PIC12LF1552

8-Pin Flash, 8-Bit Microcontrollers

High-Performance RISC CPU

- C Compiler Optimized Architecture
- Only 49 Instructions
- 2K Words Linear Program Memory Addressing
- 256 bytes Linear Data Memory Addressing
- Operating Speed:
  - DC – 32 MHz clock input
  - DC – 125 ns instruction cycle
- Interrupt Capability with Automatic Context Saving
- 16-Level Deep Hardware Stack with Optional Overflow/Underflow Reset
- Direct, Indirect and Relative Addressing modes:
  - Two full 16-bit File Select Registers (FSRs)
  - FSRs can read program and data memory

Flexible Oscillator Structure

- 16 MHz Internal Oscillator Block:
  - Factory calibrated to ±1%, typical
  - Software selectable frequency range from 32 MHz to 31 kHz
- 4x Phase-Lock Loop (PLL), usable with 16 MHz internal oscillator
  - Allows 32 MHz software selectable clock frequency
- 31 kHz Low-Power Internal Oscillator
- Three External Clock modes up to 20 MHz

Special Microcontroller Features

- Operating Voltage Range:
  - 1.8V to 3.6V
- Self-Programmable under Software Control
- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Programmable Low-Power Brown-Out Reset (LPBOR)
- Extended Watchdog Timer (WDT):
  - Programmable period from 1 ms to 256s
- Programmable Code Protection
- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- Enhanced Low-Voltage Programming (LVP)
- Power-Saving Sleep mode:
  - Low-Power Sleep mode
  - Low-Power BOR (LPBOR)
- Integrated Temperature Indicator
- 128 Bytes High-Endurance Flash:
  - 100,000 write Flash endurance (minimum)

eXtreme Low-Power (XLP) Features

- Sleep Current:
  - 20 nA @ 1.8V, typical
- Watchdog Timer Current:
  - 200 nA @ 1.8V, typical
- Operating Current:
  - 30 μA/MHz @ 1.8V, typical

Peripheral Features

- Analog-to-Digital Converter (ADC):
  - 10-bit resolution
  - 5 external channels
  - 2 internal channels:
    - Fixed Voltage Reference
    - Temperature Indicator channel
  - Auto acquisition capability
  - Conversion available during Sleep
  - Special Event Triggers
- Hardware Capacitive Voltage Divider (CVD)
  - Double sample conversions
  - Two sets of result registers
  - Inverted acquisition
  - 7-bit pre-charge timer
  - 7-bit acquisition timer
  - Two guard ring output drives
  - Adjustable sample and hold capacitor array
- Voltage Reference module:
  - Fixed Voltage Reference (FVR) with 1.024V and 2.048V output levels
- 6 I/O Pins (1 Input-only Pin):
  - High current sink/source 25 mA/25 mA
  - Individually programmable weak pull-ups
  - Individually programmable interrupt-on-change (IOC) pins
- Timer0: 8-Bit Timer/Counter with 8-Bit Programmable Prescaler
- Master Synchronous Serial Port (MSSP) with SPI and I^2C with:
  - 7-bit address masking
  - SMBus/PMBus™ compatibility

TABLE 1: PIC12LF1552 Family Types

Device	Data Sheet Index	Program Memory Flash (words)	Data EEPROM (bytes)	SRAM (bytes)	I/Os (1)	10-bit ADCs(4)	Analog Channels(2)(3) CVD RX Channels	CVD TX Channels(5)	Timers 8/16-bit	EUSART	MSSP	PWM	Debug	XLP
PIC12LF1552	**(A)**	2048	0	256	6	1	4	1	1 / 0	-	1	-	-	Y
PIC16LF1554	**(B)**	4096	0	256	12	2	10	2	2 / 1	1	1	2	I	Y
PIC16LF1559	**(B)**	8192	0	512	18	2	16	2	2 / 1	1	1	2	I	Y
PIC16LF1566	**(C)**	8192	0	1024	25	2	23	23	3 / 1	1	2	2	I	Y
PIC16LF1567	**(C)**	8192	0	1024	36	2	34	34	3 / 1	1	2	2	I	Y

Note 1: The $\overline{\text{MCLR}}$ pin is input only.

2: Analog channels are split between the available ADCs.

3: Maximum usable analog channels assuming one pin must be assigned to output.

4: If V_{DD} > 2.4V, ADC may be overclocked 4x (T_{AD} = 0.25 µs).

5: Includes functionality of ADxGRDA output pin.

Data Sheet Index (Unshaded devices are described in this document.)

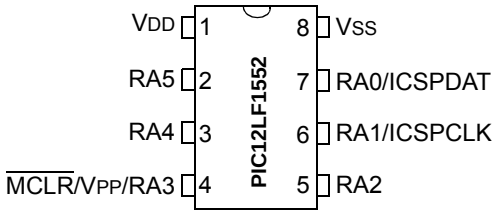
A: DS40001674 PIC12LF1552 Data Sheet, 8-Pin Flash, 8-Bit Microcontrollers

B: DS40001761 PIC16LF1554/1559 Data Sheet, 20-Pin Flash, 8-Bit Microcontrollers with XLP Technology

C: DS40001817 PIC16LF1566/1567 Data Sheet 28/40/44-Pin Flash, 8-Bit Microcontrollers with XLP Technology

Note: For other small form-factor package availability and marking information, please visit **http://www.microchip.com/packaging** or contact your local sales office.

FIGURE 1: 8-PIN PDIP, SOIC, MSOP, UDFN



Note: See Table 2 for location of all peripheral functions.

TABLE 2: 8-PIN ALLOCATION TABLE

I/O	8-Pin PDIP/SOIC/MSOP/UDFN	ADC/Hardware CVD	Reference	Timer	MSSP	Interrupt	Pull-Up	Basic
RA0	7	AN0	—	—	SDO(1) $\overline{SS}$(2)	IOC	Y	ICSPDAT
RA1	6	AN1	VREF+	—	SCK SCL	IOC	Y	ICSPCLK
RA2	5	AN2 ADOUT	—	T0CKI	SDI(1) SDA(1)	INT IOC	Y	—
RA3	4	—	—	—	$\overline{SS}$(1) SDA(2) SDI(2)	IOC	Y	$\overline{MCLR}$ VPP
RA4	3	AN3 ADGRDA	—	—	SDO(2)	IOC	Y	CLKOUT
RA5	2	AN4 ADGRDB	—	—	—	IOC	Y	CLKIN
VDD	1	—	—	—	—	—	—	VDD
VSS	8	—	—	—	—	—	—	VSS

Note 1: Default location for peripheral pin function. Alternate location can be selected using the APFCON register.
2: Alternate location for peripheral pin function selected by the APFCON register.

Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Website at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Website; **http://www.microchip.com**
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our website at **www.microchip.com** to receive the most current information on all of our products.

1.0 DEVICE OVERVIEW

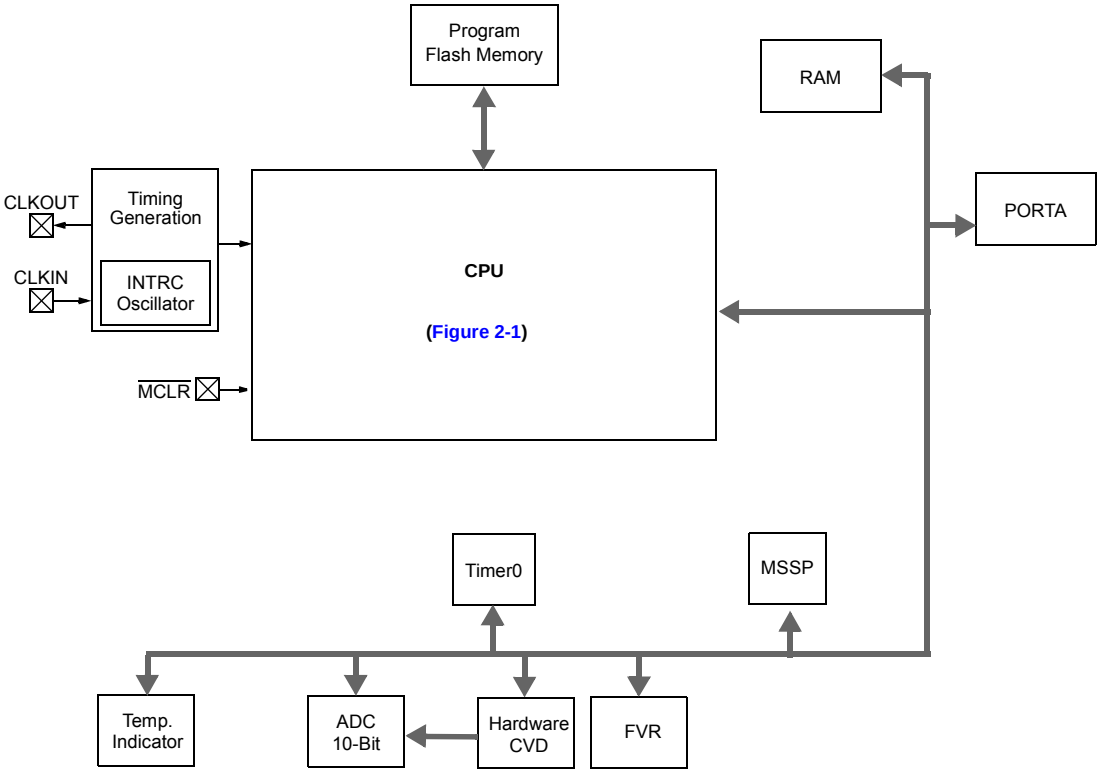
The PIC12LF1552 are described within this data sheet. They are available in 8-pin packages. Figure 1-1 shows a block diagram of the PIC12LF1552 devices. Table 1-2 shows the pinout descriptions.

Reference Table 1-1 for peripherals available per device.

TABLE 1-1: DEVICE PERIPHERAL SUMMARY

Peripheral		PIC12LF1552
Analog-to-Digital Converter (ADC)		●
Hardware Capacitor Voltage Divider (CVD)		●
Fixed Voltage Reference (FVR)		●
Temperature Indicator		●
Master Synchronous Serial Ports		
	MSSP1	●
Timers		
	Timer0	●

FIGURE 1-1: PIC12LF1552 BLOCK DIAGRAM



Note 1: See applicable chapters for more information on peripherals.
2: See Table 1-1 for peripherals available on specific devices.

TABLE 1-2: PIC12LF1552 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0/SDO(1)/$\overline{SS}$(2)/ ICSPDAT	RA0	TTL	CMOS	General purpose I/O.
	AN0	AN	—	ADC Channel input.
	SDO	—	CMOS	SPI data output.
	$\overline{SS}$	ST	—	Slave Select input.
	ICSPDAT	ST	CMOS	ICSP™ Data I/O.
RA1/AN1/VREF+/SCK/SCL/ ICSPCLK	RA1	TTL	CMOS	General purpose I/O.
	AN1	AN	—	ADC Channel input.
	VREF+	AN	—	ADC Positive Voltage Reference input.
	SCK	ST	CMOS	SPI clock.
	SCL	I^2C	OD	I^2C clock.
	ICSPCLK	ST	—	ICSP™ Programming Clock.
RA2/AN2/ADOUT/T0CKI/ SDI(1)/SDA(1)/INT	RA2	ST	CMOS	General purpose I/O.
	AN2	AN	—	ADC Channel input.
	ADOUT	CMOS	—	ADC with CVD output.
	T0CKI	ST	—	Timer0 clock input.
	SDI	ST	—	SPI data input.
	SDA	I^2C	OD	I^2C data input/output.
	INT	ST	—	External interrupt.
RA3/$\overline{MCLR}$/VPP/$\overline{SS}$(1)/SDI(2)/ SDA(2)	RA3	TTL	—	General purpose input.
	$\overline{MCLR}$	ST	—	Master Clear with internal pull-up.
	VPP	HV	—	Programming voltage.
	$\overline{SS}$	ST	—	Slave Select input.
	SDI	ST	—	SPI data input.
	SDA	I^2C	OD	I^2C data input/output.
RA4/AN3/SDO(2)/CLKOUT/ ADGRDA	RA4	TTL	CMOS	General purpose I/O.
	AN3	AN	—	ADC Channel input.
	SDO	—	CMOS	SPI data output.
	CLKOUT	—	CMOS	FOSC/4 output.
	ADGRDA	—	CMOS	Guard ring output A.
RA5/AN4/CLKIN/ADGRDB	RA5	TTL	CMOS	General purpose I/O.
	AN4	AN	—	ADC Channel input.
	CLKIN	CMOS	—	External clock input (EC mode).
	ADGRDB	—	CMOS	Guard ring output B.
VDD	VDD	Power	—	Positive supply.
VSS	VSS	Power	—	Ground reference.

Legend: AN = Analog input or output; CMOS = CMOS compatible input or output; OD = Open-Drain
TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels; I^2C = Schmitt Trigger input with I^2C levels
HV = High Voltage; XTAL = Crystal

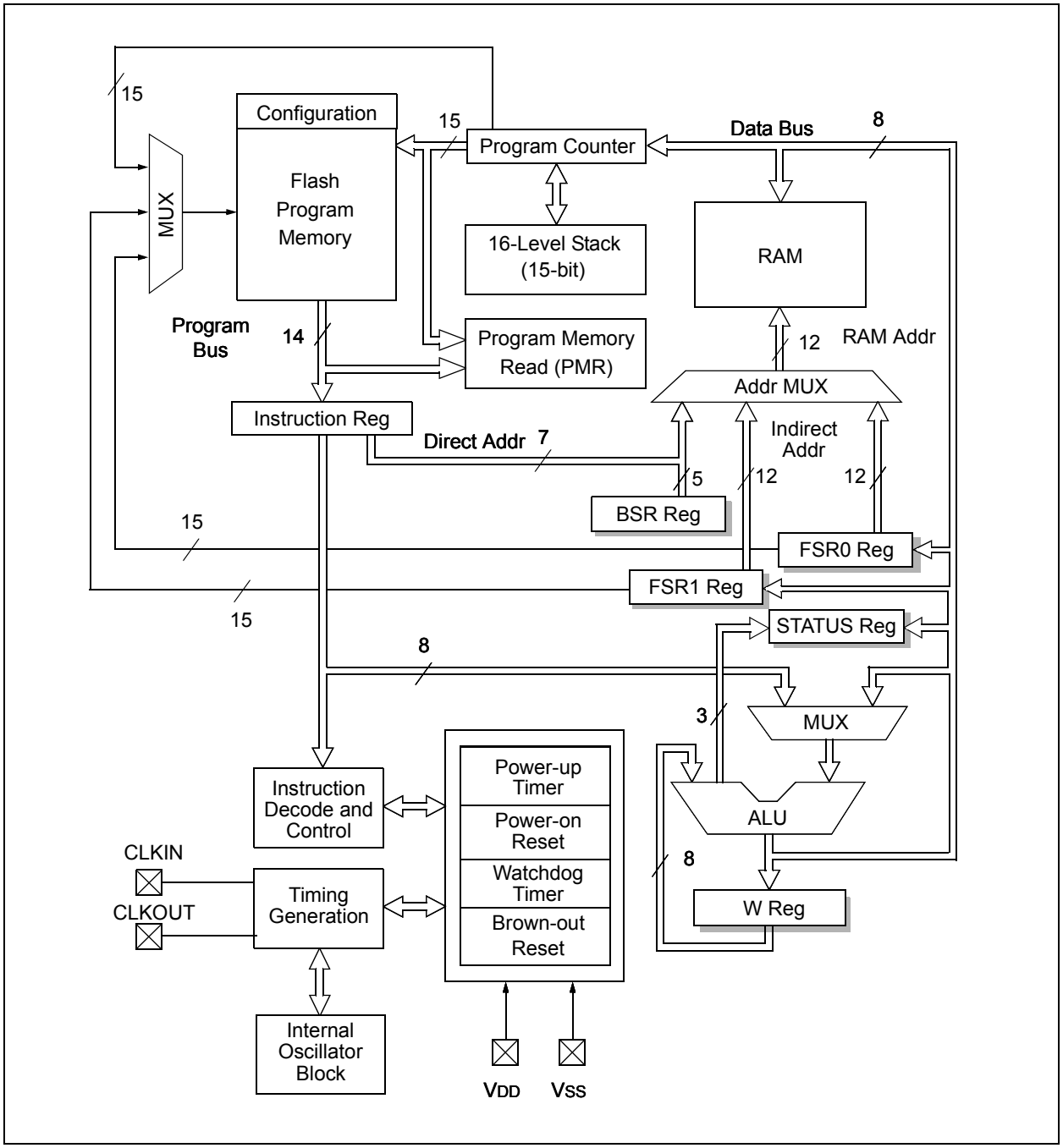
Note 1: Default location for peripheral pin function. Alternate location can be selected using the APFCON register.
2: Alternate location for peripheral pin function selected by the APFCON register.

2.0 ENHANCED MID-RANGE CPU

This family of devices contain an enhanced mid-range 8-bit CPU core. The CPU has 49 instructions. Interrupt capability includes automatic context saving. The hardware stack is 16 levels deep and has Overflow and Underflow Reset capability. Direct, Indirect, and Relative addressing modes are available. Two File Select Registers (FSRs) provide the ability to read program and data memory.

- Automatic Interrupt Context Saving
- 16-level Stack with Overflow and Underflow
- File Select Registers
- Instruction Set

FIGURE 2-1: CORE BLOCK DIAGRAM



2.1 Automatic Interrupt Context Saving

During interrupts, certain registers are automatically saved in shadow registers and restored when returning from the interrupt. This saves stack space and user code. See **Section 7.5 "Automatic Context Saving"**, for more information.

2.2 16-Level Stack with Overflow and Underflow

These devices have an external stack memory 15 bits wide and 16 words deep. A Stack Overflow or Underflow will set the appropriate bit (STKOVF or STKUNF) in the PCON register and, if enabled, will cause a software Reset. See section **Section 3.5 "Stack"** for more details.

2.3 File Select Registers

There are two 16-bit File Select Registers (FSR). FSRs can access all file registers and program memory, which allows one Data Pointer for all memory. When an FSR points to program memory, there is one additional instruction cycle in instructions using INDF to allow the data to be fetched. General purpose memory can now also be addressed linearly, providing the ability to access contiguous data larger than 80 bytes. There are also new instructions to support the FSRs. See **Section 3.6 "Indirect Addressing"** for more details.

2.4 Instruction Set

There are 49 instructions for the enhanced mid-range CPU to support the features of the CPU. See **Section 20.0 "Instruction Set Summary"** for more details.

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

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

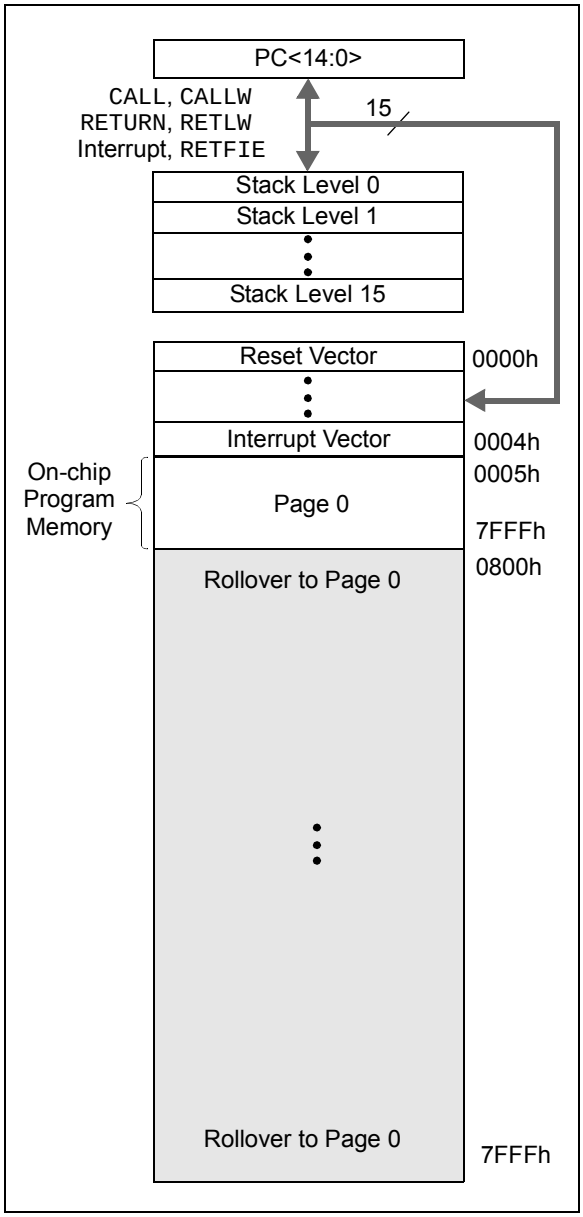
The enhanced mid-range core has a 15-bit program counter capable of addressing a 32K x 14 program memory space. Table 3-1 shows the memory sizes implemented. Accessing a location above these boundaries will cause a wrap-around within the implemented memory space. The Reset vector is at 0000h and the 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	High-Endurance Flash Memory Address Range (1)
PIC12LF1552	2,048	07FFh	0780h-07FFh

Note 1: High-endurance Flash applies to the low byte of each address in the range.

FIGURE 3-1: PROGRAM MEMORY MAP AND STACK FOR PIC12LF1552



3.1.1 READING PROGRAM MEMORY AS DATA

There are two methods of accessing constants in program memory. The first method is to use tables of RETLW instructions. The second method is to set an FSR to point to the program memory.

3.1.1.1 RETLW Instruction

The RETLW instruction can be used to provide access to tables of constants. The recommended way to create such a table is shown in Example 3-1.

EXAMPLE 3-1: RETLW INSTRUCTION

```
constants
    BRW                 ;Add Index in W to
                        ;program counter to
                        ;select data
    RETLW DATA0         ;Index0 data
    RETLW DATA1         ;Index1 data
    RETLW DATA2
    RETLW DATA3


my_function
    ;… LOTS OF CODE…
    MOVLW       DATA_INDEX
    call constants
    ;… THE CONSTANT IS IN W
```

The BRW instruction makes this type of table very simple to implement. If your code must remain portable with previous generations of microcontrollers, then the BRW instruction is not available so the older table read method must be used.

3.1.1.2 Indirect Read with FSR

The program memory can be accessed as data by setting bit 7 of the FSRxH register and reading the matching INDFx register. The MOVIW instruction will place the lower eight bits of the addressed word in the W register. Writes to the program memory cannot be performed via the INDF registers. Instructions that access the program memory via the FSR require one extra instruction cycle to complete. Example 3-2 demonstrates accessing the program memory via an FSR.

The High directive will set bit<7> if a label points to a location in program memory.

EXAMPLE 3-2: ACCESSING PROGRAM MEMORY VIA FSR

```
constants
       DW     DATA0          ; First
              constant
       DW     DATA1          ; Second
              constant
       DW     DATA2
       DW     DATA3
my_function
       ;… LOTS OF CODE…
        MOVLW   DATA_INDEX
        ADDLW   LOW constants
        MOVWF   FSR1L
        MOVLW   HIGH constants; MSb is
             set automatically
       MOVWF   FSR1H
       BTFSC   STATUS,C     ; carry from
               ADDLW?
       INCF    FSR1H,f   ; yes
       MOVIW  0[FSR1]
;THE PROGRAM MEMORY IS IN W
```

3.2 Data Memory Organization

The data memory is partitioned into 32 memory banks with 128 bytes in a bank. Each bank consists of (Figure 3-2):

- 12 core registers
- 20 Special Function Registers (SFR)
- Up to 80 bytes of General Purpose RAM (GPR)
- 16 bytes of common RAM

The active bank is selected by writing the bank number into the Bank Select Register (BSR). Unimplemented memory will read as '0'. All data memory can be accessed either directly (via instructions that use the file registers) or indirectly via the two File Select Registers (FSR). See **Section 3.6 "Indirect Addressing"** for more information.

Data memory uses a 12-bit address. The upper seven bits of the address define the Bank address and the lower five bits select the registers/RAM in that bank.

3.2.1 CORE REGISTERS

The core registers contain the registers that directly affect the basic operation. The core registers occupy the first 12 addresses of every data memory bank (addresses x00h/x08h through x0Bh/x8Bh). These registers are listed below in Table 3-2. For detailed information, see Table 3-5.

TABLE 3-2: CORE REGISTERS

Addresses	BANKx
x00h or x80h	INDF0
x01h or x81h	INDF1
x02h or x82h	PCL
x03h or x83h	STATUS
x04h or x84h	FSR0L
x05h or x85h	FSR0H
x06h or x86h	FSR1L
x07h or x87h	FSR1H
x08h or x88h	BSR
x09h or x89h	WREG
x0Ah or x8Ah	PCLATH
x0Bh or x8Bh	INTCON

3.2.1.1 STATUS Register

The STATUS register, shown in Register 3-1, contains:

- the arithmetic status of the ALU
- the Reset status

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as '`000u u1uu`' (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect any Status bits. For other instructions not affecting any Status bits (Refer to **Section 20.0 "Instruction Set Summary"**).

Note 1: The C and DC bits operate as $\overline{Borrow}$ and $\overline{Digit\ Borrow}$ out bits, respectively, in subtraction.

3.3 Register Definitions: Status

REGISTER 3-1: STATUS: STATUS REGISTER

U-0	U-0	U-0	R-1/q	R-1/q	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	$\overline{TO}$	$\overline{PD}$	Z	DC(1)	C(1)
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	q = Value depends on condition

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **$\overline{TO}$:** Time-Out bit
1 = After power-up, `CLRWDT` instruction or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$:** Power-Down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{Digit\ Borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)(1)
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/$\overline{Borrow}$ bit(1) (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)(1)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note 1: For $\overline{Borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high-order or low-order bit of the source register.

3.3.1 SPECIAL FUNCTION REGISTER

The Special Function Registers are registers used by the application to control the desired operation of peripheral functions in the device. The Special Function Registers occupy the 20 bytes after the core registers of every data memory bank (addresses x0Ch/x8Ch through x1Fh/x9Fh). The registers associated with the operation of the peripherals are described in the appropriate peripheral chapter of this data sheet.

3.3.2 GENERAL PURPOSE RAM

There are up to 80 bytes of GPR in each data memory bank. The Special Function Registers occupy the 20 bytes after the core registers of every data memory bank (addresses x0Ch/x8Ch through x1Fh/x9Fh).

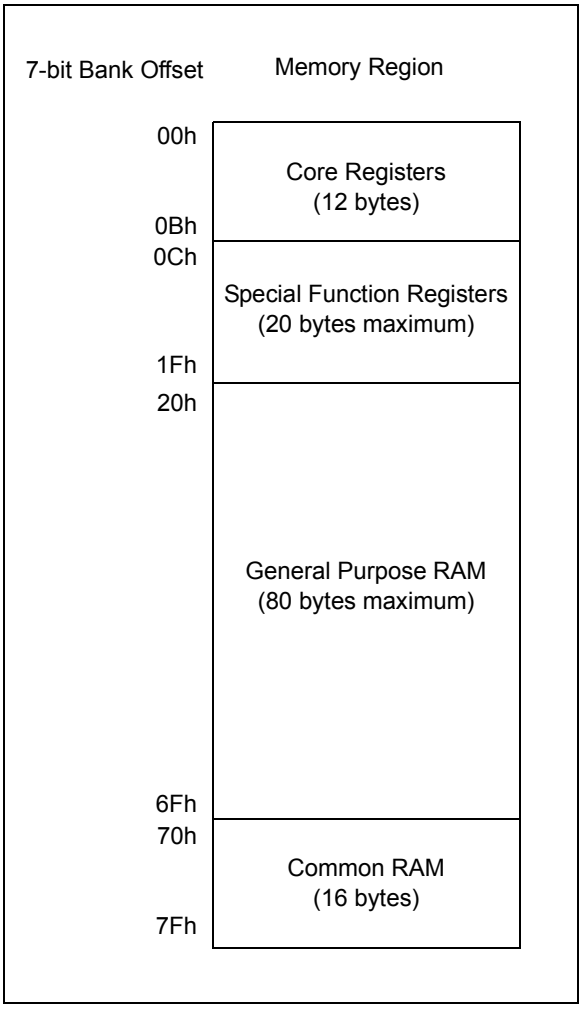
3.3.2.1 Linear Access to GPR

The general purpose RAM can be accessed in a non-banked method via the FSRs. This can simplify access to large memory structures. See **Section 3.6.2 "Linear Data Memory"** for more information.

3.3.3 COMMON RAM

There are 16 bytes of common RAM accessible from all banks.

FIGURE 3-2: BANKED MEMORY PARTITIONING



3.3.4 DEVICE MEMORY MAPS

The memory maps for PIC12LF1552 are as shown in Table 3-3.

TABLE 3-3: PIC12LF1552 MEMORY MAP

BANK 0		BANK 1		BANK 2		BANK 3		BANK 4		BANK 5		BANK 6		BANK 7	
000h	Core Registers (Table 3-2)	080h	Core Registers (Table 3-2)	100h	Core Registers (Table 3-2)	180h	Core Registers (Table 3-2)	200h	Core Registers (Table 3-2)	280h	Core Registers (Table 3-2)	300h	Core Registers (Table 3-2)	380h	Core Registers (Table 3-2)
00Bh		08Bh		10Bh		18Bh		20Bh		28Bh		30Bh		38Bh	
00Ch	PORTA	08Ch	TRISA	10Ch	LATA	18Ch	ANSELA	20Ch	WPUA	28Ch	—	30Ch	—	38Ch	—
00Dh	—	08Dh	—	10Dh	—	18Dh	—	20Dh	—	28Dh	—	30Dh	—	38Dh	—
00Eh	—	08Eh	—	10Eh	—	18Eh	—	20Eh	—	28Eh	—	30Eh	—	38Eh	—
00Fh	—	08Fh	—	10Fh	—	18Fh	—	20Fh	—	28Fh	—	30Fh	—	38Fh	—
010h	—	090h	—	110h	—	190h	—	210h	—	290h	—	310h	—	390h	—
011h	PIR1	091h	PIE1	111h	—	191h	PMADRL	211h	SSPBUF	291h	—	311h	—	391h	IOCAP
012h	PIR2	092h	PIE2	112h	—	192h	PMADRH	212h	SSPADD	292h	—	312h	—	392h	IOCAN
013h	—	093h	—	113h	—	193h	PMDATL	213h	SSPMSK	293h	—	313h	—	393h	IOCAF
014h	—	094h	—	114h	—	194h	PMDATH	214h	SSPSTAT	294h	—	314h	—	394h	—
015h	TMR0	095h	OPTION_REG	115h	—	195h	PMCON1	215h	SSPCON1	295h	—	315h	—	395h	—
016h	—	096h	PCON	116h	BORCON	196h	PMCON2	216h	SSPCON2	296h	—	316h	—	396h	—
017h	—	097h	WDTCON	117h	FVRCON	197h	—	217h	SSPCON3	297h	—	317h	—	397h	—
018h	—	098h	—	118h	—	198h	—	218h	—	298h	—	318h	—	398h	—
019h	—	099h	OSCCON	119h	—	199h	—	219h	—	299h	—	319h	—	399h	—
01Ah	—	09Ah	OSCSTAT	11Ah	—	19Ah	—	21Ah	—	29Ah	—	31Ah	—	39Ah	—
01Bh	—	09Bh	ADRESL(1)	11Bh	—	19Bh	—	21Bh	—	29Bh	—	31Bh	—	39Bh	—
01Ch	—	09Ch	ADRESH(1)	11Ch	—	19Ch	—	21Ch	—	29Ch	—	31Ch	—	39Ch	—
01Dh	—	09Dh	ADCON0(1)	11Dh	APFCON	19Dh	—	21Dh	—	29Dh	—	31Dh	—	39Dh	—
01Eh	—	09Eh	ADCON1(1)	11Eh	—	19Eh	—	21Eh	—	29Eh	—	31Eh	—	39Eh	—
01Fh	—	09Fh	ADCON2(1)	11Fh	—	19Fh	—	21Fh	—	29Fh	—	31Fh	—	39Fh	—
020h	General Purpose Register 48 Bytes	0A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h	Unimplemented Read as '0'	220h	Unimplemented Read as '0'	2A0h	Unimplemented Read as '0'	320h	Unimplemented Read as '0'	3A0h	Unimplemented Read as '0'
07Fh		0EFh		16Fh		1EFh		26Fh		2EFh		36Fh		3EFh	
		0F0h	Common RAM (Accesses 70h – 7Fh)	170h	Common RAM (Accesses 70h – 7Fh)	1F0h	Accesses 70h – 7Fh	270h	Accesses 70h – 7Fh	2F0h	Accesses 70h – 7Fh	370h	Accesses 70h – 7Fh	3F0h	Accesses 70h – 7Fh
		0FFh		17Fh		1FFh		27Fh		2FFh		37Fh		3FFh	

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

Note 1: These ADC registers are the same as the registers in Bank 14.

TABLE 3-3: PIC12LF1552 MEMORY MAP (CONTINUED)

BANK 8		BANK 9		BANK 10		BANK 11		BANK 12		BANK 13		BANK 14		BANK 15	
400h	Core Registers (Table 3-2)	480h	Core Registers (Table 3-2)	500h	Core Registers (Table 3-2)	580h	Core Registers (Table 3-2)	600h	Core Registers (Table 3-2)	680h	Core Registers (Table 3-2)	700h	Core Registers (Table 3-2)	780h	Core Registers (Table 3-2)
40Bh		48Bh		50Bh		58Bh		60Bh		68Bh		70Bh		78Bh	
40Ch	—	48Ch	—	50Ch	—	58Ch	—	60Ch	—	68Ch	—	70Ch	—	78Ch	—
40Dh	—	48Dh	—	50Dh	—	58Dh	—	60Dh	—	68Dh	—	70Dh	—	78Dh	—
40Eh	—	48Eh	—	50Eh	—	58Eh	—	60Eh	—	68Eh	—	70Eh	—	78Eh	—
40Fh	—	48Fh	—	50Fh	—	58Fh	—	60Fh	—	68Fh	—	70Fh	—	78Fh	—
410h	—	490h	—	510h	—	590h	—	610h	—	690h	—	710h	—	790h	—
411h	—	491h	—	511h	—	591h	—	611h	—	691h	—	711h	AADCON0(1)	791h	—
412h	—	492h	—	512h	—	592h	—	612h	—	692h	—	712h	AADCON1(1)	792h	—
413h	—	493h	—	513h	—	593h	—	613h	—	693h	—	713h	AADCON2(1)	793h	—
414h	—	494h	—	514h	—	594h	—	614h	—	694h	—	714h	AADCON3	794h	—
415h	—	495h	—	515h	—	595h	—	615h	—	695h	—	715h	AADSTAT	795h	—
416h	—	496h	—	516h	—	596h	—	616h	—	696h	—	716h	AADPRE	796h	—
417h	—	497h	—	517h	—	597h	—	617h	—	697h	—	717h	AADACQ	797h	—
418h	—	498h	—	518h	—	598h	—	618h	—	698h	—	718h	AADGRD	798h	—
419h	—	499h	—	519h	—	599h	—	619h	—	699h	—	719h	AADCAP	799h	—
41Ah	—	49Ah	—	51Ah	—	59Ah	—	61Ah	—	69Ah	—	71Ah	AADRES0L(1)	79Ah	—
41Bh	—	49Bh	—	51Bh	—	59Bh	—	61Bh	—	69Bh	—	71Bh	AADRES0H(1)	79Bh	—
41Ch	—	49Ch	—	51Ch	—	59Ch	—	61Ch	—	69Ch	—	71Ch	AADRES1L	79Ch	—
41Dh	—	49Dh	—	51Dh	—	59Dh	—	61Dh	—	69Dh	—	71Dh	AADRES1H	79Dh	—
41Eh	—	49Eh	—	51Eh	—	59Eh	—	61Eh	—	69Eh	—	71Eh	—	79Eh	—
41Fh	—	49Fh	—	51Fh	—	59Fh	—	61Fh	—	69Fh	—	71Fh	—	79Fh	—
420h	Unimplemented Read as '0'	4A0h	Unimplemented Read as '0'	520h	Unimplemented Read as '0'	5A0h	Unimplemented Read as '0'	620h	Unimplemented Read as '0'	6A0h	Unimplemented Read as '0'	720h	Unimplemented Read as '0'	7A0h	Unimplemented Read as '0'
46Fh		4EFh		56Fh		5EFh		66Fh		6EFh		76Fh		7EFh	
470h	Accesses 70h – 7Fh	4F0h	Accesses 70h – 7Fh	570h	Accesses 70h – 7Fh	5F0h	Accesses 70h – 7Fh	670h	Accesses 70h – 7Fh	6F0h	Accesses 70h – 7Fh	770h	Accesses 70h – 7Fh	7F0h	Accesses 70h – 7Fh
47Fh		4FFh		57Fh		5FFh		67Fh		6FFh		77Fh		7FFh	

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

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

Note 1: These ADC registers are the same as the registers in Bank 1.

TABLE 3-3: PIC12LF1552 MEMORY MAP (CONTINUED)

BANK 24		BANK 25		BANK 26		BANK 27		BANK 28		BANK 29		BANK 30		BANK 31	
C00h	Core Registers (Table 3-2)	C80h	Core Registers (Table 3-2)	D00h	Core Registers (Table 3-2)	D80h	Core Registers (Table 3-2)	E00h	Core Registers (Table 3-2)	E80h	Core Registers (Table 3-2)	F00h	Core Registers (Table 3-2)	F80h	Core Registers (Table 3-2)
C0Bh		C8Bh		D0Bh		D8Bh		E0Bh		E8Bh		F0Bh		F8Bh	
C0Ch	—	C8Ch	—	D0Ch	—	D8Ch	—	E0Ch	—	E8Ch	—	F0Ch	—	F8Ch	See Table 3-4 for register mapping details
C0Dh	—	C8Dh	—	D0Dh	—	D8Dh	—	E0Dh	—	E8Dh	—	F0Dh	—	F8Dh	
C0Eh	—	C8Eh	—	D0Eh	—	D8Eh	—	E0Eh	—	E8Eh	—	F0Eh	—	F8Eh	
C0Fh	—	C8Fh	—	D0Fh	—	D8Fh	—	E0Fh	—	E8Fh	—	F0Fh	—	F8Fh	
C10h	—	C90h	—	D10h	—	D90h	—	E10h	—	E90h	—	F10h	—	F90h	
C11h	—	C91h	—	D11h	—	D91h	—	E11h	—	E91h	—	F11h	—	F91h	
C12h	—	C92h	—	D12h	—	D92h	—	E12h	—	E92h	—	F12h	—	F92h	
C13h	—	C93h	—	D13h	—	D93h	—	E13h	—	E93h	—	F13h	—	F93h	
C14h	—	C94h	—	D14h	—	D94h	—	E14h	—	E94h	—	F14h	—	F94h	
C15h	—	C95h	—	D15h	—	D95h	—	E15h	—	E95h	—	F15h	—	F95h	
C16h	—	C96h	—	D16h	—	D96h	—	E16h	—	E96h	—	F16h	—	F96h	
C17h	—	C97h	—	D17h	—	D97h	—	E17h	—	E97h	—	F17h	—	F97h	
C18h	—	C98h	—	D18h	—	D98h	—	E18h	—	E98h	—	F18h	—	F98h	
C19h	—	C99h	—	D19h	—	D99h	—	E19h	—	E99h	—	F19h	—	F99h	
C1Ah	—	C9Ah	—	D1Ah	—	D9Ah	—	E1Ah	—	E9Ah	—	F1Ah	—	F9Ah	
C1Bh	—	C9Bh	—	D1Bh	—	D9Bh	—	E1Bh	—	E9Bh	—	F1Bh	—	F9Bh	
C1Ch	—	C9Ch	—	D1Ch	—	D9Ch	—	E1Ch	—	E9Ch	—	F1Ch	—	F9Ch	
C1Dh	—	C9Dh	—	D1Dh	—	D9Dh	—	E1Dh	—	E9Dh	—	F1Dh	—	F9Dh	
C1Eh	—	C9Eh	—	D1Eh	—	D9Eh	—	E1Eh	—	E9Eh	—	F1Eh	—	F9Eh	
C1Fh	—	C9Fh	—	D1Fh	—	D9Fh	—	E1Fh	—	E9Fh	—	F1Fh	—	F9Fh	
C20h	Unimplemented Read as '0'	CA0h	Unimplemented Read as '0'	D20h	Unimplemented Read as '0'	DA0h	Unimplemented Read as '0'	E20h	Unimplemented Read as '0'	EA0h	Unimplemented Read as '0'	F20h	Unimplemented Read as '0'	FA0h	
C6Fh		CEFh		D6Fh		DEFh		E6Fh		EEFh		F6Fh		FEFh	
C70h	Accesses 70h – 7Fh	CF0h	Accesses 70h – 7Fh	D70h	Accesses 70h – 7Fh	DF0h	Accesses 70h – 7Fh	E70h	Accesses 70h – 7Fh	EF0h	Accesses 70h – 7Fh	F70h	Accesses 70h – 7Fh	FF0h	Accesses 70h – 7Fh
CFFh		CFFh		D7Fh		DFFh		E7Fh		EFFh		F7Fh		FFFh	

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

Note 1: These ADC registers are the same as the registers in Bank 1.

TABLE 3-4: PIC12LF1552 MEMORY MAP DETAIL (BANK 31)

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'.

3.3.5 CORE FUNCTION REGISTERS SUMMARY

The Core Function registers listed in Table 3-5 can be addressed from any Bank.

TABLE 3-5: CORE FUNCTION REGISTERS SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 0-31											
x00h or x80h	INDF0	Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
x01h or x81h	INDF1	Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
x02h or x82h	PCL	Program Counter (PC) Least Significant Byte								0000 0000	0000 0000
x03h or x83h	STATUS	—	—	—	$\overline{TO}$	$\overline{PD}$	Z	DC	C	---1 1000	---q quuu
x04h or x84h	FSR0L	Indirect Data Memory Address 0 Low Pointer								0000 0000	uuuu uuuu
x05h or x85h	FSR0H	Indirect Data Memory Address 0 High Pointer								0000 0000	0000 0000
x06h or x86h	FSR1L	Indirect Data Memory Address 1 Low Pointer								0000 0000	uuuu uuuu
x07h or x87h	FSR1H	Indirect Data Memory Address 1 High Pointer								0000 0000	0000 0000
x08h or x88h	BSR	—	—	—	BSR<4:0>					---0 0000	---0 0000
x09h or x89h	WREG	Working Register								0000 0000	uuuu uuuu
x0Ah or x8Ah	PCLATH	—	Write Buffer for the upper 7 bits of the Program Counter							-000 0000	-000 0000
x0Bh or x8Bh	INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	0000 0000	0000 0000

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

TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 0											
00Ch	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxx	--xx xxxx
00Eh to 010h	—	Unimplemented								—	—
011h	PIR1	—	ADIF	—	—	SSPIF	—	—	—	-0-- 0---	-0-- 0---
012h	PIR2	—	—	—	—	BCLIF	—	—	—	---- 0---	---- 0---
013h	—	Unimplemented								—	—
014h	—	Unimplemented								—	—
015h	TMR0	Holding Register for the 8-bit Timer0 Count								xxxx xxxx	uuuu uuuu
016h to 01Fh	—	Unimplemented								—	—
Bank 1											
08Ch	TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	--11 1111	--11 1111
08Dh	—	Unimplemented								—	—
08Eh	—	Unimplemented								—	—
08Fh	—	Unimplemented								—	—
090h	—	Unimplemented								—	—
091h	PIE1	—	ADIE	—	—	SSPIE	—	—	—	-0-- 0---	-0-- 0---
092h	PIE2	—	—	—	—	BCLIE	—	—	—	---- 0---	---- 0---
093h	—	Unimplemented								—	—
094h	—	Unimplemented								—	—
095h	OPTION_REG	$\overline{\text{WPUEN}}$	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>			1111 1111	1111 1111
096h	PCON	STKOVF	STKUNF	—	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	00-1 11qq	qq-q qquu
097h	WDTCON	—	—	WDTPS<4:0>					SWDTEN	--01 0110	--01 0110
098h	—	Unimplemented								—	—
099h	OSCCON	SPLLEN	IRCF<3:0>				—	SCS<1:0>		0011 1-00	0011 1-00
09Ah	OSCSTAT	—	PLLR	—	HFIOFR	—	—	LFIOFR	HFIOFS	-0-0 --00	-q-q --qq
09Bh	ADRESL(2)	ADC Result Register 0 Low								xxxx xxxx	uuuu uuuu
09Ch	ADRESH(2)	ADC Result Register 0 High								xxxx xxxx	uuuu uuuu
09Dh	ADCON0(2)	—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON	-000 0000	-000 0000
09Eh	ADCON1(2)	ADFM	ADCS<2:0>			—	—	ADPREF<1:0>		0000 --00	0000 --00
09Fh	ADCON2(2)	TRIGSEL<2:0>				—	—	—	—	0000 ----	0000 ----
Bank 2											
10Ch	LATA	—	—	LATA5	LATA4	—	LATA2	LATA1	LATA0	--xx -xxx	--uu -uuu
10Dh to 115h	—	Unimplemented								—	—
116h	BORCON	SBOREN	BORFS	—	—	—	—	—	BORRDY	10-- ---q	uu-- ---u
117h	FVRCON	FVREN	FVRRDY	TSEN	TSRNG	—	—	ADFVR<1:0>		0q00 --00	0q00 --00
118h to 11Ch	—	Unimplemented								—	—
11Dh	APFCON	—	SDOSEL	SSSEL	SDSEL	—	—	—	—	-000 ----	-000 ----
11Eh	—									—	—
11Fh	—	Unimplemented								—	—

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

Note 1: Unimplemented, read as '1'.

2: This register is available in Bank 1 and Bank 14 under similar register names. See **Section 16.1.11 "Hardware CVD Register Mapping"**.

TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY (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 all other Resets
Bank 3											
18Ch	ANSELA	—	—	ANSA5	ANSA4	—	ANSA2	ANSA1	ANSA0	--11 -111	--11 -111
18Dh	—	Unimplemented								—	—
18Eh	—	Unimplemented								—	—
18Fh	—	Unimplemented								—	—
190h	—	Unimplemented								—	—
191h	PMADRL	Flash Program Memory Address Register Low Byte								0000 0000	0000 0000
192h	PMADRH	—(1)	Flash Program Memory Address Register High Byte							1000 0000	1000 0000
193h	PMDATL	Flash Program Memory Read Data Register Low Byte								xxxx xxxx	uuuu uuuu
194h	PMDATH	—	—	Flash Program Memory Read Data Register High Byte						--xx xxxx	--uu uuuu
195h	PMCON1	—(1)	CFGS	LWLO	FREE	WRERR	WREN	WR	RD	0000 x000	0000 q000
196h	PMCON2	Flash Program Memory Control Register 2								0000 0000	0000 0000
197h to 19Fh	—	Unimplemented								—	—
Bank 4											
20Ch	WPUA	—	—	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	--11 1111	--11 1111
20Dh to 210h	—	Unimplemented								—	—
211h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
212h	SSPADD	ADD<7:0>								0000 0000	0000 0000
213h	SSPMSK	MSK<7:0>								1111 1111	1111 1111
214h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000
215h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				0000 0000	0000 0000
216h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
217h	SSPCON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN	0000 0000	0000 0000
218h to 21Fh	—	Unimplemented								—	—
Bank 5											
28Ch to 29Fh	—	Unimplemented								—	—
Bank 6											
30Ch to 31Fh	—	Unimplemented								—	—

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

Note 1: Unimplemented, read as '1'.

2: This register is available in Bank 1 and Bank 14 under similar register names. See **Section 16.1.11 "Hardware CVD Register Mapping"**.

TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY (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 all other Resets
Bank 7											
38Ch to 390h	—	Unimplemented								—	—
391h	IOCAP	—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0	--00 0000	--00 0000
392h	IOCAN	—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0	--00 0000	--00 0000
393h	IOCAF	—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0	--00 0000	--00 0000
394h to 39Fh	—	Unimplemented								—	—
Bank 8-13											
x0Ch/ x8Ch — x1Fh/ x9Fh	—	Unimplemented								—	—
Bank 14											
70Ch to 710h	—	Unimplemented								—	—
711h	AADCON0(2)	—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON	-000 0000	-000 0000
712h	AADCON1(2)	ADFM	ADCS<2:0>			—	—	ADPREF<1:0>		0000 --00	0000 --00
713h	AADCON2(2)	—	TRIGSEL<2:0>			—	—	—	—	-000 ----	-000 ----
714h	AADCON3	ADEPPOL	ADIPPOL	—	ADOEN	ADOOEN	—	ADIPEN	ADDSEN	0000 0-00	0000 0-00
715h	AADSTAT	—	—	—	—	—	ADCONV	ADSTG<1:0>		---- -000	---- -000
716h	AADPRE	—	ADPRE<6:0>							-000 0000	-000 0000
717h	AADACQ	—	ADACQ<6:0>							-000 0000	-000 0000
718h	AADGRD	GRDBOE	GRDAOE	GRDPOL	—	—	—	—	—	000- ----	000- ----
719h	AADCAP	—	—	—	—	—	ADCAP<2:0>			---- -000	---- -000
71Ah	AADRES0L(2)	ADC Result Register 0 Low								xxxx xxxx	uuuu uuuu
71Bh	AADRES0H(2)	ADC Result Register 0 High								xxxx xxxx	uuuu uuuu
71Ch	AADRES1L(2)	ADC Result Register 1 Low								xxxx xxxx	uuuu uuuu
71Dh	AADRES1H(2)	ADC Result Register 1 High								xxxx xxxx	uuuu uuuu
71Eh	—	Unimplemented								—	—
71Fh	—	Unimplemented								—	—

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

Note 1: Unimplemented, read as '1'.

2: This register is available in Bank 1 and Bank 14 under similar register names. See **Section 16.1.11 "Hardware CVD Register Mapping"**.

TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

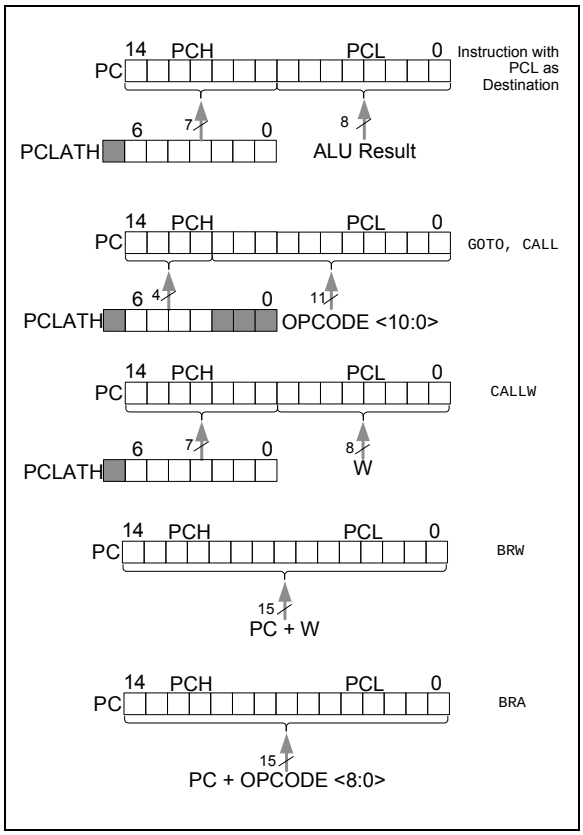
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Banks 15-30											
x0Ch/ x8Ch — x1Fh/ x9Fh	—	Unimplemented								—	—
Bank 31											
F8Ch — FE3h	—	Unimplemented								—	—
FE4h	STATUS_ SHAD	—	—	—	—	—	Z_SHAD	DC_SHAD	C_SHAD	---- -xxx	---- -uuu
FE5h	WREG_ SHAD	Working Register Shadow								xxxx xxxx	uuuu uuuu
FE6h	BSR_ SHAD	—	—	—	Bank Select Register Shadow					---x xxxx	---u uuuu
FE7h	PCLATH_ SHAD	—	Program Counter Latch High Register Shadow							-xxx xxxx	uuuu uuuu
FE8h	FSR0L_ SHAD	Indirect Data Memory Address 0 Low Pointer Shadow								xxxx xxxx	uuuu uuuu
FE9h	FSR0H_ SHAD	Indirect Data Memory Address 0 High Pointer Shadow								xxxx xxxx	uuuu uuuu
FEAh	FSR1L_ SHAD	Indirect Data Memory Address 1 Low Pointer Shadow								xxxx xxxx	uuuu uuuu
FEBh	FSR1H_ SHAD	Indirect Data Memory Address 1 High Pointer Shadow								xxxx xxxx	uuuu uuuu
FECh	—	Unimplemented								—	—
FEDh	STKPTR	—	—	—	Current Stack Pointer					---1 1111	---1 1111
FEEh	TOSL	Top-of-Stack Low byte								xxxx xxxx	uuuu uuuu
FEFh	TOSH	—	Top-of-Stack High byte							-xxx xxxx	-uuu uuuu

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, r = reserved. Shaded locations are unimplemented, read as '0'.
Note 1: Unimplemented, read as '1'.
2: This register is available in Bank 1 and Bank 14 under similar register names. See **Section 16.1.11 "Hardware CVD Register Mapping"**.

3.4 PCL and PCLATH

The Program Counter (PC) is 15 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<14:8>) is not directly readable or writable and comes from PCLATH. On any Reset, the PC is cleared. Figure 3-3 shows the five situations for the loading of the PC.

FIGURE 3-3: LOADING OF PC IN DIFFERENT SITUATIONS



3.4.1 MODIFYING PCL

Executing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC<14:8> bits (PCH) to be replaced by the contents of the PCLATH register. This allows the entire contents of the program counter to be changed by writing the desired upper seven bits to the PCLATH register. When the lower eight bits are written to the PCL register, all 15 bits of the program counter will change to the values contained in the PCLATH register and those being written to the PCL register.

3.4.2 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When performing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to Application Note AN556, *"Implementing a Table Read"* (DS00556).

3.4.3 COMPUTED FUNCTION CALLS

A computed function CALL allows programs to maintain tables of functions and provide another way to execute state machines or look-up tables. When performing a table read using a computed function CALL, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block).

If using the CALL instruction, the PCH<2:0> and PCL registers are loaded with the operand of the CALL instruction. PCH<6:3> is loaded with PCLATH<6:3>.

The CALLW instruction enables computed calls by combining PCLATH and W to form the destination address. A computed CALLW is accomplished by loading the W register with the desired address and executing CALLW. The PCL register is loaded with the value of W and PCH is loaded with PCLATH.

3.4.4 BRANCHING

The branching instructions add an offset to the PC. This allows relocatable code and code that crosses page boundaries. There are two forms of branching, BRW and BRA. The PC will have incremented to fetch the next instruction in both cases. When using either branching instruction, a PCL memory boundary may be crossed.

If using BRW, load the W register with the desired unsigned address and execute BRW. The entire PC will be loaded with the address PC + 1 + W.

If using BRA, the entire PC will be loaded with PC + 1 +, the signed value of the operand of the BRA instruction.

3.5 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figures 3-4 through 3-7). 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.

3.5.1 ACCESSING THE STACK

The stack is available 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. At any time, STKPTR can be inspected to see how much stack is left. 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 and then decrement the STKPTR.

Reference Figure 3-4 through Figure 3-7 for examples of accessing the stack.

FIGURE 3-4: ACCESSING THE STACK EXAMPLE 1

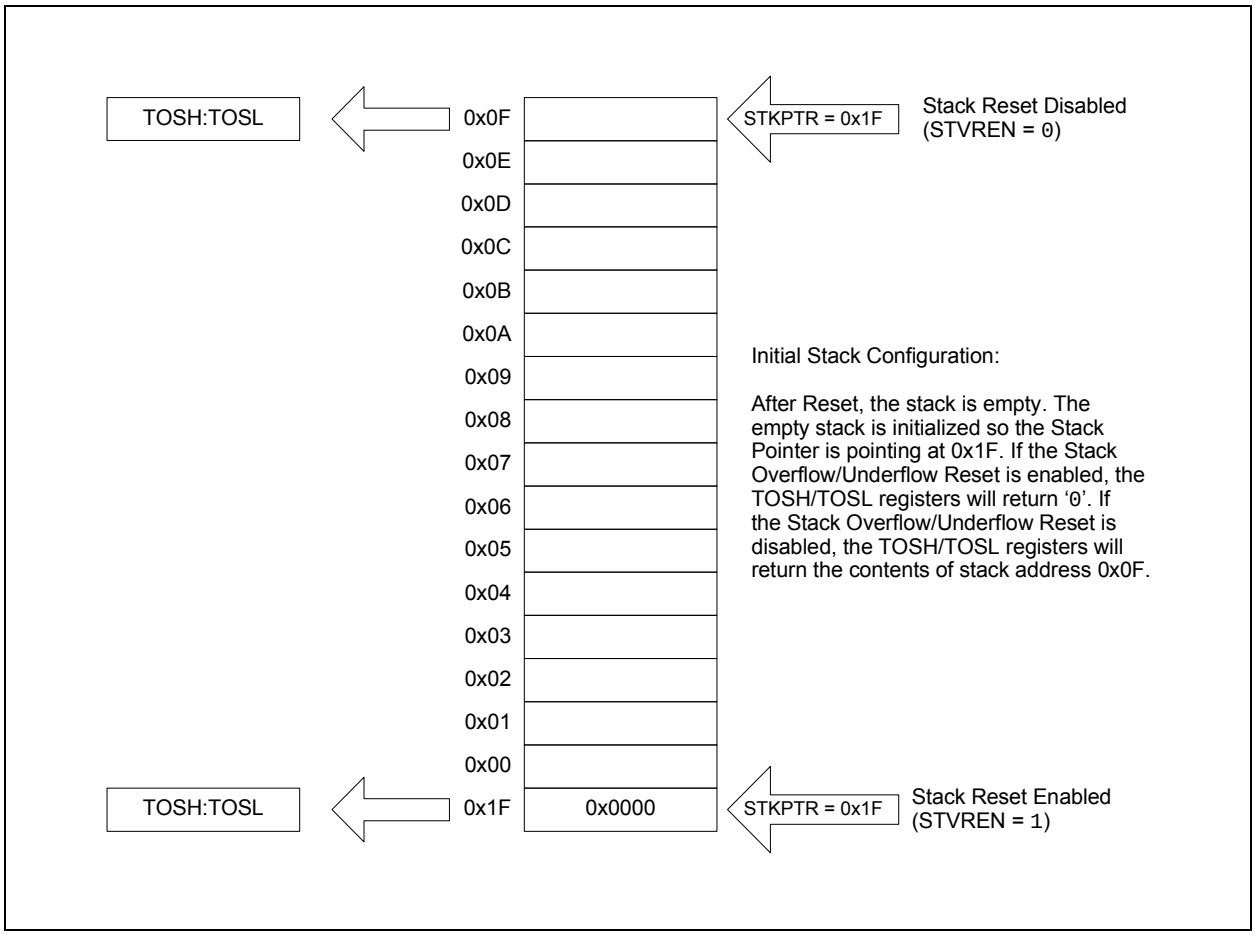


FIGURE 3-5: ACCESSING THE STACK EXAMPLE 2

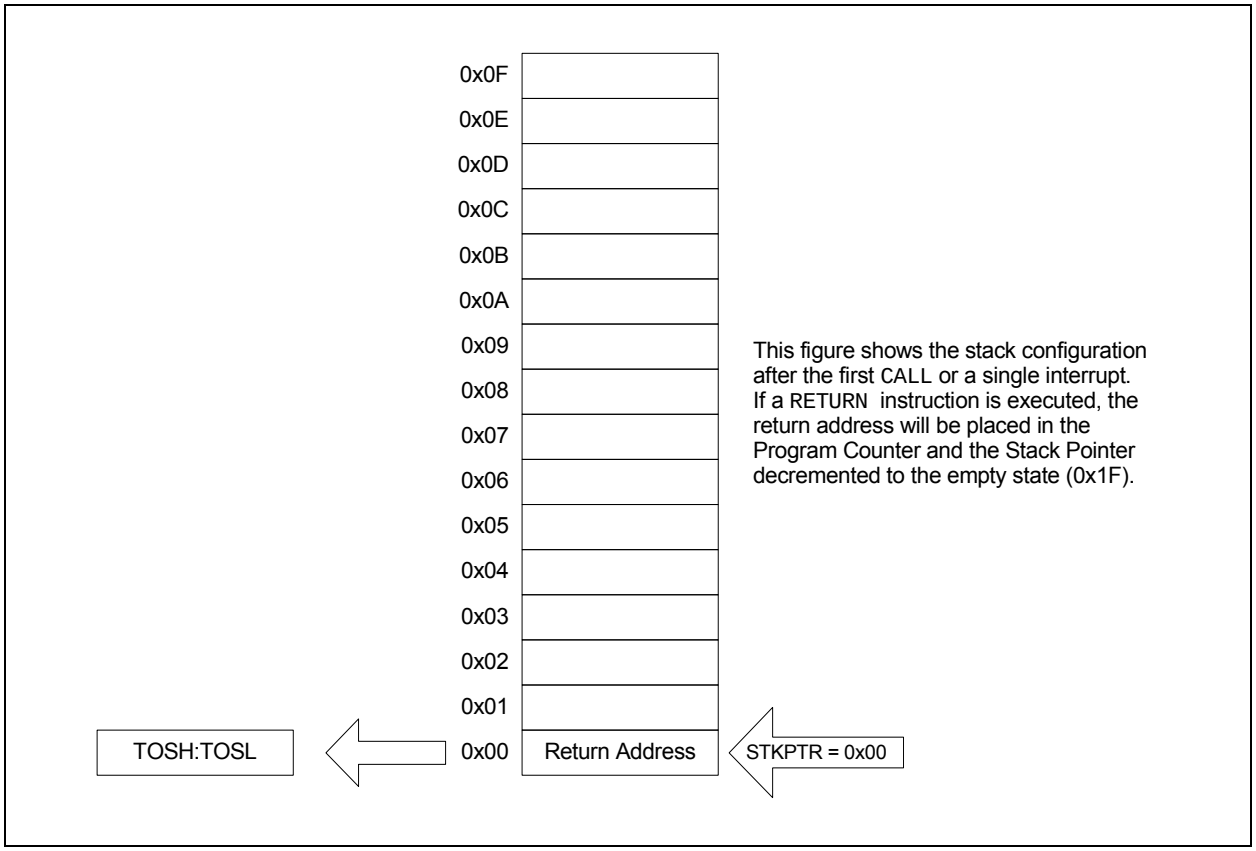


FIGURE 3-6: ACCESSING THE STACK EXAMPLE 3

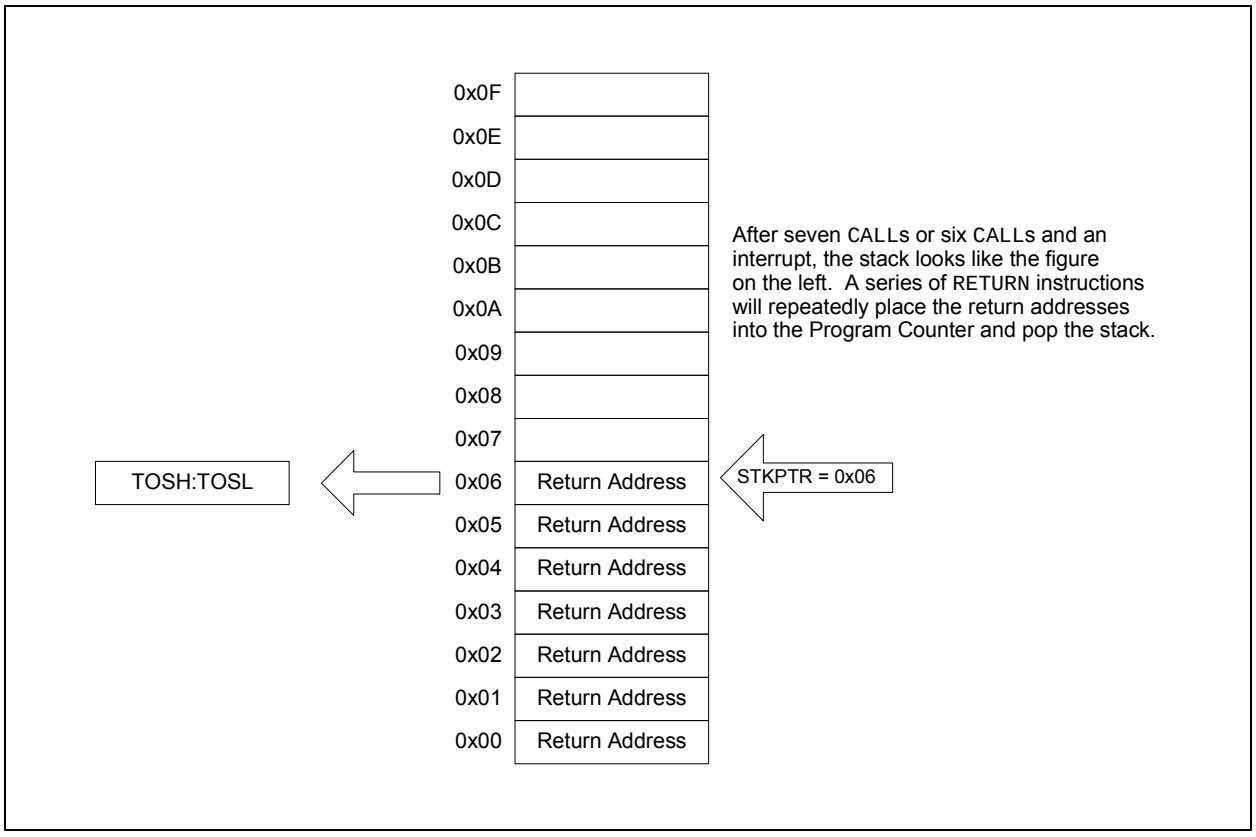
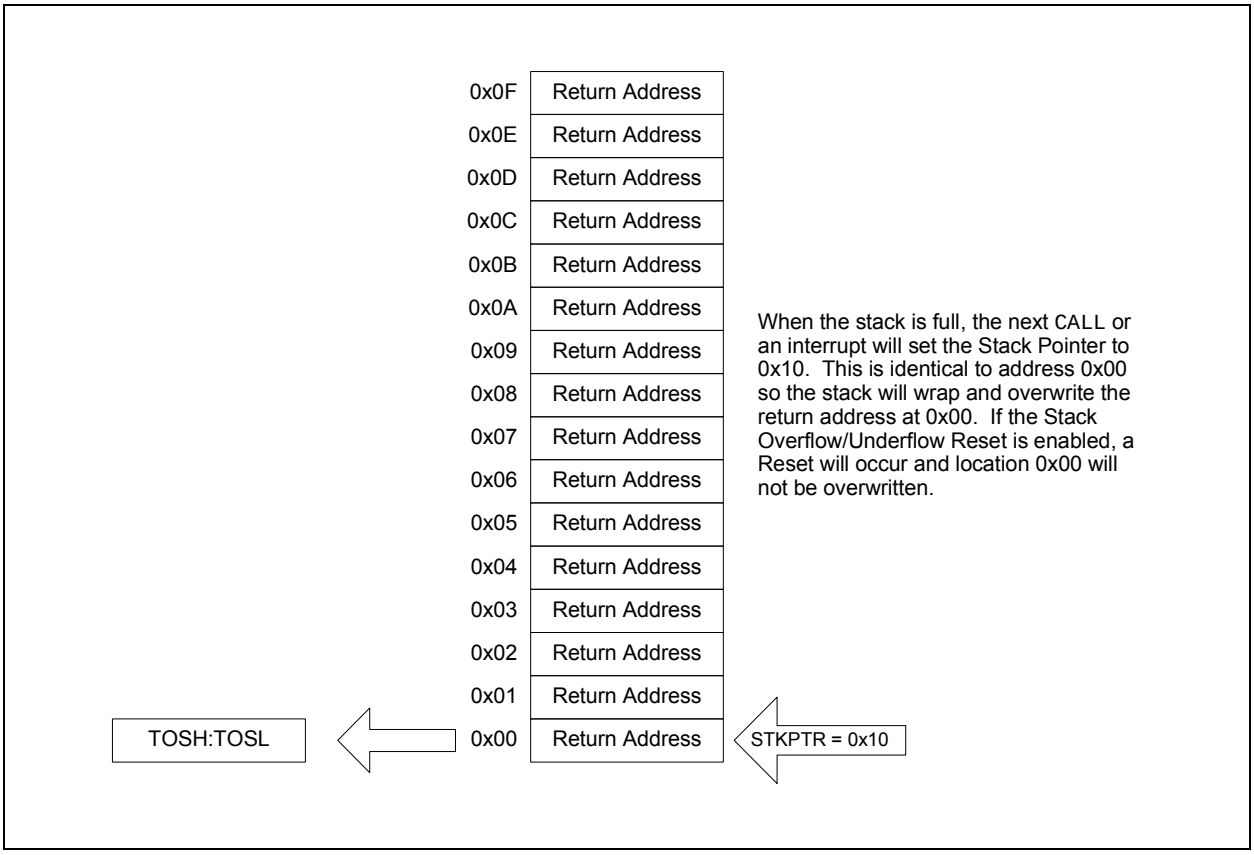


FIGURE 3-7: ACCESSING THE STACK EXAMPLE 4



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.

