 8-bit registers:

- PSMC Duty Cycle Count Low Byte (PSMCxDCL) register (Register 26-23)
- PSMC Duty Cycle Count High Byte (PSMCxDCH) register (Register 26-24)

The 16-bit duty cycle value is double-buffered before it is presented to the 16-bit time base for comparison. The buffered registers are updated on the first period event Reset after the PSMCxLD bit of the PSMCxCON register is set.

When the period, phase, and duty cycle are all determined from the time base, the effective PWM duty cycle can be expressed as shown in Equation 26-2.

EQUATION 26-2: PWM DUTY CYCLE

$$DUTYCYCLE = \frac{\text{PSMCxDC[15:0]} - \text{PSMCxPH[15:0]}}{(\text{PSMCxPR[15:0]} + 1)}$$

26.2.2 0% DUTY CYCLE OPERATION USING TIME BASE

To configure the PWM for 0% duty cycle set PSMCxDC<15:0> = PSMCxPH<15:0>. This will trigger a falling edge event simultaneous with the rising edge event and prevent the PWM from being asserted.

26.2.3 100% DUTY CYCLE OPERATION USING TIME BASE

To configure the PWM for 100% duty cycle set PSMCxDC<15:0> > PSMCxPR<15:0>.

This will prevent a falling edge event from occurring as the PSMCxDC<15:0> value and the time base value PSMCxTMR<15:0> will never be equal.

26.2.4 TIME BASE INTERRUPT GENERATION

The Time Base section can generate four unique interrupts:

- Time Base Counter Overflow Interrupt
- Time Base Phase Register Match Interrupt
- Time Base Duty Cycle Register Match Interrupt
- Time Base Period Register Match Interrupt

Each interrupt has an interrupt flag bit and an interrupt enable bit. The interrupt flag bit is set anytime a given event occurs, regardless of the status of the enable bit.

Time base interrupt enables and flags are located in the PSMC Time Base Interrupt Control (PSMCxINT) register (Register 26-34).

PSMC time base interrupts also require that the PSMCxTIE bit in the PIE4 register and the PEIE and GIE bits in the INTCON register be set in order to generate an interrupt. The PSMCxTIF interrupt flag in the PIR4 register will only be set by a time base interrupt when one or more of the enable bits in the PSMCxINT register is set.

The interrupt flag bits need to be cleared in software. However, all PMSCx time base interrupt flags, except PSMCxTIF, are cleared when the PSMCxEN bit is cleared.

Interrupt bits that are set by software will generate an interrupt provided that the corresponding interrupt is enabled.

Note:	Interrupt flags in both the PIE4 and PSMCxINT registers must be cleared to clear the interrupt. The PSMCxINT flags must be cleared first.

26.2.5 PSMC TIME BASE CLOCK SOURCES

There are three clock sources available to the module:

- Internal 64 MHz clock
- Fosc system clock
- External clock input pin

The clock source is selected with the PxCSRC<1:0> bits of the PSMCx Clock Control (PSMCxCLK) register (Register 26-7).

When the Internal 64 MHz clock is selected as the source, the HFINTOSC continues to operate and clock the PSMC circuitry in Sleep. However, the system clock to other peripherals and the CPU is suppressed.

Note:	When the 64 MHz clock is selected, the clock continues to operate in Sleep, even when the PSMC is disabled (PSMCxEN = 0). Select a clock other than the 64 MHz clock to minimize power consumption when the PSMC is not enabled.

The Internal 64 MHz clock utilizes the system clock 4x PLL. When the system clock source is external and the PSMC is using the Internal 64 MHz clock, the 4x PLL should not be used for the system clock.

26.2.9 OUTPUT WAVEFORM GENERATION

The PSMC PWM output waveform is generated based upon the different input events. However, there are several other factors that affect the PWM waveshapes:

- Output Control
  - Output Enable
  - Output Polarity
- Waveform Mode Selection
- Dead-band Control
- Steering control

26.2.10 OUTPUT CONTROL

26.2.10.1 Output Pin Enable

Each PSMC PWM output pin has individual output enable control.

When the PSMC output enable control is disabled, the module asserts no control over the pin. In this state, the pin can be used for general purpose I/O or other associate peripheral use.

When the PSMC output enable is enabled, the active PWM waveform is applied to the pin per the port priority selection.

PSMC output enable selections are made with the PSMC Output Enable Control (PSMCxOEN) register (Register 26-8).

