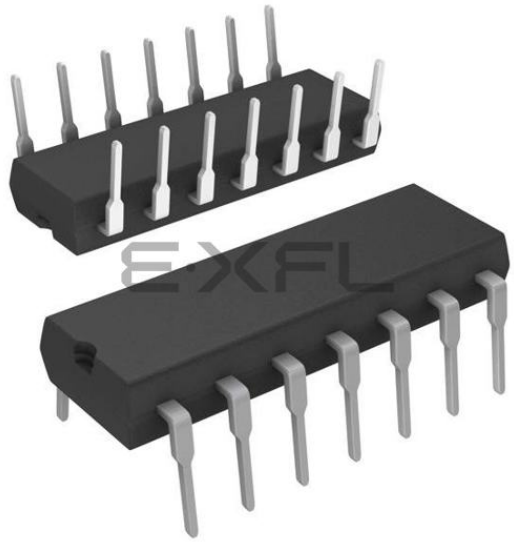


Welcome to E-XFL.COM

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, PWM, WDT
Number of I/O	12
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	FLASH
EEPROM Size	224 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 11x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	14-DIP (0.300", 7.62mm)
Supplier Device Package	14-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf15323-e-p

1.1 Register and Bit Naming Conventions

1.1.1 REGISTER NAMES

When there are multiple instances of the same peripheral in a device, the peripheral control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance, and control identifier. The control registers section will show just one instance of all the register names with an 'x' in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

1.1.2 BIT NAMES

There are two variants for bit names:

- Short name: Bit function abbreviation
- Long name: Peripheral abbreviation + short name

1.1.2.1 Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is *RegisterName*bits.*ShortName*. For example, the enable bit, EN, in the COG1CON0 register can be set in C programs with the instruction `COG1CON0bits.EN = 1`.

Short names are generally not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When this occurs, during the include file generation, all instances of that short bit name are appended with an underscore plus the name of the register in which the bit resides to avoid naming contentions.

1.1.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral thereby making every long bit name unique. The long bit name for the COG1 enable bit is the COG1 prefix, G1, appended with the enable bit short name, EN, resulting in the unique bit name G1EN.

Long bit names are useful in both C and assembly programs. For example, in C the COG1CON0 enable bit can be set with the `G1EN = 1` instruction. In assembly, this bit can be set with the `BSF COG1CON0,G1EN` instruction.

1.1.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. Bit fields adhere only to the short bit naming convention. For example, the three Least Significant bits of the COG1CON0 register contain the mode control bits. The short name for this field is MD. There is no long bit name variant. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the COG1 to the Push-Pull mode:

```
COG1CON0bits.MD = 0x5;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant mode bit has the short bit name MD2 and the long bit name is G1MD2. The following two examples demonstrate assembly program sequences for setting the COG1 to Push-Pull mode:

Example 1:

```
MOVLW  ~(1<<G1MD1)
ANDWF  COG1CON0,F
MOVLW  1<<G1MD2 | 1<<G1MD0
IORWF  COG1CON0,F
```

Example 2:

```
BSF    COG1CON0,G1MD2
BCF    COG1CON0,G1MD1
BSF    COG1CON0,G1MD0
```

1.1.3 REGISTER AND BIT NAMING EXCEPTIONS

1.1.3.1 Status, Interrupt, and Mirror Bits

Status, interrupt enables, interrupt flags, and mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

1.1.3.2 Legacy Peripherals

There are some peripherals that do not strictly adhere to these naming conventions. Peripherals that have existed for many years and are present in almost every device are the exceptions. These exceptions were necessary to limit the adverse impact of the new conventions on legacy code. Peripherals that do adhere to the new convention will include a table in the registers section indicating the long name prefix for each peripheral instance. Peripherals that fall into the exception category will not have this table. These peripherals include, but are not limited to, the following:

- EUSART
- MSSP

2.5 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 9.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for details). The PIC16(L)F15313/23 devices do not have a secondary oscillator.

The oscillator circuit should be placed on the same side of the board as the device. Place the oscillator circuit close to the respective oscillator pins with no more than 0.5 inch (12 mm) between the circuit components and the pins. The load capacitors should be placed next to the oscillator itself, on the same side of the board.

Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. Also, if using a two-sided board, avoid any traces on the other side of the board where the crystal is placed.

Layout suggestions are shown in Figure 2-3. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

In planning the application's routing and I/O assignments, ensure that adjacent port pins, and other signals in close proximity to the oscillator, are benign (i.e., free of high frequencies, short rise and fall times, and other similar noise).

For additional information and design guidance on oscillator circuits, refer to these Microchip Application Notes, available at the corporate website (www.microchip.com):

- *AN826,* "*Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices"*
- *AN849, "Basic PICmicro® Oscillator Design"*
- *AN943, "Practical PICmicro® Oscillator Analysis and Design"*
- *AN949, "Making Your Oscillator Work"*

2.6 Unused I/Os

Unused I/O pins should be configured as outputs and driven to a logic low state. Alternatively, connect a 1 kΩ to 10 kΩ resistor to VSS on unused pins and drive the output to logic low.

FIGURE 2-3: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT

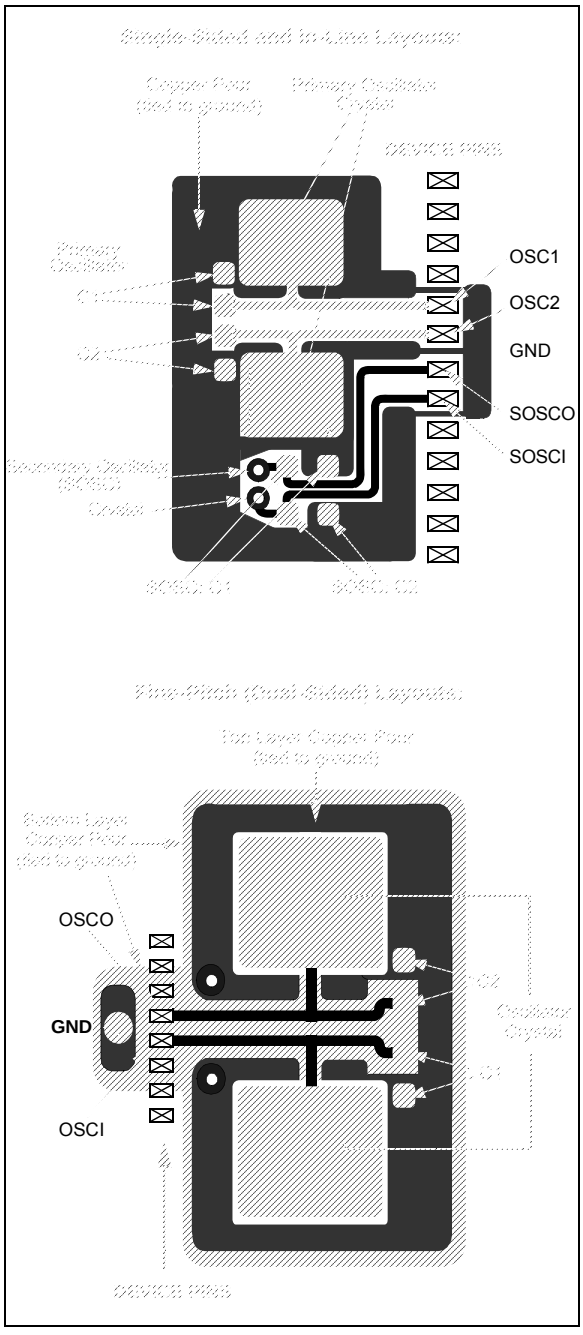


TABLE 4-10: SPECIAL FUNCTION REGISTER SUMMARY BANKS 0-63 (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on: MCLR
Bank 13											
CPU CORE REGISTERS; see Table 4-3 for specifics											
68Ch — 69Fh	—	Unimplemented								—	—

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

4.5 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figure 4-4 through Figure 4-7). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when CALL or CALLW instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer if the STVREN bit is programmed to '0' (Configuration Words). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

Note 1: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, CALLW, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

4.5.1 ACCESSING THE STACK

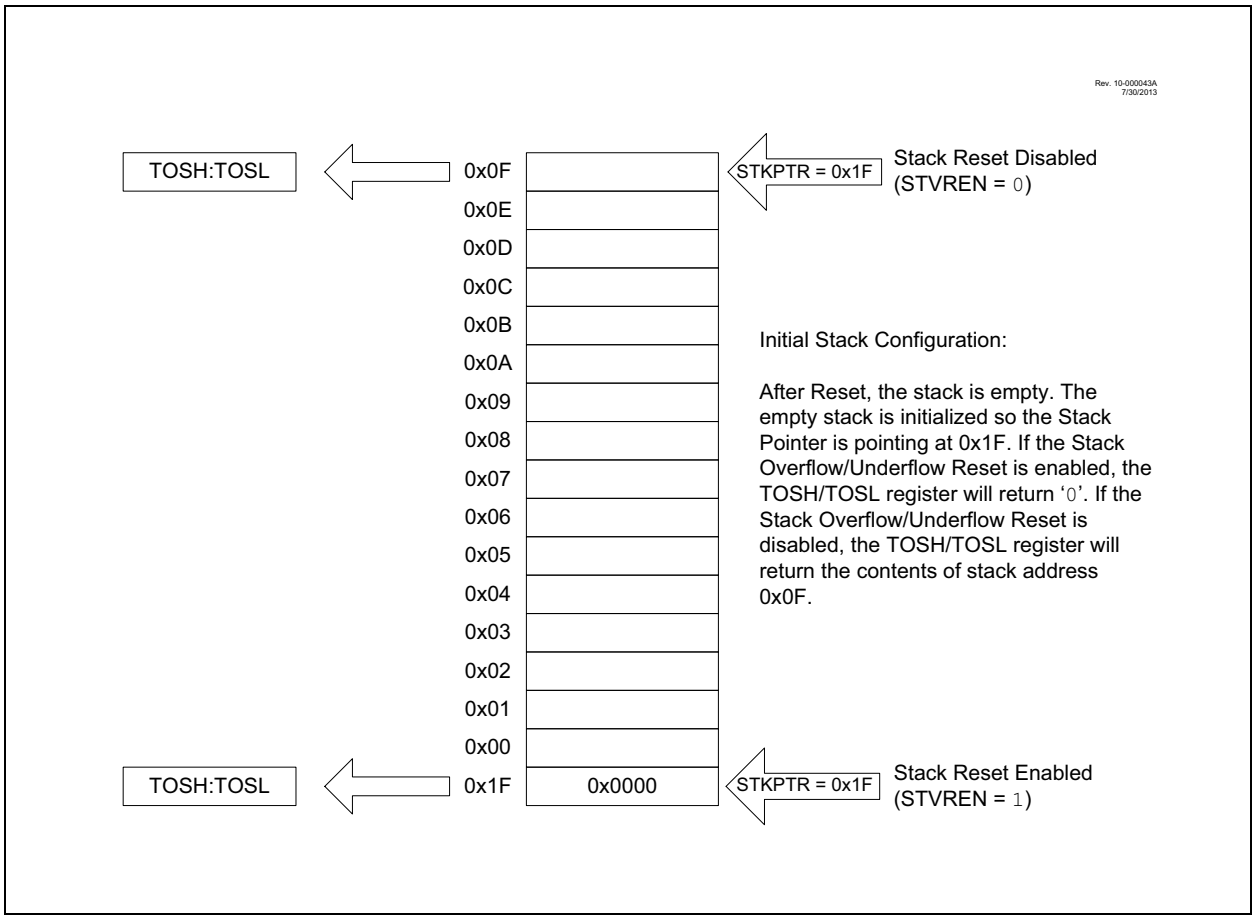
The stack is accessible through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is five bits to allow detection of overflow and underflow.

Note: Care should be taken when modifying the STKPTR while interrupts are enabled.

During normal program operation, CALL, CALLW and interrupts will increment STKPTR while RETLW, RETURN, and RETFIE will decrement STKPTR. STKPTR can be monitored to obtain to value of stack memory left at any given time. The STKPTR always points at the currently used place on the stack. Therefore, a CALL or CALLW will increment the STKPTR and then write the PC, and a return will unload the PC value from the stack and then decrement the STKPTR.

Reference Figure 4-4 through Figure 4-7 for examples of accessing the stack.

FIGURE 4-4: ACCESSING THE STACK EXAMPLE 1



REGISTER 10-8: PIE6: PERIPHERAL INTERRUPT ENABLE REGISTER 6

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	CCP2IE	CCP1IE
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	HS = Hardware set

bit 7-2 **Unimplemented:** Read as '0'.

bit 1 **CCP2IE:** CCP2 Interrupt Enable bit
- 1 = CCP2 interrupt is enabled
- 0 = CCP2 interrupt is disabled

bit 0 **CCP1IE:** CCP1 Interrupt Enable bit
- 1 = CCP1 interrupt is enabled
- 0 = CCP1 interrupt is disabled

Note: Bit PEIE of the INTCON register must be set to enable any peripheral interrupt controlled by registers PIE1-PIE7.