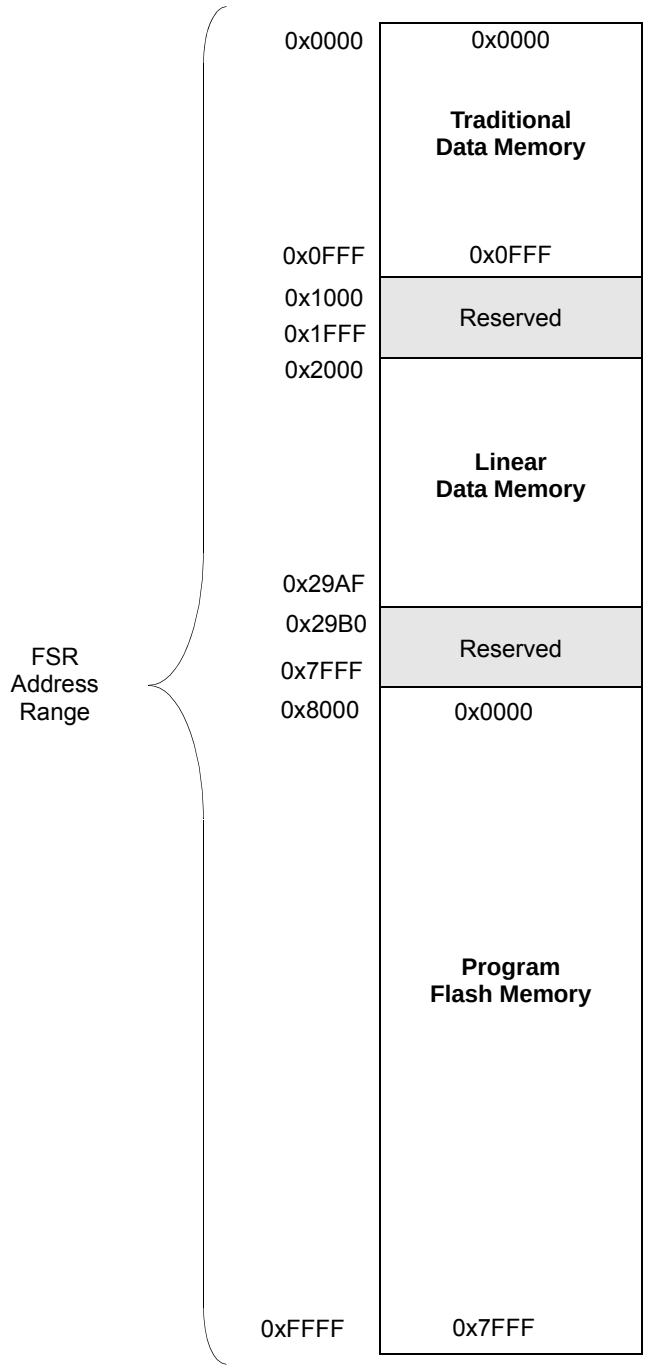
3.6 Indirect Addressing

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

FIGURE 3-8: INDIRECT ADDRESSING

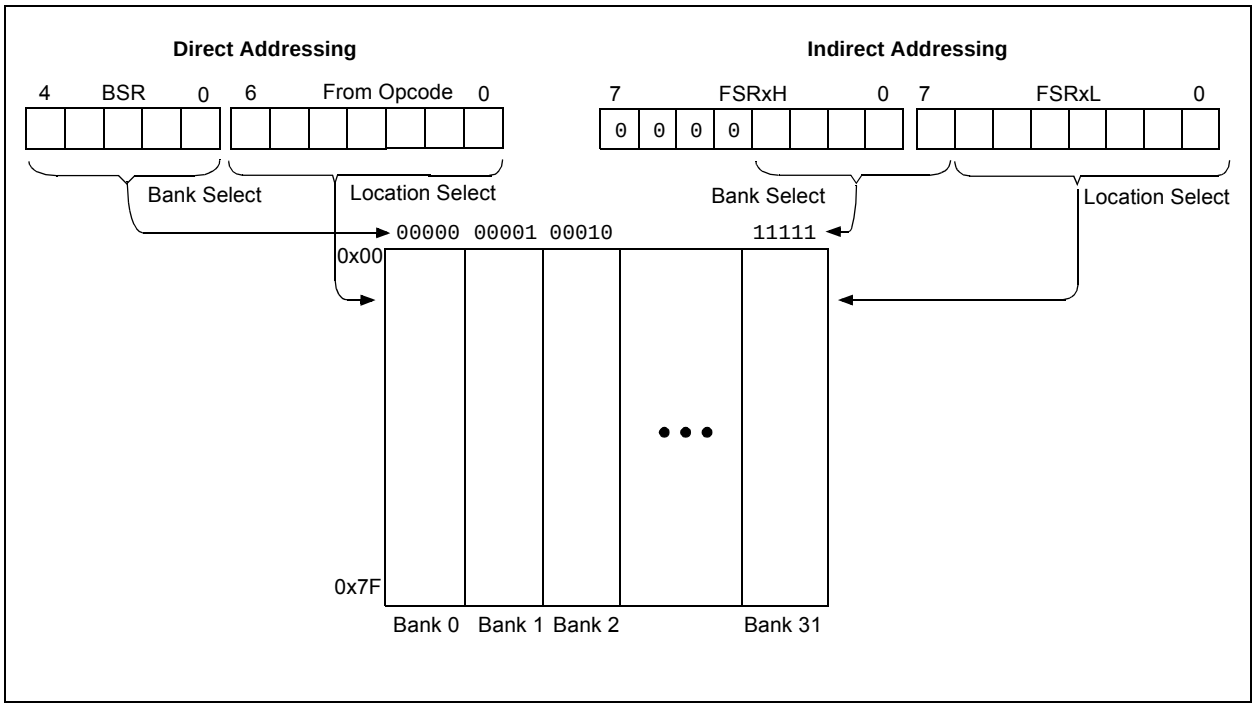


Note: Not all memory regions are completely implemented. Consult device memory tables for memory limits.

3.6.1 TRADITIONAL DATA MEMORY

The traditional data memory is a region from FSR address 0x000 to FSR address 0xFFF. The addresses correspond to the absolute addresses of all SFR, GPR and common registers.

FIGURE 3-9: TRADITIONAL DATA MEMORY MAP



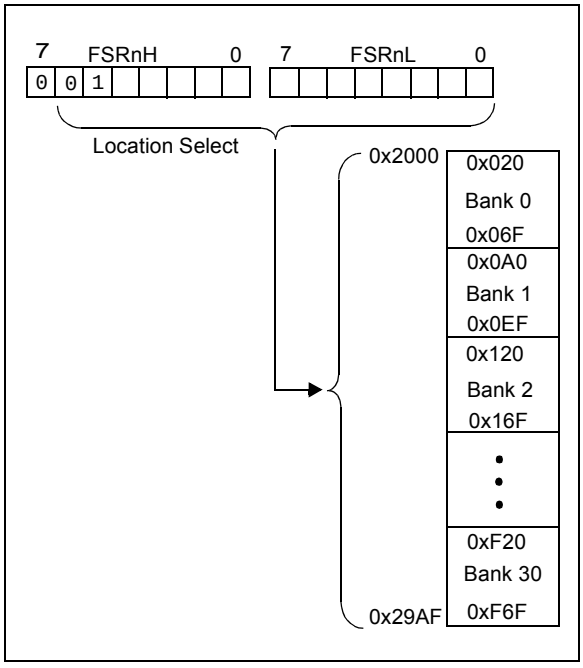
3.6.2 LINEAR DATA MEMORY

The linear data memory is the region from FSR address 0x2000 to FSR address 0x29AF. This region is a virtual region that points back to the 80-byte blocks of GPR memory in all the banks.

Unimplemented memory reads as 0x00. Use of the linear data memory region allows buffers to be larger than 80 bytes because incrementing the FSR beyond one bank will go directly to the GPR memory of the next bank.

The 16 bytes of common memory are not included in the linear data memory region.

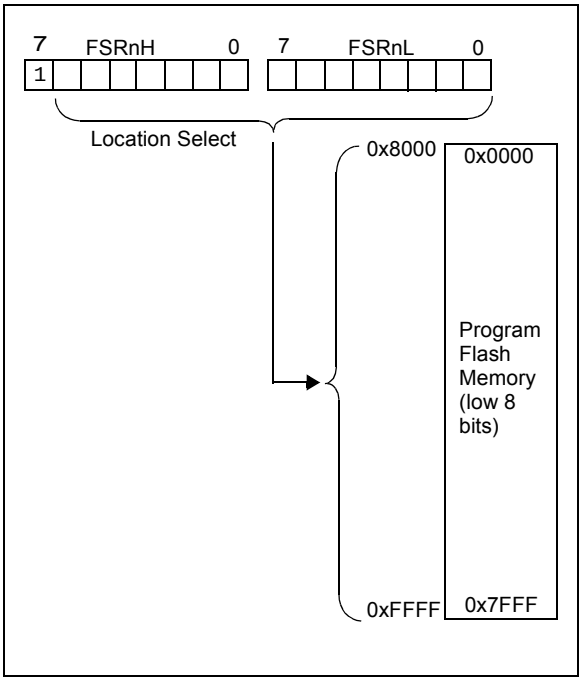
FIGURE 3-10: LINEAR DATA MEMORY MAP



3.6.3 PROGRAM FLASH MEMORY

To make constant data access easier, the entire program Flash memory is mapped to the upper half of the FSR address space. When the MSB of FSRnH is set, the lower 15 bits are the address in program memory which will be accessed through INDF. Only the lower eight bits of each memory location is accessible via INDF. Writing to the program Flash memory cannot be accomplished via the FSR/INDF interface. All instructions that access program Flash memory via the FSR/INDF interface will require one additional instruction cycle to complete.

FIGURE 3-11: PROGRAM FLASH MEMORY MAP



4.0 DEVICE CONFIGURATION

Device configuration consists of Configuration Words, Code Protection and Device ID.

4.1 Configuration Words

There are several Configuration Word bits that allow different oscillator and memory protection options. These are implemented as Configuration Word 1 at 8007h and Configuration Word 2 at 8008h.

4.2 Register Definitions: Configuration Words

REGISTER 4-1: CONFIG1: CONFIGURATION WORD 1

U-1	U-1	R/P-1	R/P-1	R/P-1	U-1
—	—	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		—
bit 13					bit 8

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-1	R/P-1	R/P-1
$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		—	FOSC<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘1’
‘0’ = Bit is cleared	‘1’ = Bit is set	-n = Value when blank or after Bulk Erase

bit 13-12 **Unimplemented:** Read as ‘1’

bit 11 **$\overline{\text{CLKOUTEN}}$:** Clock Out Enable bit
1 = CLKOUT function is disabled. I/O function on the CLKOUT pin
0 = CLKOUT function is enabled on the CLKOUT pin

bit 10-9 **BOREN<1:0>:** Brown-out Reset Enable bits(1)
11 = BOR enabled
10 = BOR enabled during operation and disabled in Sleep
01 = BOR controlled by SBOREN bit of the BORCON register
00 = BOR disabled

bit 8 **Unimplemented:** Read as ‘1’

bit 7 **$\overline{\text{CP}}$:** Code Protection bit(2)
1 = Program memory code protection is disabled
0 = Program memory code protection is enabled

bit 6 **MCLRE:** $\overline{\text{MCLR}}$/VPP Pin Function Select bit
If LVP bit = 1:
This bit is ignored.
If LVP bit = 0:
1 = $\overline{\text{MCLR}}$/VPP pin function is $\overline{\text{MCLR}}$; Weak pull-up enabled.
0 = $\overline{\text{MCLR}}$/VPP pin function is digital input; $\overline{\text{MCLR}}$ internally disabled; Weak pull-up under control of WPUE3 bit.

bit 5 **$\overline{\text{PWRTE}}$:** Power-Up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 4-3 **WDTE<1:0>:** Watchdog Timer Enable bits
11 = WDT enabled
10 = WDT enabled while running and disabled in Sleep
01 = WDT controlled by the SWDTEN bit in the WDTCON register
00 = WDT disabled

bit 2 **Unimplemented:** Read as ‘1’

bit 1-0 **FOSC<1:0>:** Oscillator Selection bits
11 = ECH: External Clock, High-Power mode: on CLKIN pin
10 = ECM: External Clock, Medium-Power mode: on CLKIN pin
01 = ECL: External Clock, Low-Power mode: on CLKIN pin
00 = INTOSC oscillator: I/O function on CLKIN pin

Note 1: Enabling Brown-out Reset does not automatically enable Power-up Timer.
2: Once enabled, code-protect can only be disabled by bulk erasing the device.

REGISTER 4-2: CONFIG2: CONFIGURATION WORD 2

R/P-1	U-1	R/P-1	R/P-1	R/P-1	U-1
LVP	—	$\overline{\text{LPBOR}}$	BORV	STVREN	—
bit 13					bit 8

U-1	U-1	U-1	U-1	U-1	U-1	R/P-1	R/P-1
—	—	—	—	—	—	WRT<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '1'
'0' = Bit is cleared	'1' = Bit is set	-n = Value when blank or after Bulk Erase

bit 13 **LVP:** Low-Voltage Programming Enable bit(1)
1 = Low-voltage programming enabled
0 = High-voltage on MCLR must be used for programming

bit 12 **Unimplemented:** Read as '1'

bit 11 **LPBOR:** Low-Power BOR Enable bit
1 = Low-Power Brown-out Reset is disabled
0 = Low-Power Brown-out Reset is enabled

bit 10 **BORV:** Brown-out Reset Voltage Selection bit(2)
1 = Brown-out Reset voltage (Vbor), low trip point selected
0 = Brown-out Reset voltage (Vbor), high trip point selected

bit 9 **STVREN:** Stack Overflow/Underflow Reset Enable bit
1 = Stack Overflow or Underflow will cause a Reset
0 = Stack Overflow or Underflow will not cause a Reset

bit 8-2 **Unimplemented:** Read as '1'

bit 1-0 **WRT<1:0>:** Flash Memory Self-Write Protection bits
2 kW Flash memory:
11 = Write protection off
10 = 000h to 1FFh write-protected, 200h to 7FFh may be modified
01 = 000h to 3FFh write-protected, 400h to 7FFh may be modified
00 = 000h to 7FFh write-protected, no addresses may be modified

Note 1: The LVP bit cannot be programmed to '0' when Programming mode is entered via LVP.
2: See Vbor parameter for specific trip point voltages.

4.3 Code Protection

Code protection allows the device to be protected from unauthorized access. Internal access to the program memory is unaffected by any code protection setting.

4.3.1 PROGRAM MEMORY PROTECTION

The entire program memory space is protected from external reads and writes by the $\overline{CP}$ bit in Configuration Words. When $\overline{CP}$ = 0, external reads and writes of program memory are inhibited and a read will return all '0's. The CPU can continue to read program memory, regardless of the protection bit settings. Writing the program memory is dependent upon the write protection setting. See **Section 4.4 "Write Protection"** for more information.

4.4 Write Protection

Write protection allows the device to be protected from unintended self-writes. Applications, such as bootloader software, can be protected while allowing other regions of the program memory to be modified.

The WRT<1:0> bits in Configuration Words define the size of the program memory block that is protected.

4.5 User ID

Four memory locations (8000h-8003h) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See **Section 10.4 "User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations. For more information on checksum calculation, see the "*PIC12LF1552 Memory Programming Specification*" (DS41642).

4.6 Device ID and Revision ID

The memory location 8006h is where the Device ID and Revision ID are stored. The upper nine bits hold the Device ID. The lower five bits hold the Revision ID. See **Section 10.4 "User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations.

Development tools, such as device programmers and debuggers, may be used to read the Device ID and Revision ID.

4.7 Register Definitions: Device

REGISTER 4-3: DEVID: DEVICE ID REGISTER

R	R	R	R	R	R
DEV<8:3>					
bit 13					bit 8

R	R	R	R	R	R	R	R
DEV<2:0>			REV<4:0>				
bit 7							bit 0

Legend:	
R = Readable bit	
'1' = Bit is set	'0' = Bit is cleared

bit 13-5 **DEV<8:0>:** Device ID bits

Device	DEVID<13:0> Values	
	DEV<8:0>	REV<4:0>
PIC12LF1552	0010 1011 110	x xxxx

bit 4-0 **REV<4:0>:** Revision ID bits
These bits are used to identify the revision (see Table under DEV<8:0> above).

5.0 OSCILLATOR MODULE

5.1 Overview

The oscillator module has a wide variety of clock sources and selection features that allow it to be used in a wide range of applications while maximizing performance and minimizing power consumption. Figure 5-1 illustrates a block diagram of the oscillator module.

Clock sources can be supplied from external clock oscillators. In addition, the system clock source can be supplied from one of two internal oscillators and PLL circuits, with a choice of speeds selectable via software. Additional clock features include:

- Selectable system clock source between external or internal sources via software.

The oscillator module can be configured in one of the following clock modes.

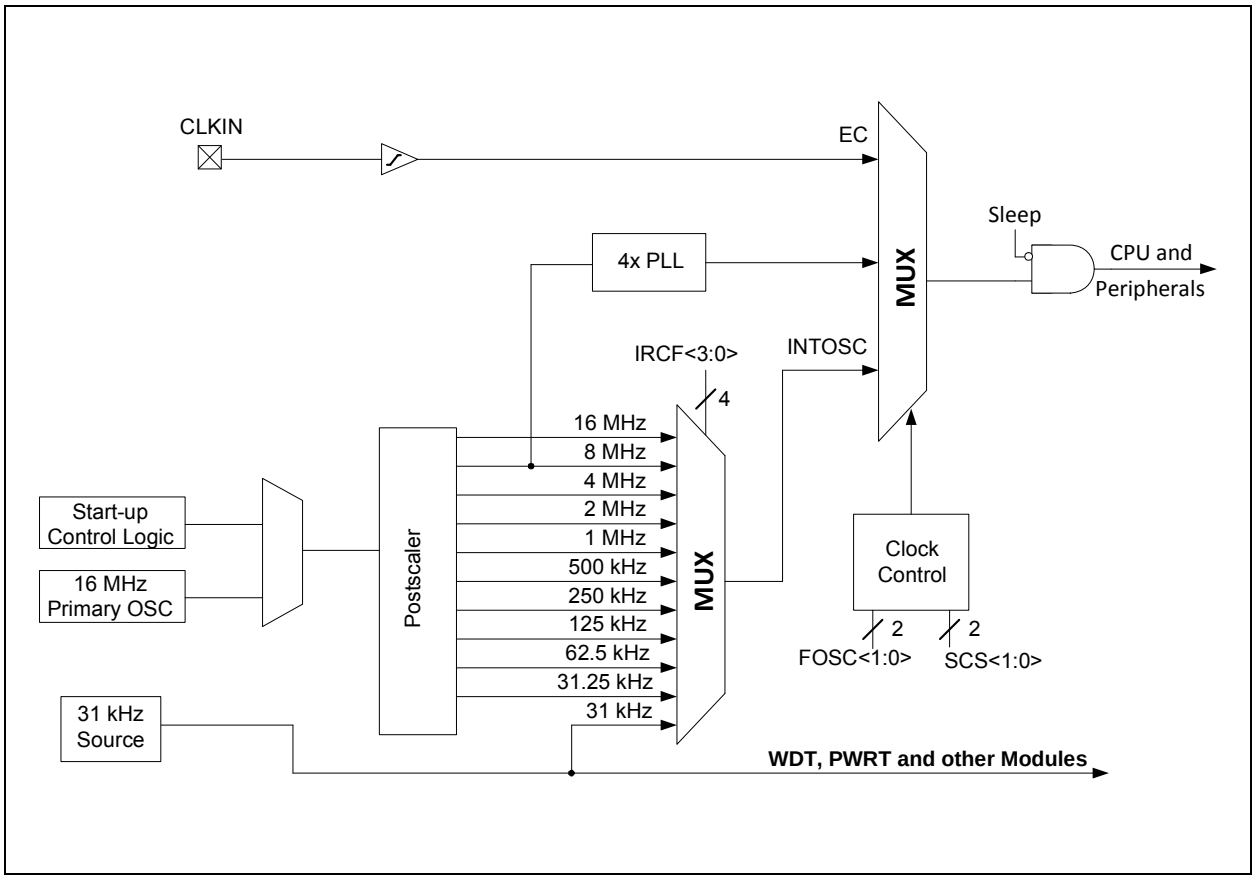
1. ECL – External Clock Low-Power mode (0 MHz to 0.5 MHz)
2. ECM – External Clock Medium-Power mode (0.5 MHz to 4 MHz)
3. ECH – External Clock High-Power mode (4 MHz to 20 MHz)
4. INTOSC – Internal oscillator (31 kHz to 32 MHz)

Clock Source modes are selected by the FOSC<1:0> bits in the Configuration Words. The FOSC bits determine the type of oscillator that will be used when the device is first powered.

The EC clock mode relies on an external logic level signal as the device clock source.

The INTOSC internal oscillator block produces low and high-frequency clock sources, designated LFINTOSC and HFINTOSC. (see Internal Oscillator Block, Figure 5-1). A wide selection of device clock frequencies may be derived from these clock sources.

FIGURE 5-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM



5.2 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (EC mode).

Internal clock sources are contained within the oscillator module. The oscillator block has two internal oscillators that are used to generate two system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC) and the 31 kHz Low-Frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS) bits in the OSCCON register. See **Section 5.3 "Clock Switching"** for additional information.

5.2.1 EXTERNAL CLOCK SOURCES

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

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

See **Section 5.3 "Clock Switching"** for more information.

5.2.1.1 EC Mode

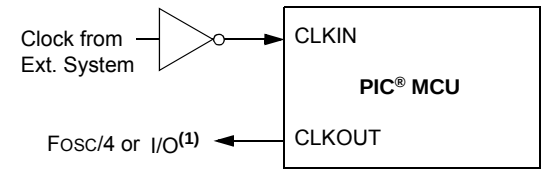
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the CLKIN input. CLKOUT is available for general purpose I/O or CLKOUT. Figure 5-2 shows the pin connections for EC mode.

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

- High power, 4-20 MHz (FOSC = 11)
- Medium power, 0.5-4 MHz (FOSC = 10)
- Low power, 0-0.5 MHz (FOSC = 01)

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

FIGURE 5-2: EXTERNAL CLOCK (EC) MODE OPERATION



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

5.2.2 INTERNAL CLOCK SOURCES

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

- Program the FOSC<1:0> bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Set the SCS<1:0> bits in the OSCCON register to '1x' to switch the system clock source to the internal oscillator during run-time. See **Section 5.3 "Clock Switching"** for more information.

In INTOSC mode, the CLKIN pin is available for general purpose I/O. The CLKOUT pin is available for general purpose I/O or CLKOUT.

The function of the CLKOUT pin is determined by the CLKOUTEN bit in Configuration Words.

The internal oscillator block has two independent oscillators.

1. The HFINTOSC (High-Frequency Internal Oscillator) is factory calibrated and operates at 16 MHz.
2. The LFINTOSC (Low-Frequency Internal Oscillator) is uncalibrated and operates at 31 kHz.

5.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a factory calibrated 16 MHz internal clock source.

The outputs of the HFINTOSC connects to a prescaler and multiplexer (see Figure 5-1). One of multiple frequencies derived from the HFINTOSC can be selected via software using the IRCF<3:0> bits of the OSCCON register. See **Section 5.2.2.4 "Internal Oscillator Clock Switch Timing"** for more information.

The HFINTOSC is enabled by:

- Configure the IRCF<3:0> bits of the OSCCON register for the desired HF frequency, and
- FOSC<1:0> = 00, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '1x'.

A fast start-up oscillator allows internal circuits to power-up and stabilize before switching to HFINTOSC.

The High-Frequency Internal Oscillator Ready bit (HFIOFR) of the OSCSTAT register indicates when the HFINTOSC is running.

The High-Frequency Internal Oscillator Stable bit (HFIOFS) of the OSCSTAT register indicates when the HFINTOSC is running within 0.5% of its final value.

5.2.2.2 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is an uncalibrated 31 kHz internal clock source.

The output of the LFINTOSC connects to a multiplexer (see Figure 5-1). Select 31 kHz, via software, using the IRCF<3:0> bits of the OSCCON register. See **Section 5.2.2.4 "Internal Oscillator Clock Switch Timing"** for more information. The LFINTOSC is also the source for the Power-up Timer (PWRT) and Watchdog Timer (WDT).

The LFINTOSC is enabled by selecting 31 kHz (IRCF<3:0> bits of the OSCCON register = 000x) as the system clock source (SCS bits of the OSCCON register = 1x), or when any of the following are enabled:

- Configure the IRCF<3:0> bits of the OSCCON register for the LF frequency, and
- FOSC<1:0> = 00, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '1x'

Peripherals that use the LFINTOSC are:

- Power-up Timer (PWRT)
- Watchdog Timer (WDT)

The Low-Frequency Internal Oscillator Ready bit (LFIOFR) of the OSCSTAT register indicates when the LFINTOSC is running.

5.2.2.3 Internal Oscillator Frequency Selection

The system clock speed can be selected via software using the Internal Oscillator Frequency Select bits IRCF<3:0> of the OSCCON register.

The outputs of the 16 MHz HFINTOSC postscaler and the LFINTOSC connect to a multiplexer (see Figure 5-1). The Internal Oscillator Frequency Select bits IRCF<3:0> of the OSCCON register select the frequency. One of the following frequencies can be selected via software:

- 32 MHz (requires 4x PLL)
- 16 MHz
- 8 MHz
- 4 MHz
- 2 MHz
- 1 MHz
- 500 kHz (default after Reset)
- 250 kHz
- 125 kHz
- 62.5 kHz
- 31.25 kHz
- 31 kHz (LFINTOSC)

Note:	Following any Reset, the IRCF<3:0> bits of the OSCCON register are set to '0111' and the frequency selection is set to 500 kHz. The user can modify the IRCF bits to select a different frequency.

The IRCF<3:0> bits of the OSCCON register allow duplicate selections for some frequencies. These duplicate choices can offer system design trade-offs. Lower power consumption can be obtained when changing oscillator sources for a given frequency. Faster transition times can be obtained between frequency changes that use the same oscillator source.

5.2.2.4 Internal Oscillator Clock Switch Timing

When switching between the HFINTOSC and the LFINTOSC, the new oscillator may already be shut down to save power (see Figure 5-3). If this is the case, there is a delay after the IRCF<3:0> bits of the OSCCON register are modified before the frequency selection takes place. The OSCSTAT register will reflect the current active status of the HFINTOSC and LFINTOSC oscillators. The sequence of a frequency selection is as follows:

1. IRCF<3:0> bits of the OSCCON register are modified.
2. If the new clock is shut down, a clock start-up delay is started.
3. Clock switch circuitry waits for a falling edge of the current clock.
4. Clock switch is complete.

See Figure 5-3 for more details.

If the internal oscillator speed is switched between two clocks of the same source, there is no start-up delay before the new frequency is selected.

Start-up delay specifications are located in the oscillator tables of **Section 21.0 "Electrical Specifications"**.

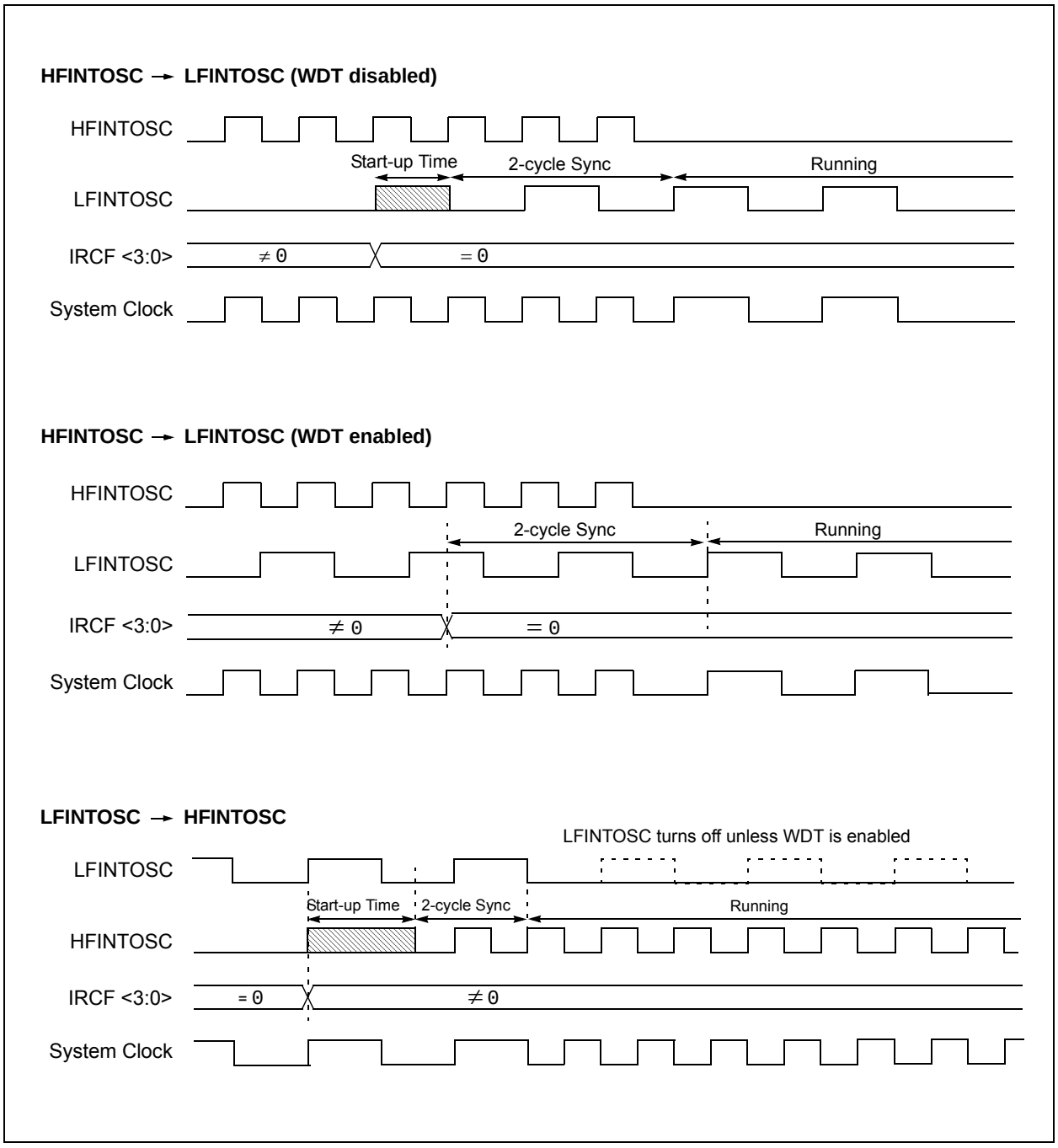
5.2.2.5 32 MHz Internal Oscillator Frequency Selection

The Internal Oscillator Block can be used with the 4x PLL to produce a 32 MHz internal system clock source. The following settings are required to use the 32 MHz internal clock source:

- The FOSC bits in Configuration Word 1 must be set to use the INTOSC source as the device system clock (FOSC<1:0> = 00).
- The SCS bits in the OSCCON register must be cleared to use the clock determined by FOSC<1:0> in Configuration Word 1 (SCS<1:0> = 00).
- The IRCF bits in the OSCCON register must be set to the 8 MHz HFINTOSC set to use (IRCF<3:0> = 1110).
- The SPLLEN bit in the OSCCON register must be set to enable the 4x PLL.

The 4x PLL is not available for use with the internal oscillator when the SCS bits of the OSCCON register are set to '1x'. The SCS bits must be set to '00' to use the 4x PLL with the internal oscillator.

FIGURE 5-3: INTERNAL OSCILLATOR SWITCH TIMING



5.3 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the System Clock Select (SCS) bits of the OSCCON register. The following clock sources can be selected using the SCS bits:

- Default system oscillator determined by FOSC bits in Configuration Words
- Internal Oscillator Block (INTOSC)

5.3.1 SYSTEM CLOCK SELECT (SCS) BITS

The System Clock Select (SCS) bits of the OSCCON register selects the system clock source that is used for the CPU and peripherals.

- When the SCS bits of the OSCCON register = 00, the system clock source is determined by value of the FOSC<1:0> bits in the Configuration Words.
- When the SCS bits of the OSCCON register = 1x, the system clock source is chosen by the internal oscillator frequency selected by the IRCF<3:0> bits of the OSCCON register. After a Reset, the SCS bits of the OSCCON register are always cleared.

When switching between clock sources, a delay is required to allow the new clock to stabilize. These oscillator delays are shown in Table 5-2.

5.3.2 CLOCK SWITCHING BEFORE SLEEP

When clock switching from an old clock to a new clock is requested just prior to entering Sleep mode, it is necessary to confirm that the switch is complete before the SLEEP instruction is executed. Failure to do so may result in an incomplete switch and consequential loss of the system clock altogether. Clock switching is confirmed by monitoring the Clock Status bits in the OSCSTAT register. Switch confirmation can be accomplished by sensing that the Ready bit for the new clock is set or the Ready bit for the old clock is cleared. For example, when switching between the internal oscillator with the PLL and the internal oscillator without the PLL, monitor the PLLR bit. When PLLR is set, the switch to 32 MHz operation is complete. Conversely, when PLLR is cleared, the switch from 32 MHz operation to the selected internal clock is complete.

TABLE 5-1: OSCILLATOR SWITCHING DELAYS

Switch From	Switch To	Frequency	Oscillator Delay
Sleep	LFINTOSC(1) MFINTOSC(1) HFINTOSC(1)	31 kHz 31.25 kHz-500 kHz 31.25 kHz-16 MHz	Oscillator Warm-Up Delay TWARM(2)
Sleep/POR	EC(1)	DC – 32 MHz	2 cycles
LFINTOSC	EC(1)	DC – 32 MHz	1 cycle of each
Any clock source	MFINTOSC(1) HFINTOSC(1)	31.25 kHz-500 kHz 31.25 kHz-16 MHz	2 μs (approx.)
Any clock source	LFINTOSC(1)	31 kHz	1 cycle of each
PLL inactive	PLL active	16-32 MHz	2 ms (approx.)

Note 1: PLL inactive.
2: See **Section 21.0 "Electrical Specifications"**.

5.4 Register Definitions: Oscillator Control

REGISTER 5-1: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-1/1	R/W-1/1	R/W-1/1	U-0	R/W-0/0	R/W-0/0
SPLLEN	IRCF<3:0>				—	SCS<1:0>	
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 **SPLLEN:** Software PLL Enable bit
1 = 4x PLL Is enabled
0 = 4x PLL is disabled

bit 6-3 **IRCF<3:0>:** Internal Oscillator Frequency Select bits
1111 = 16 MHz
1110 = 8 MHz
1101 = 4 MHz
1100 = 2 MHz
1011 = 1 MHz
1010 = 500 kHz(1)
1001 = 250 kHz(1)
1000 = 125 kHz(1)
0111 = 500 kHz (default upon Reset)
0110 = 250 kHz
0101 = 125 kHz
0100 = 62.5 kHz
001x = 31.25 kHz
000x = 31 kHz (LFINTOSC)

bit 2 **Unimplemented:** Read as ‘0’

bit 1-0 **SCS<1:0>:** System Clock Select bits
1x = Internal oscillator block
01 = Reserved
00 = Clock determined by FOSC<1:0> in Configuration Words

Note 1: Duplicate frequency derived from HFINTOSC.

REGISTER 5-2: OSCSTAT: OSCILLATOR STATUS REGISTER

U-0	R-0/q	U-0	R-0/q	U-0	U-0	R-0/q	R-0/q
—	PLLR	—	HFIOFR	—	—	LFIOFR	HFIOFS
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	q = Conditional

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **PLLR** 4x PLL Ready bit
1 = 4x PLL is ready
0 = 4x PLL is not ready

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **HFIOFR:** High-Frequency Internal Oscillator Ready bit
1 = 16 MHz Internal Oscillator (HFINTOSC) is ready
0 = 16 MHz Internal Oscillator (HFINTOSC) is not ready

bit 3-2 **Unimplemented:** Read as ‘0’

bit 1 **LFIOFR:** Low-Frequency Internal Oscillator Ready bit
1 = 31 kHz Internal Oscillator (LFINTOSC) is ready
0 = 31 kHz Internal Oscillator (LFINTOSC) is not ready

bit 0 **HFIOFS:** High-Frequency Internal Oscillator Stable bit
1 = 16 MHz Internal Oscillator (HFINTOSC) is stable
0 = 16 MHz Internal Oscillator (HFINTOSC) is not yet stable

TABLE 5-2: 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>		42
OSCSTAT	—	PLLR	—	HFIOFR	—	—	LFIOFR	HFIOFS	43

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

TABLE 5-3: 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	—	—	—	—	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		—	32
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		—	FOSC<1:0>		

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

6.0 RESETS

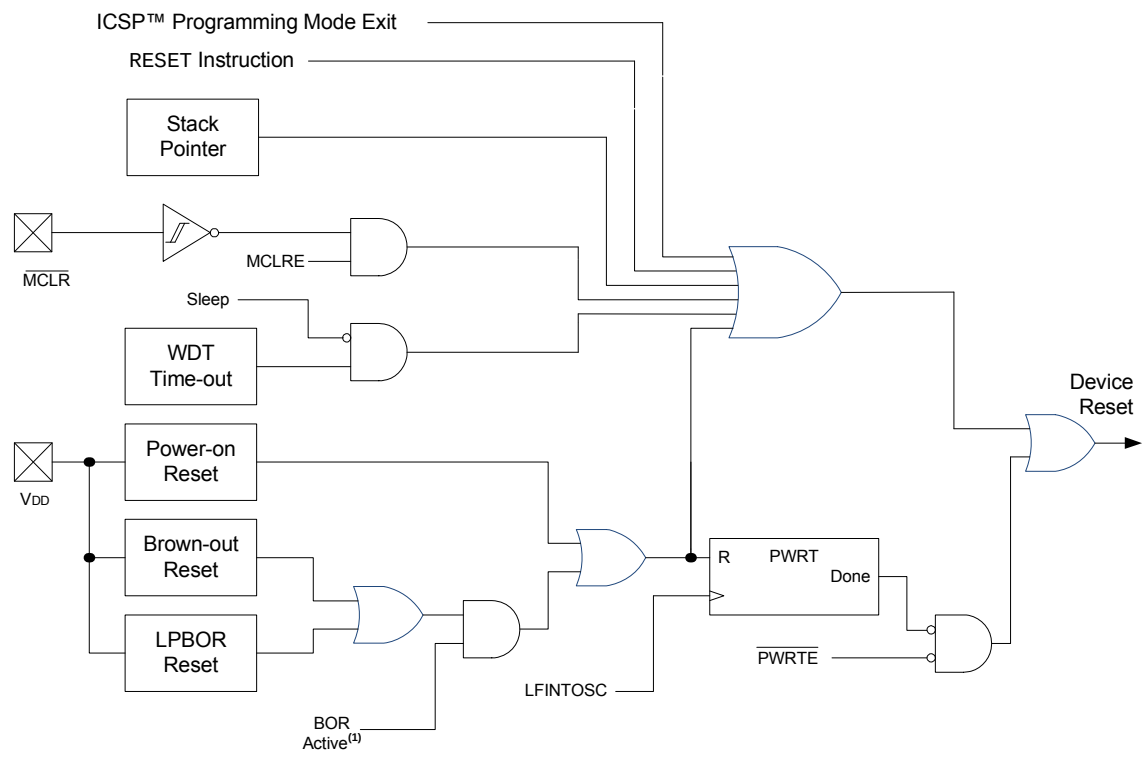
There are multiple ways to reset this device:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Low-Power Brown-out Reset (LPBOR)
- $\overline{\text{MCLR}}$ Reset
- WDT Reset
- RESET instruction
- Stack Overflow
- Stack Underflow
- Programming mode exit

To allow V_{DD} to stabilize, an optional Power-up Timer can be enabled to extend the Reset time after a BOR or POR event.

A simplified block diagram of the On-chip Reset Circuit is shown in Figure 6-1.

FIGURE 6-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: See Table 6-1 for BOR active conditions.

6.1 Power-on Reset (POR)

The POR circuit holds the device in Reset until VDD has reached an acceptable level for minimum operation. Slow rising VDD, fast operating speeds or analog performance may require greater than minimum VDD. The PWRT, BOR or MCLR features can be used to extend the start-up period until all device operation conditions have been met.

6.1.1 POWER-UP TIMER (PWRT)

The Power-up Timer provides a nominal 64 ms time-out on POR or Brown-out Reset.

The device is held in Reset as long as PWRT is active. The PWRT delay allows additional time for the VDD to rise to an acceptable level. The Power-up Timer is enabled by clearing the PWRTE bit in Configuration Words.

The Power-up Timer starts after the release of the POR and BOR.

For additional information, refer to Application Note AN607, *"Power-up Trouble Shooting"* (DS00607).

6.2 Brown-out Reset (BOR)

The BOR circuit holds the device in Reset when VDD reaches a selectable minimum level. Between the POR and BOR, complete voltage range coverage for execution protection can be implemented.

The Brown-out Reset module has four operating modes controlled by the BOREN<1:0> bits in Configuration Words. The four operating modes are:

- BOR is always on
- BOR is off when in Sleep
- BOR is controlled by software
- BOR is always off

Refer to Table 6-1 for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV bit in Configuration Words.

A VDD noise rejection filter prevents the BOR from triggering on small events. If VDD falls below VBOR for a duration greater than parameter TBORDC, the device will reset. See Figure 6-2 for more information.

TABLE 6-1: BOR OPERATING MODES

BOREN<1:0>	SBOREN	Device Mode	BOR Mode	Instruction Execution upon: Release of POR or Wake-up from Sleep
11	X	X	Active	Waits for BOR ready(1) (BORRDY = 1)
10	X	Awake	Active	Waits for BOR ready (BORRDY = 1)
		Sleep	Disabled	
01	1	X	Active	Waits for BOR ready(1) (BORRDY = 1)
	0	X	Disabled	Begins immediately (BORRDY = x)
00	X	X	Disabled	

Note 1: In these specific cases, "release of POR" and "wake-up from Sleep," there is no delay in start-up. The BOR ready flag, (BORRDY = 1), will be set before the CPU is ready to execute instructions because the BOR circuit is forced on by the BOREN<1:0> bits.

6.2.1 BOR IS ALWAYS ON

When the BOREN bits of Configuration Words are programmed to '11', the BOR is always on. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

BOR protection is active during Sleep. The BOR does not delay wake-up from Sleep.

6.2.2 BOR IS OFF IN SLEEP

When the BOREN bits of Configuration Words are programmed to '10', the BOR is on, except in Sleep. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

BOR protection is not active during Sleep. The device wake-up will be delayed until the BOR is ready.

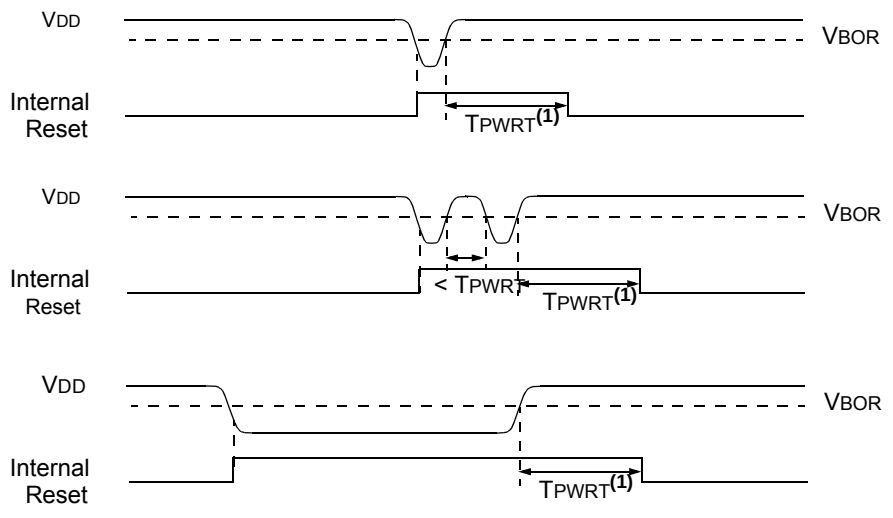
6.2.3 BOR CONTROLLED BY SOFTWARE

When the BOREN bits of Configuration Words are programmed to '01', the BOR is controlled by the SBOREN bit of the BORCON register. The device start-up is not delayed by the BOR ready condition or the VDD level.

BOR protection begins as soon as the BOR circuit is ready. The status of the BOR circuit is reflected in the BORRDY bit of the BORCON register.

BOR protection is unchanged by Sleep.

FIGURE 6-2: BROWN-OUT SITUATIONS



Note 1: TPWRT delay only if $\overline{\text{PWRTE}}$ bit is programmed to '0'.

6.3 Register Definitions: BOR Control

REGISTER 6-1: BORCON: BROWN-OUT RESET CONTROL REGISTER

R/W-1/u	R/W-0/u	U-0	U-0	U-0	U-0	U-0	R-q/u
SBOREN	BORFS	—	—	—	—	—	BORRDY
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	q = Value depends on condition

bit 7 **SBOREN:** Software Brown-out Reset Enable bit
If BOREN <1:0> in Configuration Words = 01:
1 = BOR Enabled
0 = BOR Disabled
If BOREN <1:0> in Configuration Words ≠ 00:
SBOREN is read/write, but has no effect on the BOR.

bit 6 **BORFS:** Brown-out Reset Fast Start bit(1)
If BOREN <1:0> = 10 (Disabled in Sleep) or BOREN<1:0> = 01 (Under software control):
1 = Band gap is forced on always (covers sleep/wake-up/operating cases)
0 = Band gap operates normally, and may turn off
If BOREN<1:0> = 11 (Always on) or BOREN<1:0> = 00 (Always off)
BORFS is Read/Write, but has no effect.

bit 5-1 **Unimplemented:** Read as '0'

bit 0 **BORRDY:** Brown-out Reset Circuit Ready Status bit
1 = The Brown-out Reset circuit is active
0 = The Brown-out Reset circuit is inactive

Note 1: BOREN<1:0> bits are located in Configuration Words.

6.4 Low-Power Brown-out Reset (LPBOR)

The Low-Power Brown-Out Reset (LPBOR) is an essential part of the Reset subsystem. Refer to Figure 6-1 to see how the BOR interacts with other modules.

The LPBOR is used to monitor the external VDD pin. When too low of a voltage is detected, the device is held in Reset. When this occurs, a register bit ($\overline{BOR}$) is changed to indicate that a BOR Reset has occurred. The same bit is set for both the BOR and the LPBOR. Refer to Register 6-2.

6.4.1 ENABLING LPBOR

The LPBOR is controlled by the $\overline{LPBOR}$ bit of Configuration Words. When the device is erased, the LPBOR module defaults to disabled.

6.4.1.1 LPBOR Module Output

The output of the LPBOR module is a signal indicating whether or not a Reset is to be asserted. This signal is OR'd together with the Reset signal of the BOR module to provide the generic $\overline{BOR}$ signal which goes to the PCON register and to the power control block.

6.5 $\overline{MCLR}$

The $\overline{MCLR}$ is an optional external input that can reset the device. The $\overline{MCLR}$ function is controlled by the MCLRE bit of Configuration Words and the LVP bit of Configuration Words (Table 6-2).

TABLE 6-2: $\overline{MCLR}$ CONFIGURATION

MCLRE	LVP	$\overline{MCLR}$
0	0	Disabled
1	0	Enabled
x	1	Enabled

6.5.1 $\overline{MCLR}$ ENABLED

When $\overline{MCLR}$ is enabled and the pin is held low, the device is held in Reset. The $\overline{MCLR}$ pin is connected to VDD through an internal weak pull-up.

The device has a noise filter in the $\overline{MCLR}$ Reset path. The filter will detect and ignore small pulses.

Note:	A Reset does not drive the $\overline{MCLR}$ pin low.

6.5.2 $\overline{MCLR}$ DISABLED

When $\overline{MCLR}$ is disabled, the pin functions as a general purpose input and the internal weak pull-up is under software control. See **Section 11.3 "PORTA Registers"** for more information.

6.6 Watchdog Timer (WDT) Reset

The Watchdog Timer generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register are changed to indicate the WDT Reset. See **Section 9.0 "Watchdog Timer (WDT)"** for more information.

6.7 RESET Instruction

A RESET instruction will cause a device Reset. The $\overline{RI}$ bit in the PCON register will be set to '0'. See Table 6-4 for default conditions after a RESET instruction has occurred.

6.8 Stack Overflow/Underflow Reset

The device can reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits of the PCON register indicate the Reset condition. These Resets are enabled by setting the STVREN bit in Configuration Words. See **Section 3.5.2 "Overflow/Underflow Reset"** for more information.

6.9 Programming Mode Exit

Upon exit of Programming mode, the device will behave as if a POR had just occurred.

6.10 Power-up Timer

The Power-up Timer optionally delays device execution after a BOR or POR event. This timer is typically used to allow VDD to stabilize before allowing the device to start running.

The Power-up Timer is controlled by the $\overline{PWRTE}$ bit of Configuration Words.

6.11 Start-up Sequence

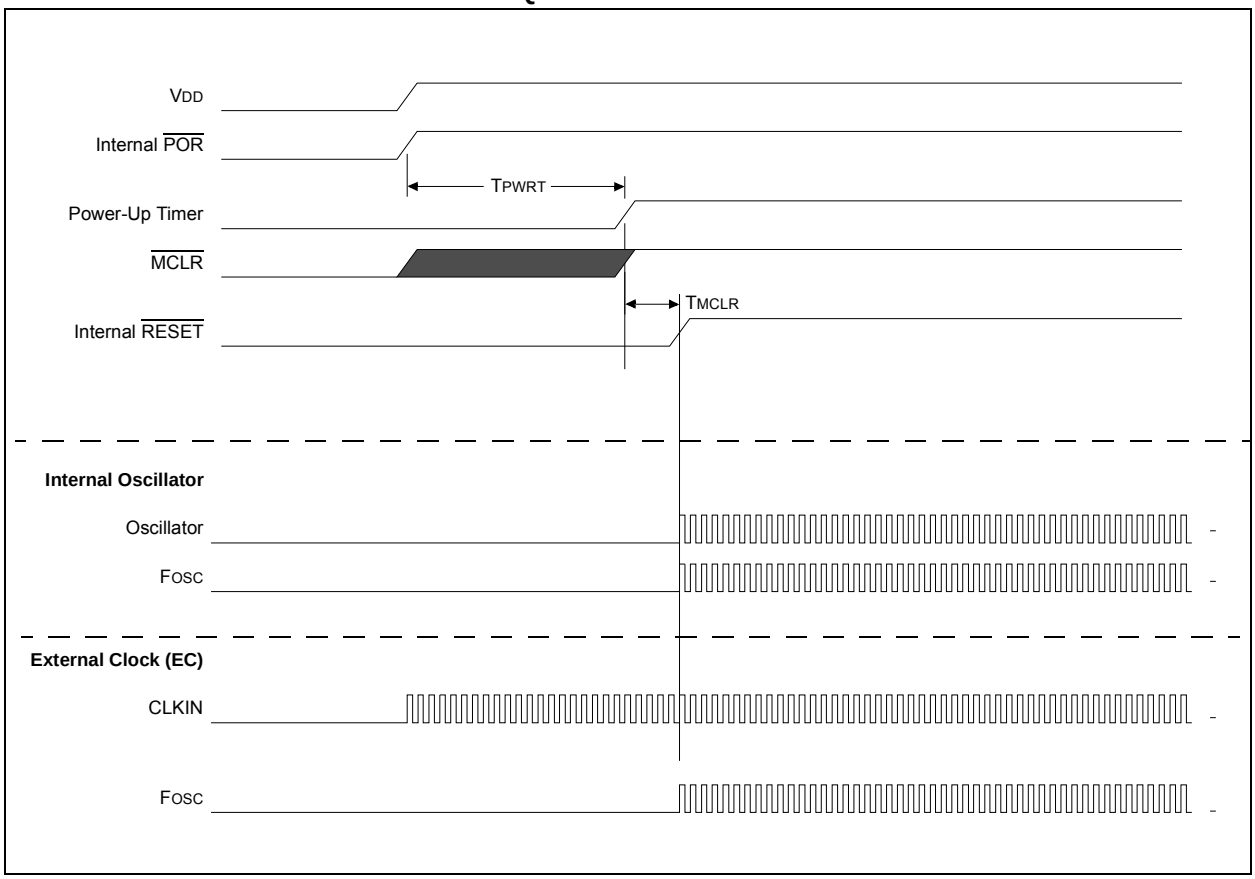
Upon the release of a POR or BOR, the following must occur before the device will begin executing:

1. Power-up Timer runs to completion (if enabled).
2. $\overline{MCLR}$ must be released (if enabled).

The total time-out will vary based on oscillator configuration and Power-up Timer configuration. See **Section 5.0 "Oscillator Module"** for more information.

The Power-up Timer runs independently of $\overline{MCLR}$ Reset. If $\overline{MCLR}$ is kept low long enough, the Power-up Timer will expire. Upon bringing $\overline{MCLR}$ high, the device will begin execution immediately (see Figure 6-3). This is useful for testing purposes or to synchronize more than one device operating in parallel.

FIGURE 6-3: RESET START-UP SEQUENCE



6.12 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS and PCON registers are updated to indicate the cause of the Reset. Table 6-3 and Table 6-4 show the Reset conditions of these registers.

TABLE 6-3: RESET STATUS BITS AND THEIR SIGNIFICANCE

STKOVF	STKUNF	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Condition
0	0	1	1	1	0	x	1	1	Power-on Reset
0	0	1	1	1	0	x	0	x	Illegal, $\overline{\text{TO}}$ is set on $\overline{\text{POR}}$
0	0	1	1	1	0	x	x	0	Illegal, $\overline{\text{PD}}$ is set on $\overline{\text{POR}}$
0	0	u	1	1	u	0	1	1	Brown-out Reset
u	u	0	u	u	u	u	0	u	WDT Reset
u	u	u	u	u	u	u	0	0	WDT Wake-up from Sleep
u	u	u	u	u	u	u	1	0	Interrupt Wake-up from Sleep
u	u	u	0	u	u	u	u	u	$\overline{\text{MCLR}}$ Reset during normal operation
u	u	u	0	u	u	u	1	0	$\overline{\text{MCLR}}$ Reset during Sleep
u	u	u	u	0	u	u	u	u	RESET Instruction Executed
1	u	u	u	u	u	u	u	u	Stack Overflow Reset (STVREN = 1)
u	1	u	u	u	u	u	u	u	Stack Underflow Reset (STVREN = 1)

TABLE 6-4: RESET CONDITION FOR SPECIAL REGISTERS

Condition	Program Counter	STATUS Register	PCON Register
Power-on Reset	0000h	---1 1000	00-- 110x
$\overline{\text{MCLR}}$ Reset during normal operation	0000h	---u uuuu	uu-- 0uuu
$\overline{\text{MCLR}}$ Reset during Sleep	0000h	---1 0uuu	uu-- 0uuu
WDT Reset	0000h	---0 uuuu	uu-- uuuu
WDT Wake-up from Sleep	PC + 1	---0 0uuu	uu-- uuuu
Brown-out Reset	0000h	---1 1uuu	00-- 11u0
Interrupt Wake-up from Sleep	PC + 1(1)	---1 0uuu	uu-- uuuu
RESET Instruction Executed	0000h	---u uuuu	uu-- u0uu
Stack Overflow Reset (STVREN = 1)	0000h	---u uuuu	1u-- uuuu
Stack Underflow Reset (STVREN = 1)	0000h	---u uuuu	u1-- uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0'.

Note 1: When the wake-up is due to an interrupt and the Global Interrupt Enable bit (GIE) is set, the return address is pushed on the stack and PC is loaded with the interrupt vector (0004h) after execution of PC + 1.

6.13 Power Control (PCON) Register

The Power Control (PCON) register contains flag bits to differentiate between a:

- Power-on Reset ($\overline{POR}$)
- Brown-out Reset ($\overline{BOR}$)
- Reset Instruction Reset ($\overline{RI}$)
- $\overline{MCLR}$ Reset ($\overline{RMCLR}$)
- Watchdog Timer Reset ($\overline{RWDT}$)
- Stack Underflow Reset (STKUNF)
- Stack Overflow Reset (STKOVF)

The PCON register bits are shown in Register 6-2.

6.14 Register Definitions: Power Control

REGISTER 6-2: PCON: POWER CONTROL REGISTER

R/W/HS-0/q	R/W/HS-0/q	U-0	R/W/HC-1/q	R/W/HC-1/q	R/W/HC-1/q	R/W/HC-q/u	R/W/HC-q/u
STKOVF	STKUNF	—	$\overline{RWDT}$	$\overline{RMCLR}$	$\overline{RI}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

Legend:		
HC = Bit is cleared by hardware		HS = Bit is set by hardware
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	q = Value depends on condition

bit 7 **STKOVF:** Stack Overflow Flag bit
1 = A Stack Overflow occurred
0 = A Stack Overflow has not occurred or cleared by firmware

bit 6 **STKUNF:** Stack Underflow Flag bit
1 = A Stack Underflow occurred
0 = A Stack Underflow has not occurred or cleared by firmware

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **$\overline{RWDT}$:** Watchdog Timer Reset Flag bit
1 = A Watchdog Timer Reset has not occurred or set by firmware
0 = A Watchdog Timer Reset has occurred (cleared by hardware)

bit 3 **$\overline{RMCLR}$:** $\overline{MCLR}$ Reset Flag bit
1 = A $\overline{MCLR}$ Reset has not occurred or set by firmware
0 = A $\overline{MCLR}$ Reset has occurred (cleared by hardware)

bit 2 **$\overline{RI}$:** RESET Instruction Flag bit
1 = A RESET instruction has not been executed or set by firmware
0 = A RESET instruction has been executed (cleared by hardware)

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Power-on Reset or Brown-out Reset occurs)

TABLE 6-5: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BORCON	SBOREN	BORFS	—	—	—	—	—	BORRDY	46
PCON	STKOVF	STKUNF	—	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	50
STATUS	—	—	—	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	13
WDTCON	—	—	WDTPS<4:0>					SWDTEN	66

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

TABLE 6-6: SUMMARY OF CONFIGURATION WORD WITH RESETS

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	—	—	—	—	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		—	32
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		—	FOSC<1:0>		
CONFIG2	13:8	—	—	LVP	—	$\overline{\text{LPBOR}}$	BORV	STVREN	—	33
	7:0	—	—	—	—	—	—	WRT<1:0>		

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

7.0 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.

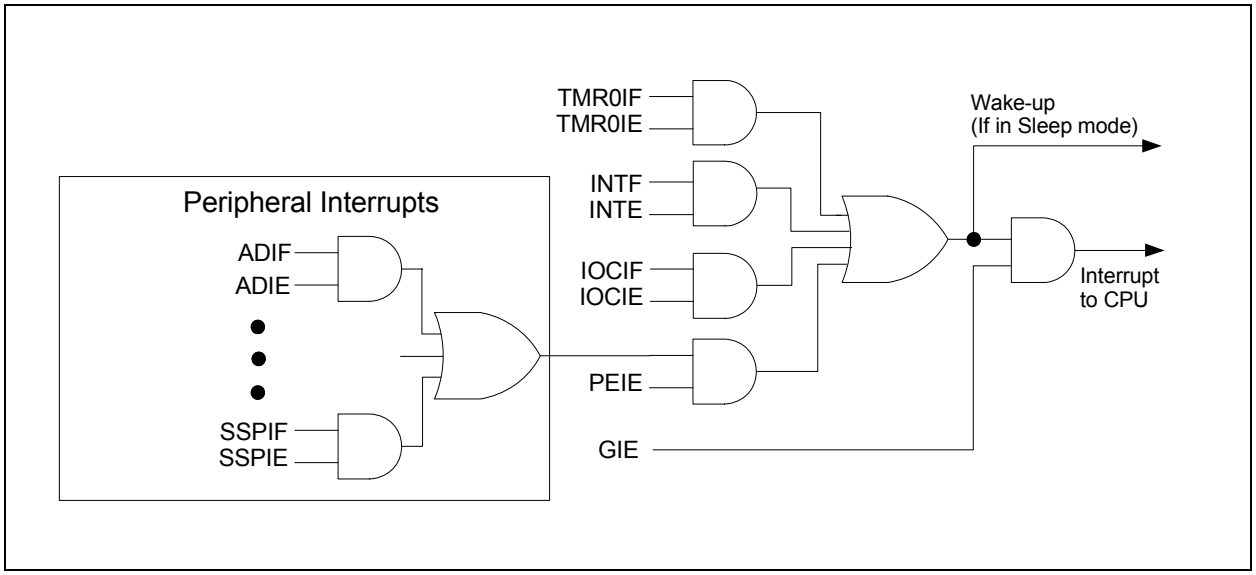
This chapter contains the following information for Interrupts:

- 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 7-1.

FIGURE 7-1: INTERRUPT LOGIC



7.1 Operation

Interrupts are disabled upon any device Reset. They are enabled by setting the following bits:

- GIE bit of the INTCON register
- Interrupt Enable bit(s) for the specific interrupt event(s)
- PEIE bit of the INTCON register (if the Interrupt Enable bit of the interrupt event is contained in the PIE1 and PIE2 registers)

The INTCON, PIR1, and PIR2 registers record individual interrupts via interrupt flag bits. Interrupt flag bits will be set, regardless of the status of the GIE, PEIE and individual interrupt enable bits.

The following events happen when an interrupt event occurs while the GIE bit is set:

- Current prefetched instruction is flushed
- GIE bit is cleared
- Current Program Counter (PC) is pushed onto the stack
- Critical registers are automatically saved to the shadow registers (See **"Section 7.5 "Automatic Context Saving"."**)
- PC is loaded with the interrupt vector 0004h

The firmware within the Interrupt Service Routine (ISR) should determine the source of the interrupt by polling the interrupt flag bits. The interrupt flag bits must be cleared before exiting the ISR to avoid repeated interrupts. Because the GIE bit is cleared, any interrupt that occurs while executing the ISR will be recorded through its interrupt flag, but will not cause the processor to redirect to the interrupt vector.

The `RETFIE` instruction exits the ISR by popping the previous address from the stack, restoring the saved context from the shadow registers and setting the GIE bit.

For additional information on a specific interrupt's operation, refer to its peripheral chapter.

Note 1:	Individual interrupt flag bits are set, regardless of the state of any other enable bits.
2:	All interrupts will be ignored while the GIE bit is cleared. Any interrupt occurring while the GIE bit is clear will be serviced when the GIE bit is set again.

7.2 Interrupt Latency

Interrupt latency is defined as the time from when the interrupt event occurs to the time code execution at the interrupt vector begins. The latency for synchronous interrupts is three or four instruction cycles. For asynchronous interrupts, the latency is three to five instruction cycles, depending on when the interrupt occurs. See Figure 7-2 and Figure 7-3 for more details.