26.2.10.2 Output Steering

PWM output will be presented only on pins for which output steering is enabled. The PSMC has up to six PWM outputs. The PWM signal in some modes can be steered to one or more of these outputs.

Steering differs from output enable in the following manner: When the output is enabled but the PWM steering to the corresponding output is not enabled, then general purpose output to the pin is disabled and the pin level will remain constantly in the inactive PWM state. Output steering is controlled with the PSMCS Steering Control 0 (PSMCxSTR0) register (Register 26-32).

Steering operates only in the following modes:

- Single-phase
- Complementary Single-phase
- 3-phase 6-step PWM

26.2.10.3 Polarity Control

Each PSMC output has individual output polarity control. Polarity is set with the PSMC Polarity Control (PSMCxPOL) register (Register 26-9).

26.3 Modes of Operation

All modes of operation use the period, rising edge, and falling edge events to generate the various PWM output waveforms.

The 3-phase 6-step PWM mode makes special use of the software controlled steering to generate the required waveform.

Modes of operation are selected with the PSMC Control (PSMCxCON) register (Register 26-1).

26.3.1 SINGLE-PHASE MODE

The single PWM is the most basic of all the waveshapes generated by the PSMC module. It consists of a single output that uses all three events (rising edge, falling edge and period events) to generate the waveform.

26.3.1.1 Mode Features

- No dead-band control available
- PWM can be steered to any combination of the following PSMC outputs:
  - PSMCxA
  - PSMCxB
  - PSMCxC
  - PSMCxD
  - PSMCxE
  - PSMCxF
- Identical PWM waveform is presented to all pins for which steering is enabled.

26.3.1.2 Waveform Generation

Rising Edge Event

- All outputs with PxSTR enabled are set to the active state

Falling Edge Event

- All outputs with PxSTR enabled are set to the inactive state

Code for setting up the PSMC generate the single-phase waveform shown in Figure 26-4, and given in Example 26-1.

26.3.8 PULSE-SKIPPING PWM WITH COMPLEMENTARY OUTPUTS

The pulse-skipping PWM is used to generate a series of fixed-length pulses that may or not be triggered at each period event. If any of the sources enabled to generate a rising edge event are high when a period event occurs, a pulse will be generated. If the rising edge sources are low at the period event, no pulse will be generated.

The rising edge 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.

26.3.8.1 Mode Features

- Dead-band control is available
- No steering control available
- Primary PWM is output on only PSMCxA.
- Complementary PWM is output on only PSMCxB.

26.3.8.2 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:

- Complementary output is set inactive
- Dead-band rising is activated (if enabled)
- Primary output is set active

Falling Edge Event

- Primary output is set inactive
- Dead-band falling is activated (if enabled)
- Complementary output is set active

