PIC18C601/801

High-Performance ROM-less Microcontrollers with External Memory Bus

High Performance RISC CPU:

- C compiler optimized architecture instruction set
- Linear program memory addressing up to 2 Mbytes
- Linear data memory addressing to 4 Kbytes

Device	External Program Memory		On-Chip RAM (bytes)
	On-Chip		
	Maximum Addressing (bytes)	Maximum Single Word Instructions	
PIC18C601	256K	128K	1.5K
PIC18C801	2M	1M	1.5K

- Up to 160 ns instruction cycle:
  - DC - 25 MHz clock input
  - 4 MHz - 6 MHz clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Up to 47 I/O pins with individual direction control
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter (time-base for CCP)
- Timer2 module: 8-bit timer/counter with 8-bit period register
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules
  CCP pins can be configured as:
  - Capture input: 16-bit, max. resolution 10 ns
  - Compare is 16-bit, max. resolution 160 ns (T_{CY})
  - PWM output: PWM resolution is 1- to 10-bit
    Max. PWM freq. @:
    8-bit resolution = 99 kHz
    10-bit resolution = 24.4 kHz
- Master Synchronous Serial Port (MSSP) with two modes of operation:
  - 3-wire SPI™ (Supports all 4 SPI modes)
  - I^2C™ Master and Slave mode
- Addressable USART module: Supports Interrupt on Address bit

Advanced Analog Features:

- 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
  - Conversion available during SLEEP
  - DNL = ±1 LSb, INL = ±1 LSb
  - Up to 12 channels available
- Programmable Low Voltage Detection (LVD) module
  - Supports interrupt on Low Voltage Detection

Special Microcontroller Features:

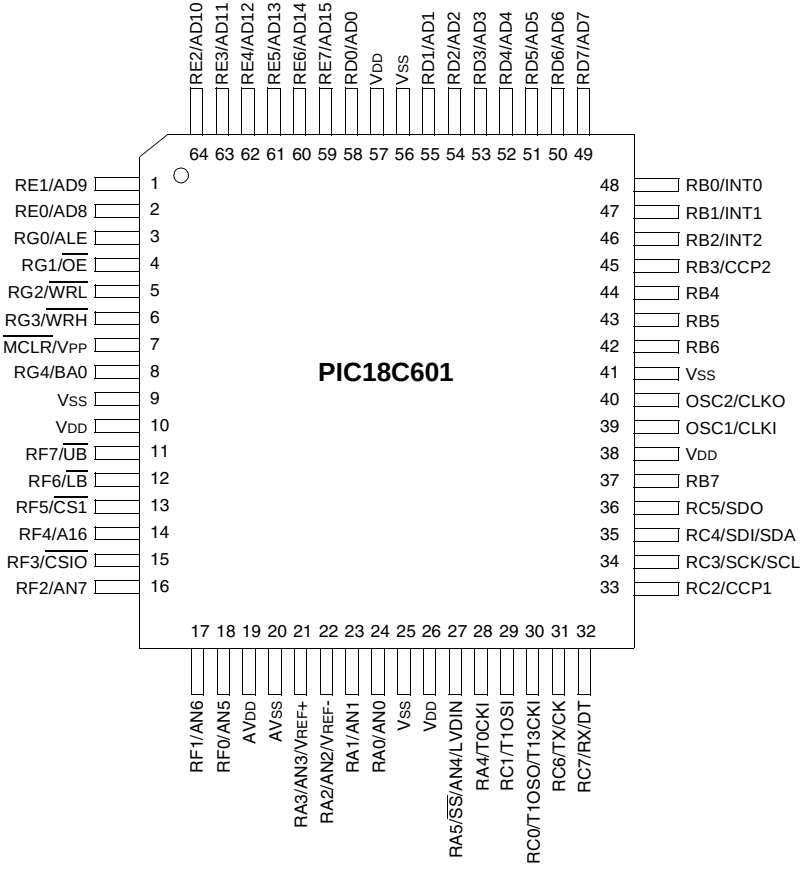
- Power-on Reset (POR), Power-up Timer (PWRT), and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator
- On-chip Boot RAM for boot loader application
- 8-bit or 16-bit external memory interface modes
- Up to two software programmable chip select signals ($\overline{CS1}$ and $\overline{CS2}$)
- One programmable chip I/O select signal ($\overline{CSIO}$) for memory mapped I/O expansion
- Power saving SLEEP mode
- Different oscillator options, including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input

CMOS Technology:

- Low power, high speed CMOS technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption

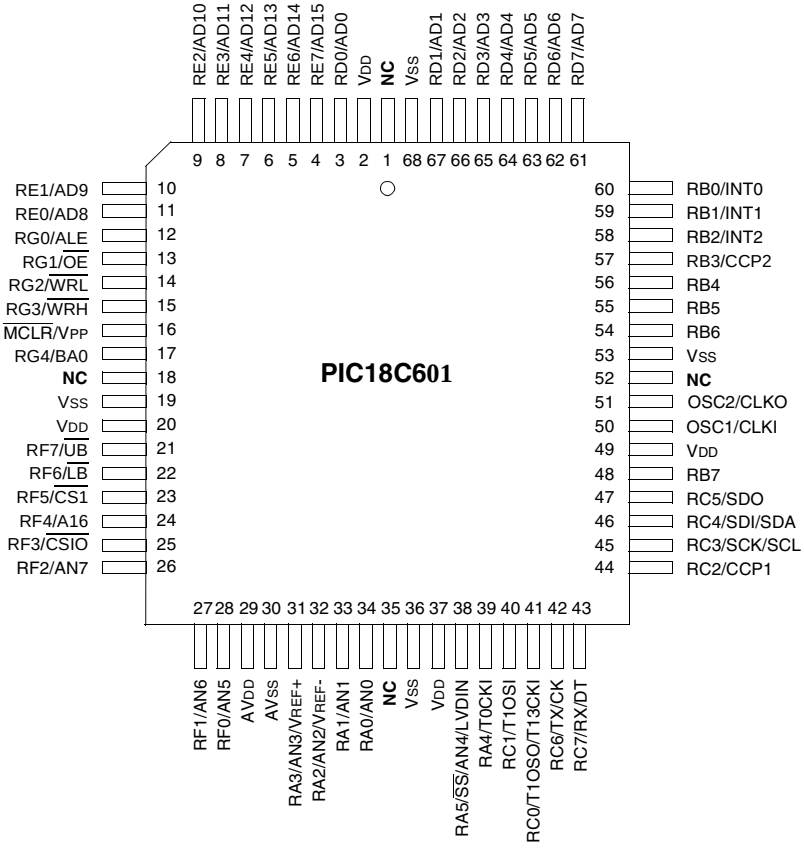
Pin Diagrams

64-Pin TQFP



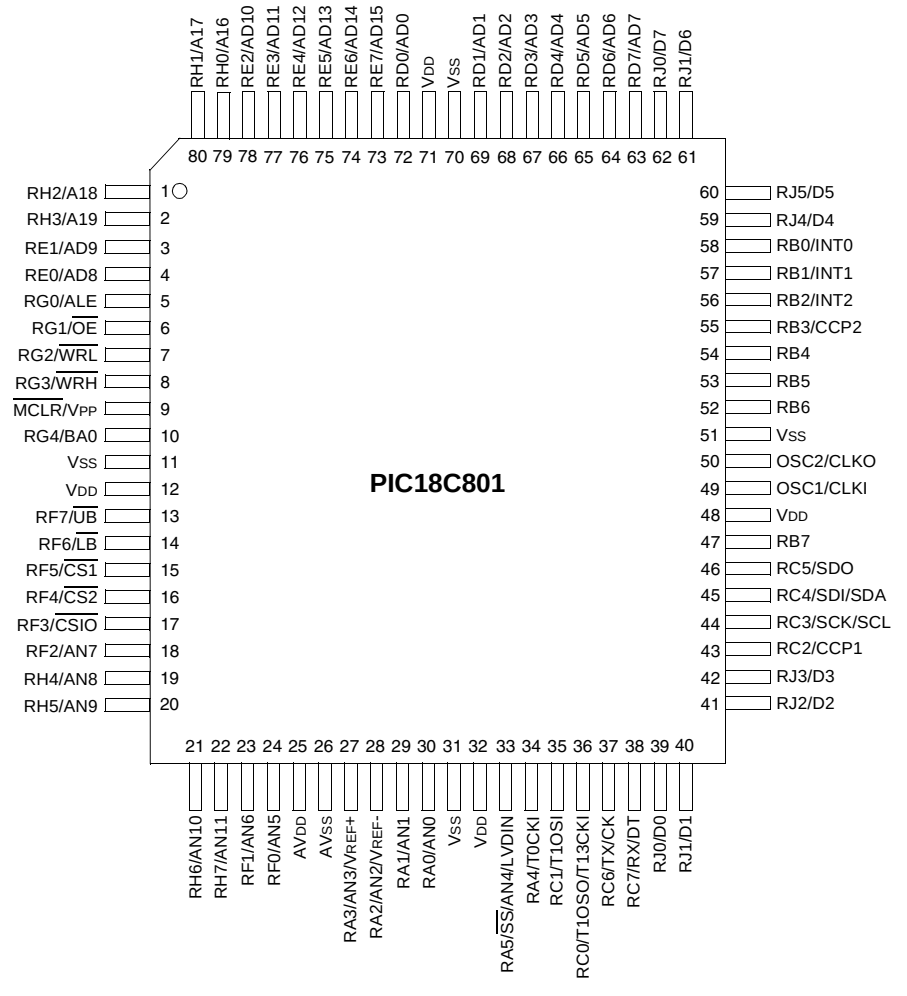
Pin Diagrams (Cont.'d)

68-Pin PLCC



Pin Diagrams (Cont.'d)

80-Pin TQFP



Pin Diagrams (Cont.'d)

84-Pin PLCC

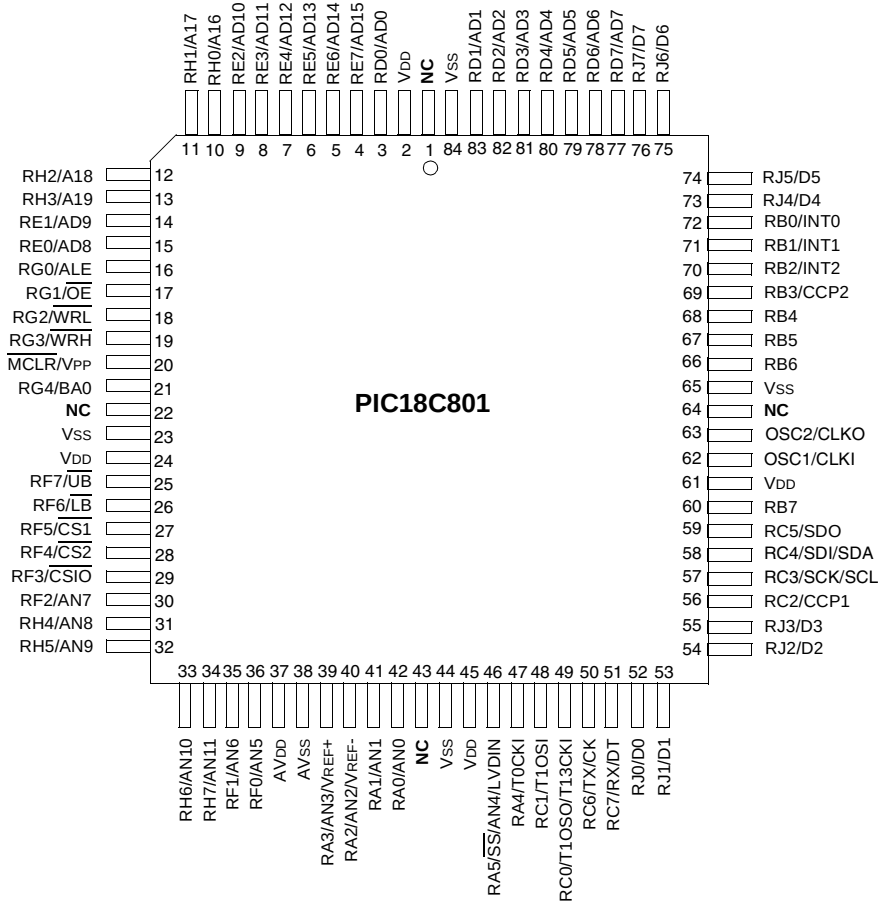


Table of Contents

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@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. 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 Web site 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., DS30000A is version A of document DS30000).

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 Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

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

Customer Notification System

Register on our web site at **www.microchip.com/cn** to receive the most current information on all of our products.

NOTES:

1.0 DEVICE OVERVIEW

This document contains device specific information for the following two devices:

1. PIC18C601
2. PIC18C801

The PIC18C601 is available in 64-pin TQFP and 68-pin PLCC packages. The PIC18C801 is available in 80-pin TQFP and 84-pin PLCC packages.

An overview of features is shown in Table 1-1.

Device block diagrams are provided in Figure 1-1 for the 64/68-pin configuration, and Figure 1-2 for the 80/84-pin configuration. The pinouts for both packages are listed in Table 1-2.

TABLE 1-1: DEVICE FEATURES

Features		PIC18C601	PIC18C801
Operating Frequency		DC - 25 MHz	DC - 25 MHz
External Program Memory	Bytes	256K	2M
	Max. # of Single Word Instructions	128K	1M
Data Memory (Bytes)		1536	1536
Interrupt Sources		15	15
I/O Ports		Ports A - G	Ports A - H, J
Timers		4	4
Capture/Compare/PWM modules		2	2
Serial Communications		MSSP, Addressable USART	MSSP, Addressable USART
10-bit Analog-to-Digital Module		8 input channels	12 input channels
RESETS (and Delays)		POR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low Voltage Detect		Yes	Yes
8-bit External Memory Interface		Yes	Yes
8-bit De-multiplexed External Memory Interface		No	Yes
16-bit External Memory Interfaces		Yes	Yes
On-chip Chip Select Signals		$\overline{CS1}$	$\overline{CS1}$, $\overline{CS2}$
On-chip I/O Chip Select Signal		Yes	Yes
Instruction Set		75 Instructions	75 Instructions
Packages		64-pin TQFP 68-pin PLCC	80-pin TQFP 84-pin PLCC

FIGURE 1-1: PIC18C601 BLOCK DIAGRAM

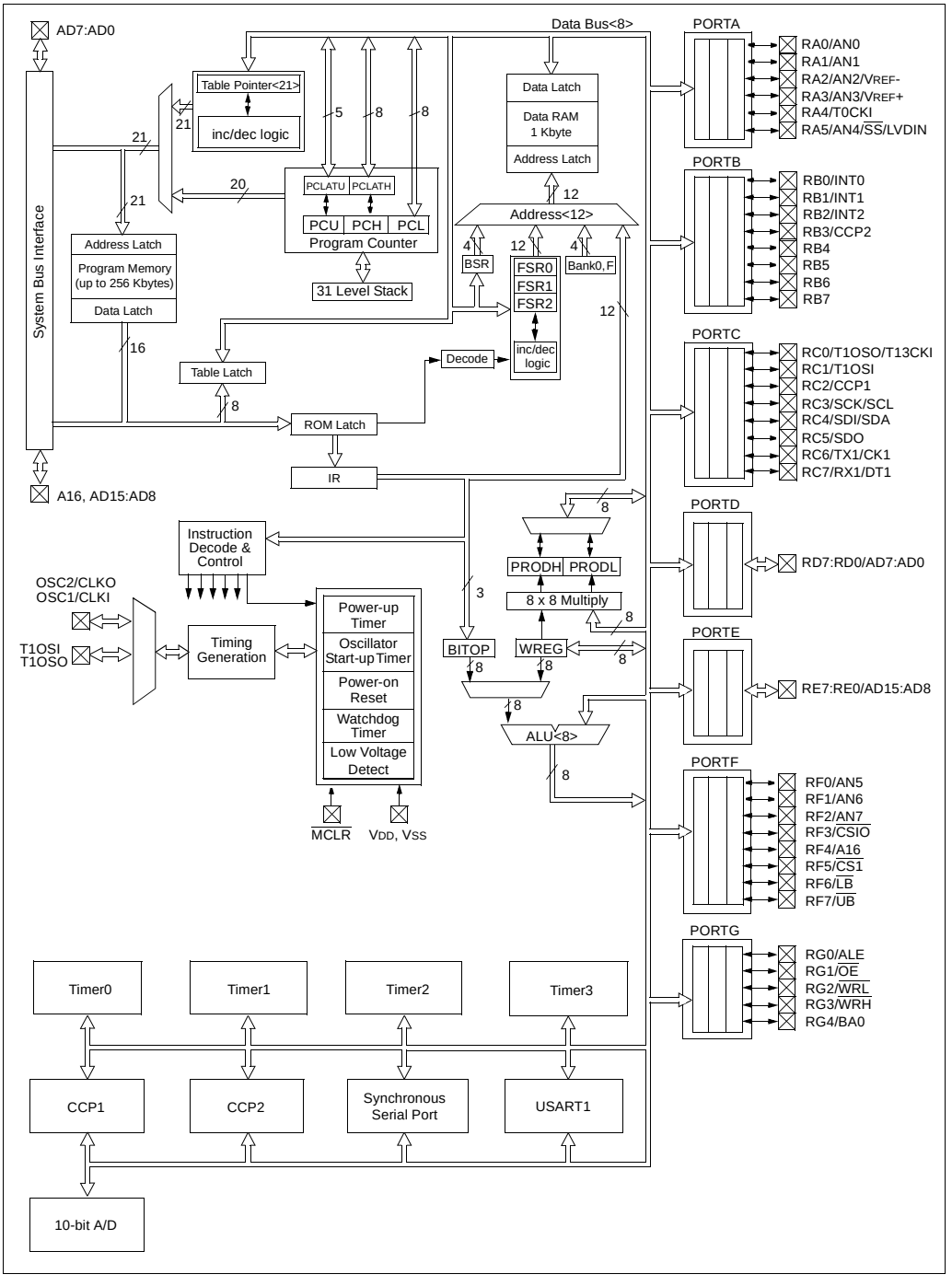


FIGURE 1-2: PIC18C801 BLOCK DIAGRAM

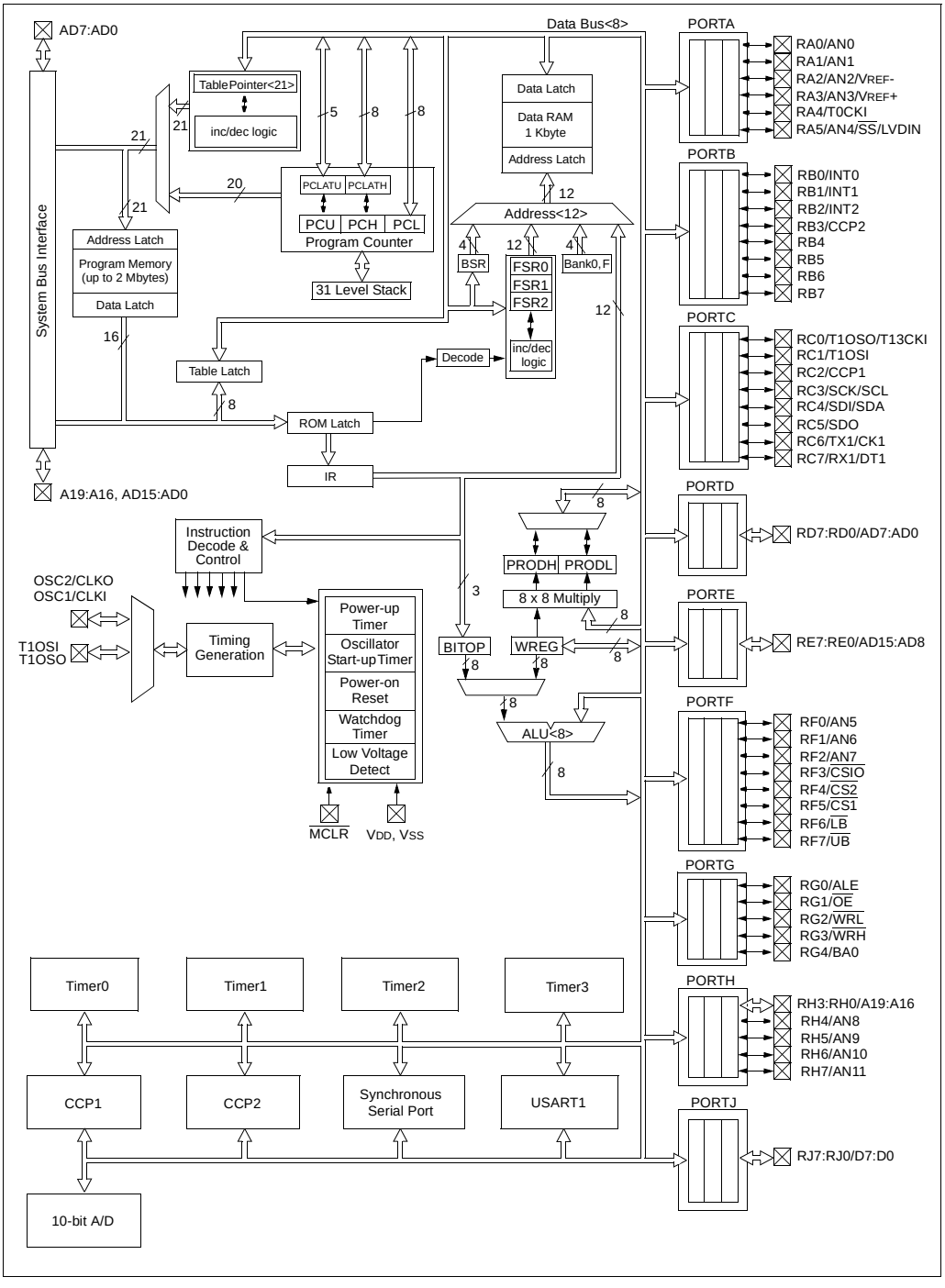


TABLE 1-2: PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
$\overline{MCLR}$/VPP	7	16	9	20			
$\overline{MCLR}$					I	ST	Master clear (RESET) input. This pin is an active low RESET to the device.
VPP					P		Programming voltage input.
NC	—	1, 18, 35, 52	—	1, 22, 43, 64	—	—	These pins should be left unconnected.
OSC1/CLKI	39	50	49	62			
OSC1					I	CMOS/ST	Oscillator crystal input or external clock source input. ST buffer when in RC mode. Otherwise CMOS.
CLKI					I	CMOS	External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO	40	51	50	63			
OSC2					O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO					O	—	In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to VDD)

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTA is a bi-directional I/O port.
RA0/AN0	24	34	30	42			
RA0					I/O	TTL	Digital I/O.
AN0					I	Analog	Analog input 0.
RA1/AN1	23	33	29	41			
RA1					I/O	TTL	Digital I/O.
AN1					I	Analog	Analog input 1.
RA2/AN2/VREF-	22	32	28	40			
RA2					I/O	TTL	Digital I/O.
AN2					I	Analog	Analog input 2.
VREF-					I	Analog	A/D reference voltage (Low) input.
RA3/AN3/VREF+	21	31	27	39			
RA3					I/O	TTL	Digital I/O.
AN3					I	Analog	Analog input 3.
VREF+					I	Analog	A/D reference voltage (High) input.
RA4/T0CKI	28	39	34	47			
RA4					I/O	ST/OD	Digital I/O – Open drain when configured as output.
T0CKI					I	ST	Timer0 external clock input.
RA5/AN4/$\overline{SS}$/LVDIN	27	38	33	46			
RA5					I/O	TTL	Digital I/O.
AN4					I	Analog	Analog input 4.
$\overline{SS}$					I	ST	SPI slave select input.
LVDIN					I	Analog	Low voltage detect input.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to VDD)

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0	48	60	58	72			
RB0					I/O	TTL	Digital I/O.
INT0					I	ST	External interrupt 0.
RB1/INT1	47	59	57	71			
RB1					I/O	TTL	Digital I/O.
INT1					I	ST	External interrupt 1.
RB2/INT2	46	58	56	70			
RB2					I/O	TTL	Digital I/O.
INT2					I	ST	External interrupt 2.
RB3/CCP2	45	57	55	69			
RB3					I/O	TTL	Digital I/O.
CCP2					I/O	ST	Capture2 input, Compare2 output, PWM2 output.
RB4	44	56	54	68	I/O	TTL	Digital I/O, Interrupt-on-change pin.
RB5	43	55	53	67	I/O	TTL	Digital I/O, Interrupt-on-change pin.
RB6	42	54	52	66	I/O	TTL	Digital I/O, Interrupt-on-change pin.
					I	ST	ICSP programming clock.
RB7	37	48	47	60	I/O	TTL	Digital I/O, Interrupt-on-change pin.
					I/O	ST	ICSP programming data.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
Analog = Analog input
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to V_{DD})

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTC is a bi-directional I/O port.
RC0/T1OSO/T13CKI	30	41	36	49			
RC0					I/O	ST	Digital I/O.
T1OSO					O	—	Timer1 oscillator output.
T13CKI					I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI	29	40	35	48			
RC1					I/O	ST	Digital I/O.
T1OSI					I	CMOS	Timer1 oscillator input.
RC2/CCP1	33	44	43	56			
RC2					I/O	ST	Digital I/O.
CCP1					I/O	ST	Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	34	45	44	57			
RC3					I/O	ST	Digital I/O.
SCK					I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL					I/O	ST	Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	35	46	45	58			
RC4					I/O	ST	Digital I/O.
SDI					I	ST	SPI data in.
SDA					I/O	ST	I^2C data I/O.
RC5/SDO	36	47	46	59			
RC5					I/O	ST	Digital I/O.
SDO					O	—	SPI data out.
RC6/TX/CK	31	42	37	50			
RC6					I/O	ST	Digital I/O.
TX					O	—	USART asynchronous transmit.
CK					I/O	ST	USART synchronous clock.
RC7/RX/DT	32	43	38	51			
RC7					I/O	ST	Digital I/O.
RX					I	ST	USART asynchronous receive.
DT					I/O	ST	USART synchronous data.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to V_{DD})

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTD is a bi-directional I/O port. These pins have TTL input buffers when external memory is enabled.
RD0/AD0	58	3	72	3			
RD0					I/O	ST	Digital I/O.
AD0					I/O	TTL	External memory address/data 0.
RD1/AD1	55	67	69	83			
RD1					I/O	ST	Digital I/O.
AD1					I/O	TTL	External memory address/data 1.
RD2/AD2	54	66	68	82			
RD2					I/O	ST	Digital I/O.
AD2					I/O	TTL	External memory address/data 2.
RD3/AD3	53	65	67	81			
RD3					I/O	ST	Digital I/O.
AD3					I/O	TTL	External memory address/data 3.
RD4/AD4	52	64	66	80			
RD4					I/O	ST	Digital I/O.
AD4					I/O	TTL	External memory address/data 4.
RD5/AD5	51	63	65	79			
RD5					I/O	ST	Digital I/O.
AD5					I/O	TTL	External memory address/data 5.
RD6/AD6	50	62	64	78			
RD6					I/O	ST	Digital I/O.
AD6					I/O	TTL	External memory address/data 6.
RD7/AD7	49	61	63	77			
RD7					I/O	ST	Digital I/O.
AD7					I/O	TTL	External memory address/data 7.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to V_{DD})

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTE is a bi-directional I/O port.
RE0/AD8	2	11	4	15			
RE0					I/O	ST	Digital I/O.
AD8					I/O	TTL	External memory address/data 8.
RE1/AD9	1	10	3	14			
RE1					I/O	ST	Digital I/O.
AD9					I/O	TTL	External memory address/data 9.
RE2/AD10	64	9	78	9			
RE2					I/O	ST	Digital I/O.
AD10					I/O	TTL	External memory address/data 10.
RE3/AD11	63	8	77	8			
RE3					I/O	ST	Digital I/O.
AD11					I/O	TTL	External memory address/data 11.
RE4/AD12	62	7	76	7			
RE4					I/O	ST	Digital I/O.
AD12					I/O	TTL	External memory address/data 12.
RE5/AD13	61	6	75	6			
RE5					I/O	ST	Digital I/O.
AD13					I/O	TTL	External memory address/data 13.
RE6/AD14	60	5	74	5			
RE6					I/O	ST	Digital I/O.
AD14					I/O	TTL	External memory address/data 14.
RE7/AD15	59	4	73	4			
RE7					I/O	ST	Digital I/O.
AD15					I/O	ST	External memory address/data 15.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
Analog = Analog input
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTF is a bi-directional I/O port.
RF0/AN5	18	28	24	36			
RF0					I/O	ST	Digital I/O.
AN5					I	Analog	Analog input 5.
RF1/AN6	17	27	23	35			
RF1					I/O	ST	Digital I/O.
AN6					I	Analog	Analog input 6.
RF2/AN7	16	26	18	30			
RF2					I/O	ST	Digital I/O.
AN7					I	Analog	Analog input 7.
RF3/$\overline{\text{CSIO}}$	15	25	17	29			
RF3					I/O	ST	Digital I/O.
$\overline{\text{CSIO}}$					I/O	ST	System bus chip select I/O.
RF4/A16	14	24	—	—			
RF4/$\overline{\text{CS2}}$	—	—	16	28			
RF4					I/O	ST	Digital I/O.
A16					I/O	TTL	External memory address 16.
$\overline{\text{CS2}}$					O	TTL	Chip select 2.
RF5/$\overline{\text{CS1}}$	13	23	15	27			
RF5					I/O	ST	Digital I/O.
$\overline{\text{CS1}}$					O	TTL	Chip select 1.
RF6/$\overline{\text{LB}}$	12	22	14	26			
RF6					I/O	ST	Digital I/O.
$\overline{\text{LB}}$					O	TTL	Low byte select signal for external memory interface.
RF7/$\overline{\text{UB}}$	11	21	13	25			
RF7					I/O	ST	Digital I/O.
$\overline{\text{UB}}$					O	TTL	High byte select signal for external memory interface.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
Analog = Analog input
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to V_{DD})

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTG is a bi-directional I/O port.
RG0/ALE	3	12	5	16			
RG0					I/O	ST	Digital I/O.
ALE					O	TTL	Address Latch Enable.
RG1/$\overline{OE}$	4	13	6	17			
RG1					I/O	ST	Digital I/O.
$\overline{OE}$					O	TTL	Output Enable.
RG2/$\overline{WRL}$	5	14	7	18			
RG2					I/O	ST	Digital I/O.
$\overline{WRL}$					O	TTL	Write Low control.
RG3/$\overline{WRH}$	6	15	8	19			
RG3					I/O	ST	Digital I/O.
$\overline{WRH}$					O	TTL	Write High control.
RG4/BA0	8	17	10	21			
RG4					I/O	ST	Digital I/O.
BA0					O	TTL	System bus byte address 0.
							PORTH is a bi-directional I/O port.
RH0/A16	—	—	79	10			
RH0					I/O	ST	Digital I/O.
A16					O	TTL	External memory address 16.
RH1/A17	—	—	80	11			
RH1					I/O	ST	Digital I/O.
A17					O	—	External memory address 17.
RH2/A18	—	—	1	12			
RH2					I/O	ST	Digital I/O.
A18					O	—	External memory address 18.
RH3/A19	—	—	2	13			
RH3					I/O	ST	Digital I/O.
A19					O	—	External memory address 19.
RH4/AN8	—	—	19	31			
RH4					I/O	ST	Digital I/O.
AN8					I	Analog	Analog input 8.
RH5/AN9	—	—	20	32			
RH5					I/O	ST	Digital I/O.
AN9					I	Analog	Analog input 9.
RH6/AN10	—	—	21	33			
RH6					I/O	ST	Digital I/O.
AN10					I	Analog	Analog input 10.
RH7/AN11	—	—	22	34			
RH7					I/O	ST	Digital I/O.
AN11					I	Analog	Analog input 11.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
Analog = Analog input
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C601		PIC18C801				
	TQFP	PLCC	TQFP	PLCC			
							PORTJ is a bi-directional I/O port.
RJ0/D0	—	—	39	52			
RJ0					I/O	ST	Digital I/O.
D0					I/O	TTL	System bus data bit 0.
RJ1/D1	—	—	40	53			
RJ1					I/O	ST	Digital I/O.
D1					I/O	TTL	System bus data bit 1.
RJ2/D2	—	—	41	54			
RJ2					I/O	ST	Digital I/O.
D2					I/O	TTL	System bus data bit 2.
RJ3/D3	—	—	42	55			
RJ3					I/O	ST	Digital I/O.
D3					I/O	TTL	System bus data bit 3.
RJ4/D4	—	—	59	73			
RJ4					I/O	ST	Digital I/O.
D4					I/O	TTL	System bus data bit 4.
RJ5/D5	—	—	60	74			
RJ5					I/O	ST	Digital I/O.
D5					I/O	TTL	System bus data bit 5.
RJ6/D6	—	—	61	75			
RJ6					I/O	ST	Digital I/O.
D6					I/O	TTL	System bus data bit 6.
RJ7/D7	—	—	62	76			
RJ7					I/O	ST	Digital I/O.
D7					I/O	TTL	System bus data bit 7.
VSS	9, 25, 41, 56	19, 36, 53, 68	11,31, 51, 70	23, 44, 65, 84	P	—	Ground reference for logic and I/O pins.
VDD	10,26, 38, 57	2, 20, 37, 49	12,32, 48, 71	2, 24, 45, 61	P	—	Positive supply for logic and I/O pins.
AVSS	20	30	26	38	P	—	Ground reference for analog modules.
AVDD	19	29	25	37	P	—	Positive supply for analog modules.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to VDD)

2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

PIC18C601/801 can be operated in one of four oscillator modes, programmable by configuration bits FOSC1:FOSC0 in CONFIG1H register:

1. LP Low Power Crystal
2. HS High Speed Crystal/Resonator
3. RC External Resistor/Capacitor
4. EC External Clock

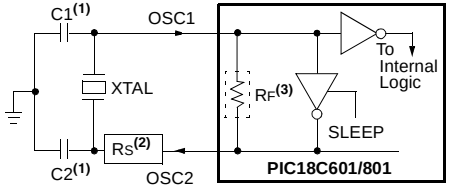
2.2 Crystal Oscillator/Ceramic Resonators

In LP or HS oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections. An external clock source may also be connected to the OSC1 pin, as shown in Figure 2-3 and Figure 2-4.

PIC18C601/801 oscillator design requires the use of a parallel cut crystal.

Note:	Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications.

FIGURE 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS OR LP OSC CONFIGURATION)



Note 1: See Table 2-1 and Table 2-2 for recommended values of C1 and C2.

2: A series resistor (RS) may be required for AT strip cut crystals.

3: RF varies with the crystal chosen.

TABLE 2-1: CERAMIC RESONATORS

Ranges Tested:			
Mode	**Freq.**	**OSC1**	**OSC2**
HS	8.0 MHz	10 - 68 pF	10 - 68 pF
	16.0 MHz	10 - 22 pF	10 - 22 pF
	20.0 MHz	TBD	TBD
	25.0 MHz	TBD	TBD
HS+PLL	4.0 MHz	TBD	TBD
These values are for design guidance only. See notes on this page.			
Resonators Used:			
4.0 MHz	Murata Erie CSA4.00MG		± 0.5%
8.0 MHz	Murata Erie CSA8.00MT		± 0.5%
16.0 MHz	Murata Erie CSA16.00MX		± 0.5%
All resonators used did not have built-in capacitors.			