FIGURE 7-2: INTERRUPT LATENCY

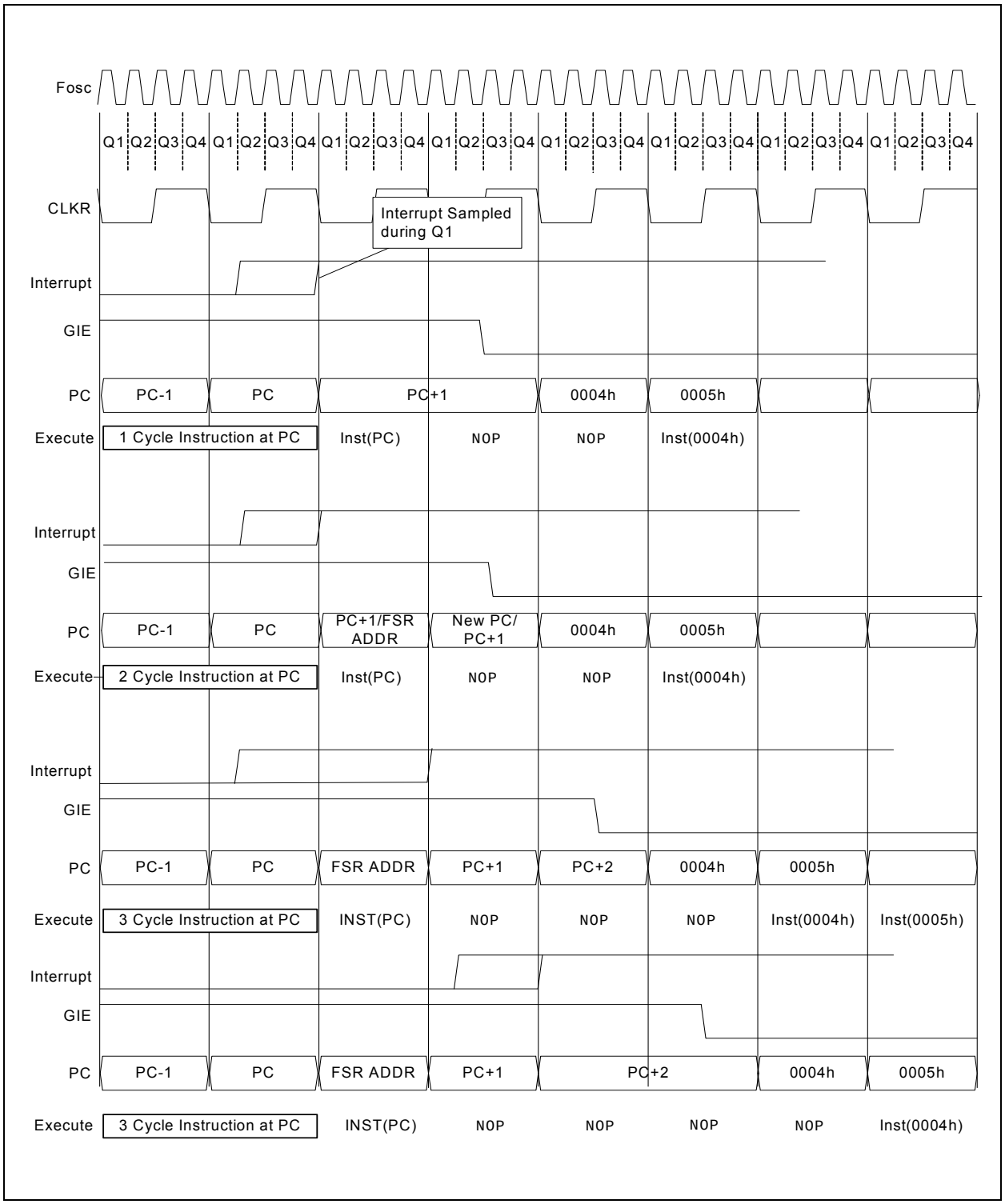
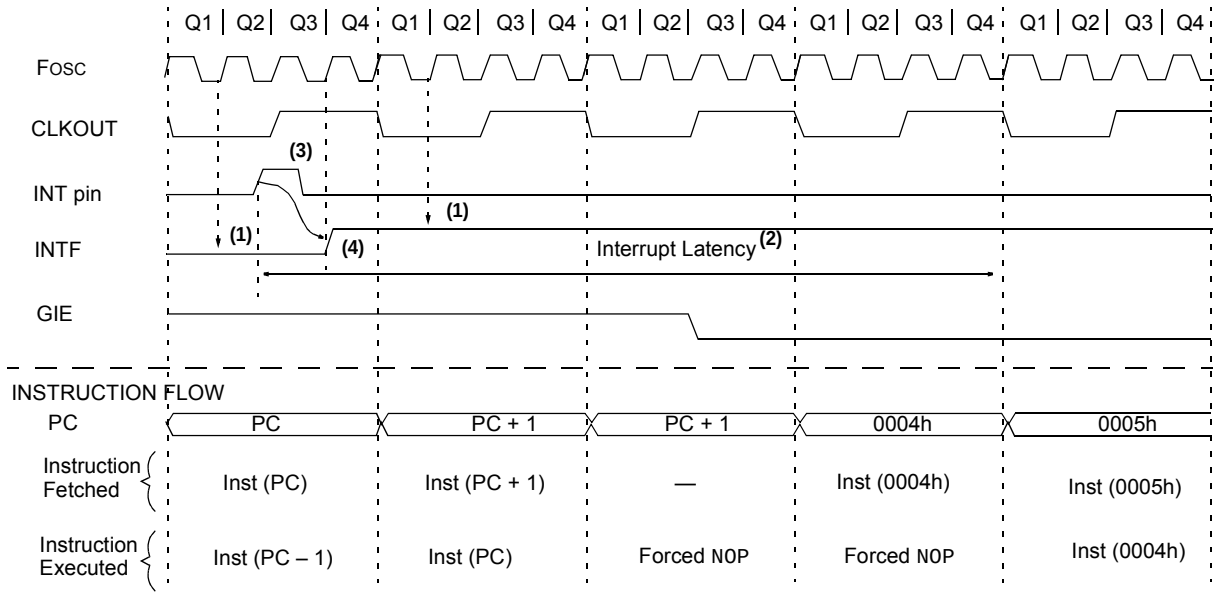


FIGURE 7-3: INT PIN INTERRUPT TIMING



Note 1: INTF flag is sampled here (every Q1).

2: Asynchronous interrupt latency = 3-5 TCY. Synchronous latency = 3-4 TCY, where TCY = instruction cycle time. Latency is the same whether Inst (PC) is a single cycle or a 2-cycle instruction.

3: For minimum width of INT pulse, refer to AC specifications in **Section 21.0 "Electrical Specifications"**".

4: INTF is enabled to be set any time during the Q4-Q1 cycles.

7.3 Interrupts During Sleep

Some interrupts can be used to wake from Sleep. To wake from Sleep, the peripheral must be able to operate without the system clock. The interrupt source must have the appropriate Interrupt Enable bit(s) set prior to entering Sleep.

On waking from Sleep, if the GIE bit is also set, the processor will branch to the interrupt vector. Otherwise, the processor will continue executing instructions after the SLEEP instruction. The instruction directly after the SLEEP instruction will always be executed before branching to the ISR. Refer to **Section 8.0 "Power-down Mode (Sleep)"** for more details.

7.4 INT Pin

The INT pin can be used to generate an asynchronous edge-triggered interrupt. This interrupt is enabled by setting the INTE bit of the INTCON register. The INTEDG bit of the OPTION_REG register determines on which edge the interrupt will occur. When the INTEDG bit is set, the rising edge will cause the interrupt. When the INTEDG bit is clear, the falling edge will cause the interrupt. The INTF bit of the INTCON register will be set when a valid edge appears on the INT pin. If the GIE and INTE bits are also set, the processor will redirect program execution to the interrupt vector.

7.5 Automatic Context Saving

Upon entering an interrupt, the return PC address is saved on the stack. Additionally, the following registers are automatically saved in the Shadow registers:

- W register
- STATUS register (except for $\overline{TO}$ and $\overline{PD}$)
- BSR register
- FSR registers
- PCLATH register

Upon exiting the Interrupt Service Routine, these registers are automatically restored. Any modifications to these registers during the ISR will be lost. If modifications to any of these registers are desired, the corresponding shadow register should be modified and the value will be restored when exiting the ISR. The shadow registers are available in Bank 31 and are readable and writable. Depending on the user's application, other registers may also need to be saved.

7.6 Register Definitions: Interrupt Control

REGISTER 7-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R-0/0
GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF(1)
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 **GIE:** Global Interrupt Enable bit
1 = Enables all active interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all active peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE:** Timer0 Overflow Interrupt Enable bit
1 = Enables the Timer0 interrupt
0 = Disables the Timer0 interrupt

bit 4 **INTE:** INT External Interrupt Enable bit
1 = Enables the INT external interrupt
0 = Disables the INT external interrupt

bit 3 **IOCIE:** Interrupt-on-Change Enable bit
1 = Enables the interrupt-on-change
0 = Disables the interrupt-on-change

bit 2 **TMR0IF:** Timer0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed
0 = TMR0 register did not overflow

bit 1 **INTF:** INT External Interrupt Flag bit
1 = The INT external interrupt occurred
0 = The INT external interrupt did not occur

bit 0 **IOCIF:** Interrupt-on-Change Interrupt Flag bit(1)
1 = When at least one of the interrupt-on-change pins changed state
0 = None of the interrupt-on-change pins have changed state

Note 1: The IOCIF Flag bit is read-only and cleared when all the interrupt-on-change flags in the IOCBF register have been cleared by software.

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

REGISTER 7-2: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

U-0	R/W-0/0	U-0	U-0	R/W-0/0	U-0	U-0	U-0
—	ADIE	—	—	SSPIE	—	—	—
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 **Unimplemented:** Read as '0'

bit 6 **ADIE:** Analog-to-Digital Converter (ADC) Interrupt Enable bit
1 = Enables the ADC interrupt
0 = Disables the ADC interrupt

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **SSPIE:** Synchronous Serial Port (MSSP) Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

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

Note: Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.

REGISTER 7-3: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

U-0	U-0	U-0	U-0	R/W-0/0	U-0	U-0	U-0
—	—	—	—	BCLIE	—	—	—
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 7-4 **Unimplemented:** Read as '0'

bit 3 **BCLIE:** MSSP Bus Collision Interrupt Enable bit
1 = Enables the MSSP Bus Collision Interrupt
0 = Disables the MSSP Bus Collision Interrupt

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

Note: Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.

REGISTER 7-4: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

U-0	R/W-0/0	U-0	U-0	R/W-0/0	U-0	U-0	U-0
—	ADIF	—	—	SSPIF	—	—	—
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 **Unimplemented:** Read as '0'

bit 6 **ADIF:** ADC Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **SSPIF:** Synchronous Serial Port (MSSP) Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

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

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

REGISTER 7-5: PIR2: PERIPHERAL INTERRUPT REQUEST REGISTER 2

U-0	U-0	U-0	U-0	R/W-0/0	U-0	U-0	U-0
—	—	—	—	BCLIF	—	—	—
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 7-4 **Unimplemented:** Read as '0'

bit 3 **BCLIF:** MSSP Bus Collision Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

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

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

TABLE 7-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
OPTION_REG	$\overline{\text{WPUEN}}$	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>			132
PIE1	—	ADIE	—	—	SSPIE	—	—	—	58
PIE2	—	—	—	—	BCLIE	—	—	—	59
PIR1	—	ADIF	—	—	SSPIF	—	—	—	60
PIR2	—	—	—	—	BCLIF	—	—	—	61

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

8.0 POWER-DOWN MODE (SLEEP)

The Power-Down mode is entered by executing a SLEEP instruction.

Upon entering Sleep mode, the following conditions exist:

1. WDT will be cleared but keeps running, if enabled for operation during Sleep.
2. $\overline{PD}$ bit of the STATUS register is cleared.
3. $\overline{TO}$ bit of the STATUS register is set.
4. CPU clock is disabled.
5. 31 kHz LFINTOSC is unaffected and peripherals that operate from it may continue operation in Sleep.
6. ADC is unaffected, if the dedicated FRC clock is selected.
7. I/O ports maintain the status they had before SLEEP was executed (driving high, low or high-impedance).
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 31 kHz LFINTOSC

I/O pins that are high-impedance inputs should be pulled to V_{DD} or V_{SS} externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include the FVR module. See **Section 13.0 "Fixed Voltage Reference (FVR)"** for more information on this module.

8.1 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 6.12 "Determining the Cause of a Reset"**.

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.

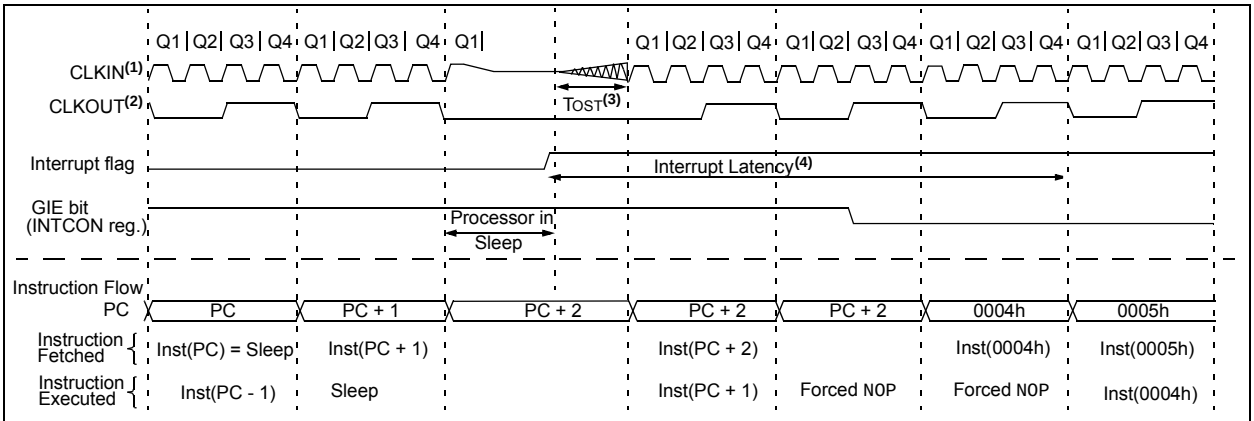
8.1.1 WAKE-UP USING INTERRUPTS

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

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

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

FIGURE 8-1: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: External clock. High, Medium, Low mode assumed.
2: CLKOUT is not available in XT, HS, or LP Oscillator modes, but shown here for timing reference.
3: TOST = 1024 TOSC (drawing not to scale). This delay applies only to XT, HS or LP Oscillator modes.
4: GIE = 1 assumed. In this case after wake-up, the processor calls the ISR at 0004h. If GIE = 0, execution will continue in-line.

TABLE 8-1: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
IOCAF	—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0	92
IOCAN	—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0	92
IOCAP	—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0	92
PIE1	—	ADIE	—	—	SSPIE	—	—	—	58
PIE2	—	—	—	—	BCLIE	—	—	—	59
PIR1	—	ADIF	—	—	SSPIF	—	—	—	60
PIR2	—	—	—	—	BCLIF	—	—	—	61
STATUS	—	—	—	$\overline{TO}$	$\overline{PD}$	Z	DC	C	13
WDTCON	—	—	WDTPS<4:0>					SWDTEN	66

Legend: — = unimplemented, read as '0'. Shaded cells are not used in Power-Down mode.

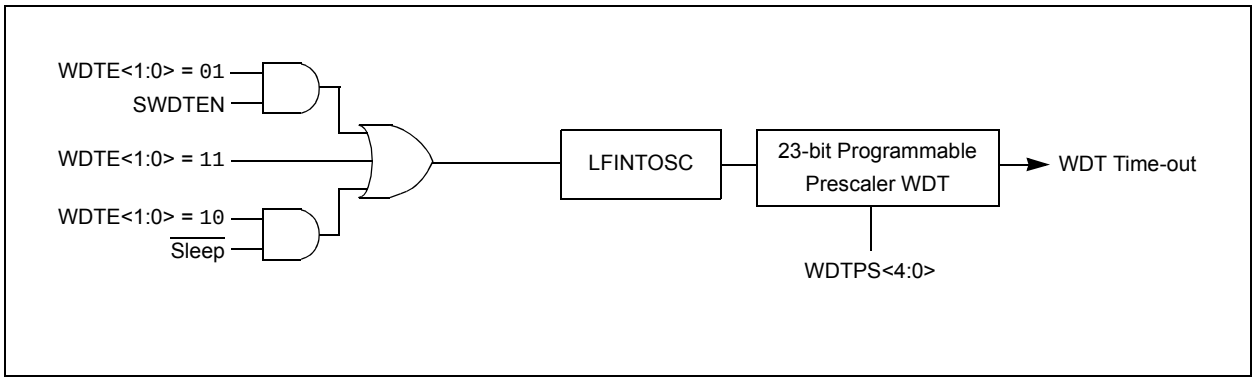
9.0 WATCHDOG TIMER (WDT)

The Watchdog Timer is a system timer that generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period. The Watchdog Timer is typically used to recover the system from unexpected events.

The WDT has the following features:

- Independent clock source
- Multiple operating modes
  - WDT is always on
  - WDT is off when in Sleep
  - WDT is controlled by software
  - WDT is always off
- Configurable time-out period is from 1 ms to 256 seconds (nominal)
- Multiple Reset conditions
- Operation during Sleep

FIGURE 9-1: WATCHDOG TIMER BLOCK DIAGRAM



9.1 Independent Clock Source

The WDT derives its time base from the 31 kHz LFINTOSC internal oscillator. Time intervals in this chapter are based on a nominal interval of 1 ms. See **Section 21.0 "Electrical Specifications"** for the LFINTOSC tolerances.

9.2 WDT Operating Modes

The Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Words. See Table 9-1.

9.2.1 WDT IS ALWAYS ON

When the WDTE bits of Configuration Words are set to '11', the WDT is always on.

WDT protection is active during Sleep.

9.2.2 WDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to '10', the WDT is on, except in Sleep.

WDT protection is not active during Sleep.

9.2.3 WDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to '01', the WDT is controlled by the SWDTEN bit of the WDTCON register.

WDT protection is unchanged by Sleep. See Table 9-1 for more details.

TABLE 9-1: WDT OPERATING MODES

WDTE<1:0>	SWDTEN	Device Mode	WDT Mode
11	X	X	Active
10	X	Awake	Active
		Sleep	Disabled
01	1	X	Active
	0	X	Disabled
00	X	X	Disabled

9.3 Time-out Period

The WDTPS bits of the WDTCON register set the time-out period from 1 ms to 256 seconds (nominal). After a Reset, the default time-out period is two seconds.

9.4 Clearing the WDT

The WDT is cleared when any of the following conditions occur:

- Any Reset
- CLRWDT instruction is executed
- Device enters Sleep
- Device wakes up from Sleep
- Oscillator fail
- WDT is disabled

See Table 9-2 for more information.

9.5 Operation During Sleep

When the device enters Sleep, the WDT is cleared. If the WDT is enabled during Sleep, the WDT resumes counting. When the device exits Sleep, the WDT is cleared again.

When a WDT time-out occurs while the device is in Sleep, no Reset is generated. Instead, the device wakes up and resumes operation. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register are changed to indicate the event. The $\overline{RWDT}$ bit in the PCON register can also be used. See **Section 3.0 "Memory Organization"** for more information.

TABLE 9-2: WDT CLEARING CONDITIONS

Conditions	WDT
WDTE<1:0> = 00	Cleared
WDTE<1:0> = 01 and SWDTEN = 0	
WDTE<1:0> = 10 and enter Sleep	
CLRWDT Command	
Oscillator Fail Detected	
Exit Sleep + System Clock = INTOSC, EXTCLK	
Change INTOSC divider (IRCF bits)	Unaffected

9.6 Register Definitions: Watchdog Control

REGISTER 9-1: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-1/1	R/W-0/0
—	—	WDTPS<4:0>					SWDTEN
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 5-1 **WDTPS<4:0>:** Watchdog Timer Period Select bits(1)

Bit Value = Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)
•
•
•
10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 (2^{23}) (Interval 256s nominal)
10001 = 1:4194304 (2^{22}) (Interval 128s nominal)
10000 = 1:2097152 (2^{21}) (Interval 64s nominal)
01111 = 1:1048576 (2^{20}) (Interval 32s nominal)
01110 = 1:524288 (2^{19}) (Interval 16s nominal)
01101 = 1:262144 (2^{18}) (Interval 8s nominal)
01100 = 1:131072 (2^{17}) (Interval 4s nominal)
01011 = 1:65536 (Interval 2s nominal) (Reset value)
01010 = 1:32768 (Interval 1s nominal)
01001 = 1:16384 (Interval 512 ms nominal)
01000 = 1:8192 (Interval 256 ms nominal)
00111 = 1:4096 (Interval 128 ms nominal)
00110 = 1:2048 (Interval 64 ms nominal)
00101 = 1:1024 (Interval 32 ms nominal)
00100 = 1:512 (Interval 16 ms nominal)
00011 = 1:256 (Interval 8 ms nominal)
00010 = 1:128 (Interval 4 ms nominal)
00001 = 1:64 (Interval 2 ms nominal)
00000 = 1:32 (Interval 1 ms nominal)

bit 0 **SWDTEN:** Software Enable/Disable for Watchdog Timer bit

If WDTE<1:0> = 1x:
This bit is ignored.
If WDTE<1:0> = 01:
1 = WDT is turned on
0 = WDT is turned off
If WDTE<1:0> = 00:
This bit is ignored.

Note 1: Times are approximate. WDT time is based on 31 kHz LFINTOSC.

TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH WATCHDOG TIMER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
OSCCON	PLLEN	IRCF<3:0>				—	SCS<1:0>		42
PCON	STKOVF	STKUNF	—	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	50
STATUS	—	—	—	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	13
WDTCON	—	—	WDTPS<4:0>					SWDTEN	66

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by Watchdog Timer.

TABLE 9-4: SUMMARY OF CONFIGURATION WORD WITH WATCHDOG TIMER

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	—	—	—	—	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		—	32
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		—	FOSC<1:0>		

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

10.0 FLASH PROGRAM MEMORY CONTROL

The Flash program memory is readable and writable during normal operation over the full VDD range. Program memory is indirectly addressed using Special Function Registers (SFRs). The SFRs used to access program memory are:

- PMCON1
- PMCON2
- PMDATL
- PMDATH
- PMADRL
- PMADRH

When accessing the program memory, the PMDATH:PMDATL register pair forms a 2-byte word that holds the 14-bit data for read/write, and the PMADRH:PMADRL register pair forms a 2-byte word that holds the 15-bit address of the program memory location being read.

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump.

The Flash program memory can be protected in two ways; by code protection ($\overline{CP}$ bit in Configuration Words) and write protection (WRT<1:0> bits in Configuration Words).

Code protection ($\overline{CP}$ = 0)**(1)**, disables access, reading and writing, to the Flash program memory via external device programmers. Code protection does not affect the self-write and erase functionality. Code protection can only be reset by a device programmer performing a Bulk Erase to the device, clearing all Flash program memory, Configuration bits and User IDs.

Write protection prohibits self-write and erase to a portion or all of the Flash program memory as defined by the bits WRT<1:0>. Write protection does not affect a device programmers ability to read, write or erase the device.

Note 1:	Code protection of the entire Flash program memory array is enabled by clearing the $\overline{CP}$ bit of Configuration Words.

10.1 PMADRL and PMADRH Registers

The PMADRH:PMADRL register pair can address up to a maximum of 16K words of program memory. When selecting a program address value, the MSB of the address is written to the PMADRH register and the LSB is written to the PMADRL register.

10.1.1 PMCON1 AND PMCON2 REGISTERS

PMCON1 is the control register for Flash program memory accesses.

Control bits RD and WR initiate read and write, respectively. These bits cannot be cleared, only set, in software. They are cleared by hardware at completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental, premature termination of a write operation.

The WREN bit, when set, will allow a write operation to occur. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a Reset during normal operation. In these situations, following Reset, the user can check the WRERR bit and execute the appropriate error handling routine.

The PMCON2 register is a write-only register. Attempting to read the PMCON2 register will return all ‘0’s.

To enable writes to the program memory, a specific pattern (the unlock sequence), must be written to the PMCON2 register. The required unlock sequence prevents inadvertent writes to the program memory write latches and Flash program memory.

10.2 Flash Program Memory Overview

It is important to understand the Flash program memory structure for erase and programming operations. Flash program memory is arranged in rows. A row consists of a fixed number of 14-bit program memory words. A row is the minimum size that can be erased by user software.

After a row has been erased, the user can reprogram all or a portion of this row. Data to be written into the program memory row is written to 14-bit wide data write latches. These write latches are not directly accessible to the user, but may be loaded via sequential writes to the PMDATH:PMDATL register pair.

Note:	If the user wants to modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, new data and retained data can be written into the write latches to reprogram the row of Flash program memory. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations.

See Table 10-1 for Erase Row size and the number of write latches for Flash program memory.

TABLE 10-1: FLASH MEMORY ORGANIZATION BY DEVICE

Device	Row Erase (words)	Write Latches (words)
PIC12LF1552	16	16

10.2.1 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

1. Write the desired address to the PMADRH:PMADRL register pair.
2. Clear the CFGS bit of the PMCON1 register.
3. Then, set control bit RD of the PMCON1 register.

Once the read control bit is set, the program memory Flash controller will use the second instruction cycle to read the data. This causes the second instruction immediately following the "`BSF PMCON1,RD`" instruction to be ignored. The data is available in the very next cycle, in the PMDATH:PMDATL register pair; therefore, it can be read as two bytes in the following instructions.

PMDATH:PMDATL register pair will hold this value until another read or until it is written to by the user.

Note:	The two instructions following a program memory read are required to be `NOP`s. This prevents the user from executing a two-cycle instruction on the next instruction after the RD bit is set.

FIGURE 10-1: FLASH PROGRAM MEMORY READ FLOWCHART

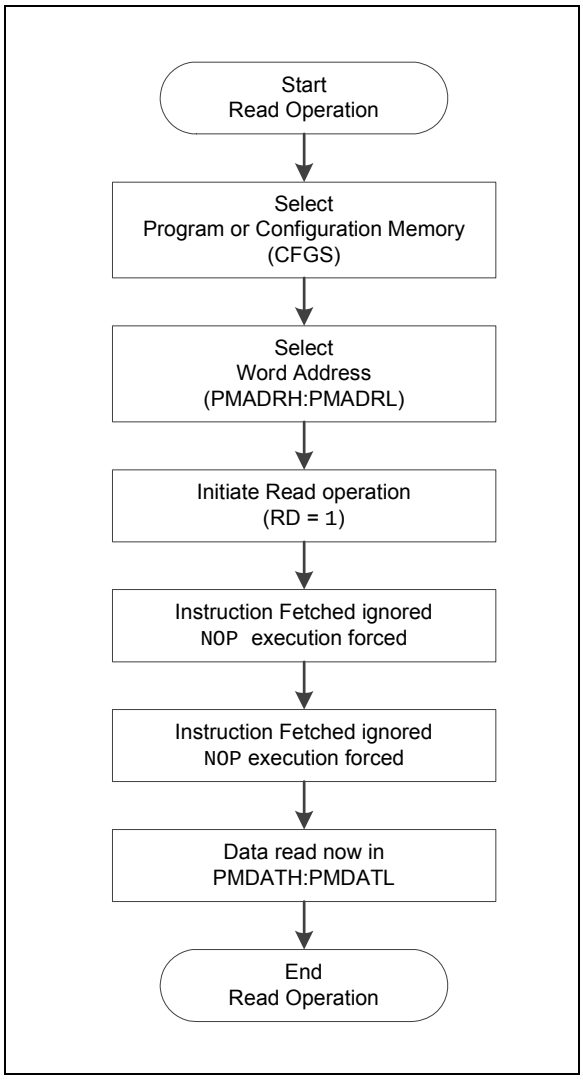
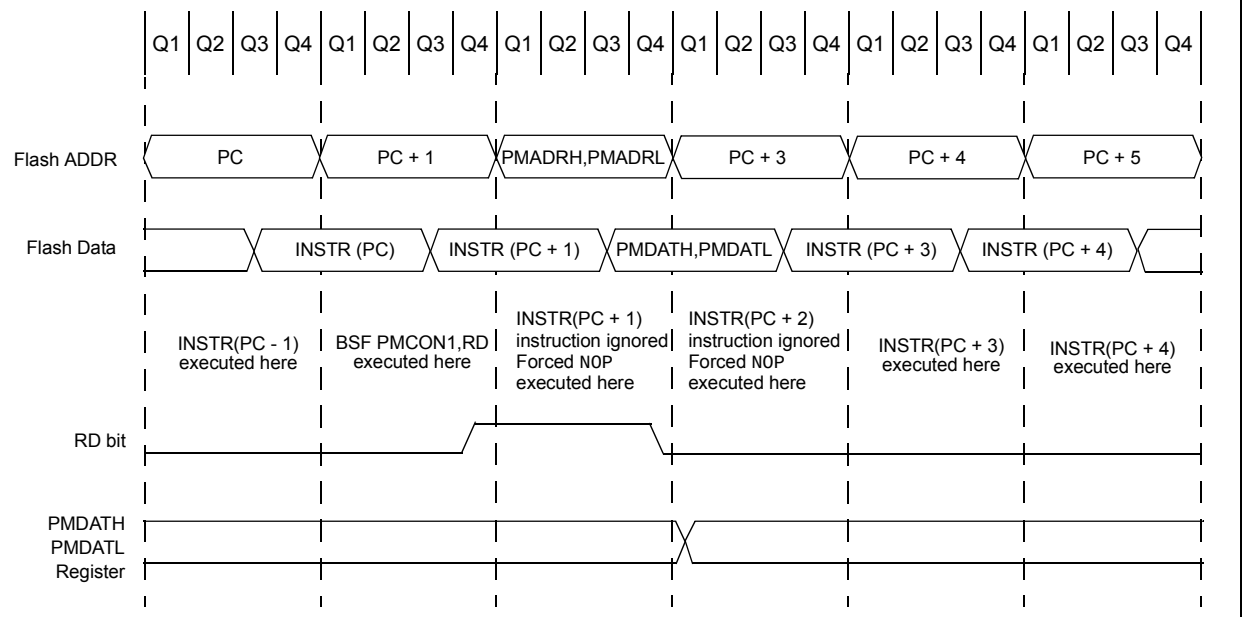


FIGURE 10-2: FLASH PROGRAM MEMORY READ CYCLE EXECUTION



EXAMPLE 10-1: 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  PMADRL           ; Select Bank for PMCON registers
    MOVLW    PROG_ADDR_LO     ;
    MOVWF    PMADRL           ; Store LSB of address
    MOVLW    PROG_ADDR_HI     ;
    MOVWF    PMADRH           ; Store MSB of address

    BCF      PMCON1,CFGS      ; Do not select Configuration Space
    BSF      PMCON1,RD        ; Initiate read
    NOP                       ; Ignored (Figure 10-2)
    NOP                       ; Ignored (Figure 10-2)

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

10.2.2 FLASH MEMORY UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the Flash program memory from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- Row Erase
- Load program memory write latches
- Write of program memory write latches to program memory
- Write of program memory write latches to User IDs

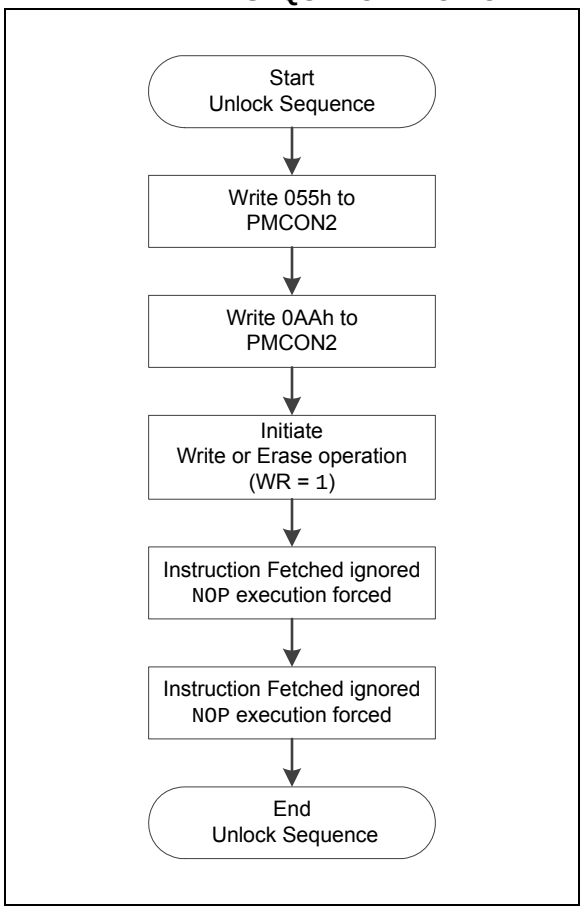
The unlock sequence consists of the following steps:

1. Write 55h to PMCON2
2. Write AAh to PMCON2
3. Set the WR bit in PMCON1
4. NOP instruction
5. NOP instruction

Once the WR bit is set, the processor will always force two NOP instructions. When an Erase Row or Program Row operation is being performed, the processor will stall internal operations (typical 2 ms), until the operation is complete and then resume with the next instruction. When the operation is loading the program memory write latches, the processor will always force the two NOP instructions and continue uninterrupted with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

FIGURE 10-3: FLASH PROGRAM MEMORY UNLOCK SEQUENCE FLOWCHART



10.2.3 ERASING FLASH PROGRAM MEMORY

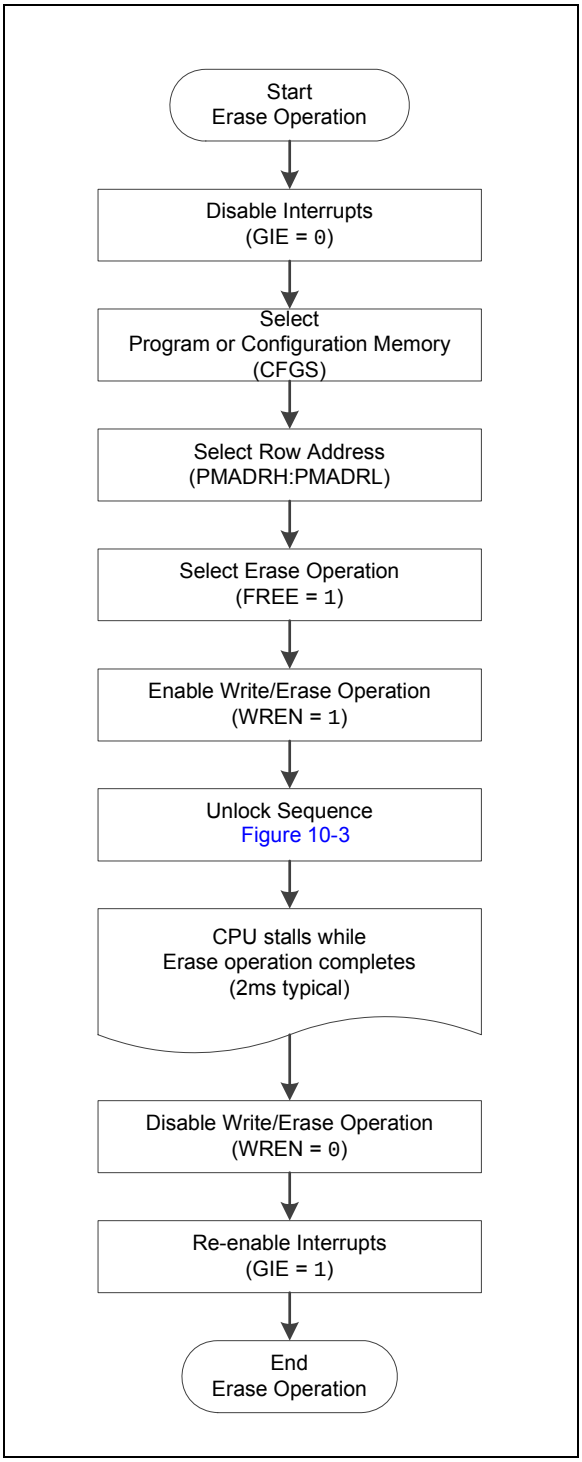
While executing code, program memory can only be erased by rows. To erase a row:

1. Load the PMADRH:PMADRL register pair with any address within the row to be erased.
2. Clear the CFGS bit of the PMCON1 register.
3. Set the FREE and WREN bits of the PMCON1 register.
4. Write 55h, then AAh, to PMCON2 (Flash programming unlock sequence).
5. Set control bit WR of the PMCON1 register to begin the erase operation.

See Example 10-2.

After the "`BSF PMCON1,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 PMCON1 write instruction.

FIGURE 10-4: FLASH PROGRAM MEMORY ERASE FLOWCHART



EXAMPLE 10-2: ERASING ONE ROW OF PROGRAM MEMORY

```
; This row erase routine assumes the following:
; 1. A valid address within the erase row is loaded in ADDRH:ADDRL
; 2. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)


        BCF         INTCON,GIE      ; Disable ints so required sequences will execute properly
        BANKSEL     PMADRL
        MOVF        ADDRL,W         ; Load lower 8 bits of erase address boundary
        MOVWF       PMADRL
        MOVF        ADDRH,W         ; Load upper 6 bits of erase address boundary
        MOVWF       PMADRH
        BCF         PMCON1,CFGS     ; Not configuration space
        BSF         PMCON1,FREE     ; Specify an erase operation
        BSF         PMCON1,WREN     ; Enable writes

Required
Sequence
        MOVLW       55h             ; Start of required sequence to initiate erase
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin erase
        NOP                         ; NOP instructions are forced as processor starts
        NOP                         ; row erase of program memory.
                                    ;
                                    ; The processor stalls until the erase process is complete
                                    ; after erase processor continues with 3rd instruction

        BCF         PMCON1,WREN     ; Disable writes
        BSF         INTCON,GIE      ; Enable interrupts
```

10.2.4 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the address in PMADRH:PMADRL of the row to be programmed.
2. Load each write latch 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 10-5 (row writes to program memory with 16 write latches) for more details.

The write latches are aligned to the Flash row address boundary defined by the upper eleven bits of PMADRH:PMADRL, (PMADRH<6:0>:PMADRL<7:4>) with the lower four bits of PMADRL, (PMADRL<3:0>) determining the write latch being loaded. Write operations do not cross these boundaries. At the completion of a program memory write operation, the data in the write latches is reset to contain 0x3FFF.

The following steps should be completed to load the write latches and program a row of program memory. These steps are divided into two parts. First, each write latch is loaded with data from the PMDATH:PMDATL using the unlock sequence with LWLO = 1. When the last word to be loaded into the write latch is ready, the LWLO bit is cleared and the unlock sequence executed. This initiates the programming operation, writing all the latches into Flash program memory.

Note:	The special unlock sequence is required to load a write latch with data or initiate a Flash programming operation. If the unlock sequence is interrupted, writing to the latches or program memory will not be initiated.

1. Set the WREN bit of the PMCON1 register.
2. Clear the CFGS bit of the PMCON1 register.
3. Set the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 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 PMADRH:PMADRL register pair with the address of the location to be written.
5. Load the PMDATH:PMDATL register pair with the program memory data to be written.
6. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The write latch is now loaded.
7. Increment the PMADRH:PMADRL 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 PMCON1 register. When the LWLO bit of the PMCON1 register is '0', the write sequence will initiate the write to Flash program memory.
10. Load the PMDATH:PMDATL register pair with the program memory data to be written.
11. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The entire program memory latch content is now written to Flash program memory.

Note:	The program memory write latches are reset to the blank state (0x3FFF) at the completion of every write or erase operation. As a result, it is not necessary to load all the program memory write latches. Unloaded latches will remain in the blank state.

An example of the complete write sequence is shown in Example 10-3. The initial address is loaded into the PMADRH:PMADRL register pair; the data is loaded using indirect addressing.

FIGURE 10-5: BLOCK WRITES TO FLASH PROGRAM MEMORY WITH 16 WRITE LATCHES

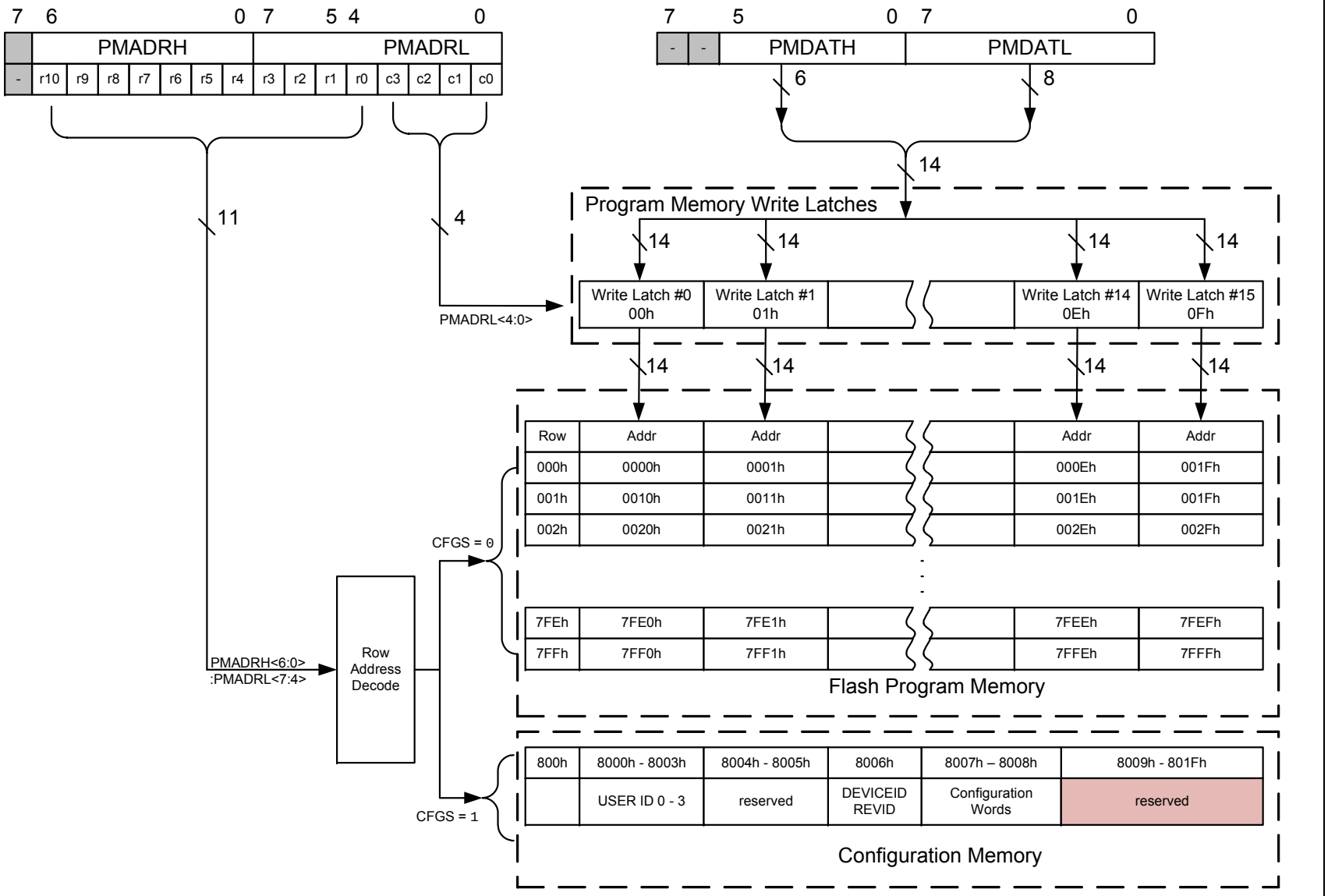
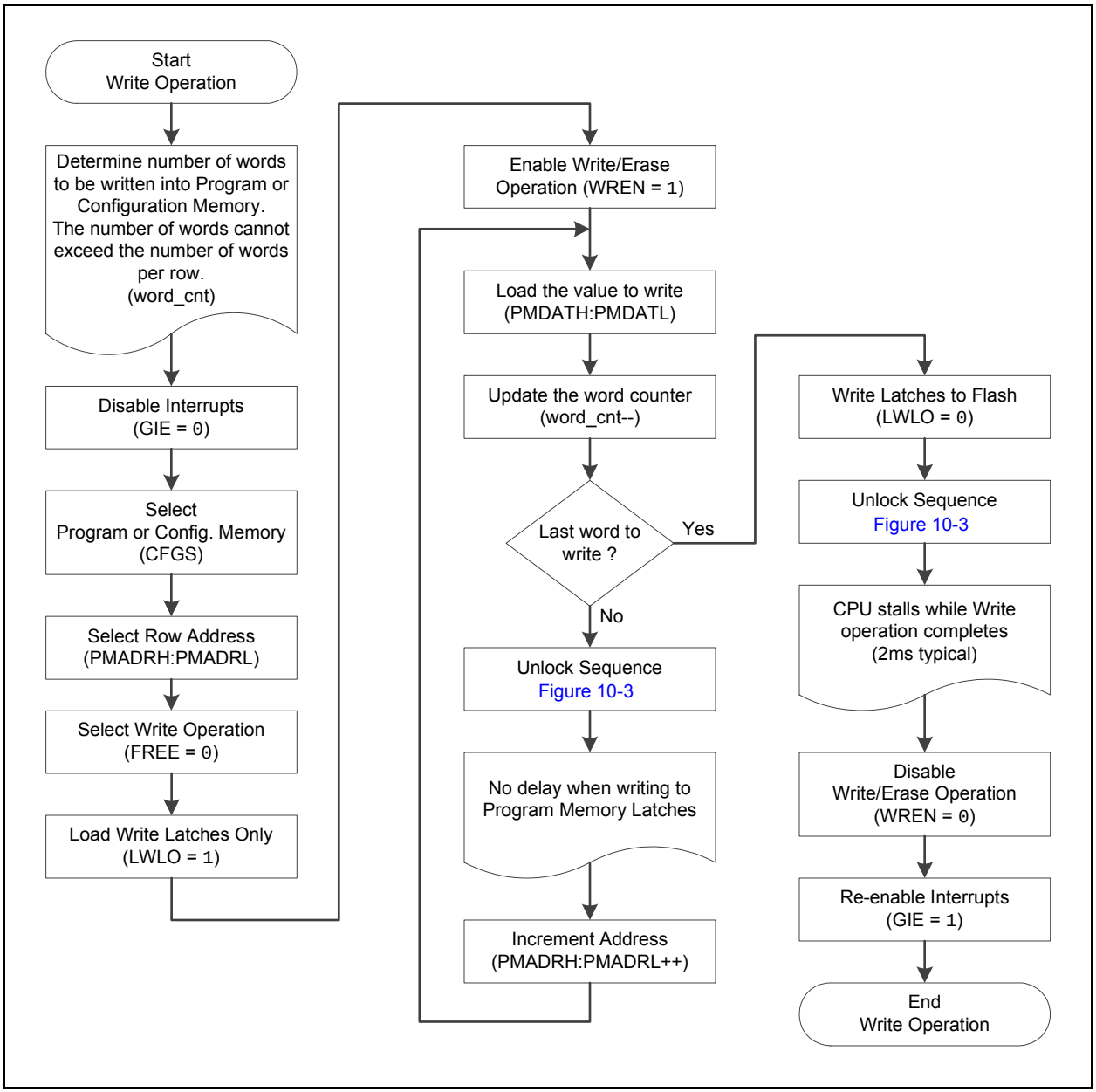


FIGURE 10-6: FLASH PROGRAM MEMORY WRITE FLOWCHART



EXAMPLE 10-3: WRITING TO FLASH PROGRAM MEMORY

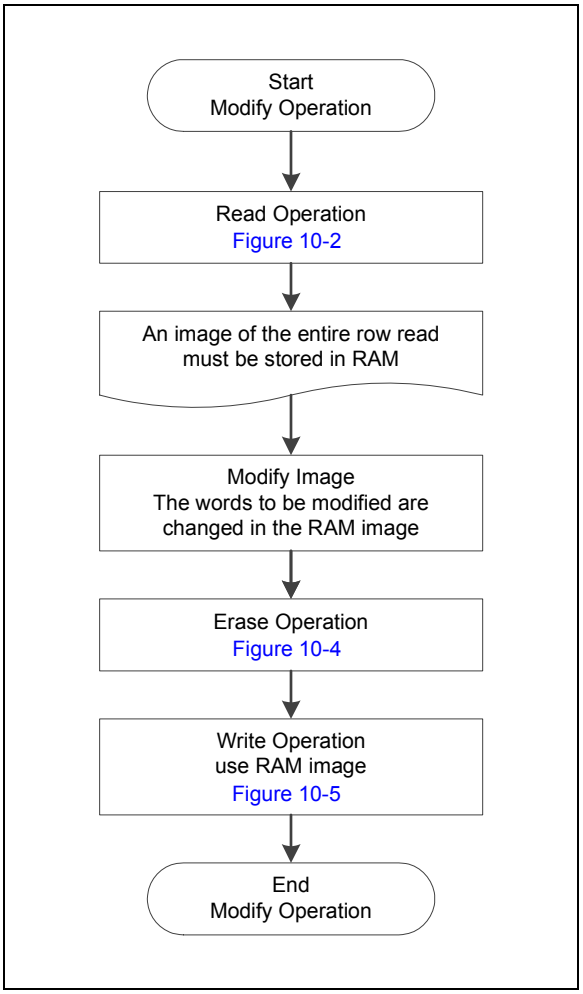
```
; This write routine assumes the following:
; 1. 32 bytes of data are loaded, starting at the address in DATA_ADDR
; 2. Each word of data to be written is made up of two adjacent bytes in DATA_ADDR,
; stored in little endian format
; 3. A valid starting address (the least significant bits = 00000) is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)
;
        BCF         INTCON,GIE      ; Disable ints so required sequences will execute properly
        BANKSEL     PMADRH          ; Bank 3
        MOVF        ADDRH,W         ; Load initial address
        MOVWF       PMADRH          ;
        MOVF        ADDRL,W         ;
        MOVWF       PMADRL          ;
        MOVLW       LOW DATA_ADDR   ; Load initial data address
        MOVWF       FSR0L           ;
        MOVLW       HIGH DATA_ADDR  ; Load initial data address
        MOVWF       FSR0H           ;
        BCF         PMCON1,CFGS     ; Not configuration space
        BSF         PMCON1,WREN     ; Enable writes
        BSF         PMCON1,LWLO     ; Only Load Write Latches
LOOP
        MOVIW       FSR0++          ; Load first data byte into lower
        MOVWF       PMDATL          ;
        MOVIW       FSR0++          ; Load second data byte into upper
        MOVWF       PMDATH          ;

        MOVF        PMADRL,W        ; Check if lower bits of address are '00000'
        XORLW       0x0F            ; Check if we're on the last of 16 addresses
        ANDLW       0x0F            ;
        BTFSC       STATUS,Z        ; Exit if last of 16 words,
        GOTO        START_WRITE     ;

  Required Sequence
        MOVLW       55h             ; Start of required write sequence:
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin write
        NOP                         ; NOP instructions are forced as processor
                                    ; loads program memory write latches
        NOP                         ;

        INCF        PMADRL,F        ; Still loading latches Increment address
        GOTO        LOOP            ; Write next latches

START_WRITE
        BCF         PMCON1,LWLO     ; No more loading latches - Actually start Flash program
                                    ; memory write

  Required Sequence
        MOVLW       55h             ; Start of required write sequence:
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin write
        NOP                         ; NOP instructions are forced as processor writes
                                    ; all the program memory write latches simultaneously
        NOP                         ; to program memory.
                                    ; After NOPs, the processor
                                    ; stalls until the self-write process in complete
                                    ; after write processor continues with 3rd instruction
        BCF         PMCON1,WREN     ; Disable writes
        BSF         INTCON,GIE      ; Enable interrupts
```

10.3 Modifying Flash Program Memory

When modifying existing data in a program memory row, and data within that row must be preserved, it must first be read and saved in a RAM image. Program memory is modified using the following steps:

1. Load the starting address of the row to be modified.
2. Read the existing data from the row into a RAM image.
3. Modify the RAM image to contain the new data to be written into program memory.
4. Load the starting address of the row to be rewritten.
5. Erase the program memory row.
6. Load the write latches with data from the RAM image.
7. Initiate a programming operation.

FIGURE 10-7: FLASH PROGRAM MEMORY MODIFY FLOWCHART



10.4 User ID, Device ID and Configuration Word Access

Instead of accessing program memory, the User ID's, Device ID/Revision ID and Configuration Words can be accessed when CFGS = 1 in the PMCON1 register. This is the region that would be pointed to by PC<15> = 1, but not all addresses are accessible. Different access may exist for reads and writes. Refer to Table 10-2.

When read access is initiated on an address outside the parameters listed in Table 10-2, the PMDATH:PMDATL register pair is cleared, reading back '0's.

TABLE 10-2: USER ID, DEVICE ID AND CONFIGURATION WORD ACCESS (CFGS = 1)

Address	Function	Read Access	Write Access
8000h-8003h	User IDs	Yes	Yes
8006h	Device ID/Revision ID	Yes	No
8007h-8008h	Configuration Words 1 and 2	Yes	No

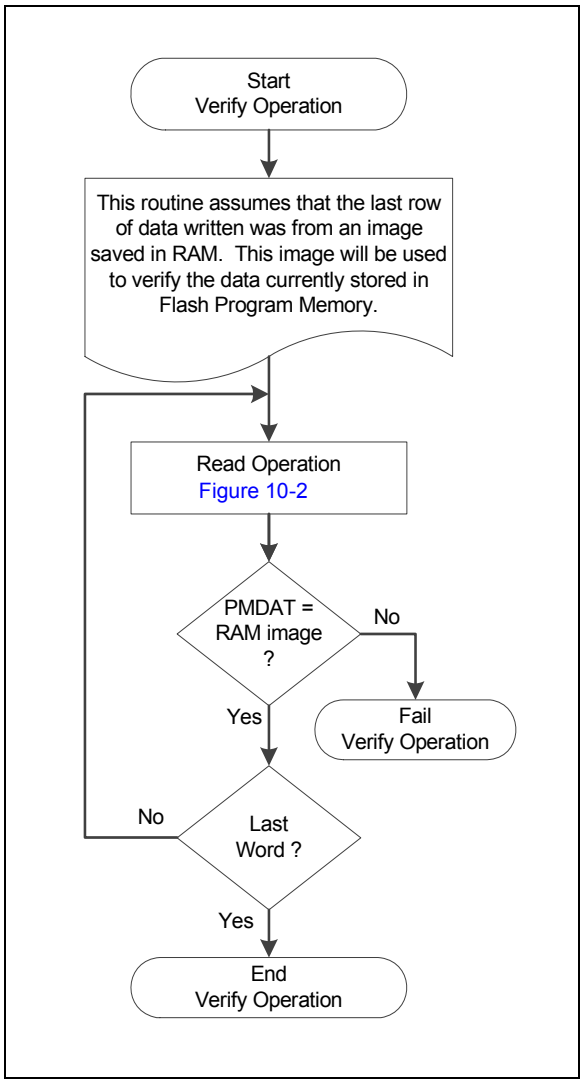
EXAMPLE 10-4: CONFIGURATION WORD AND DEVICE ID ACCESS

```
* This code block will read 1 word of program memory at the memory address:
*   PROG_ADDR_LO (must be 00h-08h) data will be returned in the variables;
*   PROG_DATA_HI, PROG_DATA_LO

    BANKSEL  PMADRL           ; Select correct Bank
    MOVLW    PROG_ADDR_LO     ;
    MOVWF    PMADRL           ; Store LSB of address
    CLRF     PMADRH           ; Clear MSB of address

    BSF      PMCON1,CFGS      ; Select Configuration Space
    BCF      INTCON,GIE       ; Disable interrupts
    BSF      PMCON1,RD        ; Initiate read
    NOP                       ; Executed (See Figure 10-2)
    NOP                       ; Ignored (See Figure 10-2)
    BSF      INTCON,GIE       ; Restore interrupts

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

10.5 Write Verify

It is considered good programming practice to verify that program memory writes agree with the intended value. Since program memory is stored as a full page then the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

FIGURE 10-8: FLASH PROGRAM MEMORY VERIFY FLOWCHART



10.6 Register Definitions: Flash Program Memory Control

REGISTER 10-1: PMDATL: PROGRAM MEMORY DATA LOW BYTE REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
PMDAT<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 **PMDAT<7:0>**: Read/write value for Least Significant bits of program memory

REGISTER 10-2: PMDATH: PROGRAM MEMORY DATA HIGH BYTE REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	PMDAT<13:8>					
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 **PMDAT<13:8>**: Read/write value for Most Significant bits of program memory

REGISTER 10-3: PMADRL: PROGRAM MEMORY ADDRESS LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
PMADR<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 **PMADR<7:0>**: Specifies the Least Significant bits for program memory address

REGISTER 10-4: PMADRH: PROGRAM MEMORY ADDRESS HIGH BYTE REGISTER

U-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
—(1)	PMADR<14:8>						
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 **Unimplemented:** Read as '1'

bit 6-0 **PMADR<14:8>**: Specifies the Most Significant bits for program memory address

Note 1: Unimplemented bit, read as '1'.

REGISTER 10-5: PMCON1: PROGRAM MEMORY CONTROL 1 REGISTER

U-1	R/W-0/0	R/W-0/0	R/W/HC-0/0	R/W/HC-x/q[2]	R/W-0/0	R/S/HC-0/0	R/S/HC-0/0
—[1]	CFGS	LWLO	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

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

bit 7 **Unimplemented:** Read as '1'

bit 6 **CFGS:** Configuration Select bit
- 1 = Access Configuration, User ID and Device ID Registers
- 0 = Access Flash program memory

bit 5 **LWLO:** Load Write Latches Only bit[3]
- 1 = Only the addressed program memory write latch is loaded/updated on the next WR command
- 0 = The addressed program memory write latch is loaded/updated and a write of all program memory write latches will be initiated on the next WR command

bit 4 **FREE:** Program Flash Erase Enable bit
- 1 = Performs an erase operation on the next WR command (hardware cleared upon completion)
- 0 = Performs an write operation on the next WR command

bit 3 **WRERR:** Program/Erase Error Flag bit
- 1 = Condition indicates an improper program or erase sequence attempt or termination (bit is set automatically on any set attempt (write '1') of the WR bit).
- 0 = The program or erase operation completed normally

bit 2 **WREN:** Program/Erase Enable bit
- 1 = Allows program/erase cycles
- 0 = Inhibits programming/erasing of program Flash

bit 1 **WR:** Write Control bit
- 1 = Initiates a program Flash program/erase operation.
  The operation is self-timed and the bit is cleared by hardware once operation is complete.
  The WR bit can only be set (not cleared) in software.
- 0 = Program/erase operation to the Flash is complete and inactive

bit 0 **RD:** Read Control bit
- 1 = Initiates a program Flash read. Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
- 0 = Does not initiate a program Flash read

Note 1: Unimplemented bit, read as '1'.
2: The WRERR bit is automatically set by hardware when a program memory write or erase operation is started (WR = 1).
3: The LWLO bit is ignored during a program memory erase operation (FREE = 1).

REGISTER 10-6: PMCON2: PROGRAM MEMORY CONTROL 2 REGISTER

W-0/0	W-0/0	W-0/0	W-0/0	W-0/0	W-0/0	W-0/0	W-0/0
Program Memory Control Register 2							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
S = Bit can only be set	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 Flash Memory Unlock Pattern bits

To unlock writes, a 55h must be written first, followed by an AAh, before setting the WR bit of the PMCON1 register. The value written to this register is used to unlock the writes. There are specific timing requirements on these writes.

TABLE 10-3: SUMMARY OF REGISTERS ASSOCIATED WITH FLASH PROGRAM MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
PMCON1	—(1)	CFGS	LWLO	FREE	WRERR	WREN	WR	RD	82
PMCON2	Program Memory Control Register 2								83
PMADRL	PMADRL<7:0>								81
PMADRH	—(1)	PMADRH<6:0>							81
PMDATL	PMDATL<7:0>								81
PMDATH	—	—	PMDATH<5:0>						81

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by Flash program memory module.
Note 1: Unimplemented, read as '1'.

TABLE 10-4: SUMMARY OF CONFIGURATION WORD WITH FLASH PROGRAM MEMORY

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	—	—	—	—	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		—	32
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		—	FOSC<1:0>		
CONFIG2	13:8	—	—	LVP	—	$\overline{\text{LPBOR}}$	BORV	STVREN	—	33
	7:0	—	—	—	—	—	—	WRT<1:0>		

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

11.0 I/O PORTS

Each port has three standard registers for its operation. These registers are:

- TRISx registers (data direction)
- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)

Some ports may have one or more of the following additional registers. These registers are:

- ANSELx (analog select)
- WPUx (weak pull-up)

In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output. However, the pin can still be read.

TABLE 11-1: PORT AVAILABILITY PER DEVICE

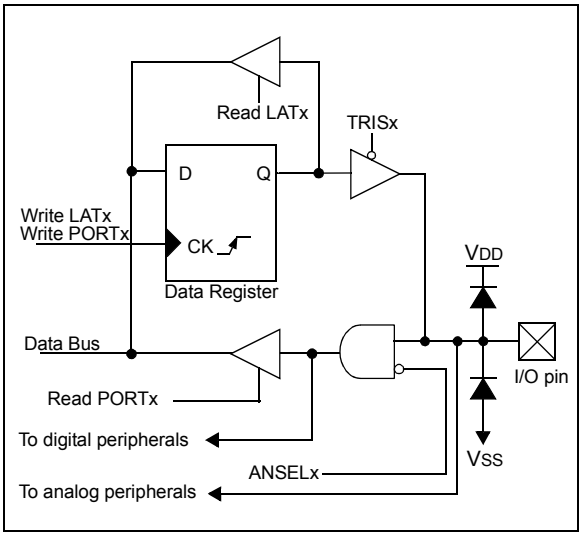
Device	PORTA
PIC12LF1552	•

The Data Latch (LATx registers) is useful for read-modify-write operations on the value that the I/O pins are driving.

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSEL bit is set, the digital input buffer associated with that bit is disabled. Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 11-1.

FIGURE 11-1: GENERIC I/O PORT OPERATION



EXAMPLE 11-1: INITIALIZING PORTA