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 26-11: PULSE-SKIPPING WITH COMPLEMENTARY OUTPUT PWM WAVEFORM

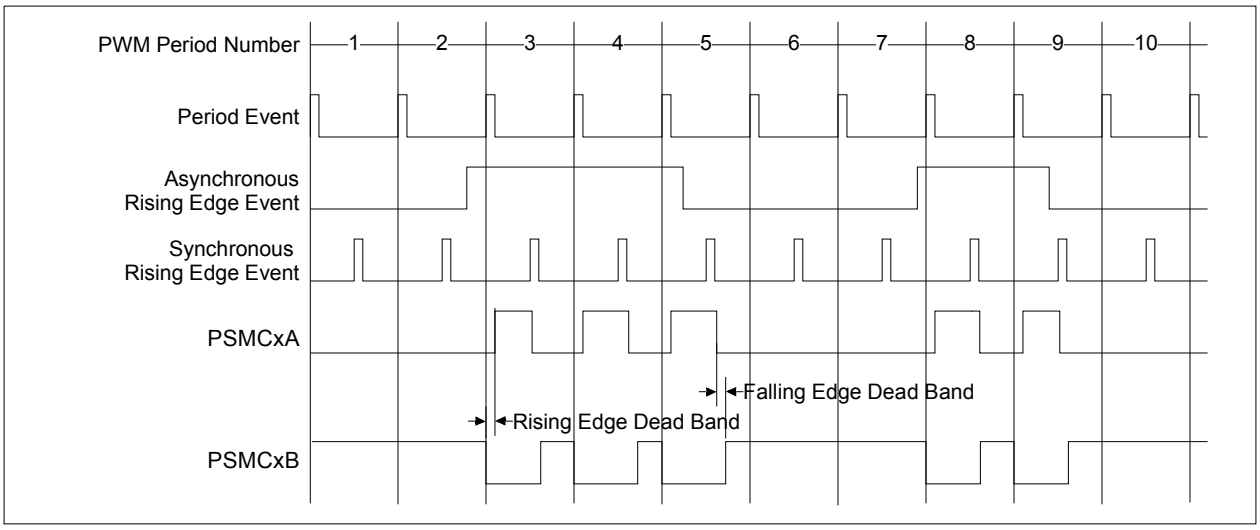
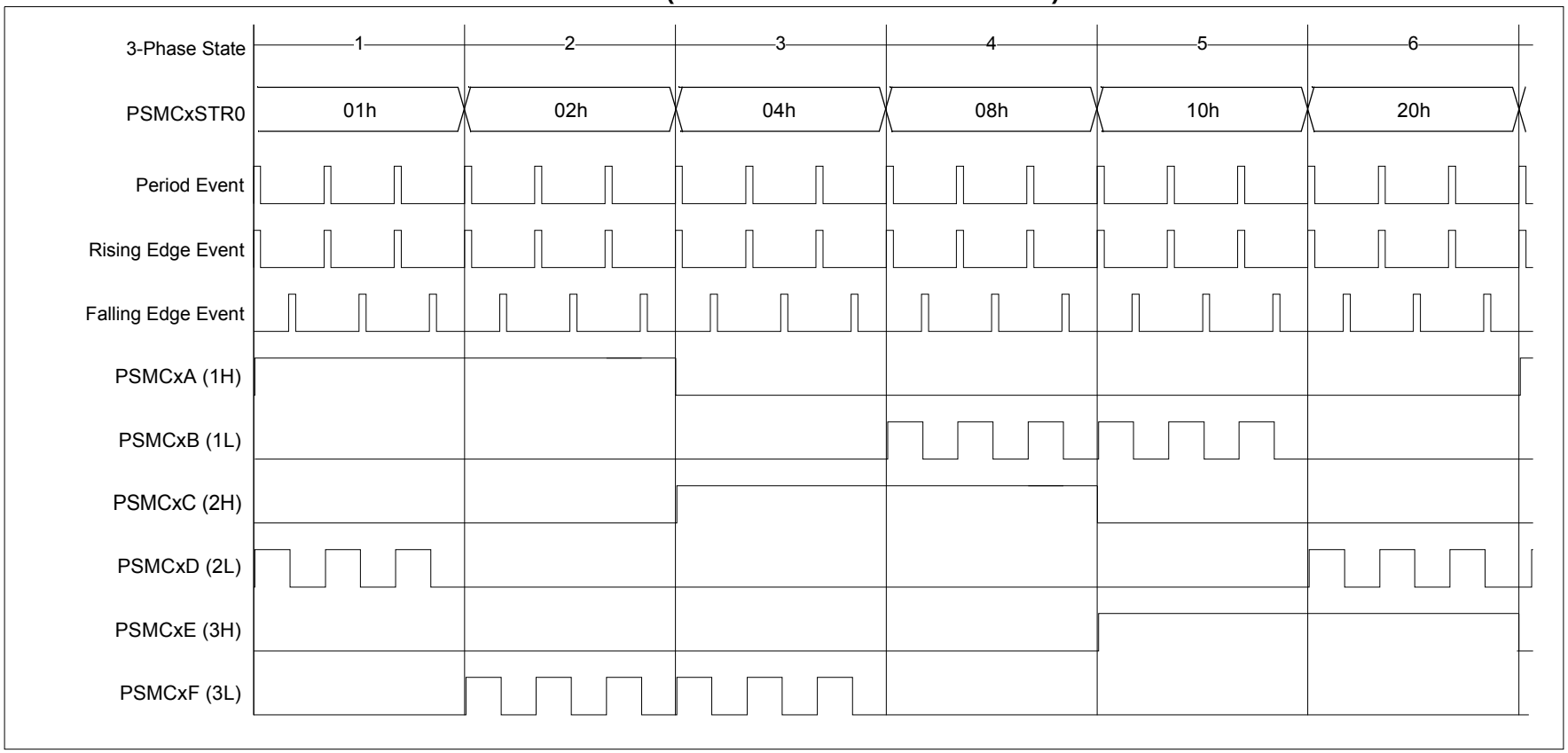