REGISTER 10-17: PIR7: PERIPHERAL INTERRUPT REQUEST REGISTER 7

U-0	U-0	R/W/HS-0/0	R/W/HS-0/0	U-0	U-0	U-0	R/W/HS-0/0
—	—	NVMIF	NCO1IF	—	—	—	CWG1IF
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	HS = Hardware set

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **NVMIF:** Nonvolatile Memory (NVM) Interrupt Flag bit
1 = The requested NVM operation has completed
0 = NVM interrupt not asserted

bit 4 **NCO1IF:** Numerically Controlled Oscillator (NCO) Interrupt Flag bit
1 = The NCO has rolled over
0 = No NCO interrupt event has occurred

bit 3-1 **Unimplemented:** Read as '0'

bit 0 **CWG1IF:** CWG1 Interrupt Flag bit
1 = CWG1 has gone into shutdown
0 = CWG1 is operating normally, or interrupt cleared

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Enable bit, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

11.1.2 INTERRUPTS DURING DOZE

If an interrupt occurs and the Recover-on-Interrupt bit is clear (ROI = 0) at the time of the interrupt, the Interrupt Service Routine (ISR) continues to execute at the rate selected by DOZE<2:0>. Interrupt latency is extended by the DOZE<2:0> ratio.

If an interrupt occurs and the ROI bit is set (ROI = 1) at the time of the interrupt, the DOZEN bit is cleared and the CPU executes at full speed. The prefetched instruction is executed and then the interrupt vector sequence is executed. In Figure 11-1, the interrupt occurs during the 2nd instruction cycle of the Doze period, and immediately brings the CPU out of Doze. If the Doze-On-Exit (DOE) bit is set (DOE = 1) when the RETFIE operation is executed, DOZEN is set, and the CPU executes at the reduced rate based on the DOZE<2:0> ratio.

11.2 Sleep Mode

Sleep mode is entered by executing the SLEEP instruction, while the Idle Enable (IDLEN) bit of the CPUDOZE register is clear (IDLEN = 0). If the SLEEP instruction is executed while the IDLEN bit is set (IDLEN = 1), the CPU will enter the IDLE mode (**Section 11.2.3 “Low-Power Sleep Mode”**).

Upon entering Sleep mode, the following conditions exist:

1. WDT will be cleared but keeps running if enabled for operation during Sleep
2. The $\overline{PD}$ bit of the STATUS register is cleared
3. The $\overline{TO}$ bit of the STATUS register is set
4. CPU Clock and System Clock
5. 31 kHz LFINTOSC, HFINTOSC are unaffected and peripherals using them may continue operation in Sleep.
6. ADC is unaffected if the dedicated FRC oscillator is selected the conversion will be left abandoned if FOSC is selected and ADRES will have an incorrect value
7. I/O ports maintain the status they had before Sleep was executed (driving high, low, or high-impedance). This does not apply in the case of any asynchronous peripheral which is active and may affect the I/O port value
8. Resets other than WDT are not affected by Sleep mode

Refer to individual chapters for more details on peripheral operation during Sleep.

To minimize current consumption, the following conditions should be considered:

- I/O pins should not be floating
- External circuitry sinking current from I/O pins
- Internal circuitry sourcing current from I/O pins
- Current draw from pins with internal weak pull-ups
- Modules using any oscillator

I/O pins that are high-impedance inputs should be pulled to VDD or VSS externally to avoid switching currents caused by floating inputs.

Any module with a clock source that is not FOSC can be enabled. Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See **Section 21.0 “5-Bit Digital-to-Analog Converter (DAC1) Module”**, **Section 18.0 “Fixed Voltage Reference (FVR)”** for more information on these modules.

11.2.1 WAKE-UP FROM SLEEP

The device can wake-up from Sleep through one of the following events:

1. External Reset input on $\overline{MCLR}$ pin, if enabled.
2. BOR Reset, if enabled.
3. POR Reset.
4. Watchdog Timer, if enabled.
5. Any external interrupt.
6. Interrupts by peripherals capable of running during Sleep (see individual peripheral for more information).

The first three events will cause a device Reset. The last three events are considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to **Section 8.12 “Memory Execution Violation”**.

When the SLEEP instruction is being executed, the next instruction (PC + 1) is prefetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be enabled. Wake-up will occur regardless of the state of the GIE bit. If the GIE bit is disabled, the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is enabled, the device executes the instruction after the SLEEP instruction, the device will then call the Interrupt Service Routine. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

The WDT is cleared when the device wakes-up from Sleep, regardless of the source of wake-up.

EXAMPLE 13-5: DEVICE ID ACCESS