```
; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL  PORTA       ;
CLRF     PORTA       ;Init PORTA
BANKSEL  LATA        ;Data Latch
CLRF     LATA        ;
BANKSEL  ANSELA      ;
CLRF     ANSELA      ;digital I/O
BANKSEL  TRISA       ;
MOVLW    B'00111000' ;Set RA<5:3> as inputs
MOVWF    TRISA       ;and set RA<2:0> as
                     ;outputs
```

11.1 Alternate Pin Function

The Alternate Pin Function Control (APFCON) register is used to steer specific peripheral input and output functions between different pins. The APFCON register is shown in Register 11-1. For this device family, the following functions can be moved between different pins.

- SDO
- $\overline{SS}$
- SDA/SDI

These bits have no effect on the values of any TRIS register. PORT and TRIS overrides will be routed to the correct pin. The unselected pin will be unaffected.

11.2 Register Definitions: Alternate Pin Function Control

REGISTER 11-1: APFCON: ALTERNATE PIN FUNCTION CONTROL REGISTER

R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	U-0	R/W-0/0	R/W-0/0
—	SDOSEL	SSSEL	SDSEL	—	—	—	—
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 **Unimplemented:** Read as ‘0’

bit 6 **SDOSEL:** Pin Selection bit
1 = SDO function is on RA4
0 = SDO function is on RA0

bit 5 **SSSEL:** Pin Selection bit
1 = $\overline{SS}$ function is on RA0
0 = $\overline{SS}$ function is on RA3

bit 4 **SDSEL:** Pin Selection bit
1 = SDA/SDI function is on RA3(1)
0 = SDA/SDI function is on RA2

bit 3-0 **Unimplemented:** Read as ‘0’

Note 1: The MSSP module has the ability to output low on RA3 when it is used as SDA/SDI.

11.3 PORTA Registers

11.3.1 DATA REGISTER

PORTA is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 11-3). Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). The exception is RA3, which is input-only and its TRIS bit will always read as '1'. Example 11-1 shows how to initialize an I/O port.

Reading the PORTA register (Register 11-2) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATA).

11.3.2 DIRECTION CONTROL

The TRISA register (Register 11-3) controls the PORTA pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISA register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

11.3.3 ANSELA REGISTER

The ANSELA register (Register 11-5) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELA bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

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

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

11.3.4 PORTA FUNCTIONS AND OUTPUT PRIORITIES

Each PORTA pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 11-2.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the highest priority.

Analog input functions, such as ADC inputs, are not shown in the priority lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown in Table 11-2.

TABLE 11-2: PORTA OUTPUT PRIORITY

Pin Name	Function Priority(1)
RA0	ICSPDAT SDO(2) $\overline{SS}$(3) RA0
RA1	SCL SCK RA1
RA2	ADOUT SDA(2) SDI(2) RA2
RA3	SDA(3) SDI(3) $\overline{SS}$(2) RA3
RA4	CLKOUT SDO(3) ADGRDA RA4
RA5	ADGRDB RA5

Note 1: Priority listed from highest to lowest.
2: Default pin (see APFCON register).
3: Alternate pin (see APFCON register).

11.4 Register Definitions: PORTA

REGISTER 11-2: PORTA: PORTA REGISTER

U-0	U-0	R/W-x/x	R/W-x/x	R-x/x	R/W-x/x	R/W-x/x	R/W-x/x
—	—	RA5	RA4	RA3	RA2	RA1	RA0
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 **RA<5:0>**: PORTA I/O Value bits(1)
1 = Port pin is ≥ VIH
0 = Port pin is ≤ VIL

Note 1: Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.

REGISTER 11-3: TRISA: PORTA TRI-STATE REGISTER

U-0	U-0	R/W-1/1	R/W-1/1	U-1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0
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 5-4 **TRISA<5:4>:** PORTA Tri-State Control bit
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

bit 3 **Unimplemented:** Read as ‘1’

bit 2-0 **TRISA<2:0>:** PORTA Tri-State Control bit
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

Note 1: Unimplemented, read as ‘1’.

REGISTER 11-4: LATA: PORTA DATA LATCH REGISTER

U-0	U-0	R/W-x/u	R/W-x/u	U-0	R/W-x/u	R/W-x/u	R/W-x/u
—	—	LATA5	LATA4	—	LATA2	LATA1	LATA0
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 5-4 **LATA<5:4>**: RA<5:4> Output Latch Value bits(1)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **LATA<2:0>**: RA<2:0> Output Latch Value bits(1)

Note 1: Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.

REGISTER 11-5: ANSELA: PORTA ANALOG SELECT REGISTER

U-0	U-0	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1	R/W-1/1
—	—	ANSA5	ANSA4	—	ANSA2	ANSA1	ANSA0
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 5-4 **ANSA<5:4>**: Analog Select between Analog or Digital Function on pins RA<5:4>, respectively
1 = Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **ANSA<2:0>**: Analog Select between Analog or Digital Function on pins RA<2:0>, respectively
1 = Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

Note 1: When setting a pin to an analog input, the corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.

REGISTER 11-6: WPUA: WEAK PULL-UP PORTA REGISTER(1,2)

U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
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 **WPUA<5:0>**: Weak Pull-up Register bits(3)
1 = Pull-up enabled
0 = Pull-up disabled

Note 1: Global $\overline{\text{WPUEN}}$ bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.
2: The weak pull-up device is automatically disabled if the pin is in configured as an output.
3: For the WPUA3 bit, when MCLRE = 1, weak pull-up is internally enabled, but not reported here.

TABLE 11-3: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	ANSA5	ANSA4	—	ANSA2	ANSA1	ANSA0	88
APFCON	—	SDOSEL	SSSEL	SDSEL	—	—	—	—	85
LATA	—	—	LATA5	LATA4	—	LATA2	LATA1	LATA0	88
OPTION_REG	$\overline{\text{WPUEN}}$	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>			132
PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	87
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	87
WPUA	—	—	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	89

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by PORTA.
Note 1: Unimplemented, read as '1'.

TABLE 11-4: SUMMARY OF CONFIGURATION WORD WITH PORTA

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	—	—	—	—	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		—	32
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		—	FOSC<1:0>		

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

12.0 INTERRUPT-ON-CHANGE

The PORTA pins can be configured to operate as Interrupt-On-Change (IOC) pins. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual port pin, or combination of port pins, can be configured to generate an interrupt. The interrupt-on-change module has the following features:

- Interrupt-on-Change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

Figure 12-1 is a block diagram of the IOC module.

12.1 Enabling the Module

To allow individual port pins to generate an interrupt, the IOCIE bit of the INTCON register must be set. If the IOCIE bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

12.2 Individual Pin Configuration

For each port pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated bit of the IOCxP register is set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting both associated bits of the IOCxP and IOCxN registers, respectively.

12.3 Interrupt Flags

The IOCAFx bits located in the IOCAF register, respectively, are status flags that correspond to the interrupt-on-change pins of the associated port. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit of the INTCON register reflects the status of all IOCAFx bits.

12.4 Clearing Interrupt Flags

The individual status flags, (IOCAFx bits), can be cleared by resetting them to zero. If another edge is detected during this clearing operation, the associated status flag will be set at the end of the sequence, regardless of the value actually being written.

In order to ensure that no detected edge is lost while clearing flags, only AND operations masking out known changed bits should be performed. The following sequence is an example of what should be performed.

EXAMPLE 12-1: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)

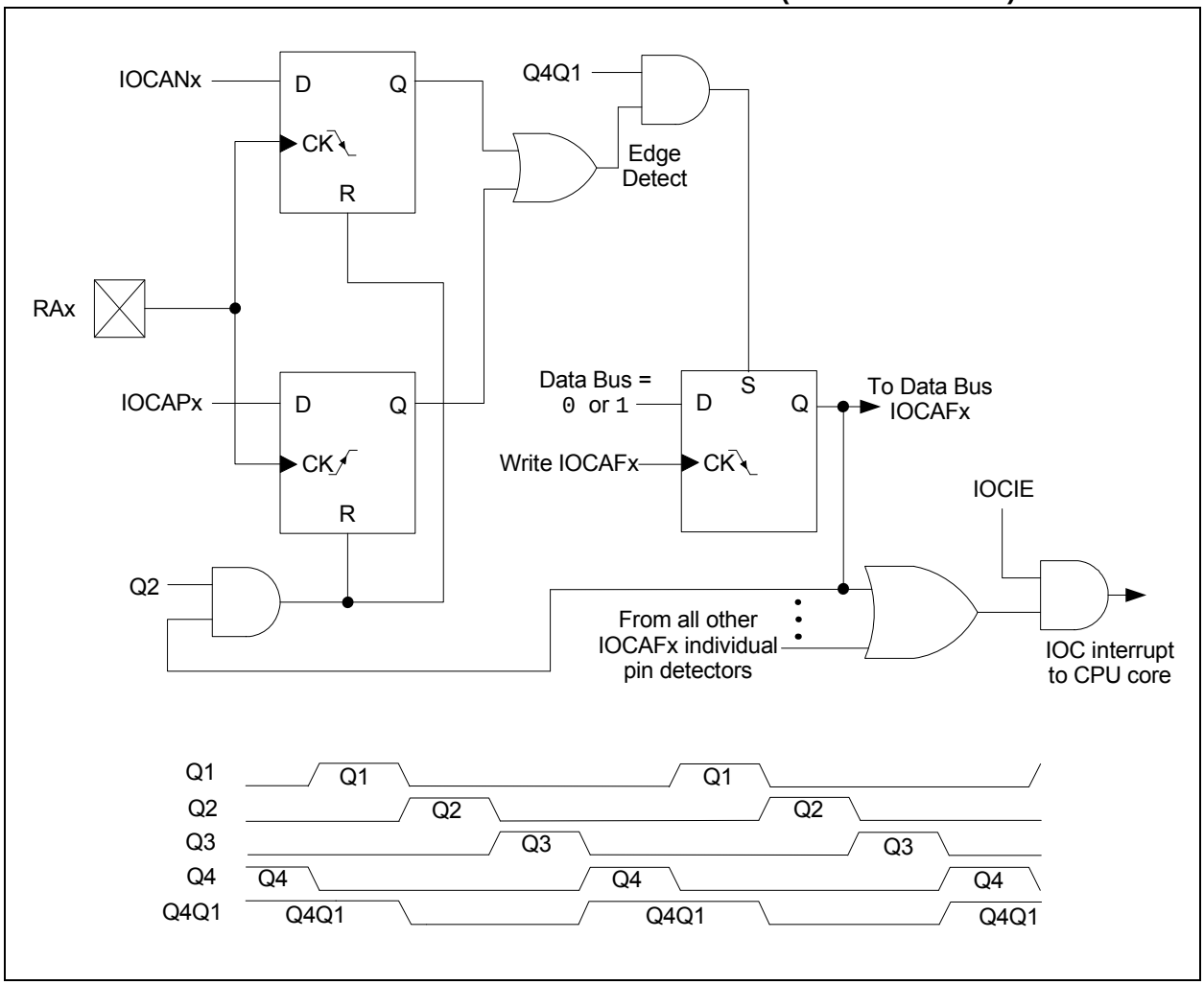
```
MOVLW  0xff
XORWF  IOCAF, W
ANDWF  IOCAF, F
```

12.5 Operation in Sleep

The interrupt-on-change interrupt sequence will wake the device from Sleep mode, if the IOCIE bit is set.

If an edge is detected while in Sleep mode, the IOCxF register will be updated prior to the first instruction executed out of Sleep.

FIGURE 12-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)



12.6 Register Definitions: Interrupt-on-Change Control

REGISTER 12-1: IOCAP: INTERRUPT-ON-CHANGE PORTA POSITIVE EDGE 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
—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
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 **IOCAP<5:0>:** Interrupt-on-Change PORTA Positive Edge Enable bits
1 = Interrupt-on-Change enabled on the pin for a positive going edge. IOCAFx bit and IOCIF flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin.

REGISTER 12-2: IOCAN: INTERRUPT-ON-CHANGE PORTA NEGATIVE EDGE 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
—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
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 **IOCAN<5:0>:** Interrupt-on-Change PORTA Negative Edge Enable bits
1 = Interrupt-on-Change enabled on the pin for a negative going edge. IOCAFx bit and IOCIF flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin.

REGISTER 12-3: IOCAF: INTERRUPT-ON-CHANGE PORTA FLAG REGISTER

U-0	U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
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 - Bit is set in hardware

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **IOCAF<5:0>:** Interrupt-on-Change PORTA Flag bits
1 = An enabled change was detected on the associated pin.
Set when IOCAPx = 1 and a rising edge was detected on RAx, or when IOCANx = 1 and a falling edge was detected on RAx.
0 = No change was detected, or the user cleared the detected change

TABLE 12-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	ANSA5	ANSA4	—	ANSA2	ANSA1	ANSA0	88
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
IOCAF	—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0	92
IOCAN	—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0	92
IOCAP	—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0	92
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	87

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.
Note 1: Unimplemented, read as '1'.

13.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of VDD, with 1.024V and 2.048V selectable output levels. The output of the FVR can be configured as the FVR input channel on the ADC.

The FVR can be enabled by setting the FVREN bit of the FVRCON register.

13.1 Independent Gain Amplifier

The output of the FVR supplied to the ADC is routed through a programmable gain amplifier. Each amplifier can be programmed for a gain of 1x or 2x, to produce the two possible voltage levels.

The ADFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference **Section 16.0 "Hardware Capacitive Voltage Divider (CVD) Module"** for additional information.

13.2 FVR Stabilization Period

When the Fixed Voltage Reference module is enabled, it requires time for the reference and amplifier circuits to stabilize. Once the circuits stabilize and are ready for use, the FVRRDY bit of the FVRCON register will be set. See **Section 21.0 "Electrical Specifications"** for the minimum delay requirement.

FIGURE 13-1: VOLTAGE REFERENCE BLOCK DIAGRAM

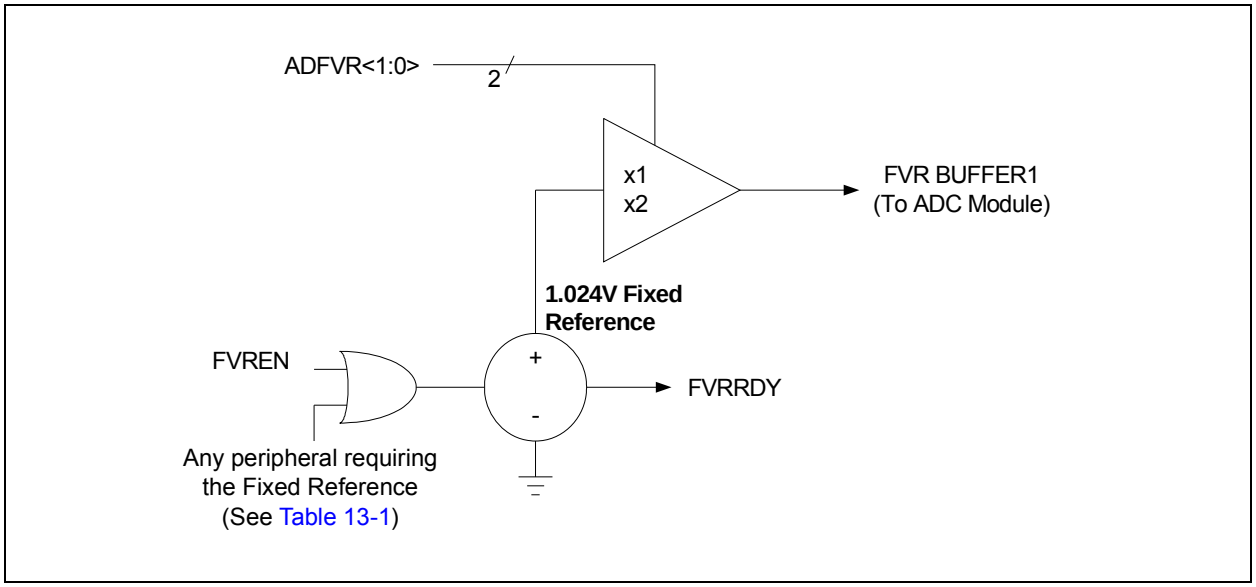


TABLE 13-1: PERIPHERALS REQUIRING THE FIXED VOLTAGE REFERENCE (FVR)

Peripheral	Conditions	Description
HFINTOSC	FOSC<1:0> = 00 and IRCF<3:0> = 000x	INTOSC is active and device is not in Sleep.
BOR	BOREN<1:0> = 11	BOR always enabled.
	BOREN<1:0> = 10 and BORFS = 1	BOR disabled in Sleep mode, BOR Fast Start enabled.
	BOREN<1:0> = 01 and BORFS = 1	BOR under software control, BOR Fast Start enabled.

13.3 Register Definitions: FVR Control

REGISTER 13-1: FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER

R/W-0/0	R-q/q	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
FVREN	FVRRDY(1)	TSEN	TSRNG	—	—	ADFVR<1:0>	
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	q = Value depends on condition

bit 7 **FVREN:** Fixed Voltage Reference Enable bit
1 = Fixed Voltage Reference is enabled
0 = Fixed Voltage Reference is disabled

bit 6 **FVRRDY:** Fixed Voltage Reference Ready Flag bit(1)
1 = Fixed Voltage Reference output is ready for use
0 = Fixed Voltage Reference output is not ready or not enabled

bit 5 **TSEN:** Temperature Indicator Enable bit(3)
1 = Temperature Indicator is enabled
0 = Temperature Indicator is disabled

bit 4 **TSRNG:** Temperature Indicator Range Selection bit(3)
1 = $V_{OUT} = V_{DD} - 4V_T$ (High Range)
0 = $V_{OUT} = V_{DD} - 2V_T$ (Low Range)

bit 3-2 **Unimplemented:** Read as '0'

bit 1-0 **ADFVR<1:0>:** ADC Fixed Voltage Reference Selection bit
11 = ADC Fixed Voltage Reference Peripheral output is off
10 = ADC Fixed Voltage Reference Peripheral output is 2x (2.048V)(2)
01 = ADC Fixed Voltage Reference Peripheral output is 1x (1.024V)
00 = ADC Fixed Voltage Reference Peripheral output is off

Note 1: FVRRDY is always '1' for the PIC12LF1552 devices.
2: Fixed Voltage Reference output cannot exceed V_{DD}.
3: See **Section 14.0 "Temperature Indicator Module"** for additional information.

TABLE 13-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE FIXED VOLTAGE REFERENCE

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	—	—	ADFVR<1:0>		95

Legend: Shaded cells are unused by the Fixed Voltage Reference module.

14.0 TEMPERATURE INDICATOR MODULE

This family of devices is equipped with a temperature circuit designed to measure the operating temperature of the silicon die. The circuit's range of operating temperature falls between -40°C and +85°C. The output is a voltage that is proportional to the device temperature. The output of the temperature indicator is internally connected to the device ADC.

The circuit may be used as a temperature threshold detector or a more accurate temperature indicator, depending on the level of calibration performed. A one-point calibration allows the circuit to indicate a temperature closely surrounding that point. A two-point calibration allows the circuit to sense the entire range of temperature more accurately. Reference Application Note AN1333, "*Use and Calibration of the Internal Temperature Indicator*" (DS01333) for more details regarding the calibration process.

14.1 Circuit Operation

Figure 14-1 shows a simplified block diagram of the temperature circuit. The proportional voltage output is achieved by measuring the forward voltage drop across multiple silicon junctions.

Equation 14-1 describes the output characteristics of the temperature indicator.

EQUATION 14-1: VOUT RANGES

High Range: $V_{OUT} = V_{DD} - 4V_T$

Low Range: $V_{OUT} = V_{DD} - 2V_T$

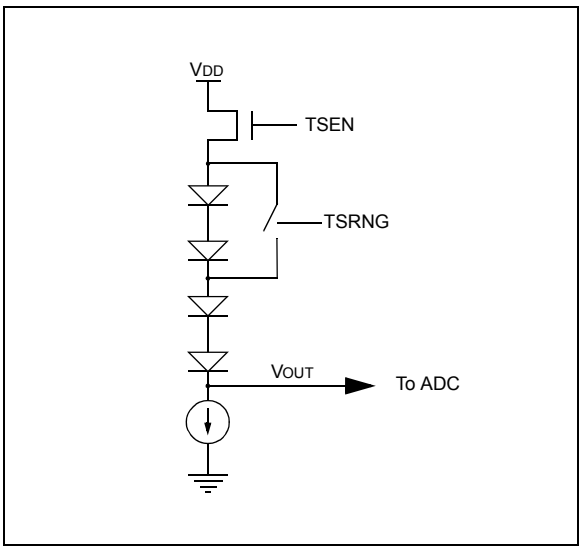
The temperature sense circuit is integrated with the Fixed Voltage Reference (FVR) module. See **Section 13.0 "Fixed Voltage Reference (FVR)"** for more information.

The circuit is enabled by setting the TSEN bit of the FVRCON register. When disabled, the circuit draws no current.

The circuit operates in either high or low range. The high range, selected by setting the TSRNG bit of the FVRCON register, provides a wider output voltage. This provides more resolution over the temperature range, but may be less consistent from part to part. This range requires a higher bias voltage to operate and thus, a higher VDD is needed.

The low range is selected by clearing the TSRNG bit of the FVRCON register. The low range generates a lower voltage drop and thus, a lower bias voltage is needed to operate the circuit. The low range is provided for low voltage operation.

FIGURE 14-1: TEMPERATURE CIRCUIT DIAGRAM



14.2 Minimum Operating VDD

When the temperature circuit is operated in low range, the device may be operated at any operating voltage that is within specifications.

When the temperature circuit is operated in high range, the device operating voltage, VDD, must be high enough to ensure that the temperature circuit is correctly biased.

Table 14-1 shows the recommended minimum VDD vs. range setting.

TABLE 14-1: RECOMMENDED VDD VS. RANGE

Min. VDD, TSRNG = 1	Min. VDD, TSRNG = 0
3.6V	1.8V

14.3 Temperature Output

The output of the circuit is measured using the internal Analog-to-Digital Converter. A channel is reserved for the temperature circuit output. Refer to **Section 16.0 "Hardware Capacitive Voltage Divider (CVD) Module"** for detailed information.

14.4 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait at least 200 µs after the ADC input multiplexer is connected to the temperature indicator output before the conversion is performed. In addition, the user must wait 200 µs between sequential conversions of the temperature indicator output.

TABLE 14-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR

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	—		ADFVR<1:0>		118

Legend: Shaded cells are unused by the temperature indicator module.

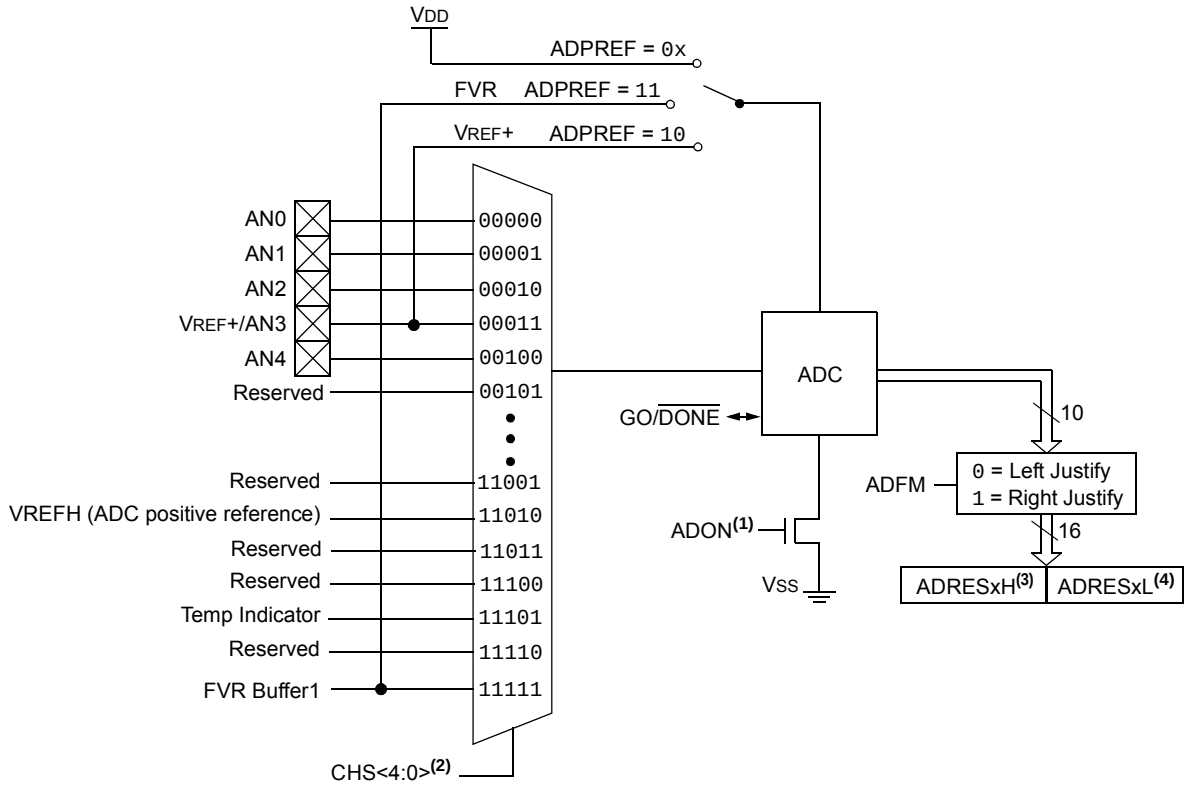
15.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair). Figure 15-1 shows the block diagram of the ADC.

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

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

FIGURE 15-1: ADC BLOCK DIAGRAM



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

2: See AADCON0 register (Register 16-1) for detailed analog channel selection per device.

3: ADRES0H and AADRES0H are the same register in two locations, Bank 1 and Bank 14. See Table 3-3.

4: ADRES0L and AADRES0L are the same register in two locations, Bank 1 and Bank 14. See Table 3-3.

15.1 ADC Configuration

When configuring and using the ADC, the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting

15.1.1 PORT CONFIGURATION

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to **Section 11.0 "I/O Ports"** for more information.

Note:	Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

15.1.2 CHANNEL SELECTION

There are up to eight channel selections available:

- AN<4:0> pins
- VREF+ (ADC positive reference)
- Temperature Indicator
- FVR (Fixed Voltage Reference) Output

Refer to **Section 13.0 "Fixed Voltage Reference (FVR)"** and **Section 14.0 "Temperature Indicator Module"** for more information on these channel selections.

The CHS bits of the ADCON0 register determine which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion. Refer to **Section 16.1 "Hardware CVD Operation"** for more information.

15.1.3 ADC VOLTAGE REFERENCE

The ADPREF bits of the ADCON1 register provides control of the positive voltage reference. The positive voltage reference can be:

- VREF+ pin
- VDD
- FVR (Fixed Voltage Reference)

See **Section 13.0 "Fixed Voltage Reference (FVR)"** for more details on the fixed voltage reference.

15.1.4 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCS bits of the ADCON1 register. There are seven possible clock options:

- FOSC/2
- FOSC/4
- FOSC/8
- FOSC/16
- FOSC/32
- FOSC/64
- FRC (dedicated internal oscillator)

The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11.5 TAD periods as shown in Figure 15-2.

For correct conversion, the appropriate TAD specification must be met. Refer to the ADC conversion requirements in **Section 21.0 "Electrical Specifications"** for more information. Table 15-1 gives examples of appropriate ADC clock selections.

Note:	Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.

TABLE 15-1: ADC CLOCK PERIOD (TAD) VS. DEVICE OPERATING FREQUENCIES

ADC Clock Period (TAD)		Device Frequency (FOSC)				
ADC Clock Source	ADCS<2:0>	20 MHz	16 MHz	8 MHz	4 MHz	1 MHz
Fosc/2	000	100 ns(2)	125 ns(2)	250 ns(2)	500 ns(2)	2.0 μs
Fosc/4	100	200 ns(2)	250 ns(2)	500 ns(2)	1.0 μs	4.0 μs
Fosc/8	001	400 ns(2)	0.5 μs(2)	1.0 μs	2.0 μs	8.0 μs(3)
Fosc/16	101	800 ns	1.0 μs	2.0 μs	4.0 μs	16.0 μs(3)
Fosc/32	010	1.6 μs	2.0 μs	4.0 μs	8.0 μs(3)	32.0 μs(3)
Fosc/64	110	3.2 μs	4.0 μs	8.0 μs(3)	16.0 μs(3)	64.0 μs(3)
FRC	x11	1.0-6.0 μs(1,4)	1.0-6.0 μs(1,4)	1.0-6.0 μs(1,4)	1.0-6.0 μs(1,4)	1.0-6.0 μs(1,4)

Legend: Shaded cells are outside of recommended range.

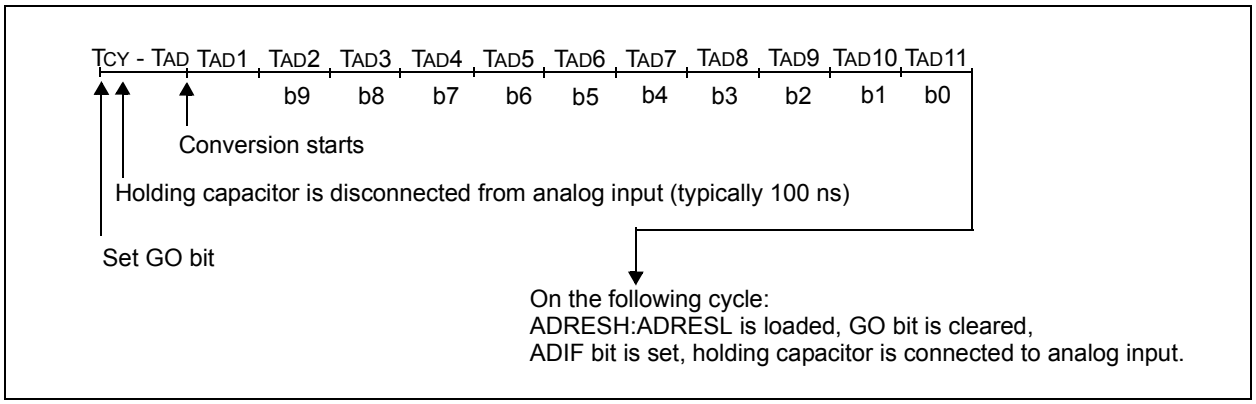
Note 1: The FRC source has a typical TAD time of 1.6 μs for VDD.

2: These values violate the minimum required TAD time.

3: For faster conversion times, the selection of another clock source is recommended.

4: The ADC clock period (TAD) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the FRC clock source must be used when conversions are to be performed with the device in Sleep mode.

FIGURE 15-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES



15.1.5 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADC Interrupt Flag is the ADIF bit in the PIR1 register. The ADC Interrupt Enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared in software.

Note 1:	The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.
2:	The ADC operates during Sleep only when the FRC oscillator is selected.

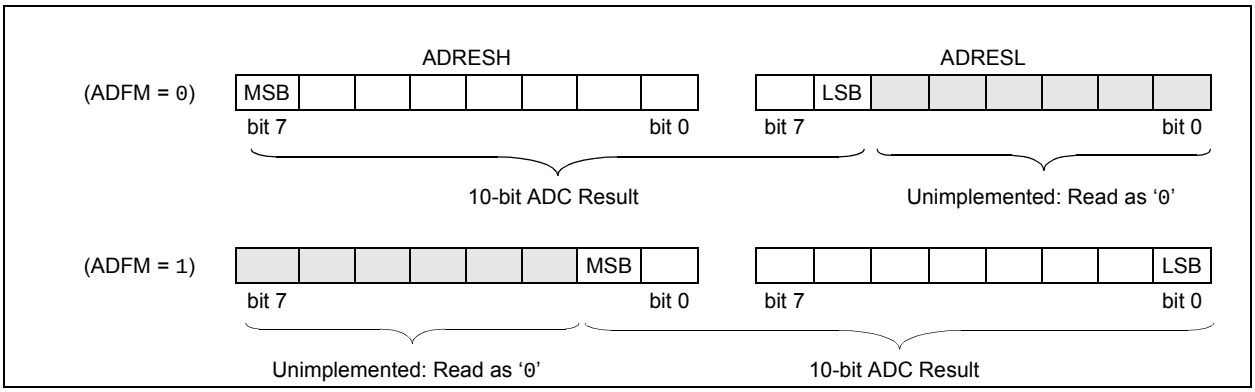
This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake-up the device. Upon waking from Sleep, the next instruction following the SLEEP instruction is always executed. If the user is attempting to wake-up from Sleep and resume in-line code execution, the GIE and PEIE bits of the INTCON register must be disabled. If the GIE and PEIE bits of the INTCON register are enabled, execution will switch to the Interrupt Service Routine.

15.1.6 RESULT FORMATTING

The 10-bit ADC conversion result can be supplied in two formats, left justified or right justified. The ADFM bit of the ADCON1 register controls the output format.

Figure 15-3 shows the two output formats.

FIGURE 15-3: 10-BIT ADC CONVERSION RESULT FORMAT



15.2 ADC Operation

15.2.1 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. Setting the GO/DONE bit of the ADCON0 register to a '1' will start the Analog-to-Digital conversion.

Note:	The GO/DONE bit should not be set in the same instruction that turns on the ADC. Refer to **Section 15.2.6 "ADC Conversion Procedure"**.

15.2.2 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/DONE bit
- Set the ADIF Interrupt Flag bit
- Update the ADRESH and ADRESL registers with new conversion result

15.2.3 TERMINATING A CONVERSION

If a conversion must be terminated before completion, the GO/DONE bit can be cleared in software. The ADRESH and ADRESL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.

Note:	A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

15.2.4 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC clock source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the ADC module is turned off after the conversion completes, although the ADON bit remains set.

When the ADC clock source is something other than FRC, a SLEEP instruction causes the present conversion to be aborted and the ADC module is turned off, although the ADON bit remains set.

15.2.5 SPECIAL EVENT TRIGGER

The Special Event Trigger allows periodic ADC measurements without software intervention, using the TRIGSEL bits of the AADCON2 register. When this trigger occurs, the GO/DONE bit is set by hardware from the Timer0 Overflow.

TABLE 15-2: SPECIAL EVENT TRIGGER

Device	Source
PIC12LF1552	TMR0

Using the Special Event Trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met.

Refer to **Section 17.0 "Timer0 Module"** for more information.

15.2.6 ADC CONVERSION PROCEDURE

This is an example procedure for using the ADC to perform an Analog-to-Digital conversion:

1. Configure Port:
   - Disable pin output driver (Refer to the TRIS register)
   - Configure pin as analog (Refer to the ANSEL register)
   - Disable weak pull-ups either globally (Refer to the OPTION_REG register) or individually (Refer to the appropriate WPUx register).
2. Configure the ADC module:
   - Select ADC conversion clock
   - Configure voltage reference
   - Select ADC input channel
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt
   - Enable global interrupt(1)
4. Wait the required acquisition time(2).
5. Start conversion by setting the GO/$\overline{\text{DONE}}$ bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the GO/$\overline{\text{DONE}}$ bit
   - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result in ADRES0H and ADRES0L.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

2: Refer to **Section 15.4 "ADC Acquisition Requirements"**.

EXAMPLE 15-1: ADC CONVERSION

```
;This code block configures the ADC
;for polling, Vdd and Vss references, Frc
;clock and AN0 input.
;
;Conversion start & polling for completion
; are included.
;
BANKSEL   ADCON1       ;
MOVLW     B'11110000'  ;Right justify, Frc
                       ;clock
MOVWF     ADCON1       ;Vdd and Vss Vref
BANKSEL   TRISA        ;
BSF       TRISA,0      ;Set RA0 to input
BANKSEL   ANSEL        ;
BSF       ANSEL,0      ;Set RA0 to analog
BANKSEL   WPUA
BCF       WPUA,0        ;Disable RA0 weak
                         pull-up
BANKSEL   ADCON0       ;
MOVLW     B'00000001'  ;Select channel AN0
MOVWF     ADCON0       ;Turn ADC On
CALL      SampleTime   ;Acquisiton delay
BSF       ADCON0,ADGO  ;Start conversion
BTFSC     ADCON0,ADGO  ;Is conversion done?
GOTO      $-1          ;No, test again
BANKSEL   ADRES0H      ;
MOVF      ADRES0H,W    ;Read upper 2 bits
MOVWF     RESULTHI     ;store in GPR space
```

15.3 ADC Register Definitions

The following registers are used to control the operation of the ADC.

REGISTER 15-1: ADCON0: ADC CONTROL REGISTER 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	R/W-0/0
—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON
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 **Unimplemented:** Read as ‘0’

bit 6-2 **CHS<4:0>:** Analog Channel Select bits
- 11111 = FVR (Fixed Voltage Reference) Buffer 1 Output(1)
- 11110 = Reserved. No channel connected.
- 11101 = Temperature Indicator(2).
- 11100 = Reserved. No channel connected.
- 11011 = Reserved. No channel connected.
- 11010 = VREFH (ADC Positive Reference)
- 11001 = Reserved. No channel connected.
- •
- •
- •
- 00101 = Reserved. No channel connected.
- 00100 = AN4
- 00011 = AN3
- 00010 = AN2
- 00001 = AN1
- 00000 = AN0

bit 1 **GO/$\overline{\text{DONE}}$:** ADC Conversion Status bit
- 1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. This bit is automatically cleared by hardware when the ADC conversion has completed.
- 0 = ADC conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit
- 1 = ADC is enabled
- 0 = ADC is disabled and consumes no operating current

Note 1: See **Section 13.0 “Fixed Voltage Reference (FVR)”** for more information.
2: See **Section 14.0 “Temperature Indicator Module”** for more information.

REGISTER 15-2: ADCON1: ADC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
ADFM	ADCS<2:0>			—	—	ADPREF<1:0>	
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 **ADFM:** ADC Result Format Select bit
1 = Right justified. Six Most Significant bits of ADRESH are set to ‘0’ when the conversion result is loaded.
0 = Left justified. Six Least Significant bits of ADRESL are set to ‘0’ when the conversion result is loaded.

bit 6-4 **ADCS<2:0>:** ADC Conversion Clock Select bits
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
011 = FRC (clock supplied from a dedicated RC oscillator)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64
111 = FRC (clock supplied from a dedicated RC oscillator)

bit 3-2 **Unimplemented:** Read as ‘0’

bit 1-0 **ADPREF<1:0>:** ADC Positive Voltage Reference Configuration bits
00 = VREF is connected to VDD
01 = Reserved
10 = VREF is connected to external VREF+ pin(1)
11 = VREF is connected to internal Fixed Voltage Reference (FVR) module(1)

Note 1: When selecting the FVR or the VREF+ pin as the source of the positive reference, be aware that a minimum voltage specification exists. See **Section 21.0 “Electrical Specifications”** for details.

REGISTER 15-3: ADRES0H: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 0

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADRES<9:2>							
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 **ADRES<9:2>**: ADC Result Register bits
Upper eight bits of 10-bit conversion result

REGISTER 15-4: ADRES0L: ADC RESULT REGISTER LOW (ADRESL) ADFM = 0

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADRES<1:0>		—	—	—	—	—	—
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 **ADRES<1:0>**: ADC Result Register bits
Lower two bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

REGISTER 15-5: ADRES0H: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	—	—	—	—	ADRES<9:8>	
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 7-2 **Reserved**: Do not use.

bit 1-0 **ADRES<9:8>**: ADC Result Register bits
Upper two bits of 10-bit conversion result

REGISTER 15-6: ADRES0L: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADRES<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 **ADRES<7:0>**: ADC Result Register bits
Lower eight bits of 10-bit conversion result

15.4 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (C_{HOLD}) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 15-4. The source impedance (R_S) and the internal sampling switch (R_{SS}) impedance directly affect the time required to charge the capacitor C_{HOLD}. The sampling switch (R_{SS}) impedance varies over the device voltage (V_{DD}), refer to Figure 15-4. **The maximum recommended impedance for analog sources is 10 kΩ.** As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be done before the conversion can be started. To calculate the minimum acquisition time, Equation 15-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

EQUATION 15-1: ACQUISITION TIME EXAMPLE

Assumptions: $Temperature = 50°C$ *and external impedance of* $10k\Omega$ *3.3V* V_{DD}

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$
$$= 2\mu s + T_C + [(Temperature - 25°C)(0.05\mu s/°C)]$$

The value for T_C *can be approximated with the following equations:*

$$V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) = V_{CHOLD} \quad ;[1]\ V_{CHOLD}\ \text{charged to within 1/2 LSB}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD} \quad ;[2]\ V_{CHOLD}\ \text{charge response to}\ V_{APPLIED}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) \quad ;\text{combining [1] and [2]}$$

Note: Where n = number of bits of the ADC.

Solving for T_C*:*

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln(1/511)$$
$$= -10pF(1k\Omega + 7k\Omega + 10k\Omega)\ ln(0.001957)$$
$$= 1.12\mu s$$

Therefore:

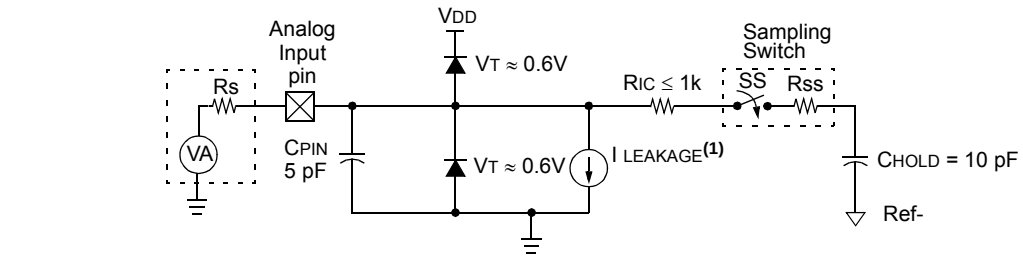
$$T_{ACQ} = 2\mu s + 1.12\mu s + [(50°C - 25°C)(0.05\mu s/°C)]$$
$$= 4.42\mu s$$

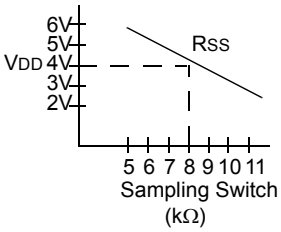
Note 1: The reference voltage (V_{REF}) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (C_{HOLD}) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

FIGURE 15-4: ANALOG INPUT MODEL





Legend:

CHOLD	=	Sample/Hold Capacitance
CPIN	=	Input Capacitance
I LEAKAGE	=	Leakage current at the pin due to various junctions
RIC	=	Interconnect Resistance
RSS	=	Resistance of Sampling Switch
SS	=	Sampling Switch
VT	=	Threshold Voltage

Note 1: Refer to **Section 21.0 "Electrical Specifications"**.

FIGURE 15-5: ADC TRANSFER FUNCTION

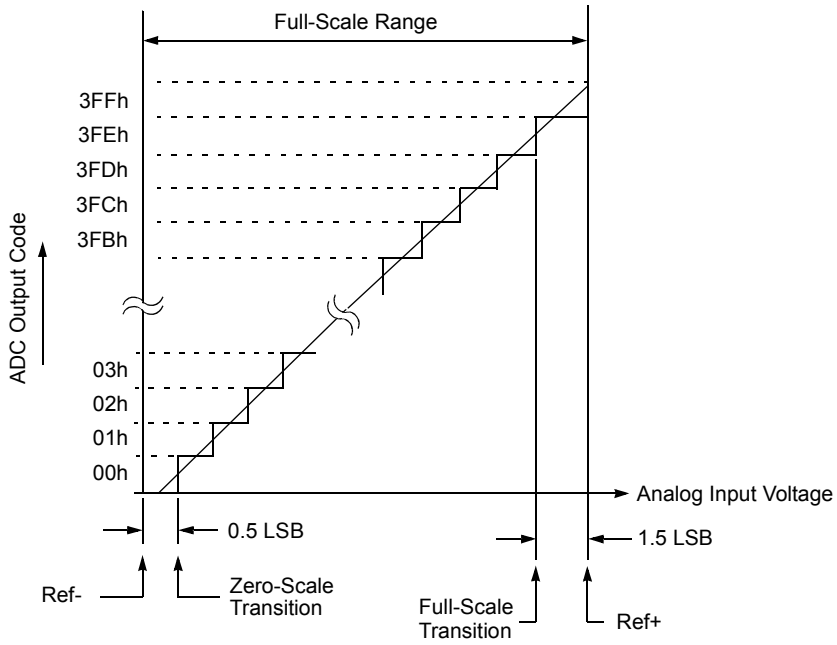


TABLE 15-3: SUMMARY OF REGISTERS ASSOCIATED WITH ADC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ADCON0	—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON	104
ADCON1	ADFM	ADCS<2:0>			—	—	ADPREF<1:0>		105
ADRES0H	ADC Result Register High								106, 107
ADRES0L	ADC Result Register Low								106, 107
ANSELA	—	—	ANSA5	ANSA4	—	ANSA2	ANSA1	ANSA0	88
FVRCON	FVREN	FVRRDY	TSEN	TSRNG	—	—	ADFVR<1:0>		95
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
PIE1	—	ADIE	—	—	SSPIE	—	—	—	58
PIR1	—	ADIF	—	—	SSPIF	—	—	—	60
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	87

Legend: — = unimplemented read as '0'. Shaded cells are not used for ADC module.
Note 1: Unimplemented, read as '1'.

16.0 HARDWARE CAPACITIVE VOLTAGE DIVIDER (CVD) MODULE

The hardware Capacitive Voltage Divider (CVD) module is a peripheral, which allows the user to perform a relative capacitance measurement on any ADC channel using the internal ADC sample and hold capacitance as a reference. This relative capacitance measurement can be used to implement capacitive touch or proximity sensing applications.

The CVD operation begins with the ADC's internal sample and hold capacitor (CHOLD) being disconnected from the path which connects it to the external capacitive sensor node. While disconnected, CHOLD is pre-charged to VDD or VSS, while the path to the sensor node is also discharged to VDD or VSS. Typically, this node is discharged to the level opposite that of CHOLD. When the pre-charge phase is complete, the VDD/VSS bias paths for the two nodes are shut off and CHOLD and the path to the external sensor node are re-connected, at which time the acquisition phase of the CVD operation begins. During acquisition, a capacitive voltage divider is formed between the pre-charged CHOLD the and sensor nodes, which results in a final voltage level settling on CHOLD, which is determined by the capacitances and pre-charge levels of the two nodes involved. After acquisition, the ADC converts the voltage level held on CHOLD. This process is then usually repeated with the selected pre-charge levels for both the CHOLD and the inverted sensor nodes. Figure 16-1 shows the waveform for two inverted CVD measurements, which is also known is differential CVD measurement.

In a typical application, an Analog-to-Digital Converter (ADC) channel is attached to a pad on a Printed Circuit Board (PCB), which is electrically isolated from the end user. A capacitive change is detected on the ADC channel using the CVD conversion method when the end user places a finger over the PCB pad, the developer then can implement software to detect a touch or proximity event. Key features of this module include:

- Automated double sample conversions
- Two result registers
- Inversion of second sample
- 7-bit pre-charge timer
- 7-bit acquisition timer
- Two guard ring output drives
- Adjustable sample and hold capacitor array

Note: For more information on capacitive voltage divider sensing method refer to the Application Note AN1478, *"mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider"* (DS01478).

FIGURE 16-1: DIFFERENTIAL CVD MEASUREMENT WAVEFORM

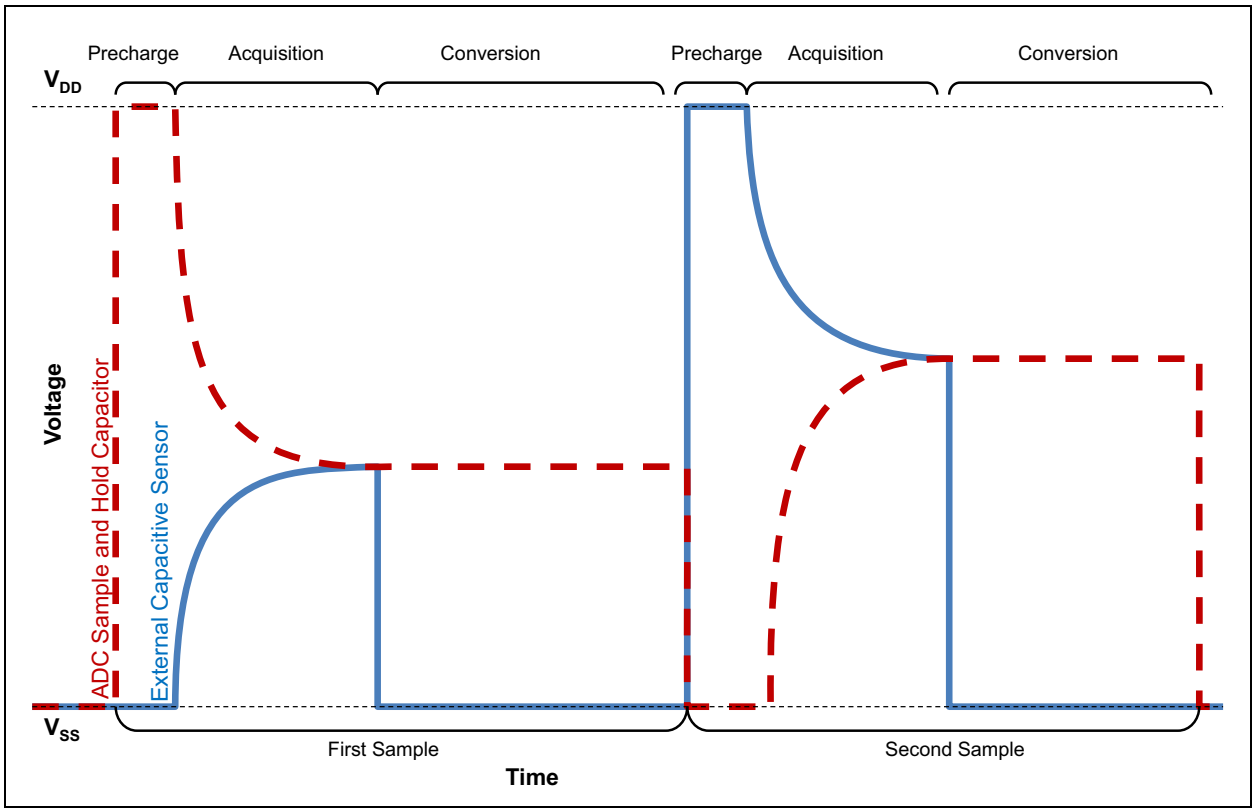
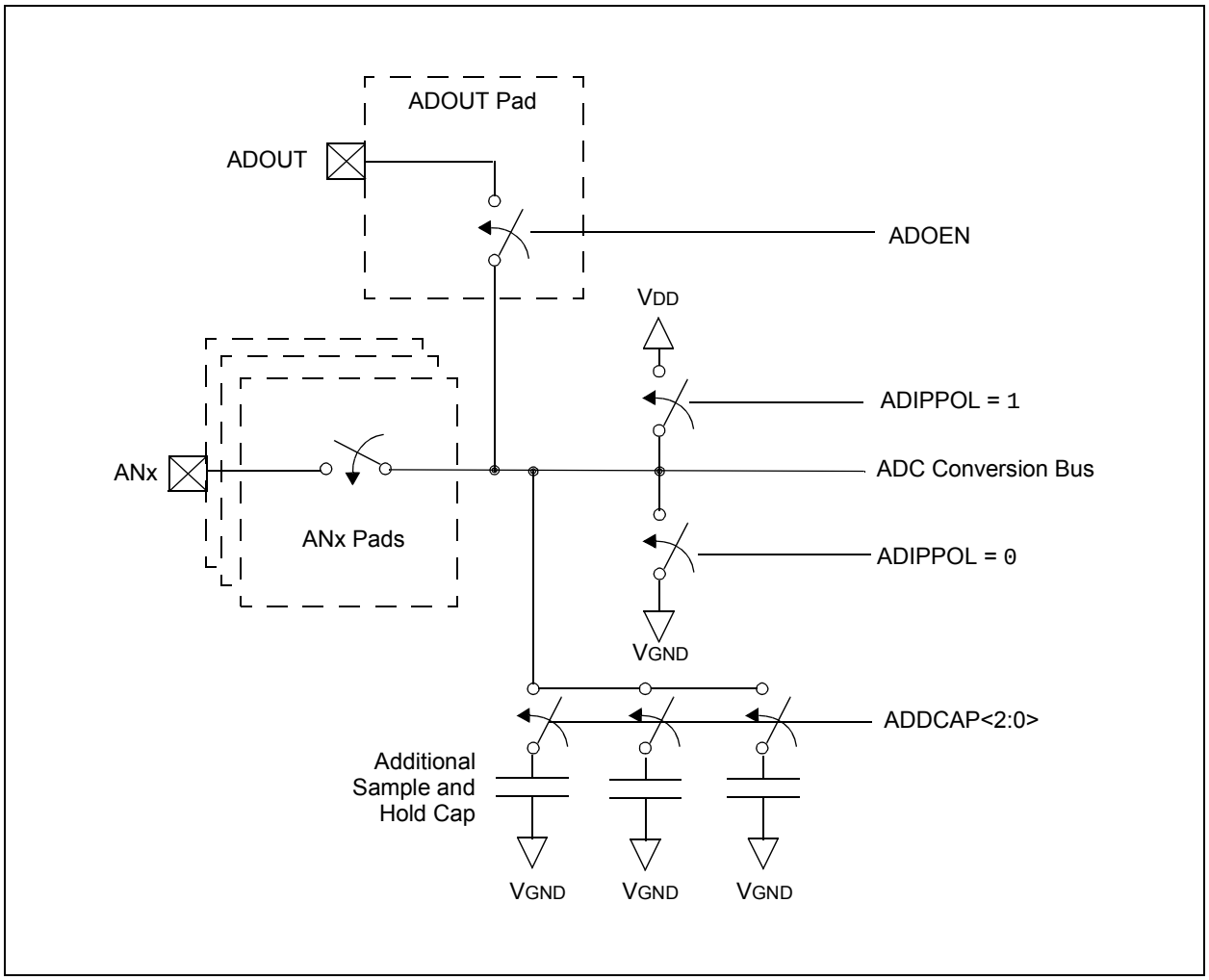


FIGURE 16-2: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM



16.1 Hardware CVD Operation

Capacitive Voltage Divider is a charge averaging capacitive sensing method. The hardware CVD module will automate the process of charging, averaging between the external sensor and the internal ADC sample and hold capacitor, and then initiating the ADC conversions. The whole process can be expanded into three stages: pre-charge, acquisition, and conversion. See Figure 16-5 for basic information on the timing of three stages.

16.1.1 PRE-CHARGE TIMER

The pre-charge stage is an optional 1-127 instruction cycle time delay used to put the external ADC channel and the internal sample and hold capacitor (CHOLD) into pre-conditioned states. The pre-charge stage of conversion is enabled by writing a non-zero value to the ADPRE<6:0> bits of the AADPRE register. This stage is initiated when a conversion sequence is started by either the GO/$\overline{\text{DONE}}$ bit or a Special Event Trigger. When initiating an ADC conversion, if the ADPRE bits are cleared, this stage is skipped.

During the pre-charge time, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the ADIPPOL bit of the AADCON3 register. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to pre-charge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is determined by the ADEPPOL bit of the AADCON3 register.

When both the ADOEN and ADOOEN bits of the AADCON3 register are set, the ADOUT pin is overridden during pre-charge. See **Section 16.1.9 "Analog Bus Visibility"** for more information. This override functions the same as the channel pin overrides, but the polarity is selected by the ADIPPOL bit of the AADCON3 register. See Figure 16-2.

Even though the analog channel of the pin is selected, the analog multiplexer is forced open during the pre-charge stage. The ADC multiplex or logic is overridden and disabled only during the pre-charge time.

16.1.2 ACQUISITION TIMER

The acquisition timer controls the time allowed to acquire the signal to be sampled. The acquisition delay time is from 1 to 127 instruction cycles and is used to allow the voltage on the internal sample and hold capacitor (CHOLD) to settle to a final value through charge averaging. The acquisition time of conversion is enabled by writing a non-zero value to the ADACQ<6:0> bits of the AADACQ register. When the acquisition time is enabled, the time starts immediately following the pre-charge stage. If the ADPRE<6:0> bits of the AADPRE register are set to zero, the acquisition time is initiated by either setting the GO/$\overline{\text{DONE}}$ bit or a Special Event Trigger.

At the start of the acquisition stage, the port pin logic of the selected analog channel is again overridden to turn off the digital high/low output drivers so that they do not affect the final result of charge averaging. Also, the selected ADC channel is connected to CHOLD. This allows charge averaging to proceed between the pre-charged channel and the CHOLD capacitor. It is noted that the port pin logic override that occurs during acquisition related to the selected sample channel does not occur on the ADOUT pin (see **Section 16.1.9 "Analog Bus Visibility"**) for more information.

16.1.3 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the AADCON0 register must be set. Setting the GO/$\overline{\text{DONE}}$ bit of the AADCON0 register or by the Special Event Trigger inputs will start the Analog-to-Digital conversion.

Once a conversion begins, it proceeds until complete, while the ADON bit is set. If the ADON bit is cleared, the conversion is halted. The GO/$\overline{\text{DONE}}$ bit of the AADCON0 register indicates that a conversion is occurring, regardless of the starting trigger.

Note:	The GO/$\overline{\text{DONE}}$ bit should not be set in the same instruction that turns on the ADC. Refer to Section **Section 16.1.10 "Hardware CVD Double Conversion Procedure"**

16.1.4 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/$\overline{\text{DONE}}$ bit of the AADCON0 register.
- Set the ADIF Interrupt Flag bit of the PIR1 register.
- Update the AADRESxH and AADRESxL registers with new conversion results.

16.1.5 TERMINATING A CONVERSION

If a conversion must be terminated before completion, clear the GO/$\overline{\text{DONE}}$ bit. The AADRESxH and AADRESxL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.

The AADSTAT register can be used to track the status of the hardware CVD module during a conversion.

Note:	A device Reset forces all registers to their reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

16.1.6 DOUBLE SAMPLE CONVERSION

Double sampling can be enabled by setting the ADDSEN bit of the AADCON3 register. When this bit is set, two conversions are completed each time the GO/DONE bit is set or a Special Event Trigger occurs. The GO/DONE bit remains set for the duration of both conversions and is used to signal the end of the conversion.

Without setting the ADIPEN bit, the double conversion will have identical charge/discharge on the internal and external capacitor for these two conversions. Setting the ADIPEN bit prior to a double conversion will allow the user to perform a pseudo-differential CVD measurement by subtracting the results from the double conversion. This is highly recommended for noise immunity purposes.

The result of the first conversion is written to the AADRES0H and AADRES0L registers. The second conversion starts two clock cycles after the first has completed, while the GO/DONE bit remains set. When the ADIPEN bit of AADCON3 is set, the value used by the ADC for the ADEPPOL, ADIPPOL, and GRDPOL bits are inverted. The value stored in those bit locations is unchanged. All other control signals remain unchanged from the first conversion. The result of the second conversion is stored in the AADRES1H and AADRES1L registers. See Figure 16-4 and Figure 16-5 for more information.

16.1.7 GUARD RING OUTPUTS

The guard ring outputs consist of a pair of digital outputs from the hardware CVD module. This function is enabled by the GRDAOE and GRDBOE bits of the AADGRD register. Polarity of the output is controlled by the GRDPOL bit.

Once enabled and while ADON = 1, the guard ring outputs of the ADC are active at all times. The outputs are initialized at the start of the pre-charge stage to match the polarity of the GRDPOL bit. The guard output signal changes polarity at the start of the acquisition phase. The value stored by the GRDPOL bit does not change. When in double sampling mode, the ring output levels are inverted during the second pre-charge and acquisition phases if ADDSEN = 1 and ADIPEN = 1. For more information on the timing of the guard ring output, refer to Figure 16-4 and Figure 16-5.

A typical guard ring circuit is displayed in Figure 16-2. CGUARD represents the capacitance of the guard ring trace placed on a PCB board. The user selects values for RA and RB that will create a voltage profile on CGUARD, which will match the selected channel during acquisition.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, *"mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider"* (DS01478).

FIGURE 16-3: GUARD RING CIRCUIT

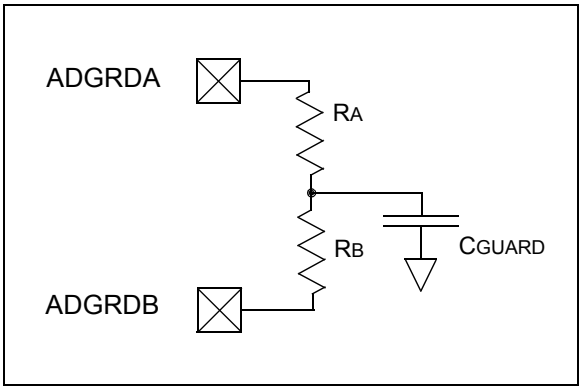
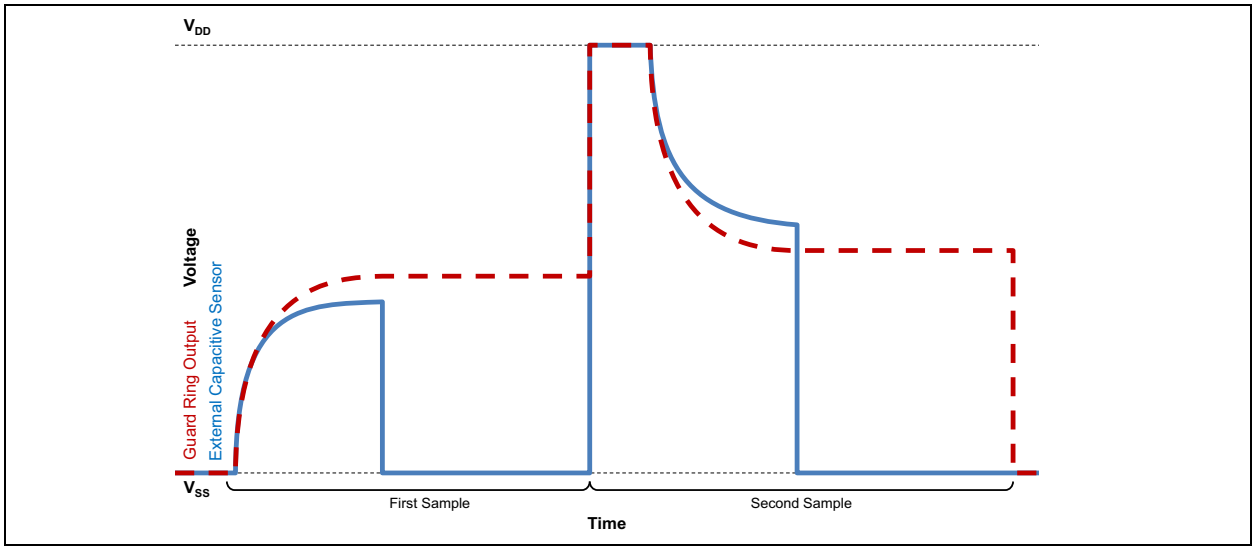


FIGURE 16-4: DIFFERENTIAL CVD WITH GUARD RING OUTPUT WAVEFORM



16.1.8 ADDITIONAL SAMPLE AND HOLD CAPACITOR

Additional capacitance can be added in parallel with the sample and hold capacitor (CHOLD) by setting the ADDCAP<2:0> bits of the AADCAP register. This bit connects a digitally programmable capacitance to the ADC conversion bus, increasing the effective internal capacitance of the sample and hold capacitor in the ADC module. This is used to improve the match between internal and external capacitance for a better sensing performance. The additional capacitance does not affect analog performance of the ADC because it is not connected during conversion. See Figure 16-1.

16.1.9 ANALOG BUS VISIBILITY

The ADOEN of the AADCON3 register can be used to connect the ADC conversion bus to the ADOUT pin. This connection can be used to monitor the state and behavior of the internal analog bus and it also can be used to improve the match between internal and external capacitance by connecting a external capacitor to increase the effective internal capacitance. The ADOEN bit provides the connection via a standard channel pass gate.

The ADOUT pin function can be overridden during the pre-charge stage of conversion. This override function is controlled by ADOOEN. The polarity of the override is set by the ADIPPOL bit. It should be noted that, outside of the pre-charge phase, no ADOUT override is in effect. Therefore, the user must manage the state of the ADOUT pin via the relevant TRIS bit in order to avoid unintended affects on conversion results. If the user wishes to have the ADOUT path active during conversions, then the relevant TRIS bit should be set to ensure that the ADOUT pin logic is in the Input mode during the acquisition phase of conversions.

FIGURE 16-5: HARDWARE CVD SEQUENCE TIMING DIAGRAM

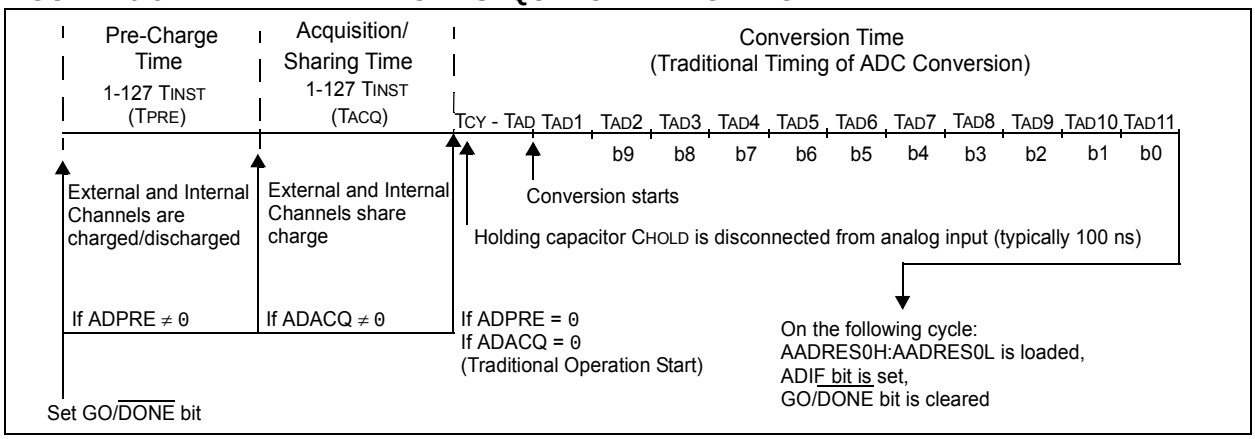
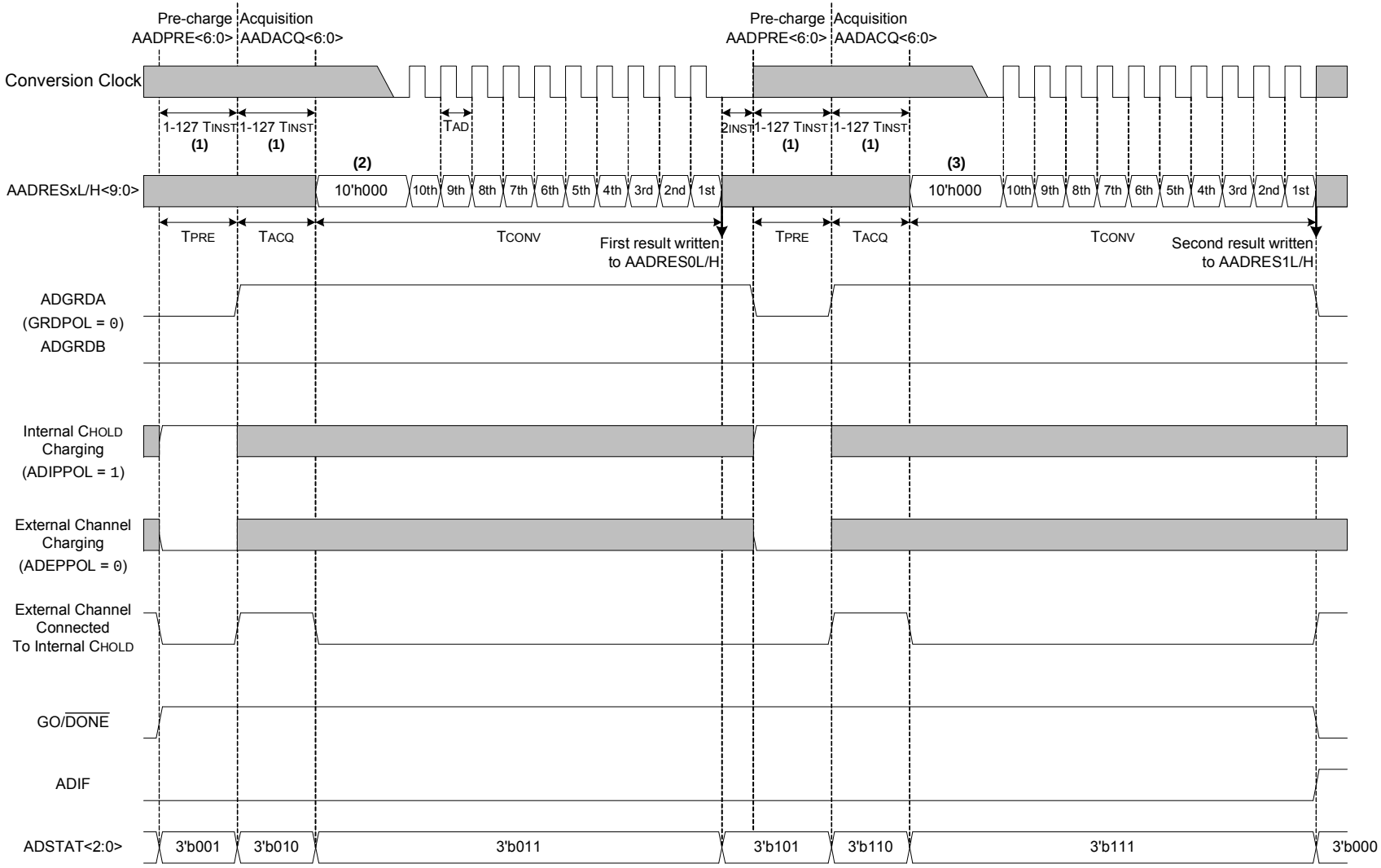
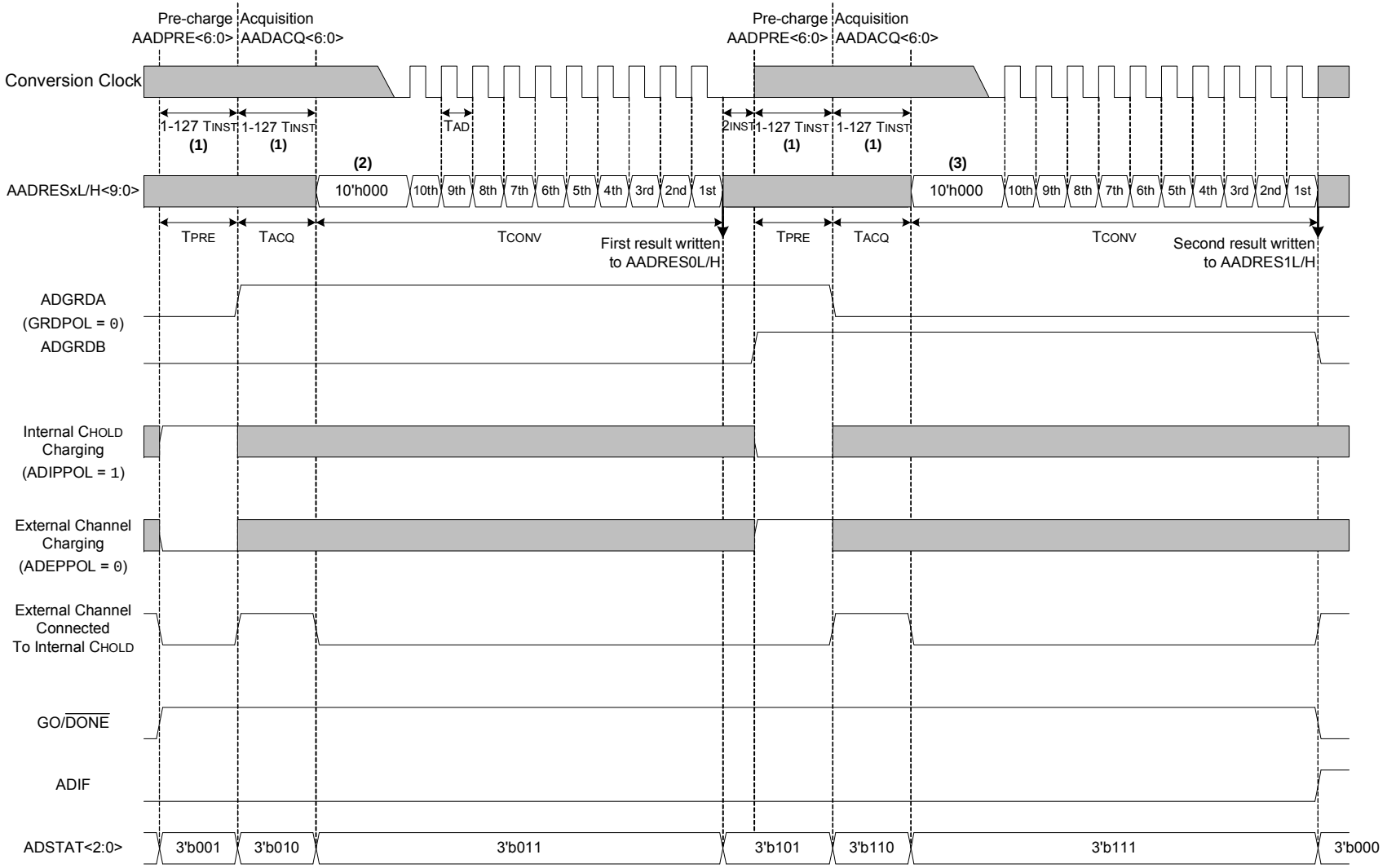


FIGURE 16-6: DOUBLE SAMPLE CONVERSION SEQUENCE (ADDSEN = 1 AND ADIPEN = 0)



Note 1: When the conversion clock is ADCRC, the pre-charge and acquisition timers are clocked by ADCRC.
2: The AADRES0L/H registers are set to zero during this period.
3: The AADRES1L/H registers are set to zero during this period.

FIGURE 16-7: DOUBLE SAMPLE CONVERSION SEQUENCE (ADDSEN = 1 AND ADIPEN = 1)



Note 1: When the conversion clock is ADCRC, the pre-charge and acquisition timers are clocked by ADCRC.
2: The AADRES0L/H registers are set to zero during this period.
3: The AADRES1L/H registers are set to zero during this period.

16.1.10 HARDWARE CVD DOUBLE CONVERSION PROCEDURE

This is an example procedure for using hardware CVD to perform a double conversion for differential CVD measurement with active guard drive.

1. Configure Port:
   - Enable pin output driver (Refer to the TRIS register).
   - Configure pin output low (Refer to the LAT register).
   - Disable weak pull-up (Refer to the WPU register).
2. Configure the ADC module:
   - Select an appropriate ADC conversion clock for your oscillator frequency.
   - Configure voltage reference.
   - Select ADC input channel.
   - Turn on the ADC module.
3. Configure the hardware CVD module:
   - Configure charge polarity and double conversion.
   - Configure pre-charge and acquisition timer.
   - Configure guard ring (optional).
   - Select additional capacitance (optional).
4. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt
   - Enable global interrupt(1)
5. Start conversion by setting the GO/$\overline{\text{DONE}}$ bit or by enabling the Special Event Trigger in the ADDCON2 register.
6. Wait for the ADC conversion to complete by one of the following:
   - Polling the GO/$\overline{\text{DONE}}$ bit.
   - Waiting for the ADC interrupt (interrupts enabled).
7. Read ADC result:
   - Conversion 1 result in ADDRES0H and ADDRES0L
   - Conversion 2 result in ADDRES1H and ADDRES1L
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

EXAMPLE 16-1: HARDWARE CVD DOUBLE CONVERSION