FIGURE 26-15: 3-PHASE PWM STEERING WAVEFORM (PXHSMEN = 0 AND PXLSMEN = 1)



REGISTER 26-11: PSMCxREBS: PSMC RISING EDGE BLANKED SOURCE REGISTER

R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0
PxREBSIN	—	—	PxREBSC4	PxREBSC3	PxREBSC2	PxREBSC1	—
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 **PxREBSIN:** PSMCx Rising Edge Event Blanked from PSMCxIN pin
1 = PSMCxIN pin cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = PSMCxIN pin is not blanked

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **PxREBSC4:** PSMCx Rising Edge Event Blanked from sync_C4OUT
1 = sync_C4OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C4OUT is not blanked

bit 3 **PxREBSC3:** PSMCx Rising Edge Event Blanked from sync_C3OUT
1 = sync_C3OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C3OUT is not blanked

bit 2 **PxREBSC2:** PSMCx Rising Edge Event Blanked from sync_C2OUT
1 = sync_C2OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C2OUT is not blanked

bit 1 **PxREBSC1:** PSMCx Rising Edge Event Blanked from sync_C1OUT
1 = sync_C1OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C1OUT is not blanked

bit 0 **Unimplemented:** Read as '0'

REGISTER 26-12: PSMCxFEBS: PSMC FALLING EDGE BLANKED SOURCE REGISTER

R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0
PxFEBSIN	—	—	PxFEBSC4	PxFEBSC3	PxFEBSC2	PxFEBSC1	—
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 **PxFEBSIN:** PSMCx Falling Edge Event Blanked from PSMCxIN pin
1 = PSMCxIN pin cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = PSMCxIN pin is not blanked

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **PxFEBSC4:** PSMCx Falling Edge Event Blanked from sync_C4OUT
1 = sync_C4OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C4OUT is not blanked

bit 3 **PxFEBSC3:** PSMCx Falling Edge Event Blanked from sync_C3OUT
1 = sync_C3OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C3OUT is not blanked

bit 2 **PxFEBSC2:** PSMCx Falling Edge Event Blanked from sync_C2OUT
1 = sync_C2OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C2OUT is not blanked

bit 1 **PxFEBSC1:** PSMCx Falling Edge Event Blanked from sync_C1OUT
1 = sync_C1OUT cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = sync_C1OUT is not blanked

bit 0 **Unimplemented:** Read as '0'

27.2.4 SPI SLAVE MODE

In Slave mode, the data is transmitted and received as external clock pulses appear on SCK. When the last bit is latched, the SSP1IF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit of the SSPCON1 register.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

27.2.4.1 Daisy-Chain Configuration

The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device.

Figure 27-7 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPCON3 register will enable writes to the SSPBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

27.2.5 SLAVE SELECT SYNCHRONIZATION

The Slave Select can also be used to synchronize communication. The Slave Select line is held high until the master device is ready to communicate. When the Slave Select line is pulled low, the slave knows that a new transmission is starting.

If the slave fails to receive the communication properly, it will be reset at the end of the transmission, when the Slave Select line returns to a high state. The slave is then ready to receive a new transmission when the Slave Select line is pulled low again. If the Slave Select line is not used, there is a risk that the slave will eventually become out of sync with the master. If the slave misses a bit, it will always be one bit off in future transmissions. Use of the Slave Select line allows the slave and master to align themselves at the beginning of each transmission.

The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 0100).

When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven.

When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

Note 1: When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to VDD.

2: When the SPI is used in Slave mode with CKE set; the user must enable $\overline{SS}$ pin control.

3: While operated in SPI Slave mode the SMP bit of the SSPSTAT register must remain clear.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{SS}$ pin to a high level or clearing the SSPEN bit.

27.6.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 27-32).

b) SCL is sampled low before SDA is asserted low (Figure 27-33).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