```
; This write routine assumes the following:
; 1.  A full row of data are loaded, starting at the address in DATA_ADDR
; 2.  Each word of data to be written is made up of two adjacent bytes in DATA_ADDR,
; stored in little endian format
; 3.  A valid starting address (the least significant bits = 00000) is loaded in ADDRH:ADDRL
; 4.  ADDRH and ADDRL are located in common RAM (locations 0x70 - 0x7F)
; 5.  NVM interrupts are not taken into account

          BANKSEL             NVMADRH
          MOVF                ADDRH,W
          MOVWF               NVMADRH                 ; Load initial address
          MOVF                ADDRL,W
          MOVWF               NVMADRL
          MOVLW               LOW DATA_ADDR           ; Load initial data address
          MOVWF               FSR0L
          MOVLW               HIGH DATA_ADDR
          MOVWF               FSR0H
          BCF                 NVMCON1,NVMREGS         ; Set PFM as write location
          BSF                 NVMCON1,WREN            ; Enable writes
          BSF                 NVMCON1,LWLO            ; Load only write latches

LOOP
          MOVIW               FSR0++
          MOVWF               NVMDATL                 ; Load first data byte
          MOVIW               FSR0++
          MOVWF               NVMDATH                 ; Load second data byte

          CALL                UNLOCK_SEQ              ; If not, go load latch
          INCF                NVMADRL,F               ; Increment address
          MOVF                NVMADRL,W
          XORLW               0x1F                    ; Check if lower bits of address are 00000
          ANDLW               0x1F                    ; and if on last of 32 addresses
          BTFSC               STATUS,Z                ; Last of 32 words?
          GOTO                START_WRITE             ; If so, go write latches into memory

          GOTO                LOOP

START_WRITE
          BCF                 NVMCON1,LWLO            ; Latch writes complete, now write memory
          CALL                UNLOCK_SEQ              ; Perform required unlock sequence
          BCF                 NVMCON1,LWLO            ; Disable writes