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32.0 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	TBD	TBD
HS+PLL	4.0 MHz	15 pF	15 pF
These values are for design guidance only. See notes on this page.			
Crystals Used			
32.0 kHz	Epson C-001R32.768K-A		± 20 PPM
200 kHz	STD XTL 200.000kHz		± 20 PPM
1.0 MHz	ECS ECS-10-13-1		± 50 PPM
4.0 MHz	ECS ECS-40-20-1		± 50 PPM
8.0 MHz	EPSON CA-301 8.000M-C		± 30 PPM
20.0 MHz	EPSON CA-301 20.000M-C		± 30 PPM

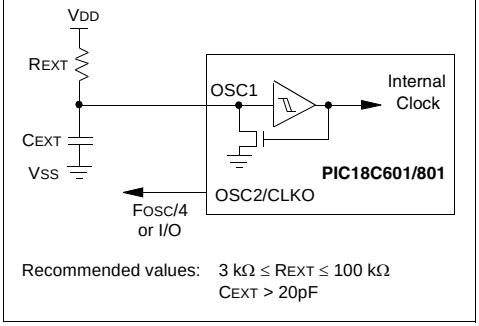
Note 1: Recommended values of C1 and C2 are identical to the ranges tested (Table 2-1).
2: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.
3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
4: Rs may be required in HS mode to avoid overdriving crystals with low drive level specification.

2.3 RC Oscillator

For timing insensitive applications, the "RC" oscillator mode offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 2-2 shows how the RC combination is connected.

In the RC oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic.

FIGURE 2-2: RC OSCILLATOR MODE

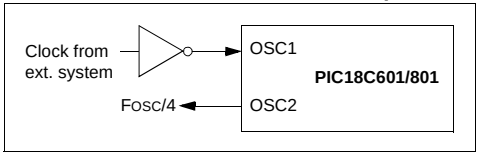


2.4 External Clock Input

The EC oscillator mode requires an external clock source to be connected to the OSC1 pin. The feedback device between OSC1 and OSC2 is turned off in these modes to save current. There is no oscillator start-up time required after a Power-on Reset or after a recovery from SLEEP mode.

In the EC oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC oscillator mode.

FIGURE 2-3: EXTERNAL CLOCK INPUT OPERATION (EC OSC CONFIGURATION)



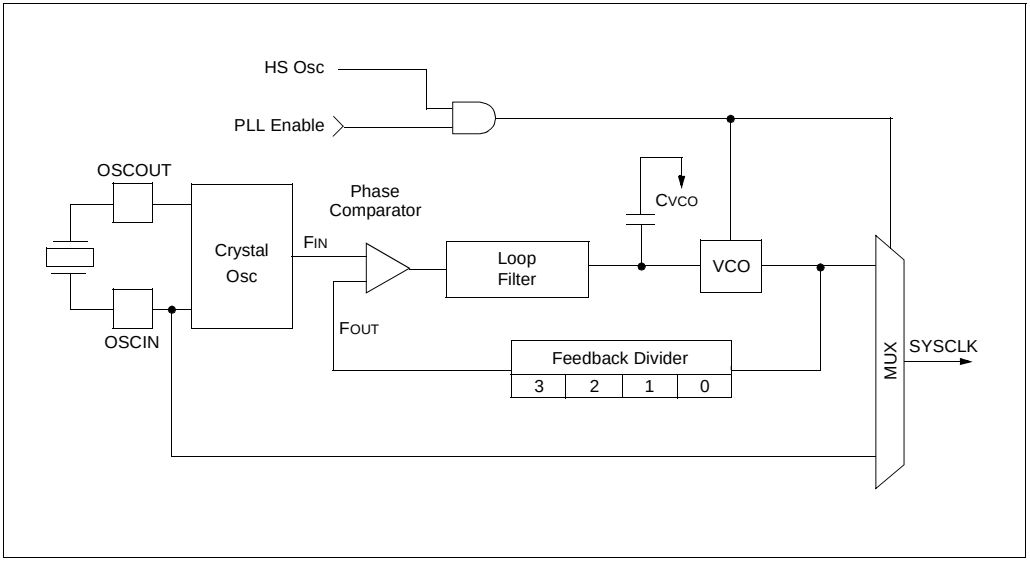
2.5 HS4 (PLL)

A Phase Lock Loop (PLL) circuit is provided as a software programmable option for users that want to multiply the frequency of the incoming crystal oscillator signal by 4. For an input clock frequency of 6 MHz, the internal clock frequency will be multiplied to 24 MHz. This is useful for customers who are concerned with EMI due to high frequency crystals.

The PLL is enabled by configuring HS oscillator mode and setting the PLLEN bit in the OSCON register. If HS oscillator mode is not selected, or PLLEN bit in OSCCON register is clear, the PLL is not enabled and the system clock will come directly from OSC1. HS oscillator mode is the default for PIC18C601/801. In all other modes, the PLLEN bit and the SCS1 bit are forced to '0'.

A PLL lock timer is used to ensure that the PLL has locked before device execution starts. The PLL lock timer has a time-out, referred to as TPLL.

FIGURE 2-4: PLL BLOCK DIAGRAM



2.6 Oscillator Switching Feature

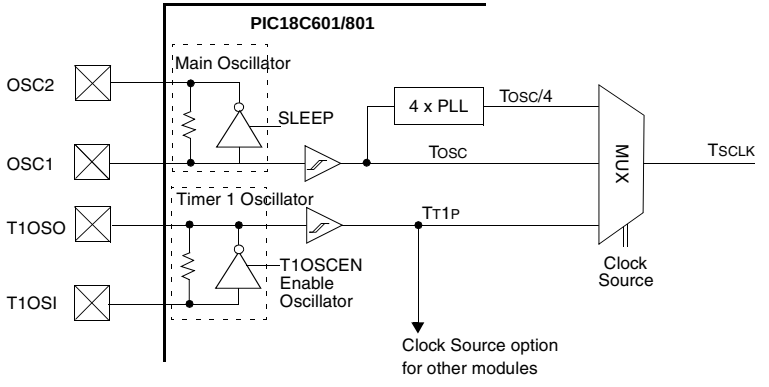
PIC18C601/801 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low frequency clock source. For PIC18C601/801 devices, this alternate clock source is the Timer1 oscillator. If a low frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a low power execution mode. Figure 2-5 shows a block diagram of the system clock sources.

2.6.1 SYSTEM CLOCK SWITCH BIT

The system clock source switching is performed under software control. The system clock switch bit, SCS0 (OSCCON register), controls the clock switching. When the SCS0 bit is '0', the system clock source comes from the main oscillator, selected by the FOSC2:FOSC0 configuration bits in CONFIG1H register. When the SCS0 bit is set, the system clock source will come from the Timer1 oscillator. The SCS0 bit is cleared on all forms of RESET.

Note:	The Timer1 oscillator must be enabled to switch the system clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 control register (T1CON). If the Timer1 oscillator is not enabled, any write to the SCS0 bit will be ignored (SCS0 bit forced cleared) and the main oscillator will continue to be the system clock source.

FIGURE 2-5: DEVICE CLOCK SOURCES



Note: I/O pins have diode protection to VDD and VSS.

REGISTER 2-1: OSCCON REGISTER

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	LOCK	PLLEN	SCS1	SCS0
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **LOCK:** Phase Lock Loop Lock Status bit
1 = Phase Lock Loop output is stable as system clock
0 = Phase Lock Loop output is not stable and cannot be used as system clock

bit 2 **PLLEN:** Phase Lock Loop Enable bit
1 = Enable Phase Lock Loop output as system clock
0 = Disable Phase Lock Loop

bit 1 **SCS1:** System Clock Switch bit 1
When PLLEN and LOCK bit are set:
1 = Use PLL output
0 = Use primary oscillator/clock input pin
When PLLEN bit or LOCK bit is cleared:
Bit is forced clear

bit 0 **SCS0:** System Clock Switch bit 0
When T1OSCEN bit is set:
1 = Switch to Timer1 oscillator/clock pin
0 = Use primary oscillator/clock input pin
When T1OSCEN is cleared:
Bit is forced clear

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.6.2 OSCILLATOR TRANSITIONS

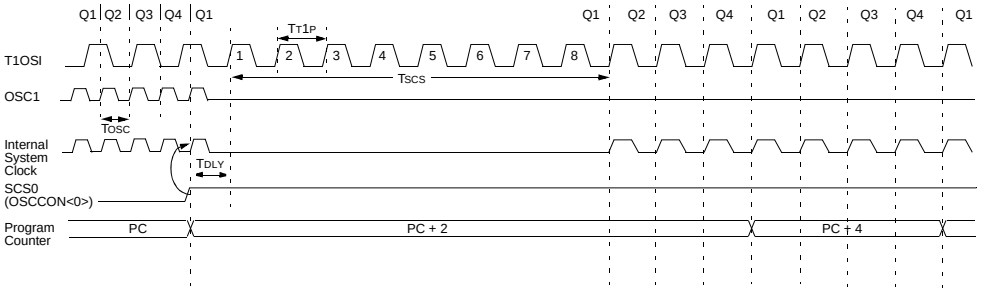
PIC18C601/801 devices contain circuitry to prevent "glitches" when switching between oscillator sources. Essentially, the circuitry waits for eight rising edges of the clock source that the processor is switching to. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

A timing diagram indicating the transition from the main oscillator to the Timer1 oscillator is shown in Figure 2-6. The Timer1 oscillator is assumed to be running all the time. After the SCS0 bit is set, the processor is frozen at the next occurring Q1 cycle. After eight synchronization cycles are counted from the Timer1 oscillator, operation resumes. No additional delays are required after the synchronization cycles.

The sequence of events that takes place when switching from the Timer1 oscillator to the main oscillator will depend on the mode of the main oscillator. In addition to eight clock cycles of the main oscillator, additional delays may take place.

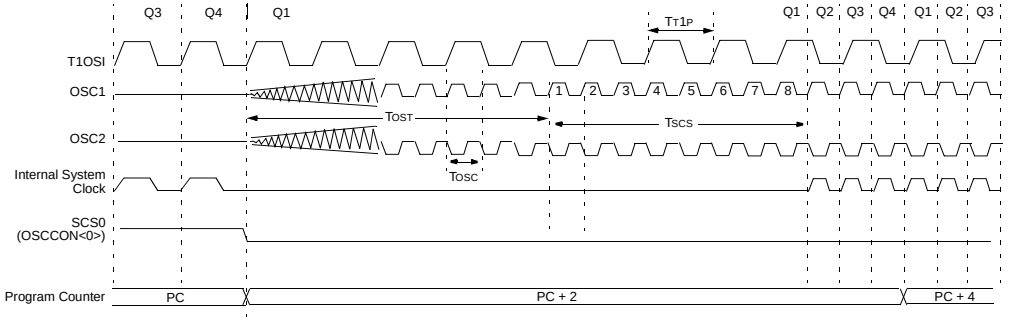
If the main oscillator is configured for an external crystal (HS, LP), the transition will take place after an oscillator start-up time (TOST) has occurred. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS and LP modes is shown in Figure 2-7.

FIGURE 2-6: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR



Note: Delay on internal system clock is eight oscillator cycles for synchronization.

FIGURE 2-7: TIMING DIAGRAM FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS, LP)



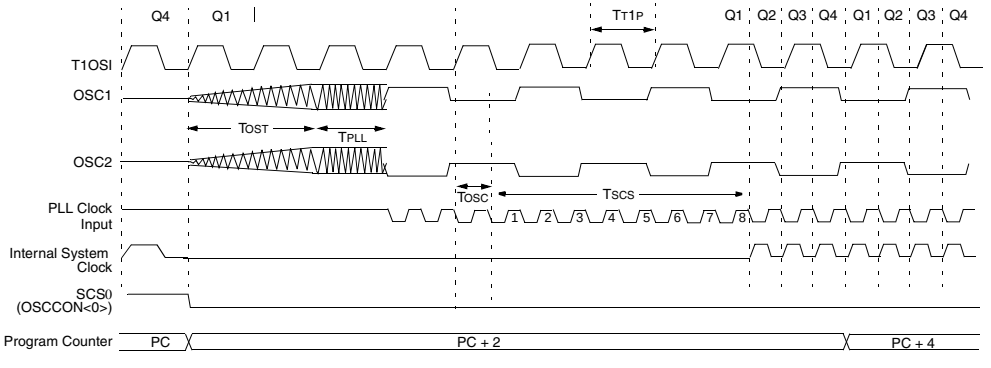
Note: TOST = 1024TOSC (drawing not to scale).

If the main oscillator is configured for HS4 (PLL) mode with SCS1 bit set to '1', an oscillator start-up time (TOST), plus an additional PLL time-out (TPLL) will occur. The PLL time-out is typically 2 ms and allows the PLL to lock to the main oscillator frequency. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS4 mode is shown in Figure 2-8.

If the main oscillator is configured for HS4 (PLL) mode, with SCS1 bit set to '0', only oscillator start-up time (TOST) will occur. Since SCS1 bit is set to '0', PLL output is not used, so the system oscillator will come from OSC1 directly and additional delay of TPLL is not required. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS4 mode is shown in Figure 2-9.

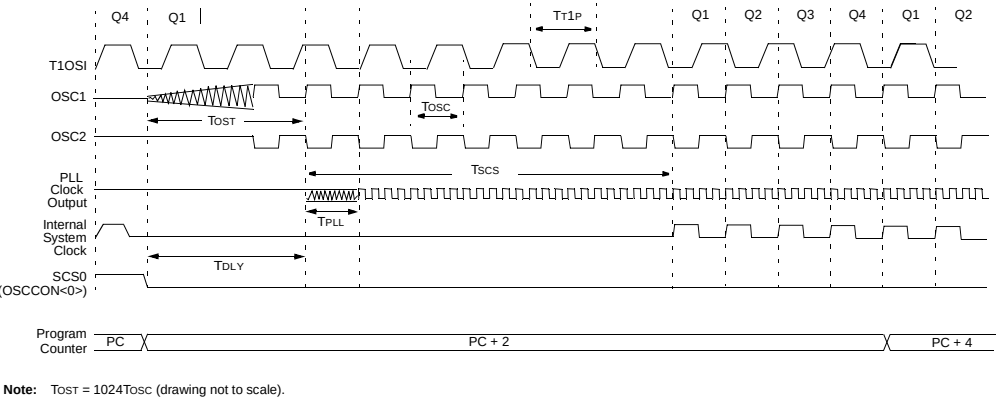
If the main oscillator is configured in the RC or EC modes, there is no oscillator start-up time-out. Operation will resume after eight cycles of the main oscillator have been counted. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for RC and EC modes is shown in Figure 2-10.

FIGURE 2-8: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS4 WITH SCS1 = 1)



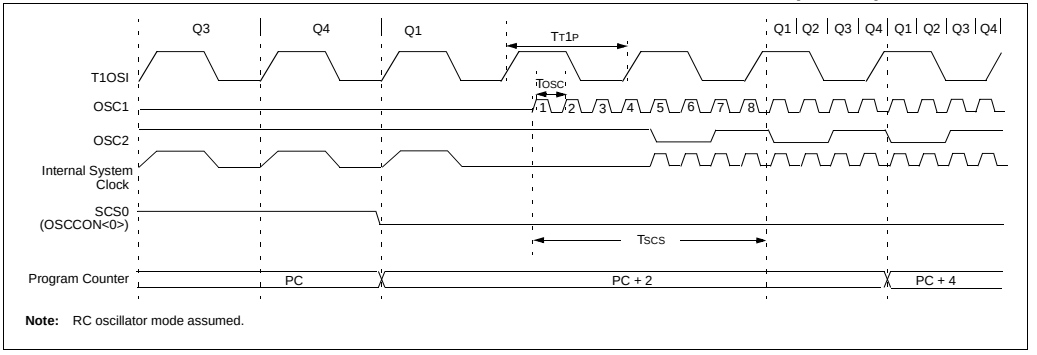
Note: TOST = 1024TOSC (drawing not to scale).

FIGURE 2-9: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS4 WITH SCS = 0)



Note: TOST = 1024TOSC (drawing not to scale).

FIGURE 2-10: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (RC, EC)



Note: RC oscillator mode assumed.

2.6.3 SCS0, SCS1 PRIORITY

If both SCS0 and SCS1 are set to '1' simultaneously, the SCS0 bit has priority over the SCS1 bit. This means that the low power option will take precedence over the PLL option. If both bits are cleared simultaneously, the system clock will come from OSC1, after a TOST time-out. If only the SCS0 bit is cleared, the system clock will come from the PLL output, following TOST and TPLL time.

TABLE 2-3: SCS0, SCS1 PRIORITY

SCS1	SCS0	Clock Source
0	0	Ext Oscillator OSC1
0	1	Timer1 Oscillator
1	0	HS + PLL
1	1	Timer1 Oscillator

2.7 Effects of SLEEP Mode on the On-Chip Oscillator

When the device executes a SLEEP instruction, the on-chip clocks and oscillator are turned off and the device is held at the beginning of an instruction cycle (Q1 state). With the oscillator off, the OSC1 and OSC2 signals will stop oscillating. Since all the transistor switching currents have been removed, SLEEP mode achieves the lowest current consumption of the device (only leakage currents). Enabling any on-chip feature that will operate during SLEEP, will increase the current consumed during SLEEP. The user can wake from SLEEP through external RESET, Watchdog Timer Reset, or through an interrupt.

2.8 Power-up Delays

Power-up delays are controlled by two timers, so that no external RESET circuitry is required for most applications. The delays ensure that the device is kept in RESET until the device power supply and clock are stable. For additional information on RESET operation, see Section 3.0 RESET.

The first timer is the Power-up Timer (PWRT), which optionally provides a fixed delay of TPWRT (parameter #33) on power-up only. The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable.

PIC18C601/801 devices provide a configuration bit, PWRTEN in CONFIG2L register, to enable or disable the Power-up Timer. By default, the Power-up Timer is enabled.

With the PLL enabled (HS4 oscillator mode), the time-out sequence following a Power-on Reset is different from other oscillator modes. The time-out sequence is as follows: the PWRT time-out is invoked after a POR time delay has expired, then, the Oscillator Start-up Timer (OST) is invoked. However, this is still not a sufficient amount of time to allow the PLL to lock at high frequencies. The PWRT timer is used to provide an additional time-out, called TPLL (parameter #7), to allow the PLL ample time to lock to the incoming clock frequency.

TABLE 2-4: OSC1 AND OSC2 PIN STATES IN SLEEP MODE

OSC Mode	OSC1 Pin	OSC2 Pin
RC	Floating, external resistor should pull high	At logic low
EC	Floating	At logic low
LP and HS	Feedback inverter disabled, at quiescent voltage level	Feedback inverter disabled, at quiescent voltage level

Note: See Table 3-1 in Section 3.0 RESET, for time-outs due to SLEEP and MCLR Reset.

3.0 RESET

PIC18C601/801 devices differentiate between various kinds of RESET:

a) Power-on Reset (POR)
b) $\overline{\text{MCLR}}$ Reset during normal operation
c) $\overline{\text{MCLR}}$ Reset during SLEEP
d) Watchdog Timer (WDT) Reset during normal operation
e) RESET Instruction
f) Stack Full Reset
g) Stack Underflow Reset

Most registers are unaffected by a RESET. Their status is unknown on POR and unchanged by all other RESETS. The other registers are forced to a "RESET" state on Power-on Reset, $\overline{\text{MCLR}}$, WDT Reset, $\overline{\text{MCLR}}$ Reset during SLEEP, and by the RESET instruction.

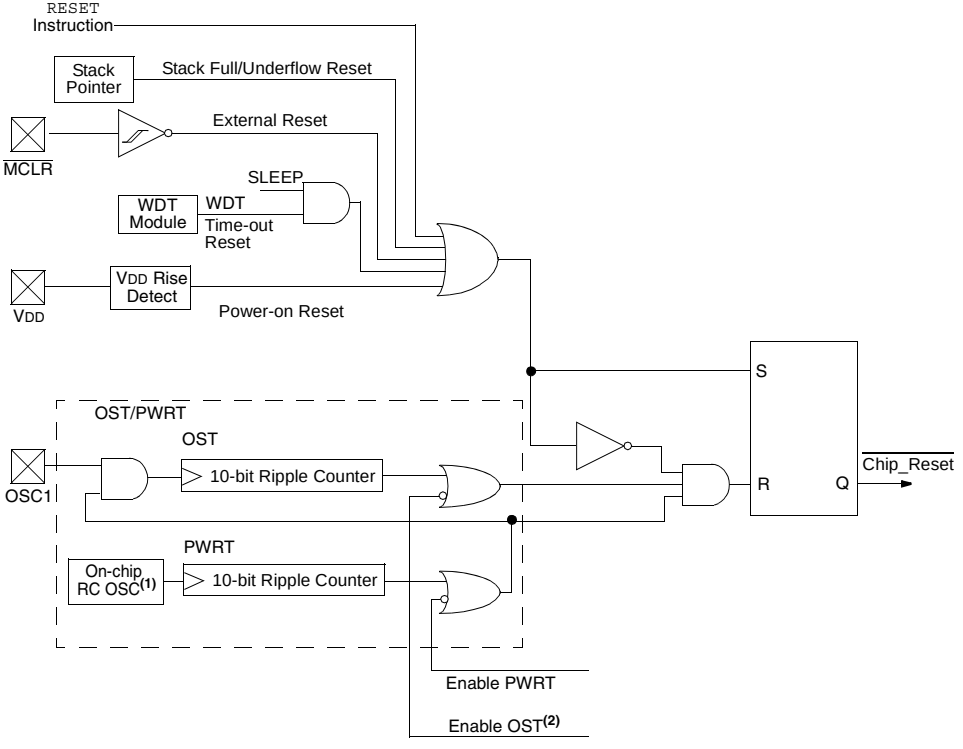
Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register, $\overline{\text{RI}}$, $\overline{\text{TO}}$, $\overline{\text{PD}}$ and $\overline{\text{POR}}$, are set or cleared differently in different RESET situations, as indicated in Table 3-2. These bits are used in software to determine the nature of the RESET. See Table 3-3 for a full description of the RESET states of all registers.

A simplified block diagram of the on-chip RESET circuit is shown in Figure 3-1.

PIC18C601/801 has a $\overline{\text{MCLR}}$ noise filter in the $\overline{\text{MCLR}}$ Reset path. The filter will detect and ignore small pulses.

A WDT Reset does not drive $\overline{\text{MCLR}}$ pin low.

FIGURE 3-1: SIMPLIFIED BLOCK DIAGRAM OF THE ON-CHIP RESET CIRCUIT



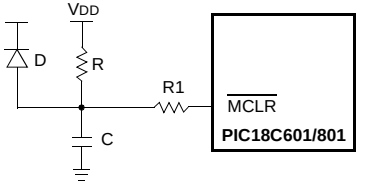
Note 1: This is a separate oscillator from the RC oscillator of the CLKI pin.
2: See Table 3-1 for time-out situations.

3.1 Power-on Reset (POR)

A Power-on Reset pulse is generated on-chip when a VDD rise is detected. To take advantage of the POR circuitry, connect the MCLR pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified (parameter D004). For a slow rise time, see Figure 3-2.

When the device starts normal operation (exits the RESET condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in RESET until the operating conditions are met. Power-on Reset may be used to meet the voltage start-up condition.

FIGURE 3-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 = 100Ω to 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown due to Electrostatic Discharge (ESD), or Electrical Overstress (EOS).

3.2 Power-up Timer (PWRT)

The Power-up Timer provides a fixed nominal time-out (parameter #33), only on power-up from the POR. The Power-up Timer operates on an internal RC oscillator. The chip is kept in RESET as long as the PWRT is active. The PWRT time delay allows VDD to rise to an acceptable level. PIC18C601/801 devices are available with PWRT enabled or disabled.

The power-up time delay will vary from chip to chip, due to VDD, temperature and process variation. See DC parameter #33 for details.

3.3 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over (parameter #32). This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for LP, HS and HS4 modes and only on Power-on Reset or wake-up from SLEEP.

3.4 PLL Lock Time-out

With the PLL enabled, the time-out sequence following a Power-on Reset is different from other oscillator modes. A portion of the Power-up Timer is used to provide a fixed time-out that is sufficient for the PLL to lock to the main oscillator frequency. This PLL lock time-out (TPLL) is typically 1 ms and follows the oscillator start-up time-out (OST).

3.5 Time-out Sequence

On power-up, the time-out sequence is as follows: First, PWRT time-out is invoked after the POR time delay has expired; then, OST is activated. The total time-out will vary based on oscillator configuration and the status of the PWRT. For example, in RC mode with the PWRT disabled, there will be no time-out at all. Figure 3-3, Figure 3-4, Figure 3-5, Figure 3-6 and Figure 3-7 depict time-out sequences on power-up.

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, the time-outs will expire. Bringing MCLR high will begin execution immediately (Figure 3-5). This is useful for testing purposes or to synchronize more than one PIC18C601/801 device operating in parallel.

Table 3-2 shows the RESET conditions for some Special Function Registers, while Table 3-3 shows the RESET conditions for all registers.

TABLE 3-1: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up[2]		Wake-up from SLEEP or Oscillator Switch[1]
	$\overline{PWRTEN}$ = 0	$\overline{PWRTEN}$ = 1	
HS with PLL enabled[1]	72 ms + 1024TOSC	1024TOSC	1024TOSC + 1 ms
HS, LP	72 ms + 1024TOSC	1024TOSC	1024TOSC
EC	72 ms	—	—
External RC	72 ms	—	—

Note 1: 1 ms is the nominal time required for the 4X PLL to lock. Maximum time is 2 ms.
2: 72 ms is the nominal Power-up Timer delay.

REGISTER 3-1: RCON REGISTER BITS AND POSITIONS

R/W-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	U-0
IPEN	r	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	r
bit 7							bit 0

TABLE 3-2: STATUS BITS, THEIR SIGNIFICANCE, AND THE INITIALIZATION CONDITION FOR RCON REGISTER

Condition	Program Counter	RCON Register	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	STKFUL	STKUNF
Power-on Reset	00000h	0r-1 110r	1	1	1	0	u	u
$\overline{MCLR}$ Reset during normal operation	00000h	0r-u uuur	u	u	u	u	u	u
Software Reset during normal operation	00000h	0r-0 uuur	0	u	u	u	u	u
Stack Full Reset during normal operation	00000h	0r-u uu1r	u	u	u	1	u	1
Stack Underflow Reset during normal operation	00000h	0r-u uu1r	u	u	u	1	1	u
$\overline{MCLR}$ Reset during SLEEP	00000h	0r-u 10ur	u	1	0	u	u	u
WDT Reset	00000h	0r-u 01ur	u	0	1	u	u	u
WDT Wake-up	PC + 2	ur-u 00ur	u	0	0	u	u	u
Interrupt wake-up from SLEEP	PC + 2[1]	ur-u 00ur	u	0	0	u	u	u

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', r = reserved, maintain '0'
Note 1: When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (000008h or 000018h).

FIGURE 3-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ TIED TO V_{DD})

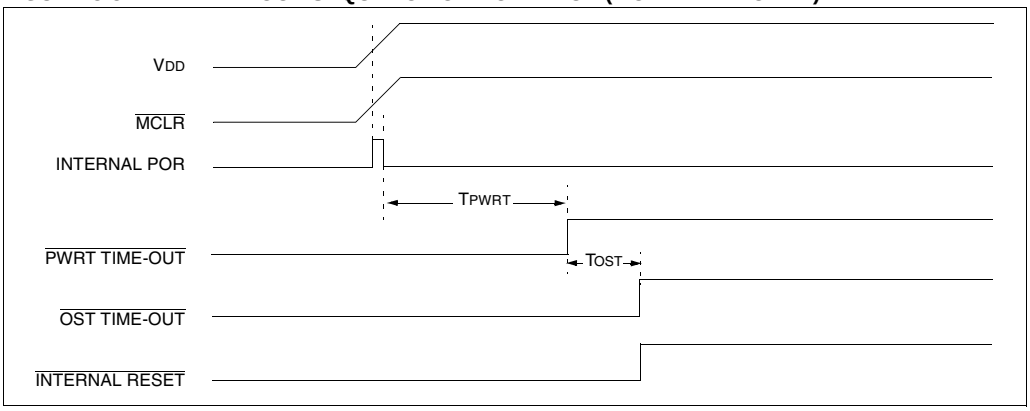


FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO V_{DD}): CASE 1

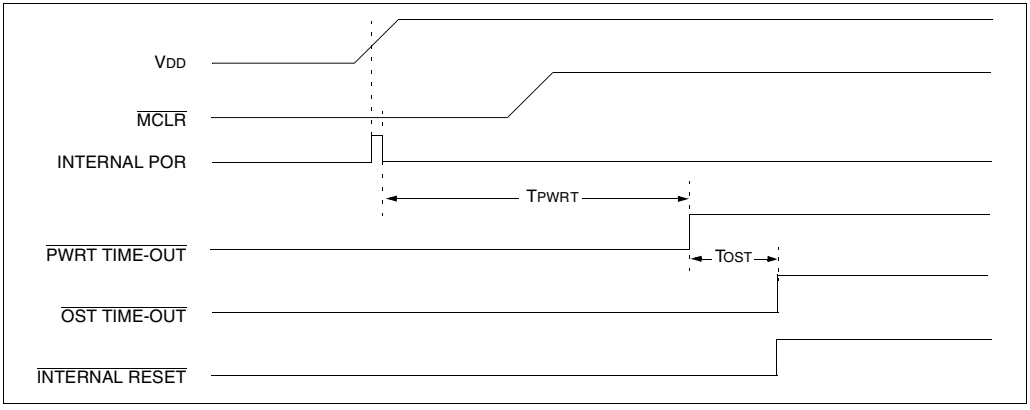


FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO V_{DD}): CASE 2

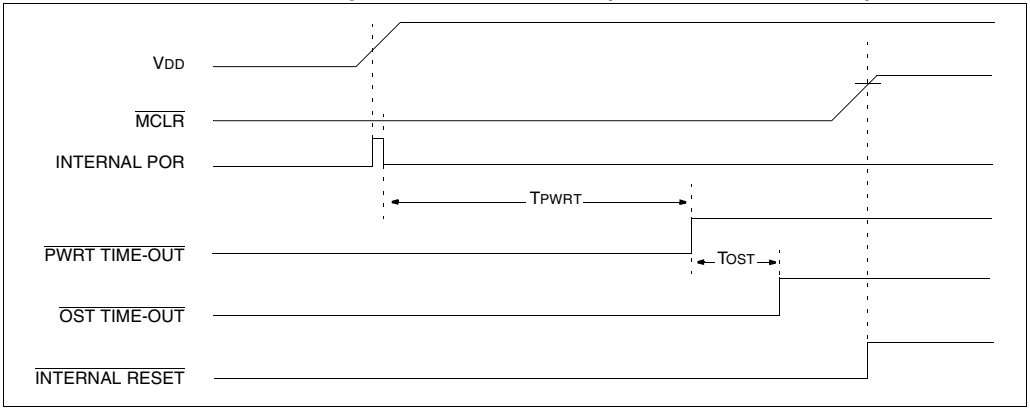


FIGURE 3-6: SLOW RISE TIME ($\overline{MCLR}$ TIED TO V_{DD})

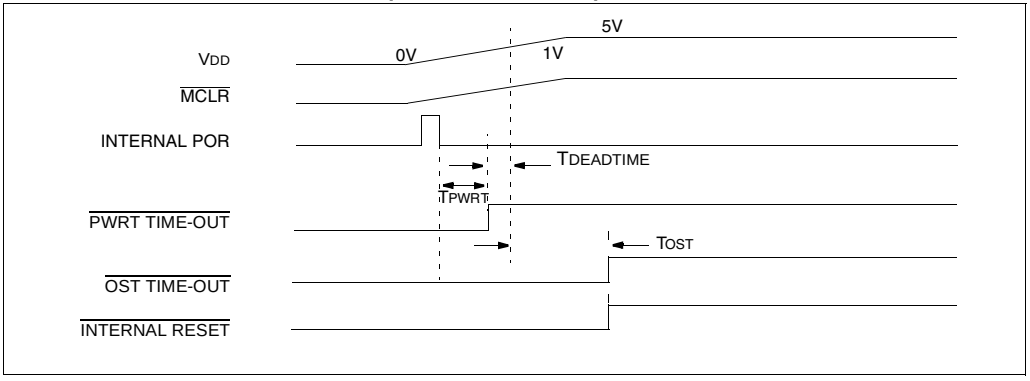
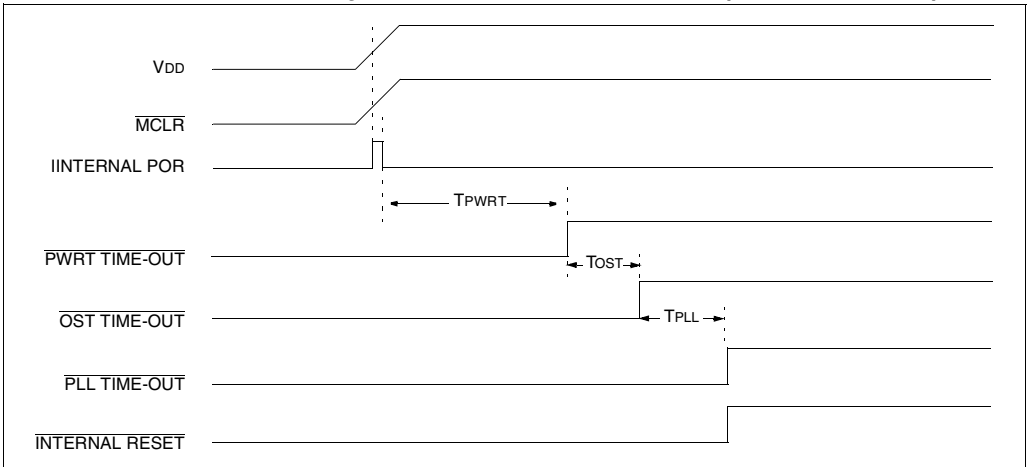


FIGURE 3-7: TIME-OUT SEQUENCE ON POR W/ PLL ENABLED ($\overline{MCLR}$ TIED TO V_{DD})



TOST = 1024 clock cycles.