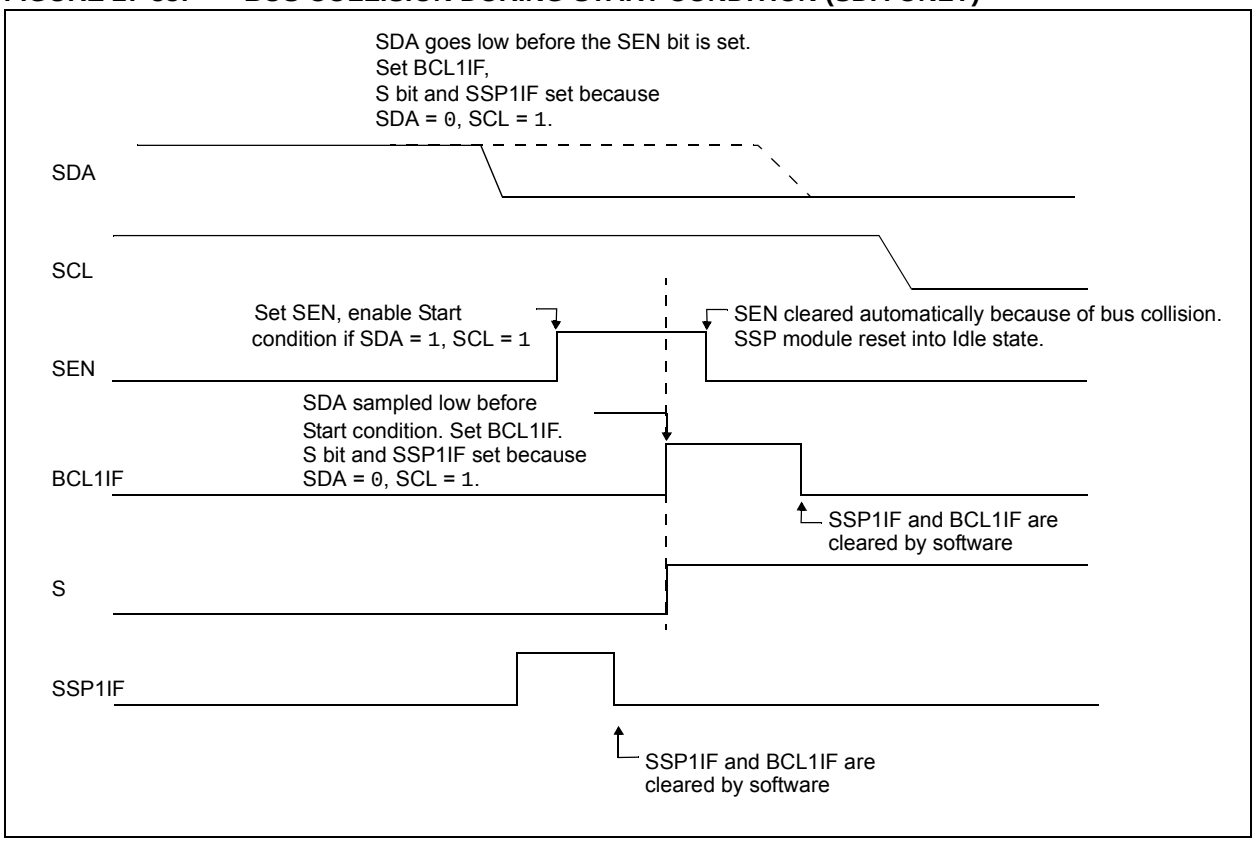
- the Start condition is aborted,
- the BCL1IF flag is set and
- the MSSP module is reset to its Idle state (Figure 27-32).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded and counts down. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 27-34). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to zero; if the SCL pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

Note:	The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 27-33: BUS COLLISION DURING START CONDITION (SDA ONLY)



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)