UNLOCK_SEQ
          MOVLW               55h
          BCF                 INTCON,GIE              ; Disable interrupts
          MOVWF               NVMCON2                 ; Begin unlock sequence
          MOVLW               AAh
          MOVWF               NVMCON2
          BSF                 NVMCON1,WR
          BSF                 INTCON,GIE              ; Unlock sequence complete, re-enable interrupts
          return
```

REGISTER 22-2: NCO1CLK: NCO1 INPUT CLOCK CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
N1PWS<2:0>(1,2)			—	N1CKS<3:0>			
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 7-5 **N1PWS<2:0>:** NCO1 Output Pulse Width Select bits(1)
- 111 = NCO1 output is active for 128 input clock periods
- 110 = NCO1 output is active for 64 input clock periods
- 101 = NCO1 output is active for 32 input clock periods
- 100 = NCO1 output is active for 16 input clock periods
- 011 = NCO1 output is active for 8 input clock periods
- 010 = NCO1 output is active for 4 input clock periods
- 001 = NCO1 output is active for 2 input clock periods
- 000 = NCO1 output is active for 1 input clock period

bit 4 **Unimplemented:** Read as '0'

bit 3-0 **N1CKS<3:0>:** NCO1 Clock Source Select bits
- 1011-1111 = Reserved
- 1010 = LC4_out
- 1001 = LC3_out
- 1000 = LC2_out
- 0111 = LC1_out
- 0110 = CLKR
- 0101 = Reserved
- 0100 = MFINTOSC (32 kHz)
- 0011 = MFINTOSC (500 kHz)
- 0010 = LFINTOSC
- 0001 = HFINTOSC
- 0000 = FOSC

Note 1: N1PWS applies only when operating in Pulse Frequency mode.

TABLE 27-1: TIMER2 OPERATING MODES

Mode	MODE<4:0>		Output Operation	Operation	Timer Control		
	<4:3>	<2:0>			Start	Reset	Stop
Free Running Period	00	000	Period Pulse	Software gate (Figure 27-4)	ON = 1	—	ON = 0
		001		Hardware gate, active-high (Figure 27-5)	ON = 1 and TMRx_ers = 1	—	ON = 0 or TMRx_ers = 0
		010		Hardware gate, active-low	ON = 1 and TMRx_ers = 0	—	ON = 0 or TMRx_ers = 1
		011	Period Pulse with Hardware Reset	Rising or falling edge Reset	ON = 1	TMRx_ers ↕	ON = 0
		100		Rising edge Reset (Figure 27-6)		TMRx_ers ↑	
		101		Falling edge Reset		TMRx_ers ↓	
		110		Low level Reset		TMRx_ers = 0	ON = 0 or TMRx_ers = 0
		111		High level Reset (Figure 27-7)		TMRx_ers = 1	ON = 0 or TMRx_ers = 1
One-shot	01	000	One-shot	Software start (Figure 27-8)	ON = 1	—	ON = 0 or Next clock after TMRx = PRx (**Note 2**)
		001	Edge triggered start (**Note 1**)	Rising edge start (Figure 27-9)	ON = 1 and TMRx_ers ↑	—	
		010		Falling edge start	ON = 1 and TMRx_ers ↓	—	
		011		Any edge start	ON = 1 and TMRx_ers ↕	—	
		100	Edge triggered start and hardware Reset (**Note 1**)	Rising edge start and Rising edge Reset (Figure 27-10)	ON = 1 and TMRx_ers ↑	TMRx_ers ↑	
		101		Falling edge start and Falling edge Reset	ON = 1 and TMRx_ers ↓	TMRx_ers ↓	
		110		Rising edge start and Low level Reset (Figure 27-11)	ON = 1 and TMRx_ers ↑	TMRx_ers = 0	
		111		Falling edge start and High level Reset	ON = 1 and TMRx_ers ↓	TMRx_ers = 1	
Mono-stable	10	000	Reserved				
		001	Edge triggered start (**Note 1**)	Rising edge start (Figure 27-12)	ON = 1 and TMRx_ers ↑	—	ON = 0 or Next clock after TMRx = PRx (**Note 3**)
		010		Falling edge start	ON = 1 and TMRx_ers ↓	—	
		011		Any edge start	ON = 1 and TMRx_ers ↕	—	
Reserved		100	Reserved				
Reserved		101	Reserved				
One-shot		110	Level triggered start and hardware Reset	High level start and Low level Reset (Figure 27-13)	ON = 1 and TMRx_ers = 1	TMRx_ers = 0	ON = 0 or Held in Reset (**Note 2**)
		111		Low level start & High level Reset	ON = 1 and TMRx_ers = 0	TMRx_ers = 1	
Reserved	11	xxx	Reserved				

Note 1: If ON = 0 then an edge is required to restart the timer after ON = 1.

2: When TMRx = PRx then the next clock clears ON and stops TMRx at 00h.

3: When TMRx = PRx then the next clock stops TMRx at 00h but does not clear ON.

27.5.7 EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE

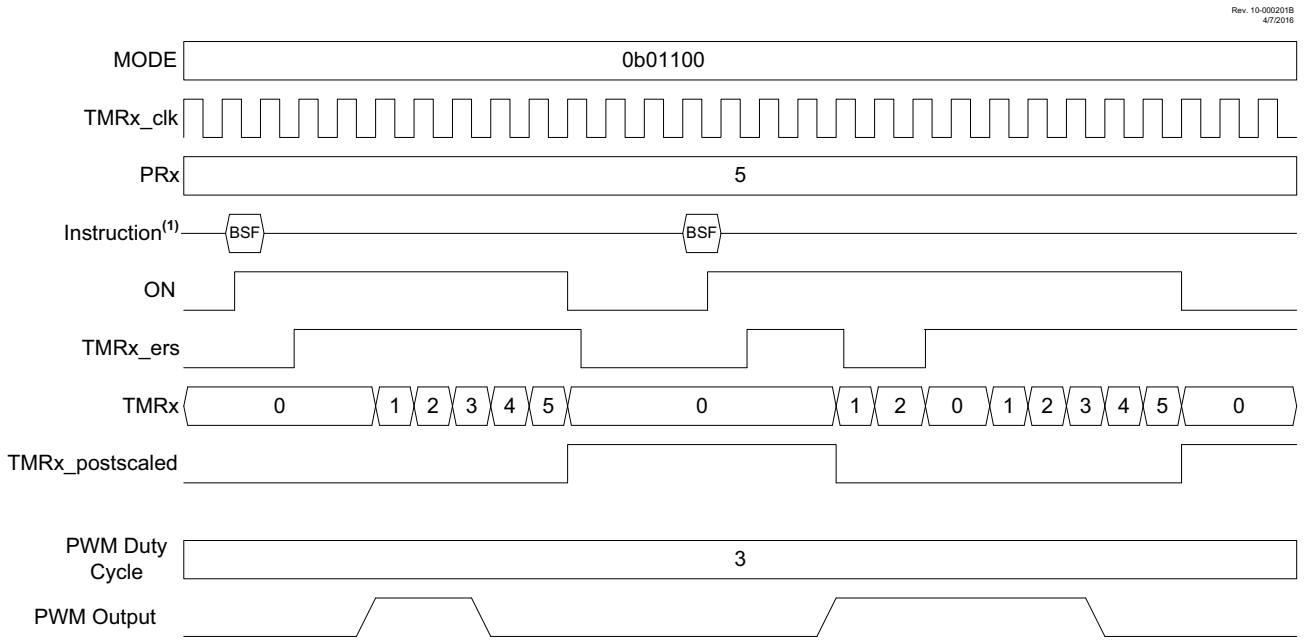