TPLL ≈ 2 ms max. First three stages of the PWRT timer.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices		Power-on Reset	$\overline{\text{MCLR}}$ Reset WDT Reset Reset Instruction Stack Over/Underflow Reset	Wake-up via WDT or Interrupt
TOSU	601	801	---0 0000	---0 0000	---u uuuu(3)
TOSH	601	801	0000 0000	0000 0000	uuuu uuuu(3)
TOSL	601	801	0000 0000	0000 0000	uuuu uuuu(3)
STKPTR	601	801	00-0 0000	00-0 0000	uu-u uuuu(3)
PCLATU	601	801	---0 0000	---0 0000	---u uuuu
PCLATH	601	801	0000 0000	0000 0000	uuuu uuuu
PCL	601	801	0000 0000	0000 0000	PC + 2(2)
TBLPTRU	601	801	--00 0000	--00 0000	--uu uuuu
TBLPTRH	601	801	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	601	801	0000 0000	0000 0000	uuuu uuuu
TABLAT	601	801	0000 0000	0000 0000	uuuu uuuu
PRODH	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	601	801	0000 000x	0000 000u	uuuu uuuu(1)
INTCON2	601	801	1111 -1-1	1111 -1-1	uuuu -u-u(1)
INTCON3	601	801	11-0 0-00	11-0 0-00	uu-u u-uu(1)
INDF0	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
POSTINC0	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
POSTDEC0	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
PREINC0	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
PLUSW0	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
FSR0H	601	801	---- 0000	---- 0000	---- uuuu
FSR0L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
POSTINC1	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
POSTDEC1	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
PREINC1	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
PLUSW1	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
FSR1H	601	801	---- 0000	---- 0000	---- uuuu
FSR1L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	601	801	---- 0000	---- 0000	---- uuuu
INDF2	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition, r = reserved, maintain '0'

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (00008h or 00018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH, and TOSL are updated with the current value of the PC. The SKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: This is not a physical register. It is an indirect pointer that addresses another register. The contents returned is the value contained in the addressed register.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset	$\overline{\text{MCLR}}$ Reset WDT Reset Reset Instruction Stack Over/Underflow Reset	Wake-up via WDT or Interrupt
POSTINC2	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
POSTDEC2	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
PREINC2	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
PLUSW2	601	801	**(Note 5)**	**(Note 5)**	**(Note 5)**
FSR2H	601	801	---- 0000	---- 0000	---- uuuu
FSR2L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	601	801	---x xxxx	---u uuuu	---u uuuu
TMR0H	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	601	801	1111 1111	1111 1111	uuuu uuuu
OSCCON	601	801	--00 0-00	--uu u-u0	--uu u-uu
LVDCON	601	801	--00 0101	--00 0101	--uu uuuu
WDTCON	601	801	---- 1111	---- uuuu	---- uuuu
RCON(4)	601	801	0r-1 11qr	0r-1 qqur	ur-u qqur
TMR1H	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	601	801	0-00 0000	u-uu uuuu	u-uu uuuu
TMR2	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR2	601	801	1111 1111	1111 1111	1111 1111
T2CON	601	801	-000 0000	-000 0000	-uuu uuuu
SSPBUF	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	601	801	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	601	801	0000 0000	0000 0000	uuuu uuuu
SSPCON1	601	801	0000 0000	0000 0000	uuuu uuuu
SSPCON2	601	801	0000 0000	0000 0000	uuuu uuuu
ADRESH	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	601	801	--00 0000	--00 0000	--uu uuuu
ADCON1	601	801	-000 0000	-000 0000	-uuu uuuu
ADCON2	601	801	0--- -000	0--- -000	u--- -uuu
CCPR1H	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	601	801	--00 0000	--00 0000	--uu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition, r = reserved, maintain '0'

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (00008h or 00018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH, and TOSL are updated with the current value of the PC. The SKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: This is not a physical register. It is an indirect pointer that addresses another register. The contents returned is the value contained in the addressed register.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset	$\overline{\text{MCLR}}$ Reset WDT Reset Reset Instruction Stack Over/Underflow Reset	Wake-up via WDT or Interrupt
CCPR2H	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	601	801	--00 0000	--00 0000	--uu uuuu
TMR3H	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	601	801	0000 0000	uuuu uuuu	uuuu uuuu
SPBRG	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
RCREG	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXREG	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXSTA	601	801	0000 -01x	0000 -01u	uuuu -uuu
RCSTA	601	801	0000 000x	0000 000u	uuuu uuuu
IPR2	601	801	-1-- 1111	-1-- 1111	-u-- uuuu
PIR2	601	801	-1-- 0000	-1-- 0000	-u-- uuuu(1)
PIE2	601	801	-1-- 0000	-1-- 0000	-u-- uuuu
IPR1	601	801	1111 1111	1111 1111	uuuu uuuu
	601	801	-111 1111	-111 1111	-uuu uuuu
PIR1	601	801	0000 0000	0000 0000	uuuu uuuu(1)
	601	801	-000 0000	-000 0000	-uuu uuuu(1)
PIE1	601	801	0000 0000	0000 0000	uuuu uuuu
	601	801	-000 0000	-000 0000	-uuu uuuu
MEMCON	601	801	0000 --00	0000 --00	uuuu --uu
TRISJ	601	801	1111 1111	1111 1111	uuuu uuuu
TRISH	601	801	1111 1111	1111 1111	uuuu uuuu
TRISG	601	801	---1 1111	---1 1111	---u uuuu
TRISF	601	801	1111 1111	1111 1111	uuuu uuuu
TRISE	601	801	1111 1111	1111 1111	uuuu uuuu
TRISD	601	801	1111 1111	1111 1111	uuuu uuuu
TRISC	601	801	1111 1111	1111 1111	uuuu uuuu
TRISB	601	801	1111 1111	1111 1111	uuuu uuuu
TRISA	601	801	--11 1111	--11 1111	--uu uuuu
LATG	601	801	---x xxxx	---u uuuu	---u uuuu
LATF	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATE	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition, r = reserved, maintain '0'

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (00008h or 00018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH, and TOSL are updated with the current value of the PC. The SKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: This is not a physical register. It is an indirect pointer that addresses another register. The contents returned is the value contained in the addressed register.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset	$\overline{\text{MCLR}}$ Reset WDT Reset Reset Instruction Stack Over/Underflow Reset	Wake-up via WDT or Interrupt
LATD	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA	601	801	--xx xxxx	--uu uuuu	--uu uuuu
PORTJ	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTH	601	801	0000 xxxx	0000 uuuu	uuuu uuuu
PORTG	601	801	---x xxxx	---u uuuu	---u uuuu
PORTF	601	801	xxxx x000	uuuu u000	uuuu uuuu
PORTE	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTD	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	601	801	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	601	801	--0x 0000	--0u 0000	--uu uuuu
CSEL2	601	801	1111 1111	uuuu uuuu	uuuu uuuu
CSELIO	601	801	1111 1111	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition, r = reserved, maintain '0'

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (00008h or 00018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH, and TOSL are updated with the current value of the PC. The SKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: This is not a physical register. It is an indirect pointer that addresses another register. The contents returned is the value contained in the addressed register.

NOTES:

4.0 MEMORY ORGANIZATION

There are two memory blocks in PIC18C601/801 devices. These memory blocks are:

- Program Memory
- Data Memory

Each block has its own bus so that concurrent access can occur.

4.1 Program Memory Organization

PIC18C601/801 devices have a 21-bit program counter that is capable of addressing up to 2 Mbyte of external program memory space. The PIC18C601 has an external program memory address space of 256 Kbytes. Any program fetch or TBLRD from a program location greater than 256K will return all NOPs. The PIC18C801 has an external program memory address space of 2Mbytes. Refer to Section 5.0 (“External Memory Interface”) for additional details.

The RESET vector address is mapped to 000000h and the interrupt vector addresses are at 000008h and 000018h. PIC18C601/801 devices have a 31-level stack to store the program counter values during subroutine calls and interrupts. Figure 4-1 shows the program memory map and stack for PIC18C601. Figure 4-2 shows the program memory map and stack for the PIC18C801.

4.1.1 “BOOT RAM” PROGRAM MEMORY

PIC18C601/801 devices have a provision for configuring the last 512 bytes of general purpose user RAM as program memory, called “Boot RAM”. This is achieved by configuring the PGRM bit in the MEMCON register to ‘1’. (Refer to Section 5.0, “External Memory Interface” for more information.) When the PGRM bit is ‘1’, the RAM located in data memory locations 400h through 5FFh (bank 4 through 5) is mapped to program memory locations 1FFE00h to 1FFFFFh.

When configured as program memory, the Boot RAM is to be used as a temporary “boot loader” for programming purposes. It can only be used for program execution. A read from locations 400h to 5FFh in data memory returns all ‘0’s. Any attempt to write this RAM as data memory when PGRM = 1, does not modify any of these locations. TBLWT instructions to these locations will cause writes to occur on the external memory bus. The boot RAM program memory cannot be modified using TBLWT instruction. TBLRD instructions from boot RAM will read memory located on the external memory bus, not from the on-board RAM. Constants that are stored in boot RAM are retrieved using the RETLW instruction.

The default RESET state (power-up) for the PGRM bit is ‘0’, which configures 1.5K of data RAM and all program memory as external. The PGRM bit can be set and cleared in the software.

When execution takes place from “Boot RAM”, the external system bus and all of its control signals will be deactivated. If execution takes place from outside of “Boot RAM”, the external system bus and all of its control signals are activated again.

Figure 4-3 and Figure 4-4 show the program memory map and stack for PIC18C601 and PIC18C801, when the PGRM bit is set.

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR PIC18C601 (PGRM = 0)

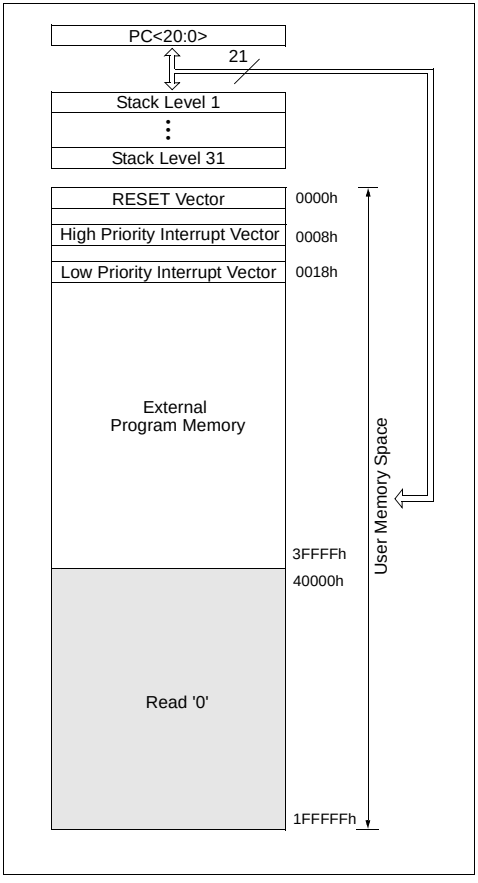


FIGURE 4-2: PROGRAM MEMORY MAP AND STACK FOR PIC18C801 (PGRM = 0)

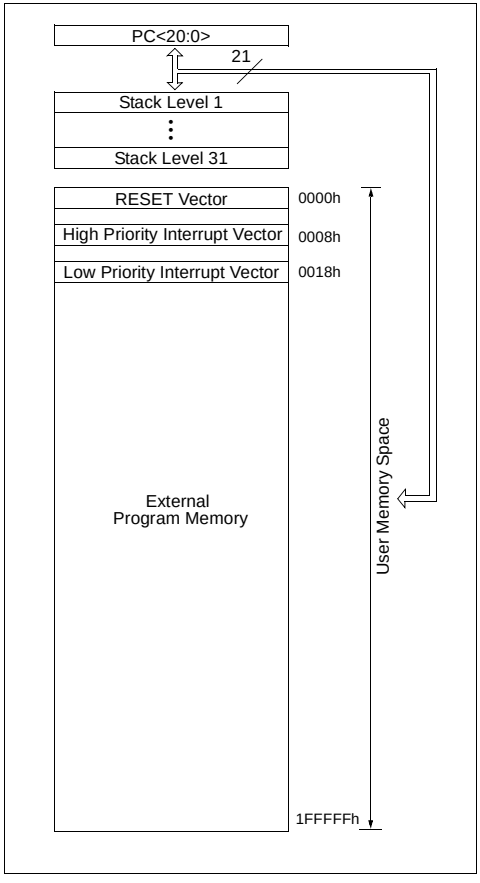


FIGURE 4-3: PROGRAM MEMORY MAP AND STACK FOR PIC18C601 (PGRM = 1)

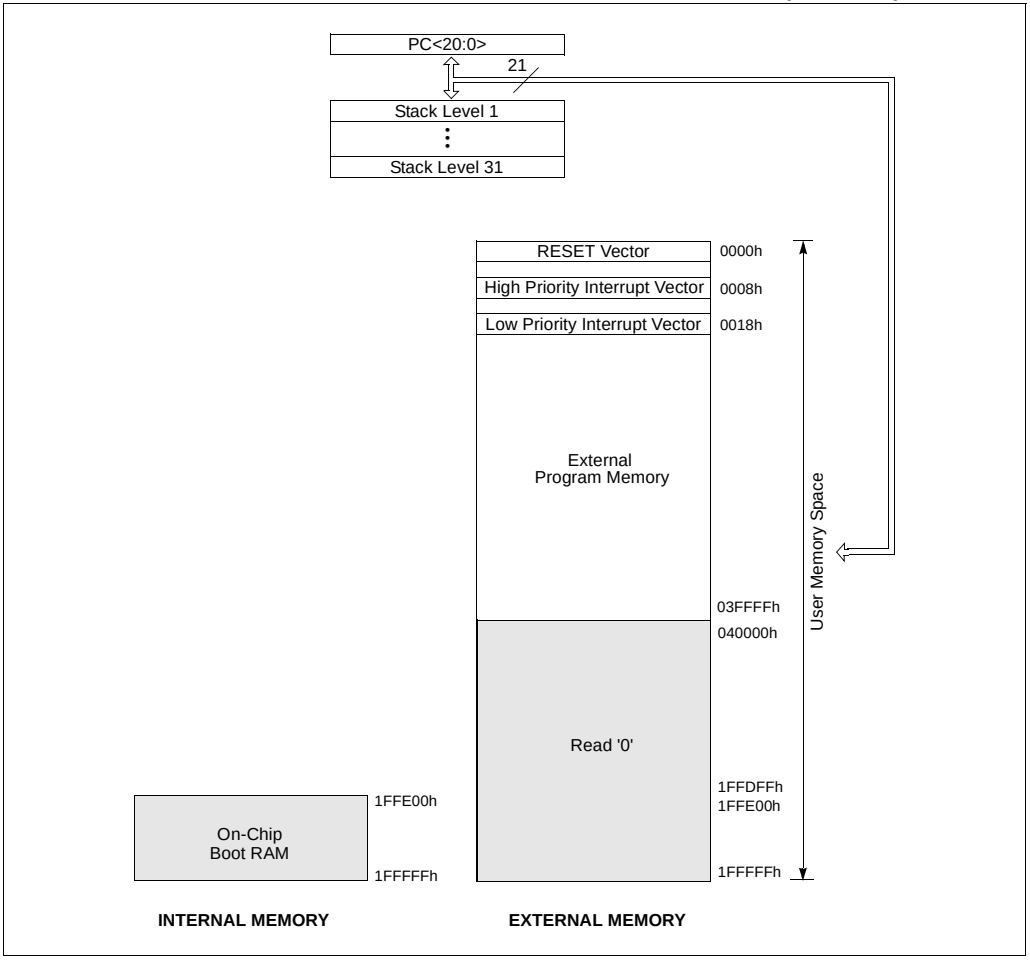
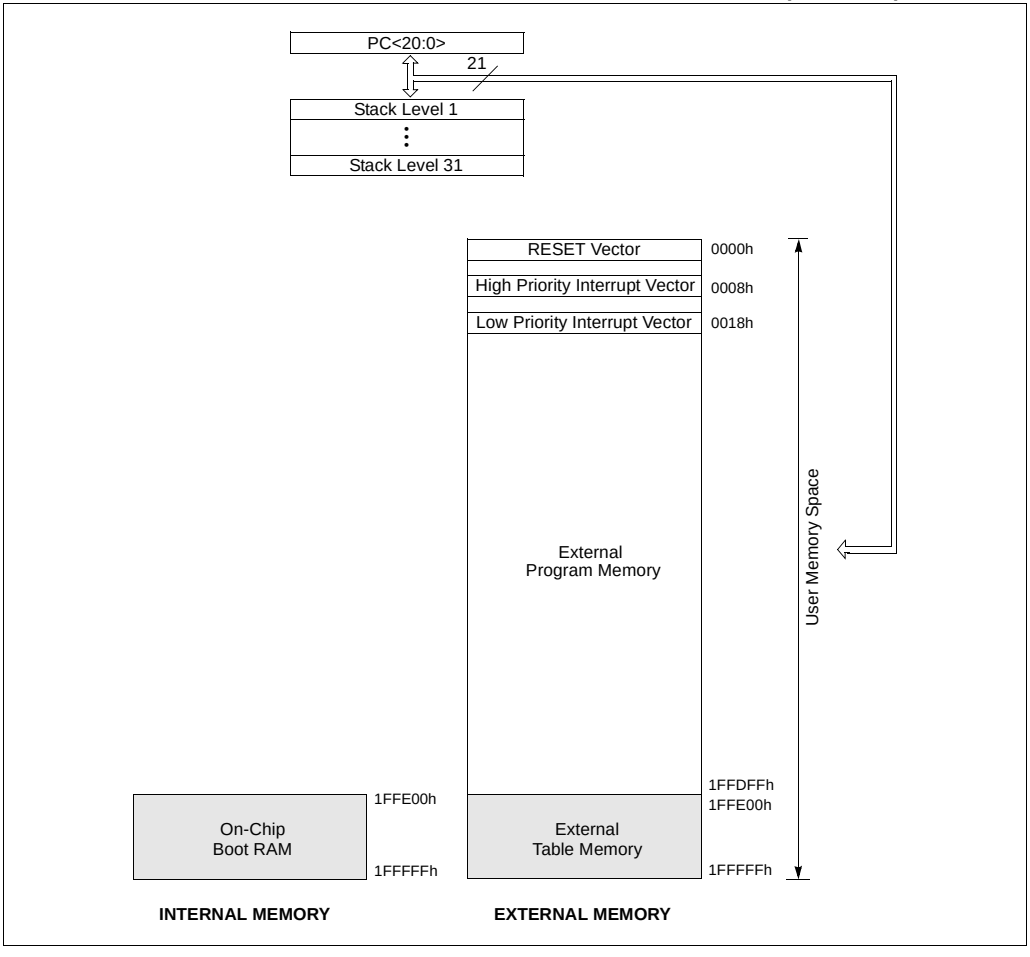


FIGURE 4-4: PROGRAM MEMORY MAP AND STACK FOR PIC18C801 (PGRM = 1)



4.1.2 BOOT LOADER

When configured as Program Memory, Boot RAM can be used as a temporary "Boot Loader" for programming purposes. If an external memory device is used as program memory, any updates performed by the user program will have to be performed in the "Boot RAM", because the user program cannot program and fetch from external memory, simultaneously.

A typical boot loader execution and external memory programming sequence would be as follows:

- The boot loader program is transferred from the external program memory to the last 2 banks of data RAM by TBLRD and MOVWF instructions.
- Once the "boot loader" program is loaded into internal memory and verified, open combination lock and set PGRM bit to configure the data RAM into program RAM.
- Jump to beginning of Boot code in Boot RAM. Program execution begins in Boot RAM to begin programming the external memory. System bus changes to an inactive state.
- Boot loader program performs the necessary external TBLWT and TBLWRD instructions to perform programming functions.
- When the boot loader program is finished programming external memory, jump to known valid external program memory location and clear PGRM bit in MEMCON register to set Boot RAM as data memory, or reset the part.

4.2 Return Address Stack

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC (Program Counter) is pushed onto the stack when a PUSH, CALL or RCALL instruction is executed, or an interrupt is acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the return instructions.

The stack operates as a 31-word by 21-bit stack memory and a five-bit stack pointer, with the stack pointer initialized to 00000b after all RESETS. There is no RAM associated with stack pointer 00000b. This is only a RESET value. During a CALL type instruction, causing a push onto the stack, the stack pointer is first incremented and the RAM location pointed to by the stack pointer is written with the contents of the PC. During a RETURN type instruction, causing a pop from the stack, the contents of the RAM location indicated by the STKPTR is transferred to the PC and then the stack pointer is decremented.

The stack space is not part of either program or data space. The stack pointer is readable and writable, and the data on the top of the stack is readable and writable through SFR registers. Status bits STKOVF and STKUNF in STKPTR register, indicate whether stack over/underflow has occurred or not.

4.2.1 TOP-OF-STACK ACCESS

The top of the stack is readable and writable. Three register locations, TOSU, TOSH and TOSL, allow access to the contents of the stack location indicated by the STKPTR register. This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU, TOSH and TOSL registers. These values can be placed on a user defined software stack. At return time, the software can replace the TOSU, TOSH and TOSL and do a return.

The user should disable the global interrupt enable bits during this time to prevent inadvertent stack operations.

4.2.2 RETURN STACK POINTER (STKPTR)

The STKPTR register contains the stack pointer value, the STKFUL (stack full) status bit, and the STKUNF (stack underflow) status bits. Register 4-1 shows the STKPTR register. The value of the stack pointer can be 0 through 31. The stack pointer increments when values are pushed onto the stack and decrements when values are popped off the stack. At RESET, the stack pointer value will be 0. The user may read and write the stack pointer value. This feature can be used by a Real Time Operating System for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit can only be cleared in software or by a POR. Any subsequent push operation that causes stack overflow will be ignored.

The action that takes place when the stack becomes full, depends on the state of STVREN (stack overflow RESET enable) configuration bit in CONFIG4L register. Refer to Section 4.2.4 for more information. If STVREN is set (default), stack over/underflow will set the STKFUL bit, and reset the device. The STKFUL bit will remain set and the stack pointer will be set to 0.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the stack pointer will increment to 31. All subsequent push attempts will be ignored and STKPTR remains at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit, while the stack pointer remains at 0. The STKUNF bit will remain set until cleared in software, or a POR occurs.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the RESET vector, where the stack conditions can be verified and appropriate actions can be taken.

REGISTER 4-1: STKPTR - STACK POINTER REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL	STKUNF	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

bit 7 **STKFUL**: Stack Full Flag bit
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF**: Stack Underflow Flag bit
1 = Stack underflow occurred
0 = Stack underflow did not occur

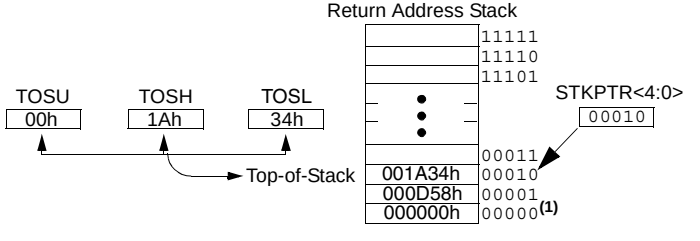
bit 5 **Unimplemented**: Read as '0'

bit 4-0 **SP4:SP0**: Stack Pointer Location bits

Note: Bit 7 and bit 6 can only be cleared in user software, or by a POR.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	C = Clearable bit

FIGURE 4-5: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



Note 1: No RAM is associated with this address; always maintained '0's.

4.2.3 PUSH AND POP INSTRUCTIONS

Since the Top-of-Stack (TOS) is readable and writable, the ability to push values onto the stack and pop values off the stack, without disturbing normal program execution, is a desirable option. To push the current PC value onto the stack, a `PUSH` instruction can be executed. This will increment the stack pointer and load the current PC value onto the stack. TOSU, TOSH and TOSL can then be modified to place a return address on the stack.

The `POP` instruction discards the current TOS by decrementing the stack pointer. The previous value pushed onto the stack then becomes the TOS value.

4.2.4 STACK FULL/UNDERFLOW RESETS

These RESETS are enabled/disabled by programming the STVREN configuration bit in CONFIG4L register.

When the STVREN bit is disabled, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a RESET. When the STVREN bit is enabled, a full or underflow will set the appropriate STKFUL or STKUNF bit and then cause a RESET. The STKFUL or STKUNF bits are only cleared by the user software or a POR.

4.3 Fast Register Stack

A “fast return” option is available for interrupts and calls. A fast register stack is provided for the STATUS, WREG and BSR registers, and is only one layer in depth. The stack is not readable or writable and is loaded with the current value of the corresponding register when the processor vectors for an interrupt. The values in the fast register stack are then loaded back into the working registers, if the `fast return` instruction is used to return from the interrupt.

A low or high priority interrupt source will push values into the stack registers. If both low and high priority interrupts are enabled, the stack registers cannot be used reliably for low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten.

If high priority interrupts are not disabled during low priority interrupts, users must save the key registers in software during a low priority interrupt.

If no interrupts are used, the fast register stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a `fast call` instruction must be executed.

Example 4-1 shows a source code example that uses the fast register stack.

EXAMPLE 4-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
       •
       •

SUB1   •
       •
       •
    RETURN FAST    ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

4.4 PCL, PCLATH and PCLATU

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<15:8> bits and is not directly readable or writable. Updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits and is not directly readable or writable. Updates to the PCU register may be performed through the PCLATU register.

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the LSb of the PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

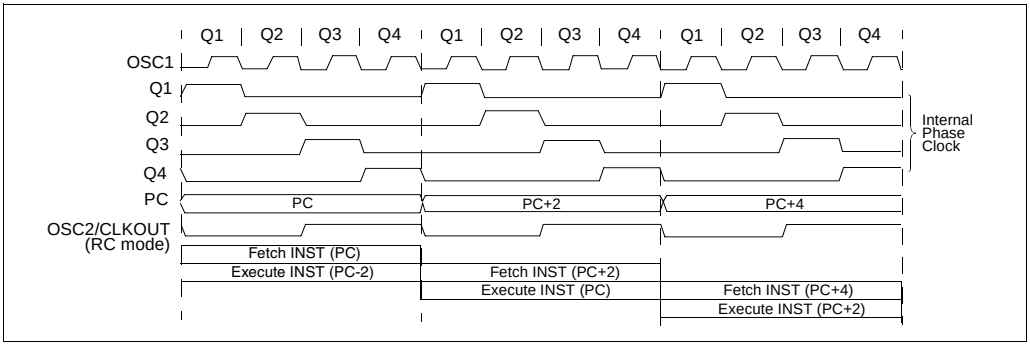
The `CALL`, `RCALL`, `GOTO` and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

The contents of PCLATH and PCLATU will be transferred to the program counter by an operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (See Section 4.8.1).

4.5 Clocking Scheme/Instruction Cycle

The clock input (from OSC1 or PLL output) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 4-6.

FIGURE 4-6: CLOCK/INSTRUCTION CYCLE



4.6 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined, such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), two cycles are required to complete the instruction (Example 4-2).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the “Instruction Register” (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

4.7 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = ’0’). Figure 4-1 shows an example of how instruction words are stored in the program memory. To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read ’0’ (see Section 4.4).

The CALL and GOTO instructions have an absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 4-1 shows how the instruction “GOTO 0x06” is encoded in the program memory. Program branch instructions that encode a relative address offset operate in the same manner. The offset value stored in a branch instruction represents the number of single word instructions by which the PC will be offset. Section 20.0 provides further details of the instruction set.

EXAMPLE 4-2: INSTRUCTION PIPELINE FLOW

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. BRA SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, BIT3 (Forced NOP)				Fetch 4	Flush	
5. Instruction @ address SUB_1					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles, since the fetch instruction is “flushed” from the pipeline, while the new instruction is being fetched and then executed.

TABLE 4-1: INSTRUCTIONS IN PROGRAM MEMORY

Instruction	Opcode	Memory	Address
—	—	—	000007h
MOVLW 055h	0E55h	55h	000008h
		0Eh	000009h
GOTO 000006h	EF03h, F000h	03h	00000Ah
		EFh	00000Bh
		00h	00000Ch
		F0h	00000Dh
MOVFF 123h, 456h	C123h, F456h	23h	00000Eh
		C1h	00000Fh
		56h	000010h
		F4h	000011h
—	—	—	000012h

4.7.1 TWO-WORD INSTRUCTIONS

PIC18C601/801 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the four MSB's set to 1's and is a special kind of NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC and skips one instruction. A program example that demonstrates this concept is shown in Example 4-3. Refer to Section 19.0 for further details of the instruction set.

4.8 Lookup Tables

Lookup tables are implemented two ways:

- Computed GOTO
- Table Reads

4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A lookup table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG is loaded with an offset into the table, before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW 0xnn instructions that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

Warning:	The LSb of the PCL is fixed to a value of '0'. Hence, computed GOTO to an odd address is not possible.

4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Lookup table data may be stored as 2 bytes per program word by using table reads and writes. The table pointer (TBLPTR) specifies the byte address and the table latch (TABLAT) contains the data that is read from, or written to, program memory. Data is transferred to/from program memory one byte at a time.

A description of the Table Read/Table Write operation is shown in Section 6.0.

Note:	If execution is taking place from Boot RAM Program Memory, RETLW instructions must be used to read lookup values from the Boot RAM itself.

EXAMPLE 4-3: Two-Word Instructions

CASE 1:		
Object Code	**Source Code**	
0110 0110 0000 0000	TSTFSZ REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2	; No, execute 2-word instruction
1111 0100 0101 0110		; 2nd operand holds address of REG2
0010 0100 0000 0000	ADDWF REG3	; continue code
CASE 2:		
Object Code	**Source Code**	
0110 0110 0000 0000	TSTFSZ REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2	; Yes
1111 0100 0101 0110		; 2nd operand executed as NOP
0010 0100 0000 0000	ADDWF REG3	; continue code

4.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. Figure 4-8 shows the data memory organization for PIC18C601/801 devices.

The data memory map is divided into banks that contain 256 bytes each. The lower four bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits for the BSR are not implemented.

The data memory contains Special Function Registers (SFR) and General Purpose Registers (GPR). The SFR's are used for control and status of the controller and peripheral functions, while GPR's are used for data storage and scratch pad operations in the user's application. The SFR's start at the last location of Bank 15 (0FFFh) and grow downwards. GPR's start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '0's.

GPR banks 4 and 5 serve as a Program Memory called "Boot RAM", when PGRM bit in MEMCON is set. When PGRM bit is set, any read from "Boot RAM" returns '0's, while any write to it is ignored.

The entire data memory may be accessed directly or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of a File Select Register (FSR). Each FSR holds a 12-bit address value that can be used to access any location in the Data Memory map without banking.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing, or by the use of the `MOVFF` instruction. The `MOVFF` instruction is a two-word/two-cycle instruction that moves a value from one register to another.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access bank. Section 4.10 provides a detailed description of the Access bank.

4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly. Indirect addressing operates through the File Select Registers (FSR). The operation of indirect addressing is shown in Section 4.12.

PIC18C601/801 devices have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other RESETS.

Data RAM is available for use as GPR registers by all instructions. Bank 15 (0F80h to 0FFFh) contains SFR's. All other banks of data memory contain GPR registers starting with bank 0.

4.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-2.

The SFR's can be classified into two sets: those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature.

The SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations are unimplemented and read as '0's. See Table 4-2 for addresses for the SFRs.

4.9.3 SECURED ACCESS REGISTERS

PIC18C601/801 devices contain software programming options for safety critical peripherals. Because these safety critical peripherals can be programmed in software, registers used to control these peripherals are given limited access by the user code. This way, errant code will not accidentally change settings in peripherals that could cause catastrophic results.

The registers that are considered safety critical are the Watchdog Timer register (WDTCON), the External Memory Control register (MEMCON), the Oscillator Control register (OSCCON) and the Chip Select registers (CSSEL2 and CSELIO).

Two bits called Combination Lock (CMLK) bits, located in the lower two bits of the PSPCON register, must be set in sequence by user code to gain access to Secured Access registers.

REGISTER 4-2: PSPCON REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	W-0	W-0
—	—	—	—	—	—	CMLK1	CMLK0
bit 7							bit 0

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

bit 1-0 **CMLK<1:0>**: Combination Lock bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

The Combination Lock bits must be set sequentially, meaning that as soon as Combination Lock bit CMLK1 is set, the second Combination Lock bit CMLK0 must be set on the following instruction cycle. If user waits more than one machine cycle to set the second bit after setting the first, both bits will automatically be cleared in hardware and the lock will remain closed. To satisfy this condition, all interrupts must be disabled before attempting to unlock the Combination Lock. Once secured registers are modified, interrupts may be re-enabled.

Each instruction must only modify one combination lock bit at a time. This means, user code must use the BSF instruction to set CMLK bits in the PSPCON register.

Note: The Combination Lock bits are write-only bits. These bits will always return '0' when read.

When the Combination Lock is opened, the user will have three instruction cycles to modify the safety critical register of choice. After three instruction cycles have expired, the CMLK bits are cleared, the lock will close and the user will have to set the CMLK bits again, in order to open the lock. Since there are only three instruction cycles allowed after the Combination Lock is opened, if a subroutine is used to unlock Combination Lock bits, user code must preload WREG with the desired value, call unlock subroutine, and write to the desired safety critical register itself.

Note: Successive attempts to unlock the Combination Lock must be separated by at least three instruction cycles.