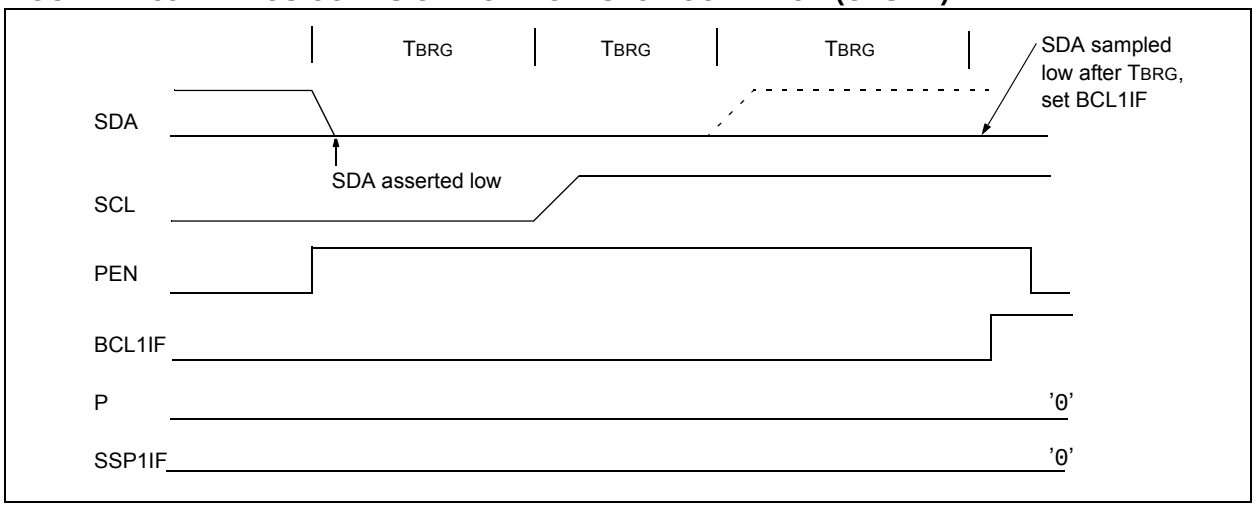
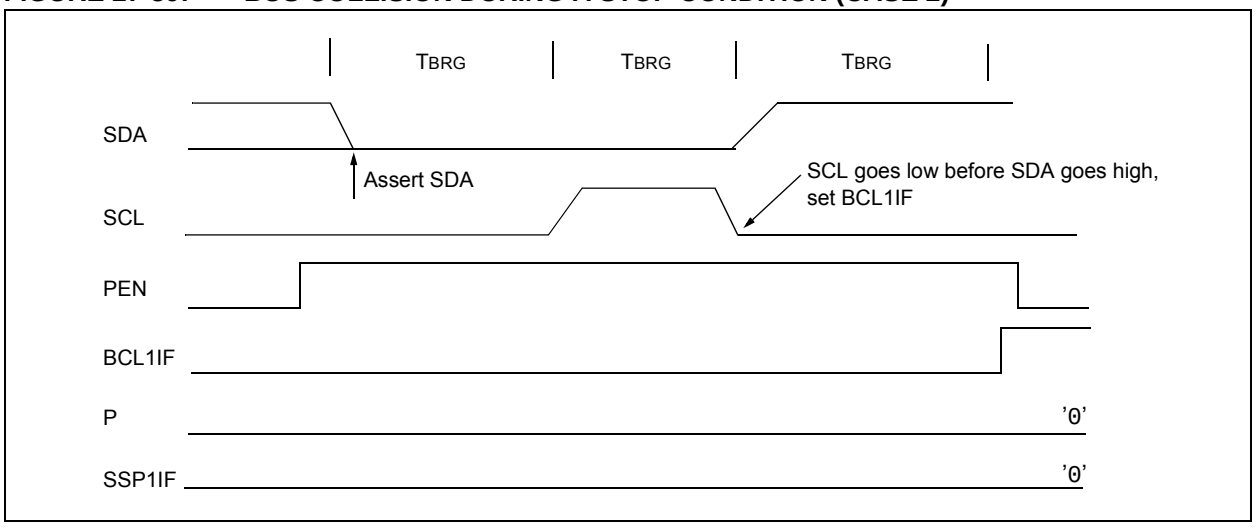


FIGURE 27-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)



28.1.1.5 TSR Status

The TRMT bit of the TXSTA register indicates the status of the TSR register. This is a read-only bit. The TRMT bit is set when the TSR register is empty and is cleared when a character is transferred to the TSR register from the TXREG. The TRMT bit remains clear until all bits have been shifted out of the TSR register. No interrupt logic is tied to this bit, so the user has to poll this bit to determine the TSR status.

Note:	The TSR register is not mapped in data memory, so it is not available to the user.

28.1.1.6 Transmitting 9-Bit Characters

The EUSART supports 9-bit character transmissions. When the TX9 bit of the TXSTA register is set, the EUSART will shift nine bits out for each character transmitted. The TX9D bit of the TXSTA register is the ninth, and Most Significant, data bit. When transmitting 9-bit data, the TX9D data bit must be written before writing the eight Least Significant bits into the TXREG. All nine bits of data will be transferred to the TSR shift register immediately after the TXREG is written.

A special 9-bit Address mode is available for use with multiple receivers. See **Section 28.1.2.7 "Address Detection"** for more information on the address mode.

28.1.1.7 Asynchronous Transmission Set-up:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 28.4 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the eight Least Significant data bits are an address when the receiver is set for address detection.
4. Set SCKP bit if inverted transmit is desired.
5. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.
6. If interrupts are desired, set the TXIE interrupt enable bit of the PIE1 register. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.
7. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
8. Load 8-bit data into the TXREG register. This will start the transmission.