In Edge-Triggered Hardware Limit One-Shot modes the timer starts on the first external signal edge after the ON bit is set and resets on all subsequent edges. Only the first edge after the ON bit is set is needed to start the timer. The counter will resume counting automatically two clocks after all subsequent external Reset edges. Edge triggers are as follows:

- Rising edge start and Reset (MODE<4:0> = `01100`)
- Falling edge start and Reset (MODE<4:0> = `01101`)

The timer resets and clears the ON bit when the timer value matches the PRx period value. External signal edges will have no effect until after software sets the ON bit. Figure 27-10 illustrates the rising edge hardware limit one-shot operation.

When this mode is used in conjunction with the CCP then the first starting edge trigger, and all subsequent Reset edges, will activate the PWM drive. The PWM drive will deactivate when the timer matches the CCPRx pulse-width value and stay deactivated until the timer halts at the PRx period match unless an external signal edge resets the timer before the match occurs.

FIGURE 27-10: EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = `01100`)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

30.1.2 PUSH-PULL MODE

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in Figure 30-2. This alternation creates the push-pull effect required for driving some transformer-based power supply designs.

The push-pull sequencer is reset whenever EN = 0 or if an auto-shutdown event occurs. The sequencer is clocked by the first input pulse, and the first output appears on CWG1A.

The unused outputs CWG1C and CWG1D drive copies of CWG1A and CWG1B, respectively, but with polarity controlled by the POLC and POLD bits of the CWG1CON1 register, respectively.

30.1.3 FULL-BRIDGE MODES

In Forward and Reverse Full-Bridge modes, three outputs drive static values while the fourth is modulated by the input data signal. In Forward Full-Bridge mode, CWG1A is driven to its active state, CWG1B and CWG1C are driven to their inactive state, and CWG1D is modulated by the input signal. In Reverse Full-Bridge mode, CWG1C is driven to its active state, CWG1A and CWG1D are driven to their inactive states, and CWG1B is modulated by the input signal. In Full-Bridge mode, the dead-band period is used when there is a switch from forward to reverse or vice-versa. This dead-band control is described in **Section 30.5 “Dead-Band Control”**, with additional details in **Section 30.6 “Rising Edge and Reverse Dead Band”** and **Section 30.7 “Falling Edge and Forward Dead Band”**.

The mode selection may be toggled between forward and reverse toggling the MODE<0> bit of the CWG1CON0 while keeping MODE<2:1> static, without disabling the CWG module.

TABLE 31-4: SUMMARY OF REGISTERS ASSOCIATED WITH CLCx (continued)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
CLC4GLS1	—	—	LC4G2D3T	LC4G2D3N	LC4G2D2T	LC4G2D2N	LC4G2D1T	LC4G2D1N	354
CLC4GLS2	—	—	LC4G3D3T	LC4G3D3N	LC4G3D2T	LC4G3D2N	LC4G3D1T	LC4G3D1N	355
CLC4GLS3	—	—	LC4G4D3T	LC4G4D3N	LC4G4D2T	LC4G4D2N	LC4G4D1T	LC4G4D1N	356
CLCIN0PPS	—	—	CLCIN0PPS<5:0>						191
CLCIN1PPS	—	—	CLCIN1PPS<5:0>						191
CLCIN2PPS	—	—	CLCIN2PPS<5:0>						191
CLCIN3PPS	—	—	CLCIN3PPS<5:0>						191

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the CLCx modules.

32.2 SPI Mode Overview

The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select ($\overline{SS}$)

Figure 32-1 shows the block diagram of the MSSP module when operating in SPI mode.

The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

Figure 32-4 shows a typical connection between a master device and multiple slave devices.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions involve two shift registers, eight bits in size, one in the master and one in the slave. Data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new Least Significant bit (LSb) is shifted into the same register.

Figure 32-5 shows a typical connection between two processors configured as master and slave devices.

Data is shifted out of both shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDO output pin which is connected to, and received by, the slave's SDI input pin. The slave device transmits information out on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