```
;This code block configures the ADC
;for polling, Vdd and Vss references, Fosc/16
;clock and AN0 input.
;
; The Hardware CVD will perform an inverted
; double conversion, Guard A and B drive are
; both enabled.
;Conversion start & polling for completion
are included.
;
BANKSEL    TRISA
BCF        TRISA,0      ;Set RA0 to output
BANKSEL    LATA
BCF        LATA,0       ;RA0 output low
BANKSEL    ANSELA
BCF        ANSELA,0     ;Set RA0 to digital
BANKSEL    WPUA
BCF        WPUA,0       ;Disable pull-up on
RA0
; Initialize ADC and Hardware CVD

BANKSEL    AADCON0
MOVLW      B'00000001'  ;Select channel AN0
MOVWF      AADCON0
BANKSEL    AADCON1
MOVLW      B'11010000'  ;Vdd and Vss Vref
MOVWF      AADCON1

BANKSEL    AADCON3
MOVLW      B'01000011'  ;Double and inverted
MOVWF      AADCON3      ;ADOUT disabled
BANKSEL    AADPRE
MOVLW      .10
MOVWF      AADPRE       ;Pre-charge Timer
BANKSEL    AADACQ
MOVLW      .10
MOVWF      AADACQ       ;Acquisition Timer
BANKSEL    AADGRD
MOVLW      B'11000000'  ;Guard on A and B
MOVWF      AADGRD
BANKSEL    AADCAP
MOVLW      B'00000000'
MOVWF      AADCAP       ;No additional
                        ;Capacitor
BANKSEL    ADCON0
BSF        ADCON0, GO
BTFSC      ADCON0, GO
GOTO       $-1          ;No, test again

;RESULTS OF CONVERIONS 1.
BANKSEL    AADRES0H         ;
MOVF       AADRES0H,W   ;Read upper 2 bits
MOVWF      RESULT0H     ;store in GPR space
MOVF       AADRES0L,W   ;Read lower 8 bits
MOVWF      RESULT0L     ;Store in GPR space