FIGURE 28-3: ASYNCHRONOUS TRANSMISSION

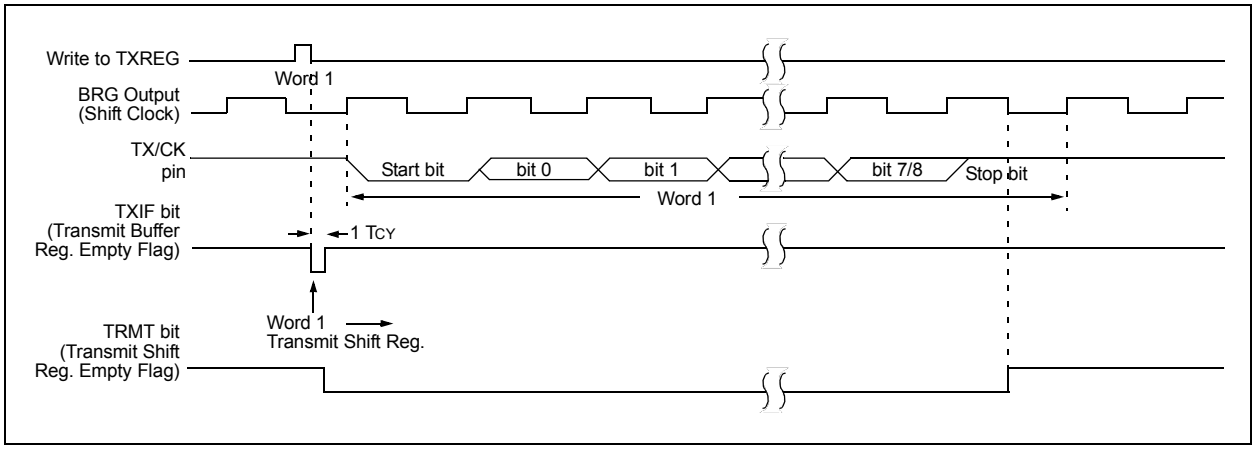


TABLE 28-3: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	FOSC/[64 (n+1)]
0	0	1	8-bit/Asynchronous	FOSC/[16 (n+1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	FOSC/[4 (n+1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH, SPBRGL register pair

TABLE 28-4: SUMMARY OF REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	356
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	355
SPBRGL	BRG<7:0>								357
SPBRGH	BRG<15:8>								357
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	354

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for the Baud Rate Generator.

* Page provides register information.

CALLW **Subroutine Call With W**

Syntax:	[*label*] CALLW
Operands:	None
Operation:	(PC) +1 → TOS, (W) → PC<7:0>, (PCLATH<6:0>) → PC<14:8>
Status Affected:	None
Description:	Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a 2-cycle instruction.

CLRF **Clear f**

Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

CLRW **Clear W**

Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

CLRWDT **Clear Watchdog Timer**

Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

COMF **Complement f**

Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
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'.

DECF **Decrement f**

Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

Note: Unless otherwise noted, VIN = 5V, FOSC = 300 kHz, CIN = 0.1 µF, TA = 25°C.