To begin communication, the master device first sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

The master device starts a transmission by sending out the MSb from its shift register. The slave device reads this bit from that same line and saves it into the LSb position of its shift register.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its shift register (on its SDO pin) and the slave device is reading this bit and saving it as the LSb of its shift register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its shift register.

After eight bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:

- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line must disregard the clock and transmission signals and must not transmit out any data of its own.

32.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSP module configured as an I^2C slave in 10-bit Addressing mode.

Figure 32-20 is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish I^2C communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSP1STAT is set; SSP1IF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with R/$\overline{W}$ bit clear; UA bit of the SSP1STAT register is set.
4. Slave sends $\overline{ACK}$ and SSP1IF is set.
5. Software clears the SSP1IF bit.
6. Software reads received address from SSP1BUF clearing the BF flag.
7. Slave loads low address into SSP1ADD, releasing SCL.
8. Master sends matching low address byte to the slave; UA bit is set.

Note:	Updates to the SSP1ADD register are not allowed until after the $\overline{ACK}$ sequence.

9. Slave sends $\overline{ACK}$ and SSP1IF is set.

Note:	If the low address does not match, SSP1IF and UA are still set so that the slave software can set SSP1ADD back to the high address. BF is not set because there is no match. CKP is unaffected.

10. Slave clears SSP1IF.
11. Slave reads the received matching address from SSP1BUF clearing BF.
12. Slave loads high address into SSP1ADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{ACK}$ on the ninth SCL pulse; SSP1IF is set.
14. If SEN bit of SSP1CON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSP1IF.
16. Slave reads the received byte from SSP1BUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCL.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

32.5.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSP1ADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. Figure 32-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 32-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

32.6.6 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSP1BUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted. SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high. When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ is written into the ACKSTAT bit on the rising edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSP1IF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSP1BUF, leaving SCL low and SDA unchanged (Figure 32-28).

After the write to the SSP1BUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{W}$ bit are completed. On the falling edge of the eighth clock, the master will release the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{ACK}$ bit is loaded into the ACKSTAT Status bit of the SSP1CON2 register. Following the falling edge of the ninth clock transmission of the address, the SSP1IF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSP1BUF takes place, holding SCL low and allowing SDA to float.

32.6.6.1 BF Status Flag

In Transmit mode, the BF bit of the SSP1STAT register is set when the CPU writes to SSP1BUF and is cleared when all eight bits are shifted out.

32.6.6.2 WCOL Status Flag

If the user writes the SSP1BUF when a transmit is already in progress (i.e., SSP1SR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

WCOL must be cleared by software before the next transmission.