EXAMPLE 4-4: COMBINATION UNLOCK SUBROUTINE EXAMPLE CODE

```
        MOVLW 5Ah           ; Preload WREG with data to be stored in a safety critical register
        BCF INTCON, GIE     ; Disable all interrupts
        CALL UNLOCK         ; Now unlock it
                            ; Write must take place in next instruction cycle
        MOVWF OSCCON
                            ; Lock is closed
        BSF INTCON, GIE     ; Re-enable interrupts
        •
        •
        UNLOCK
        BSF PSPCON, CMLK1
        BSF PSPCON, CMLK0
        RETURN
        •
        •
```


EXAMPLE 4-5: COMBINATION UNLOCK MACRO EXAMPLE CODE

```
UNLOCK_N_MODIFY @REG   MACRO
                       BCF INTCON, GIE           ; Disable interrupts
                       BSF PSPCON, CMLK1
                       BSF PSPCON, CMLK0
                       MOVWF @REG                ; Modify given register
                       BSF INTCON, GIE           ; Enable interrupts
                       ENDM
        •
        •
                       MOVLW 5Ah                 ; Preload WREG for OSCCON register
                       UNLOCK_N_MODIFY OSCCON    ; Modify OSCCON
```

FIGURE 4-7: THE DATA MEMORY MAP FOR PIC18C801/601 (PGRM = 0)

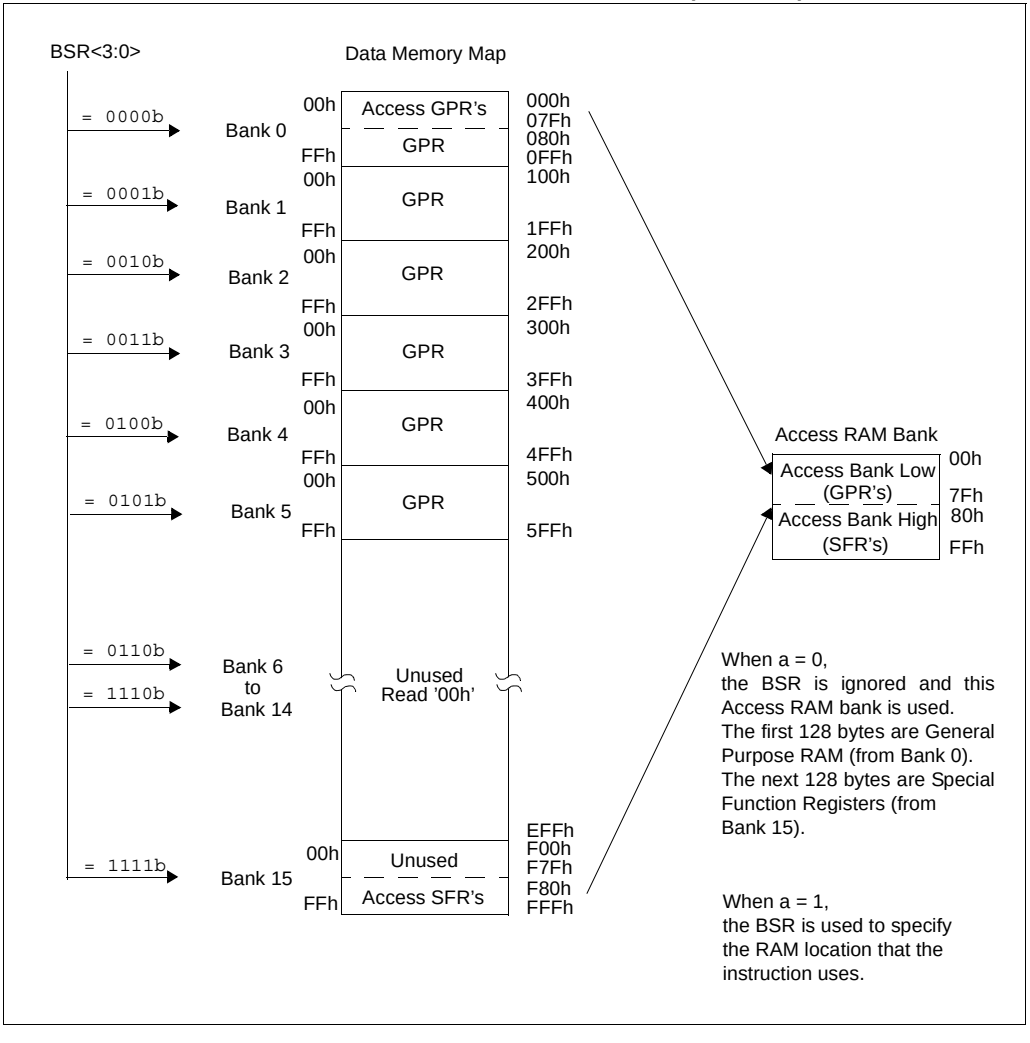


FIGURE 4-8: DATA MEMORY MAP FOR PIC18C601/801 (PGRM = 1)

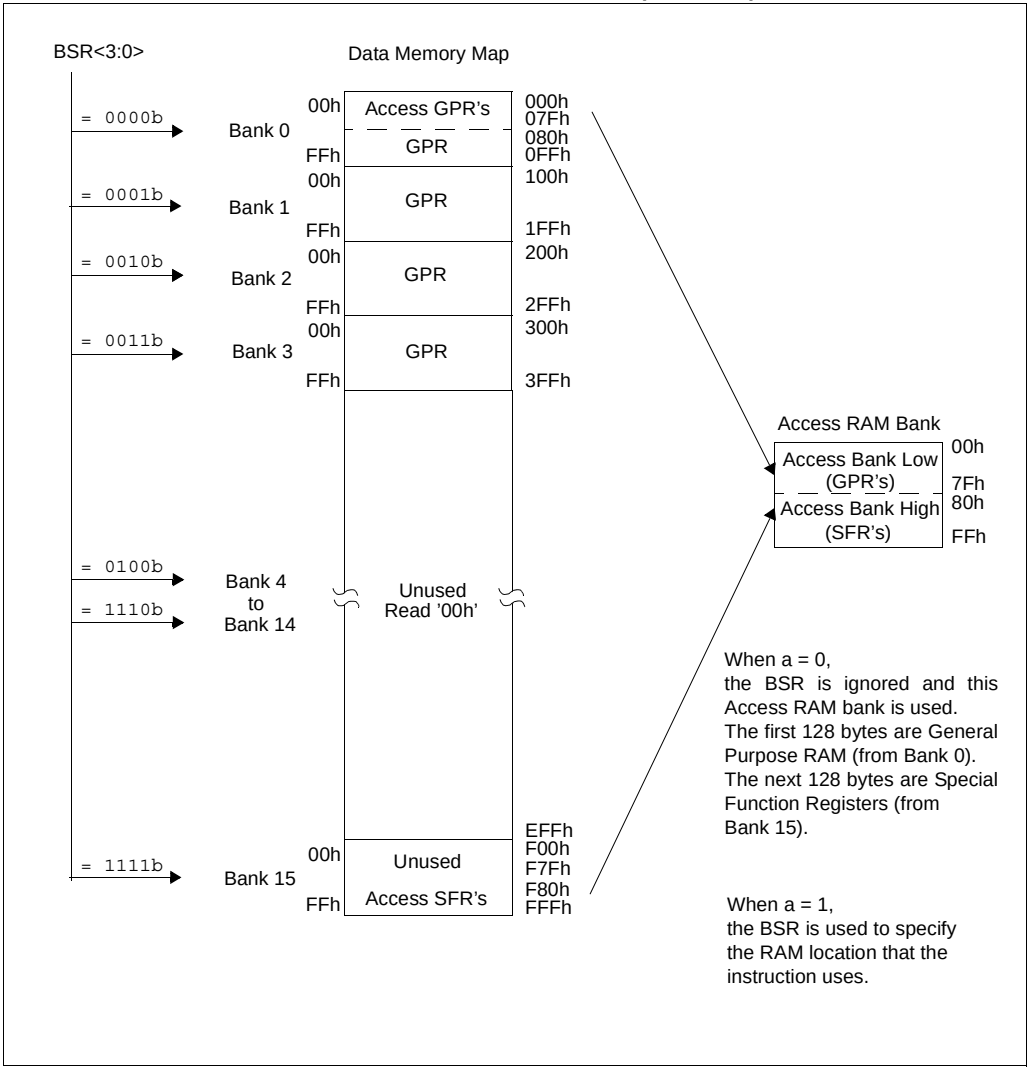


FIGURE 4-9: SPECIAL FUNCTION REGISTER MAP

FFFh	TOSU	FDFh	INDF2	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2	FBCh	CCPR2H	F9Ch	MEMCON
FFBh	PCLATU	FDBh	PLUSW2	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	TRISJ
FF9h	PCL	FD9h	FSR2L	FB9h	Reserved	F99h	TRISH
FF8h	TBLPTRU	FD8h	STATUS	FB8h	Reserved	F98h	TRISG
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	Reserved	F97h	TRISF
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD
FF4h	PRODH	FD4h	Reserved	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ
FF0h	INTCON3	FD0h	RCON	FB0h	PSPCON	F90h	LATH
FEFh	INDF0	FCFh	TMR1H	FAFh	SPBRG	F8Fh	LATG
FEEh	POSTINC0	FCEh	TMR1L	FAEh	RCREG	F8Eh	LATF
FEDh	POSTDEC0	FCDh	T1CON	FADh	TXREG	F8Dh	LATE
FECh	PREINC0	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD
FEBh	PLUSW0	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	—	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	—	F88h	PORTJ
FE7h	INDF1	FC7h	SSPSTAT	FA7h	CSEL2	F87h	PORTH
FE6h	POSTINC1	FC6h	SSPCON1	FA6h	CSELIO	F86h	PORTG
FE5h	POSTDEC1	FC5h	SSPCON2	FA5h	—	F85h	PORTF
FE4h	PREINC1	FC4h	ADRESH	FA4h	—	F84h	PORTE
FE3h	PLUSW1	FC3h	ADRESL	FA3h	—	F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

TABLE 4-2: REGISTER FILE SUMMARY - PIC18C601/801

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS(1)
FFFh	TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					---0 0000	---0 0000
FFEh	TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	0000 0000
FFDh	TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	0000 0000
FFCh	STKPTR	STKOVF	STKUNF	—	Return Stack Pointer					00-0 0000	00-0 0000
FFBh	PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000	---0 0000
FFAh	PCLATH	Holding Register for PC<15:8>								0000 0000	0000 0000
FF9h	PCL	PC Low Byte (PC<7:0>)								0000 0000	0000 0000
FF8h	TBLPTRU	—	—	r	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--r0 0000	--r0 0000
FF7h	TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	0000 0000
FF6h	TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	0000 0000
FF5h	TABLAT	Program Memory Table Latch								0000 0000	0000 0000
FF4h	PRODH	Product Register High Byte								xxxx xxxx	uuuu uuuu
FF3h	PRODL	Product Register Low Byte								xxxx xxxx	uuuu uuuu
FF2h	INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0E	RBIE	TMR0IF	INT0F	RBIF	0000 000x	0000 000u
FF1h	INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	T0IP	—	RBIP	1111 -1-1	1111 -1-1
FF0h	INTCON3	INT2P	INT1P	—	INT2E	INT1E	—	INT2F	INT1F	11-0 0-00	11-0 0-00
FEFh	INDF0	Uses contents of FSR0 to address data memory - value of FSR0 not changed (not a physical register)								N/A	N/A
FEEh	POSTINC0	Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register)								N/A	N/A
FEDh	POSTDEC0	Uses contents of FSR0 to address data memory - value of FSR0 post-decremented (not a physical register)								N/A	N/A
FECh	PREINC0	Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register)								N/A	N/A
FEBh	PLUSW0	Uses contents of FSR0 to address data memory -value of FSR0 offset by WREG (not a physical register)								N/A	N/A
FEAh	FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High				---- xxxx	---- uuuu
FE9h	FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	uuuu uuuu
FE8h	WREG	Working Register								xxxx xxxx	uuuu uuuu
FE7h	INDF1	Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register)								N/A	N/A
FE6h	POSTINC1	Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a physical register)								N/A	N/A
FE5h	POSTDEC1	Uses contents of FSR1 to address data memory - value of FSR1 post-decremented (not a physical register)								N/A	N/A
FE4h	PREINC1	Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register)								N/A	N/A
FE3h	PLUSW1	Uses contents of FSR1 to address data memory - value of FSR1 offset by WREG (not a physical register)								N/A	N/A
FE2h	FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High				---- xxxx	---- uuuu
FE1h	FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	uuuu uuuu
FE0h	BSR	—	—	—	—	Bank Select Register				---- 0000	---- 0000
FDFh	INDF2	Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register)								N/A	N/A
FDEh	POSTINC2	Uses contents of FSR2 to address data memory - value of FSR2 post-incremented (not a physical register)								N/A	N/A
FDDh	POSTDEC2	Uses contents of FSR2 to address data memory - value of FSR2 post-decremented (not a physical register)								N/A	N/A
FDCh	PREINC2	Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register)								N/A	N/A
FDBh	PLUSW2	Uses contents of FSR2 to address data memory -value of FSR2 offset by WREG (not a physical register)								N/A	N/A
FDAh	FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High				---- xxxx	---- uuuu
FD9h	FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	uuuu uuuu
FD8h	STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	---u uuuu

Legend x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved

Note 1: Other (non-power-up) RESETS include external RESET through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.
2: These registers can only be modified when the Combination Lock is open.
3: These registers are available on PIC18C801 only.

TABLE 4-2: REGISTER FILE SUMMARY - PIC18C601/801 (CONTINUED)

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS(1)
FD7h	TMR0H	Timer0 Register High Byte								0000 0000	0000 0000
FD6h	TMR0L	Timer0 Register Low Byte								xxxx xxxx	uuuu uuuu
FD5h	T0CON	TMR0ON	16BIT	T0CS	T0SE	T0PS3	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
FD4h	Reserved									rrrr rrrr	rrrr rrrr
FD3h	OSCCON(2)	—	—	—	—	LOCK	PLLEN	SCS1	SCS0	---- 0000	---- uuu0
FD2h	LVDCON(2)	—	—	IRVST	LVDEN	LVV3	LVV2	LVV1	LVV0	--00 0101	--00 0101
FD1h	WDTCON(2)	—	—	—	—	WDPS2	WDPS1	WDPS0	SWDTEN	---- 0000	---- xxxx
FD0h	RCON	IPEN	r	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	r	00-1 11qq	00-q qquu
FCFh	TMR1H	Timer1 Register High Byte								xxxx xxxx	uuuu uuuu
FCEh	TMR1L	Timer1 Register Low Byte								xxxx xxxx	uuuu uuuu
FCDh	T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
FCCh	TMR2	Timer2 Register								0000 0000	0000 0000
FCBh	PR2	Timer2 Period Register								1111 1111	1111 1111
FCAh	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
FC9h	SSPBUF	SSP Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
FC8h	SSPADD	SSP Address Register in I^2C Slave Mode. SSP Baud Rate Reload Register in I^2C Master Mode								0000 0000	0000 0000
FC7h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000
FC6h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
FC5h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
FC4h	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
FC3h	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
FC2h	ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	--00 0000	--00 0000
FC1h	ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000
FC0h	ADCON2	ADFM	—	—	—	—	ADCS2	ADCS1	ADCS0	0--- -000	0--- -000
FBFh	CCPR1H	Capture/Compare/PWM Register1 High Byte								xxxx xxxx	uuuu uuuu
FBEh	CCPR1L	Capture/Compare/PWM Register1 Low Byte								xxxx xxxx	uuuu uuuu
FBDh	CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
FBCh	CCPR2H	Capture/Compare/PWM Register2 High Byte								xxxx xxxx	uuuu uuuu
FBBh	CCPR2L	Capture/Compare/PWM Register2 Low Byte								xxxx xxxx	uuuu uuuu
FBAh	CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--uu uuuu
FB9h	Reserved									rrrr rrrr	rrrr rrrr
FB8h	Reserved									rrrr rrrr	rrrr rrrr
FB7h	Reserved									rrrr rrrr	rrrr rrrr
FB6h											
FB5h											
FB4h											
FB3h	TMR3H	Timer3 Register High Byte								xxxx xxxx	uuuu uuuu
FB2h	TMR3L	Timer3 Register Low Byte								xxxx xxxx	uuuu uuuu
FB1h	T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved

Note 1: Other (non-power-up) RESETS include external RESET through $\overline{MCLR}$ and Watchdog Timer Reset.
2: These registers can only be modified when the Combination Lock is open.
3: These registers are available on PIC18C801 only.

TABLE 4-2: REGISTER FILE SUMMARY - PIC18C601/801 (CONTINUED)

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS(1)
FB0h	PSPCON	—	—	—	—	—	—	CMLK1	CMLK0	---- --00	---- --00
FAFh	SPBRG	USART Baud Rate Generator								0000 0000	0000 0000
FAEh	RCREG	USART Receive Register								0000 0000	0000 0000
FADh	TXREG	USART Transmit Register								0000 0000	0000 0000
FACh	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
FABh	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	0000 000x
FAAh											
FA9h											
FA8h											
FA7h	CSEL2(2)	CSL7	CSL6	CSL5	CSL4	CSL3	CSL2	CSL1	CSL0	1111 1111	uuuu uuuu
FA6h	CSELIO(2)	CSIO7	CSIO6	CSIO5	CSIO4	CSIO3	CSIO2	CSIO1	CSIO0	1111 1111	uuuu uuuu
FA5h											
FA4h											
FA3h											
FA2h	IPR2	—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP	---- 1111	---- 1111
FA1h	PIR2	—	—	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	---- 0000	---- 0000
FA0h	PIE2	—	—	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	---- 0000	---- 0000
F9Fh	IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-111 1111	-111 1111
F9Eh	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
F9Dh	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
F9Ch	MEMCON(2)	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00
F9Bh											
F9Ah	TRISJ(3)	Data Direction Control Register for PORTJ								1111 1111	1111 1111
F99h	TRISH(3)	Data Direction Control Register for PORTH								1111 1111	1111 1111
F98h	TRISG	—	—	—	Read PORTG Data Latch, Write PORTG Data Latch					---1 1111	---1 1111
F96h	TRISF	Read PORTF Data Latch, Write PORTF Data Latch								1111 1111	1111 1111
F96h	TRISE	Data Direction Control Register for PORTE								1111 1111	1111 1111
F95h	TRISD	Data Direction Control Register for PORTD								1111 1111	1111 1111
F94h	TRISC	Data Direction Control Register for PORTC								1111 1111	1111 1111
F93h	TRISB	Data Direction Control Register for PORTB								1111 1111	1111 1111
F92h	TRISA	—	—	Data Direction Control Register for PORTA						--11 1111	--11 1111
F91h	LATJ(3)	Read PORTJ Data Latch, Write PORTJ Data Latch								xxxx xxxx	uuuu uuuu
F90h	LATH(3)	Read PORTH Data Latch, Write PORTH Data Latch								xxxx xxxx	uuuu uuuu
F8Fh	LATG	—	—	—	Read PORTG Data Latch, Write PORTG Data Latch					---x xxxx	---u uuuu
F8Eh	LATF	Read PORTF Data Latch, Write PORTF Data Latch								xxxx xxxx	uuuu uuuu
F8Dh	LATE	Read PORTE Data Latch, Write PORTE Data Latch								xxxx xxxx	uuuu uuuu
F8Ch	LATD	Read PORTD Data Latch, Write PORTD Data Latch								xxxx xxxx	uuuu uuuu
F8Bh	LATC	Read PORTC Data Latch, Write PORTC Data Latch								xxxx xxxx	uuuu uuuu
F8Ah	LATB	Read PORTB Data Latch, Write PORTB Data Latch								xxxx xxxx	uuuu uuuu
F89h	LATA	—	—	Read PORTA Data Latch, Write PORTA Data Latch						--xx xxxx	--uu uuuu
F88h	PORTJ(3)	Read PORTJ Pins, Write PORTJ Data Latch								xxxx xxxx	uuuu uuuu
F87h	PORTH(3)	Read PORTH pins, Write PORTH Data Latch								xxxx xxxx	uuuu uuuu
F86h	PORTG	—	—	—	Read PORTG pins, Write PORTG Data Latch					---x xxxx	---u uuuu
F85h	PORTF	Read PORTF pins, Write PORTF Data Latch								xxxx xx00	uuuu uu00

Legend x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved

Note 1: Other (non-power-up) RESETS include external RESET through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.
2: These registers can only be modified when the Combination Lock is open.
3: These registers are available on PIC18C801 only.

TABLE 4-2: REGISTER FILE SUMMARY - PIC18C601/801 (CONTINUED)

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS(1)
F84h	PORTE	Read PORTE Pins, Write PORTE Data Latch								xxxx xxxx	uuuu uuuu
F83h	PORTD	Read PORTD pins, Write PORTD Data Latch								xxxx xxxx	uuuu uuuu
F82h	PORTC	Read PORTC pins, Write PORTC Data Latch								xxxx xxxx	uuuu uuuu
F81h	PORTB	Read PORTB pins, Write PORTB Data Latch								xxxx xxxx	uuuu uuuu
F80h	PORTA	—	—	Read PORTA pins, Write PORTA Data Latch						--0x 0000	--0u 0000

Legend x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved

Note 1: Other (non-power-up) RESETS include external RESET through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.
2: These registers can only be modified when the Combination Lock is open.
3: These registers are available on PIC18C801 only.

4.10 Access Bank

The Access Bank is an architectural enhancement that is very useful for C compiler code optimization. The techniques used by the C compiler are also useful for programs written in assembly.

This data memory region can be used for:

- Intermediate computational values
- Local variables of subroutines
- Faster context saving/switching of variables
- Common variables
- Faster evaluation/control of SFR's (no banking)

The Access Bank is comprised of the upper 128 bytes in Bank 15 (SFR's) and the lower 128 bytes in Bank 0. These two sections will be referred to as Access Bank High and Access Bank Low, respectively. Figure 4-8 indicates the Access Bank areas.

A bit in the instruction word specifies if the operation is to occur in the bank specified by the BSR register, or in the Access Bank.

When forced in the Access Bank (a = '0'), the last address in Access Bank Low is followed by the first address in Access Bank High. Access Bank High maps all Special Function Registers so that these registers can be accessed without any software overhead.

4.11 Bank Select Register (BSR)

The need for a large general purpose memory space dictates a RAM banking scheme. When using direct addressing, the BSR should be configured for the desired bank.

BSR<3:0> holds the upper 4 bits of the 12-bit RAM address. The BSR<7:4> bits will always read '0's, and writes will have no effect.

A `MOVLB` instruction has been provided in the instruction set to assist in selecting banks.

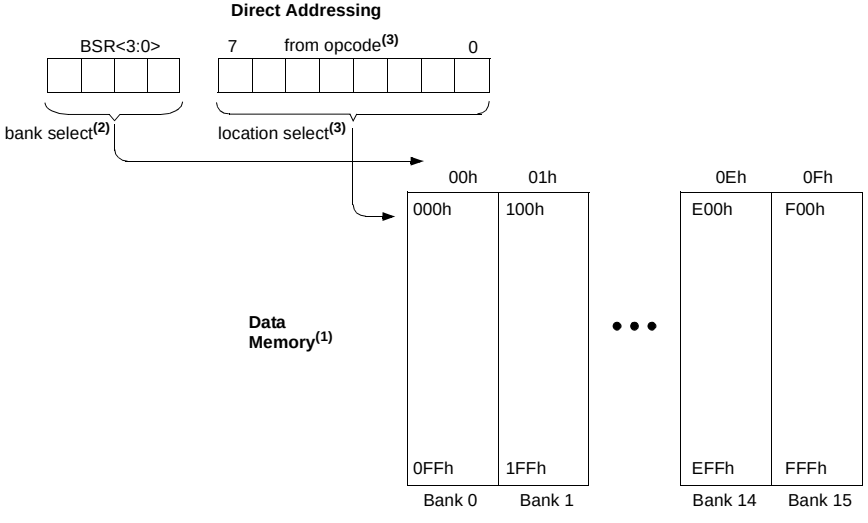
If the currently selected bank is not implemented, any read will return all '0's and all writes are ignored. The STATUS register bits will be set/cleared as appropriate for the instruction performed.

Each Bank extends up to 0FFh (256 bytes). All data memory is implemented as static RAM.

A `MOVFF` instruction ignores the BSR, since the 12-bit addresses are embedded into the instruction word.

Section 4.12 provides a description of indirect addressing, which allows linear addressing of the entire RAM space.

FIGURE 4-10: DIRECT ADDRESSING



Note 1: For register file map detail, see Table 4-2.

2: The access bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

3: The `MOVFF` instruction embeds the entire 12-bit address in the instruction.

4.12 Indirect Addressing, INDF and FSR Registers

Indirect addressing is a mode of addressing data memory, where the data memory address in the instruction is not fixed. A SFR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory and for software stacks. Figure 4-11 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Indirect addressing is possible by using one of the INDFn ($0 \le n \le 2$) registers. Any instruction using the INDFn register actually accesses the register indicated by the File Select Register, FSRn ($0 \le n \le 2$). Reading the INDFn register itself indirectly (FSRn = '0'), will read 00h. Writing to the INDFn register indirectly, results in a no-operation. The FSRn register contains a 12-bit address, which is shown in Figure 4-11.

Example 4-6 shows a simple use of indirect addressing to clear the RAM in Bank 1 (locations 100h-1FFh) in a minimum number of instructions.