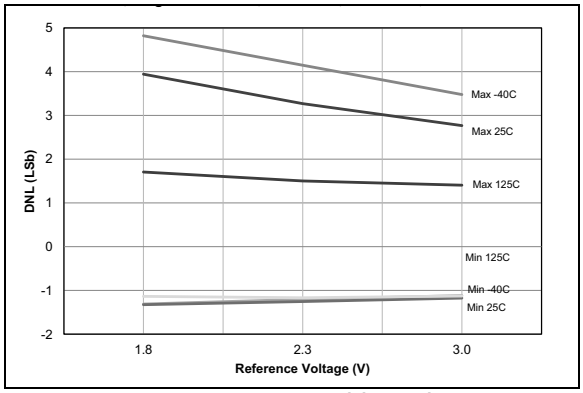


FIGURE 32-91: *ADC 12-bit Mode, Single-Ended DNL, VDD = 3.0V, TAD = 1 µS.*

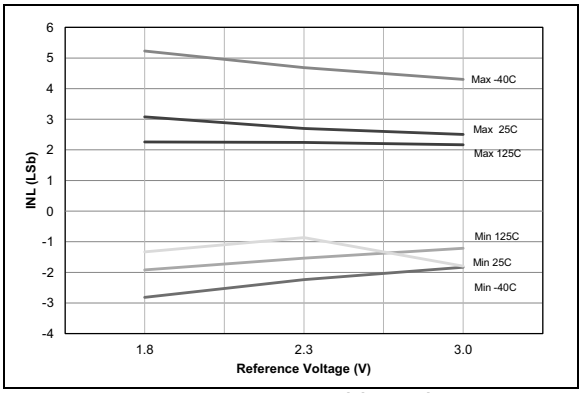


FIGURE 32-92: *ADC 12-bit Mode, Single-Ended INL, VDD = 3.0V, TAD = 1 µS.*

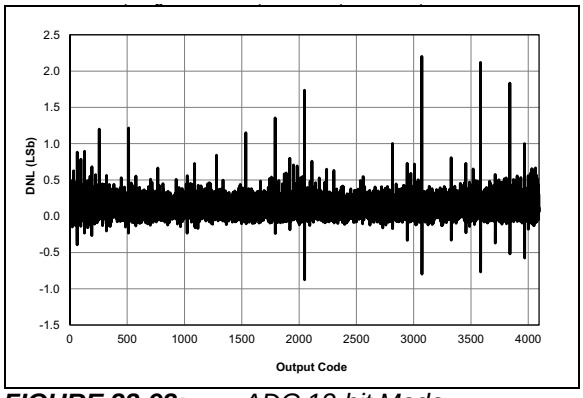


FIGURE 32-93: *ADC 12-bit Mode, Single-Ended DNL, VDD = 5.5V, TAD = 1 µS, 25°C.*

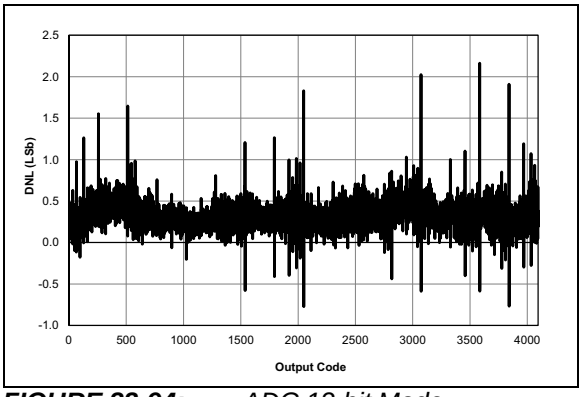


FIGURE 32-94: *ADC 12-bit Mode, Single-Ended DNL, VDD = 5.5V, TAD = 4 µS, 25°C.*

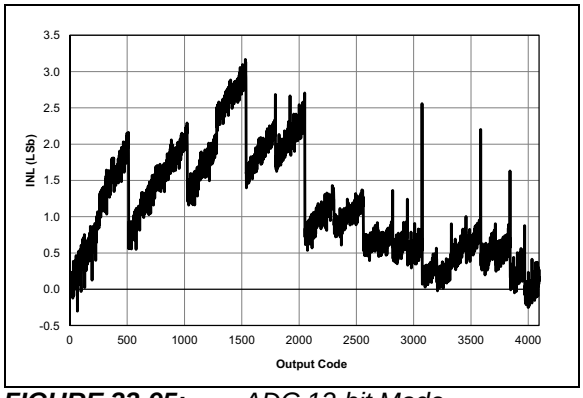


FIGURE 32-95: *ADC 12-bit Mode, Single-Ended INL, VDD = 5.5V, TAD = 1 µS, 25°C.*

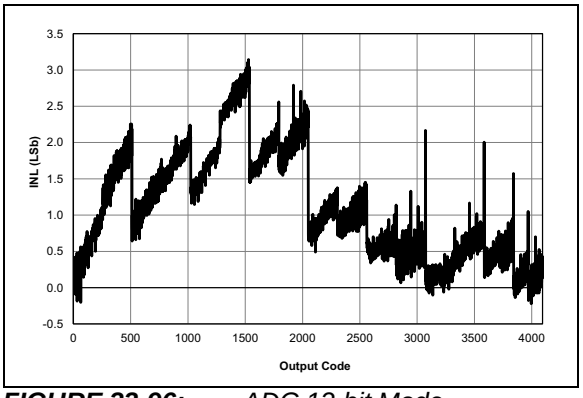


FIGURE 32-96: *ADC 12-bit Mode, Single-Ended INL, VDD = 5.5V, TAD = 4 µS, 25°C.*

33.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

33.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

33.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

33.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

33.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.