32.6.6.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSP1CON2 register is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

32.6.6.4 Typical transmit sequence:

1. The user generates a Start condition by setting the SEN bit of the SSP1CON2 register.
2. SSP1IF is set by hardware on completion of the Start.
3. SSP1IF is cleared by software.
4. The MSSP module will wait the required start time before any other operation takes place.
5. The user loads the SSP1BUF with the slave address to transmit.
6. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSP1BUF is written to.
7. The MSSP module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSP1CON2 register.
8. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSP1IF bit.
9. The user loads the SSP1BUF with eight bits of data.
10. Data is shifted out the SDA pin until all eight bits are transmitted.
11. The MSSP module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSP1CON2 register.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the PEN or RSEN bits of the SSP1CON2 register. Interrupt is generated once the Stop/Restart condition is complete.

FIGURE 32-28: I^2C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

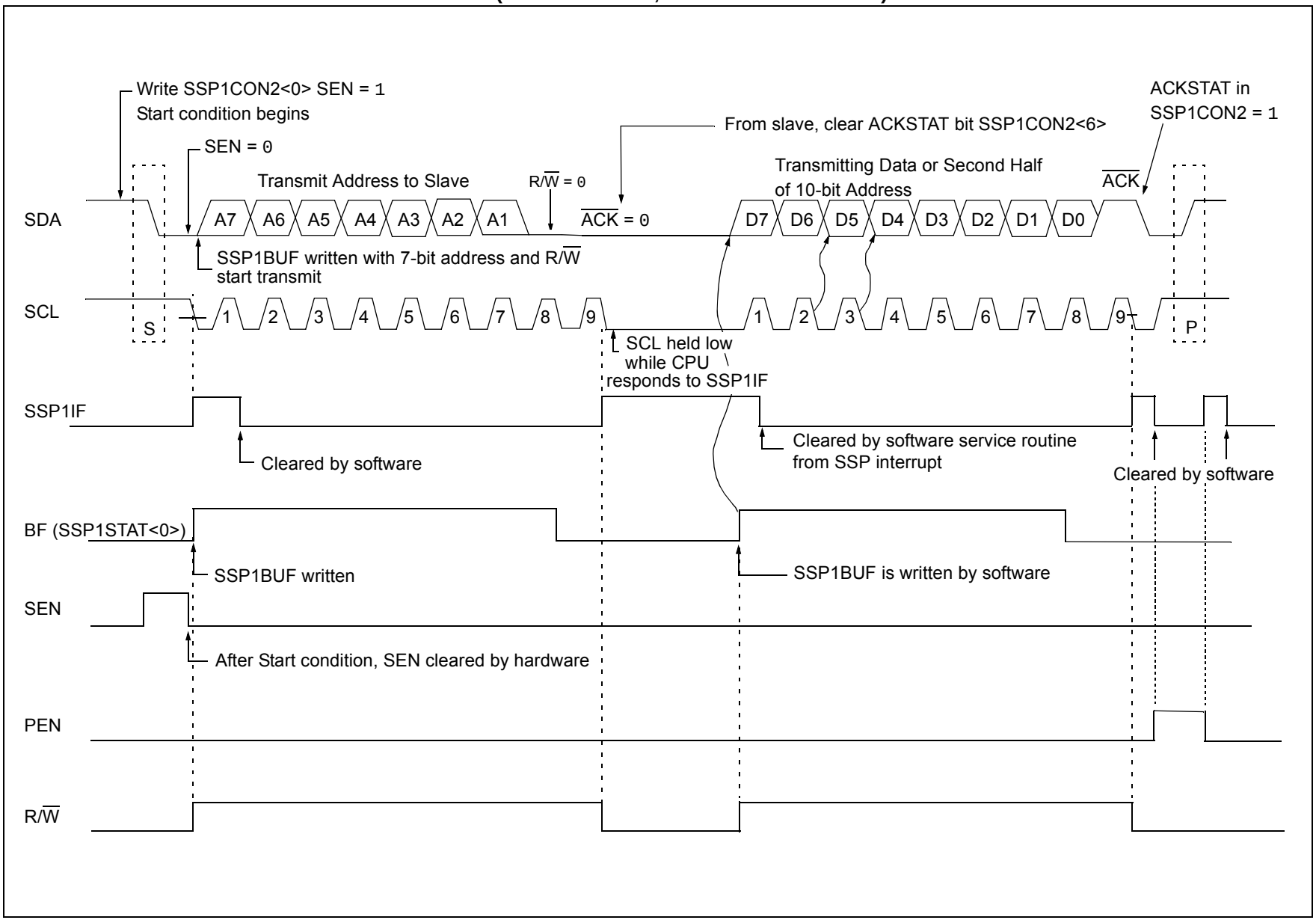


TABLE 37-4: I/O PORTS

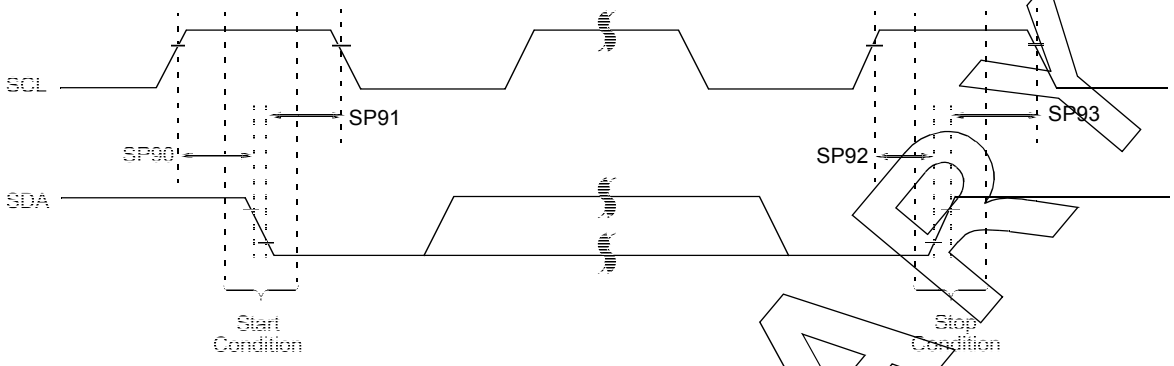
Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
	V_{IL}	**Input Low Voltage**					
		I/O PORT:					
D300		with TTL buffer	—	—	0.8	V	$4.5V \le V_{DD} \le 5.5V$
D301			—	—	0.15 V_{DD}	V	$1.8V \le V_{DD} \le 4.5V$
D302		with Schmitt Trigger buffer	—	—	0.2 V_{DD}	V	$2.0V \le V_{DD} \le 5.5V$
D303		with I^2C levels	—	—	0.3 V_{DD}	V	
D304		with SMBus levels	—	—	0.8	V	$2.7V \le V_{DD} \le 5.5V$
D305		$\overline{MCLR}$	—	—	0.2 V_{DD}	V	
	V_{IH}	**Input High Voltage**					
		I/O PORT:					
D320		with TTL buffer	2.0	—	—	V	$4.5V \le V_{DD} \le 5.5V$
D321			0.25 V_{DD} + 0.8	—	—	V	$1.8V \le V_{DD} \le 4.5V$
D322		with Schmitt Trigger buffer	0.8 V_{DD}	—	—	V	$2.0V \le V_{DD} \le 5.5V$
D323		with I^2C levels	0.7 V_{DD}	—	—	V	
D324		with SMBus levels	2.1	—	—	V	$2.7V \le V_{DD} \le 5.5V$
D325		$\overline{MCLR}$	0.7 V_{DD}	—	—	V	
	I_{IL}	**Input Leakage Current(1)**					
D340		I/O Ports	—	± 5	± 125	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C
D341			—	± 5	± 1000	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 125°C
D342		$\overline{MCLR}$(2)	—	± 50	± 200	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C
	I_{PUR}	**Weak Pull-up Current**					
D350			25	120	200	µA	V_{DD} = 3.0V, V_{PIN} = V_{SS}
	V_{OL}	**Output Low Voltage**					
D360		I/O ports	—	—	0.6	V	I_{OL} = 10.0mA, V_{DD} = 3.0V
	V_{OH}	**Output High Voltage**					
D370		I/O ports	V_{DD} - 0.7	—	—	V	I_{OH} = 6.0 mA, V_{DD} = 3.0V
D380	C_{IO}	**All I/O pins**	—	5	50	pF	

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

2: 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.

FIGURE 37-21: I²C BUS START/STOP BITS TIMING



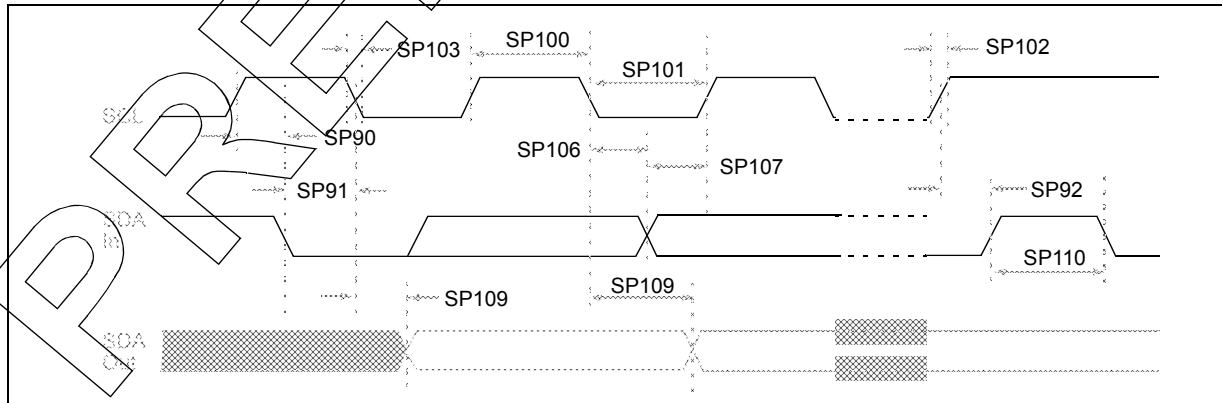
Note: Refer to Figure 37-4 for load conditions.

TABLE 37-24: I²C BUS START/STOP BITS REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)								
Param. No.	**Symbol**	**Characteristic**		**Min.**	**Typ**	**Max.**	**Units**	**Conditions**
SP90*	TSU:STA	Start condition Setup time	100 kHz mode	4700	—	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	600	—	—		
SP91*	THD:STA	Start condition Hold time	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—	—		
SP92*	TSU:STO	Stop condition Setup time	100 kHz mode	4700	—	—	ns	
			400 kHz mode	600	—	—		
SP93	THD:STO	Stop condition Hold time	100 kHz mode	4000	—	—	ns	
			400 kHz mode	600	—	—		

* These parameters are characterized but not tested.

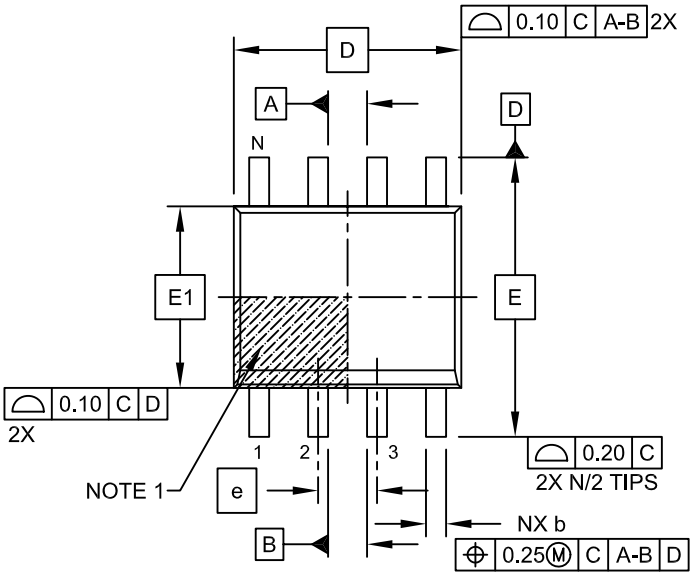
FIGURE 37-22: I²C BUS DATA TIMING



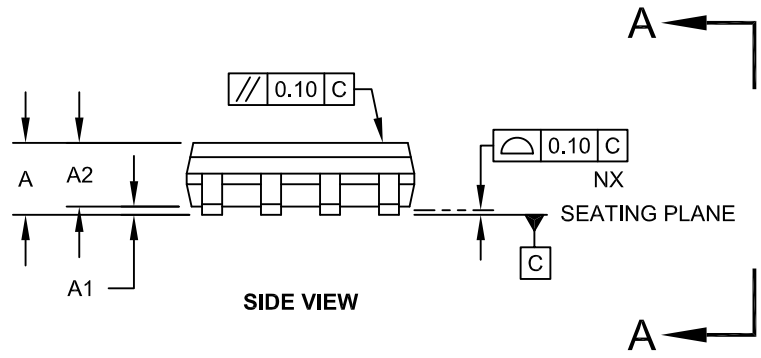
Note: Refer to Figure 37-4 for load conditions.

8-Lead Plastic Small Outline (SN) - Narrow, 3.90 mm Body [SOIC]

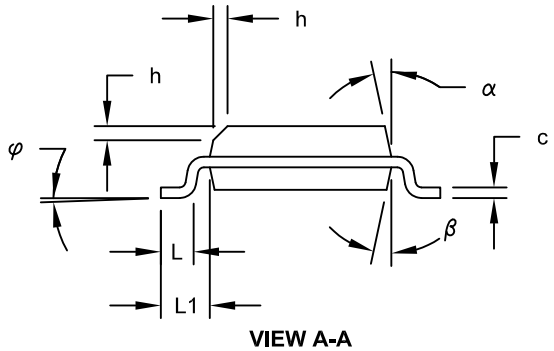
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW



VIEW A-A

Microchip Technology Drawing No. C04-057C Sheet 1 of 2