EXAMPLE 4-6: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
    LFSR   FSR0, 100h     ;
NEXTCLRF   POSTINC0       ; Clear INDF
                          ; register
                          ; & inc pointer
    BTFSS  FSR0H, 1       ; All done
                          ; with Bank1?
    BRA    NEXT           ; NO, clear next
CONTINUE;
    :                     ; YES, continue
```

There are three indirect addressing registers. To address the entire data memory space (4096 bytes), these registers are 12-bit wide. To store the 12-bits of addressing information, two 8-bit registers are required. These indirect addressing registers are:

1. FSR0: composed of FSR0H:FSR0L
2. FSR1: composed of FSR1H:FSR1L
3. FSR2: composed of FSR2H:FSR2L

In addition, there are registers INDF0, INDF1 and INDF2, which are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data.

If an instruction writes a value to INDF0, the value will be written to the address indicated by FSR0H:FSR0L. A read from INDF1 reads the data from the address indicated by FSR1H:FSR1L. INDFn can be used in code anywhere an operand can be used.

If INDF0, INDF1, or INDF2 are read indirectly via an FSR, all '0's are read (zero bit is set). Similarly, if INDF0, INDF1, or INDF2 are written to indirectly, the operation will be equivalent to a `NOP` instruction and the STATUS bits are not affected.

4.12.1 INDIRECT ADDRESSING OPERATION

Each FSR register has an INDF register associated with it, plus four additional register addresses. Performing an operation on one of these five registers determines how the FSR will be modified during indirect addressing.

When data access is done to one of the five INDFn locations, the address selected will configure the FSRn register to:

- Do nothing to FSRn after an indirect access (no change) - INDFn
- Auto-decrement FSRn after an indirect access (post-decrement) - POSTDECn
- Auto-increment FSRn after an indirect access (post-increment) - POSTINCn
- Auto-increment FSRn before an indirect access (pre-increment) - PREINCn
- Use the value in the WREG register as an offset to FSRn. Do not modify the value of the WREG or the FSRn register after an indirect access (no change) - PLUSWn

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the STATUS register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

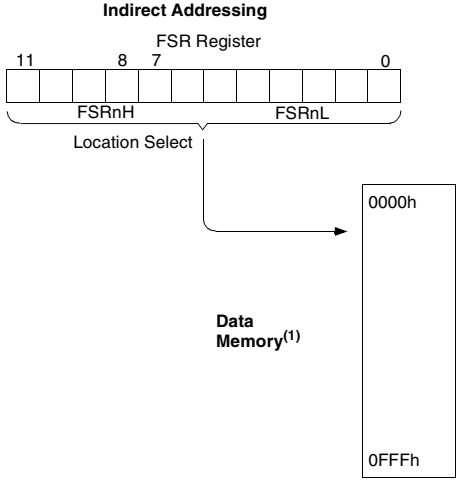
Incrementing or decrementing an FSR affects all 12 bits. That is, when FSRnL overflows from an increment, FSRnH will be incremented automatically.

Adding these features allows the FSRn to be used as a software stack pointer, in addition to its uses for table operations in data memory.

Each FSR has an address associated with it that performs an indexed indirect access. When a data access to this INDFn location (PLUSWn) occurs, the FSRn is configured to add the 2's complement value in the WREG register and the value in FSR to form the address before an indirect access. The FSR value is not changed.

If an indirect addressing operation is done where the target address is an FSRnH or FSRnL register, the write operation will dominate over the pre- or post-increment/decrement functions.

FIGURE 4-11: INDIRECT ADDRESSING



Note 1: For register file map detail, see Table 4-2.

4.13 STATUS Register

The STATUS register, shown in Register 4-3, contains the arithmetic status of the ALU. The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV, or N bits, then the write to these five bits is disabled. These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear all implemented bits and set the Z bit. This leaves the STATUS register as `---0 0100` (where - = unimplemented).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV, or N bits from the STATUS register. For other instructions which do not affect the status bits, see Table 20-2.

Note:	The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ bit respectively, in subtraction.

REGISTER 4-3: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

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

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (2's complement). It indicates whether the result of the ALU operation was negative (ALU MSb = 1).
`1` = Result was negative
`0` = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.
`1` = Overflow occurred for signed arithmetic (in this arithmetic operation)
`0` = No overflow occurred

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/$\overline{\text{borrow}}$ bit
For arithmetic addition and subtraction instructions
`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

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRCF`, `RRNCF`, `RLCF`, and `RLNCF`) instructions, this bit is loaded with either the bit 4, or bit 3 of the source register.

bit 0 **C:** Carry/$\overline{\text{borrow}}$ bit
For arithmetic addition and subtraction instructions
`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: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRCF`, `RLCF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.14 RCON Register

The Reset Control (RCON) register contains flag bits that allow differentiation between the sources of a device RESET. These flags include the $\overline{TO}$, $\overline{PD}$, $\overline{POR}$ and $\overline{RI}$ bits. This register is readable and writable.

Note:	It is recommended that the $\overline{POR}$ bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

REGISTER 4-4: RCON REGISTER

R/W-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-0	U-0
IPEN	r	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	r
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (16CXXX compatibility mode)

bit 6 **Reserved:** Maintain as '0'

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed
0 = The RESET instruction was executed causing a device RESET
(must be set in software after RESET instruction was executed)

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
1 = A Power-on Reset has not occurred
0 = A Power-on Reset occurred
(must be set in software after a Power-on Reset occurs)

bit 0 **Reserved:** Maintain as '0'

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown
r = Reserved			

5.0 EXTERNAL MEMORY INTERFACE

The External Memory Interface is a feature of the PIC18C601/801 that allows the processor to access external memory devices, such as FLASH, EPROM, SRAM, etc. Memory mapped peripherals may also be accessed.

The External Memory Interface physical implementation includes up to 26 pins on the PIC18C601 and up to 38 pins on the PIC18C801. These pins are reserved for external address/data bus functions.

These pins are multiplexed with I/O port pins, but the I/O functions are only enabled when program execution takes place in internal Boot RAM and the EBDIS bit in the MEMCON register is set (see Register 5-1).

5.1 Memory Control Register (MEMCON)

Register 5-1 shows the Memory Control Register (MEMCON). This register contains bits used to control the operation of the External Memory Interface.

REGISTER 5-1: MEMCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0
bit7							bit0

bit 7 **EBDIS**: External Bus Disable

1 = External system bus disabled, all external bus drivers are mapped as I/O ports
0 = External system bus enabled, and I/O ports are disabled

bit 6 **PGRM**: Program RAM Enable

1 = 512 bytes of internal RAM enabled as internal program memory from location 1FFE00h to 1FFFFFh, external program memory at these locations is unused. Internal GPR memory from 400h to 5FFh is disabled and returns 00h.

0 = Internal RAM enabled as internal GPR memory from 400h to 5FFh. Program memory from location 1FFE00h to 1FFFFFh is configured as external program memory.

bit 5-4 **WAIT<1:0>**: Table Reads and Writes Bus Cycle Wait Count

11 = Table reads and writes will wait 0 TCY
10 = Table reads and writes will wait 1 TCY
01 = Table reads and writes will wait 2 TCY
00 = Table reads and writes will wait 3 TCY

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

bit 1-0 **WM<1:0>**: TABLWT Operation with 16-bit Bus

1X = Word Write mode: TABLAT0 and TABLAT1 word output, $\overline{WRH}$ active when TABLAT1 written
01 = Byte Select mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ and ($\overline{UB}$ or $\overline{LB}$) will activate
00 = Byte Write mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ or $\overline{WRL}$ will activate

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

5.2 8-bit Mode

The External Memory Interface can operate in 8-bit mode. The mode selection is not software configurable, but is programmable via the configuration bits.

There are two types of connections in 8-bit mode. They are referred to as:

- 8-bit Multiplexed
- 8-bit De-Multiplexed

5.2.1 8-BIT MULTIPLEXED MODE

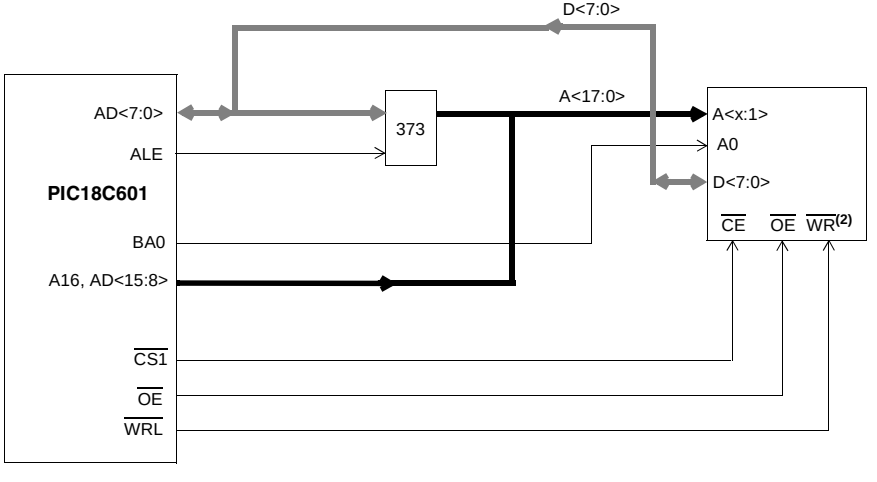
The 8-bit Multiplexed mode applies only to the PIC18C601. Data and address lines are multiplexed on port pins and must be decoded with glue logic.

For 8-bit Multiplexed mode on the PIC18C601, the instructions will be fetched as two 8-bit bytes on a shared data/address bus (PORTD). The two bytes are sequentially fetched within one instruction cycle (TCY).

Therefore, the designer must choose external memory devices according to timing calculations based on 1/2 Tcy (2 times instruction rate). For proper memory speed selection, glue logic propagation delay times must be considered along with setup and hold times.

The Address Latch Enable (ALE) pin indicates that the address bits A<7:0> are available on the External Memory Interface bus. The $\overline{OE}$ output enable signal will enable one byte of program memory for a portion of the instruction cycle, then BA0 will change and the second byte will be enabled to form the 16-bit instruction word. The least significant bit of the address, BA0, must be connected to the memory devices in this mode. Figure 5-1 shows an example of 8-bit Multiplexed mode on the PIC18C601. The control signals used in 8-bit Multiplexed mode are outlined in Table 5-1. Register 5-2 describes 8-bit Multiplexed mode timing.

FIGURE 5-1: 8-BIT MULTIPLEXED MODE EXAMPLE

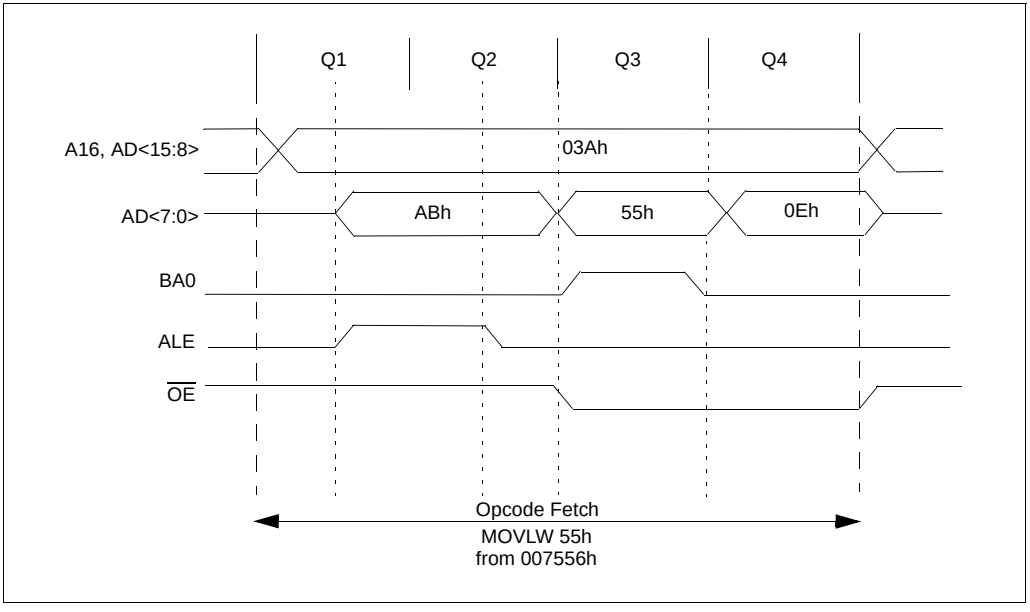


Note 1: This signal only applies to Table Writes. See Section 6.0, Table Reads and Writes.

TABLE 5-1: 8-BIT MULTIPLEXED MODE CONTROL SIGNALS

Name	8-bit Mux Mode	Function
RG0/ALE	ALE	Address Latch Enable (ALE) control pin
RG1/$\overline{OE}$	OE	Output Enable ($\overline{OE}$) control pin
RG2/$\overline{WRL}$	WRL	Write Low ($\overline{WRL}$) control pin
RG4/BA0	BA0	Byte address bit 0
RF3/$\overline{CSIO}$	CSIO	Chip Select I/O (See Section 5.4)
RF5/$\overline{CS1}$	CS1	Chip Select 1 (See Section 5.4)

FIGURE 5-2: 8-BIT MULTIPLEXED MODE TIMING



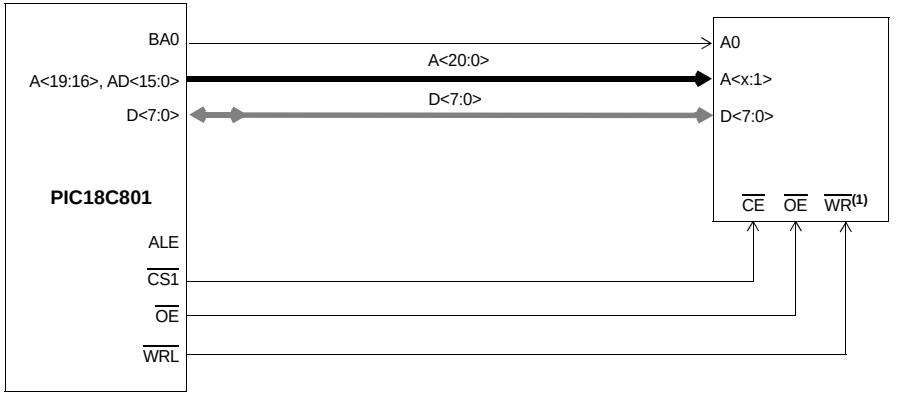
5.2.2 8-BIT DE-MULTIPLEXED MODE

The 8-bit De-Multiplexed mode applies only to the PIC18C801. Data and address lines are available separately. External components are not necessary in this mode.

For 8-bit De-Multiplexed mode on the PIC18C801, the instructions are fetched as two 8-bit bytes on a dedicated data bus (PORTJ). The address will be presented for the entire duration of the fetch cycle on a separate address bus. The two instruction bytes are sequentially fetched within one instruction cycle (TCY). Therefore, the designer must choose external memory devices according to timing calculations, based on 1/2 TCY (2 times instruction rate). For proper memory speed selection, setup and hold times must be considered.

The Address Latch Enable (ALE) pin is left unconnected, since glue logic is not necessary. The $\overline{OE}$ output enable signal will enable one byte of program memory for a portion of the instruction cycle, then BA0 will change and the second byte will be enabled to form the 16-bit instruction word. The least significant bit of the address, BA0, must be connected to the memory devices in this mode. Figure 5-3 shows an example of 8-bit De-Multiplexed mode on the PIC18C801. The control signals used in 8-bit De-Multiplexed mode are outlined in Register 5-2. Register 5-4 describes 8-bit De-Multiplexed mode timing.

FIGURE 5-3: 8-BIT DE-MULTIPLEXED MODE EXAMPLE

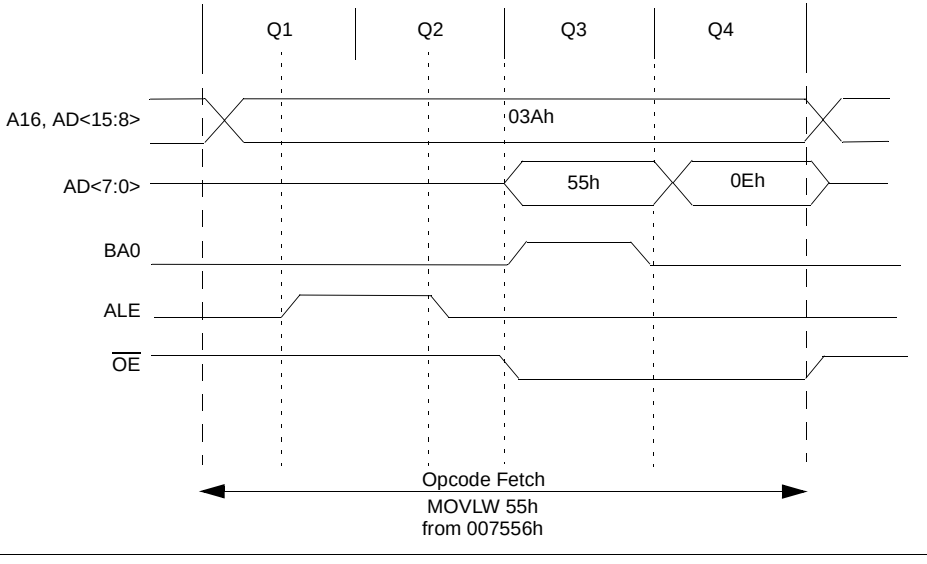


Note 1: This signal only applies to Table Writes. See Section 6.0, Table Reads and Writes.

TABLE 5-2: 8-BIT DE-MULTIPLEXED MODE CONTROL SIGNALS

Name	8-bit De-Mux Mode	Function
RG0/ALE	ALE	Address Latch Enable (ALE) control pin
RG1/$\overline{OE}$	OE	Output Enable ($\overline{OE}$) control pin
RG2/$\overline{WRL}$	WRL	Write Low ($\overline{WRL}$) control pin
RG4/BA0	BA0	Byte address bit 0
RF3/$\overline{CSIO}$	CSIO	Chip Select I/O (See Section 5.4)
RF4/$\overline{CS2}$	CS2	Chip Select 2 (See Section 5.4)
RF5/$\overline{CS1}$	CS1	Chip Select 1 (See Section 5.4)

FIGURE 5-4: 8-BIT DE-MULTIPLEXED MODE TIMING



5.3 16-bit Mode

The External Memory Interface can operate in 16-bit mode. The mode selection is not software configurable, but is programmable via the configuration bits.

The WM<1:0> bits in the MEMCON register determine three types of connections in 16-bit mode. They are referred to as:

- 16-bit Byte Write
- 16-bit Word Write
- 16-bit Byte Select

These three different configurations allow the designer maximum flexibility in using 8-bit and 16-bit memory devices.

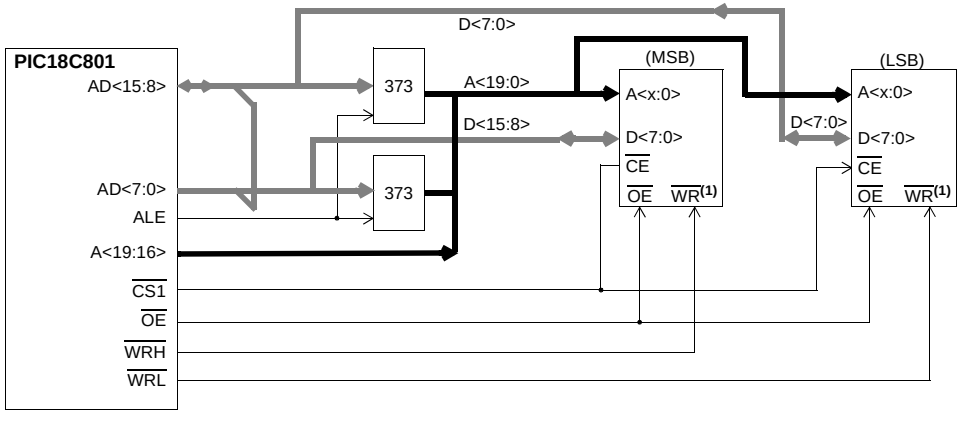
For all 16-bit modes, the Address Latch Enable (ALE) pin indicates that the address bits A<15:0> are available on the External Memory Interface bus. Following the address latch, the output enable signal ($\overline{OE}$) will enable both bytes of program memory at once to form a 16-bit instruction word.

In Byte Select mode, JEDEC standard FLASH memories will require BA0 for the byte address line, and one I/O line, to select between byte and word mode. The other 16-bit modes do not need BA0. JEDEC standard static RAM memories will use the $\overline{UB}$ or $\overline{UL}$ signals for byte selection.

5.3.1 16-BIT BYTE WRITE MODE

Figure 5-5 shows an example of 16-bit Byte Write mode for the PIC18C601/801.

FIGURE 5-5: 16-BIT BYTE WRITE MODE EXAMPLE

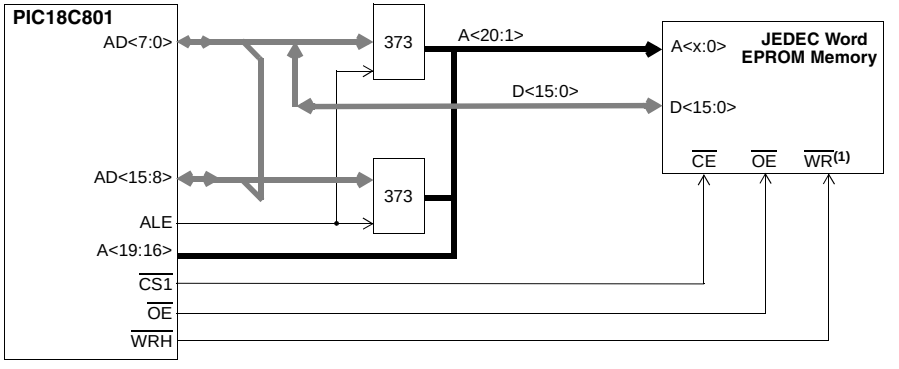


Note 1: This signal only applies to Table Writes. See Section 6.0, Table Reads and Writes.

5.3.2 16-BIT WORD WRITE MODE

Figure 5-6 shows an example of 16-bit Word Write mode for the PIC18C801.

FIGURE 5-6: 16-BIT WORD WRITE MODE EXAMPLE

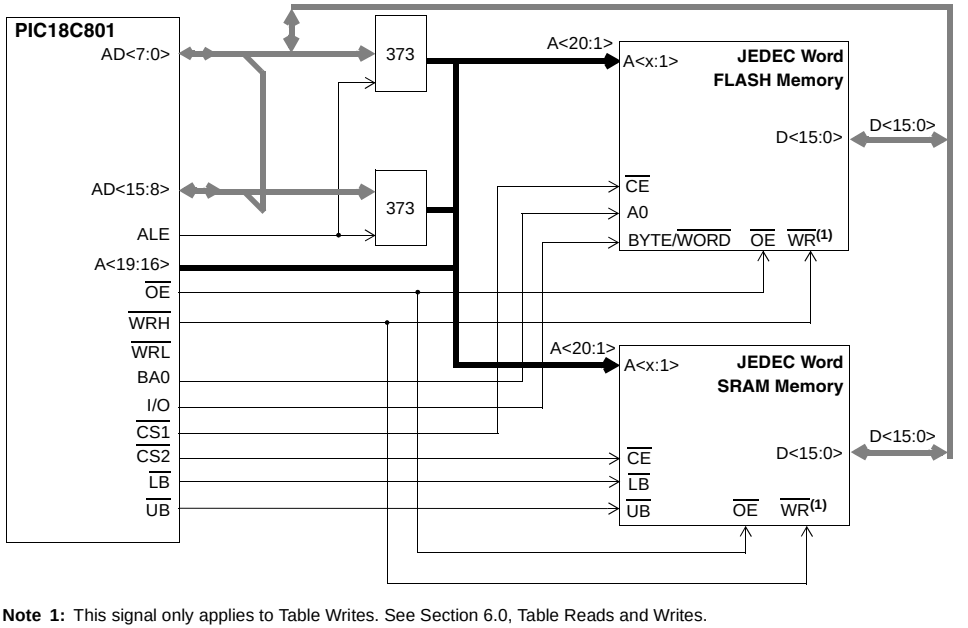


Note 1: This signal only applies to Table Writes. See Section 6.0, Table Reads and Writes.

5.3.3 16-BIT BYTE SELECT MODE

Figure 5-7 shows an example of 16-bit Byte Select mode for the PIC18C801.

FIGURE 5-7: 16-BIT BYTE SELECT MODE EXAMPLE



Note 1: This signal only applies to Table Writes. See Section 6.0, Table Reads and Writes.

5.3.4 16-BIT MODE CONTROL SIGNALS

Table 5-3 describes the 16-bit mode control signals for the PIC18C601/801.

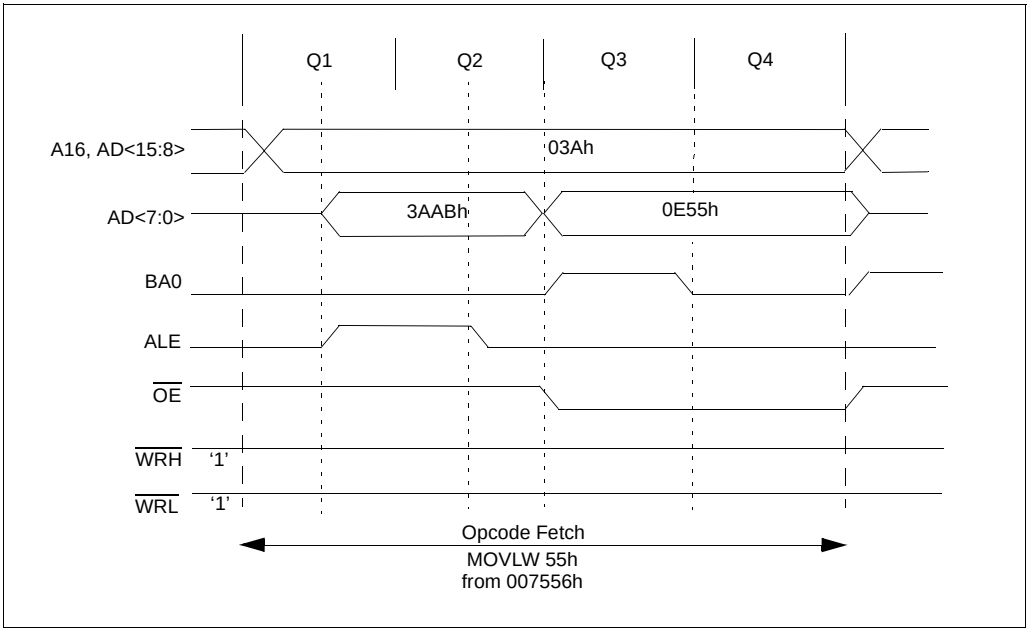
TABLE 5-3: PIC18C601/801 16-BIT MODE CONTROL SIGNALS

Name	18C601 16-bit Mode	18C801 16-bit Mode	Function
RG0/ALE	ALE	ALE	Address Latch Enable (ALE) control pin
RG1/$\overline{OE}$	OE	OE	Output Enable ($\overline{OE}$) control pin
RG2/$\overline{WRL}$	WRL	WRL	Write Low ($\overline{WRL}$) control pin
RG3/$\overline{WRH}$	WRH	WRH	Write High ($\overline{WRH}$) control pin
RG4/BA0	BA0	BA0	Byte address bit 0
RF3/$\overline{CSIO}$	CSIO	CSIO	Chip Select I/O (See Section 5.4)
RF4/$\overline{CS2}$	N/A	CS2	Chip Select 2 (See Section 5.4)
RF5/$\overline{CS1}$	CS1	CS1	Chip Select 1 (See Section 5.4)
RF6/$\overline{UB}$	UB	UB	Upper Byte Enable ($\overline{UB}$) control pin
RF7/$\overline{LB}$	LB	LB	Lower Byte Enable ($\overline{LB}$) control pin
I/O	I/O	I/O	I/O as BYTE/$\overline{WORD}$ control pin for JEDEC FLASH

5.3.5 16-BIT MODE TIMING

Figure 5-8 describes the 16-bit mode timing for the PIC18C601/801.

FIGURE 5-8: 16-BIT MODE TIMING



5.4 Chip Selects

Chip select signals are used to select regions of external memory and I/O devices for access. The PIC18C801 has three chip selects and all are programmable. The chip select signals are $\overline{CS1}$, $\overline{CS2}$ and $\overline{CSIO}$. $\overline{CS1}$ and $\overline{CS2}$ are general purpose chip selects that are used to enable large portions of program memory. $\overline{CSIO}$ is used to enable external I/O expansion. The PIC18C601uses two of these programmable chip selects: $\overline{CS1}$ and $\overline{CSIO}$.

Two SFRs are used to control the chip select signals. These are CSEL2 and CSELIO (see Register 5-2 and Register 5-3). A chip select signal is asserted low when the CPU makes an access to a dedicated range of addresses specified in the chip select registers, CSEL2 and CSELIO. The 8-bit value found in either of these registers is decoded as one of 256, 8K banks of program memory. If both chip select registers are 00h, all of the chip select signals are disabled and their corresponding pins are configured as I/O. Since the last 512 bytes of program memory are dedicated to internal program RAM, the chip select signals will not activate if the program memory address falls in this range.

REGISTER 5-2: CSEL2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CSL7	CSL6	CSL5	CSL4	CSL3	CSL2	CSL1	CSL0
bit 7							bit 0

bit 7-0 **CSL<7:0>**: Chip Select 2 Address Decode bits

XXh = All eight bits are compared to the Most Significant bits PC<20:13> of the program counter. If PC<20:13> ≥ CSL<7:0> register, then the $\overline{CS2}$ signal is low. If PC<20:13> < CSL<7:0>, $\overline{CS2}$ is high.

00h = $\overline{CS2}$ is inactive

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 5-3: CSELIO REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CSIO7	CSIO6	CSIO5	CSIO4	CSIO3	CSIO2	CSIO1	CSIO0
bit7							bit0

bit 7-0 **CSIO<7:0>**: Chip Select IO Address Decode bits

XXh = All eight bits are compared to the Most Significant bits PC<20:13> of the program counter. If PC<20:13> = CSIO<7:0>, then the $\overline{CSIO}$ signal is low. If not, $\overline{CSIO}$ is high.

00h = $\overline{CSIO}$ is inactive

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

5.4.1 CHIP SELECT 1 ($\overline{CS1}$)

$\overline{CS1}$ is enabled by writing a value other than 00h into either the CSEL2 register, or the CSELIO register. If both of the chip select registers are programmed to 00h, the $\overline{CS1}$ signal is not enabled and the RF5 pin is configured as I/O.

$\overline{CS1}$ is low for all addresses in which $\overline{CS2}$ and $\overline{CSELIO}$ are high. Therefore, if CSEL2 = 20h and CSELIO = 80h, then the $\overline{CS1}$ signal will be low for the address that falls between 000000h and (2000h x 20h) - 1 = 03FFFFh. $\overline{CS1}$ will always be low for the lower 8K of program memory. Figure 5-9 shows an example address map for $\overline{CS1}$.

5.4.2 CHIP SELECT 2 ($\overline{CS2}$)

$\overline{CS2}$ is enabled for program memory accesses, starting at the address derived by the 8-bit value contained in CSEL2. For example, if the value contained in the CSEL2 register is 80h, then the $\overline{CS2}$ signal will be asserted low whenever the address is greater than or equal to 2000h x 80h = 100000h.

A 00h value in the CSEL2 register will disable the $\overline{CS2}$ signal and will configure the RF4 pin as I/O. Figure 5-9 shows an example address map for $\overline{CS2}$.

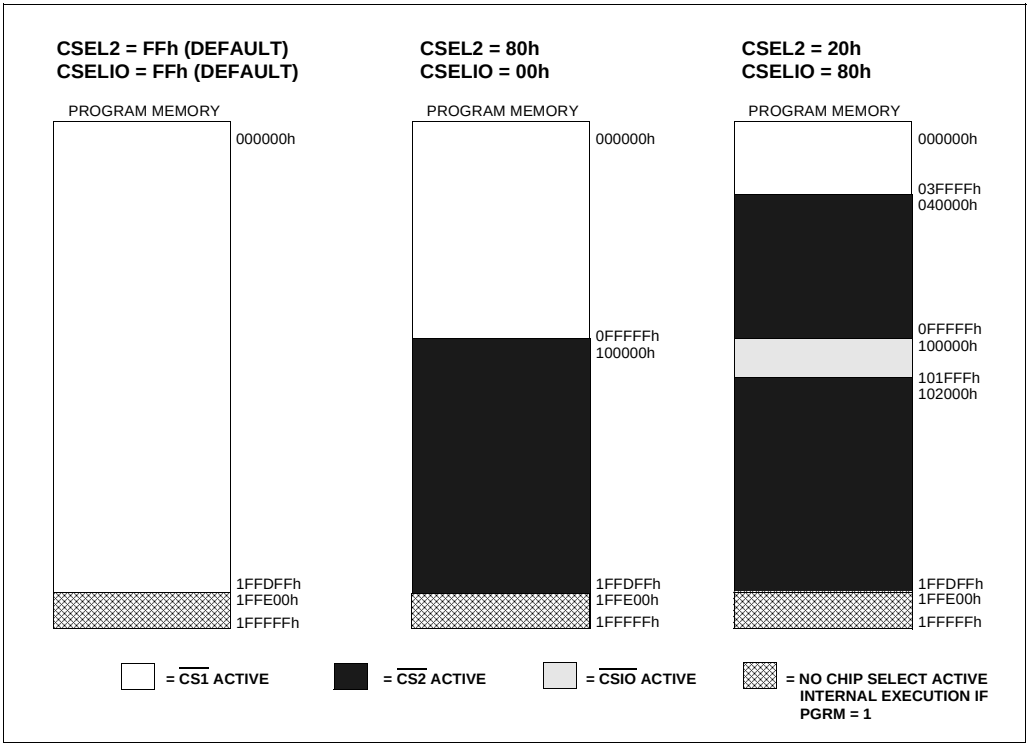
5.4.3 CHIP SELECT I/O ($\overline{CSIO}$)

$\overline{CSIO}$ is enabled for a fixed 8K address range starting at the address defined by the 8-bit value contained in CSELIO. If, for instance, the value contained in the CSELIO register is 80h, then the $\overline{CSIO}$ signal will be low for the address range between 100000h and 101FFFh.

If the 8K address block overlaps the address range specified in the CSEL2 register, the $\overline{CSIO}$ signal will be low, and the $\overline{CS2}$ signal will be high, for that region.

A 00h value in the CSELIO register will disable the $\overline{CSIO}$ signal and will configure the RF3 pin as I/O. Figure 5-9 shows an example address map for $\overline{CSIO}$.

FIGURE 5-9: EXAMPLE CONFIGURATION ADDRESS MAP FOR $\overline{CS1}$, $\overline{CS2}$, AND $\overline{CSIO}$



5.5 External Wait Cycles

The external memory interface supports wait cycles. Wait cycles only apply to Table Read and Table Write operations over the external bus. See Section 6.0 for more details.

Since the device execution is tied to instruction fetches, there is no need to execute faster than the fetch rate. So, if the program needs to be slowed, the processor speed must be slowed with a different TCY time.

6.0 TABLE READS/TABLE WRITES

PIC18C601/801 devices use two memory spaces: the external program memory space and the data memory space. Table Reads and Table Writes have been provided to move data between these two memory spaces through an 8-bit register (TABLAT).

The operations that allow the processor to move data between the data and external program memory spaces are:

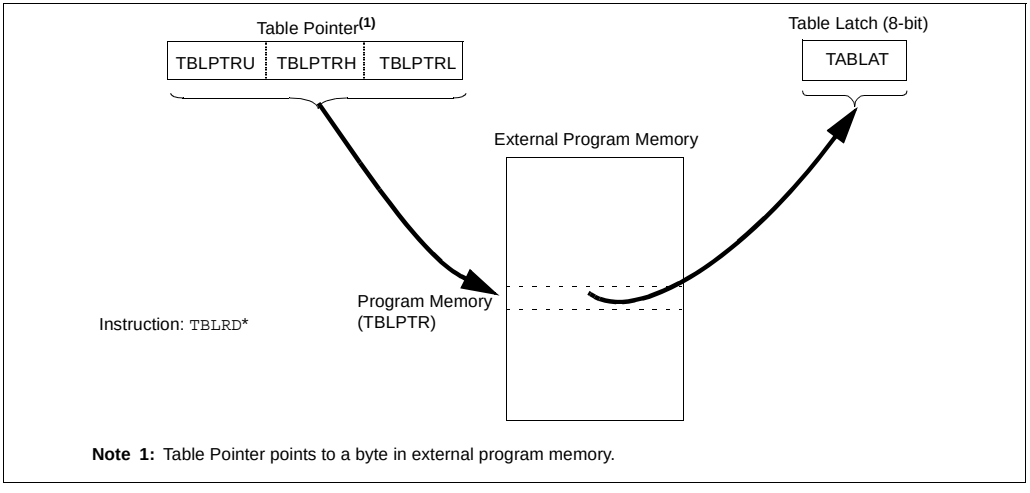
- Table Read (TBLRD)
- Table Write (TBLWT)

Table Read operations retrieve data from external program memory and place it into the data memory space. Figure 6-1 shows the operation of a Table Read with program and data memory.

Table Write operations store data from the data memory space into external program memory. Figure 6-2 shows the operation of a Table Write with external program and data memory.

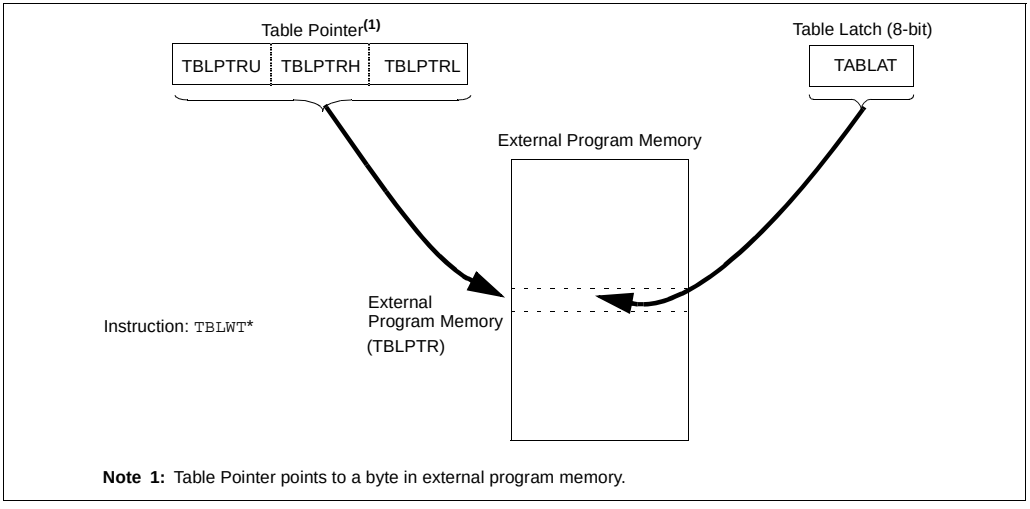
Table operations work with byte entities. A table block containing data is not required to be word aligned, so a table block can start and end at any byte address. If a Table Write is being used to write an executable program to program memory, program instructions must be word aligned.

FIGURE 6-1: TABLE READ OPERATION



Note 1: Table Pointer points to a byte in external program memory.

FIGURE 6-2: TABLE WRITE OPERATION



Note 1: Table Pointer points to a byte in external program memory.

6.1 Control Registers

Several control registers are used in conjunction with the TBLRD and TBLWT instructions. These include:

- TABLAT register
- TBLPTR registers

6.1.1 TABLAT - TABLE LATCH REGISTER

The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The Table Latch is used to hold 8-bit data during data transfers between program memory and data memory.

6.1.2 TBLPTR - TABLE POINTER REGISTER

The Table Pointer (TBLPTR) addresses a byte within the program memory. The TBLPTR is comprised of three SFR registers (Table Pointer Upper byte, High byte and Low byte). These three registers (TBLPTRU:TBLPTRH:TBLPTRL) join to form a 21-bit wide pointer. The 21-bits allow the device to address up to 2 Mbytes of program memory space.

The table pointer TBLPTR is used by the TBLRD and TBLWRT instructions. These instructions can update the TBLPTR in one of four ways, based on the table operation. These operations are shown in Table 6-1. These operations on the TBLPTR only affect the low order 21-bits.

TABLE 6-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

Example	Operation on Table Pointer
TBLRD* TBLWT*	TBLPTR is not modified
TBLRD*+ TBLWT*+	TBLPTR is incremented after the read/write
TBLRD*- TBLWT*-	TBLPTR is decremented after the read/write
TBLRD+* TBLWT+*	TBLPTR is incremented before the read/write

6.2 Table Read

The TBLRD instruction is used to retrieve data from external program memory and place it into data memory.

TBLPTR points to a byte address in external program memory space. Executing TBLRD places the byte into TABLAT. In addition, TBLPTR can be modified automatically for the next Table Read operation.

Table Reads from external program memory are performed one byte at a time. If the external interface is 8-bit, the bus interface circuitry in TABLAT will load the external value into TABLAT. If the external interface is 16-bit, interface circuitry in TABLAT will select either the high or low byte of the data from the 16-bit bus, based on the least significant bit of the address.

Example 6-1describes how to use TBLRD. Figure 6-3 and Figure 6-4 show Table Read timings for an 8-bit external interface, and Figure 6-5 describes Table Read timing for a 16-bit interface.

EXAMPLE 6-1: TABLE READ CODE EXAMPLE

```
; Read a byte from location 0020h
CLRF    TBLPTRU             ; clear upper 5 bits of TBLPTR
CLRF    TBLPTRH             ; clear higher 8 bits of TBLPTR
MOVLW   20h                 ; Load 20h into
MOVWF   TBLPTRL             ; TBLPTRL
TBLRD*                      ; Data is in TABLAT
```

FIGURE 6-3: TBLRD EXTERNAL INTERFACE TIMING (8-BIT MULTIPLEXED MODE)

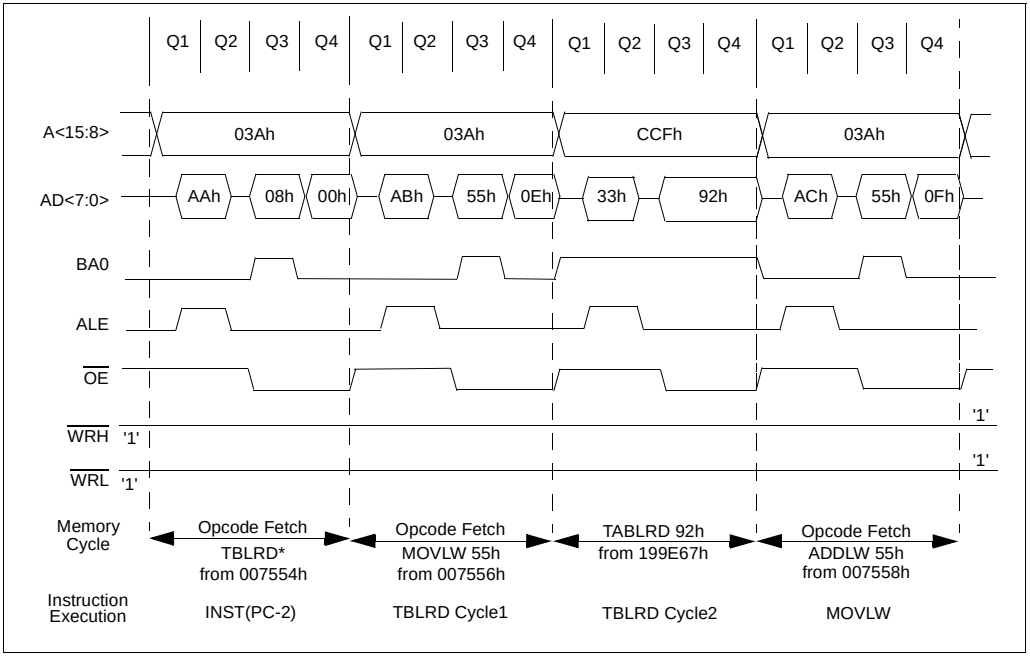


FIGURE 6-4: TBLRD EXTERNAL INTERFACE TIMING (8-BIT DE-MULTIPLEXED MODE)

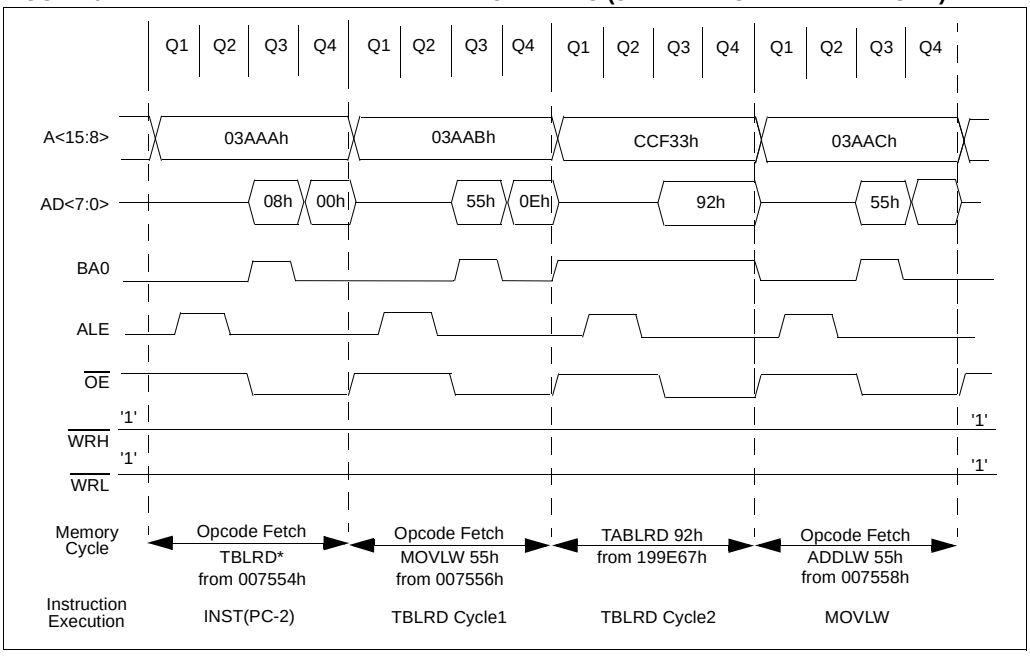
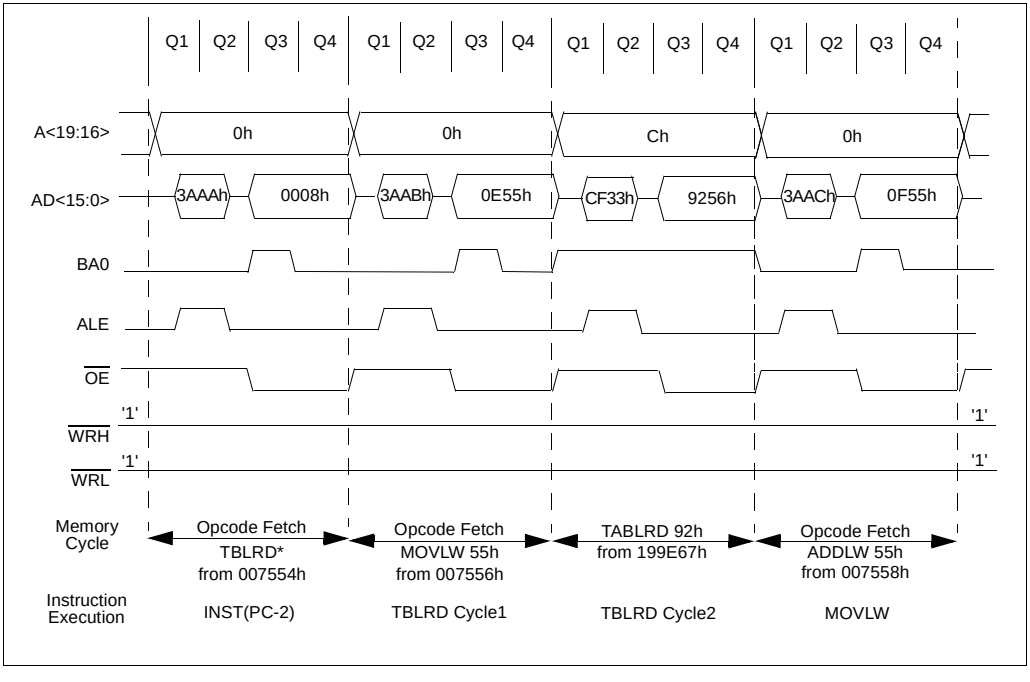


FIGURE 6-5: TBLRD EXTERNAL BUS TIMING (16-BIT MODE)



6.3 Table Write

Table Write operations store data from the data memory space into external program memory.

PIC18C601/801devices perform Table Writes one byte at a time. Table Writes to external memory are two-cycle instructions, unless wait states are enabled. The last cycle writes the data to the external memory location.

16-bit interface Table Writes depend on the type of external device that is connected and the WM<1:0> bits in the MEMCON register (See Figure 5-2).

Example 6-2 describes how to use TBLWT.

EXAMPLE 6-2: TABLE WRITE CODE EXAMPLE