;RESULTS OF CONVERIONS 2.
BANKSEL    AADRES1H     ;
MOVF       AADRES1H,W   ;Read upper 2 bits
MOVWF      RESULT1H     ;store in GPR space
MOVF       AADRES1L,W   ;Read lower 8 bits
MOVWF      RESULT1L     ;Store in GPR space
```

16.1.11 HARDWARE CVD REGISTER MAPPING

The hardware CVD module is an enhanced expansion of the standard ADC module as stated in **Section 15.0 "Analog-to-Digital Converter (ADC) Module"** and is backward compatible with the other devices in this family. Control of the standard ADC module uses Bank 1 registers, see Table 16-1. This set of registers is mapped into Bank 14 with the control registers for the hardware CVD module. Although this subset of registers has different names, they are identical. Since the registers for the standard ADC are mapped into the Bank 14 address space, any changes to registers in Bank 1 will be reflected in Bank 14 and vice-versa.

TABLE 16-1: HARDWARE CVD REGISTER MAPPING

[Bank 14 Address]	[Bank 1 Address]
Hardware CVD	**ADC**
[711h] AADCON0(1)	[09Dh] ADCON0(1)
[712h] AADCON1(1)	[09Eh] ADCON1(1)
[713h] AADCON2	[09Fh] ADCON2(1)
[714h] AADCON3	
[715h] AADSTAT	
[716h] AADPRE	
[717h] AADACQ	
[718h] AADGRD	
[719h] AADCAP	
[71Ah] AADRES0L(1)	[09Bh] ADRES0L(1)
[71Bh] AADRES0H(1)	[09Ch] ADRES0H(1)
[71Ch] AADRES1L	
[71Dh] AADRES1H	

Note 1: Register is mapped in Bank 1 and Bank 14, using different names in each bank.

16.2 Register Definitions: Hardware CVD Control

REGISTER 16-1: AADCON0: HARDWARE CVD CONTROL REGISTER 0(1)

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	R/W-0/0
—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON
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 **Unimplemented:** Read as '0'

bit 6-2 **CHS<4:0>:** Analog Channel Select bits

- 11111 = FVR (Fixed Voltage Reference) Buffer 1 Output(2)
- 11110 = Reserved. No channel connected.
- 11101 = Temperature Indicator(3)
- 11100 = Reserved. No channel connected.
- 11011 = Reserved. No channel connected.
- 11010 = VREFH (ADC Positive Reference)
- 11001 = Reserved. No channel connected.
- •
- •
- •
- 00101 = Reserved. No channel connected.
- 00100 = AN4
- 00011 = AN3
- 00010 = AN2
- 00001 = AN1
- 00000 = AN0

bit 1 **GO/$\overline{\text{DONE}}$:** ADC Conversion Status bit

1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle.
This bit is automatically cleared by hardware when the ADC conversion has completed.
0 = ADC conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit

1 = ADC is enabled
0 = ADC is disabled and consumes no operating current

Note 1: See **Section 16.1.11 "Hardware CVD Register Mapping"** for more information.
2: See **Section 13.0 "Fixed Voltage Reference (FVR)"** for more information.
3: See **Section 14.0 "Temperature Indicator Module"** for more information.

REGISTER 16-2: AADCON1: HARDWARE CVD CONTROL REGISTER 1(1)

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
ADFM	ADCS<2:0>			—	—	ADPREF<1:0>	
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 **ADFM:** ADC Result Format Select bit
1 = Right justified. Six Most Significant bits of AADRESxH are set to '0' when the conversion result is loaded.
0 = Left justified. Six Least Significant bits of AADRESxL are set to '0' when the conversion result is loaded.

bit 6-4 ADCS<2:0>: ADC Conversion Clock Select bits
111 = FRC (clock supplied from a dedicated RC oscillator)(3)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock supplied from a dedicated RC oscillator)(3)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

bit 3-2 **Unimplemented:** Read as '0'

bit 1-0 ADPREF<1:0>: ADC Positive Voltage Reference Configuration bits
11 = VREF is connected to internal Fixed Voltage Reference (FVR) module(2)
10 = VREF is connected to external VREF+ pin
01 = Reserved
00 = VREF is connected to VDD

Note 1: See **Section 16.1.11 "Hardware CVD Register Mapping"** for more information.
2: When selecting the FVR or the VREF+ pin as the source of the positive reference, be aware that a minimum voltage specification exists. See **Section 21.0 "Electrical Specifications"** for details.
3: The Hardware CVD module only supports the FRC conversion clock while in Sleep. When the internal oscillator is running, ADCS should be set to an Fosc option.

REGISTER 16-3: AADCON2: HARDWARE CVD CONTROL REGISTER 2(1)

U-0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0
—	TRIGSEL<2:0>			—	—	—	—
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 **Unimplemented:** Read as '0'

bit 6-4 TRIGSEL<2:0>: ADC Special Event Trigger Source Selection bits

- 111 = Reserved. Auto-conversion Trigger disabled.
- 110 = Reserved. Auto-conversion Trigger disabled.
- 101 = Reserved. Auto-conversion Trigger disabled.
- 100 = Reserved. Auto-conversion Trigger disabled.
- 011 = TMR0 Overflow
- 010 = Reserved. Auto-conversion Trigger disabled.
- 001 = Reserved. Auto-conversion Trigger disabled.
- 000 = No Auto Conversion Trigger Selection bits(2,3)

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

Note 1: See **Section 16.1.11 "Hardware CVD Register Mapping"** for more information.

2: This is a rising edge sensitive input for all sources.

3: Signal used to set the corresponding interrupt flag.

REGISTER 16-4: AADCON3: HARDWARE CVD CONTROL REGISTER 3

R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0
ADEPPOL	ADIPPOL	—	ADOEN	ADOOEN	—	ADIPEN	ADDSEN
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 ADEPPOL: External Pre-charge Polarity bit(1)
- 1 = Selected channel is shorted to VDDIO during pre-charge time
- 0 = Selected channel is shorted to VSS during pre-charge time

bit 6 ADIPPOL: Internal Pre-charge Polarity bit(1)
- 1 = CHOLD is shorted to VREFH during pre-charge time
- 0 = CHOLD is shorted to VREFL during pre-charge time

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

bit 4 ADOEN: ADOUT Output Enable bit
- 1 = ADOUT pin is connected to ADC bus (normal passgate)
- 0 = No external connection to ADC bus

bit 3 ADOOEN: ADOUT Override Enable bit
- 1 = ADOUT pin is overridden during pre-charge with internal polarity value
- 0 = ADOUT pin is not overridden

bit 2 **Unimplemented:** Read as '0'

bit 1 ADIPEN: ADC Invert Polarity Enable bit

If ADDSEN = 1:
- 1 = The output value of the ADEPPOL, ADIPPOL, and GRDPOL bits used by the ADC are inverted for the second conversion
- 0 = The second ADC conversion proceeds like the first

If ADDSEN = 0:
This bit has no effect.

bit 0 ADDSEN: ADC Double Sample Enable bit
- 1 = The ADC immediately starts a new conversion after completing a conversion. GO/DONE bit is not automatically clear at end of conversion
- 0 = ADC operates in the traditional, single conversion mode

Note 1: When the ADDSEN = 1 and ADIPEN = 1; the polarity of this output is inverted for the second conversion time. The stored bit value does not change.

REGISTER 16-5: AADSTAT: HARDWARE CVD STATUS REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	ADCONV	ADSTG<1:0>	
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 7-3 **Unimplemented:** Read as '0'

bit 2 ADCONV: ADC Conversion Status bit
- 1 = Indicates ADC in Conversion Sequence for AADRES1H:AADRES1L
- 0 = Indicates ADC in Conversion Sequence for AADRES0H:AADRES0L (Also reads '0' when GO/$\overline{\text{DONE}}$ = 0)

bit 1-0 ADSTG<1:0>: ADC Stage Status bit
- 11 = ADC module is in conversion stage
- 10 = ADC module is in acquisition stage
- 01 = ADC module is in pre-charge stage
- 00 = ADC module is not converting (same as GO/$\overline{\text{DONE}}$ = 0)

REGISTER 16-6: AADPRE: HARDWARE CVD PRE-CHARGE CONTROL REGISTER

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	R/W-0/0
—	ADPRE<6:0>						
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 **Unimplemented:** Read as '0'

bit 6-0 ADPRE<6:0>: Pre-charge Time Select bits(1)
- 111 1111 = Pre-charge for 127 instruction cycles
- 111 1110 = Pre-charge for 126 instruction cycles
- •
- •
- •
- 000 0001 = Pre-charge for 1 instruction cycle (Fosc/4)
- 000 0000 = ADC pre-charge time is disabled

Note 1: When the FRC clock is selected as the conversion clock source, it is also the clock used for the pre-charge and acquisition times.

REGISTER 16-7: AADACQ: HARDWARE CVD ACQUISITION TIME CONTROL REGISTER

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	R/W-0/0
—	ADACQ<6:0>						
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 **Unimplemented:** Read as ‘0’

bit 6-0 ADACQ<6:0>: Acquisition/Charge Share Time Select bits(1)

111 1111 = Acquisition/charge share for 127 instruction cycles

111 1110 = Acquisition/charge share for 126 instruction cycles

•
•
•

000 0001 = Acquisition/charge share for one instruction cycle (Fosc/4)

000 0000 = ADC Acquisition/charge share time is disabled

Note 1: When the FRC clock is selected as the conversion clock source, it is also the clock used for the pre-charge and acquisition times.

REGISTER 16-8: AADGRD: HARDWARE CVD GUARD RING CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	U-0
GRDBOE(2)	GRDAOE(2)	GRDPOL(1,2)			—	—	—
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 GRDBOE: Guard Ring B Output Enable bit(2)

1 = ADC guard ring output is enabled to ADGRDB pin. Its corresponding TRISx bit must be clear.
0 = No ADC guard ring function to this pin is enabled

bit 6 GRDAOE: Guard Ring A Output Enable bit(2)

1 = ADC Guard Ring Output is enabled to ADGRDA pin. Its corresponding TRISx, x bit must be clear.
0 = No ADC Guard Ring function is enabled

bit 5 GRDPOL: Guard Ring Polarity selection bit(1,2)

1 = ADC guard ring outputs start as digital high during pre-charge stage
0 = ADC guard ring outputs start as digital low during pre-charge stage

bit 4-0 **Unimplemented:** Read as ‘0’

Note 1: When the ADDSEN = 1 and ADIPEN = 1; the polarity of this output is inverted for the second conversion time. The stored bit value does not change.

2: Guard Ring outputs are maintained while ADON = 1. The ADGRDA output switches polarity at the start of the acquisition time.

REGISTER 16-9: AADCAP: HARDWARE CVD ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—					ADDCAP<2:0>		
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 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 ADDCAP: ADC Additional Sample Capacitor Selection bits

111 = Nominal additional sample capacitor of 28 pF
110 = Nominal additional sample capacitor of 24 pF
101 = Nominal additional sample capacitor of 20 pF
100 = Nominal additional sample capacitor of 16 pF
011 = Nominal additional sample capacitor of 12 pF
010 = Nominal additional sample capacitor of 8 pF
001 = Nominal additional sample capacitor of 4 pF
000 = Additional sample capacitor is disabled

REGISTER 16-10: AADRESxH: HARDWARE CVD RESULT REGISTER MSB ADFM = 0(1)

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADRESx<9:2>							
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 **AD<9:2>**: Most Significant ADC results

Note 1: See **Section 16.1.11 "Hardware CVD Register Mapping"** for more information.

REGISTER 16-11: AADRESxL: HARDWARE CVD RESULT REGISTER LSL ADFM = 0(1)

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
ADRESx<1:0>		—	—	—	—	—	—
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 **AD<1:0>**: ADC Result Register bits
Lower two bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

Note 1: See **Section 16.1.11 "Hardware CVD Register Mapping"** for more information.

REGISTER 16-12: AADRESxH: HARDWARE CVD RESULT REGISTER MSB ADFM = 1(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-x/u	R/W-x/u
—	—	—	—	—	—	ADRESx<9:8>	
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 7-2 **Reserved**: Do not use.

bit 1-0 **AD<9:8>**: Most Significant ADC results

Note 1: See **Section 16.1.11 “Hardware CVD Register Mapping”** for more information.

REGISTER 16-13: AADRESxL: HARDWARE CVD RESULT REGISTER LSB ADFM = 1(1)

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
ADRESx<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 **AD<7:0>**: ADC Result Register bits
Lower two bits of 10-bit conversion result

Note 1: See **Section 16.1.11 “Hardware CVD Register Mapping”** for more information.

TABLE 16-2: SUMMARY OF REGISTERS ASSOCIATED WITH HARDWARE CVD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
AADCAP	—	—	—	—	—	ADDCAP<2:0>			126
AADCON0	—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON	120
AADCON1	ADFM	ADCS<2:0>			—	—	ADPREF<1:0>		121
AADCON2	—	TRIGSEL<2:0>			—	—	—	—	122
AADCON3	ADEPPOL	ADIPPOL	—	ADOEN	ADOOEN	—	ADIPEN	ADDSEN	123
AADGRD	GRDBOE	GRDAOE	GRDPOL	—	—	—	—	—	125
AADPRE	—	ADPRE<6:0>							124
AADRES0H	ADC Result 0 Register High								127
AADRES0L	ADC Result 0 Register Low								127
AADRES1H	ADC Result 1 Register High								128
AADRES1L	ADC Result 1 Register Low								128
AADSTAT	—	—	—	—	—	ADCONV	ADSTG<1:0>		124
AADACQ	—	ADACQ<6:0>							125
ANSELA	—	—	ANSA5	ANSA4	—	ANSA2	ANSA1	ANSA0	88
FVRCON	FVREN	FVRRDY	TSEN	TSRNG	—	—	ADFVR<1:0>		95
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
PIE1	—	ADIE	—	—	SSPIE	—	—	—	58
PIR1	—	ADIF	—	—	SSPIF	—	—	—	60
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	87

Legend: — = unimplemented read as '0'. Shaded cells are not used for hardware CVD module.
Note 1: Unimplemented, read as '1'.

17.0 TIMER0 MODULE

The Timer0 module is an 8-bit timer/counter with the following features:

- 8-bit timer/counter register (TMR0)
- 8-bit prescaler (independent of Watchdog Timer)
- Programmable internal or external clock source
- Programmable external clock edge selection
- Interrupt on overflow

Figure 17-1 is a block diagram of the Timer0 module.

17.1 Timer0 Operation

The Timer0 module can be used as either an 8-bit timer or an 8-bit counter.

17.1.1 8-BIT TIMER MODE

The Timer0 module will increment every instruction cycle, if used without a prescaler. 8-bit Timer mode is selected by clearing the TMR0CS bit of the OPTION_REG register.

When TMR0 is written, the increment is inhibited for two instruction cycles immediately following the write.

Note:	The value written to the TMR0 register can be adjusted, in order to account for the two instruction cycle delay when TMR0 is written.

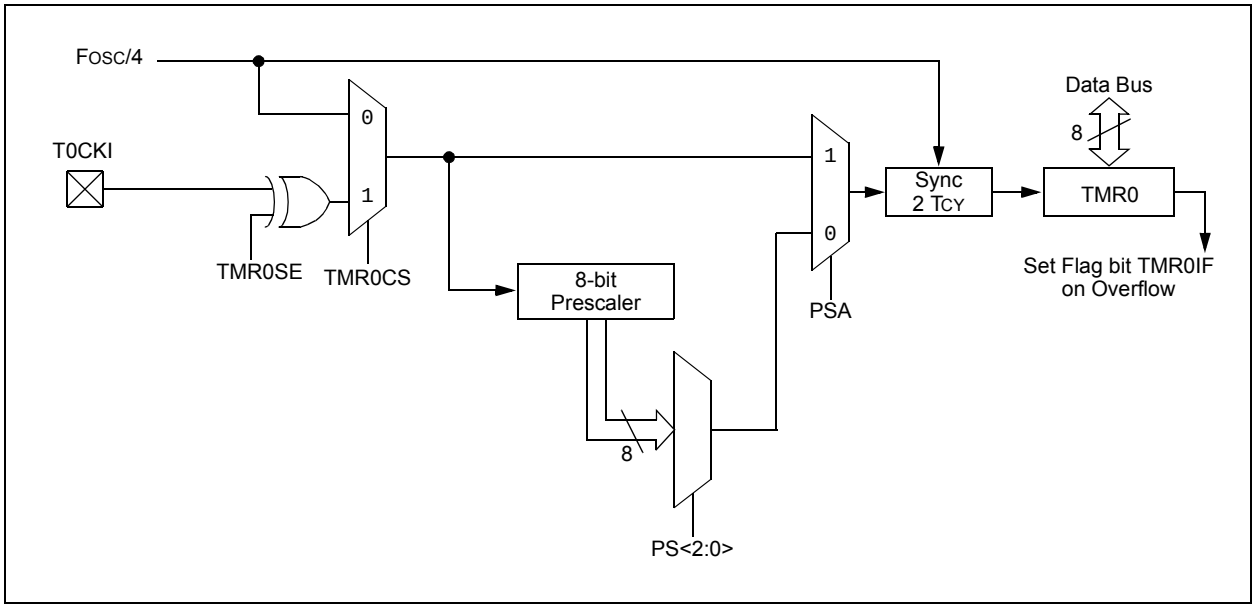
17.1.2 8-BIT COUNTER MODE

In 8-Bit Counter mode, the Timer0 module will increment on every rising or falling edge of the T0CKI pin.

8-Bit Counter mode using the T0CKI pin is selected by setting the TMR0CS bit in the OPTION_REG register to '1'.

The rising or falling transition of the incrementing edge for either input source is determined by the TMR0SE bit in the OPTION_REG register.

FIGURE 17-1: BLOCK DIAGRAM OF THE TIMER0



17.1.3 SOFTWARE PROGRAMMABLE PRESCALER

A software programmable prescaler is available for exclusive use with Timer0. The prescaler is enabled by clearing the PSA bit of the OPTION_REG register.

Note:	The Watchdog Timer (WDT) uses its own independent prescaler.

There are eight prescaler options for the Timer0 module ranging from 1:2 to 1:256. The prescale values are selectable via the PS<2:0> bits of the OPTION_REG register. In order to have a 1:1 prescaler value for the Timer0 module, the prescaler must be disabled by setting the PSA bit of the OPTION_REG register.

The prescaler is not readable or writable. All instructions writing to the TMR0 register will clear the prescaler.

17.1.4 TIMER0 INTERRUPT

Timer0 will generate an interrupt when the TMR0 register overflows from FFh to 00h. The TMR0IF interrupt flag bit of the INTCON register is set every time the TMR0 register overflows, regardless of whether or not the Timer0 interrupt is enabled. The TMR0IF bit can only be cleared in software. The Timer0 interrupt enable is the TMR0IE bit of the INTCON register.

Note:	The Timer0 interrupt cannot wake the processor from Sleep since the timer is frozen during Sleep.

17.1.5 8-BIT COUNTER MODE SYNCHRONIZATION

When in 8-Bit Counter mode, the incrementing edge on the T0CKI pin must be synchronized to the instruction clock. Synchronization can be accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the instruction clock. The high and low periods of the external clocking source must meet the timing requirements as shown in **Section 21.0 "Electrical Specifications"**.

17.1.6 OPERATION DURING SLEEP

Timer0 cannot operate while the processor is in Sleep mode. The contents of the TMR0 register will remain unchanged while the processor is in Sleep mode.

17.2 Register Definitions: Option Register

REGISTER 17-1: OPTION_REG: OPTION REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
$\overline{\text{WPUEN}}$	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>		
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 **$\overline{\text{WPUEN}}$:** Weak Pull-Up Enable bit
1 = All weak pull-ups are disabled (except $\overline{\text{MCLR}}$, if it is enabled)
0 = Weak pull-ups are enabled by individual WPUx latch values

bit 6 **INTEDG:** Interrupt Edge Select bit
1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin

bit 5 **TMR0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (FOSC/4)

bit 4 **TMR0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is not assigned to the Timer0 module
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS<2:0>:** Prescaler Rate Select bits

Bit Value	Timer0 Rate
000	1 : 2
001	1 : 4
010	1 : 8
011	1 : 16
100	1 : 32
101	1 : 64
110	1 : 128
111	1 : 256

TABLE 17-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
AADCON2		TRIGSEL<2:0>			—	—	—	—	122
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
OPTION_REG	$\overline{\text{WPUEN}}$	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>			132
TMR0	Holding Register for the 8-bit Timer0 Count								130*
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	87

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by the Timer0 module.
* Page provides register information.
Note 1: Unimplemented, read as '1'.

18.0 MASTER SYNCHRONOUS SERIAL PORT MODULE

18.1 Master SSP (MSSP1) Module Overview

The Master Synchronous Serial Port (MSSP1) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP1 module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)

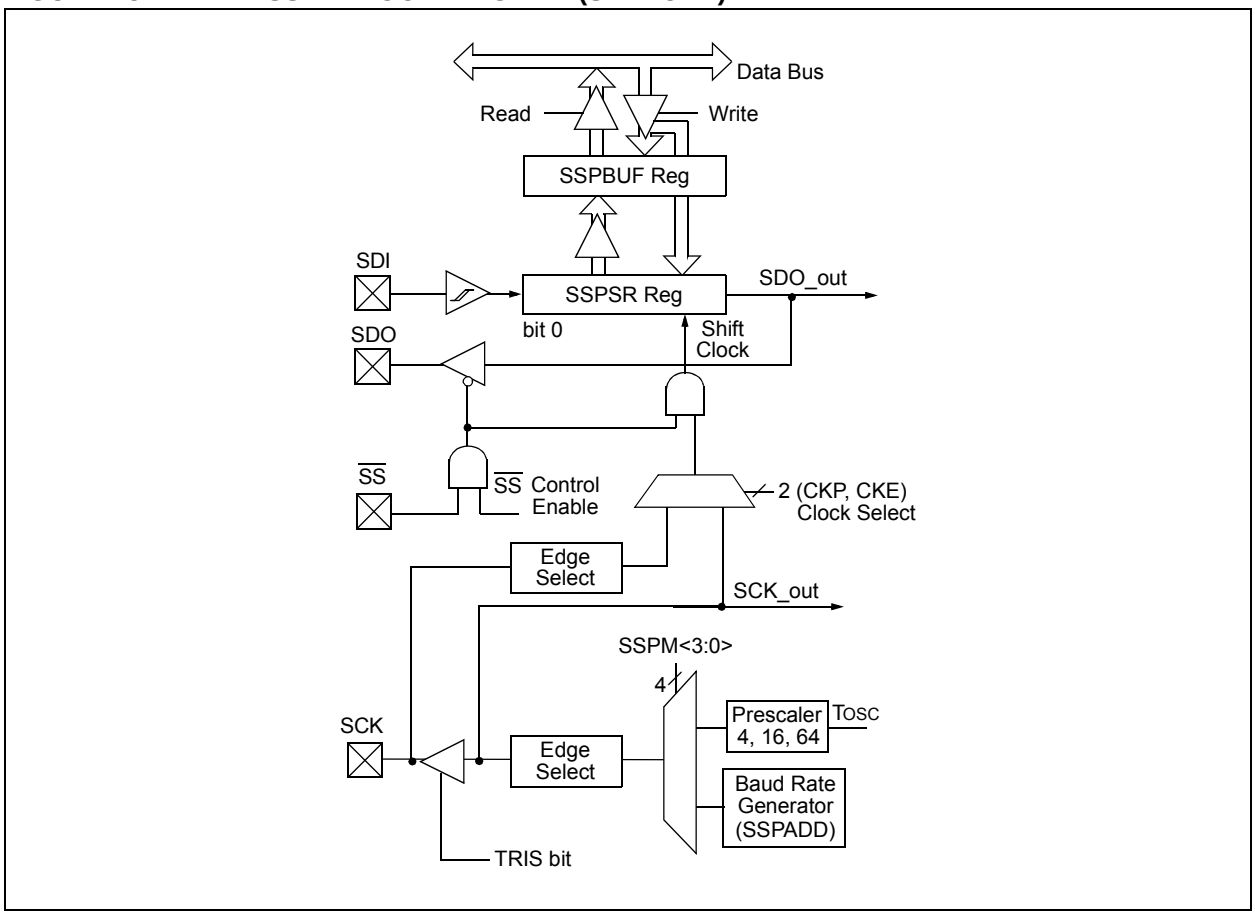
The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Parity
- Slave Select Synchronization (Slave mode only)
- Daisy-chain connection of slave devices

Figure 18-1 is a block diagram of the SPI interface module.

Note: Register names, I/O pins, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module, when required.

FIGURE 18-1: MSSP1 BLOCK DIAGRAM (SPI MODE)



The I²C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited Multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDA hold times

Figure 18-2 is a block diagram of the I²C interface module in Master mode. Figure 18-3 is a diagram of the I²C interface module in Slave mode.

The PIC12LF1552 has one MSSP module.

FIGURE 18-2: MSSP1 BLOCK DIAGRAM (I²C MASTER MODE)

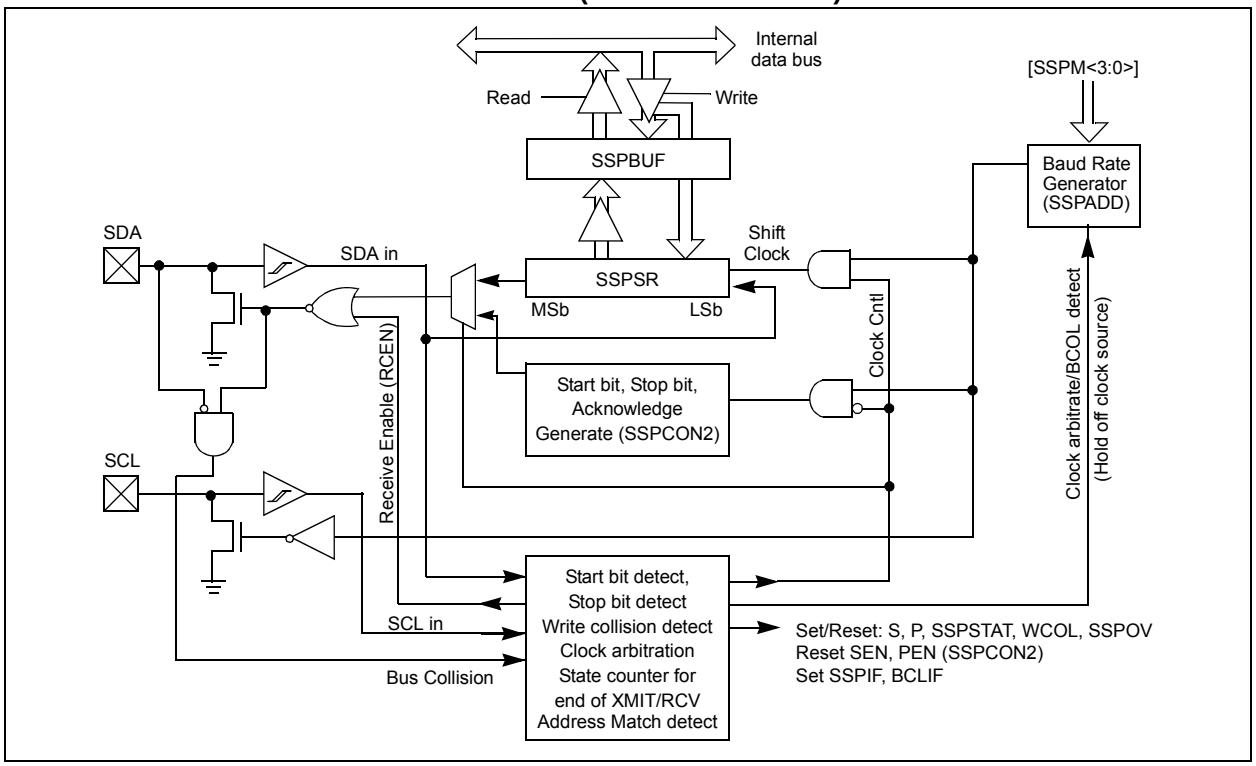
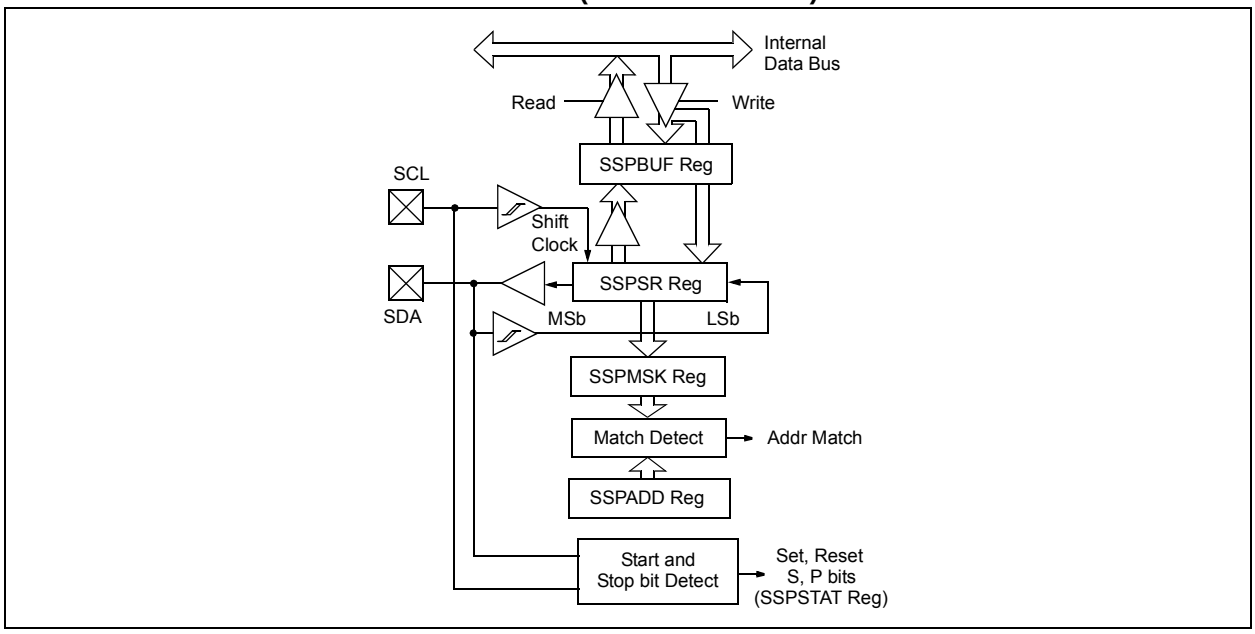


FIGURE 18-3: MSSP1 BLOCK DIAGRAM (I^2C SLAVE MODE)



18.2 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 18-1 shows the block diagram of the MSSP1 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 18-4 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. With either the master or the slave device, 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 18-5 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.

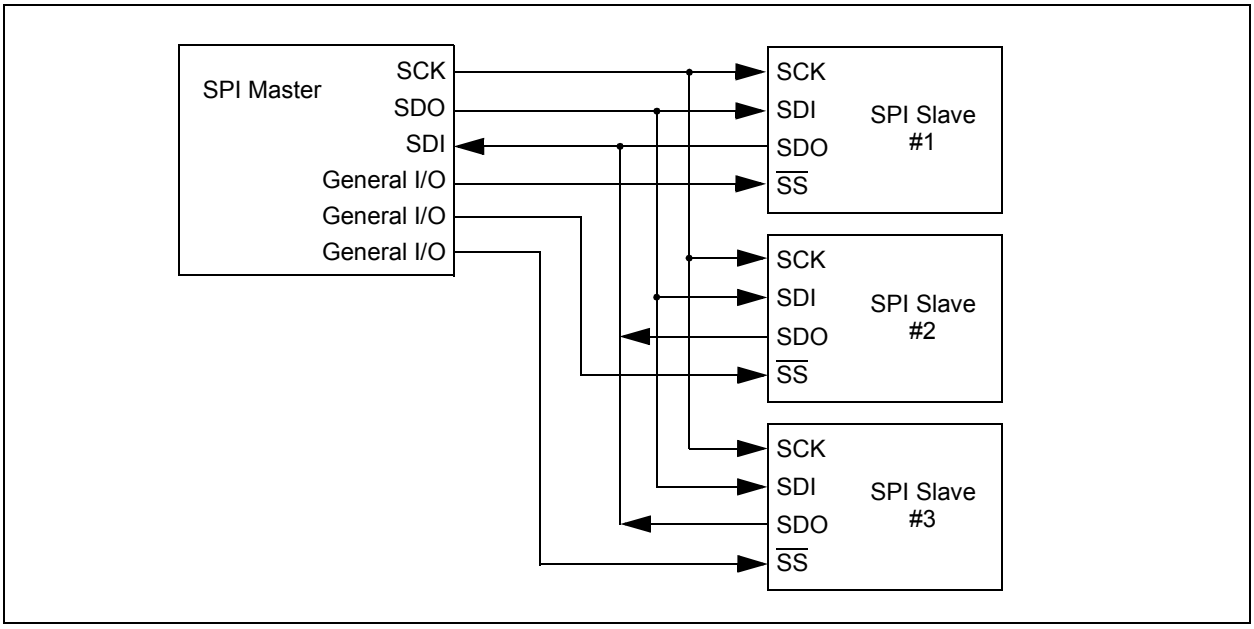
Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:

- 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.

FIGURE 18-4: SPI MASTER AND MULTIPLE SLAVE CONNECTION



18.2.1 SPI MODE REGISTERS

The MSSP1 module has five registers for SPI mode operation. These are:

- MSSP1 STATUS register (SSPSTAT)
- MSSP1 Control Register 1 (SSPCON1)
- MSSP1 Control Register 3 (SSPCON3)
- MSSP1 Data Buffer register (SSPBUF)
- MSSP1 Address register (SSPADD)
- MSSP1 Shift register (SSPSR) (Not directly accessible)

SSPCON1 and SSPSTAT are the control and STATUS registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower six bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

In SPI master mode, SSPADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in **Section 18.7 "Baud Rate Generator"**

SSPSR is the shift register used for shifting data in and out. SSPBUF provides indirect access to the SSPSR register. SSPBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPSR and SSPBUF together create a buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

18.2.2 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

To enable the serial port, SSP Enable bit, SSPEN of the SSPCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPCONx registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI must have corresponding TRIS bit set
- SDO must have corresponding TRIS bit cleared
- SCK (Master mode) must have corresponding TRIS bit cleared
- SCK (Slave mode) must have corresponding TRIS bit set
- $\overline{SS}$ must have corresponding TRIS bit set

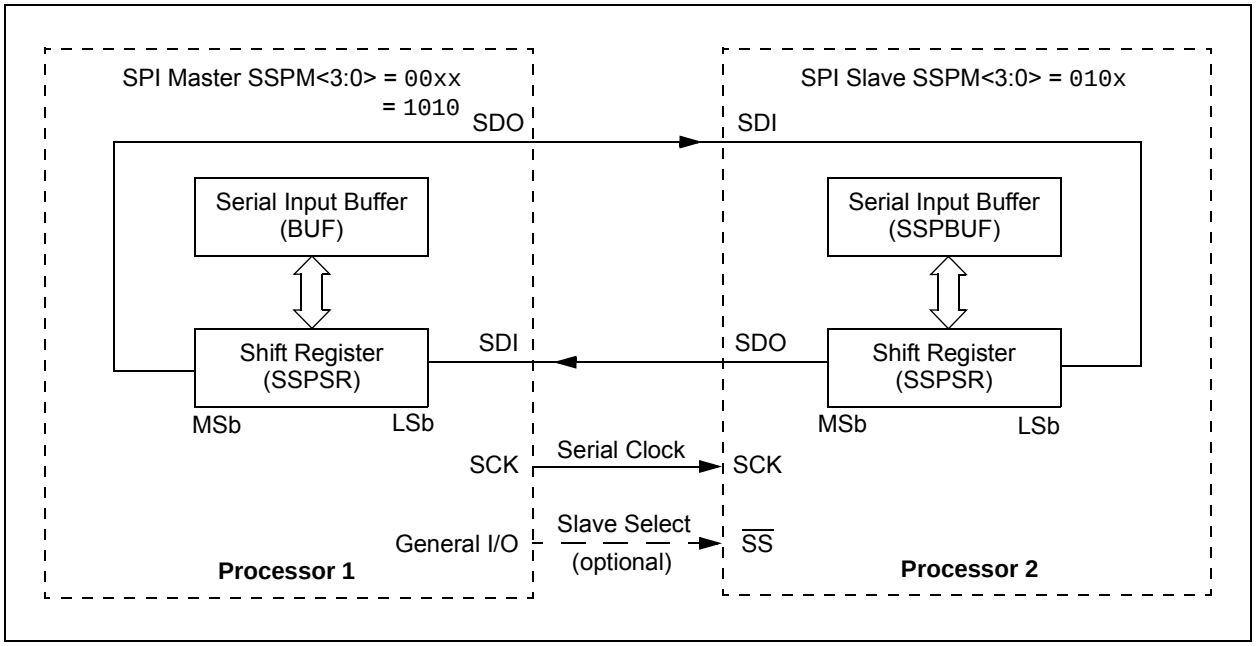
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

The MSSP1 consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full Detect bit, BF of the SSPSTAT register, and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the write collision detect bit WCOL of the SSPCON1 register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF of the SSPSTAT register, indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP1 interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the SSPSTAT register indicates the various Status conditions.

FIGURE 18-5: SPI MASTER/SLAVE CONNECTION



18.2.3 SPI MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK line. The master determines when the slave (Processor 2, Figure 18-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and Status bits appropriately set).

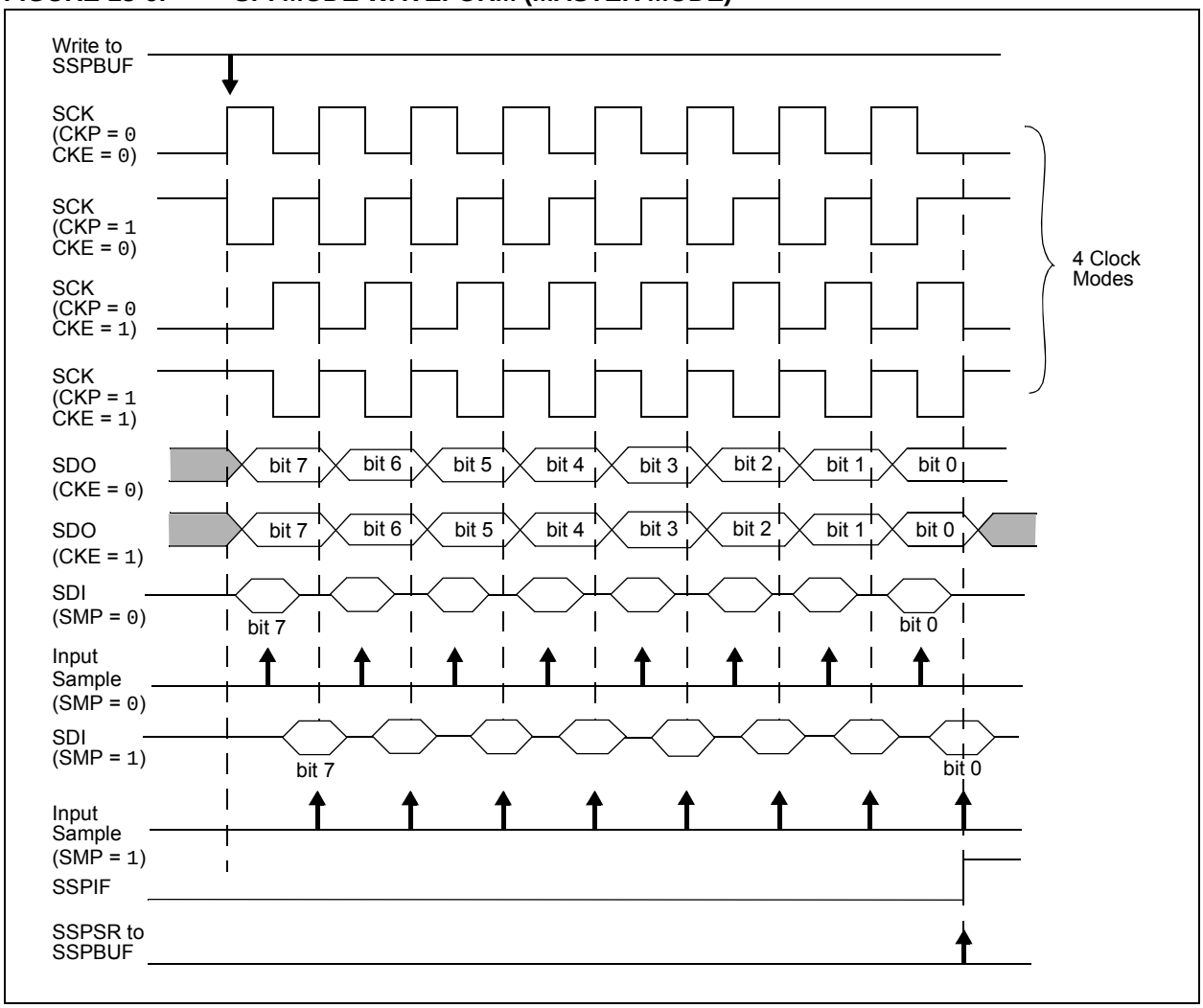
The clock polarity is selected by appropriately programming the CKP bit of the SSPCON1 register and the CKE bit of the SSPSTAT register. This then, would give waveforms for SPI communication as shown in Figure 18-6, Figure 18-8, Figure 18-9 and Figure 18-10, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 * TCY)
- FOSC/64 (or 16 * TCY)
- Fosc/(4 * (SSPADD + 1))

Figure 18-6 shows the waveforms for Master mode.

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

FIGURE 18-6: SPI MODE WAVEFORM (MASTER MODE)



18.2.4 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 SSPIF 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.

18.2.4.1 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 18-7 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.

18.2.5 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.

FIGURE 18-7: SPI DAISY-CHAIN CONNECTION

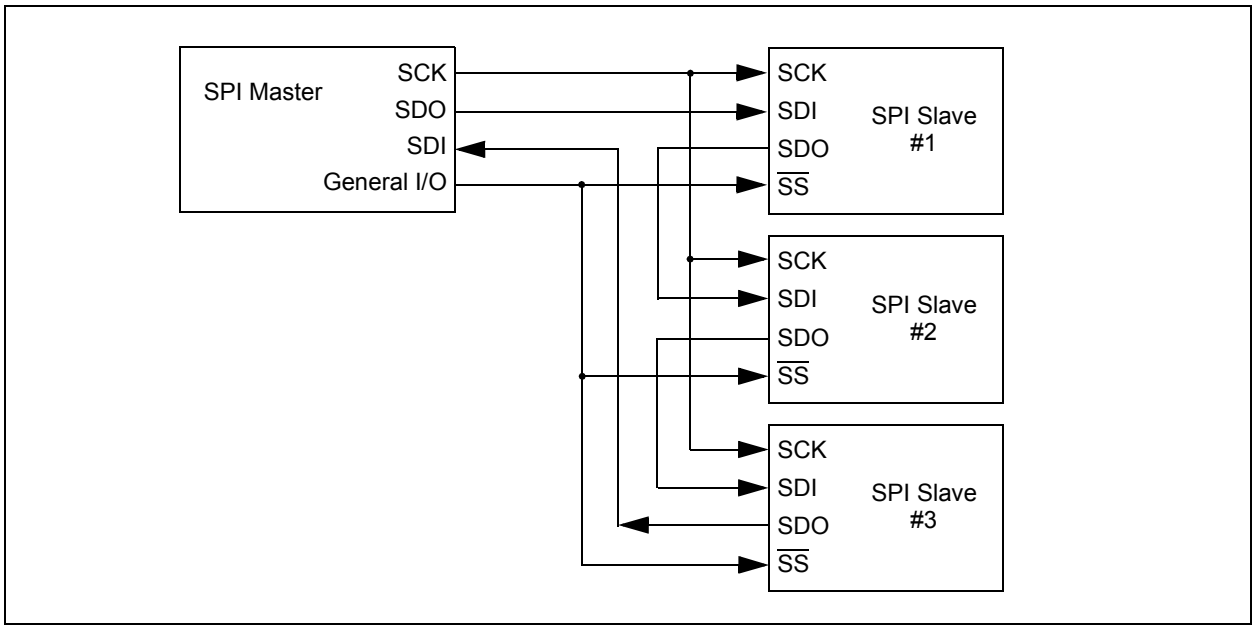


FIGURE 18-8: SLAVE SELECT SYNCHRONOUS WAVEFORM

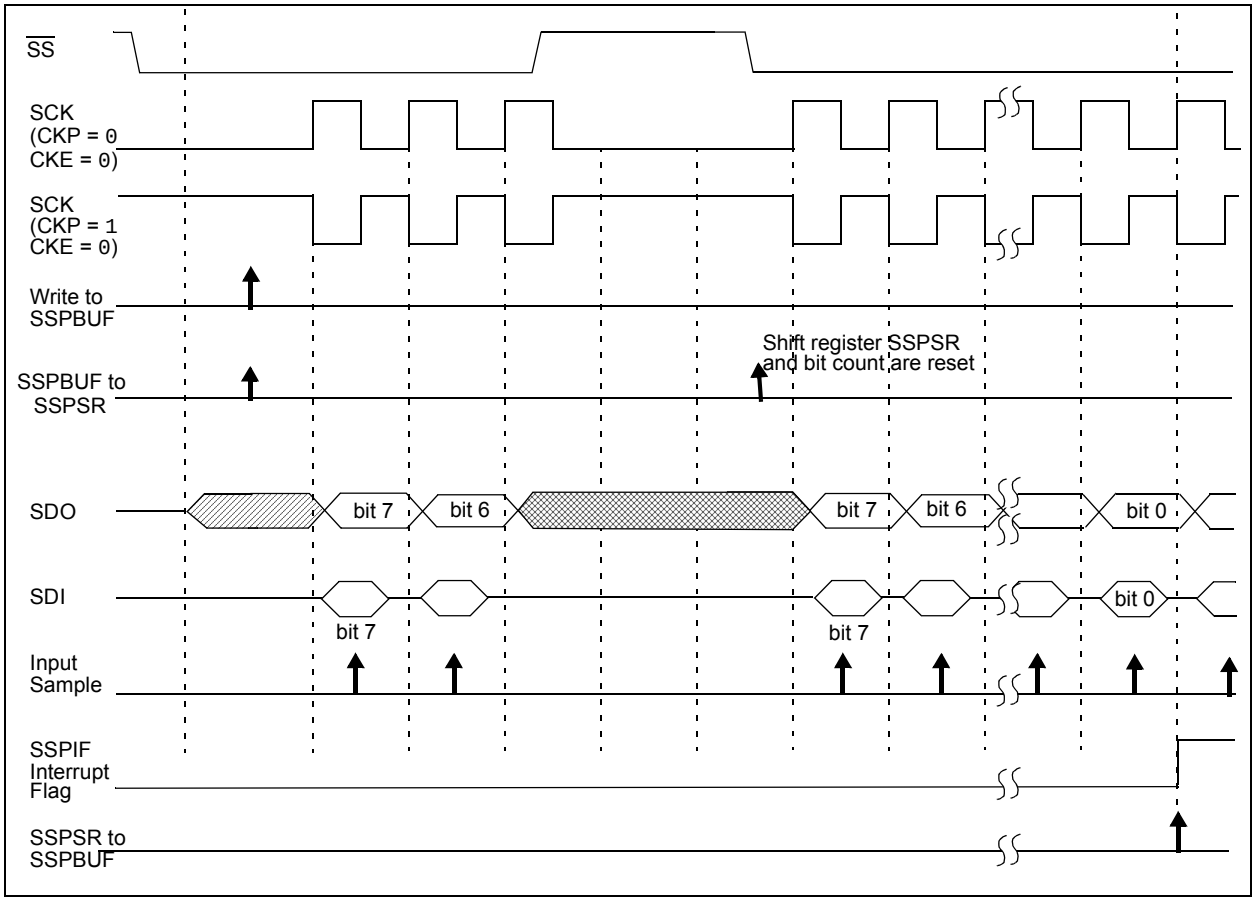


FIGURE 18-9: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

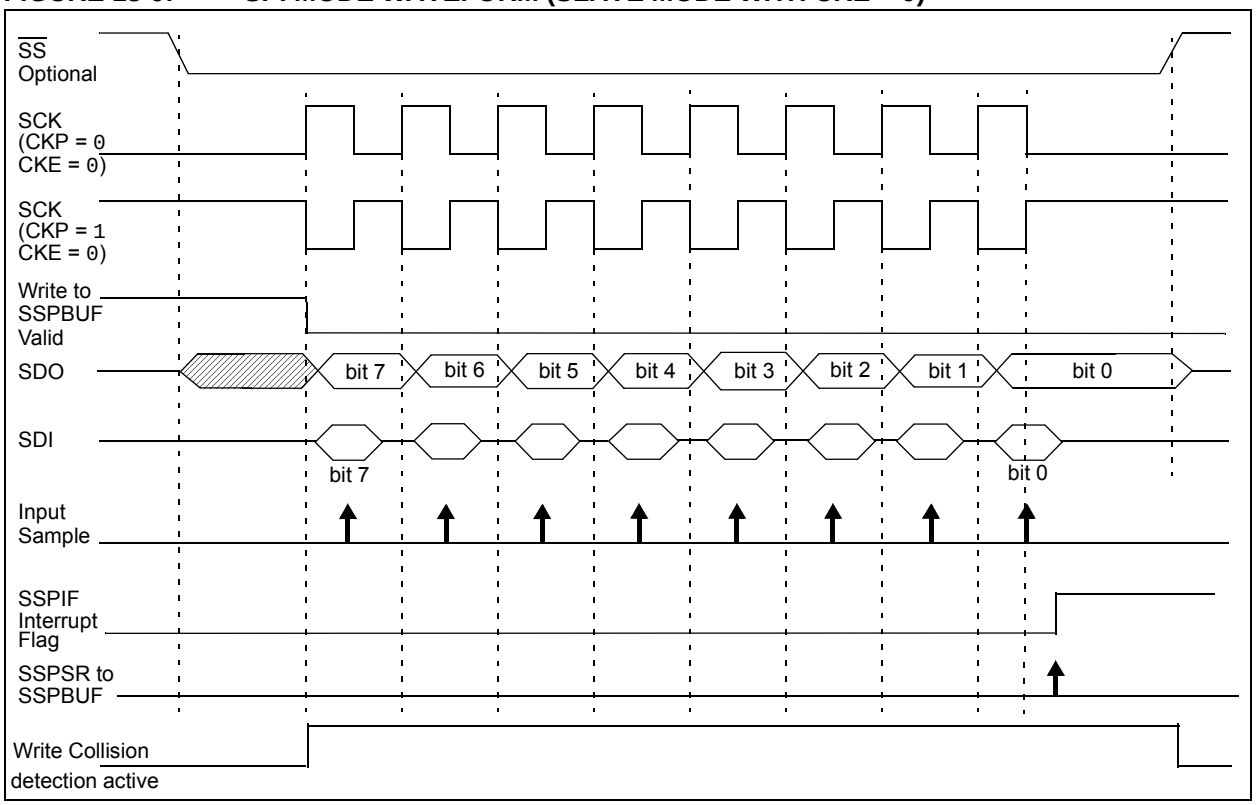
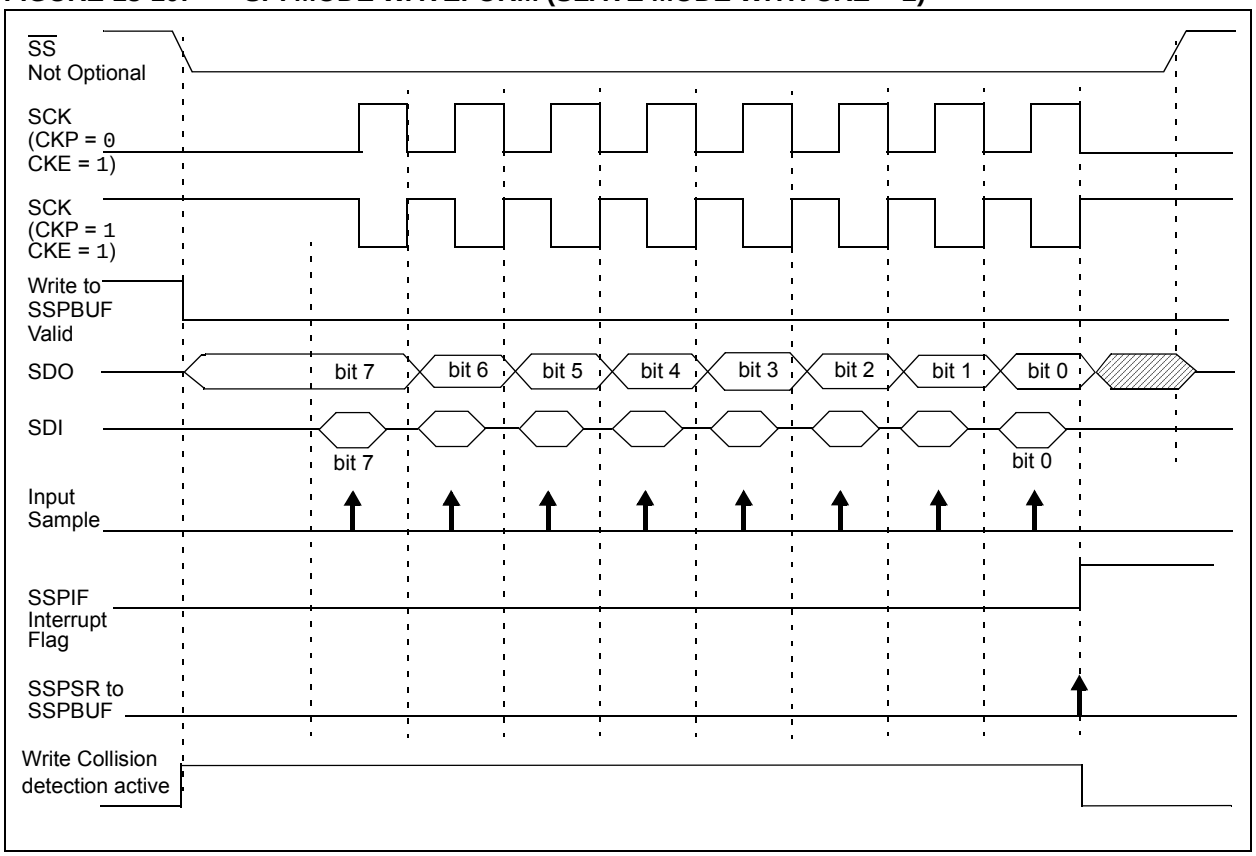


FIGURE 18-10: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



18.2.6 SPI OPERATION IN SLEEP MODE

In SPI Master mode, module clocks may be operating at a different speed than when in Full-Power mode; in the case of the Sleep mode, all clocks are halted.

Special care must be taken by the user when the MSSP1 clock is much faster than the system clock.

In Slave mode, when MSSP1 interrupts are enabled, after the master completes sending data, an MSSP1 interrupt will wake the controller from Sleep.

If an exit from Sleep mode is not desired, MSSP1 interrupts should be disabled.

In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSP1 interrupt flag bit will be set and if enabled, will wake the device.

TABLE 18-1: SUMMARY OF REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	ANSA5	ANSA4	—	ANSA2	ANSA1	ANSA0	85
APFCON	—	SDOSEL	SSSEL	SDSEL	—	—	—	—	85
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
PIE1	—	ADIE	—	—	SSPIE	—	—	—	58
PIR1	—	ADIF	—	—	SSPIF	—	—	—	60
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								137*
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				182
SSPCON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN	184
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	181
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	87

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by the MSSP1 in SPI mode.

* Page provides register information.

Note 1: Unimplemented, read as '1'.

18.3 I²C MODE OVERVIEW

The Inter-Integrated Circuit Bus (I²C) is a multi-master serial data communication bus. Devices communicate in a master/slave environment where the master devices initiate the communication. A Slave device is controlled through addressing.

The I²C bus specifies two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

Figure 18-2 and Figure 18-3 show the block diagrams of the MSSP1 module when operating in I²C mode.

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors for the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one.

Figure 18-11 shows a typical connection between two processors configured as master and slave devices.

The I²C bus can operate with one or more master devices and one or more slave devices.

There are four potential modes of operation for a given device:

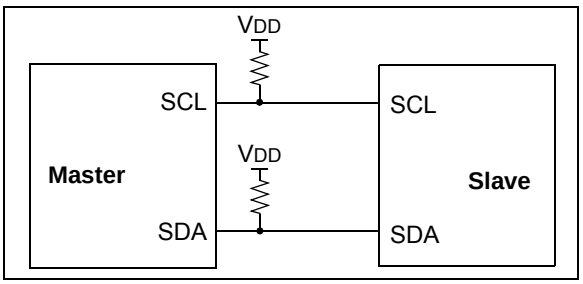
- Master Transmit mode
  (master is transmitting data to a slave)
- Master Receive mode
  (master is receiving data from a slave)
- Slave Transmit mode
  (slave is transmitting data to a master)
- Slave Receive mode
  (slave is receiving data from the master)

To begin communication, a master device starts out in Master Transmit mode. The master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an $\overline{\text{ACK}}$. The master then continues in either Transmit mode or Receive mode and the slave continues in the complement, either in Receive mode or Transmit mode, respectively.

A Start bit is indicated by a high-to-low transition of the SDA line while the SCL line is held high. Address and data bytes are sent out, Most Significant bit (MSb) first. The Read/Write bit is sent out as a logical one when the master intends to read data from the slave, and is sent out as a logical zero when it intends to write data to the slave.

FIGURE 18-11: I²C MASTER/ SLAVE CONNECTION



The Acknowledge bit ($\overline{\text{ACK}}$) is an active-low signal, which holds the SDA line low to indicate to the transmitter that the slave device has received the transmitted data and is ready to receive more.

The transition of a data bit is always performed while the SCL line is held low. Transitions that occur while the SCL line is held high are used to indicate Start and Stop bits.

If the master intends to write to the slave, then it repeatedly sends out a byte of data, with the slave responding after each byte with an $\overline{\text{ACK}}$ bit. In this example, the master device is in Master Transmit mode and the slave is in Slave Receive mode.

If the master intends to read from the slave, then it repeatedly receives a byte of data from the slave, and responds after each byte with an $\overline{\text{ACK}}$ bit. In this example, the master device is in Master Receive mode and the slave is Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop bit. If the master device is in Receive mode, it first sends a not $\overline{\text{ACK}}$ bit in place of an $\overline{\text{ACK}}$ and then terminates the transfer with a Stop bit.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send another Start bit in place of the Stop bit or last $\overline{\text{ACK}}$ bit when it is in receive mode.

The I²C bus specifies three message protocols;

- Single message where a master writes data to a slave.
- Single message where a master reads data from a slave.
- Combined message where a master initiates a minimum of two writes, or two reads, or a combination of writes and reads, to one or more slaves.

When one device is transmitting a logical one, or letting the line float, and a second device is transmitting a logical zero, or holding the line low, the first device can detect that the line is not a logical one. This detection, when used on the SCL line, is called clock stretching. Clock stretching gives slave devices a mechanism to control the flow of data. When this detection is used on

the SDA line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

18.3.1 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCL connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

18.3.2 ARBITRATION

Each master device must monitor the bus for Start and Stop bits. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, there are three conditions under which a collision can occur:

a) Two master devices may try to initiate a transmission on or about the same time
b) A device acting as multiple devices on the bus (one of which is acting as a master) may collide with another master which is trying to access a slave address on the first device
c) Two slaves may respond to a general call read at the same time

18.3.2.1 Multi-Master Collision

In the first condition each master transmitter checks the level of the SDA data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels do not match loses arbitration and must stop transmitting on the SDA line. For example, if one transmitter holds the SDA line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDA line will be low.

The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating. The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDA line. If this transmitter is also a master device, it must also stop driving the SCL line. Then it can monitor the lines for a Stop condition before trying to reissue its transmission. In the meantime, the other device that has not noticed any difference between the expected and actual levels on the SDA line continues with its original transmission. It can do so without any complications because, so far, the transmission appears exactly as expected with no other transmitter disturbing the message.

18.3.2.2 Multi-Master with Slave Recovery

In the second condition there are essentially three entities on the bus: a master and a slave that reside within the same device, and a second master attempting to access the slave. If the master coupled to the slave loses the arbitration, then the slave must be able to recover and correctly respond to the second master. To accomplish this, the master which shares a device with the slave must use the I^2C Firmware Controller Master mode of operation. This mode utilizes software to perform the master function, while the slave monitors the bus as a separate entity allowing it to respond to the second master.

To detect a collision, each device transmitting on the bus must check the level of the SDA data line and compare it to the level that it expects to find. The first transmitter to observe that the two levels do not match loses arbitration, and it must drop off the bus as well as stop transmitting on the SDA line if it is acting as a master. For example, if one transmitter holds the SDA line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDA line will be low.

If two master devices send a message to two different slave devices at the address stage, the master sending the lower slave address always wins arbitration. When two master devices send messages to the same slave address, and addresses can sometimes refer to multiple slaves, the arbitration process must continue into the data stage.

18.3.2.3 Multi-Slave Collision

When a system attempts a slave read from the General Call address, there is the possibility that more than one slave devices may attempt to return data to the master. When this happens, a read collision occurs and the slaves which fail the arbitration must fall off the bus and allow the winning device to continue its data transmission. Once the winning device has completed its data transfer, the master can then initiate an additional read from the General Call address to retrieve the second slave's data. In large systems this may require multiple reads by the master. Several protocols use this feature and the MSSP hardware incorporates a detection mechanism in the slave read to detect the condition and respond appropriately.

In most designs, arbitration usually occurs very rarely, but it is a necessary process for proper multi-master support or even multi-slave systems that use General Call address reads.

18.4 I²C MODE OPERATION

All MSSP1 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.

18.4.1 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.

18.4.2 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.

18.4.3 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.

18.4.4 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 18-2: 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 outputting and expected high state.

18.4.5 START CONDITION

The I^2C specification defines a Start condition as a transition of SDA from a high to a low state while SCL line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an Active state. Figure 18-12 shows wave forms for Start and Stop conditions.

A bus collision can occur on a Start condition if the module samples the SDA line low before asserting it low. This does not conform to the I^2C Specification that states no bus collision can occur on a Start.

18.4.6 STOP CONDITION

A Stop condition is a transition of the SDA line from low-to-high state while the SCL line is high.

Note: At least one SCL low time must appear before a Stop is valid, therefore, if the SDA line goes low then high again while the SCL line stays high, only the Start condition is detected.

18.4.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave. Figure 18-13 shows the wave form for a Restart condition.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes, the master can issue a Restart and the high address byte with the R/$\overline{W}$ bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with R/$\overline{W}$ clear in 10-bit mode, a prior match flag is set and maintained. Until a Stop condition, a high address with R/$\overline{W}$ clear, or high address match fails.

18.4.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPCON3 register can enable the generation of an interrupt in Slave modes that do not typically support this function. Slave modes where interrupt on Start and Stop detect are already enabled, these bits will have no effect.

FIGURE 18-12: I^2C START AND STOP CONDITIONS

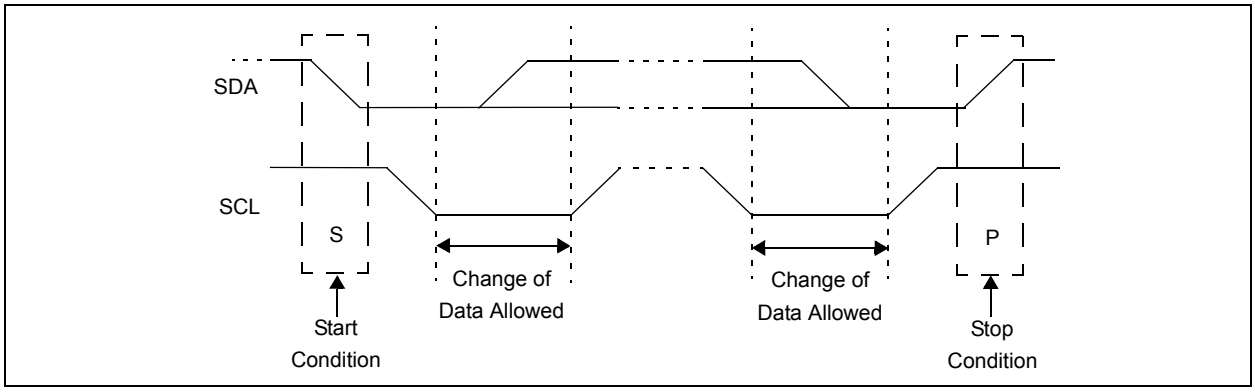
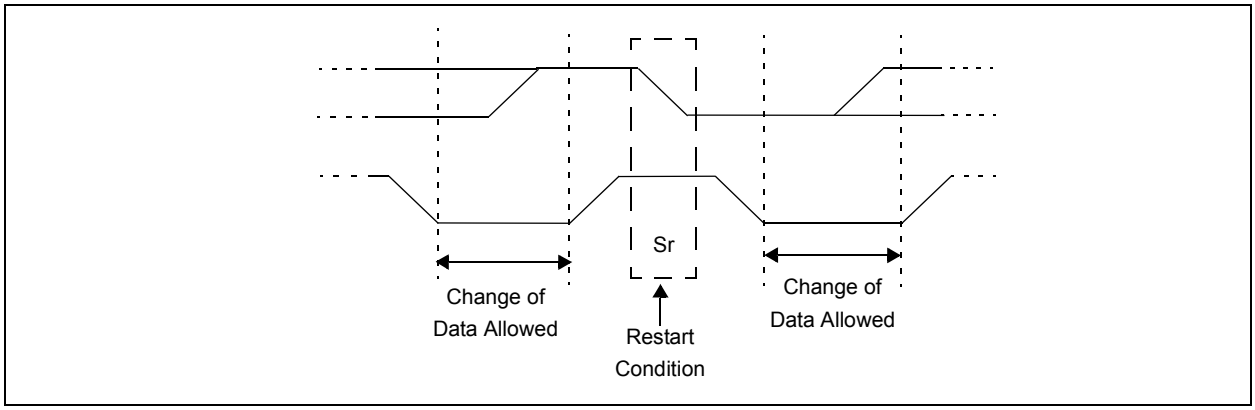


FIGURE 18-13: I^2C RESTART CONDITION



18.4.9 ACKNOWLEDGE SEQUENCE

The 9th SCL pulse for any transferred byte in I^2C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge ($\overline{ACK}$) is an active-low signal, pulling the SDA line low indicated to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an $\overline{ACK}$ is placed in the ACKSTAT bit of the SSPCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the $\overline{ACK}$ value sent back to the transmitter. The ACKDT bit of the SSPCON2 register is set/cleared to determine the response.

Slave hardware will generate an $\overline{ACK}$ response if the AHEN and DHEN bits of the SSPCON3 register are clear.

There are certain conditions where an $\overline{ACK}$ will not be sent by the slave. If the BF bit of the SSPSTAT register or the SSPOV bit of the SSPCON1 register are set when a byte is received.

When the module is addressed, after the 8th falling edge of SCL on the bus, the ACKTIM bit of the SSPCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is enabled.

18.5 I^2C SLAVE MODE OPERATION

The MSSP1 Slave mode operates in one of four modes selected in the SSPM bits of SSPCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operate the same as the other modes with SSPIF additionally getting set upon detection of a Start, Restart, or Stop condition.

18.5.1 SLAVE MODE ADDRESSES

The SSPADD register (Register 18-6) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPBUF register and an interrupt is generated. If the value does not match, the module goes idle and no indication is given to the software that anything happened.

The SSP Mask register (Register 18-5) affects the address matching process. See **Section 18.5.9 "SSP Mask Register"** for more information.

18.5.1.1 I^2C Slave 7-Bit Addressing Mode

In 7-bit Addressing mode, the LSb of the received data byte is ignored when determining if there is an address match.

18.5.1.2 I^2C Slave 10-Bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of '1 1 1 1 0 A9 A8 0'. A9 and A8 are the two MSb of the 10-bit address and stored in bits 2 and 1 of the SSPADD register.

After the acknowledge of the high byte the UA bit is set and SCL is held low until the user updates SSPADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPADD. Even if there is not an address match; SSPIF and UA are set, and SCL is held low until SSPADD is updated to receive a high byte again. When SSPADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the R/$\overline{W}$ bit set. The slave hardware will then acknowledge the read request and prepare to clock out data. This is only valid for a slave after it has received a complete high and low address byte match.

18.5.2 SLAVE RECEPTION

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

When the overflow condition exists for a received address, then not Acknowledge is given. An overflow condition is defined as either bit BF of the SSPSTAT register is set, or bit SSPOV of the SSPCON1 register is set. The BOEN bit of the SSPCON3 register modifies this operation. For more information see Register 18-4.

An MSSP1 interrupt is generated for each transferred data byte. Flag bit, SSPIF, must be cleared by software.

When the SEN bit of the SSPCON2 register is set, SCL will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit of the SSPCON1 register, except sometimes in 10-bit mode. See **Section 18.2.3 "SPI Master Mode"** for more detail.

18.5.2.1 7-Bit Addressing Reception

This section describes a standard sequence of events for the MSSP1 module configured as an I^2C Slave in 7-bit Addressing mode. Figure 18-14 and Figure 18-15 are used as visual references for this description.

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

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

18.5.2.2 7-Bit Reception with AHEN and DHEN

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

This list describes the steps that need to be taken by slave software to use these options for I^2C communication. Figure 18-16 displays a module using both address and data holding. Figure 18-17 includes the operation with the SEN bit of the SSPCON2 register set.

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

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

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

FIGURE 18-14: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)

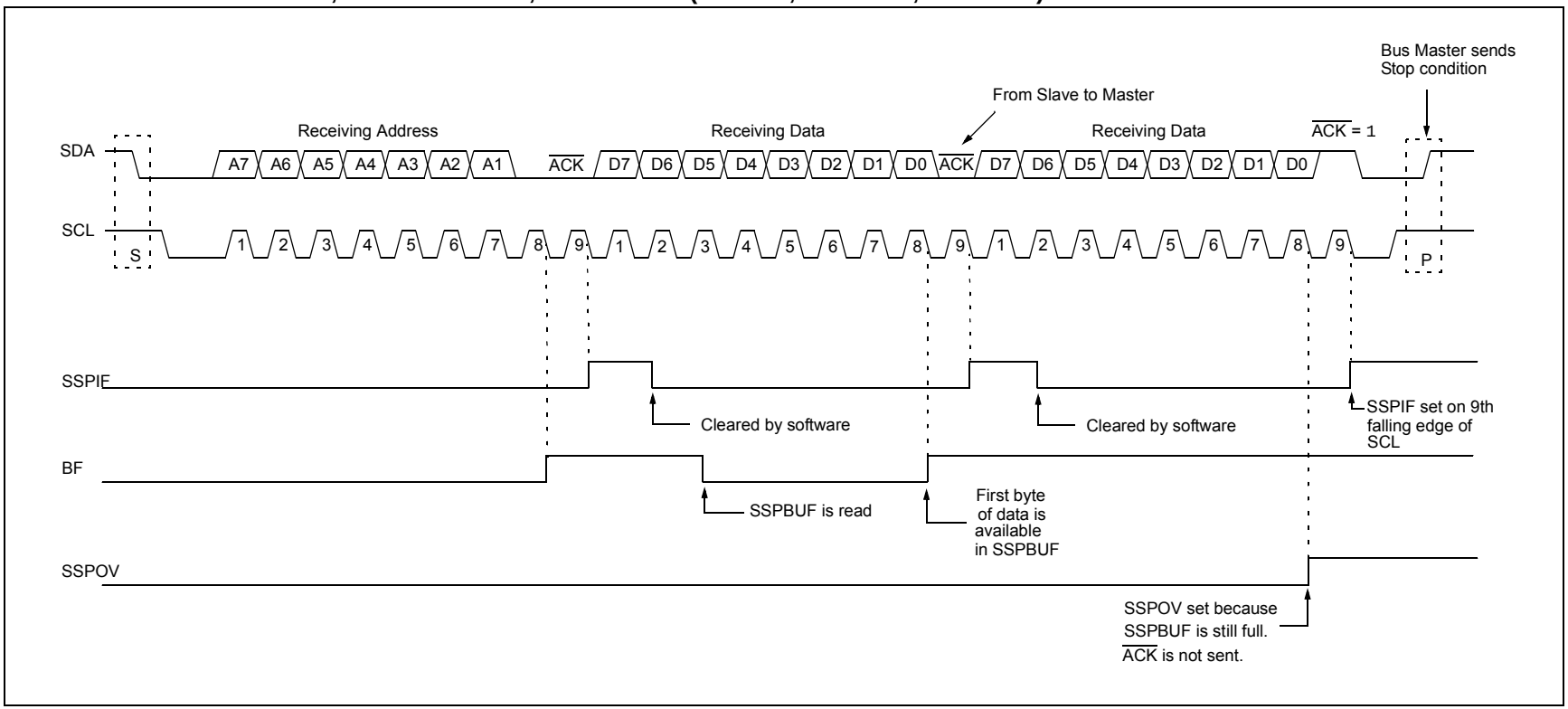


FIGURE 18-15: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)

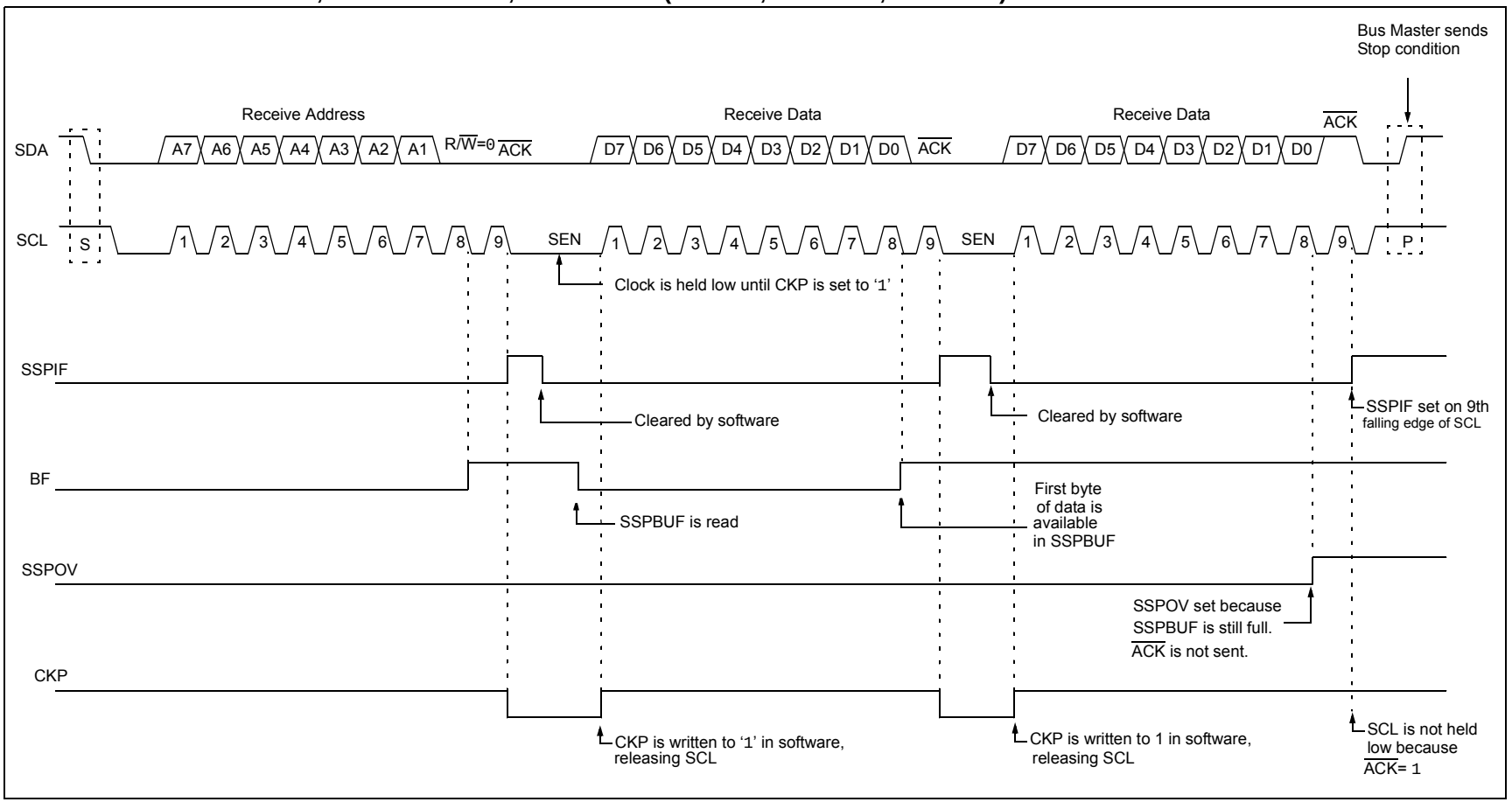


FIGURE 18-16: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)

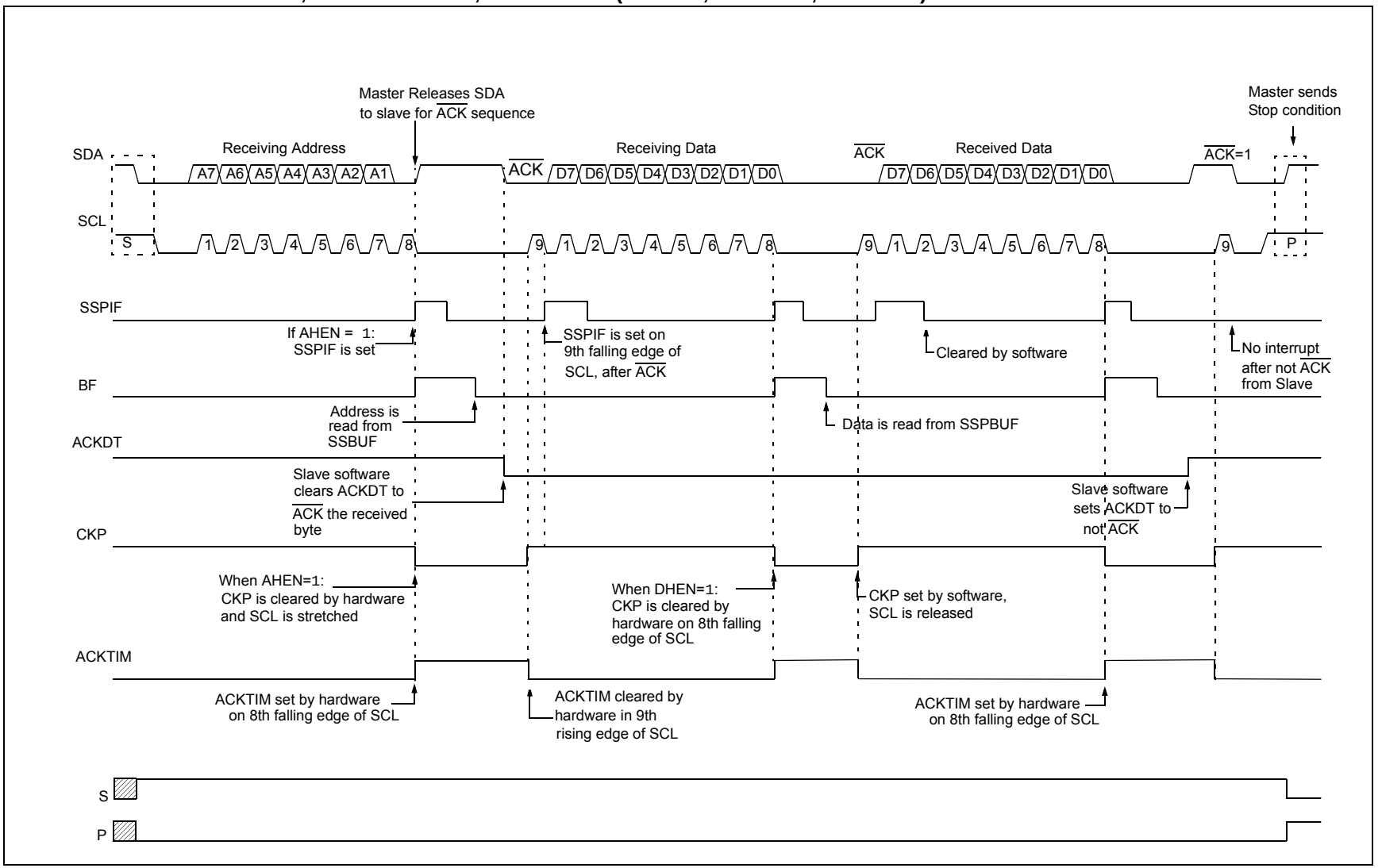
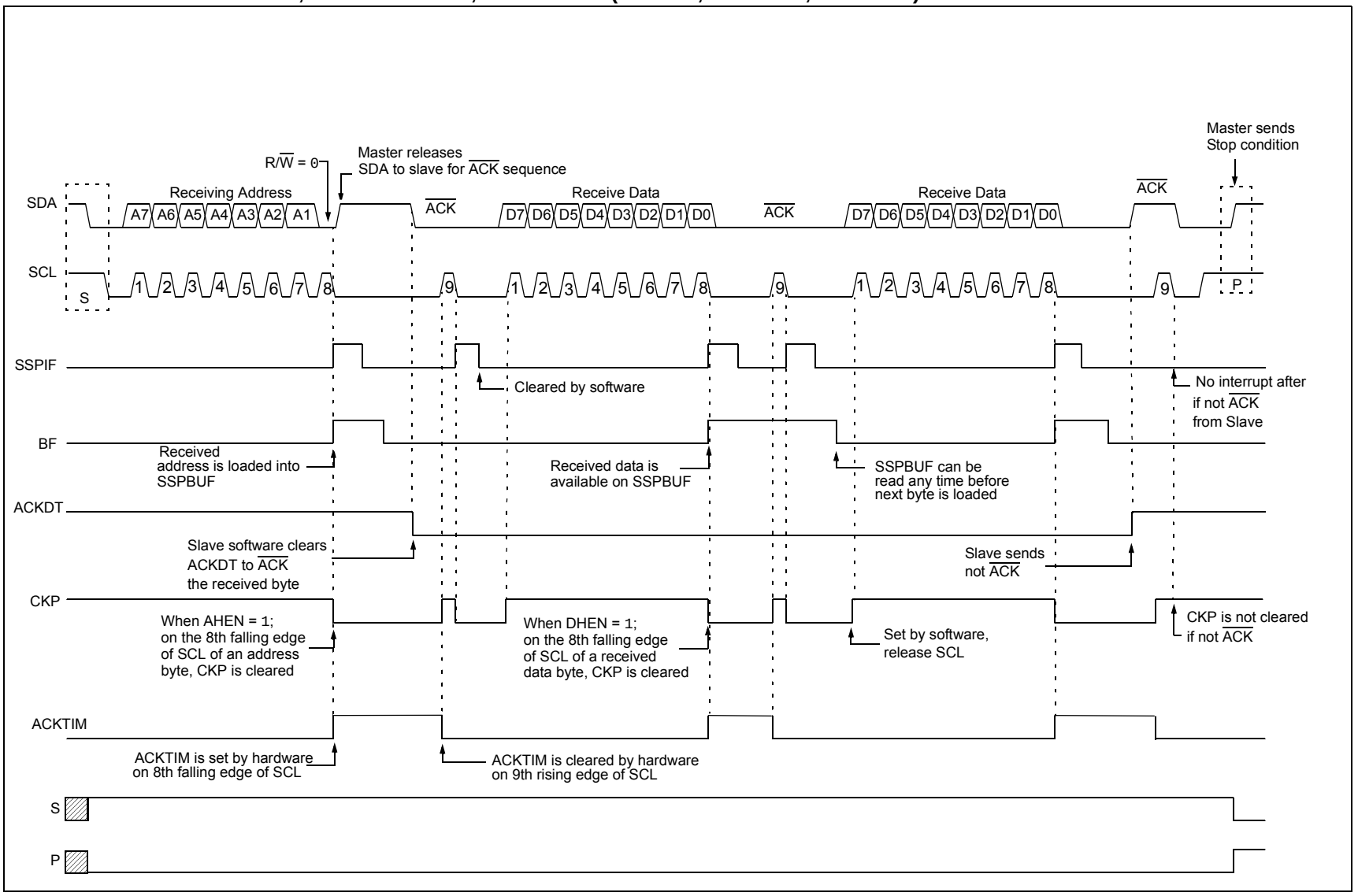


FIGURE 18-17: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)



18.5.3 SLAVE TRANSMISSION

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register, and an $\overline{ACK}$ pulse is sent by the slave on the ninth bit.

Following the $\overline{ACK}$, slave hardware clears the CKP bit and the SCL pin is held low (see **Section 18.5.6 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPBUF register which also loads the SSPSR register. Then the SCL pin should be released by setting the CKP bit of the SSPCON1 register. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time.

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. This $\overline{ACK}$ value is copied to the ACKSTAT bit of the SSPCON2 register. If ACKSTAT is set (not $\overline{ACK}$), then the data transfer is complete. In this case, when the not $\overline{ACK}$ is latched by the slave, the slave goes idle and waits for another occurrence of the Start bit. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPBUF register. Again, the SCL pin must be released by setting bit CKP.

An MSSP1 interrupt is generated for each data transfer byte. The SSPIF bit must be cleared by software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

18.5.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDA line. If a bus collision is detected and the SBCDE bit of the SSPCON3 register is set, the BCLIF bit of the PIRx register is set. Once a bus collision is detected, the slave goes Idle and waits to be addressed again. User software can use the BCLIF bit to handle a slave bus collision.

18.5.3.2 7-Bit Transmission

A master device can transmit a read request to a slave, and then clock data out of the slave. The list below outlines what software for a slave will need to do to accomplish a standard transmission. Figure 18-18 can be used as a reference to this list.

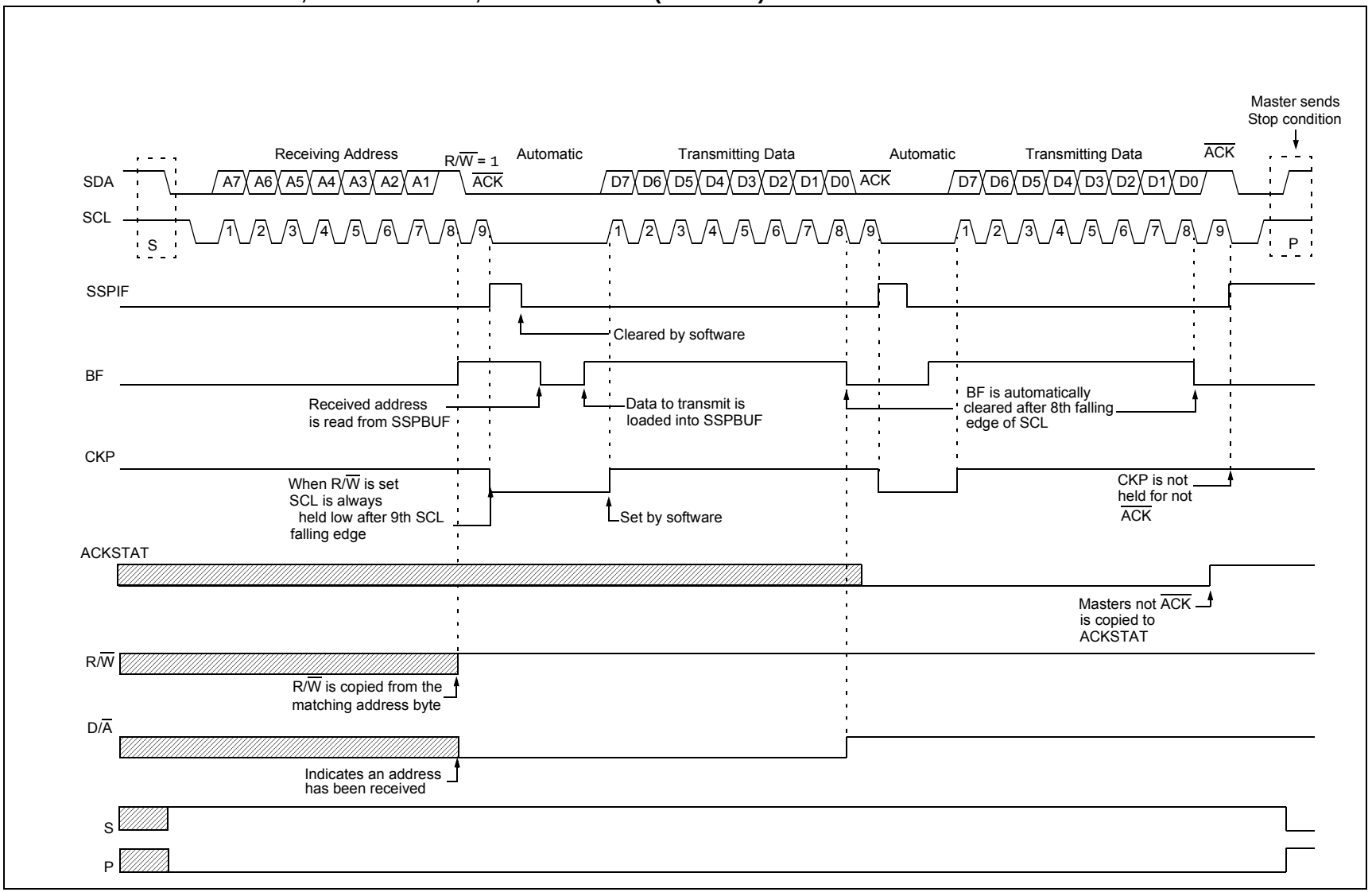
1. Master sends a Start condition on SDA and SCL.
2. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{W}$ bit set is received by the Slave setting SSPIF bit.
4. Slave hardware generates an $\overline{ACK}$ and sets SSPIF.
5. SSPIF bit is cleared by user.
6. Software reads the received address from SSPBUF, clearing BF.
7. R/$\overline{W}$ is set so CKP was automatically cleared after the $\overline{ACK}$.
8. The slave software loads the transmit data into SSPBUF.
9. CKP bit is set releasing SCL, allowing the master to clock the data out of the slave.
10. SSPIF is set after the $\overline{ACK}$ response from the master is loaded into the ACKSTAT register.
11. SSPIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

Note 1: If the master $\overline{ACK}$s the clock will be stretched.

2: ACKSTAT is the only bit updated on the rising edge of SCL (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not $\overline{ACK}$; the clock is not held, but SSPIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

FIGURE 18-18: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)



18.5.3.3 7-Bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPCON3 register enables additional clock stretching and interrupt generation after the 8th falling edge of a received matching address. Once a matching address has been clocked in, CKP is cleared and the SSPIF interrupt is set.

Figure 18-19 displays a standard waveform of a 7-bit Address Slave Transmission with AHEN enabled.

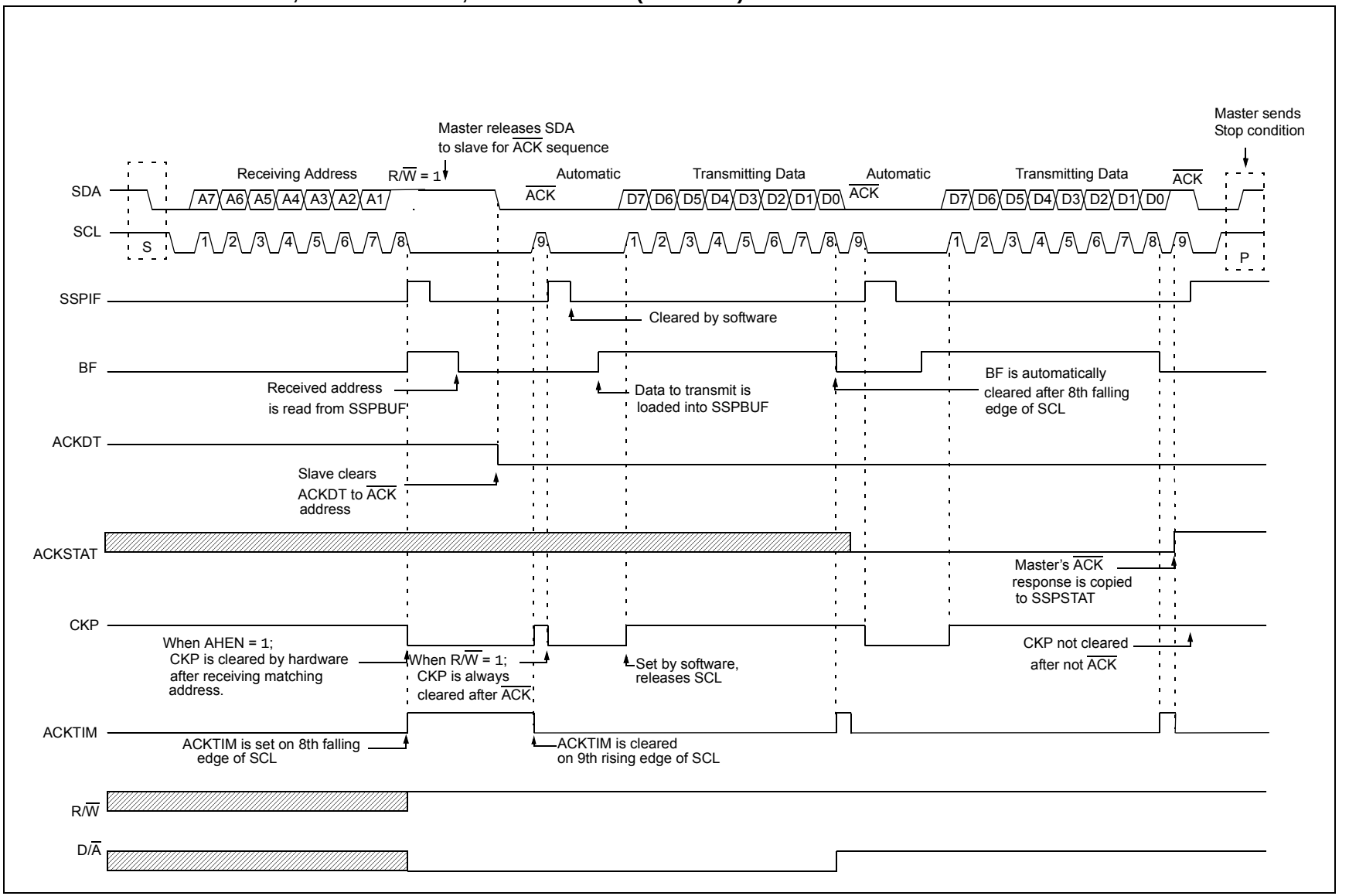
1. Bus starts Idle.
2. Master sends Start condition; the S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with R/$\overline{W}$ bit set. After the 8th falling edge of the SCL line the CKP bit is cleared and SSPIF interrupt is generated.
4. Slave software clears SSPIF.
5. Slave software reads ACKTIM bit of SSPCON3 register, and R/$\overline{W}$ and D/$\overline{A}$ of the SSPSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to $\overline{ACK}$ or not $\overline{ACK}$ and sets the ACKDT bit of the SSPCON2 register accordingly.
8. Slave sets the CKP bit releasing SCL.
9. Master clocks in the $\overline{ACK}$ value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPIF after the $\overline{ACK}$ if the R/$\overline{W}$ bit is set.
11. Slave software clears SSPIF.
12. Slave loads value to transmit to the master into SSPBUF setting the BF bit.

Note: SSPBUF cannot be loaded until after the $\overline{ACK}$.

13. Slave sets CKP bit releasing the clock.
14. Master clocks out the data from the slave and sends an $\overline{ACK}$ value on the 9th SCL pulse.
15. Slave hardware copies the $\overline{ACK}$ value into the ACKSTAT bit of the SSPCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not $\overline{ACK}$ the slave releases the bus allowing the master to send a Stop and end the communication.

Note: Master must send a not $\overline{ACK}$ on the last byte to ensure that the slave releases the SCL line to receive a Stop.

FIGURE 18-19: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)



18.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSP1 module configured as an I^2C Slave in 10-bit Addressing mode.

Figure 18-20 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 SSPSTAT is set; SSPIF 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 SSPSTAT register is set.
4. Slave sends $\overline{ACK}$ and SSPIF is set.
5. Software clears the SSPIF bit.
6. Software reads received address from SSPBUF clearing the BF flag.
7. Slave loads low address into SSPADD, releasing SCL.
8. Master sends matching low address byte to the Slave; UA bit is set.

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

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

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

10. Slave clears SSPIF.
11. Slave reads the received matching address from SSPBUF clearing BF.
12. Slave loads high address into SSPADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{ACK}$ on the 9th SCL pulse; SSPIF is set.
14. If SEN bit of SSPCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPIF.
16. Slave reads the received byte from SSPBUF 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.

18.5.5 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 SSPADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. Figure 18-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 18-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

FIGURE 18-20: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)

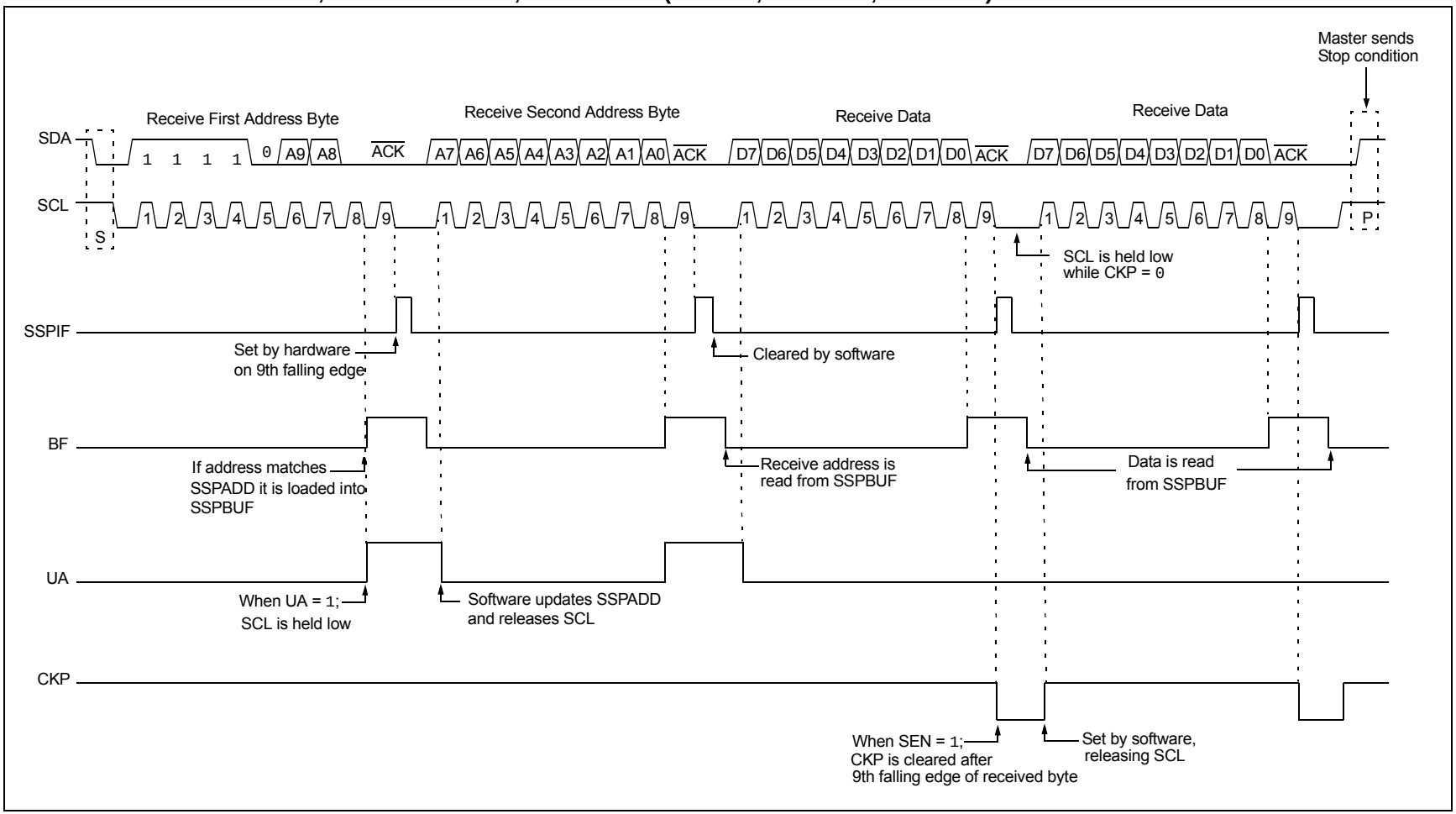


FIGURE 18-21: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 0)

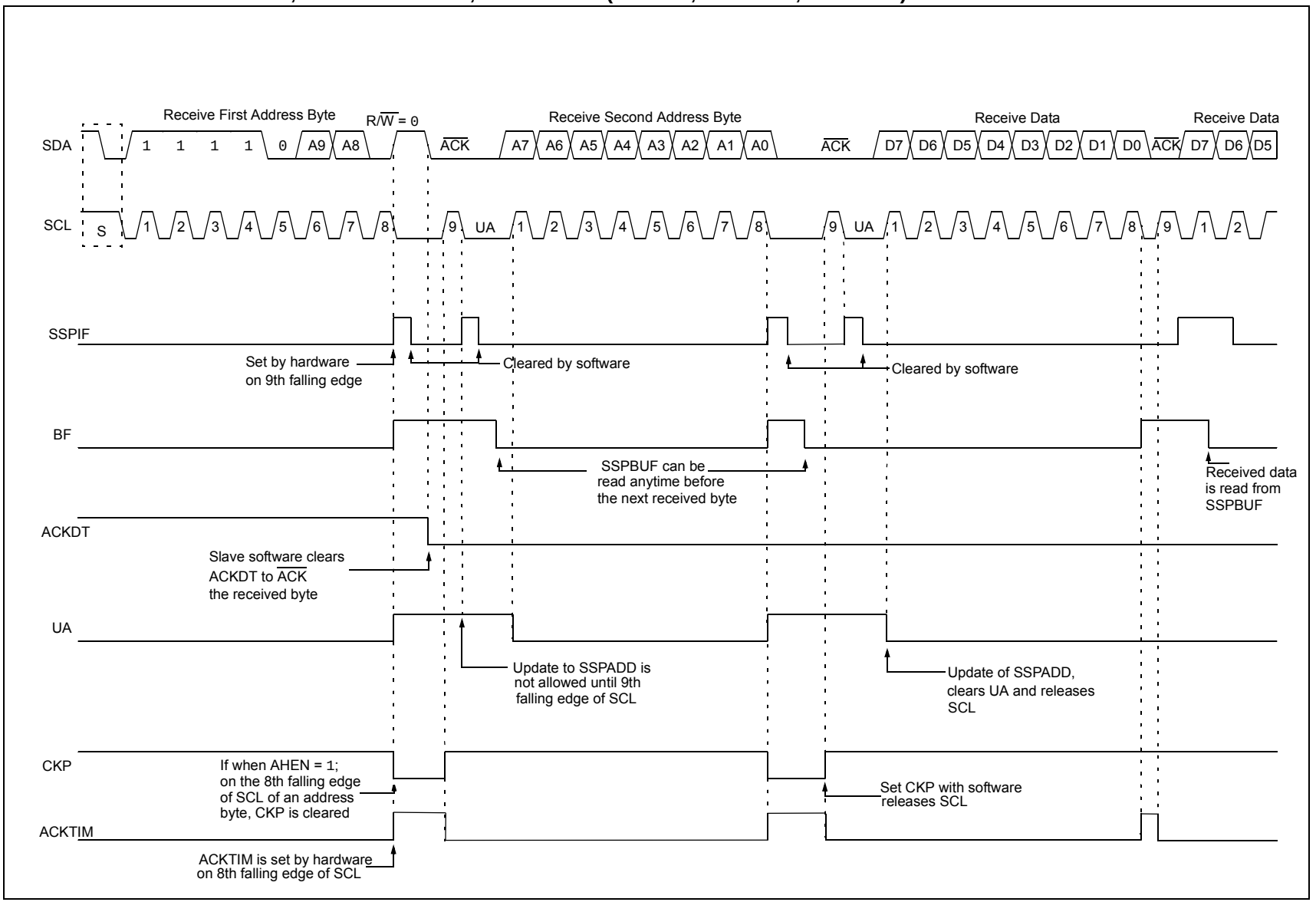
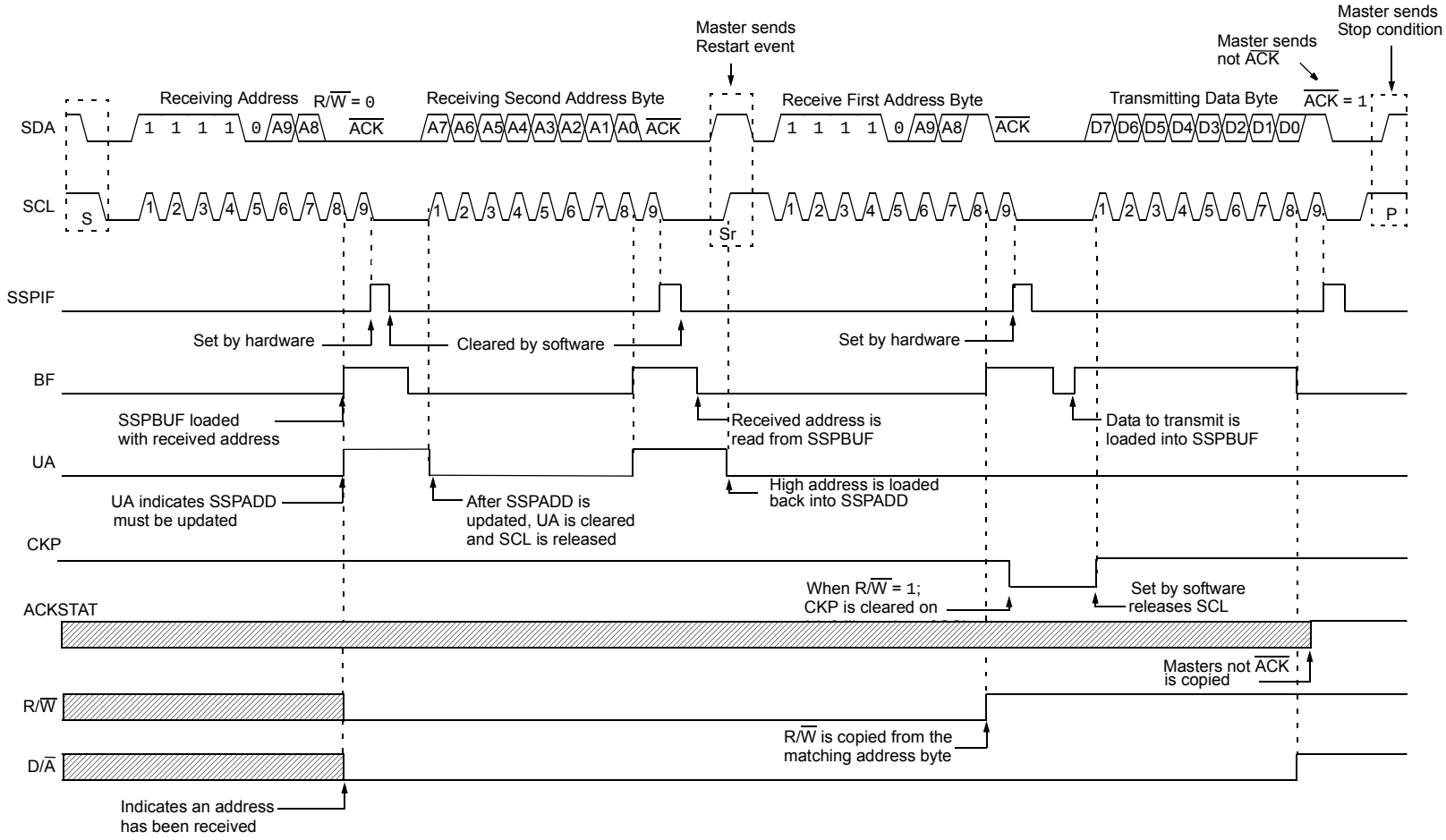


FIGURE 18-22: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)



18.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCL line low, effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data, it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCL.

The CKP bit of the SSPCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. Setting CKP will release SCL and allow more communication.

18.5.6.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the R/$\overline{W}$ bit of SSPSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPBUF with data to transfer to the master. If the SEN bit of SSPCON2 is set, the slave hardware will always stretch the clock after the $\overline{ACK}$ sequence. Once the slave is ready; CKP is set by software and communication resumes.

Note 1: The BF bit has no effect on if the clock will be stretched or not. This is different than previous versions of the module that would not stretch the clock, clear CKP, if SSPBUF was read before the 9th falling edge of SCL.

2: Previous versions of the module did not stretch the clock for a transmission if SSPBUF was loaded before the 9th falling edge of SCL. It is now always cleared for read requests.

18.5.6.2 10-Bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set, the clock is always stretched. This is the only time the SCL is stretched without CKP being cleared. SCL is released immediately after a write to SSPADD.

Note: Previous versions of the module did not stretch the clock if the second address byte did not match.

18.5.6.3 Byte NACKing

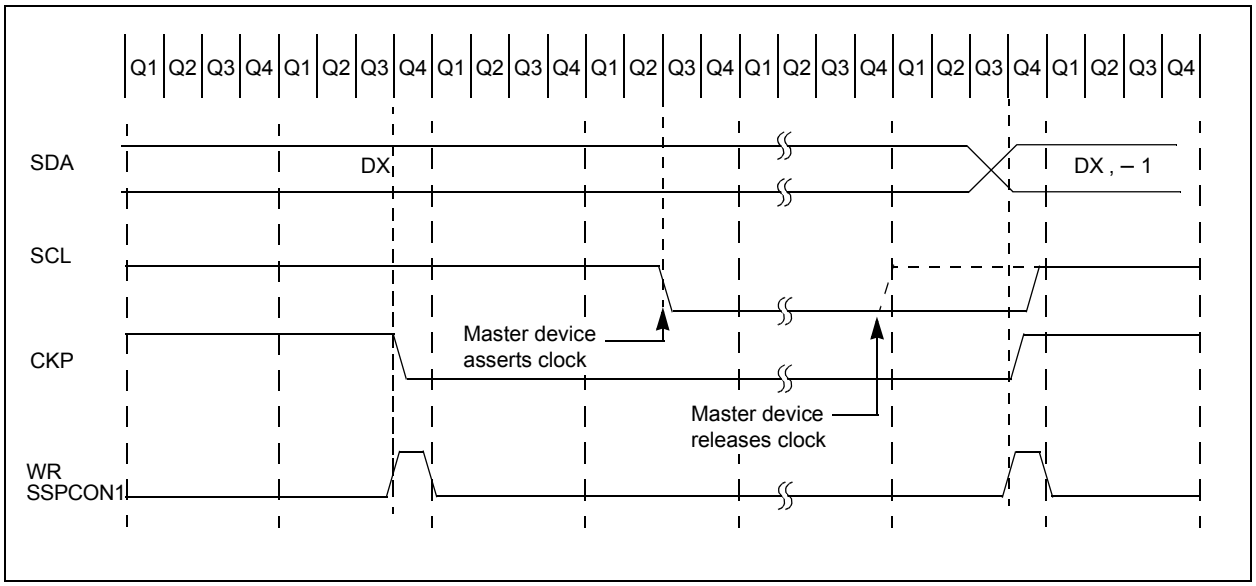
When AHEN bit of SSPCON3 is set; CKP is cleared by hardware after the 8th falling edge of SCL for a received matching address byte. When DHEN bit of SSPCON3 is set; CKP is cleared after the 8th falling edge of SCL for received data.

Stretching after the 8th falling edge of SCL allows the slave to look at the received address or data and decide if it wants to ACK the received data.

18.5.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I^2C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I^2C bus have released SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 18-23).

FIGURE 18-23: CLOCK SYNCHRONIZATION TIMING



18.5.8 GENERAL CALL ADDRESS SUPPORT

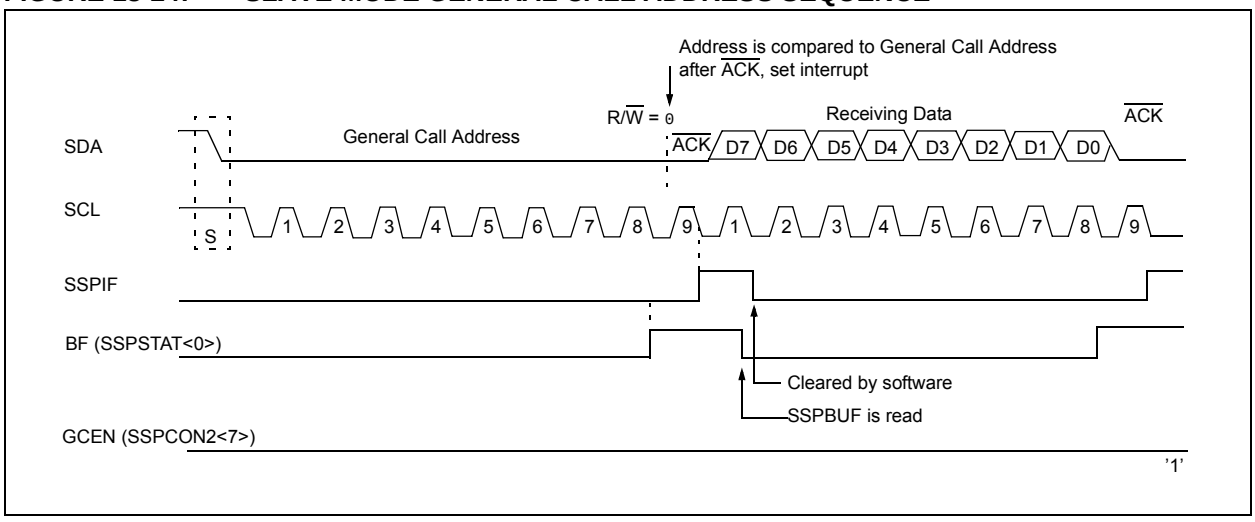
The addressing procedure for the I^2C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master device. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge.

The general call address is a reserved address in the I^2C protocol, defined as address 0x00. When the GCEN bit of the SSPCON2 register is set, the slave module will automatically $\overline{ACK}$ the reception of this address regardless of the value stored in SSPADD. After the slave clocks in an address of all zeros with the R/$\overline{W}$ bit clear, an interrupt is generated and slave software can read SSPBUF and respond. Figure 18-24 shows a general call reception sequence.

In 10-bit Address mode, the UA bit will not be set on the reception of the general call address. The slave will prepare to receive the second byte as data, just as it would in 7-bit mode.

If the AHEN bit of the SSPCON3 register is set, just as with any other address reception, the slave hardware will stretch the clock after the 8th falling edge of SCL. The slave must then set its ACKDT value and release the clock with communication progressing as it would normally.

FIGURE 18-24: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE



18.5.9 SSP MASK REGISTER

An SSP Mask (SSPMSK) register (Register 18-5) is available in I^2C Slave mode as a mask for the value held in the SSPSR register during an address comparison operation. A zero ('0') bit in the SSPMSK register has the effect of making the corresponding bit of the received address a "don't care".

This register is reset to all '1's upon any Reset condition and, therefore, has no effect on standard SSP operation until written with a mask value.

The SSP Mask register is active during:

- 7-bit Address mode: address compare of A<7:1>.
- 10-bit Address mode: address compare of A<7:0> only. The SSP mask has no effect during the reception of the first (high) byte of the address.

18.6 I²C MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in the SSPCON1 register and by setting the SSPEN bit. In Master mode, the SDA and SCK pins must be configured as inputs. The MSSP peripheral hardware will override the output driver TRIS controls when necessary to drive the pins low.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP1 module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop bit condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDA and SCL lines.

The following events will cause the SSP Interrupt Flag bit, SSPIF, to be set (SSP interrupt, if enabled):

- Start condition detected
- Stop condition detected
- Data transfer byte transmitted/received
- Acknowledge transmitted/received
- Repeated Start generated

Note 1: The MSSP1 module, when configured in I²C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur

2: When in Master mode, Start/Stop detection is masked and an interrupt is generated when the SEN/PEN bit is cleared and the generation is complete.

18.6.1 I²C MASTER MODE OPERATION

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

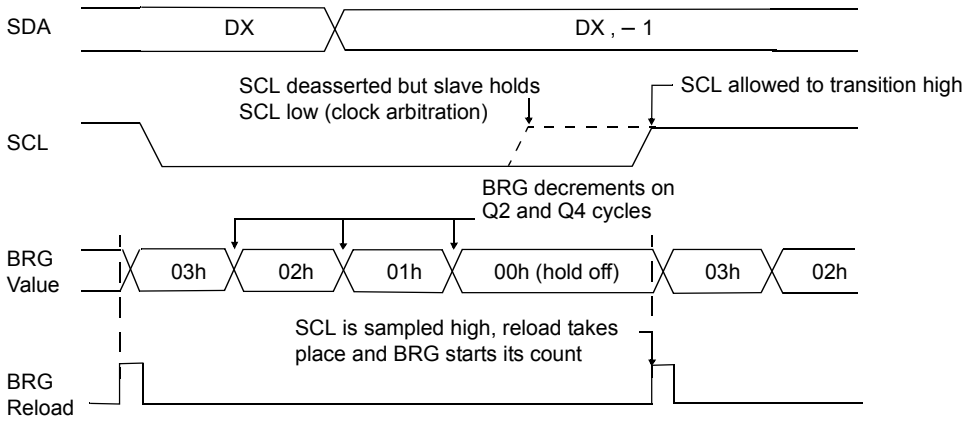
In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCL. See **Section 18.7 "Baud Rate Generator"** for more detail.

18.6.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 18-25).

FIGURE 18-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



18.6.3 WCOL STATUS FLAG

If the user writes the SSPBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur). Any time the WCOL bit is set it indicates that an action on SSPBUF was attempted while the module was not Idle.

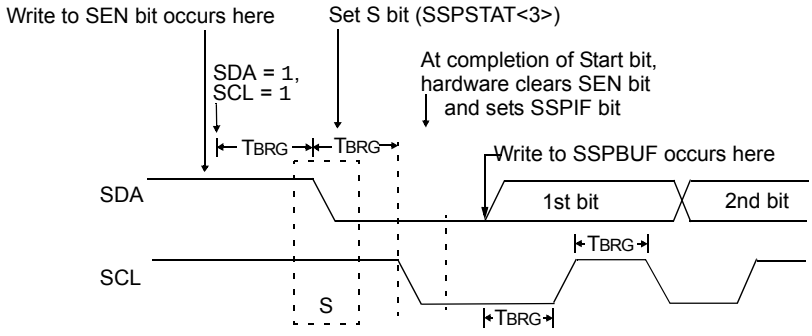
Note:	Because queuing of events is not allowed, writing to the lower five bits of SSPCON2 is disabled until the Start condition is complete.

18.6.4 I^2C MASTER MODE START CONDITION TIMING

To initiate a Start condition (Figure 18-26), 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, BCLIF, is set, the Start condition is aborted and the I^2C module is reset into its Idle state.
2:	The Philips I^2C Specification states that a bus collision cannot occur on a Start.

FIGURE 18-26: FIRST START BIT TIMING



18.6.5 I²C MASTER MODE REPEATED START CONDITION TIMING

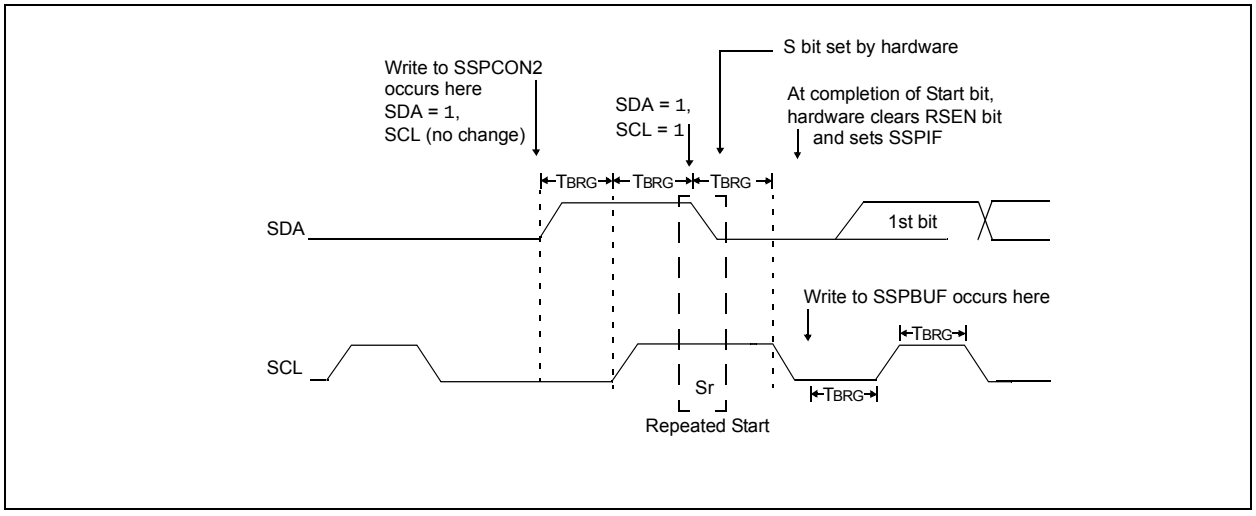
A Repeated Start condition (Figure 18-27) occurs when the RSEN bit of the SSPCON2 register is programmed high and the master state machine is no longer active. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. SCL is asserted low. Following this, the RSEN bit of the SSPCON2 register will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit of the SSPSTAT register will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

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

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

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

FIGURE 18-27: REPEAT START CONDITION WAVEFORM



18.6.6 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 SSPBUF 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 $\overline{\text{ACK}}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{\text{ACK}}$ 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 SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 18-28).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{\text{W}}$ 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 $\overline{\text{ACK}}$ bit is loaded into the ACKSTAT Status bit of the SSPCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

18.6.6.1 BF Status Flag

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

18.6.6.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR 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.

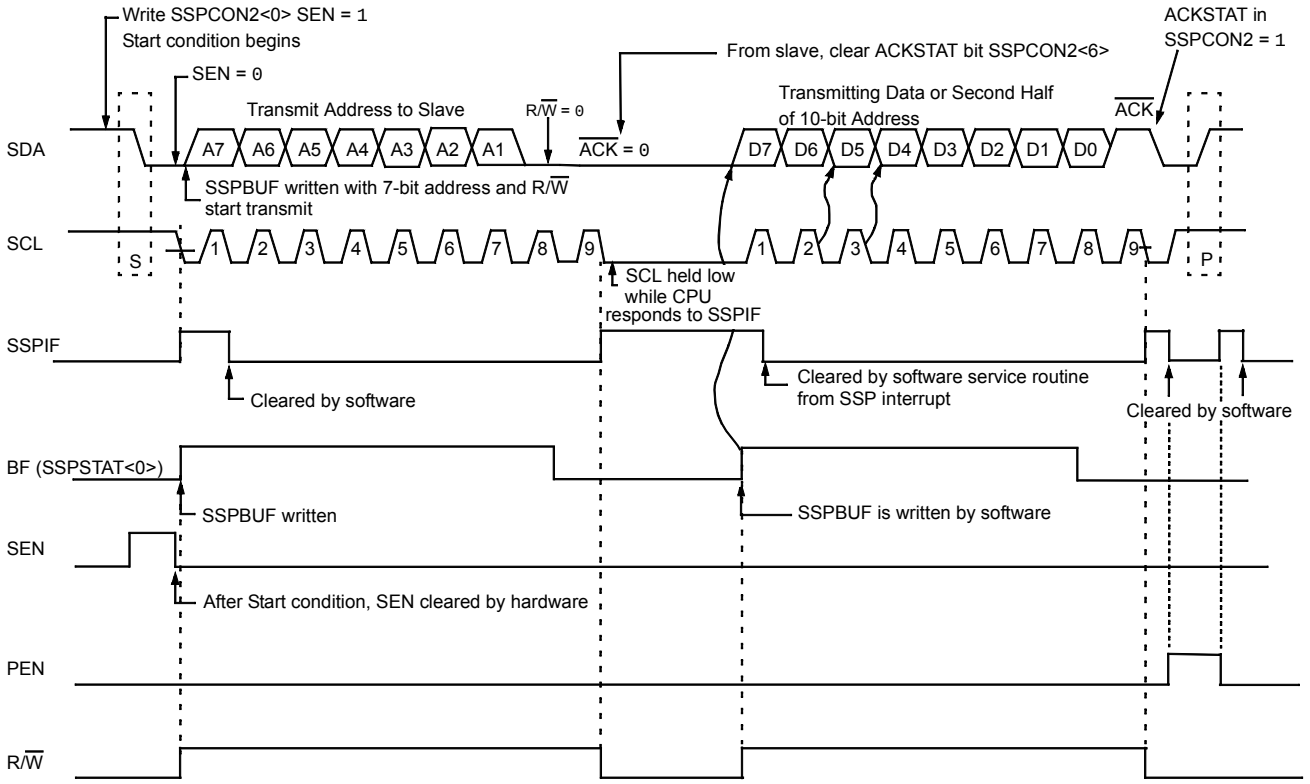
18.6.6.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPCON2 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.

18.6.6.4 Typical Transmit Sequence

1. The user generates a Start condition by setting the SEN bit of the SSPCON2 register.
2. SSPIF is set by hardware on completion of the Start.
3. SSPIF is cleared by software.
4. The MSSP1 module will wait the required start time before any other operation takes place.
5. The user loads the SSPBUF 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 SSPBUF is written to.
7. The MSSP1 module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
8. The MSSP1 module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
9. The user loads the SSPBUF with eight bits of data.
10. Data is shifted out the SDA pin until all eight bits are transmitted.
11. The MSSP1 module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 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 SSPCON2 register. Interrupt is generated once the Stop/Restart condition is complete.

FIGURE 18-28: I^2C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)



18.6.7 I²C MASTER MODE RECEPTION

Master mode reception (Figure 18-29) is enabled by programming the Receive Enable bit, RCEN bit of the SSPCON2 register.

Note:	The MSSP1 module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP1 is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPCON2 register.

18.6.7.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

18.6.7.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when eight bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

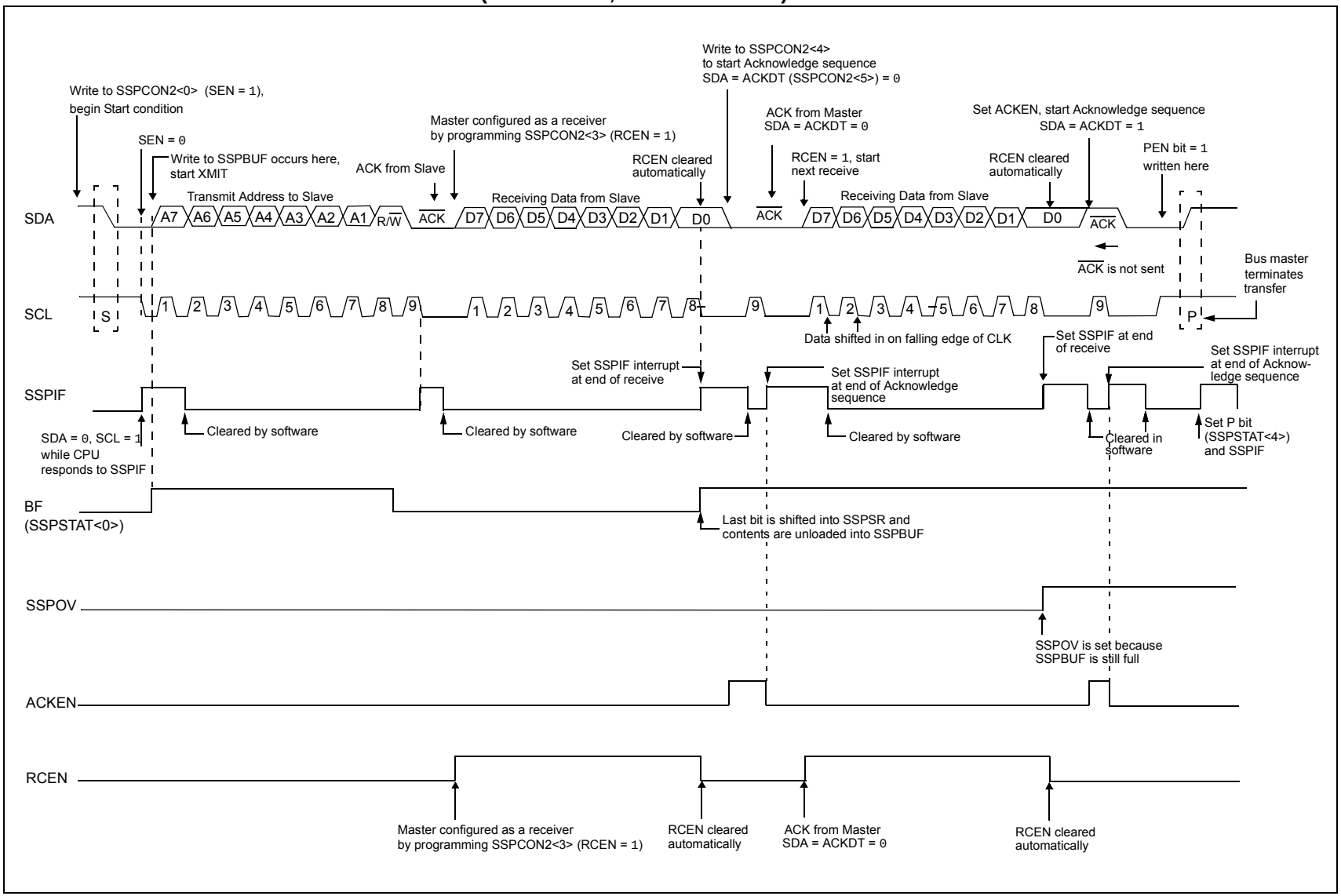
18.6.7.3 WCOL Status Flag

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

18.6.7.4 Typical Receive Sequence

1. The user generates a Start condition by setting the SEN bit of the SSPCON2 register.
2. SSPIF is set by hardware on completion of the Start.
3. SSPIF is cleared by software.
4. User writes SSPBUF with the slave address to transmit and the R/$\overline{W}$ bit set.
5. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPBUF is written to.
6. The MSSP1 module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
7. The MSSP1 module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
8. User sets the RCEN bit of the SSPCON2 register and the Master clocks in a byte from the slave.
9. After the 8th falling edge of SCL, SSPIF and BF are set.
10. User clears SSPIF and reads the received byte from SSPBUF, which clears the BF flag.
11. User either clears the SSPCON2 ACKDT bit to receive another byte or sets the ACKDT bit to suppress further data and then initiates the acknowledge sequence by setting the ACKEN bit.
12. Master's $\overline{ACK}$ or not $\overline{ACK}$ is clocked out to the slave and SSPIF is set.
13. User clears SSPIF.
14. Steps 8-13 are repeated for each received byte from the slave.
15. If the ACKDT bit was set in step 11, then the user can send a Stop to release the bus.

FIGURE 18-29: I^2C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



18.6.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPCON2 register. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP1 module then goes into Idle mode (Figure 18-30).

18.6.8.1 WCOL Status Flag

If the user writes the SSPBUF when an Acknowledge sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

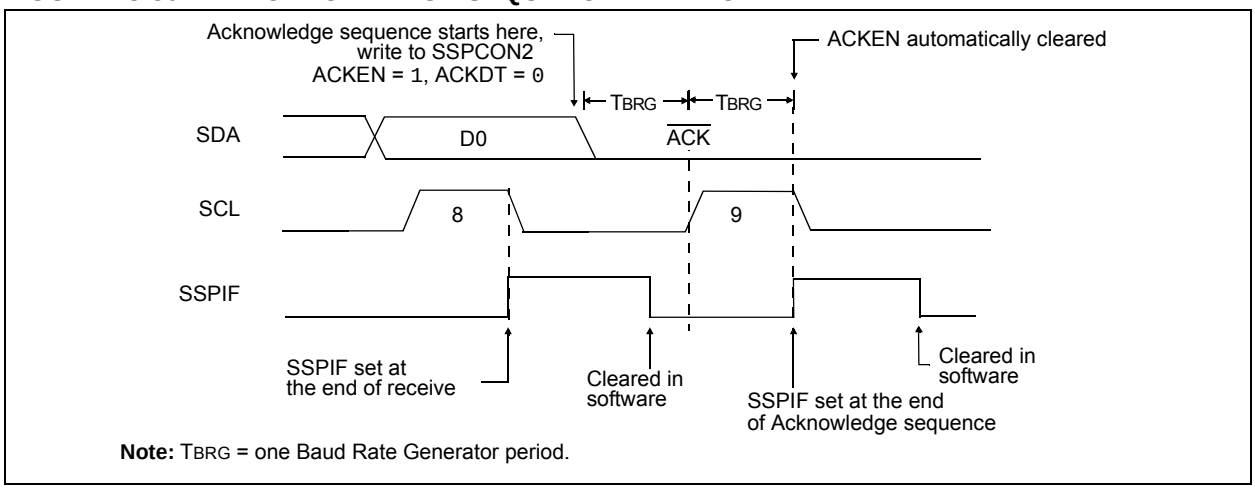
18.6.9 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPCON2 register. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit of the SSPSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 18-31).

18.6.9.1 WCOL Status Flag

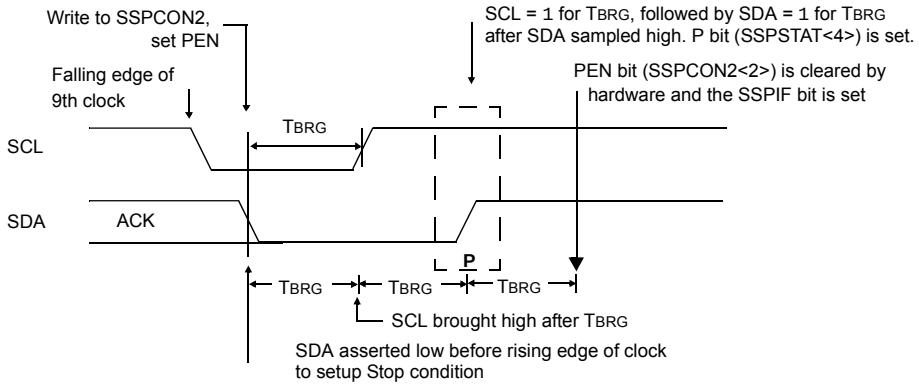
If the user writes the SSPBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 18-30: ACKNOWLEDGE SEQUENCE WAVEFORM



Note: TBRG = one Baud Rate Generator period.

FIGURE 18-31: STOP CONDITION RECEIVE OR TRANSMIT MODE



Note: TBRG = one Baud Rate Generator period.

18.6.10 SLEEP OPERATION

While in Sleep mode, the I^2C slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP1 interrupt is enabled).

18.6.11 EFFECTS OF A RESET

A Reset disables the MSSP1 module and terminates the current transfer.

18.6.12 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP1 module is disabled. Control of the I^2C bus may be taken when the P bit of the SSPSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

18.6.13 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the I^2C port to its Idle state (Figure 18-32).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I^2C bus is free, the user can resume communication by asserting a Start condition.

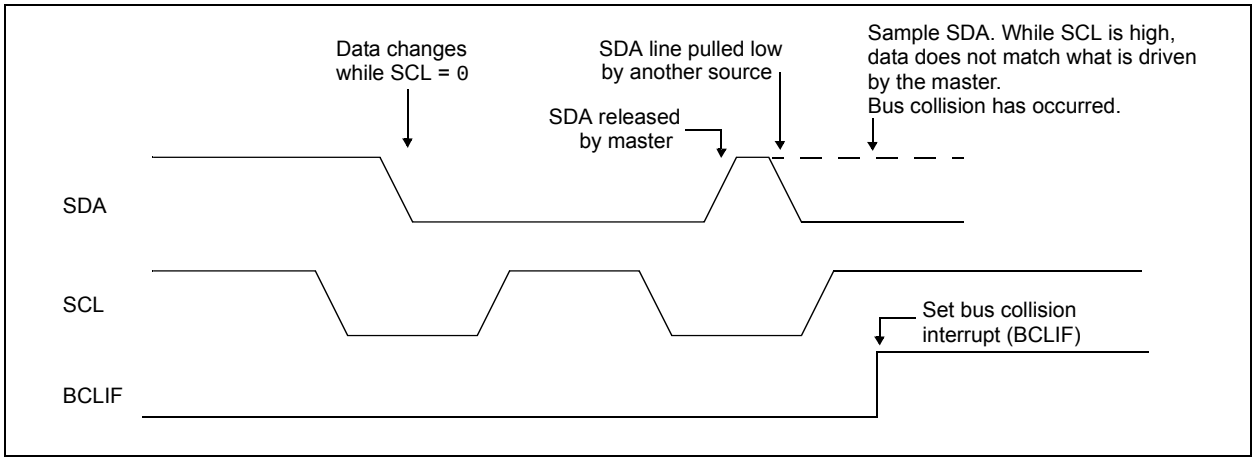
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I^2C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I^2C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is Idle and the S and P bits are cleared.

FIGURE 18-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



18.6.13.1 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 18-33).

b) SCL is sampled low before SDA is asserted low (Figure 18-34).

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 BCLIF flag is set and
- the MSSP1 module is reset to its Idle state (Figure 18-33).

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 18-35). 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 18-33: BUS COLLISION DURING START CONDITION (SDA ONLY)

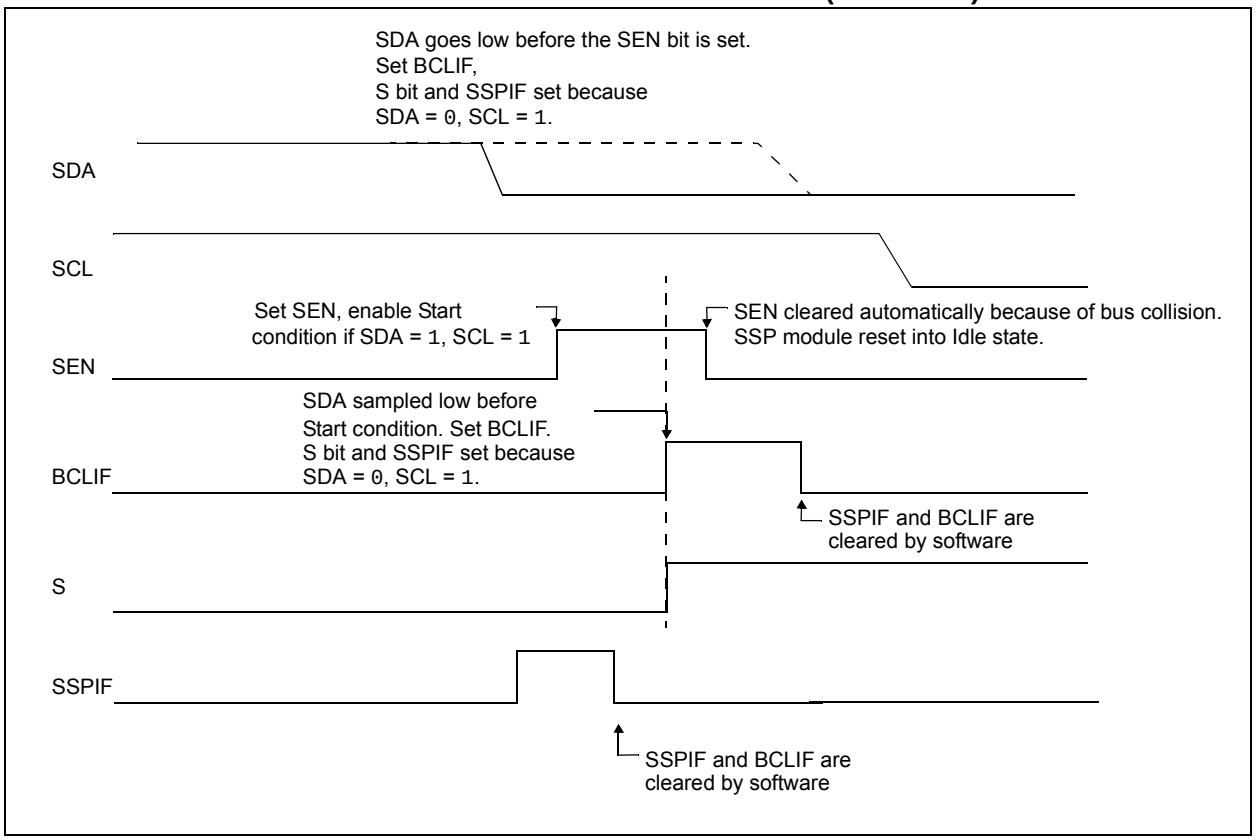


FIGURE 18-34: BUS COLLISION DURING START CONDITION (SCL = 0)

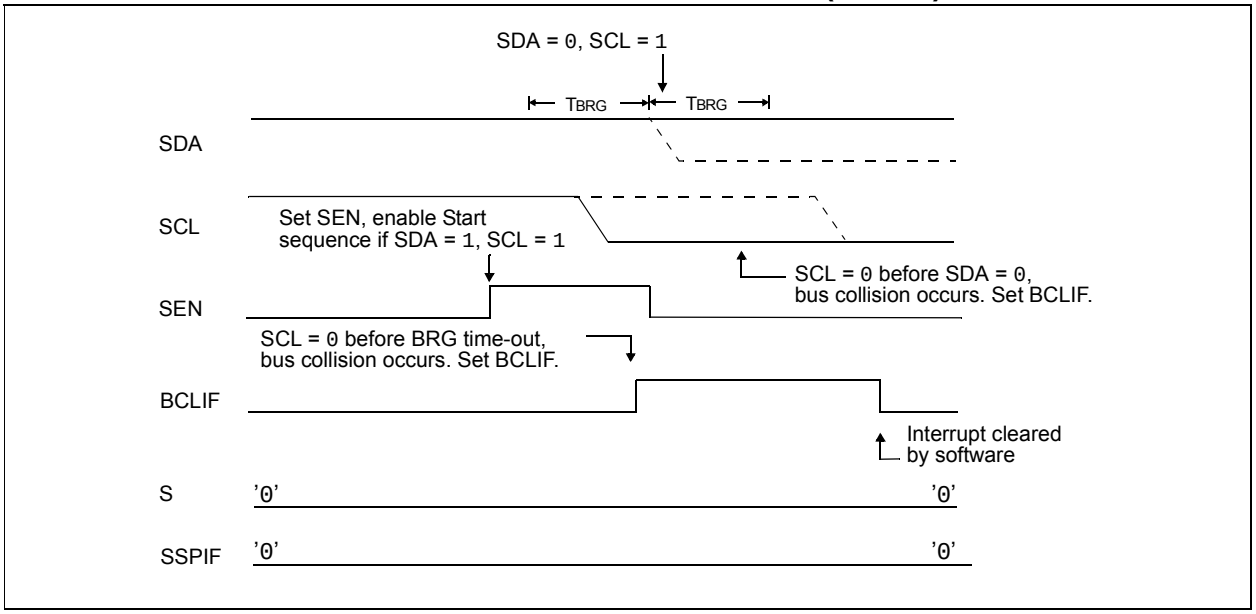
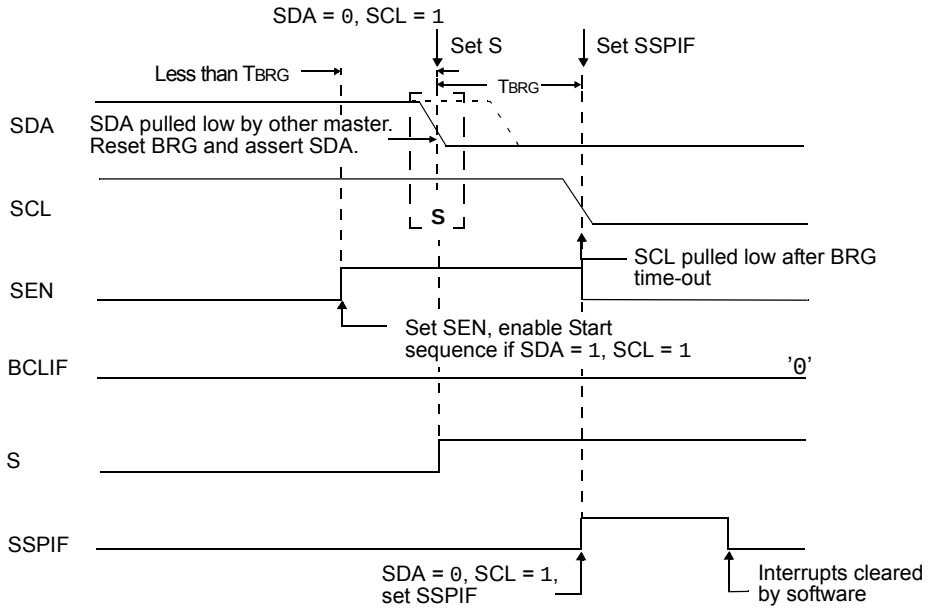


FIGURE 18-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



18.6.13.2 Bus Collision During a Repeated Start Condition

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

a) A low level is sampled on SDA when SCL goes from low level to high level (Case 1).

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1' (Case 2).

When the user releases SDA and the pin is allowed to float high, the BRG is loaded with SSPADD and counts down to zero. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 18-36). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 18-37.

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

FIGURE 18-36: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

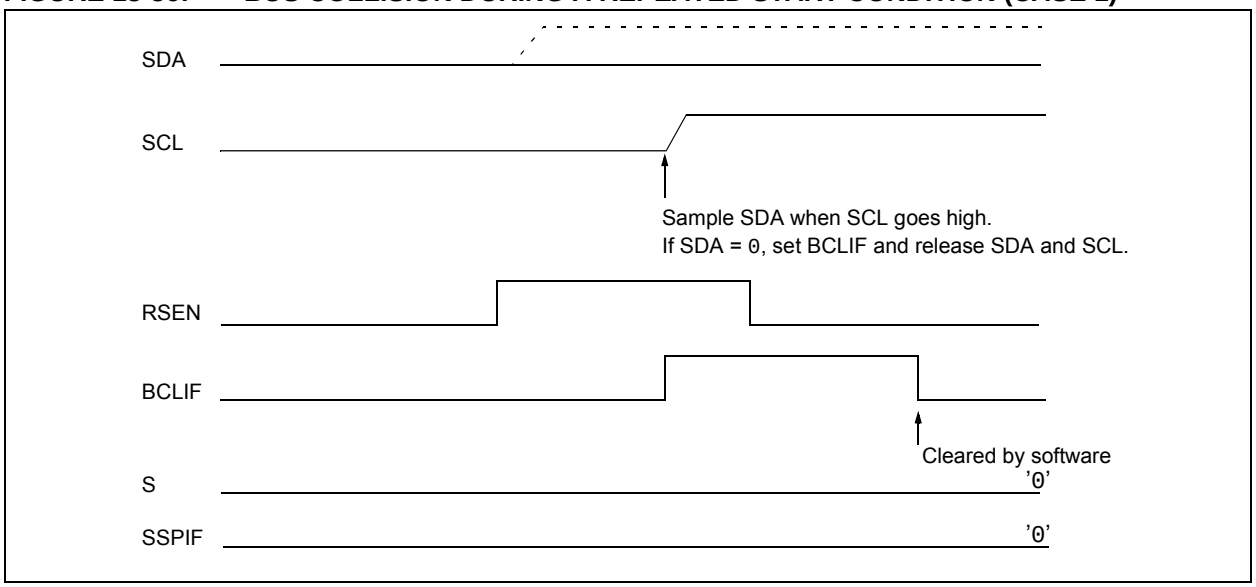
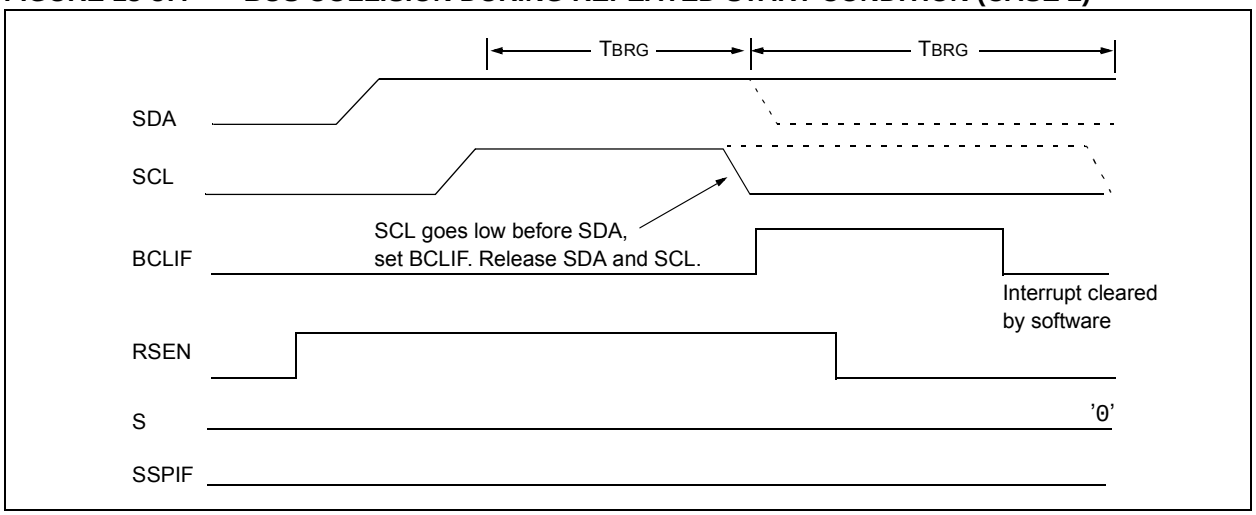


FIGURE 18-37: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



18.6.13.3 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 (Case 1).

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high (Case 2).

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 18-38). 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 18-39).

FIGURE 18-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)

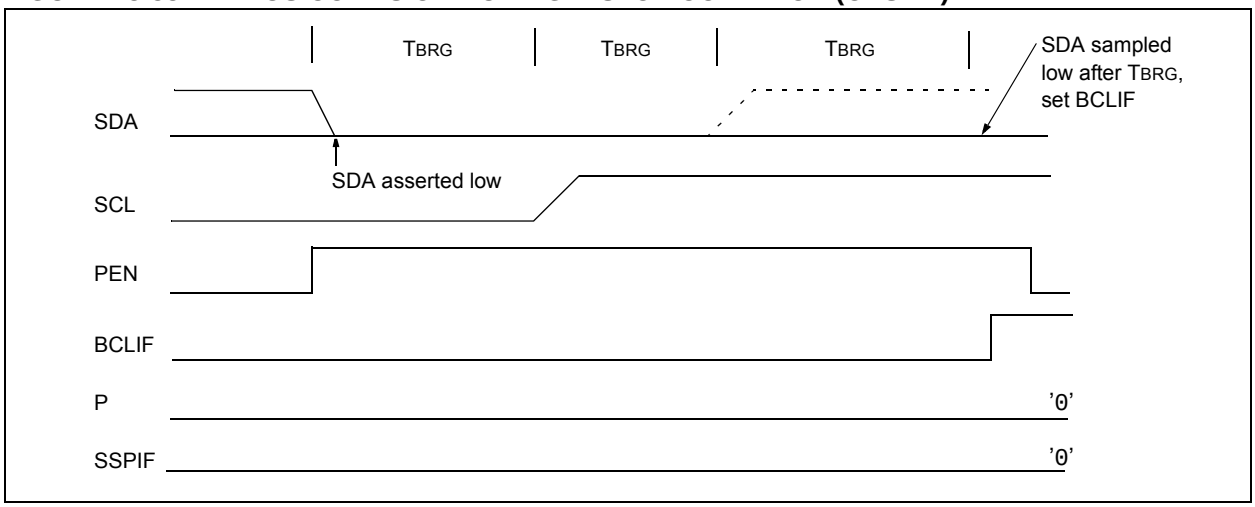


FIGURE 18-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)

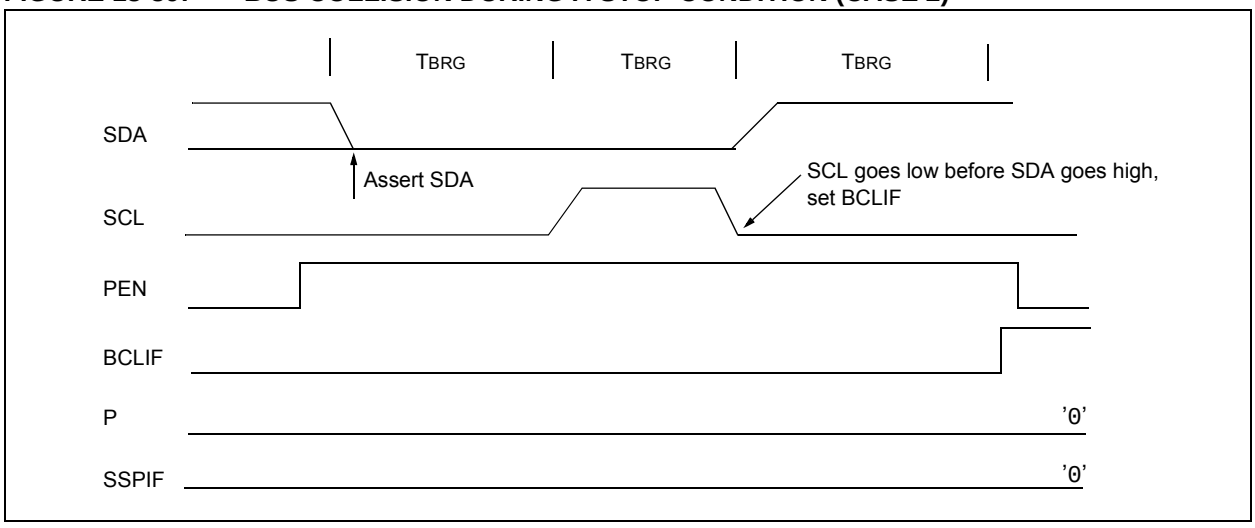


TABLE 18-3: SUMMARY OF REGISTERS ASSOCIATED WITH I^2C OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	57
PIE1	—	ADIE	—	—	SSPIE	—	—	—	58
PIE2	—	—	—	—	BCLIE	—	—	—	59
PIR1	—	ADIF	—	—	SSPIF	—	—	—	60
PIR2	—	—	—	—	BCLIF	—	—	—	61
SSPADD	ADD<7:0>								185
SSPBUF	MSSP1 Receive Buffer/Transmit Register								137*
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				182
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	183
SSPCON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN	184
SSPMSK	MSK<7:0>								185
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	181
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	87

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by the MSSP module in I^2C mode.

* Page provides register information.

Note 1: Unimplemented, read as '1'.

18.7 BAUD RATE GENERATOR

The MSSP1 module has a Baud Rate Generator available for clock generation in both I^2C and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPADD register (Register 18-6). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting down.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

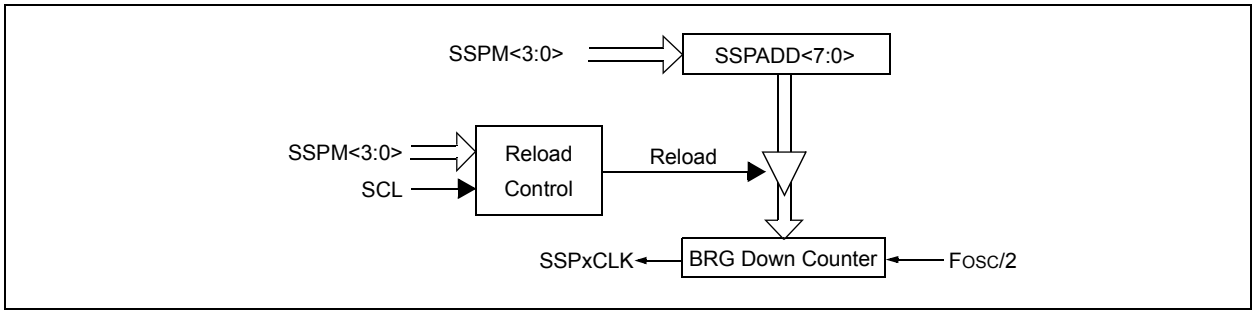
An internal signal “Reload” in Figure 18-40 triggers the value from SSPADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode the MSSP1 is being operated in.

Table 18-4 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

EQUATION 18-1:

$$F_{CLOCK} = \frac{F_{OSC}}{(SSPxADD + 1)(4)}$$

FIGURE 18-40: BAUD RATE GENERATOR BLOCK DIAGRAM



Note: Values of 0x00, 0x01 and 0x02 are not valid for SSPADD when used as a Baud Rate Generator for I^2C. This is an implementation limitation.

TABLE 18-4: MSSP1 CLOCK RATE W/BRG

FOSC	FCY	BRG Value	FCLOCK (2 Rollovers of BRG)
32 MHz	8 MHz	13h	400 kHz(1)
32 MHz	8 MHz	19h	308 kHz
32 MHz	8 MHz	4Fh	100 kHz
16 MHz	4 MHz	09h	400 kHz(1)
16 MHz	4 MHz	0Ch	308 kHz
16 MHz	4 MHz	27h	100 kHz
4 MHz	1 MHz	09h	100 kHz

Note 1: Refer to the I/O port electrical and timing specifications in Table 21-3 and Figure 21-5 to ensure the system is designed to support the I/O timing requirements.

2: The I^2C interface does not conform to the 400 kHz I^2C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

18.7.1 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register, APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see **Section 11.1 "Alternate Pin Function"** for more information.

18.8 Register Definitions: MSSP Control

REGISTER 18-1: SSPSTAT: SSP STATUS REGISTER

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

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

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

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

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

In I^2C Master or Slave mode:
1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for High Speed mode (400 kHz)

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

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

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

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

bit 4 **P:** Stop bit

(I^2C mode only. This bit is cleared when the MSSP1 module is disabled, SSPEN is cleared.)
1 = Indicates that a Stop bit has been detected last (this bit is '0' on Reset)
0 = Stop bit was not detected last

bit 3 **S:** Start bit

(I^2C mode only. This bit is cleared when the MSSP1 module is disabled, SSPEN is cleared.)
1 = Indicates that a Start bit has been detected last (this bit is '0' on Reset)
0 = Start bit was not detected last

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ bit information (I^2C mode only)

This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit, or not ACK bit.

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

In I^2C Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress
OR-ing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP1 is in Idle mode.

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

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

REGISTER 18-2: SSPCON1: SSP CONTROL REGISTER 1

R/C/HS-0/0	R/C/HS-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
WCOL	SSPOV	SSPEN	CKP	SSPM<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	HS = Bit is set by hardware	C = User cleared

bit 7 **WCOL:** Write Collision Detect bit
Master mode:
1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started
0 = No collision
Slave mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)
In SPI mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register (must be cleared in software).
0 = No overflow
In I^2C mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a "don't care" in Transmit mode (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** Synchronous Serial Port Enable bit
In both modes, when enabled, these pins must be properly configured as input or output
In SPI mode:
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as the source of the serial port pins(2)
0 = Disables serial port and configures these pins as I/O port pins
In I^2C mode:
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins(3)
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level
In I^2C Slave mode:
SCL release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)
In I^2C Master mode:
Unused in this mode

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = FOSC/4
0001 = SPI Master mode, clock = FOSC/16
0010 = SPI Master mode, clock = FOSC/64
0011 = Reserved
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0110 = I^2C Slave mode, 7-bit address
0111 = I^2C Slave mode, 10-bit address
1000 = I^2C Master mode, clock = FOSC / (4 * (SSPADD+1))(4)
1001 = Reserved
1010 = SPI Master mode, clock = FOSC/(4 * (SSPADD+1))(5)
1011 = I^2C firmware controlled Master mode (Slave idle)
1100 = Reserved
1101 = Reserved
1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
2: When enabled, these pins must be properly configured as input or output.
3: When enabled, the SDA and SCL pins must be configured as inputs.
4: SSPADD values of 0, 1 or 2 are not supported for I^2C mode.
5: SSPADD value of '0' is not supported. Use SSPM = 0000 instead.

REGISTER 18-3: SSPCON2: SSP CONTROL REGISTER 2

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

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

bit 7 **GCEN:** General Call Enable bit (in I^2C Slave mode only)
1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (in I^2C mode only)
1 = Acknowledge was not received
0 = Acknowledge was received

bit 5 **ACKDT:** Acknowledge Data bit (in I^2C mode only)
In Receive mode:
Value transmitted when the user initiates an Acknowledge sequence at the end of a receive
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (in I^2C Master mode only)
In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

bit 3 **RCEN:** Receive Enable bit (in I^2C Master mode only)
1 = Enables Receive mode for I^2C
0 = Receive idle

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

bit 1 **RSEN:** Repeated Start Condition Enable bit (in I^2C Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

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

Note 1: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I^2C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

REGISTER 18-4: SSPCON3: SSP CONTROL REGISTER 3

R-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
ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
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 **ACKTIM:** Acknowledge Time Status bit (I^2C mode only)**(3)**
1 = Indicates the I^2C bus is in an Acknowledge sequence, set on 8^{TH} falling edge of SCL clock
0 = Not an Acknowledge sequence, cleared on 9^{TH} rising edge of SCL clock

bit 6 **PCIE**: Stop Condition Interrupt Enable bit (I^2C mode only)
1 = Enable interrupt on detection of Stop condition
0 = Stop detection interrupts are disabled**(2)**

bit 5 **SCIE**: Start Condition Interrupt Enable bit (I^2C mode only)
1 = Enable interrupt on detection of Start or Restart conditions
0 = Start detection interrupts are disabled**(2)**

bit 4 **BOEN:** Buffer Overwrite Enable bit
In SPI Slave mode:**(1)**
1 = SSPBUF updates every time that a new data byte is shifted in ignoring the BF bit
0 = If new byte is received with BF bit of the SSPSTAT register already set, SSPOV bit of the SSPCON1 register is set, and the buffer is not updated
In I^2C Master mode and SPI Master mode:
This bit is ignored.
In I^2C Slave mode:
1 = SSPBUF is updated and $\overline{ACK}$ is generated for a received address/data byte, ignoring the state of the SSPOV bit only if the BF bit = 0.
0 = SSPBUF is only updated when SSPOV is clear

bit 3 **SDAHT:** SDA Hold Time Selection bit (I^2C mode only)
1 = Minimum of 300 ns hold time on SDA after the falling edge of SCL
0 = Minimum of 100 ns hold time on SDA after the falling edge of SCL

bit 2 **SBCDE:** Slave Mode Bus Collision Detect Enable bit (I^2C Slave mode only)
If on the rising edge of SCL, SDA is sampled low when the module is outputting a high state, the BCLIF bit of the PIR2 register is set, and bus goes idle
1 = Enable slave bus collision interrupts
0 = Slave bus collision interrupts are disabled

bit 1 **AHEN:** Address Hold Enable bit (I^2C Slave mode only)
1 = Following the 8th falling edge of SCL for a matching received address byte; CKP bit of the SSPCON1 register will be cleared and the SCL will be held low.
0 = Address holding is disabled

bit 0 **DHEN:** Data Hold Enable bit (I^2C Slave mode only)
1 = Following the 8th falling edge of SCL for a received data byte; slave hardware clears the CKP bit of the SSPCON1 register and SCL is held low.
0 = Data holding is disabled

Note 1: For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPBUF.
2: This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.
3: The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is set.

REGISTER 18-5: SSPMSK: SSP MASK REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
MSK<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 7-1 **MSK<7:1>:** Mask bits
1 = The received address bit n is compared to SSPADD<n> to detect I^2C address match
0 = The received address bit n is not used to detect I^2C address match

bit 0 **MSK<0>:** Mask bit for I^2C Slave mode, 10-bit Address
I^2C Slave mode, 10-bit address (SSPM<3:0> = 0111 or 1111):
1 = The received address bit 0 is compared to SSPADD<0> to detect I^2C address match
0 = The received address bit 0 is not used to detect I^2C address match
I^2C Slave mode, 7-bit address, the bit is ignored

REGISTER 18-6: SSPADD: MSSP1 ADDRESS AND BAUD RATE REGISTER (I^2C MODE)

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
ADD<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	

Master mode:

bit 7-0 **ADD<7:0>:** Baud Rate Clock Divider bits
SCL pin clock period = ((ADD<7:0> + 1) *4)/FOSC

10-Bit Slave mode — Most Significant Address Byte:

bit 7-3 **Not used:** Unused for Most Significant Address Byte. Bit state of this register is a "don't care". Bit pattern sent by master is fixed by I^2C specification and must be equal to '11110'. However, those bits are compared by hardware and are not affected by the value in this register.

bit 2-1 **ADD<2:1>:** Two Most Significant bits of 10-bit address

bit 0 **Not used:** Unused in this mode. Bit state is a "don't care".

10-Bit Slave mode — Least Significant Address Byte:

bit 7-0 **ADD<7:0>:** Eight Least Significant bits of 10-bit address

7-Bit Slave mode:

bit 7-1 **ADD<7:1>:** 7-bit address

bit 0 **Not used:** Unused in this mode. Bit state is a "don't care".

19.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

ICSP™ programming allows customers to manufacture circuit boards with unprogrammed devices. Programming can be done after the assembly process, allowing the device to be programmed with the most recent firmware or a custom firmware. Five pins are needed for ICSP™ programming:

- ICSPCLK
- ICSPDAT
- $\overline{MCLR}$/VPP
- VDD
- VSS

In Program/Verify mode the Program Memory, User IDs and the Configuration Words are programmed through serial communications. The ICSPDAT pin is a bidirectional I/O used for transferring the serial data and the ICSPCLK pin is the clock input. For more information on ICSP™ refer to the **"***PIC12LF1552 Memory Programming Specification*" (DS41642).

19.1 High-Voltage Programming Entry Mode

The device is placed into High-Voltage Programming Entry mode by holding the ICSPCLK and ICSPDAT pins low then raising the voltage on $\overline{MCLR}$/VPP to VIHH.

19.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using VDD only, without high voltage. When the LVP bit of Configuration Words is set to '1', the low-voltage ICSP programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to '0'.

Entry into the Low-Voltage Programming Entry mode requires the following steps:

1. $\overline{MCLR}$ is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, $\overline{MCLR}$ must be held at VIL for as long as Program/Verify mode is to be maintained.

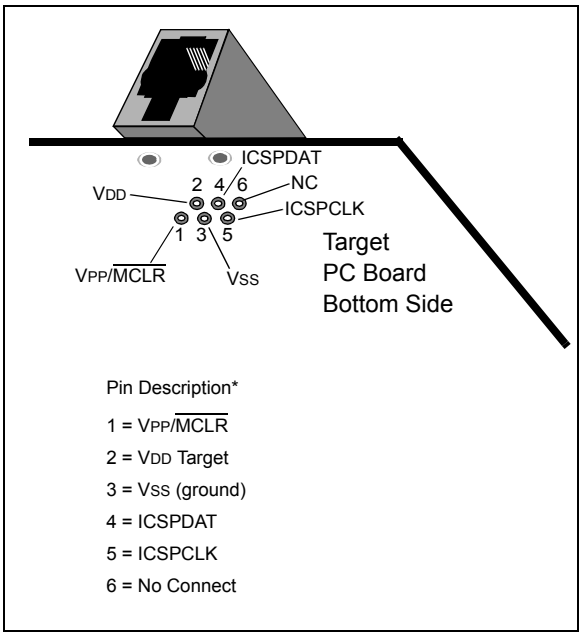
If low-voltage programming is enabled (LVP = 1), the $\overline{MCLR}$ Reset function is automatically enabled and cannot be disabled. See **Section 6.5 "MCLR"** for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

19.3 Common Programming Interfaces

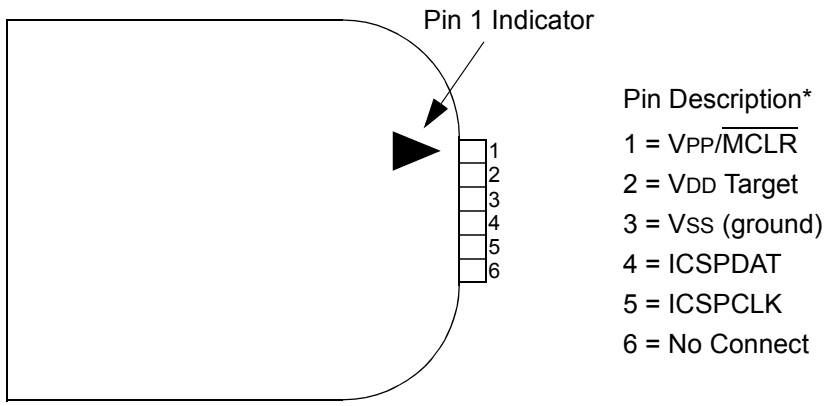
Connection to a target device is typically done through an ICSP™ header. A commonly found connector on development tools is the RJ-11 in the 6P6C (6-pin, 6-connector) configuration. See Figure 19-1.

FIGURE 19-1: ICD RJ-11 STYLE CONNECTOR INTERFACE



Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 19-2.

FIGURE 19-2: PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE

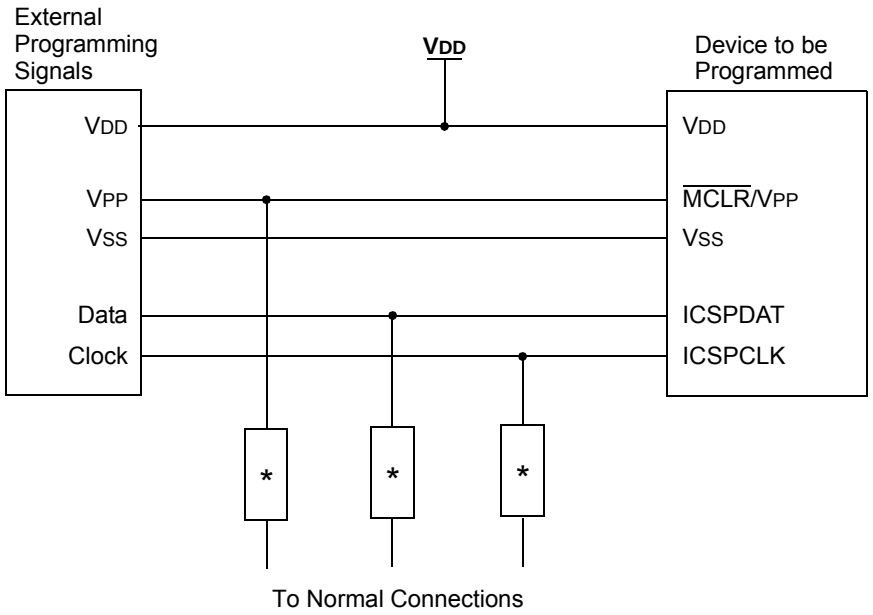


* The 6-pin header (0.100" spacing) accepts 0.025" square pins.

For additional interface recommendations, refer to your specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 19-3 for more information.

FIGURE 19-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING



* Isolation devices (as required).

20.0 INSTRUCTION SET SUMMARY

Each instruction is a 14-bit word containing the operation code (opcode) and all required operands. The opcodes are broken into three broad categories.

- Byte Oriented
- Bit Oriented
- Literal and Control

The literal and control category contains the most varied instruction word format.

Table 20-3 lists the instructions recognized by the MPASM™ assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

- Subroutine takes two cycles (CALL, CALLW)
- Returns from interrupts or subroutines take two cycles (RETURN, RETLW, RETFIE)
- Program branching takes two cycles (GOTO, BRA, BRW, BTFSS, BTFSC, DECFSZ, INCSFZ)
- One additional instruction cycle will be used when any instruction references an indirect file register and the file select register is pointing to program memory.

One instruction cycle consists of 4 oscillator cycles; for an oscillator frequency of 4 MHz, this gives a nominal instruction execution rate of 1 MHz.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

20.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

TABLE 20-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
n	FSR or INDF number. (0-1)
mm	Pre-post increment-decrement mode selection

TABLE 20-2: ABBREVIATION DESCRIPTIONS

Field	Description
PC	Program Counter
$\overline{TO}$	Time-out bit
C	Carry bit
DC	Digit carry bit
Z	Zero bit
$\overline{PD}$	Power-down bit

FIGURE 20-1: GENERAL FORMAT FOR INSTRUCTIONS

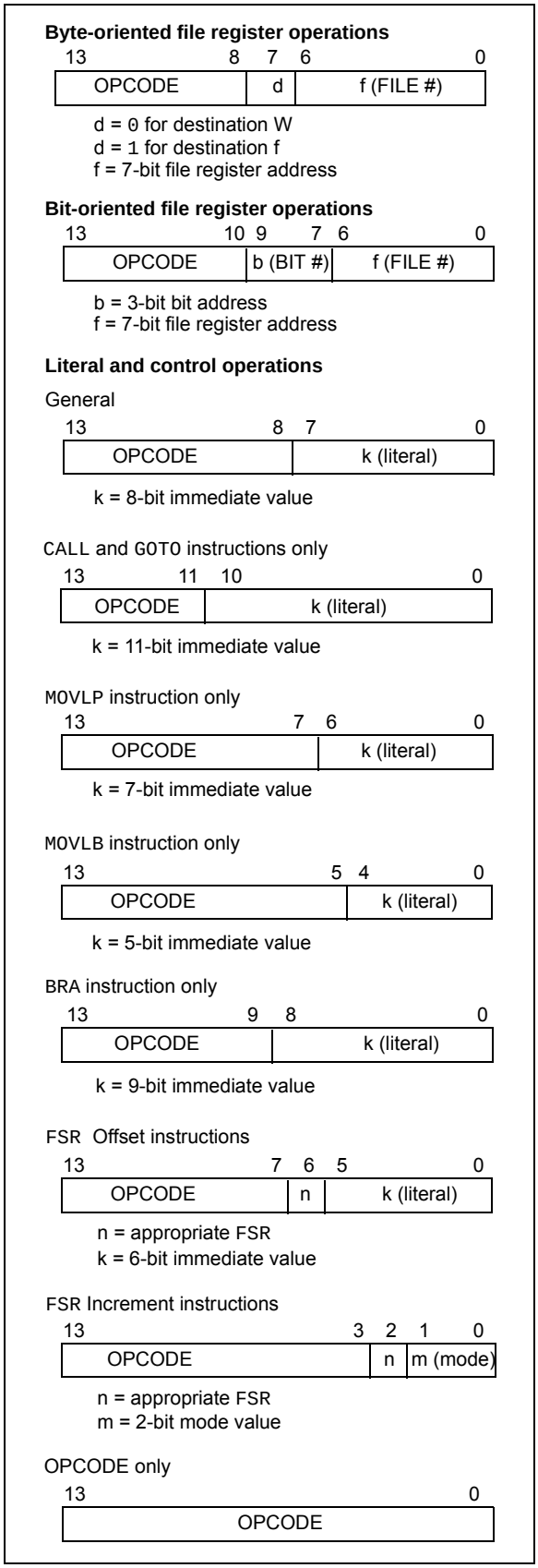


TABLE 20-3: PIC12LF1552 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	2
ADDWFC	f, d	Add with Carry W and f	1	11	1101	dfff	ffff	C, DC, Z	2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	2
ASRF	f, d	Arithmetic Right Shift	1	11	0111	dfff	ffff	C, Z	2
LSLF	f, d	Logical Left Shift	1	11	0101	dfff	ffff	C, Z	2
LSRF	f, d	Logical Right Shift	1	11	0110	dfff	ffff	C, Z	2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	–	Clear W	1	00	0001	0000	00xx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	2
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	2
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		2
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	2
SUBWFB	f, d	Subtract with Borrow W from f	1	11	1011	dfff	ffff	C, DC, Z	2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	2
BYTE ORIENTED SKIP OPERATIONS									
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		2
BIT-ORIENTED SKIP OPERATIONS									
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		1, 2
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		1, 2
LITERAL OPERATIONS									
ADDLW	k	Add literal and W	1	11	1110	kkkk	kkkk	C, DC, Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLB	k	Move literal to BSR	1	00	0000	001k	kkkk		
MOVLP	k	Move literal to PCLATH	1	11	0001	1kkk	kkkk		
MOVLW	k	Move literal to W	1	11	0000	kkkk	kkkk		
SUBLW	k	Subtract W from literal	1	11	1100	kkkk	kkkk	C, DC, Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

TABLE 20-3: PIC12LF1552 INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
CONTROL OPERATIONS						
BRA	k	Relative Branch	2	11 001k kkkk kkkk		
BRW	–	Relative Branch with W	2	00 0000 0000 1011		
CALL	k	Call Subroutine	2	10 0kkk kkkk kkkk		
CALLW	–	Call Subroutine with W	2	00 0000 0000 1010		
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
RETFIE	k	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 0100 kkkk kkkk		
RETURN	–	Return from Subroutine	2	00 0000 0000 1000		
INHERENT OPERATIONS						
CLRWDT	–	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$, $\overline{PD}$	
NOP	–	No Operation	1	00 0000 0000 0000		
OPTION	–	Load OPTION_REG register with W	1	00 0000 0110 0010		
RESET	–	Software device Reset	1	00 0000 0000 0001		
SLEEP	–	Go into Standby mode	1	00 0000 0110 0011	$\overline{TO}$, $\overline{PD}$	
TRIS	f	Load TRIS register with W	1	00 0000 0110 0fff		
C-COMPILER OPTIMIZED						
ADDFSR	n, k	Add Literal k to FSRn	1	11 0001 0nkk kkkk		
MOVIW	n mm	Move Indirect FSRn to W with pre/post inc/dec modifier, mm	1	00 0000 0001 0nmm kkkk	Z	**2, 3**
	k[n]	Move INDFn to W, Indexed Indirect.	1	11 1111 0nkk 1nmm	Z	**2**
MOVWI	n mm	Move W to Indirect FSRn with pre/post inc/dec modifier, mm	1	00 0000 0001 kkkk		**2, 3**
	k[n]	Move W to INDFn, Indexed Indirect.	1	11 1111 1nkk		**2**

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

3: See Table in the MOVIW and MOVWI instruction descriptions.

20.2 Instruction Descriptions

ADDFSR — Add Literal to FSRn

Syntax:	[*label*] ADDFSR FSRn, k
Operands:	-32 ≤ k ≤ 31 n ∈ [0, 1]
Operation:	FSR(n) + k → FSR(n)
Status Affected:	None
Description:	The signed 6-bit literal 'k' is added to the contents of the FSRnH:FSRnL register pair. FSRn is limited to the range 0000h-FFFFh. Moving beyond these bounds will cause the FSR to wrap-around.

ADDLW — Add literal and W

Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the 8-bit literal 'k' and the result is placed in the W register.

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with 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'.

ADDWFC — ADD W and CARRY bit to f

Syntax:	[*label*] ADDWFC f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) + (C) → dest
Status Affected:	C, DC, Z
Description:	Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.

ANDLW — AND literal with W

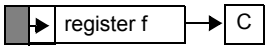
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the 8-bit literal 'k'. The result is placed in the W register.

ANDWF — AND W with f

Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with 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'.

ASRF — Arithmetic Right Shift

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



BCF **Bit Clear f**

Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BRA **Relative Branch**

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

BRW **Relative Branch with W**

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

BSF **Bit Set f**

Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSC **Bit Test f, Skip if Clear**

Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2-cycle instruction.

BTFSS **Bit Test f, Skip if Set**

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

CALL — **Call Subroutine**

Syntax:	[*label*] CALL k
Operands:	$0 \le k \le 2047$
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<6:3>) → PC<14:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a 2-cycle instruction.

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:	$0 \le f \le 127$
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:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	$(\bar{f}) \rightarrow$ (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:	$0 \le f \le 127$ $d \in [0,1]$
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'.

DECFSZ **Decrement f, Skip if 0**

Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'. If the result is '1', the next instruction is executed. If the result is '0', then a NOP is executed instead, making it a 2-cycle instruction.

GOTO **Unconditional Branch**

Syntax:	[*label*] GOTO k
Operands:	$0 \le k \le 2047$
Operation:	k → PC<10:0> PCLATH<6:3> → PC<14:11>
Status Affected:	None
Description:	GOTO is an unconditional branch. The 11-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a 2-cycle instruction.

INCF **Increment f**

Syntax:	[*label*] INCF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

INCFSZ **Increment f, Skip if 0**

Syntax:	[*label*] INCFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'. If the result is '1', the next instruction is executed. If the result is '0', a NOP is executed instead, making it a 2-cycle instruction.

IORLW **Inclusive OR literal with W**

Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR'ed with the 8-bit literal 'k'. The result is placed in the W register.

IORWF **Inclusive OR W with f**

Syntax:	[*label*] IORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register 'f'. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

LSLF **Logical Left Shift**

Syntax: [*label*] LSLF f {,d}

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f<7>) → C
(f<6:0>) → dest<7:1>
0 → dest<0>

Status Affected: C, Z

Description: The contents of register 'f' are shifted one bit to the left through the Carry flag. A '0' is shifted into the LSb. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.

LSRF **Logical Right Shift**

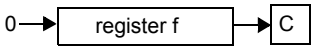
Syntax: [*label*] LSRF f {,d}

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: 0 → dest<7>
(f<7:1>) → dest<6:0>,
(f<0>) → C,

Status Affected: C, Z

Description: The contents of register 'f' are shifted one bit to the right through the Carry flag. A '0' is shifted into the MSb. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.



MOVF **Move f**

Syntax: [*label*] MOVF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) → (dest)

Status Affected: Z

Description: The contents of register f is moved to a destination dependent upon the status of d. If d = 0,destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Example:

```
MOVF    FSR, 0
```

After Instruction
W = value in FSR register
Z = 1

MOVIW **Move INDFn to W**

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

Operands: n ∈ [0,1]
mm ∈ [00,01, 10, 11]
-32 ≤ k ≤ 31

Operation: INDFn → W
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: Z

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.

MOVLB **Move literal to BSR**

Syntax: [*label*] MOVLB k

Operands: 0 ≤ k ≤ 15

Operation: k → BSR

Status Affected: None

Description: The 5-bit literal 'k' is loaded into the Bank Select Register (BSR).

MOVLP **Move literal to PCLATH**

Syntax: [*label*] MOVLP k

Operands: 0 ≤ k ≤ 127

Operation: k → PCLATH

Status Affected: None

Description: The 7-bit literal 'k' is loaded into the PCLATH register.

MOVLW **Move literal to W**

Syntax: [*label*] MOVLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W)

Status Affected: None

Description: The 8-bit literal 'k' is loaded into W register. The "don't cares" will assemble as '0's.

Words: 1

Cycles: 1

Example:
```
MOVLW   0x5A
```
After Instruction
W = 0x5A

MOVWF **Move W to f**

Syntax: [*label*] MOVWF f

Operands: 0 ≤ f ≤ 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 ∈ [0,1]
mm ∈ [00,01, 10, 11]
-32 ≤ k ≤ 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.

NOP **No Operation**

Syntax: [*label*] NOP
Operands: None
Operation: No operation
Status Affected: None
Description: No operation
Words: 1
Cycles: 1
Example:
```
NOP
```

OPTION **Load OPTION_REG Register with W**

Syntax: [*label*] OPTION
Operands: None
Operation: (W) → OPTION_REG
Status Affected: None
Description: Move data from W register to OPTION_REG register.

RESET **Software Reset**

Syntax: [*label*] RESET
Operands: None
Operation: Execute a device Reset. Resets the $\overline{RI}$ flag of the PCON register.
Status Affected: None
Description: This instruction provides a way to execute a hardware Reset by software.

RETFIE **Return from Interrupt**

Syntax: [*label*] RETFIE k

Operands: None

Operation: TOS → PC,
1 → GIE

Status Affected: None

Description: Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a 2-cycle instruction.

Words: 1

Cycles: 2

Example:

```
RETFIE
```

After Interrupt
PC = TOS
GIE = 1

RETLW **Return with literal in W**

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W);
TOS → PC

Status Affected: None

Description: The W register is loaded with the 8-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a 2-cycle instruction.

Words: 1

Cycles: 2

Example:

```
CALL TABLE;W contains table
    ;offset value
•   ;W now has table value
•
•
ADDWF PC  ;W = offset
RETLW k1  ;Begin table
RETLW k2  ;
•
•
•
RETLW kn  ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of k8

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a 2-cycle instruction.

RLF **Rotate Left f through Carry**

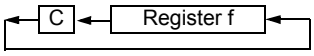
Syntax: [*label*] RLF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is stored back in register 'f'.