```
; Write a byte to location 0020h
CLRF    TBLPTRU            ; clear upper 5 bits of TBLPTR
CLRF    TBLPTRH            ; clear higher 8 bits of TBLPTR
MOVLW   20h                ; Load 20h into
MOVWF   TBLPTRL            ; TBLPTRL
MOVLW   55h                ; Load 55h into
MOVWF   TBLAT              ; TBLAT
TBLWT*                     ; Write it
```

6.3.1 8-BIT EXTERNAL TABLE WRITES

When the external bus is 8-bit, the byte-wide Table Write exactly corresponds to the bus length and there are no special considerations required.

The $\overline{WRL}$ signal is used as the active write signal.

Figure 6-6 and Figure 6-7 show the timings associated with the 8-bit modes.

FIGURE 6-6: `TBLWT` EXTERNAL INTERACE TIMING (8-BIT MULTIPLEXED MODE)

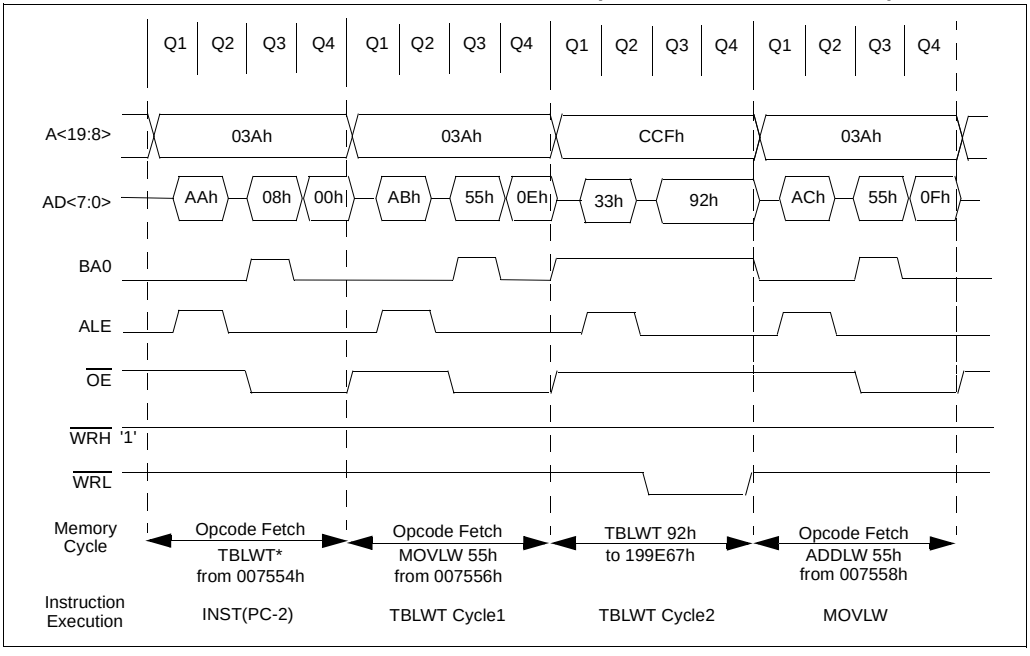
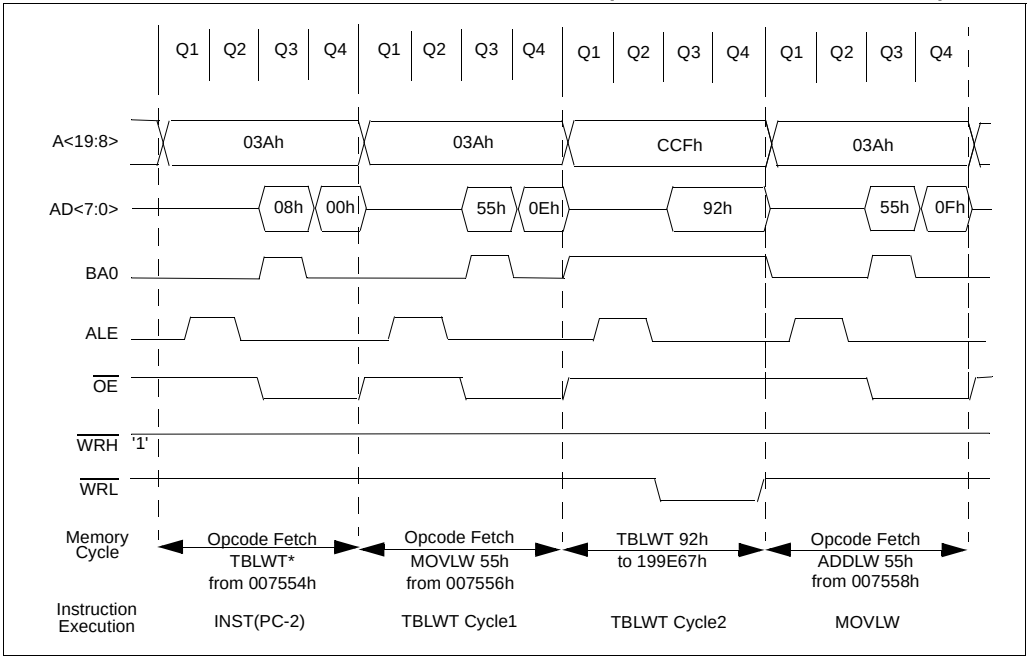


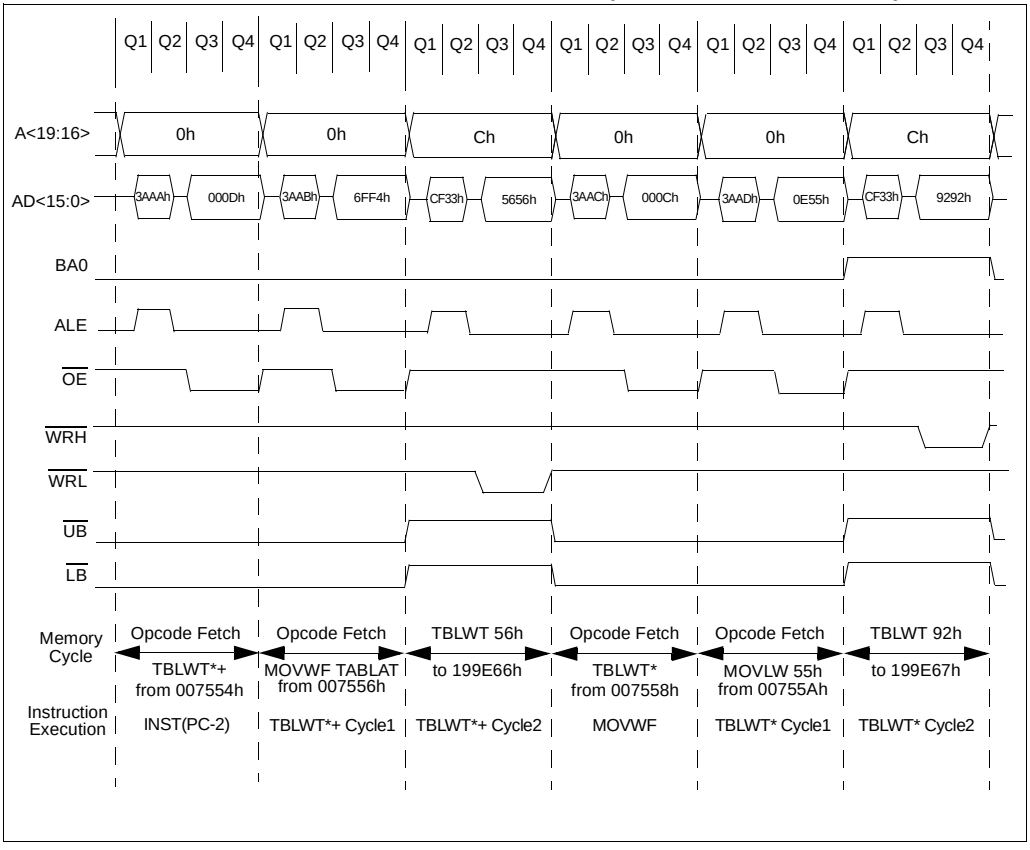
FIGURE 6-7: `TBLWT` EXTERNAL INTERFACE TIMING (8-BIT DE-MULTIPLEXED MODE)



6.3.2 16-BIT EXTERNAL TABLE WRITE (BYTE WRITE MODE)

This mode allows Table Writes to byte-wide external memories. During a TBLWT cycle, the TABLAT data is presented on the upper and lower byte of the AD<15:0> bus. The appropriate WRH or WRL line is strobed based on the LSb of the TBLPTR. Figure 6-8 shows the timing associated with this mode.

FIGURE 6-8: TBLWT EXTERNAL INTERFACE TIMING (16-BIT BYTE WRITE MODE)



6.3.3 EXTERNAL TABLE WRITE IN 16-BIT WORD WRITE MODE

This mode allows Table Writes to any type of word-wide external memories.

This method makes a distinction between TBLWT cycles to even or odd addresses.

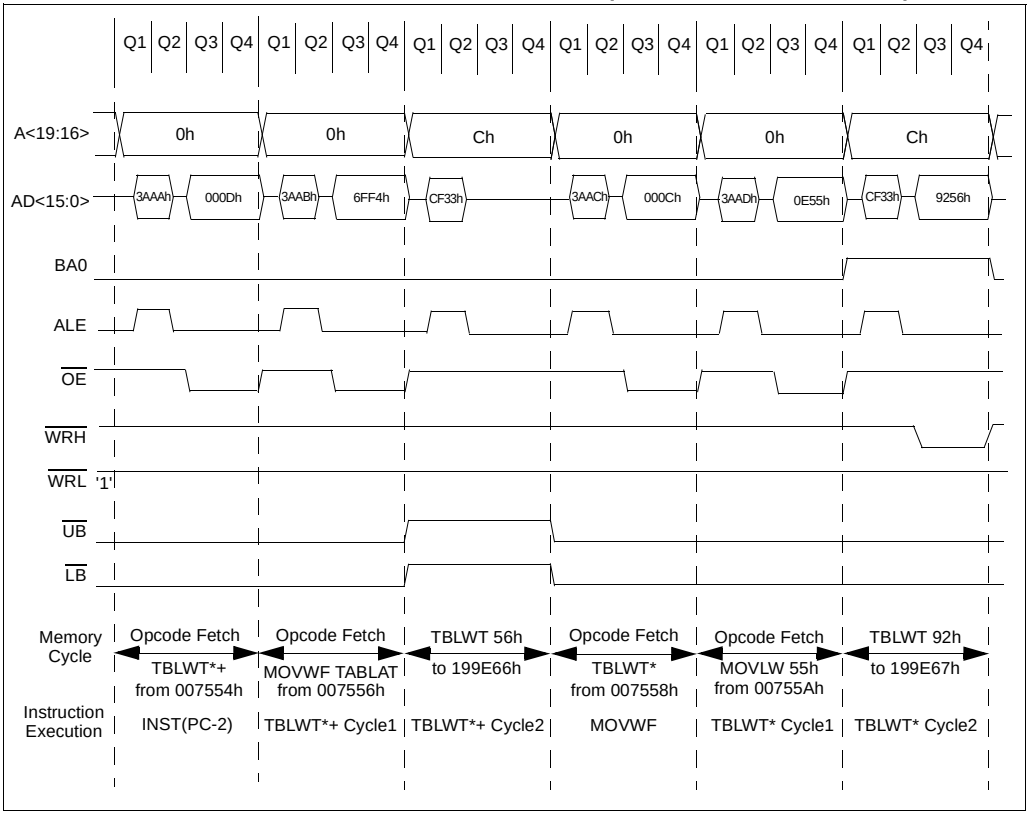
During a TBLWT cycle to an even address, where TBLPTR<0> = 0, the TABLAT data is transferred to a holding latch and the external address data bus is tri-stated for the data portion of the bus cycle. No write signals are activated.

During a TBLWT cycle to an odd address, where TBLPTR<0> = 1, the TABLAT data is presented on the upper byte of the AD<15:0> bus. The contents of the holding latch are presented on the lower byte of the AD<15:0> bus. The $\overline{WRH}$ line is strobed for each write cycle and the $\overline{WRL}$ line is unused. The BA0 line indicates the LSb of TBLPTR, but it is unnecessary. The $\overline{UB}$ and $\overline{LB}$ lines are active to select both bytes.

The obvious limitation to this method is that the TBLWT must be done in pairs on a specific word boundary to correctly write a word location.

Figure 6-9 shows the timing associated with this mode.

FIGURE 6-9: TBLWT EXTERNAL INTERFACE TIMING (16-BIT WORD WRITE MODE)



6.3.4 16-BIT EXTERNAL TABLE WRITE (BYTE SELECT MODE)

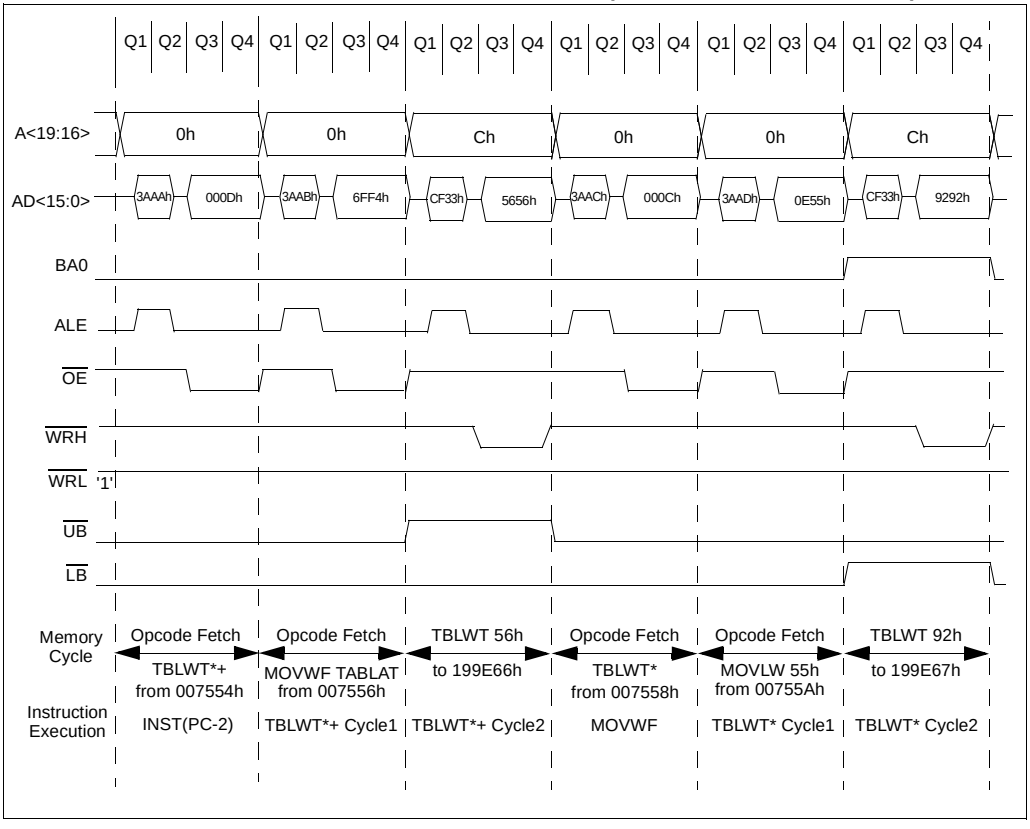
This mode allows Table Writes to word-wide external memories that have byte selection capabilities. This generally includes word-wide FLASH devices and word-wide static RAM devices.

During a TBLWT cycle, the TABLAT data is presented on the upper and lower byte of the AD<15:0> bus. The $\overline{WRH}$ line is strobed for each write cycle and the $\overline{WRL}$ line is unused. The BA0 or $\overline{UB}$ or $\overline{UL}$ lines are used to select the byte to be written, based on the LSb of the TBLPTR.

JEDEC standard flash memories will require a I/O port line to become a BYTE/WORD input signal and will use the BA0 signal as a byte address. JEDEC standard static RAM memories will use the $\overline{UB}$ or $\overline{UL}$ signals to select the byte.

Figure 6-10 shows the timing associated with this mode.

FIGURE 6-10: TBLWT EXTERNAL INTERFACE TIMING (16-BIT BYTE SELECT MODE)



6.4 Long Writes

Long writes will not be supported on the PIC18C601/801 to program FLASH configuration memory. The configuration locations can only be programmed in ICSP mode.

6.5 External Wait Cycles

The Table Reads and Writes have the capability to insert wait states when accessing external memory. These wait states only apply to the execution of a Table Read or Write to external memory and not to instruction fetches out of external memory. The guidelines presented in Section 5.0 must be followed to select the proper memory speed grade for the device operating frequency.

The WAIT<1:0> bits in the MEMCON register will select 0, 1, 2, or 3 extra TCY cycles per TBLRD/TBWLT cycle. The wait will occur on Q4.

The default setting of the wait on power-up is to assert a maximum wait of 3TCY cycles. This insures that slow memories will work in Microprocessor mode immediately after RESET.

Figure 6-11 shows 8-bit external bus timing for a Table Read with 2 wait cycles. Figure 6-12 shows 16-bit external bus timing for a Table Read with 1 wait cycle.

FIGURE 6-11: EXTERNAL INTERFACE TIMING (8-BIT MODE)

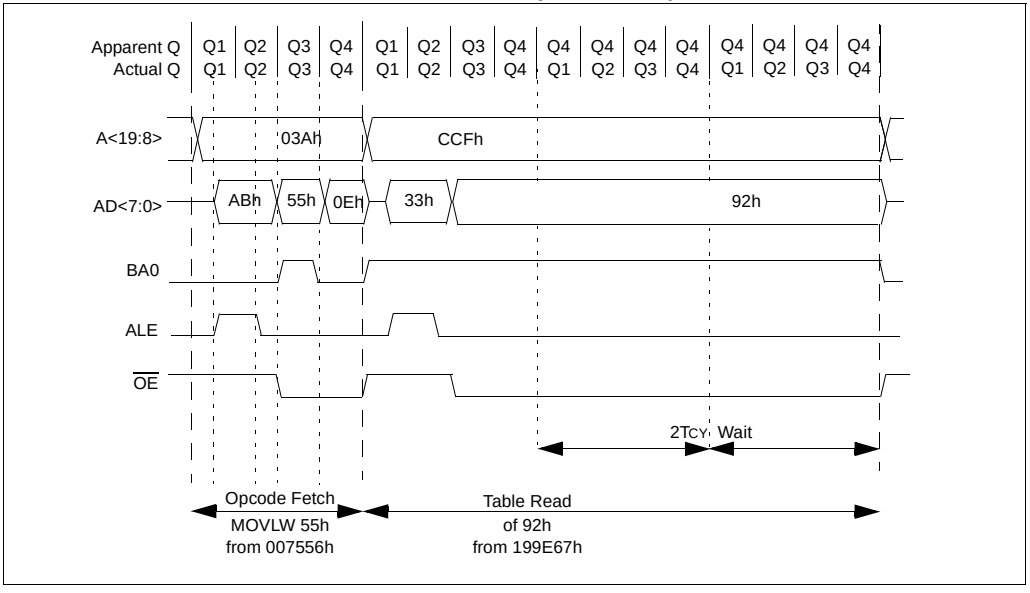
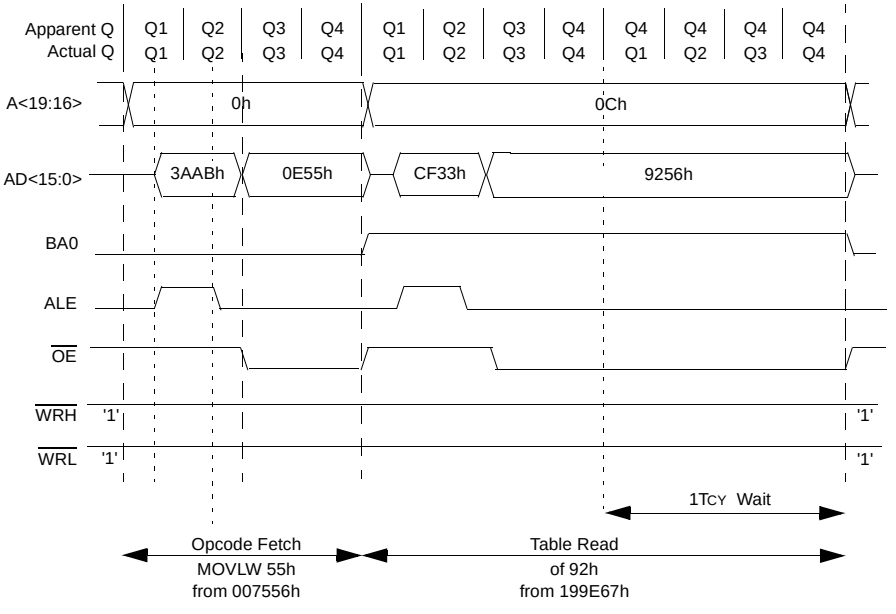


FIGURE 6-12: EXTERNAL INTERFACE TIMING (16-BIT MODE)



7.0 8 X 8 HARDWARE MULTIPLIER

An 8 x 8 hardware multiplier is included in the ALU of PIC18C601/801 devices. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit product register pair (PRODH:PRODL). The multiplier does not affect any flags in the STATUS register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

The performance increase allows the device to be used in some applications previously reserved for Digital Signal Processors.

Table 7-1 shows a performance comparison between enhanced devices using the single cycle hardware multiply, and performing the same function without the hardware multiply.

TABLE 7-1: PERFORMANCE COMPARISON

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 25 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	11.0 μs	27.6 μs	69.0 μs
	Hardware multiply	1	1	160.0 ns	400.0 ns	1.0 μs
8 x 8 signed	Without hardware multiply	33	91	14.6 μs	36.4 μs	91.0 μs
	Hardware multiply	6	6	960.0 ns	2.4 μs	6.0 μs
16 x 16 unsigned	Without hardware multiply	21	242	38.7 μs	96.8 μs	242.0 μs
	Hardware multiply	24	24	3.8 μs	9.6 μs	24.0 μs
16 x 16 signed	Without hardware multiply	52	254	40.6 μs	102.6 μs	254.0 μs
	Hardware multiply	36	36	5.8 μs	14.4 μs	36.0 μs

7.1 Operation

Example 7-1 shows the sequence to perform an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 7-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's most significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 7-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

```
MOVFF    ARG1, WREG  ;
MULWF    ARG2        ; ARG1 * ARG2 ->
                     ;   PRODH:PRODL
```

EXAMPLE 7-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVFF    ARG1, WREG
MULWF    ARG2            ; ARG1 * ARG2 ->
                         ; PRODH:PRODL
BTFSC    ARG2, SB        ; Test Sign Bit
SUBWF    PRODH           ; PRODH = PRODH
                         ;         - ARG1
MOVFF    ARG2, WREG
BTFSC    ARG1, SB        ; Test Sign Bit
SUBWF    PRODH           ; PRODH = PRODH
                         ;         - ARG2
```

Example 7-3 shows the sequence to perform a 16 x 16 unsigned multiply. Equation 7-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers RES3:RES0.

EQUATION 7-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

EXAMPLE 7-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
   MOVFF    ARG1L, WREG
   MULWF    ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
   MOVFF    PRODH, RES1     ;
   MOVFF    PRODL, RES0     ;
;
   MOVFF    ARG1H, WREG
   MULWF    ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
   MOVFF    PRODH, RES3     ;
   MOVFF    PRODL, RES2     ;
;
   MOVFF    ARG1L, WREG
   MULWF    ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
   MOVF     PRODL, W        ;
   ADDWF    RES1            ; Add cross
   MOVF     PRODH, W        ; products
   ADDWFC   RES2            ;
   CLRF     WREG            ;
   ADDWFC   RES3            ;
;
   MOVFF    ARG1H, WREG     ;
   MULWF    ARG2L           ; ARG1H * ARG2L ->
                            ; PRODH:PRODL
   MOVF     PRODL, W        ;
   ADDWF    RES1            ; Add cross
   MOVF     PRODH, W        ; products
   ADDWFC   RES2            ;
   CLRF     WREG            ;
   ADDWFC   RES3            ;
```

Example 7-4 shows the sequence to perform a 16 x 16 signed multiply. Equation 7-2 shows the algorithm used. The 32-bit result is stored in four registers, RES3:RES0. To account for the sign bits of the arguments, each argument pairs' most significant bit (MSb) is tested and the appropriate subtractions are done.

EQUATION 7-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

EXAMPLE 7-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
   MOVFF    ARG1L, WREG
   MULWF    ARG2L          ; ARG1L * ARG2L ->
                           ;   PRODH:PRODL
   MOVFF    PRODH, RES1    ;
   MOVFF    PRODL, RES0    ;
;
   MOVFF    ARG1H, WREG
   MULWF    ARG2H          ; ARG1H * ARG2H ->
                           ;   PRODH:PRODL
   MOVFF    PRODH, RES3    ;
   MOVFF    PRODL, RES2    ;
;
   MOVFF    ARG1L, WREG
   MULWF    ARG2H          ; ARG1L * ARG2H ->
                           ;   PRODH:PRODL
   MOVF     PRODL, W       ;
   ADDWF    RES1           ; Add cross
   MOVF     PRODH, W       ;    products
   ADDWFC   RES2           ;
   CLRF     WREG           ;
   ADDWFC   RES3           ;
;
   MOVFF    ARG1H, WREG    ;
   MULWF    ARG2L          ; ARG1H * ARG2L ->
                           ;   PRODH:PRODL
   MOVF     PRODL, W       ;
   ADDWF    RES1           ; Add cross
   MOVF     PRODH, W       ;    products
   ADDWFC   RES2           ;
   CLRF     WREG           ;
   ADDWFC   RES3           ;
;
   BTFSS    ARG2H, 7       ; ARG2H:ARG2L neg?
   GOTO     SIGN_ARG1      ; no, check ARG1
   MOVFF    ARG1L, WREG    ;
   SUBWF    RES2           ;
   MOVFF    ARG1H, WREG    ;
   SUBWFB   RES3
;
SIGN_ARG1
   BTFSS    ARG1H, 7       ; ARG1H:ARG1L neg?
   GOTO     CONT_CODE      ; no, done
   MOVFF    ARG2L, WREG    ;
   SUBWF    RES2           ;
   MOVFF    ARG2H, WREG    ;
   SUBWFB   RES3
;
CONT_CODE
     :
```

NOTES:

8.0 INTERRUPTS

PIC18C601/801 devices have 15 interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level, or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will override any low priority interrupts that may be in progress.

There are 10 registers that are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

Each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON register). When interrupt priority is enabled, there are two bits that enable interrupts globally. Setting the GIEH bit (INTCON register) enables all interrupts that have the priority bit set. Setting the GIEL bit (INTCON register) enables all interrupts that have the priority bit cleared. When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority level. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. The PEIE bit (INTCON register) enables/disables all peripheral interrupt sources. The GIE bit (INTCON register) enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

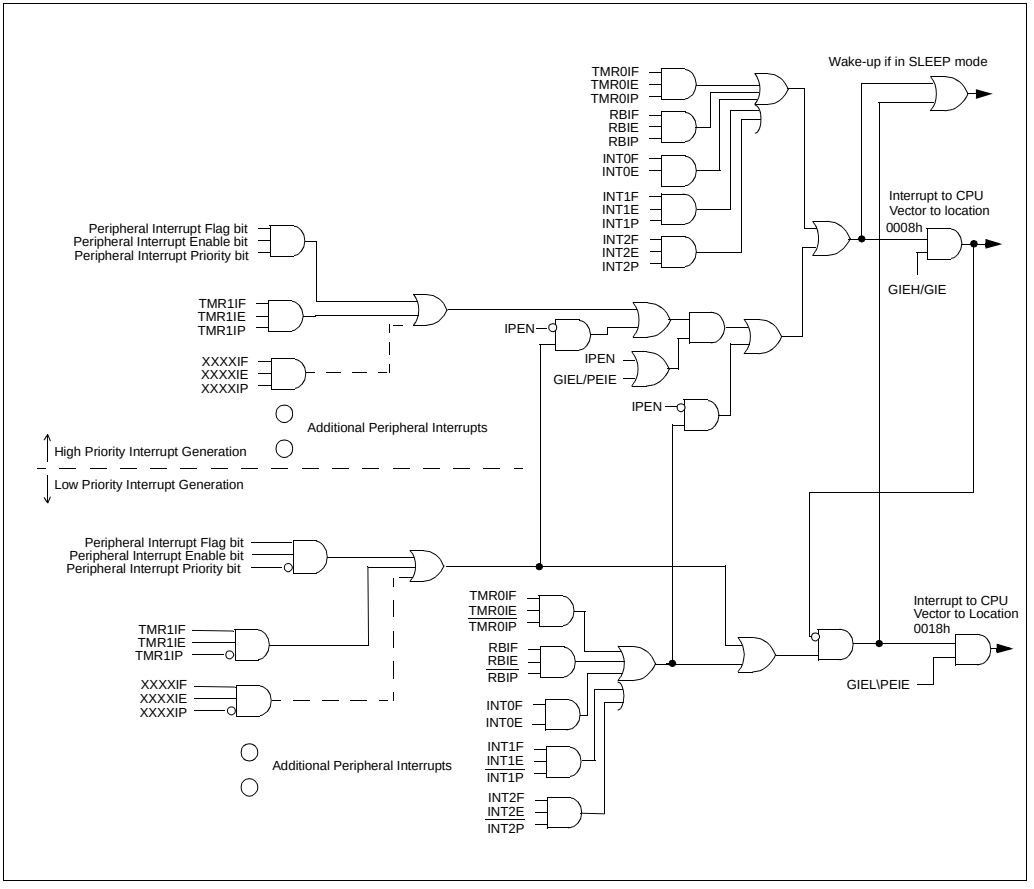
When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts, to avoid recursive interrupts.

The "return from interrupt" instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding enable bit or the GIE bit.

FIGURE 8-1: INTERRUPT LOGIC



8.1 Control Registers

This section contains the control and status registers.

8.1.1 INTCON REGISTERS

The INTCON Registers are readable and writable registers, which contain various enable, priority, and flag bits.

REGISTER 8-1: INTCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all high priority interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows software polling.

REGISTER 8-2: INTCON2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0**: External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1**: External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2**: External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

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

bit 2 **TMR0IP**: TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 0 **RBIP**: RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows software polling.

REGISTER 8-3: INTCON3 REGISTER

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
bit 7							bit 0

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

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

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows software polling.

8.1.2 PIR REGISTERS

The Peripheral Interrupt Request (PIR) registers contain the individual flag bits for the peripheral interrupts (Register 8-5). There are two Peripheral Interrupt Request (Flag) registers (PIR1, PIR2).

Note 1: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit, GIE (INTCON register).

2: User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt, and after servicing that interrupt.

8.1.3 PIE REGISTERS

The Peripheral Interrupt Enable (PIE) registers contain the individual enable bits for the peripheral interrupts (Register 8-6). There are two two Peripheral Interrupt Enable registers (PIE1, PIE2). When IPEN is clear, the PEIE bit must be set to enable any of these peripheral interrupts.

8.1.4 IPR REGISTERS

The Interrupt Priority (IPR) registers contain the individual priority bits for the peripheral interrupts (Register 8-9). There are two Peripheral Interrupt Priority registers (IPR1, IPR2). The operation of the priority bits requires that the Interrupt Priority Enable bit (IPEN) be set.

8.1.5 RCON REGISTER

The Reset Control (RCON) register contains the bit that is used to enable prioritized interrupts (IPEN).

REGISTER 8-4: RCON REGISTER

R/W-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-0	U-0
IPEN	r	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	r
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (16CXXX compatibility mode)

bit 6 **Reserved:** Maintain as '0'

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
For details of bit operation, see Register 4-4

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
For details of bit operation, see Register 4-4

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
For details of bit operation, see Register 4-4

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
For details of bit operation, see Register 4-4

bit 0 **Reserved:** Maintain as '0'

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 8-5: PIR1 REGISTER

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

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

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
(must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer, RCREG, is full
(cleared when RCREG is read)
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer, TXREG, is empty
(cleared when TXREG is written)
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete
(must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred
(must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred
(must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred
(must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed
(must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 8-6: PIR2 REGISTER

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as'0'

bit 3 **BCLIF:** Bus Collision Interrupt Flag bit

1 = A bus collision occurred
(must be cleared in software)

0 = No bus collision occurred

bit 2 **LVDIF:** Low Voltage Detect Interrupt Flag bit

1 = A low voltage condition occurred
(must be cleared in software)

0 = The device voltage is above the Low Voltage Detect trip point

bit 1 **TMR3IF:** TMR3 Overflow Interrupt Flag bit

1 = TMR3 register overflowed
(must be cleared in software)

0 = TMR3 register did not overflow

bit 0 **CCP2IF:** CCPx Interrupt Flag bit

Capture mode:

1 = A TMR1 register capture occurred
(must be cleared in software)

0 = No TMR1 register capture occurred

Compare mode:

1 = A TMR1 register compare match occurred
(must be cleared in software)

0 = No TMR1 register compare match occurred

PWM mode:

Unused in this mode

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 8-7: PIE1 REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

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

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

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

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 8-8: PIE2 REGISTER

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **BCLIE**: Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **LVDIE**: Low Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3IE**: TMR3 Overflow Interrupt Enable bit
1 = Enables the TMR3 overflow interrupt
0 = Disables the TMR3 overflow interrupt

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 8-9: IPR1 REGISTER

U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

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

bit 6 **ADIP**: A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP**: USART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP**: USART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP**: Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP**: CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP**: TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP**: TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 8-10: IPR2 REGISTER

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **BCLIP**: Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **LVDIP**: Low Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP**: TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP**: CCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

8.1.6 INT INTERRUPTS

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge triggered: either rising, if the corresponding INTEDGx bit is set in the INTCON2 register, or falling, if the INTEDGx bit is clear. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit INTxIF is set. This interrupt can be disabled by clearing the corresponding enable bit INTxIE. Flag bit INTxIF must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt. All external interrupts (INT0, INT1 and INT2) can wake-up the processor from SLEEP, if bit INTxIE was set prior to going into SLEEP. If the global interrupt enable bit GIE is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1 and INT2 is determined by the value contained in the interrupt priority bits INT1IP (INTCON3 register) and INT2IP (INTCON3 register). There is no priority bit associated with INT0; it is always a high priority interrupt source.

8.1.7 TMR0 INTERRUPT

In 8-bit mode (which is the default), an overflow (0FFh $\rightarrow$ 00h) in the TMR0 register will set flag bit TMR0IF. In 16-bit mode, an overflow (0FFFFh $\rightarrow$ 0000h) in the TMR0H:TMR0L registers will set flag bit TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit TMR0IE (INTCON register). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit TMR0IP (INTCON2 register). See Section 10.0 for further details on the Timer0 module.

8.1.8 PORTB INTERRUPT-ON-CHANGE

An input change on PORTB<7:4> sets flag bit RBIF (INTCON register). The interrupt can be enabled/disabled by setting/clearing enable bit RBIE (INTCON register). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit RBIP (INTCON2 register).

8.2 Context Saving During Interrupts

During an interrupt, the return PC value is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (See Section 4.3), the user may need to save the WREG, STATUS and BSR registers in software. Depending on the user's application, other registers may also need to be saved. Example 8-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 8-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```
MOVWF    W_TEMP                    ; W_TEMP is in Low Access bank
MOVFF    STATUS, STATUS_TEMP       ; STATUS_TEMP located anywhere
MOVFF    BSR, BSR_TEMP             ; BSR located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR             ; Restore BSR
MOVF     W_TEMP, W                 ; Restore WREG
MOVFF    STATUS_TEMP, STATUS       ; Restore STATUS
```

NOTES:

9.0 I/O PORTS

Depending on the device selected, there are up to 9 ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

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

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The data latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving.

9.1 PORTA, TRISA and LATA Registers

PORTA is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin). On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch.

Read-modify-write operations on the LATA register, reads and writes the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

The other PORTA pins are multiplexed with analog inputs and the analog VREF+ and VREF- inputs. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1). On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

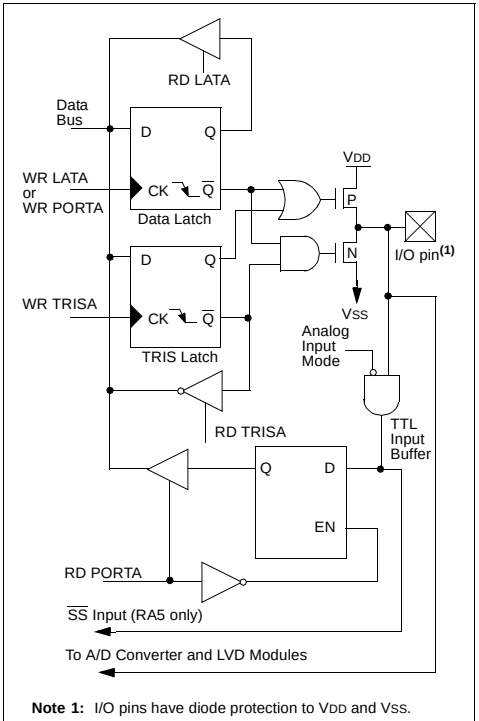
The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

Note:	On a Power-on Reset, PORTA pins RA3:RA0 and RA5 default to analog inputs.

EXAMPLE 9-1: INITIALIZING PORTA

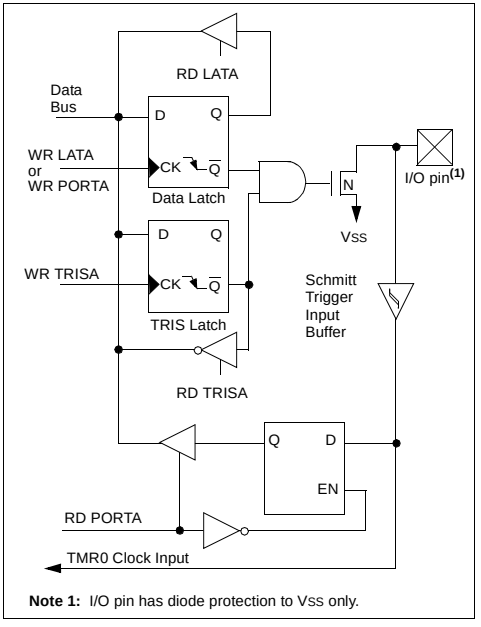
```
CLRF   PORTA    ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF   LATA     ; Alternate method
                ; to clear output
                ; data latches
MOVLW  07h      ; Configure A/D
MOVWF  ADCON1   ; for digital inputs
MOVLW  0CFh     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISA    ; Set RA3:RA0 as inputs
                ; RA5:RA4 as outputs
```

FIGURE 9-1: RA3:RA0 AND RA5 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 9-2: RA4/T0CKI PIN BLOCK DIAGRAM



Note 1: I/O pin has diode protection to VSS only.

TABLE 9-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input
RA1/AN1	bit1	TTL	Input/output or analog input
RA2/AN2/VREF-	bit2	TTL	Input/output or analog input or VREF-
RA3/AN3/VREF+	bit3	TTL	Input/output or analog input or VREF+
RA4/T0CKI	bit4	ST/OD	Input/output or external clock input for Timer0, output is open drain type
RA5/$\overline{SS}$/AN4/LVDIN	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input or low voltage detect input

Legend: TTL = TTL input, ST = Schmitt Trigger input, OD = Open Drain

TABLE 9-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

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
PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--uu uuuu
LATA	—	Latch A Data Output Register							-xxx xxxx	-uuu uuuu
TRISA	—	PORTA Data Direction Register							-111 1111	-111 1111
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--uu uuuu

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

9.2 PORTB, TRISB and LATB Registers

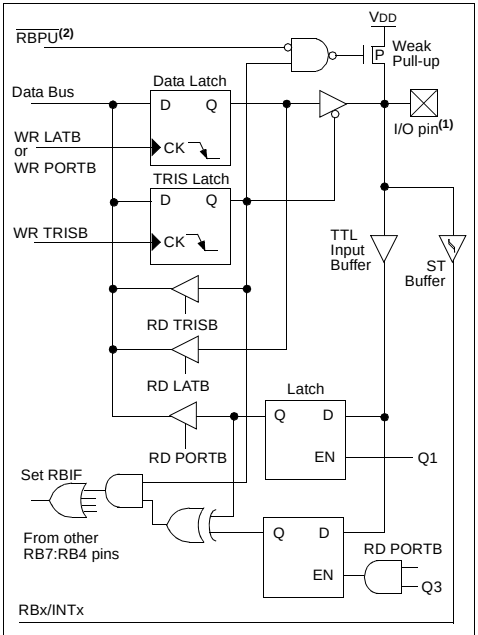
PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 9-2: INITIALIZING PORTB

```
CLRF    PORTB     ; Initialize PORTB by
                  ; clearing output
                  ; data latches
CLRF    LATB      ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW   0CFh      ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISB     ; Set RB3:RB0 as inputs
                  ; RB5:RB4 as outputs
                  ; RB7:RB6 as inputs
```

FIGURE 9-3: RB7:RB4 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2 register).

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (INTCON2 register). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Pin RB3 is multiplexed with the CCP input/output. The weak pull-up for RB3 is disabled when the RB3 pin is configured as CCP pin. By disabling the weak pull-up when pin is configured as CCP, allows the remaining weak pull-up devices of PORTB to be used while the CCP is being used.

Four of PORTB's pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'd together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON register).

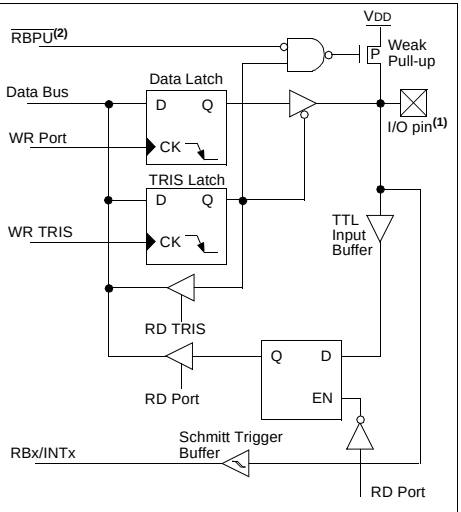
This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the `MOVFF` instruction). This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

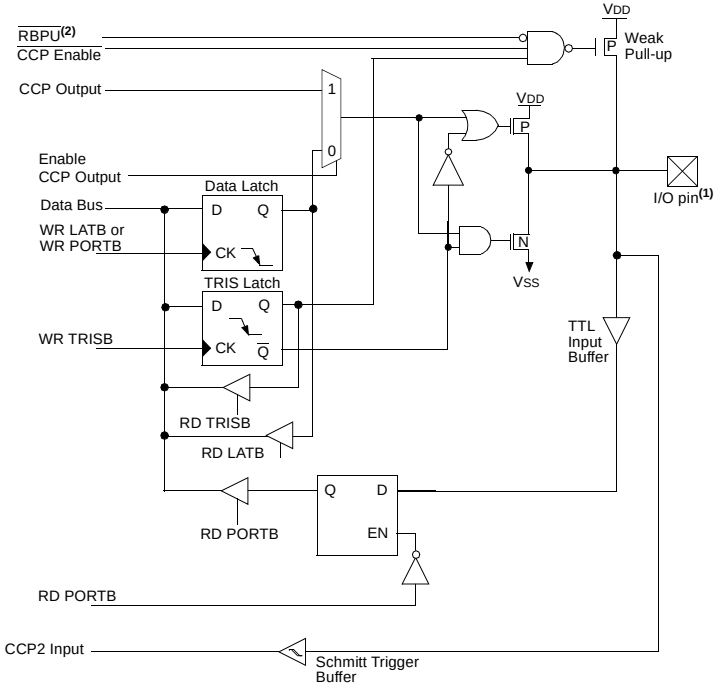
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

FIGURE 9-4: RB2:RB0 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2 register).

FIGURE 9-5: RB3 PIN BLOCK DIAGRAM



Note 1: I/O pin has diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).

TABLE 9-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT0	bit0	TTL/ST(1)	Input/output pin or external interrupt 0 input. Internal software programmable weak pull-up.
RB1/INT1	bit1	TTL/ST(1)	Input/output pin or external interrupt 1 input. Internal software programmable weak pull-up.
RB2/INT2	bit2	TTL/ST(1)	Input/output pin or external interrupt 2 input. Internal software programmable weak pull-up.
RB3/CCP2	bit3	TTL/ST(3)	Input/output pin or Capture2 input or Capture2 output or PWM2 output. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6	bit6	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Note 1: This pin is a Schmitt Trigger input when configured as the external interrupt.

2: This pin is a Schmitt Trigger input when used in Serial Programming mode.

3: This pin is a Schmitt Trigger input when used in a Capture input.

TABLE 9-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

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
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
LATB	LATB Data Output Register								xxxx xxxx	uuuu uuuu
TRISB	PORTB Data Direction Register								1111 1111	1111 1111
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	—	RBIP	1111 1111	1111 1111
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	1100 0000	1100 0000

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTD.

9.3 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATC register, read and write the latched output value for PORTC.

PORTC is multiplexed with several peripheral functions (Table 9-5). PORTC pins have Schmitt Trigger input buffers.

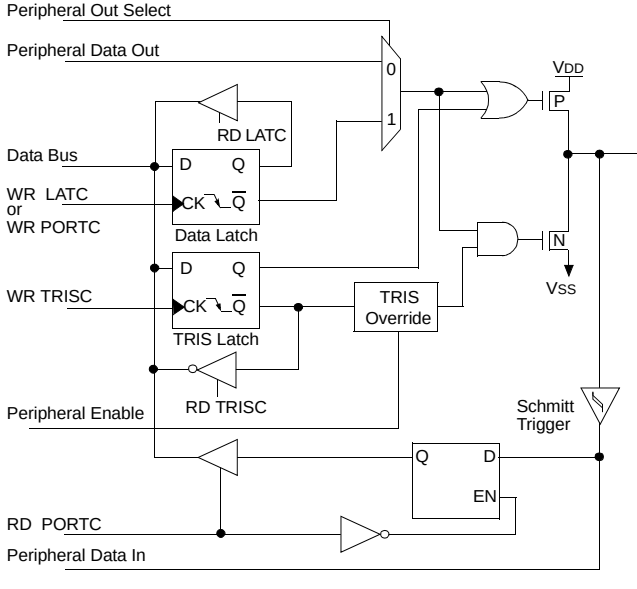
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register, without concern due to peripheral overrides.

EXAMPLE 9-3: INITIALIZING PORTC

```
CLRF    PORTC      ; Initialize PORTC by
                   ; clearing output
                   ; data latches
CLRF    LATC       ; Alternate method
                   ; to clear output
                   ; data latches
MOVLW   0CFh       ; Value used to
                   ; initialize data
                   ; direction
MOVWF   TRISC      ; Set RC3:RC0 as inputs
                   ; RC5:RC4 as outputs
                   ; RC7:RC6 as inputs
```

FIGURE 9-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE)



TRIS OVERRIDE

Pin	Override	Peripheral
RC0	Yes	Timer1 OSC for Timer1/Timer3
RC1	Yes	Timer1 OSC for Timer1/Timer3
RC2	No	—
RC3	Yes	SPI/I^{2}C Master Clock
RC4	Yes	I^{2}C Data Out
RC5	Yes	SPI Data Out
RC6	Yes	USART Async Xmit, Sync Clock
RC7	Yes	USART Sync Data Out

Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 9-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T13CKI	bit0	ST	Input/output port pin or Timer1 oscillator output or Timer1/Timer3 clock input.
RC1/T1OSI	bit1	ST	Input/output port pin, Timer1 oscillator input.
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit3	ST	Input/output port pin or synchronous serial clock for SPI/I^2C.
RC4/SDI/SDA	bit4	ST	Input/output port pin or SPI Data in (SPI mode) or Data I/O (I^2C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port Data output.
RC6/TX/CK	bit6	ST	Input/output port pin, Addressable USART Asynchronous Transmit, or Addressable USART Synchronous Clock.
RC7/RX/DT	bit7	ST	Input/output port pin, Addressable USART Asynchronous Receive, or Addressable USART Synchronous Data.

Legend: ST = Schmitt Trigger input

TABLE 9-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

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
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
LATC	LATC Data Output Register								xxxx xxxx	uuuu uuuu
TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

9.4 PORTD, TRISD and LATD Registers

PORTD is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISD. Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISD bit (= 0) will make the corresponding PORTD pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATD register reads and writes the latched output value for PORTD.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

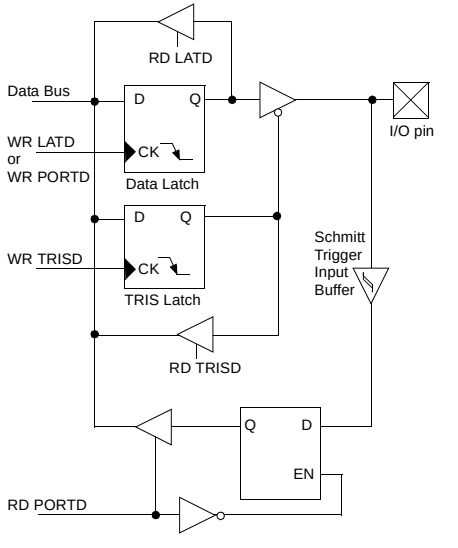
PORTD is multiplexed with the system bus and is available only when the system bus is disabled, by setting EBIDS bit in register MEMCON. When operating as the system bus, PORTD is the low order byte of the address/data bus (AD7:AD0), or as the low order address byte (A15:A8) if the address and data buses are de-multiplexed.

Note:	On a Power-on Reset, PORTD defaults to the system bus.

EXAMPLE 9-4: INITIALIZING PORTD

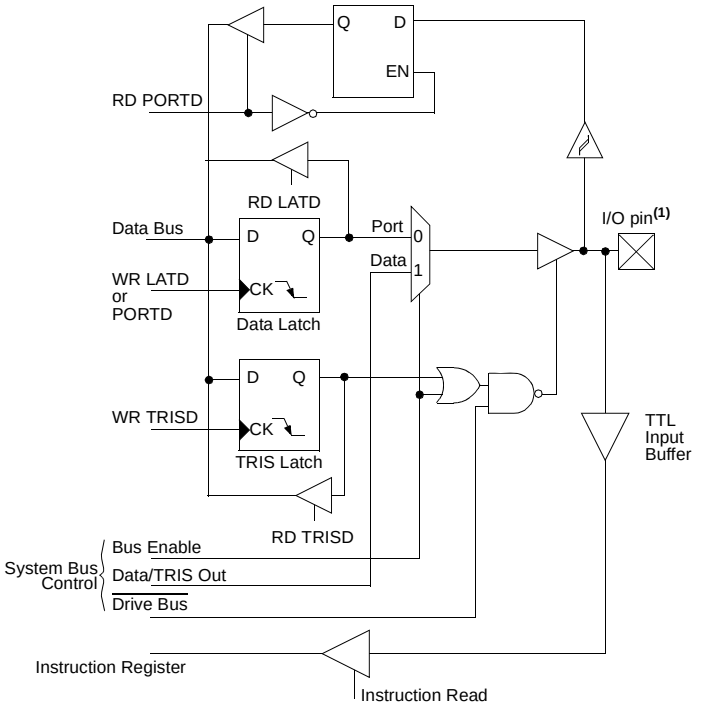
```
CLRF    PORTD      ; Initialize PORTD by
                   ; clearing output
                   ; data latches
CLRF    LATD       ; Alternate method
                   ; to clear output
                   ; data latches
MOVLW   0CFh       ; Value used to
                   ; initialize data
                   ; direction
MOVWF   TRISD      ; Set RD3:RD0 as inputs
                   ; RD5:RD4 as outputs
                   ; RD7:RD6 as inputs
```

FIGURE 9-7: PORTD BLOCK DIAGRAM IN I/O MODE



Note: I/O pins have diode protection to VDD and VSS.

FIGURE 9-8: PORTD BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 9-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/AD0/A0(2)	bit0	ST/TTL(1)	Input/output port pin or system bus bit 0
RD1/AD1/A1(2)	bit1	ST/TTL(1)	Input/output port pin or system bus bit 1
RD2/AD2/A2(2)	bit2	ST/TTL(1)	Input/output port pin or system bus bit 2
RD3/AD3/A3(2)	bit3	ST/TTL(1)	Input/output port pin or system bus bit 3
RD4/AD4/A4(3)	bit4	ST/TTL(1)	Input/output port pin or system bus bit 4
RD5/AD5/A5(2)	bit5	ST/TTL(1)	Input/output port pin or system bus bit 5
RD6/AD6/A6(2)	bit6	ST/TTL(1)	Input/output port pin or system bus bit 6
RD7/AD7/A7(2)	bit7	ST/TTL(1)	Input/output port pin or system bus bit 7

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in System Bus mode.

2: RDx is used as a multiplexed address/data bus for PIC18C601 and PIC18C801 in 16-bit mode, and as an address only for PIC18C801 in 8-bit mode.

TABLE 9-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

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
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
LATD	LATD Data Output Register								xxxx xxxx	uuuu uuuu
TRISD	PORTD Data Direction Register								1111 1111	1111 1111
MEMCON	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

9.5 PORTE, TRISE and LATE Registers

PORTE is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATE register reads and writes the latched output value for PORTE.

PORTE is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORTE is multiplexed with several peripheral functions (Table 9-9).

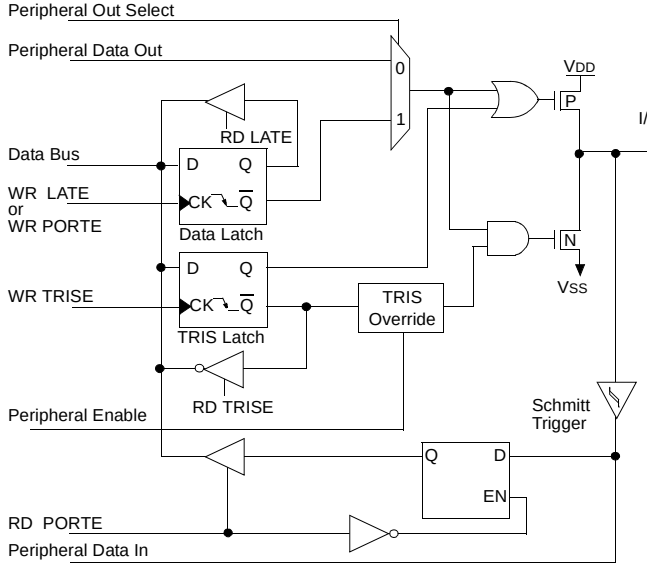
PORTE is multiplexed with the system bus and is available only when the system bus is disabled, by setting EBDIS bit in register MEMCON. When operating as the system bus, PORTE is configured as the high order byte of the address/data bus (AD15:AD8), or as the high order address byte (A15:A8), if address and data buses are de-multiplexed.

Note:	On Power-on Reset, PORTE defaults to the system bus.

EXAMPLE 9-5: INITIALIZING PORTE

```
CLRF    PORTE     ; Initialize PORTE by
                  ; clearing output
                  ; data latches
CLRF    LATE      ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW   03h       ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISE     ; Set RE1:RE0 as inputs
                  ; RE7:RE2 as outputs
```

FIGURE 9-9: PORTE BLOCK DIAGRAM IN I/O MODE

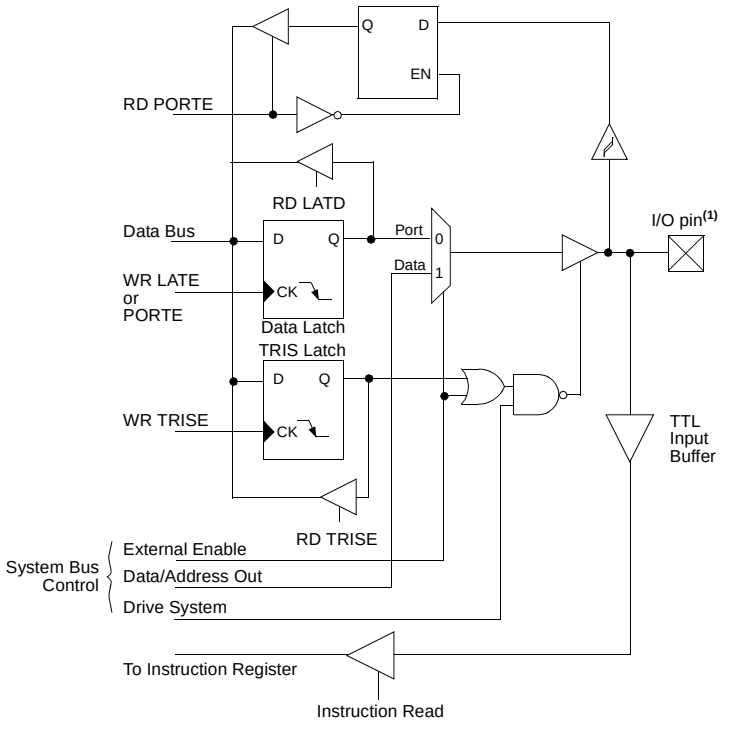


TRIS OVERRIDE

Pin	Override	Peripheral
RE0	Yes	External Bus
RE1	Yes	External Bus
RE2	Yes	External Bus
RE3	Yes	External Bus
RE4	Yes	External Bus
RE5	Yes	External Bus
RE6	Yes	External Bus
RE7	Yes	External Bus

Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 9-10: PORTE BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 9-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/AD8/A8(2)	bit0	ST/TTL(1)	Input/output port pin or Address/Data bit 8
RE1/AD9/A9(2)	bit1	ST/TTL(1)	Input/output port pin or Address/Data bit 9
RE2/AD10/A10(2)	bit2	ST/TTL(1)	Input/output port pin or Address/Data bit 10
RE3/AD11/A11(2)	bit3	ST/TTL(1)	Input/output port pin or Address/Data bit 11
RE4/AD12/A12(2)	bit4	ST/TTL(1)	Input/output port pin or Address/Data bit 12
RE5/AD13/A13(2)	bit5	ST/TTL(1)	Input/output port pin or Address/Data bit 13
RE6/AD14/A14(2)	bit6	ST/TTL(1)	Input/output port pin or Address/Data bit 14
RE7/AD15/A15(2)	bit7	ST/TTL(1)	Input/output port pin or Address/Data bit 15

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in System Bus mode.

2: REx is used as a multiplexed address/data bus for PIC18C601 and PIC18C801 in 16-bit mode, and as an address only for PIC18C801 in 8-bit mode.

TABLE 9-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

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
TRISE	PORTE Data Direction Control Register								1111 1111	1111 1111
PORTE	Read PORTE pin/Write PORTE Data Latch								xxxx xxxx	uuuu uuuu
LATE	Read PORTE Data Latch/Write PORTE Data Latch								xxxx xxxx	uuuu uuuu
MEMCON	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTE.

9.6 PORTF, LATF, and TRISF Registers

PORTF is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISF. Setting a TRISF bit (= 1) will make the corresponding PORTF pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISF bit (= 0) will make the corresponding PORTF pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATF register reads and writes the latched output value for PORTF.

PORTF pins, RF2:RF0, are multiplexed with analog inputs. The operation of these pins are selected by ADCON0 and ADCON1 registers.

PORTF pins, RF3 and RF5, are multiplexed with two of the integrated chip select signals $\overline{CSIO}$ and $\overline{CS1}$. For PIC18C801, pin RF4 is multiplexed with chip select signal $\overline{CS2}$, while for PIC18C601, it is multiplexed with system bus signal A16. For PIC18C801 devices, both CSEL2 and CSELIO registers must set to all zero, to enable these pins as I/O pins, while for PIC18C601 devices, only CSELIO register needs to be set to zero. For PIC18C601 devices, pin RF4 can only be configured as I/O when the EBDIS bit is set and execution is taking place in internal Boot RAM.

PORTF pins, RF7:RF6, are multiplexed with the system bus control signal $\overline{UB}$ and $\overline{LB}$, respectively, when a device with 16-bit bus execution is used. These pins can be configured as I/O pins by setting WM bits in the MEMCON register to any value other than '01'.

Note 1: On Power-on Reset, PORTF pins RF2:RF0 default to A/D inputs.

2: On Power-on Reset, PORTF pins RF7:RF3 for PIC18C801 and pins RF7:RF5, RF3 for PIC18C601, default to system bus signals.

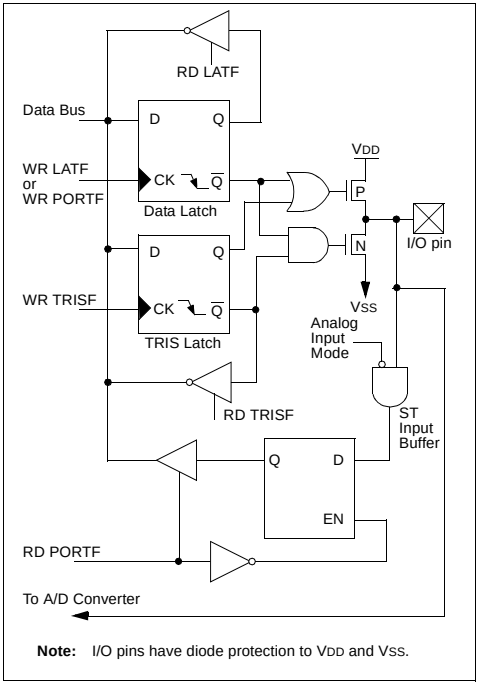
EXAMPLE 9-6: INITIALIZING PORTF

```
CLRF    PORTF       ; Initialize PORTF by
                    ; clearing output
                    ; data latches
CLRF    LATF        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   0Fh         ;
MOVWF   ADCON1      ; Set PORTF as digital I/O
MOVLW   0CFh        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISF       ; Set RF3:RF0 as inputs
                    ; RF5:RF4 as outputs
                    ; RF7:RF6 as inputs
```

EXAMPLE 9-7: PROGRAMMING CHIP SELECT SIGNALS

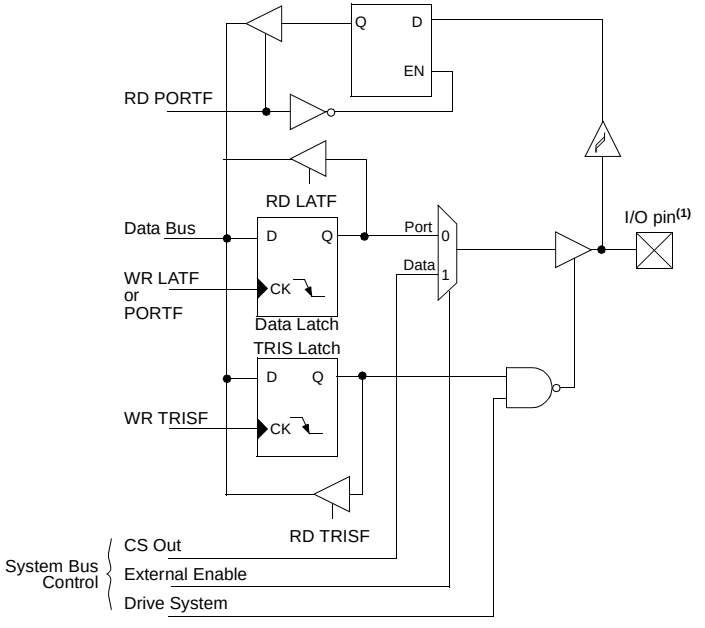
```
        •
        •
    ; Program chip select to activate CS1
    ; for all address less than 03FFFFh,
    ; while activate CS2 for rests of the
    ; addresses
    ; CSEL2 register is secured register.
    ; Before it can be modified it,
    ; combination lock must be opened
    MOVLW 20h           ; Preload WREG with
                        ; correct CSEL2 valu
    BCF INTCON, GIE     ; Disable interrupts
    CALL UNLOCK         ; Now unlock it
    ; Lock is open.  Modify CSEL2...
    MOVWF CSEL2
    ; Lock is closed
    BSF INTCON, GIE     ; Re-enable interrupts
    ; Chip select is programmed.
        •
        •