Words: 1

Cycles: 1

Example:

```
RLF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

RRF **Rotate Right f through Carry**

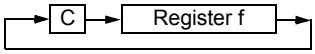
Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.



SLEEP **Enter Sleep mode**

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Description: The power-down Status bit, $\overline{PD}$ is cleared. Time-out Status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.
The processor is put into Sleep mode with the oscillator stopped.

SUBLW **Subtract W from literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the 8-bit literal 'k'. The result is placed in the W register.

C = 0	W > k
C = 1	W ≤ k
DC = 0	W<3:0> > k<3:0>
DC = 1	W<3:0> ≤ k<3:0>

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → (destination)

Status Affected: C, DC, Z

Description: Subtract (2's complement method) W register from 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.

C = 0	W > f
C = 1	W ≤ f
DC = 0	W<3:0> > f<3:0>
DC = 1	W<3:0> ≤ f<3:0>

SUBWFB **Subtract W from f with Borrow**

Syntax: SUBWFB f {,d}

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) – (W) – ($\overline{B}$) → dest

Status Affected: C, DC, Z

Description: Subtract W and the BORROW flag (CARRY) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

SWAPF **Swap Nibbles in f**

Syntax:	[*label*] SWAPF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed in register 'f'.

TRIS **Load TRIS Register with W**

Syntax:	[*label*] TRIS f
Operands:	$5 \le f \le 7$
Operation:	(W) → TRIS register 'f'
Status Affected:	None
Description:	Move data from W register to TRIS register. When 'f' = 5, TRISA is loaded. When 'f' = 6, TRISB is loaded. When 'f' = 7, TRISC is loaded.

XORLW **Exclusive OR literal with W**

Syntax:	[*label*] XORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the 8-bit literal 'k'. The result is placed in the W register.

XORWF **Exclusive OR W with f**

Syntax:	[*label*] XORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of the W register with 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'.

21.0 ELECTRICAL SPECIFICATIONS

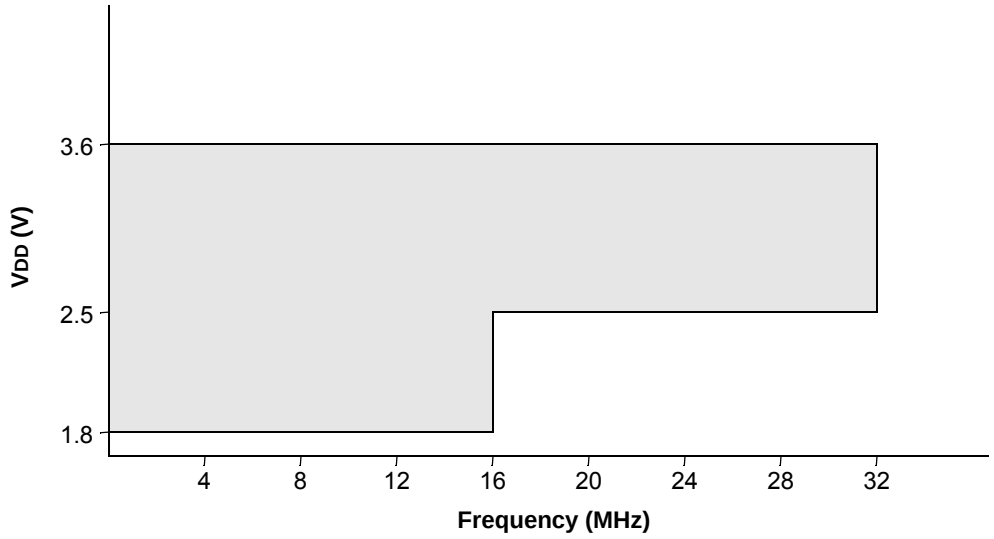
Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on V_{DD} with respect to V_{SS} ..... -0.3V to +4.0V
Voltage on $\overline{MCLR}$ with respect to Vss ..... -0.3V to +9.0V
Voltage on all other pins with respect to V_{SS} ..... -0.3V to (V_{DD} + 0.3V)
Total power dissipation(1) ..... 800 mW
Maximum current out of V_{SS} pin, -40°C ≤ T_A ≤ +85°C for industrial ..... 210 mA
Maximum current out of V_{SS} pin, -40°C ≤ T_A ≤ +125°C for extended ..... 95 mA
Maximum current into V_{DD} pin, -40°C ≤ T_A ≤ +85°C for industrial ..... 150 mA
Maximum current into V_{DD} pin, -40°C ≤ T_A ≤ +125°C for extended ..... 70 mA
Clamp current, I_K (V_{PIN} < 0 or V_{PIN} > V_{DD}) ..... ± 20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA

Note 1: Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$.

† NOTICE: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.

FIGURE 21-1: PIC12LF1552 VOLTAGE FREQUENCY GRAPH, -40°C ≤ T_A ≤ +125°C



Note 1: The shaded region indicates the permissible combinations of voltage and frequency.
2: Refer to Table 21-1 for each Oscillator mode’s supported frequencies.

21.1 DC Characteristics: PIC12LF1552-I/E (Industrial, Extended)

PIC12LF1552			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
D001	VDD	**Supply Voltage (VDDMIN, VDDMAX)**					
			1.8 2.5	— —	3.6 3.6	V V	FOSC ≤ 16 MHz: FOSC ≤ 32 MHz
D002*	VDR	**RAM Data Retention Voltage(1)**	1.5	—	—	V	Device in Sleep mode
D002A*	VPOR*	**Power-on Reset Release Voltage**	—	1.6	—	V	
D002B*	VPORR*	**Power-on Reset Rearm Voltage**	—	0.8	—	V	
D003	VADFVR	**Fixed Voltage Reference Voltage for ADC, Initial Accuracy**	-7 -8 -7 -8	— — — —	6 6 6 6	%	1.024V, VDD ≥ 2.5V, 85°C (**Note 2**) 1.024V, VDD ≥ 2.5V, 125°C (**Note 2**) 2.048V, VDD ≥ 2.5V, 85°C 2.048V, VDD ≥ 2.5V, 125°C
D003C*	TCVFVR	**Temperature Coefficient, Fixed Voltage Reference**	—	-130	—	ppm/°C	
D003D*	ΔVFVR/ ΔVIN	**Line Regulation, Fixed Voltage Reference**	—	0.270	—	%/V	
D004*	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See **Section 6.1 "Power-on Reset (POR)"** for details.

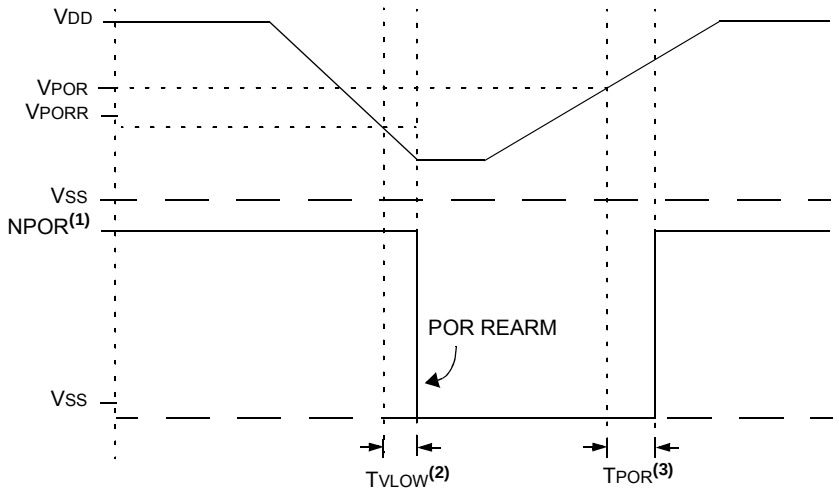
* These parameters are characterized but not tested.

† 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: This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

2: For proper operation, the minimum value of the ADC positive voltage reference must be 1.8V or greater. When selecting the FVR or the VREF+ pin as the source of the ADC positive voltage reference, be aware that the voltage must be 1.8V or greater.

FIGURE 21-2: POR AND POR REARM WITH SLOW RISING VDD



Note 1: When NPOR is low, the device is held in Reset.
2: TPOR 1 μs typical.
3: TVLOW 2.7 μs typical.

21.2 DC Characteristics: PIC12LF1552-I/E (Industrial, Extended)

PIC12LF1552			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param. No.	Device Characteristics	Min.	Typ.†	Max.	Units	Conditions	
						VDD	Note
	Supply Current (IDD)(1, 2)						
D010		—	2.5	18	µA	1.8	FOSC = 31 kHz LFINTOSC mode
		—	4	20	µA	3.0	
D011		—	0.35	0.70	mA	1.8	FOSC = 8 MHz HFINTOSC mode
		—	0.5	1.10	mA	3.0	
D012		—	0.5	1.2	mA	1.8	FOSC = 16 MHz HFINTOSC mode
		—	0.75	1.75	mA	3.0	
D013		—	1.4	3.5	mA	3.0	FOSC = 32 MHz HFINTOSC mode with PLL
D014		—	3	17	µA	1.8	FOSC = 32 kHz ECL mode
		—	5	20	µA	3.0	
D015		—	11	40	µA	1.8	FOSC = 500 kHz ECL mode
		—	16	60	µA	3.0	
D016		—	23	65	µA	1.8	FOSC = 1 MHz ECM mode
		—	36	100	µA	3.0	
D017		—	75	250	µA	1.8	FOSC = 4 MHz ECM mode
		—	120	430	µA	3.0	
D018		—	0.65	1.5	mA	3.0	FOSC = 20 MHz ECH mode

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

Note 1: The test conditions for all IDD measurements in active operation mode are: CLKIN = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; $\overline{MCLR}$ = VDD; WDT disabled.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

21.3 DC Characteristics: PIC12LF1552-I/E (Power-Down)

PIC12LF1552			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param. No.	Device Characteristics	Min.	Typ. †	Max. +85°C	Max. +125°C	Units	Conditions	
							VDD	Note
	Power-down Base Current (IPD)(2)							
D020		—	0.02	1.0	8	µA	1.8	WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive
		—	0.03	2	9	µA	3.0	
D021		—	0.24	2	9	µA	1.8	LPWDT Current **(Note 1)**
		—	0.37	3	10	µA	3.0	
D022		—	13	28	30	µA	1.8	FVR current **(Note 1)**
		—	22	30	33	µA	3.0	
D023		—	6.5	17	20	µA	3.0	BOR Current **(Note 1)**
D024		—	0.1	4	10	µA	3.0	LPBOR Current
D025		—	0.03	3.5	9	µA	1.8	ADC Current **(Note 1, Note 3)**, no conversion in progress
		—	0.04	4.0	10	µA	3.0	
D026*		—	350	—	—	µA	1.8	ADC Current **(Note 1, Note 3)**, conversion in progress
		—	350	—	—	µA	3.0	

* These parameters are characterized but not tested.

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

Note 1: The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

2: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD.

3: ADC oscillator source is FRC.

21.4 DC Characteristics: PIC12LF1552-I/E

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
	VIL	**Input Low Voltage**					
		I/O PORT:					
D030		with TTL buffer	—	—	0.15 VDD	V	1.8V ≤ VDD ≤ 3.6V
D031		with Schmitt Trigger buffer	—	—	0.2 VDD	V	2.0V ≤ VDD ≤ 3.6V
D032		$\overline{MCLR}$	—	—	0.2 VDD	V	
	VIH	**Input High Voltage**					
		I/O ports:		—	—		
D040		with TTL buffer	0.25 VDD + 0.8	—	—	V	1.8V ≤ VDD ≤ 3.6V
D041		with Schmitt Trigger buffer	0.8 VDD	—	—	V	2.0V ≤ VDD ≤ 3.6V
D042		$\overline{MCLR}$	0.8 VDD	—	—	V	
	IIL	**Input Leakage Current(1)**					
D060		I/O ports	—	± 5	± 125	nA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance at 85°C
				± 5	± 1000	nA	125°C
D061		$\overline{MCLR}$(2)	—	± 50	± 200	nA	VSS ≤ VPIN ≤ VDD at 85°C
	IPUR	**Weak Pull-up Current**					
D070*			25	100	200	µA	VDD = 3.3V, VPIN = VSS
	VOL	**Output Low Voltage(3)**					
D080		I/O ports	—	—	0.6	V	IOL = 6mA, VDD = 3.3V IOL = 1.8mA, VDD = 1.8V
	VOH	**Output High Voltage(3)**					
D090		I/O ports	VDD - 0.7	—	—	V	IOH = 3mA, VDD = 3.3V IOH = 1mA, VDD = 1.8V
		Capacitive Loading Specs on Output Pins					
D101A*	CIO	All I/O pins	—	—	50	pF	

* 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: 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.

3: Including OSC2 in CLKOUT mode.

21.5 Memory Programming Requirements

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C				
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
		Program Memory Programming Specifications					
D110	VIHH	Voltage on $\overline{MCLR}$/VPP pin	8.0	—	9.0	V	**(Note 2)**
D111	IDDP	Supply Current during Programming	—	—	10	mA	
D112	VBE	VDD for Bulk Erase	2.7	—	VDDMAX	V	
D113	VPEW	VDD for Write or Row Erase	VDDMIN	—	VDDMAX	V	
D114	IPPPGM	Current on $\overline{MCLR}$/VPP during Erase/Write	—	—	1.0	mA	
D115	IDDPGM	Current on VDD during Erase/Write	—	—	5.0	mA	
		Program Flash Memory					
D121	EP	Cell Endurance	10K	—	—	E/W	-40°C to +85°C **(Note 1)**
D122	VPRW	VDD for Read/Write	VDDMIN	—	VDDMAX	V	
D123	TIW	Self-timed Write Cycle Time	—	2	2.5	ms	
D124	TRETD	Characteristic Retention	—	40	—	Year	Provided no other specifications are violated
D125	EHEFC	High-Endurance Flash Cell	100K	—	—	E/W	0°C to +60°C, Lower byte, Last 128 Addresses in Flash Memory

† 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: Self-write and Block Erase.

2: Required only if single-supply programming is disabled.

21.6 Thermal Considerations

Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C					
Param. No.	**Sym.**	**Characteristic**	**Typ.**	**Units**	**Conditions**
TH01	θJA	Thermal Resistance Junction to Ambient	89.3	°C/W	8-pin PDIP package
			149.5	°C/W	8-pin SOIC package
			211	°C/W	8-pin MSOP package
			60	°C/W	8-pin UDFN 2X3mm package
TH02	θJC	Thermal Resistance Junction to Case	43.1	°C/W	8-pin PDIP package
			39.9	°C/W	8-pin SOIC package
			39	°C/W	8-pin MSOP package
			11	°C/W	8-pin UDFN 2X3mm package
TH03	TJMAX	Maximum Junction Temperature	150	°C	
TH04	PD	Power Dissipation	—	W	PD = PINTERNAL + PI/O
TH05	PINTERNAL	Internal Power Dissipation	—	W	PINTERNAL = IDD x VDD(1)
TH06	PI/O	I/O Power Dissipation	—	W	PI/O = Σ (IOL * VOL) + Σ (IOH * (VDD - VOH))
TH07	PDER	Derated Power	—	W	PDER = PDMAX (TJ - TA)/θJA(2)

Note 1: IDD is current to run the chip alone without driving any load on the output pins.
2: TA = Ambient Temperature; TJ = Junction Temperature.

21.7 Timing Parameter Symbology

The timing parameter symbols have been created with one of the following formats:

1. TppS2ppS
2. TppS

T			
F	Frequency	T	Time

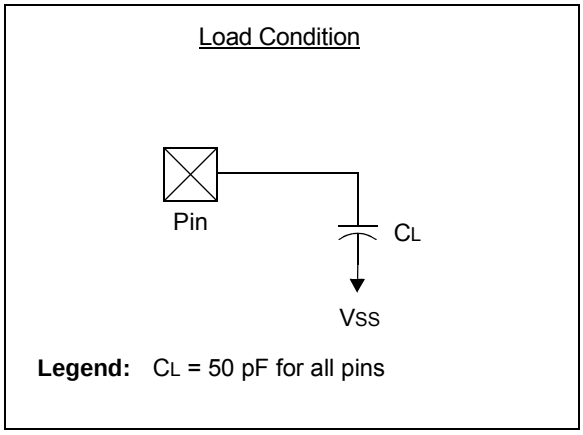
Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	CLKIN
ck	CLKOUT	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDIx	sc	SCKx
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O PORT	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (High-impedance)	V	Valid
L	Low	Z	High-impedance

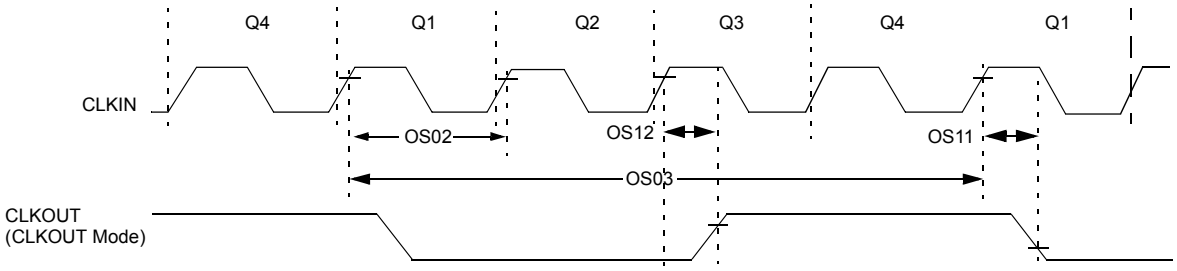
FIGURE 21-3: LOAD CONDITIONS



Legend: CL = 50 pF for all pins

21.8 AC Characteristics: PIC12LF1552-I/E

FIGURE 21-4: CLOCK TIMING



Note: See Table 21-3.

TABLE 21-1: CLOCK OSCILLATOR TIMING REQUIREMENTS

Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
OS01	FOSC	External CLKIN Frequency(1)	DC	—	0.5	MHz	EC Oscillator mode (low)
			DC	—	4	MHz	EC Oscillator mode (medium)
			DC	—	20	MHz	EC Oscillator mode (high)
OS02	TOSC	External CLKIN Period(1)	50	—	∞	ns	EC mode
OS03	TCY	Instruction Cycle Time(1)	200	—	DC	ns	TCY = FOSC/4

* 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: Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to CLKIN pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

TABLE 21-2: OSCILLATOR PARAMETERS

Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
OS08	HFOSC	Internal Calibrated HFINTOSC Frequency(1)	—	16.0	—	MHz	0°C ≤ TA ≤ +85°C
OS08A	HFTOL	Frequency Tolerance	—	±3	—	%	25°C, 16MHz
			—	±6	—	%	0°C ≤ TA ≤ +85°C, 16 MHz
OS09	LFOSC	Internal LFINTOSC Frequency	—	31	—	kHz	-40°C ≤ TA ≤ +125°C
OS10*	TWARM	HFINTOSC Wake-up from Sleep Start-up Time LFINTOSC Wake-up from Sleep Start-up Time MFINTOSC Wake-up from Sleep Start-up Time	—	5	8	μs	

* 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: To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

FIGURE 21-5: CLKOUT AND I/O TIMING

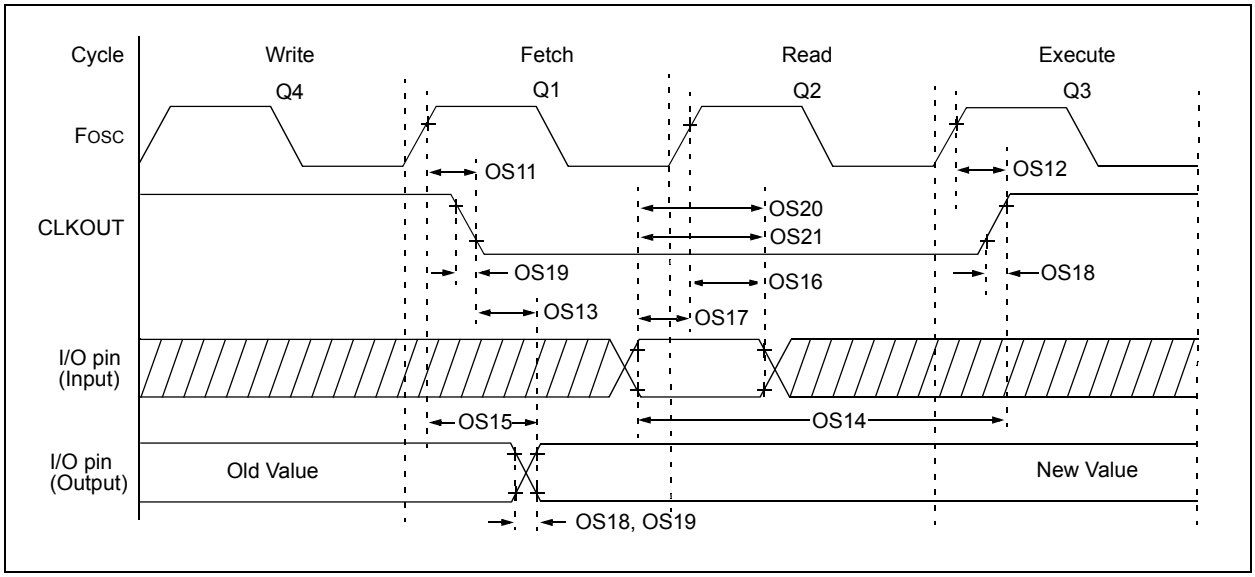


TABLE 21-3: CLKOUT AND I/O TIMING PARAMETERS

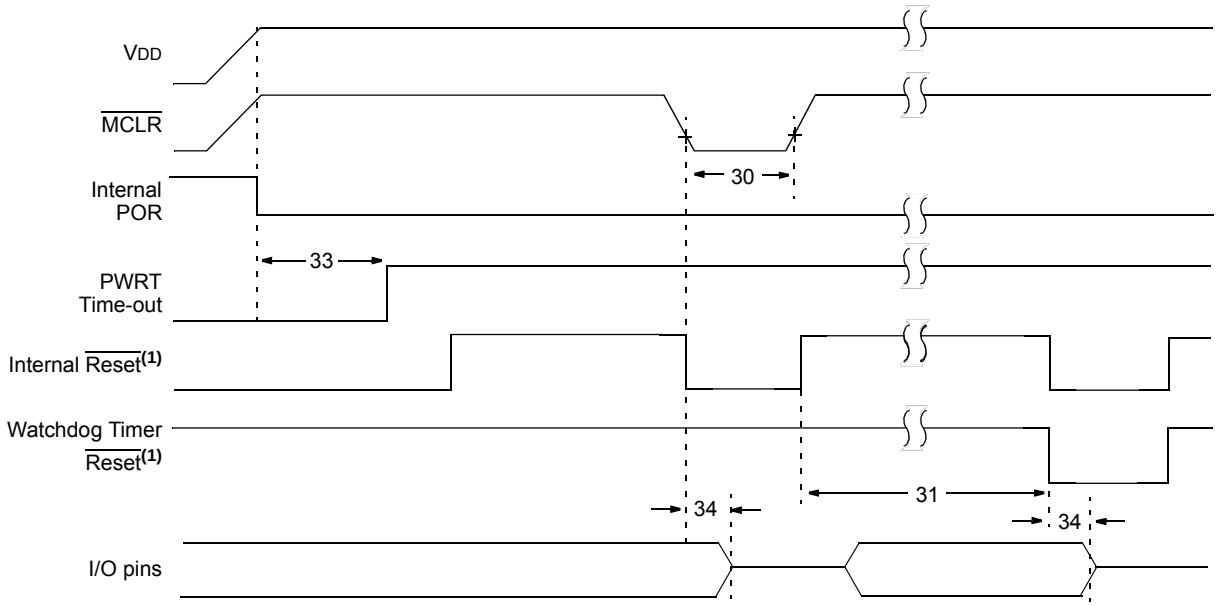
Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
OS11	TosH2ckL	FOSC↑ to CLKOUT↓ (1)	—	—	70	ns	VDD = 3.3-3.6V
OS12	TosH2ckH	FOSC↑ to CLKOUT↑ (1)	—	—	72	ns	VDD = 3.3-3.6V
OS13	TckL2ioV	CLKOUT↓ to Port out valid(1)	—	—	20	ns	
OS14	TioV2ckH	Port input valid before CLKOUT↑(1)	TOSC + 200 ns	—	—	ns	
OS15	TosH2ioV	Fosc↑ (Q1 cycle) to Port out valid	—	50	70*	ns	VDD = 3.3-3.6V
OS16	TosH2ioI	Fosc↑ (Q2 cycle) to Port input invalid (I/O in hold time)	50	—	—	ns	VDD = 3.3-3.6V
OS17	TioV2osH	Port input valid to Fosc↑ (Q2 cycle) (I/O in setup time)	20	—	—	ns	
OS18*	TioR	Port output rise time(2)	—	15	32	ns	VDD = 2.0V
OS19*	TioF	Port output fall time(2)	—	28	55	ns	VDD = 2.0V
OS20*	Tinp	INT pin input high or low time	25	—	—	ns	
OS21*	Tioc	Interrupt-on-change new input level time	25	—	—	ns	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated.

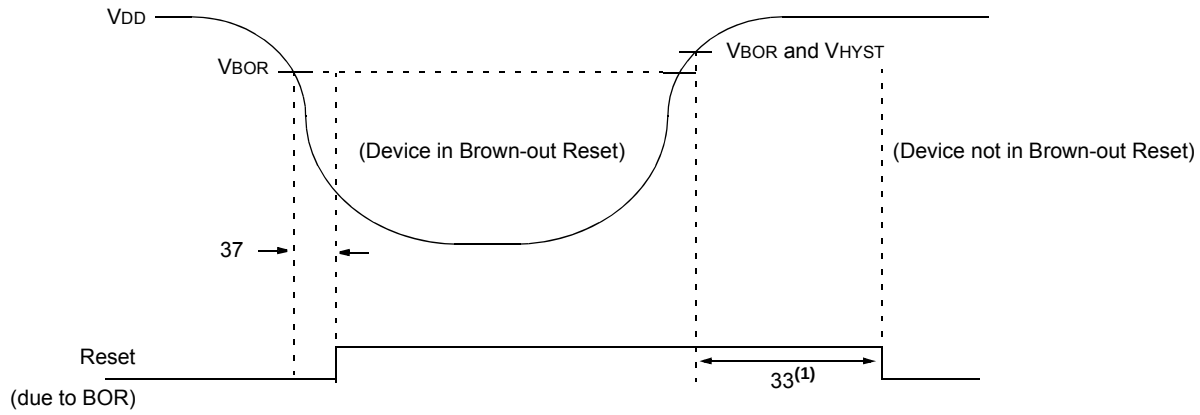
Note 1: Measurements are taken in EC mode where CLKOUT output is 4 x TOSC.

FIGURE 21-6: RESET, WATCHDOG TIMER AND POWER-UP TIMER TIMING



Note 1: Asserted low.

FIGURE 21-7: BROWN-OUT RESET TIMING AND CHARACTERISTICS



Note 1: 64 ms delay only if $\overline{\text{PWRTE}}$ bit in the Configuration Words is programmed to '0'.
2 ms delay if $\overline{\text{PWRTE}}$ = 0.

TABLE 21-4: RESET, WATCHDOG TIMER, POWER-UP TIMER AND BROWN-OUT RESET PARAMETERS

Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
30	TMCL	$\overline{MCLR}$ Pulse Width (low)	2 5	— —	— —	μs μs	-40°C to +85°C +85°C to +125°C
31	TWDTLP	Low-Power Watchdog Timer Time-out Period	10	16	27	ms	VDD = 3.3V-3.6V, 1:512 Prescaler used
33*	TPWRT	Power-up Timer Period, $\overline{PWRTE}$ = 0	40	65	140	ms	
34*	TIOZ	I/O high-impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset	—	—	2.0	μs	
35	VBOR	Brown-out Reset Voltage(1)	2.55 1.80	2.70 1.90	2.85 2.05	V V	BORV = 0 BORV = 1
36*	VHYST	Brown-out Reset Hysteresis	0	25	50	mV	-40°C to +85°C
37*	TBORDC	Brown-out Reset DC Response Time	1	3	5	μs	VDD ≤ VBOR

* 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: To ensure these voltage tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

FIGURE 21-8: TIMER0 EXTERNAL CLOCK TIMINGS

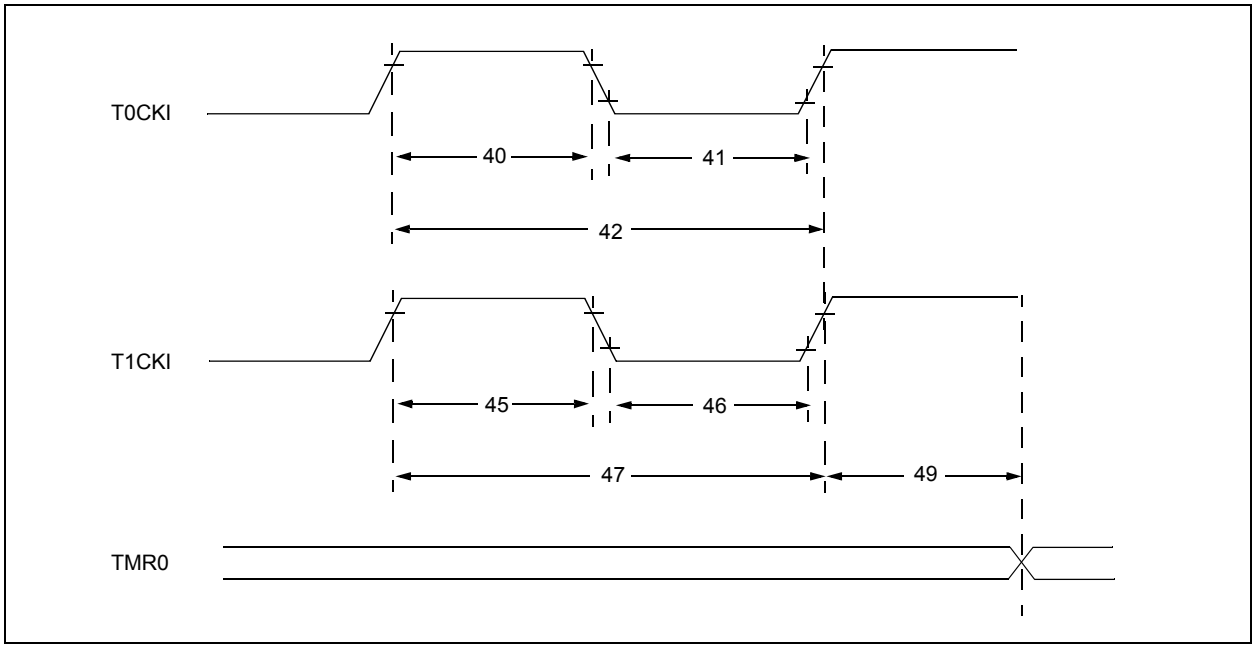


TABLE 21-5: TIMER0 EXTERNAL CLOCK REQUIREMENTS

Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C								
Param. No.	**Sym.**	**Characteristic**		**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
40*	TT0H	T0CKI High Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
41*	TT0L	T0CKI Low Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
42*	TT0P	T0CKI Period		Greater of: 20 or TCY + 40 / N	—	—	ns	N = prescale value (2, 4, ..., 256)
45*	TT1H	T1CKI High Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	

* 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.

TABLE 21-6: PIC12LF1552 A/D CONVERTER (ADC) CHARACTERISTICS:(1,2,3)

Standard Operating Conditions (unless otherwise stated) Operating temperature Tested at 25°C							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
AD01	NR	Resolution	—	—	10	bit	
AD02	EIL	Integral Error	—	±0.4	±1	LSb	-40°C to +85°C, VREF ≥ 2.0V
AD03	EDL	Differential Error	—	±0.3	±1	LSb	-40°C to +85°C, VREF ≥ 2.0V
AD04	EOFF	Offset Error	—	1.2	±3	LSb	-40°C to +85°C, VREF ≥ 2.0V
AD05	EGN	Gain Error	—	1.0	±3	LSb	-40°C to +85°C, VREF ≥ 2.0V
AD06	VREF	Reference Voltage Range (VREFH – VREFL)	1.8 2.0	— —	— —	V V	Absolute Minimum **(Note 4)** Minimum for 1LSb Accuracy
AD07	VAIN	Full-Scale Range	VSS	—	VREF	V	
AD08	ZAIN	Recommended Impedance of Analog Voltage Source	—	—	3	kΩ	Can go higher if external 0.01μF capacitor is present on input pin.

* 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: Total Absolute Error includes integral, differential, offset and gain errors.

2: The ADC conversion result never decreases with an increase in the input voltage and has no missing codes.

3: When ADC is off, it will not consume any current other than leakage current. The power-down current specification includes any such leakage from the ADC module.

4: ADC VREF is selected by ADPREF<1:0> bits.

TABLE 21-7: PIC12LF1552 ADC CONVERSION REQUIREMENTS

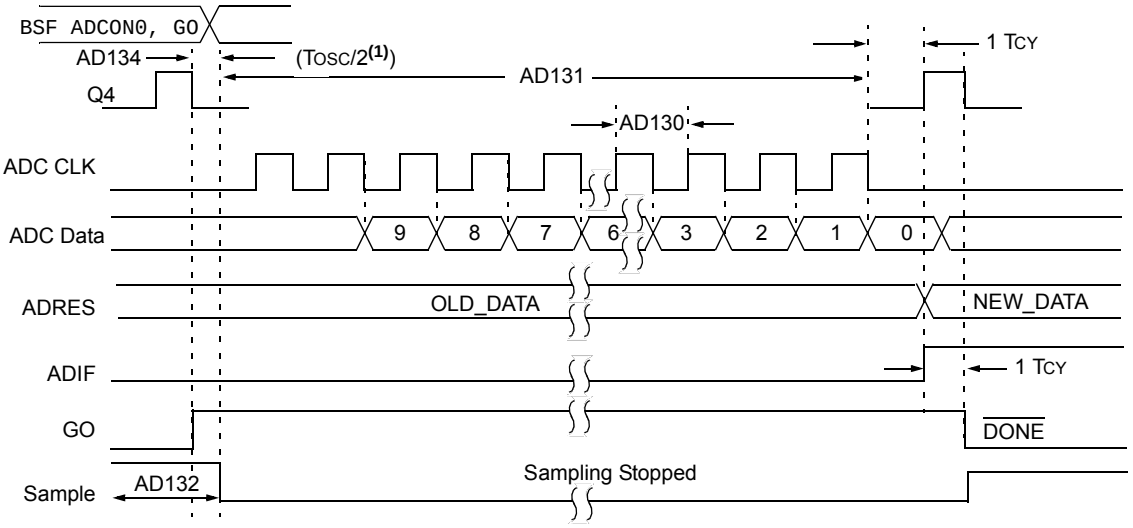
Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
AD130*	TAD	ADC Clock Period	0.25	—	25	µs	TOSC-based, -40°C to +85°C,VREF≥2.4V
			0.7	—	25	µs	TOSC-based, -40°C to +85°C, VREF<2.4V
			0.7	—	8	µs	TOSC-based, +86°C to +125°C
		ADC Internal FRC Oscillator Period	1.0	1.6	6.0	µs	ADCS<1:0> = 11 (ADFRC mode)
AD131	TCNV	Conversion Time (not including Acquisition Time)(1)	—	11	—	TAD	Set GO/$\overline{\text{DONE}}$ bit to conversion complete
AD132*	TACQ	Acquisition Time	—	5.0	—	µs	

* 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.

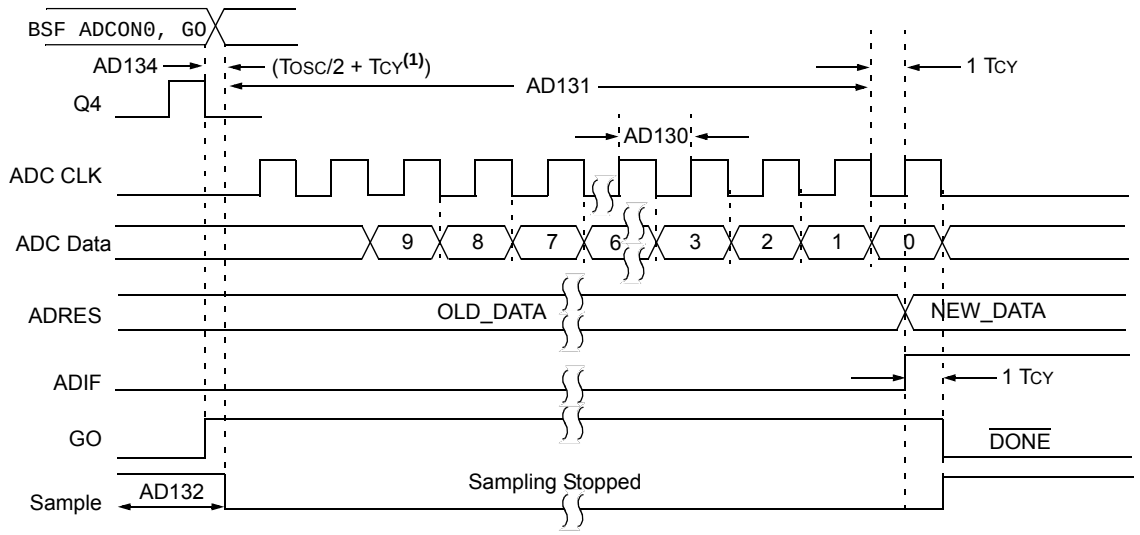
Note 1: The ADRES register may be read on the following TCY cycle.

FIGURE 21-9: PIC12LF1552 ADC CONVERSION TIMING (NORMAL MODE)



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

FIGURE 21-10: PIC12LF1552 ADC CONVERSION TIMING (SLEEP MODE)



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

22.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

The graphs and tables provided in this section are for **design guidance** and are **not tested**.

In some graphs or tables, the data presented are **outside specified operating range** (i.e., outside specified V_{DD} range). This is for **information only** and devices are ensured to operate properly only within the specified range.

Note:	The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "MAXIMUM", "Max.", "MINIMUM" or "Min." represents (mean + 3σ) or (mean - 3σ) respectively, where σ is a standard deviation, over each temperature range.

Note: Unless otherwise noted C_{IN} = 0.1 µF and T_A = 25°C.

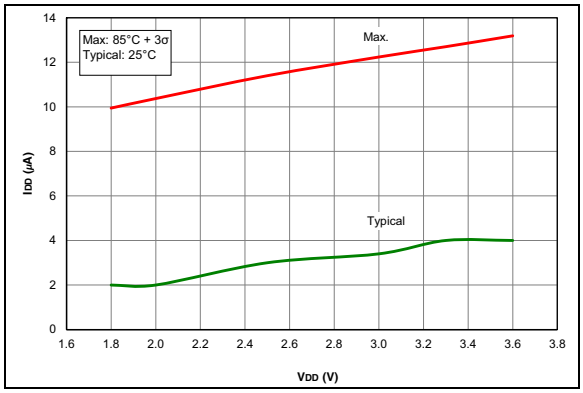


FIGURE 22-1: *IDD, EC Oscillator, Low-Power Mode, FOSC = 32 kHz*

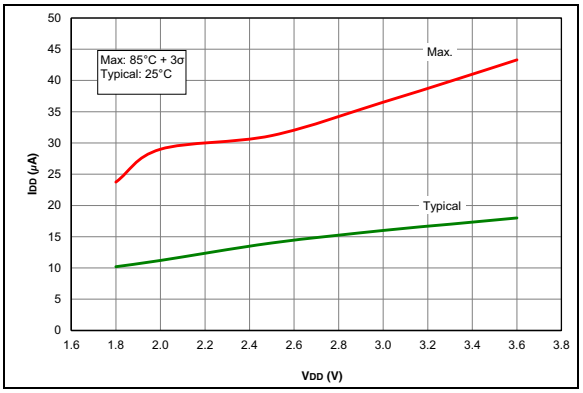


FIGURE 22-2: *IDD, Typical, EC Oscillator, Low-Power Mode, FOSC = 500 kHz*

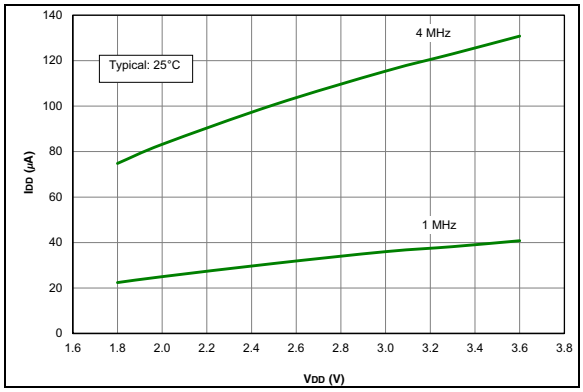


FIGURE 22-3: *IDD, Typical, EC Oscillator, High-Power Mode*

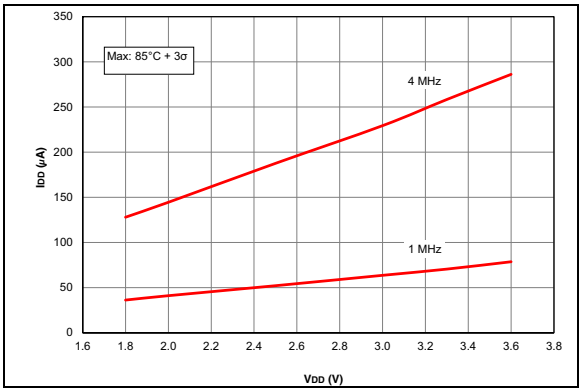


FIGURE 22-4: *IDD, Maximum, EC Oscillator, Medium-Power Mode*

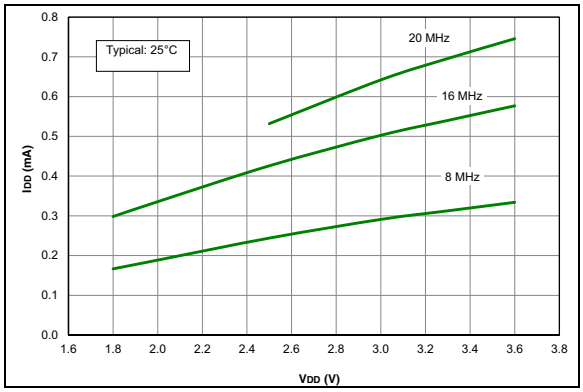


FIGURE 22-5: *IDD, Typical, EC Oscillator, High-Power Mode*

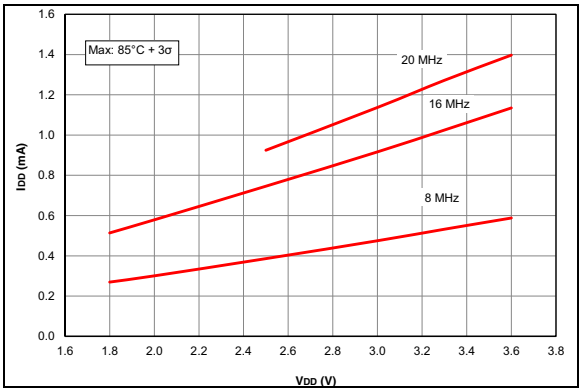


FIGURE 22-6: *IDD, Maximum, EC Oscillator, High-Power Mode*

Note: Unless otherwise noted C_{IN} = 0.1 µF and T_A = 25°C.

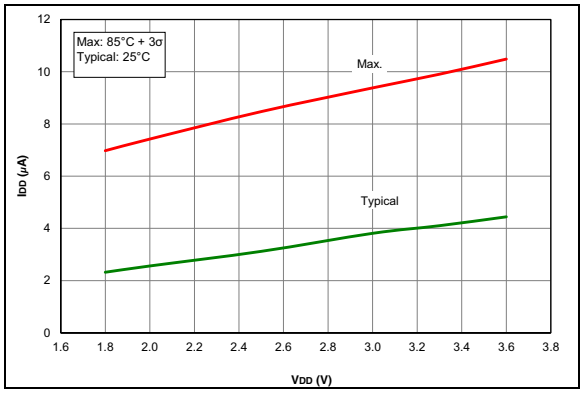


FIGURE 22-7: *IDD, LFINTOSC, FOSC = 31 kHz*

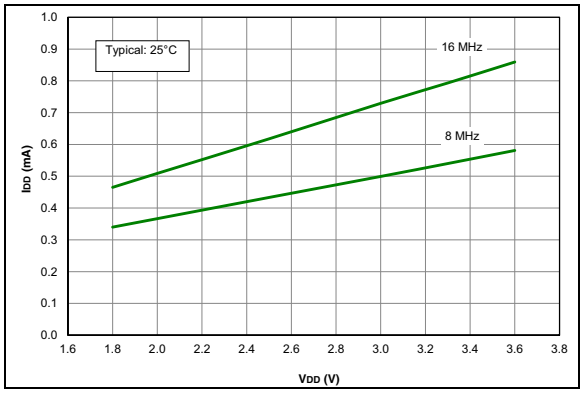


FIGURE 22-8: *IDD, Typical, HFINTOSC*

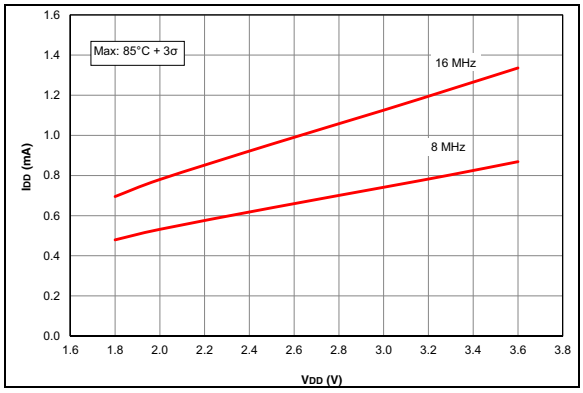


FIGURE 22-9: *IDD, Maximum, EC Oscillator, High-Power Mode*

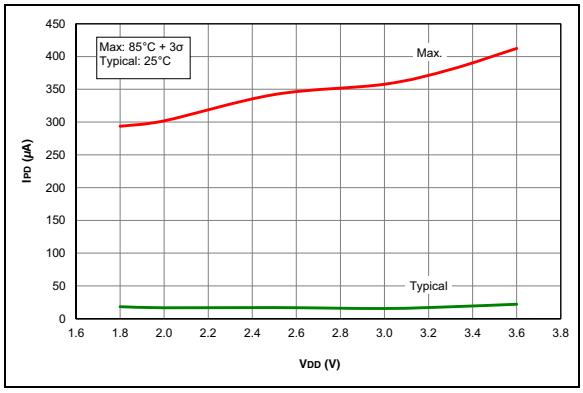


FIGURE 22-10: *IPD, Base, Low-Power Sleep Mode*

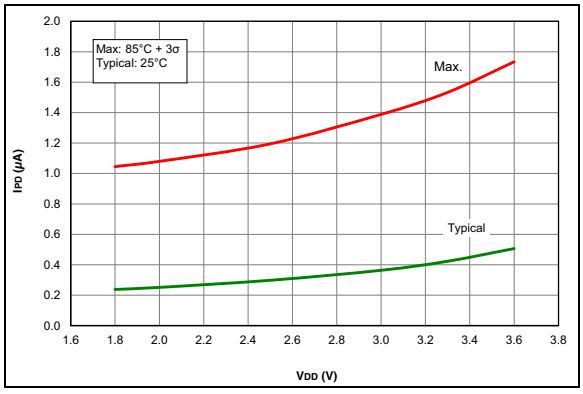


FIGURE 22-11: *IPD, Watchdog Timer (WDT)*

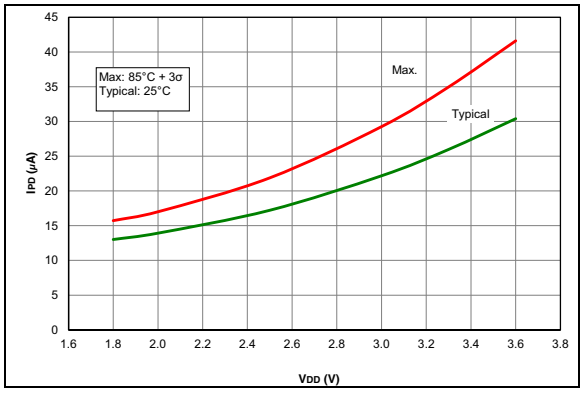


FIGURE 22-12: *IPD, Fixed Voltage Reference (FVR)*

Note: Unless otherwise noted C_{IN} = 0.1 µF and T_A = 25°C.

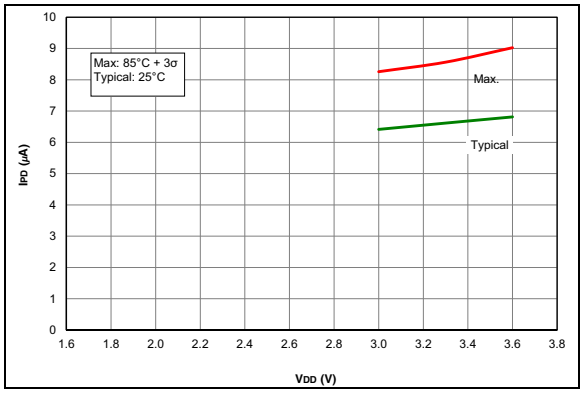


FIGURE 22-13: *I_{PD}, Brown-out Reset (BOR)*

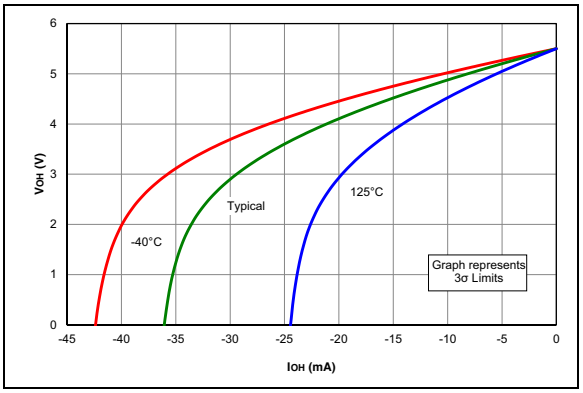


FIGURE 22-14: *V_{OH} vs. I_{OH}, Over Temperature, V_{DD} = 5.5V, PIC12FXXXX Only*

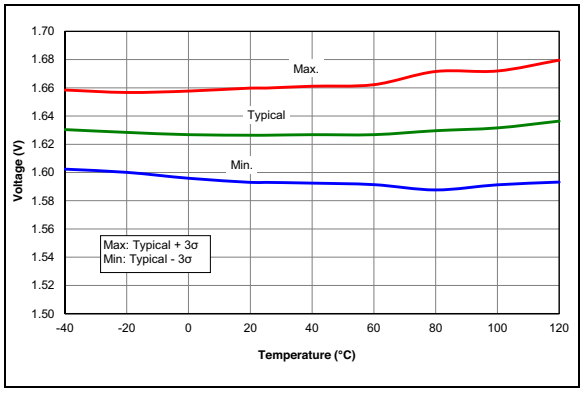


FIGURE 22-15: *POR Release Voltage*

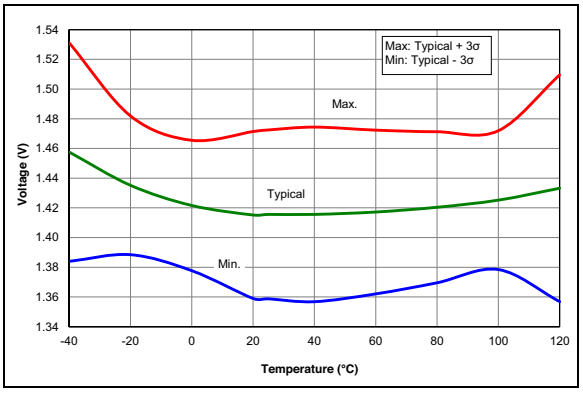


FIGURE 22-16: *POR Rearm Voltage, PIC12FXXXX Only*

23.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits and Starter Kits
- Third-party development tools

23.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

23.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

23.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

23.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

23.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

23.6 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.

23.7 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.

23.8 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.

23.9 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™).

23.10 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.

23.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

23.12 Third-Party Development Tools

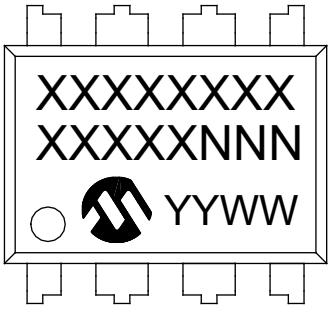
Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

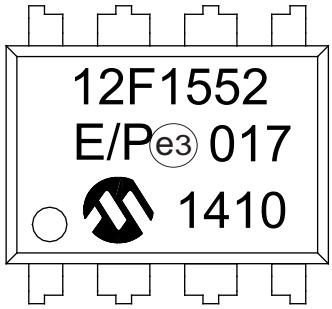
24.0 PACKAGING INFORMATION

24.1 Package Marking Information

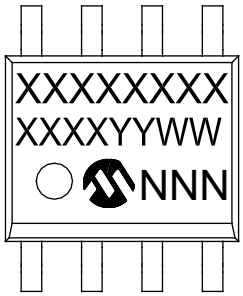
8-Lead PDIP (300 mil.)



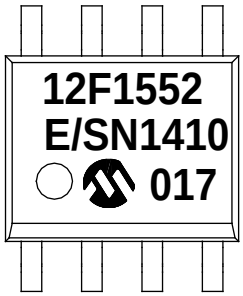
Example



8-Lead SOIC (3.90 mm)



Example

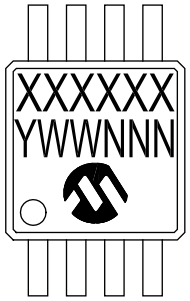


Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

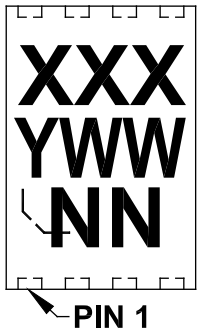
Package Marking Information (Continued)

8-Lead MSOP (3x3 mm)

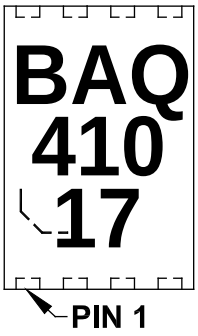


Example

8-Lead UDFN (2x3x0.5 mm)



Example



Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package.

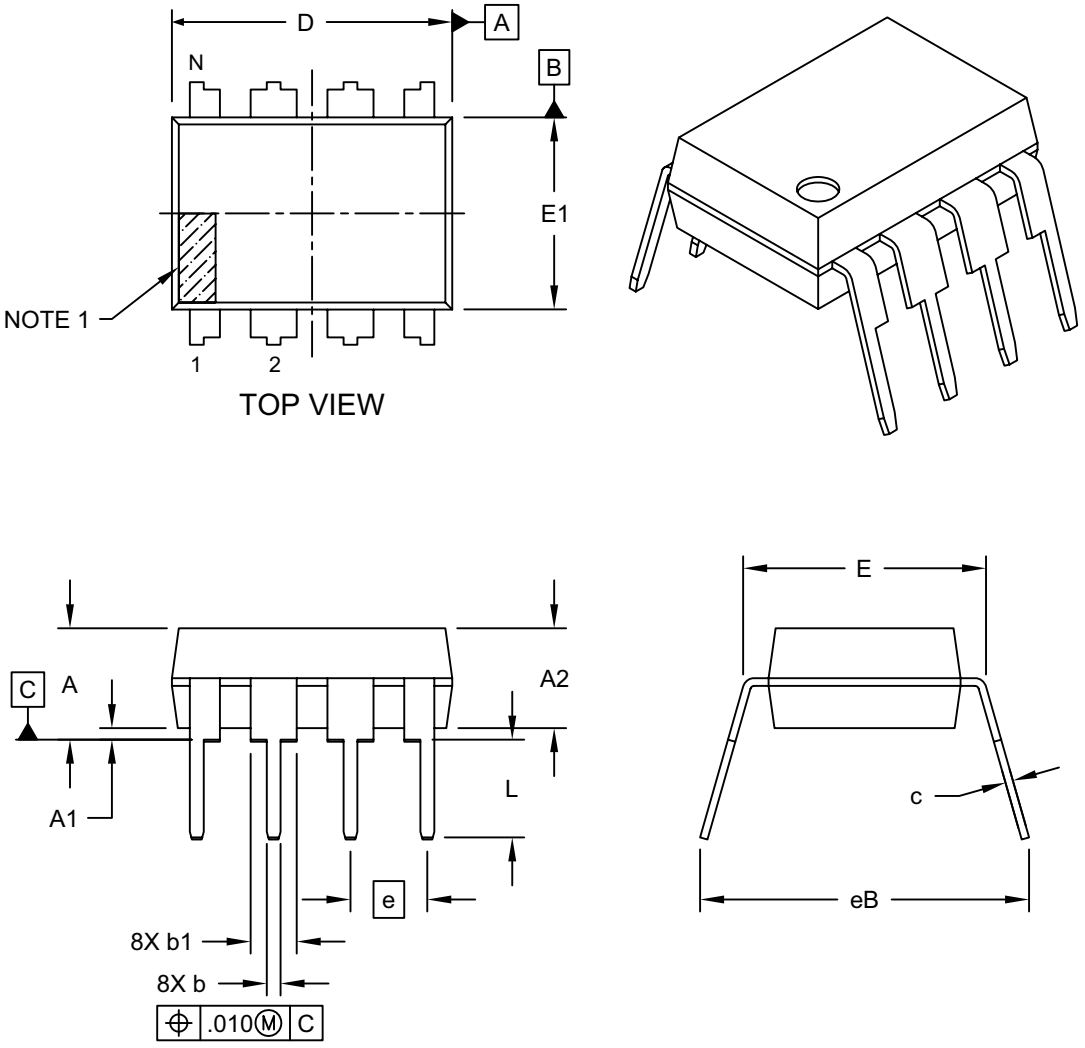
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

24.2 Package Details

The following sections give the technical details of the packages.

8-Lead Plastic Dual In-Line (P) - 300 mil Body [PDIP]

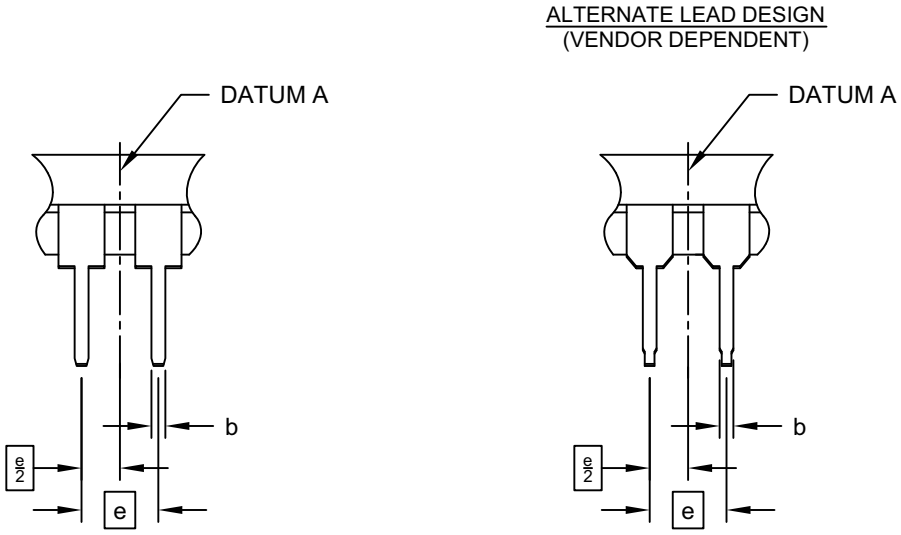
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing No. C04-018D Sheet 1 of 2

8-Lead Plastic Dual In-Line (P) - 300 mil Body [PDIP]

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



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		8	
Pitch	e		.100 BSC	
Top to Seating Plane	A	-	-	.210
Molded Package Thickness	A2	.115	.130	.195
Base to Seating Plane	A1	.015	-	-
Shoulder to Shoulder Width	E	.290	.310	.325
Molded Package Width	E1	.240	.250	.280
Overall Length	D	.348	.365	.400
Tip to Seating Plane	L	.115	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.040	.060	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	-	-	.430

Notes:

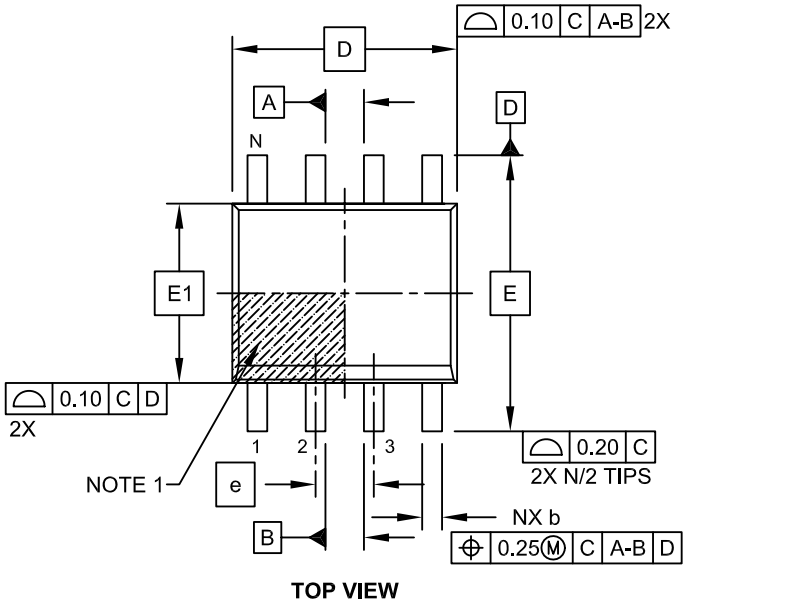
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

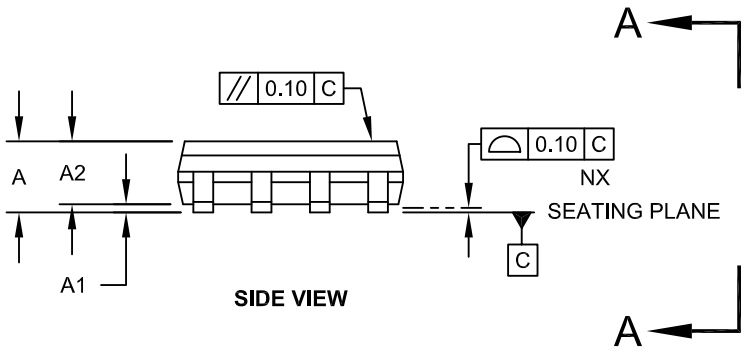
Microchip Technology Drawing No. C04-018D Sheet 2 of 2

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

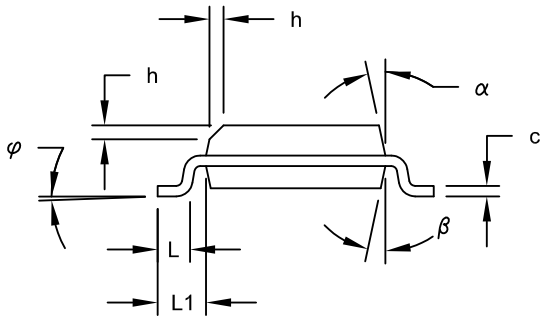
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW

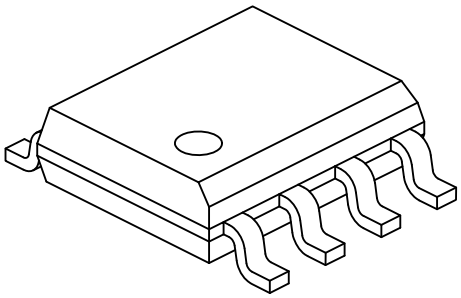


VIEW A-A

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

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



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		8	
Pitch	e		1.27 BSC	
Overall Height	A	-	-	1.75
Molded Package Thickness	A2	1.25	-	-
Standoff §	A1	0.10	-	0.25
Overall Width	E		6.00 BSC	
Molded Package Width	E1		3.90 BSC	
Overall Length	D		4.90 BSC	
Chamfer (Optional)	h	0.25	-	0.50
Foot Length	L	0.40	-	1.27
Footprint	L1		1.04 REF	
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.17	-	0.25
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

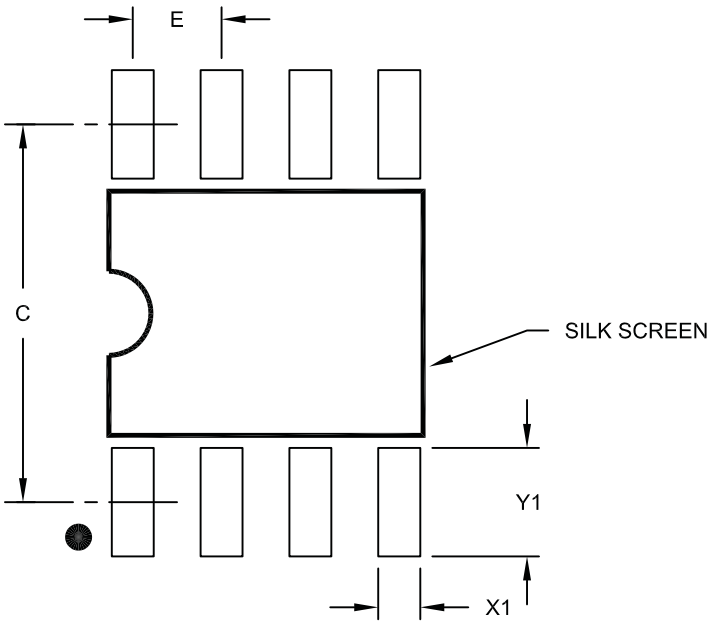
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

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

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		1.27 BSC	
Contact Pad Spacing	C		5.40	
Contact Pad Width (X8)	X1			0.60
Contact Pad Length (X8)	Y1			1.55

Notes:

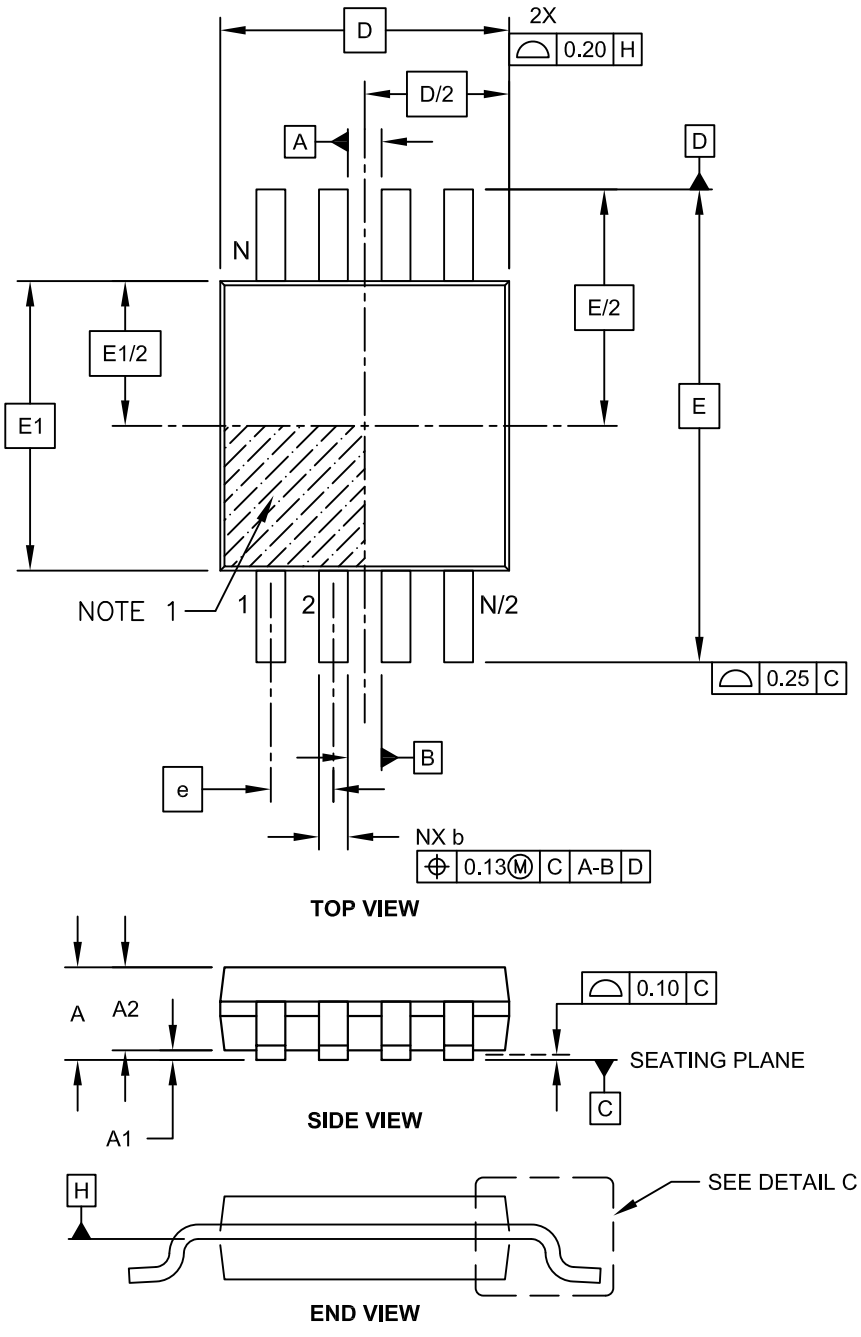
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2057A

8-Lead Plastic Micro Small Outline Package (MS) [MSOP]

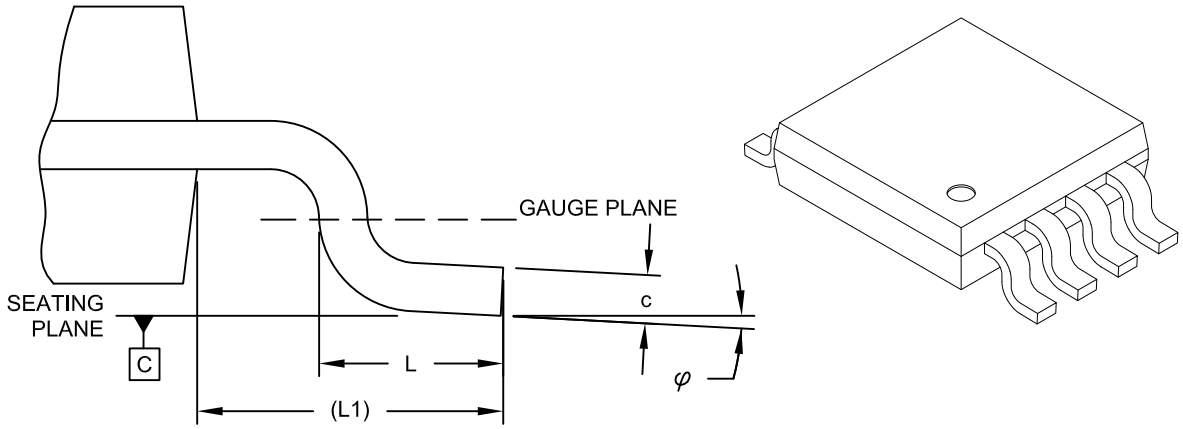
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-111C Sheet 1 of 2

8-Lead Plastic Micro Small Outline Package (MS) [MSOP]

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



DETAIL C

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		8	
Pitch	e		0.65 BSC	
Overall Height	A	-	-	1.10
Molded Package Thickness	A2	0.75	0.85	0.95
Standoff	A1	0.00	-	0.15
Overall Width	E		4.90 BSC	
Molded Package Width	E1		3.00 BSC	
Overall Length	D		3.00 BSC	
Foot Length	L	0.40	0.60	0.80
Footprint	L1		0.95 REF	
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.08	-	0.23
Lead Width	b	0.22	-	0.40

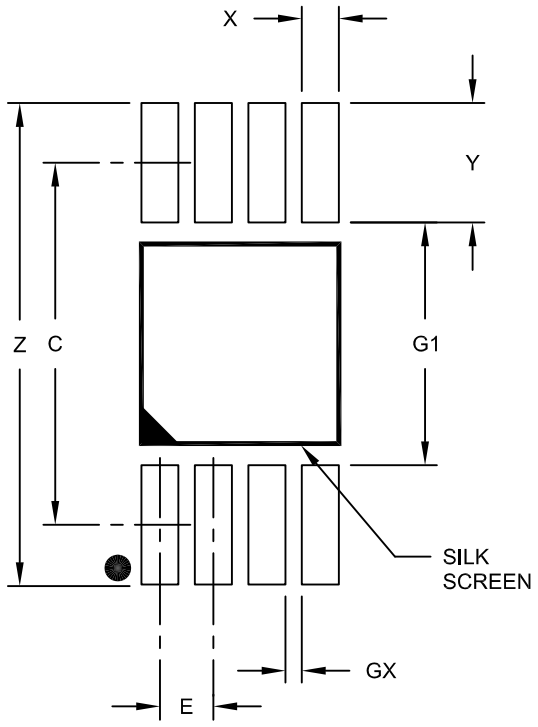
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-111C Sheet 2 of 2

8-Lead Plastic Micro Small Outline Package (MS) [MSOP]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Contact Pad Spacing	C		4.40	
Overall Width	Z			5.85
Contact Pad Width (X8)	X1			0.45
Contact Pad Length (X8)	Y1			1.45
Distance Between Pads	G1	2.95		
Distance Between Pads	GX	0.20		

Notes:

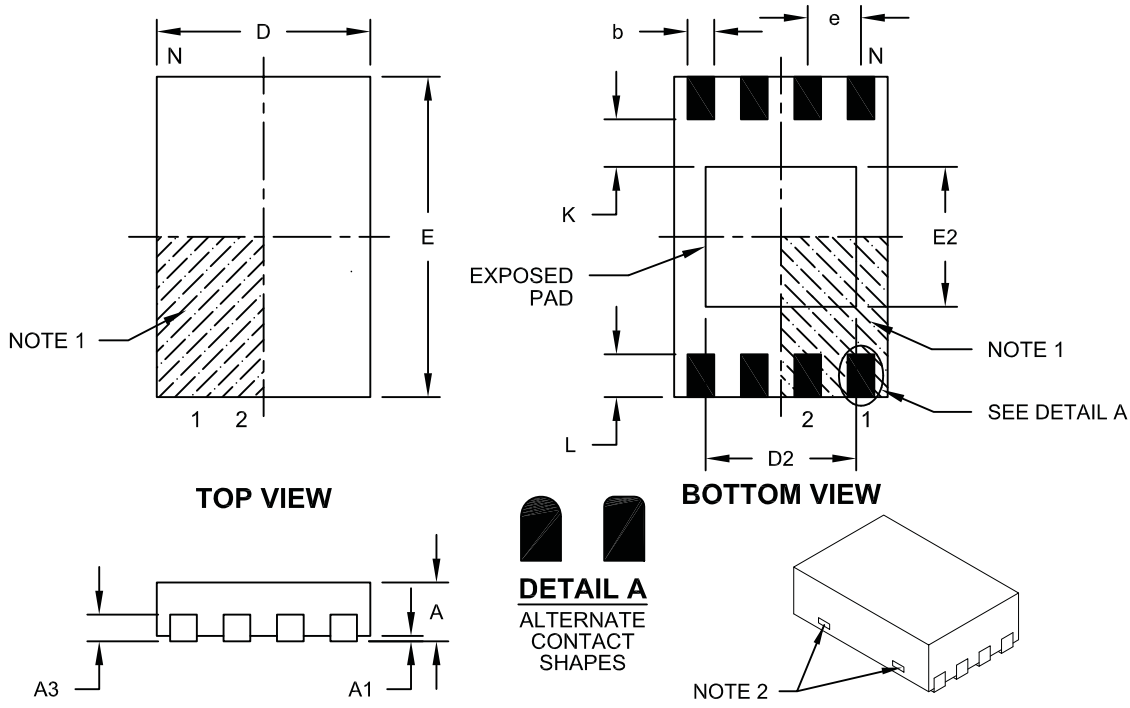
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2111A

8-Lead Plastic Dual Flat, No Lead Package (MU) – 2x3x0.5 mm Body [UDFN]

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



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		8	
Pitch	e		0.50 BSC	
Overall Height	A	0.45	0.50	0.55
Standoff	A1			0.07
Contact Thickness	A3		0.127 REF	
Overall Length	D	1.95	2.00	2.05
Overall Width	E	2.95	3.00	3.05
Exposed Pad Length	D2	1.30	1.40	1.50
Exposed Pad Width	E2	1.20	1.30	1.40
Contact Width	b	0.20	0.25	0.30
Contact Length	L	0.25	0.30	0.35
Contact-to-Exposed Pad	K		0.55 REF	

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package may have one or more exposed tie bars at ends.
3. Package is saw singulated
4. Dimensioning and tolerancing per ASME Y14.5M

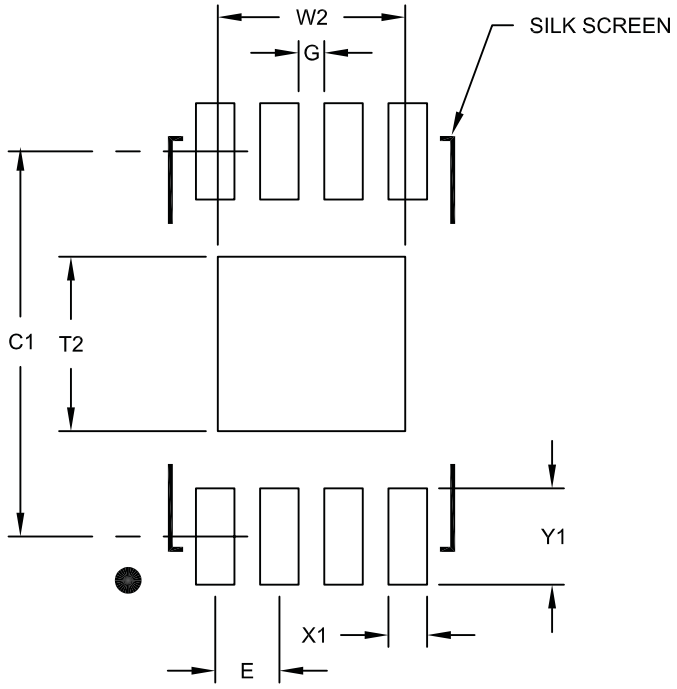
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing No. C04-136B

8-Lead Plastic Dual Flat, No Lead Package (MU) – 2x3x0.5 mm Body [UDFN]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	W2			1.46
Optional Center Pad Length	T2			1.36
Contact Pad Spacing	C1		3.00	
Contact Pad Width (X8)	X1			0.30
Contact Pad Length (X8)	Y1			0.75
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2136A

APPENDIX A: DATA SHEET REVISION HISTORY

Revision A(12/2012)

Original release.

Revision B (01/2013)

Revised Product ID System – Corrected packaged code MF to MU.

Revision C (12/2013)

Updated Sections 16, CVD and 21, Electrical Specifications; Other minor corrections.

Revision D (04/2014)

Updated Peripheral Features on page 1; Updated Figure 1-1 and Equation 15-1; Updated Sections 18.3, 18.3.2, 18.6.7.4, 21.2 and 21.3; Updated titles in Figure 21-6 and Table 21-4; Updated Note 1 in Figure 21-7 and Note 4 in Table 21-6; Added DC and AC Characteristics Graphs and Charts; Updated Chapter 24, Packaging Information; Other minor corrections.

Revision E (01/2015)

Updated Tables 21-6 and 21-7; Other minor corrections.

Revision F (10/2016)

Updated eXtreme Low-Power (XLP) Features; Updated Example 3-2; Updated Table 5-1; Added section 5.3.2; Updated Table 21-2. Updated section 15.2.6; Updated Example 15-1; Updated Register 6-2; Updated Table 18-4; Updated section 21.3; Updated Table 21-4. Added new char graphs. Other minor corrections.

THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: http://www.microchip.com/support

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	[X](1)	-	X	/XX	XXX
Device	Tape and Reel Option		Temperature Range	Package	Pattern

Device:	PIC12LF1552	
Tape and Reel Option:	Blank T	= Standard packaging (tube or tray) = Tape and Reel(1)
Temperature Range:	I E	= -40°C to +85°C (Industrial) = -40°C to +125°C (Extended)
Package:(2)	MU MS P SN	= Micro Lead Frame (UDFN) 2x3 = MSOP = Plastic DIP = SOIC
Pattern:	QTP, SQTP, Code or Special Requirements (blank otherwise)	

Examples:

a) PIC12LF1552T - I/SN
Tape and Reel,
Industrial temperature,
SOIC package

b) PIC12LF1552 - I/P
Industrial temperature
PDIP package

c) PIC12LF1552 - E/MU
Extended temperature,
UDFN package

Note 1: Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

2: Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-1045-4

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15