UNLOCK
    BSF PSPCON, CMLK1
    BSF PSPCON, CMLK0
    RETURN
        •
        •
```

FIGURE 9-11: RF2:RF0 PINS BLOCK DIAGRAM



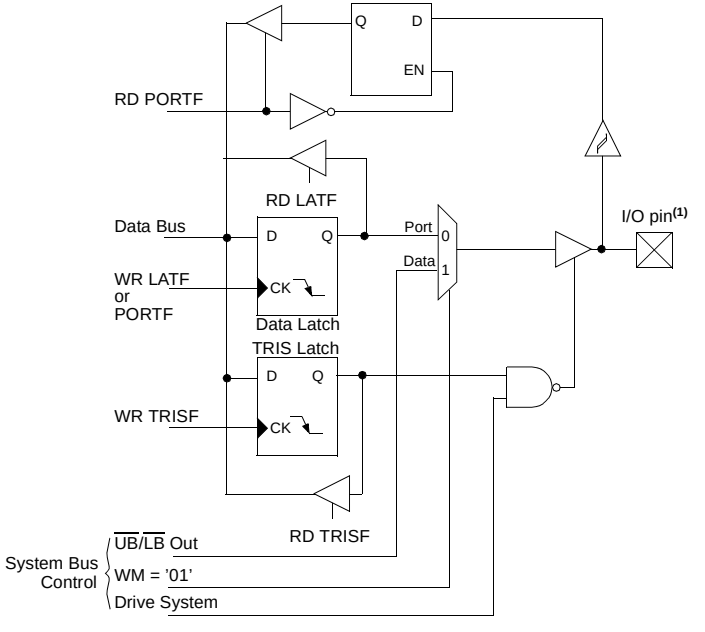
Note: I/O pins have diode protection to VDD and VSS.

FIGURE 9-12: RF5:RF3 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 9-13: RF7:RF6 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 9-11: PORTF FUNCTIONS

Name	Bit#	Buffer Type	Function
RF0/AN5	bit0	ST	Input/output port pin or analog input
RF1/AN6	bit1	ST	Input/output port pin or analog input
RF2/AN7	bit2	ST	Input/output port pin or analog input
RF3/CSIO	bit3	ST	Input/output port pin or I/O chip select
RF4/A16/$\overline{CS2}$(1)	bit4	ST	Input/output port pin or chip select 2 or address bit 16
RF5/$\overline{CS1}$	bit5	ST	Input/output port pin or chip select 1
RF6/$\overline{LB}$	bit6	ST	Input/output port pin or low byte select signal for external memory
RF7/$\overline{UB}$	bit7	ST	Input/output port pin or high byte select signal for external memory

Legend: ST = Schmitt Trigger input

Note 1: $\overline{CS2}$ is available only on PIC18C801.

TABLE 9-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF

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
TRISF	PORTF Data Direction Control Register								1111 1111	1111 1111
PORTF	Read PORTF pin/Write PORTF Data Latch								xxxx xxxx	uuuu uuuu
LATF	Read PORTF Data Latch/Write PORTF Data Latch								0000 0000	uuuu uuuu
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000
MEMCON	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTF.

9.7 PORTG, LATG, and TRISG Registers

PORTG is a 5-bit wide, bi-directional port. The corresponding data direction register is TRISG. Setting a TRISG bit (= 1) will make the corresponding PORTG pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISG bit (= 0) will make the corresponding PORTG pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATG register read and write the latched output value for PORTG.

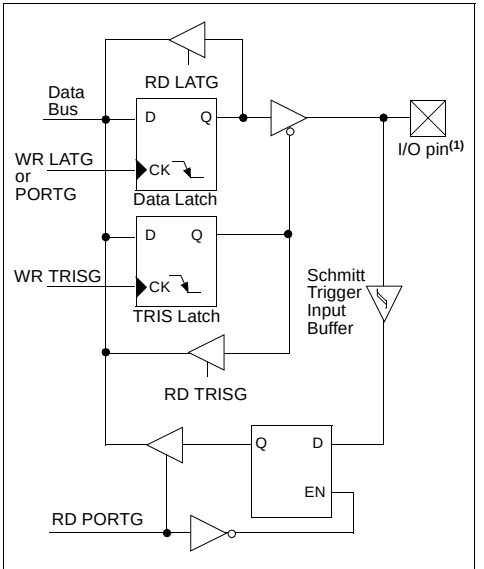
PORTG is multiplexed with system bus control signals ALE, $\overline{OE}$, $\overline{WRH}$, $\overline{WRL}$ and BA0. The $\overline{WRH}$ signal is the only signal that is disabled and configured as a port pin (RG3) during external program execution in 8-bit mode. All other pins are by default, system bus control signals. PORTG can be configured as an I/O port by setting EBDIS bit in the MEMCON register and when execution is taking place in internal program RAM.

Note:	On Power-on Reset, PORTG defaults to system bus signals.

EXAMPLE 9-8: INITIALIZING PORTG

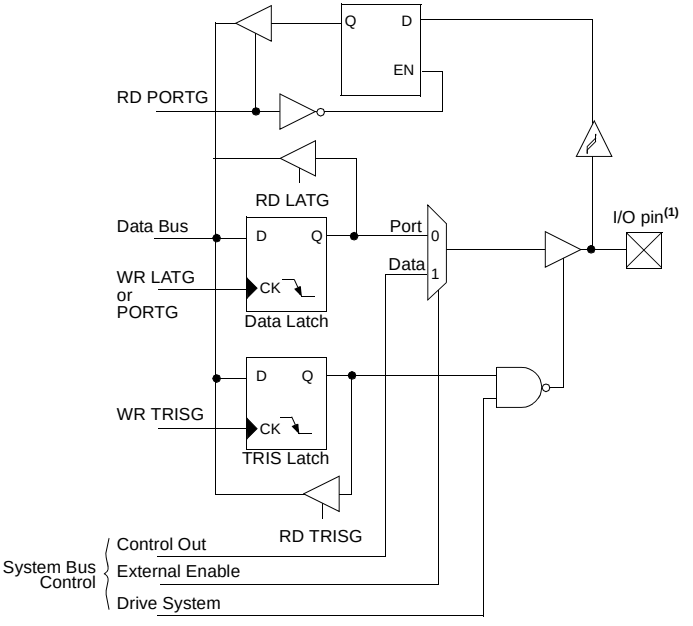
```
CLRF    PORTG      ; Initialize PORTG by
                   ; clearing output
                   ; data latches
CLRF    LATG       ; Alternate method
                   ; to clear output
                   ; data latches
MOVLW   04h        ; Value used to
                   ; initialize data
                   ; direction
MOVWF   TRISG      ; Set RG1:RG0 as outputs
                   ; RG2 as input
                   ; RG4:RG3 as outputs
```

FIGURE 9-14: PORTG BLOCK DIAGRAM IN I/O MODE



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 9-15: PORTG BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 9-13: PORTG FUNCTIONS

Name	Bit#	Buffer Type	Function
RG0/ALE	bit0	ST	Input/output port pin or Address Latch Enable signal for external memory
RG1/$\overline{OE}$	bit1	ST	Input/output port pin or Output Enable signal for external memory
RG2/$\overline{WRL}$	bit2	ST	Input/output port pin or Write Low byte signal for external memory
RG3/$\overline{WRH}$	bit3	ST	Input/output port pin or Write High byte signal for external memory
RG4/BA0	bit4	ST	Input/output port pin or Byte Address 0 signal for external memory

Legend: ST = Schmitt Trigger input

TABLE 9-14: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG

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
TRISG	PORTG Data Direction Control Register								---1 1111	---1 1111
PORTG	Read PORTG pin/Write PORTG Data Latch								---x xxxx	---u uuuu
LATG	Read PORTG Data Latch/Write PORTG Data Latch								---x xxxx	---u uuuu
MEMCON	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTG.

9.8 PORTH, LATH, and TRISH Registers

Note:	PORTH is available only on PIC18C801 devices.

PORTH is an 8-bit wide, bi-directional I/O port. The corresponding data direction register is TRISH. Setting a TRISH bit (= 1) will make the corresponding PORTH pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISH bit (= 0) will make the corresponding PORTH pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATH register read and write the latched output value for PORTH.

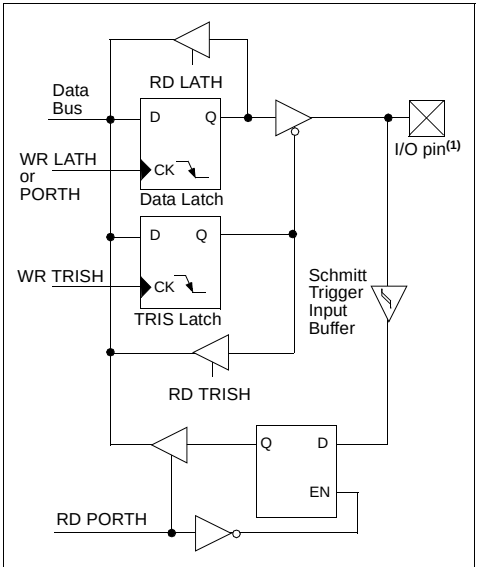
Pins RH7:RH4 are multiplexed with analog inputs AN18:AN11, while pins RH3:RH0 are multiplexed with system address bus A19:A16. By default, pins RH7:RH4 will setup as A/D inputs and pins RH3:RH0 will setup as system address bus. Register ADCON1 configures RH7:RH4 as I/O or A/D inputs. Register MEMCON configures RH3:RH0 as I/O or system bus pins.

Note 1:	On Power-on Reset, PORTH pins RH7:RH4 default to A/D inputs and read as '0'.
2:	On Power-on Reset, PORTH pins RH3:RH0 default to system bus signals.

EXAMPLE 9-9: INITIALIZING PORTH

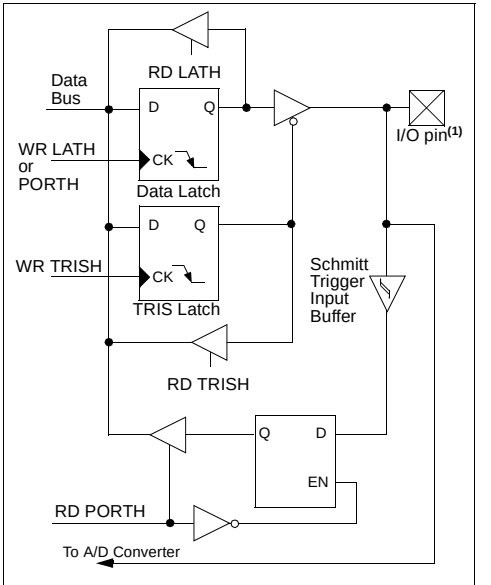
```
CLRF    PORTH       ; Initialize PORTH by
                    ; clearing output
                    ; data latches
CLRF    LATH        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   0Fh         ;
MOVWF   ADCON1      ;
MOVLW   0CFh        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISH       ; Set RH3:RH0 as inputs
                    ; RH5:RH4 as outputs
                    ; RH7:RH6 as inputs
```

FIGURE 9-16: RH3:RH0 PINS BLOCK DIAGRAM IN I/O MODE



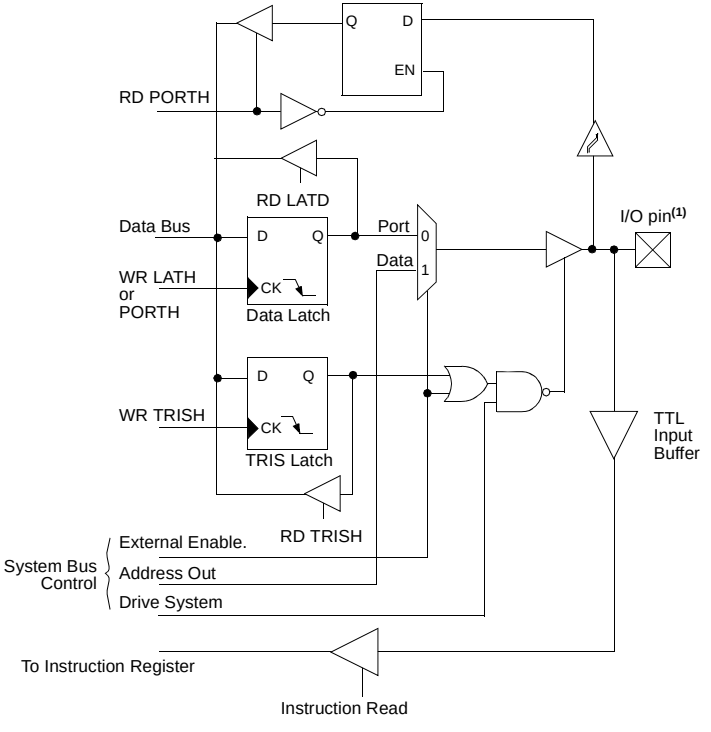
Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 9-17: RH7:RH4 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 9-18: RH3:RH0 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 9-15: PORTH FUNCTIONS

Name	Bit#	Buffer Type	Function
RH0/A16(1)	bit0	ST	Input/output port pin or Address bit 16 for external memory interface
RH1/A17(1)	bit1	ST	Input/output port pin or Address bit 17 for external memory interface
RH2/A18(1)	bit2	ST	Input/output port pin or Address bit 18 for external memory interface
RH3/A19(1)	bit3	ST	Input/output port pin or Address bit 19 for external memory interface
RH4/AN8(1)	bit4	ST	Input/output port pin or analog input channel 8
RH5/AN9(1)	bit5	ST	Input/output port pin or analog input channel 9
RH6/AN10(1)	bit6	ST	Input/output port pin or analog input channel 10
RH7/AN11(1)	bit7	ST	Input/output port pin or analog input channel 11

Legend: ST = Schmitt Trigger input

Note 1: PORTH is available only on PIC18C801 devices.

TABLE 9-16: SUMMARY OF REGISTERS ASSOCIATED WITH PORTH

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
TRISH	PORTH Data Direction Control Register								1111 1111	1111 1111
PORTH	Read PORTH pin/Write PORTH Data Latch								xxxx xxxx	uuuu uuuu
LATH	Read PORTH Data Latch/Write PORTH Data Latch								xxxx xxxx	uuuu uuuu
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000
MEMCON	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00

Legend: x = unknown, u = unchanged, - = unimplemented. Shaded cells are not used by PORTH.

9.9 PORTJ, LATJ, and TRISJ Registers

Note:	PORTJ is available only on PIC18C801 devices.

PORTJ is an 8-bit wide, bi-directional I/O port. The corresponding data direction register is TRISJ. Setting a TRISJ bit (= 1) will make the corresponding PORTJ pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISJ bit (= 0) will make the corresponding PORTJ pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATJ register read and write the latched output value for PORTJ.

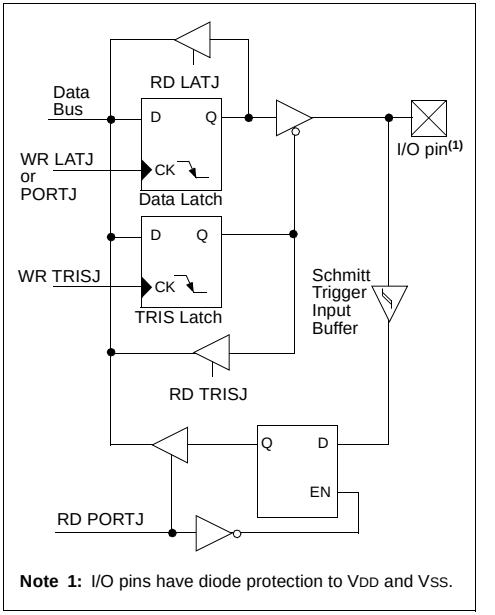
PORTJ is multiplexed with de-multiplexed system data bus D7:D0, when device is configured in 8-bit execution mode. Register MEMCON configures PORTJ as I/O or system bus pins.

Note:	On Power-on Reset, PORTJ defaults to system bus signals.

EXAMPLE 9-10: INITIALIZING PORTJ

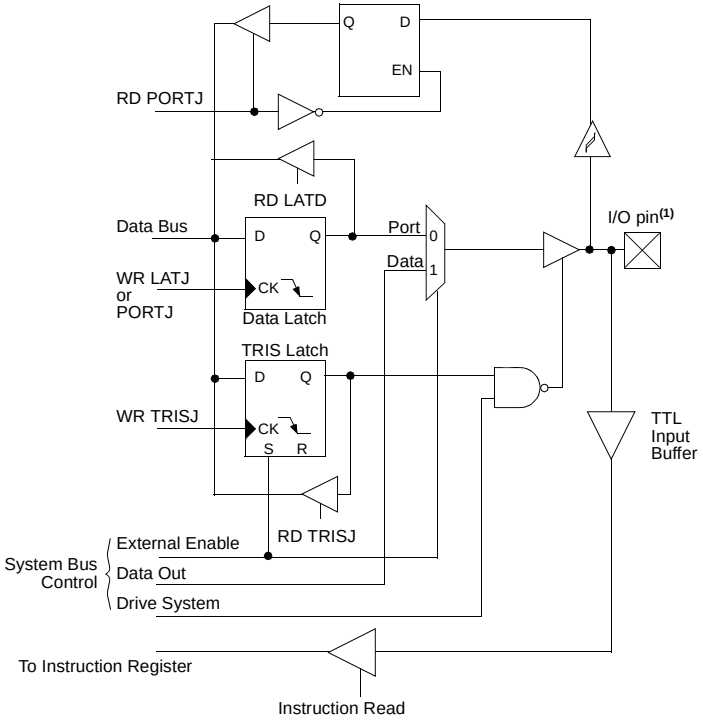
```
CLRF    PORTJ      ; Initialize PORTJ by
                   ; clearing output
                   ; data latches
CLRF    LATJ       ; Alternate method
                   ; to clear output
                   ; data latches
MOVLW   0CFh       ; Value used to
                   ; initialize data
                   ; direction
MOVWF   TRISJ      ; Set RJ3:RJ0 as inputs
                   ; RJ5:RJ4 as outputs
                   ; RJ7:RJ6 as inputs
```

FIGURE 9-19: PORTJ BLOCK DIAGRAM IN I/O MODE



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 9-20: PORTJ BLOCK DIAGRAM IN SYSTEM DATA BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 9-17: PORTJ FUNCTIONS

Name	Bit#	Buffer Type	Function
RJ0/D0(1)	bit0	ST/TTL	Input/output port pin or Data bit 0 for external memory interface
RJ1/D1(1)	bit1	ST/TTL	Input/output port pin or Data bit 1 for external memory interface
RJ2/D2(1)	bit2	ST/TTL	Input/output port pin or Data bit 2 for external memory interface
RJ3/D3(1)	bit3	ST/TTL	Input/output port pin or Data bit 3 for external memory interface
RJ4/D4(1)	bit4	ST/TTL	Input/output port pin or Data bit 4 for external memory interface
RJ5/D5(1)	bit5	ST/TTL	Input/output port pin or Data bit 5 for external memory interface
RJ6/D6(1)	bit6	ST/TTL	Input/output port pin or Data bit 6 for external memory interface
RJ7/D7(1)	bit7	ST/TTL	Input/output port pin or Data bit 7 for external memory interface

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: PORTJ is available only on PIC18C801 devices.

TABLE 9-18: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ

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
TRISJ	PORTJ Data Direction Control Register								1111 1111	1111 1111
PORTJ	Read PORTJ pin/Write PORTJ Data Latch								xxxx xxxx	uuuu uuuu
LATJ	Read PORTJ Data Latch/Write PORTJ Data Latch								xxxx xxxx	uuuu uuuu
MEMCON	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTJ.

10.0 TIMER0 MODULE

The Timer0 module has the following features:

- Software selectable as an 8-bit or 16-bit timer/ counter
- Readable and writable
- Dedicated 8-bit software programmable prescaler
- Clock source selectable to be external or internal
- Interrupt on overflow from FFh to 00h in 8-bit mode and FFFFh to 0000h in 16-bit mode
- Edge select for external clock

Register 10-1 shows the Timer0 Control register (T0CON).

Figure 10-1 shows a simplified block diagram of the Timer0 module in 8-bit mode and Figure 10-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

The T0CON register is a readable and writable register that controls all the aspects of Timer0, including the prescale selection.

Note:	Timer0 is enabled on POR.

REGISTER 10-1: T0CON REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

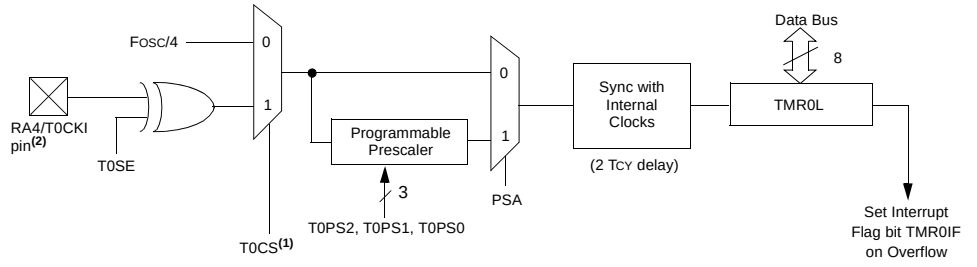
bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

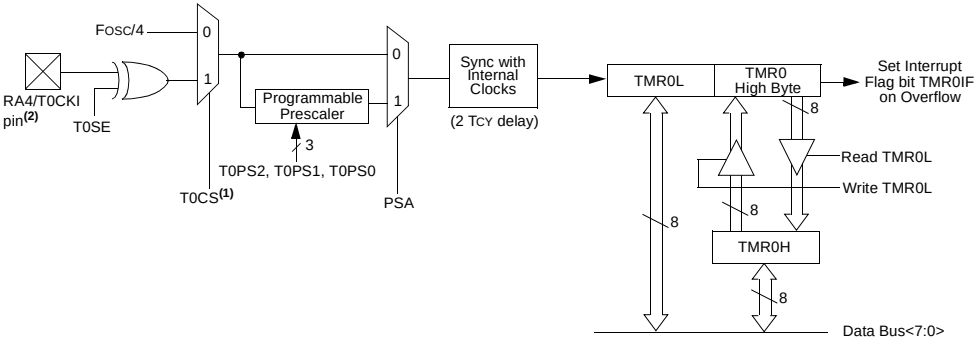
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	’1’ = Bit is set	’0’ = Bit is cleared	x = Bit is unknown

FIGURE 10-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE



Note 1: Upon RESET, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.
2: I/O pins have diode protection to VDD and VSS.

FIGURE 10-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE



Note 1: Upon RESET, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.
2: I/O pins have diode protection to VDD and VSS.

10.1 Timer0 Operation

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing the T0CS bit. In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0L register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0L register.

Counter mode is selected by setting the T0CS bit. In Counter mode, Timer0 will increment either on every rising, or falling edge, of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed below.

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

10.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not readable or writable.

The PSA and T0PS2:T0PS0 bits determine the prescaler assignment and prescale ratio.

Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4, ..., 1:256 are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g. `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0, x`.... etc.) will clear the prescaler count.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0, will clear the prescaler count but will not change the prescaler assignment.

10.2.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed “on-the-fly” during program execution).

10.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF bit. The interrupt can be masked by clearing the TMR0IE bit. The TMR0IF bit must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut-off during SLEEP.

10.4 16-Bit Mode Timer Reads and Writes

Timer0 can be set in 16-bit mode by clearing T0CON T08BIT. Registers TMR0H and TMR0L are used to access 16-bit timer value.

TMR0H is not the high byte of the timer/counter in 16-bit mode, but is actually a buffered version of the high byte of Timer0 (refer to Figure 10-1). The high byte of the Timer0 counter/timer is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16-bits of Timer0 without having to verify that the read of the high and low byte were valid, due to a rollover between successive reads of the high and low byte.

A write to the high byte of Timer0 must also take place through the TMR0H buffer register. Timer0 high byte is updated with the contents of the buffered value of TMR0H, when a write occurs to TMR0L. This allows all 16-bits of Timer0 to be updated at once.

TABLE 10-1: REGISTERS ASSOCIATED WITH TIMER0

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
TMR0L	Timer0 Module’s Low Byte Register								xxxx xxxx	uuuu uuuu
TMR0H	Timer0 Module’s High Byte Register								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
TRISA	—	PORTA Data Direction Register							--11 1111	--11 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

11.0 TIMER1 MODULE

The Timer1 module timer/counter has the following features:

- 16-bit timer/counter
  (Two 8-bit registers: TMR1H and TMR1L)
- Readable and writable (both registers)
- Internal or external clock select
- Interrupt on overflow from FFFFh to 0000h
- RESET from CCP module special event trigger

Register 11-1 shows the Timer1 Control register. This register controls the operating mode of the Timer1 module as well as contains the Timer1 oscillator enable bit (T1OSCEN). Timer1 can be enabled/disabled by setting/clearing control bit TMR1ON (T1CON register).

Figure 11-1 is a simplified block diagram of the Timer1 module.

Note:	Timer1 is disabled on POR.

REGISTER 11-1: T1CON REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register Read/Write of TImer1 in one 16-bit operation
0 = Enables register Read/Write of Timer1 in two 8-bit operations

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

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 Oscillator is enabled
0 = Timer1 Oscillator is shut-off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

11.1 Timer1 Operation

Timer1 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON register).

Note: When Timer1 is configured in an Asynchronous mode, care must be taken to make sure that there is no incoming pulse while Timer1 is being turned off. If there is an incoming pulse while Timer1 is being turned off, Timer1 value may become unpredictable.

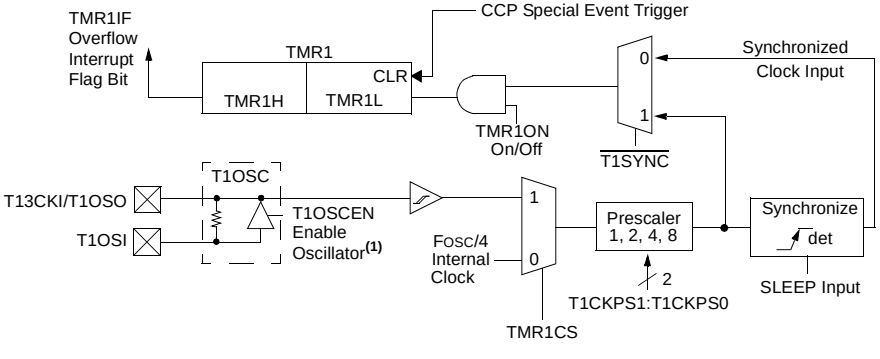
If an application requires that Timer1 be turned off and if it is possible that Timer1 may receive an incoming pulse while being turned off, synchronize the external clock first, by clearing the T1SYNC bit of register T1CON. Please note that this may cause Timer1 to miss up to one count.

When TMR1CS is clear, Timer1 increments every instruction cycle. When TMR1CS is set, Timer1 increments on every rising edge of the external clock input or the Timer1 oscillator, if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored.

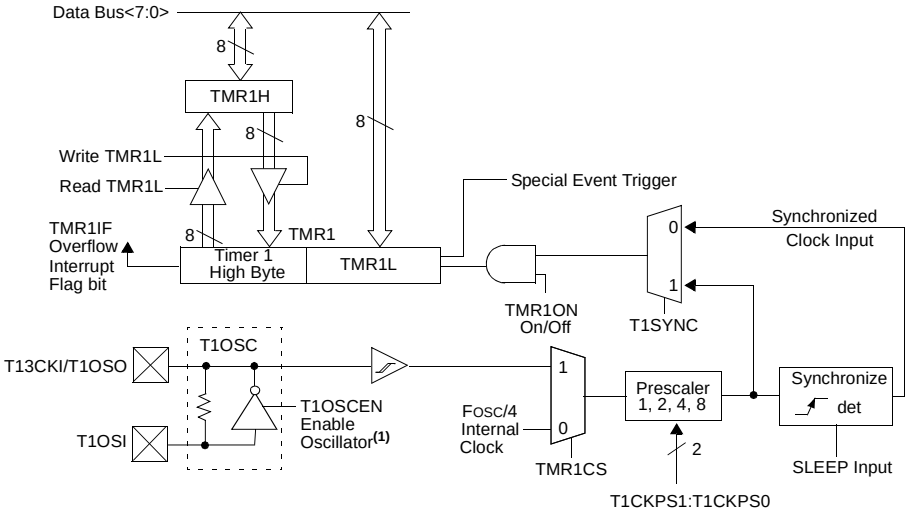
Timer1 also has an internal “RESET input”. This RESET can be generated by the CCP module (Table 14.0).

FIGURE 11-1: TIMER1 BLOCK DIAGRAM



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This reduces power drain.

FIGURE 11-2: TIMER1 BLOCK DIAGRAM: 16-BIT READ/WRITE MODE



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This reduces power drain.

11.2 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON register). The oscillator is a low power oscillator rated up to 200 kHz. It will continue to run during SLEEP. It is primarily intended for a 32 kHz crystal. Table 11-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

TABLE 11-1: CAPACITOR SELECTION FOR THE ALTERNATE OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 kHz	TBD(1)	TBD(1)
Crystal to be Tested:			
32.768 kHz	Epson C-001R32.768K-A	± 20 PPM	

Note 1: Microchip suggests 33 pF as a starting point in validating the oscillator circuit.
2: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.
3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
4: Capacitor values are for design guidance only.

11.3 Timer1 Interrupt

The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR1 interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit TMR1IF (PIR registers). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit TMR1IE (PIE registers).

11.4 Resetting Timer1 using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Note: The special event triggers from the CCP1 module will not set interrupt flag bit TMR1IF (PIR registers).

Timer1 must be configured for either Timer, or Synchronized Counter mode, to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this RESET operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L registers pair, effectively becomes the period register for Timer1.

11.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 11-2). When the RD16 control bit (T1CON register) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 high byte buffer. This provides the user with the ability to accurately read all 16 bits of Timer1, without having to determine whether a read of the high byte followed by a read of the low byte is valid, due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H buffer register. Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16-bits to both the high and low bytes of Timer1 at once.

The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 high byte buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

TABLE 11-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

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
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

12.0 TIMER2 MODULE

The Timer2 module timer has the following features:

- 8-bit timer (TMR2 register)
- 8-bit period register (PR2)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR2 match of PR2
- SSP module optional use of TMR2 output to generate clock shift

Register 12-1 shows the Timer2 Control register. Timer2 can be shut-off by clearing control bit TMR2ON (T2CON register), to minimize power consumption. Figure 12-1 is a simplified block diagram of the Timer2 module. The prescaler and postscaler selection of Timer2 are controlled by this register.

12.1 Timer2 Operation

Timer2 can be used as the PWM time-base for the PWM mode of the CCP module. The TMR2 register is readable and writable, and is cleared on any device RESET. The input clock (FOSC/4) has a prescale option of 1:1, 1:4, or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON register). The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, PIR registers).

The prescaler and postscaler counters are cleared when any of the following occurs:

- A write to the TMR2 register
- A write to the T2CON register
- Any device RESET (Power-on Reset, $\overline{\text{MCLR}}$ Reset, or Watchdog Timer Reset)

TMR2 is not cleared when T2CON is written.

Note:	Timer2 is disabled on POR.

REGISTER 12-1: T2CON REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

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

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

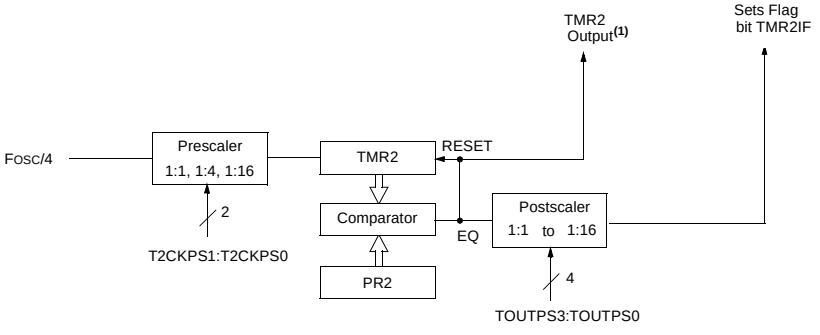
12.2 Timer2 Interrupt

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon RESET.

12.3 Output of TMR2

The output of TMR2 (before the postscaler) is a clock input to the Synchronous Serial Port module, which optionally uses it to generate the shift clock.

FIGURE 12-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP Module as a baud clock.

TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

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
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
TMR2	Timer2 Module's Register								0000 0000	0000 0000
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

13.0 TIMER3 MODULE

The Timer3 module timer/counter has the following features:

- 16-bit timer/counter
  (Two 8-bit registers: TMR3H and TMR3L)
- Readable and writable (both registers)
- Internal or external clock select
- Interrupt on overflow from FFFFh to 0000h
- RESET from CCP module trigger

Figure 13-1 is a simplified block diagram of the Timer3 module.

Register 13-1 shows the Timer3 Control register. This register controls the operating mode of the Timer3 module and sets the CCP clock source.

Register 11-1 shows the Timer1 Control register. This register controls the operating mode of the Timer1 module, as well as contains the Timer1 oscillator enable bit (T1OSCEN), which can be a clock source for Timer3.

Note:	Timer3 is disabled on POR.

REGISTER 13-1: T3CON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable
1 = Enables register Read/Write of Timer3 in one 16-bit operation
0 = Enables register Read/Write of Timer3 in two 8-bit operations

bit 6,3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the clock source for compare/capture CCP modules
01 = Timer3 is the clock source for compare/capture of CCP2,
Timer1 is the clock source for compare/capture of CCP1
00 = Timer1 is the clock source for compare/capture CCP modules

bit 5-4 **T3CKPS1:T3CKPS0**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T1CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

13.1 Timer3 Operation

Timer3 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

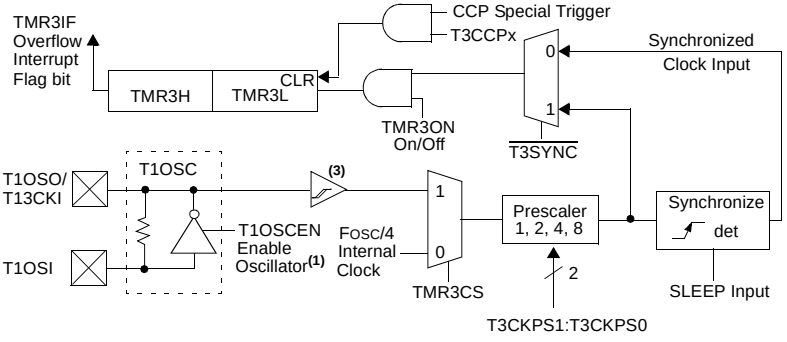
The operating mode is determined by the clock select bit, TMR3CS (T3CON register).

When TMR3CS = 0, Timer3 increments every instruction cycle. When TMR3CS = 1, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored.

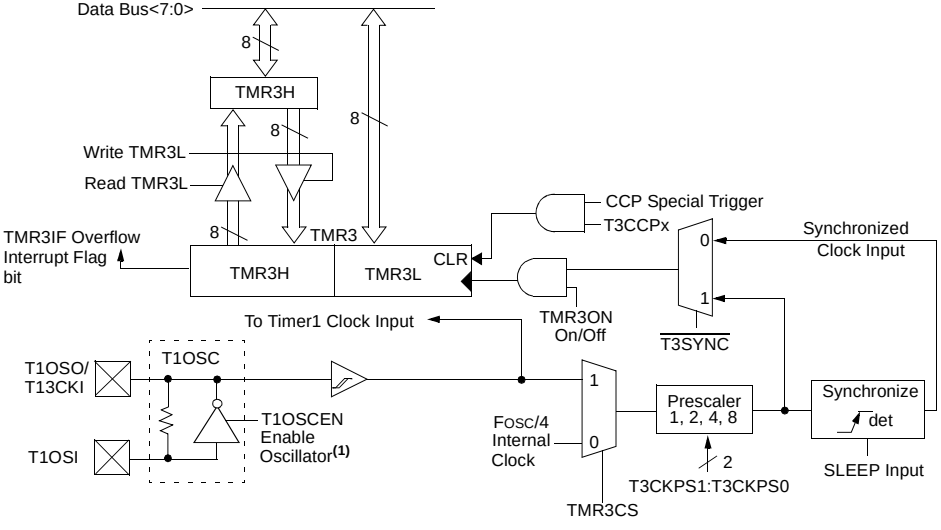
Timer3 also has an internal “RESET input”. This RESET can be generated by the CCP module (Section 13.0).

FIGURE 13-1: TIMER3 BLOCK DIAGRAM



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This reduces power drain.

FIGURE 13-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This reduces power drain.

13.2 Timer1 Oscillator

The Timer1 oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN bit (T1CON Register). The oscillator is a low power oscillator rated up to 200 kHz. Refer to "Timer1 Module", Section 11.0, for Timer1 oscillator details.

13.3 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit TMR3IF (PIE registers). This interrupt can be enabled/disabled by setting/clearing TMR3 interrupt enable bit TMR3IE (PIE registers).

13.4 Resetting Timer3 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer3.

Note:	The special event triggers from the CCP module will not set interrupt flag bit TMR3IF (PIR registers).

Timer3 must be configured for either Timer, or Synchronized Counter mode, to take advantage of this feature. If Timer3 is running in Asynchronous Counter mode, this RESET operation may not work. In the event that a write to Timer3 coincides with a special event trigger from CCP1, the write will take precedence. In this mode of operation, the CCPR1H:CCPR1L registers pair becomes the period register for Timer3. Refer to Section 14.0, "Capture/Compare/PWM (CCP) Modules", for CCP details.

TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

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
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR2	—	—	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	---- 0000	-0-- 0000
PIE2	—	—	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	---- 0000	-0-- 0000
IPR2	—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP	---- 0000	-0-- 0000
TMR3L	Holding register for the Least Significant Byte of the 16-bit TMR3 register								xxxx xxxx	uuuu uuuu
TMR3H	Holding register for the Most Significant Byte of the 16-bit TMR3 register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer3 module.

NOTES:

14.0 CAPTURE/COMPARE/PWM (CCP) MODULES

Each CCP (Capture/Compare/PWM) module contains a 16-bit register that can operate as a 16-bit capture register, as a 16-bit compare register, or as a PWM Duty Cycle register. Table 14-1 shows the timer resources of the CCP module modes.

The operation of CCP1 is identical to that of CCP2, with the exception of the special event trigger. Therefore, operation of a CCP module in the following sections is described, with respect to CCP1.

Table 14-2 shows the interaction of the CCP modules.

Register 14-1 shows the CCPx Control registers (CCPxCON). For the CCP1 module, the register is called CCP1CON and for the CCP2 module, the register is called CCP2CON.

REGISTER 14-1: CCP1CON REGISTER
CCP2CON REGISTER

	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
	bit 7							bit 0
	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
	bit 7							bit 0

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

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle bit1 and bit0

Capture mode:
Unused

Compare mode:
Unused

PWM mode:
These bits are the two LSbs (bit1 and bit0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets CCPx module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPxIF bit is set)
0011 = Reserved
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode,
Initialize CCP pin Low, on compare match force CCP pin High (CCPIF bit is set)
1001 = Compare mode,
Initialize CCP pin High, on compare match force CCP pin Low (CCPIF bit is set)
1010 = Compare mode,
Generate software interrupt on compare match
(CCPIF bit is set, CCP pin is unaffected)
1011 = Compare mode,
Trigger special event (CCPIF bit is set, reset TMR1 or TMR3)
11xx = PWM mode

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

14.1 CCP1 Module

Capture/Compare/PWM Register1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. All are readable and writable.

14.2 CCP2 Module

Capture/Compare/PWM Register2 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. All are readable and writable.

TABLE 14-1: CCP MODE - TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2

14.3 Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 or TMR3 registers, when an event occurs on pin RC2/CCP1. An event is defined as:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

An event is selected by control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit CCP1IF (PIR registers) is set. It must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value will be lost.

14.3.1 CCP PIN CONFIGURATION

In Capture mode, the RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit.

Note:	If the RC2/CCP1 is configured as an output, a write to the port can cause a capture condition.

14.3.2 TIMER1/TIMER3 MODE SELECTION

The timers used with the capture feature (either Timer1 and/or Timer3) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer used with each CCP module is selected in the T3CON register.

TABLE 14-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	TMR1 or TMR3 time-base. Time-base can be different for each CCP.
Capture	Compare	The compare could be configured for the special event trigger, which clears either TMR1 or TMR3, depending upon which time-base is used.
Compare	Compare	The compare(s) could be configured for the special event trigger, which clears TMR1 or TMR3, depending upon which time-base is used.
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt).
PWM	Capture	None.
PWM	Compare	None.

14.3.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE registers) clear to avoid false interrupts and should clear the flag bit CCP1IF, following any such change in operating mode.

14.3.4 CCP PRESCALER

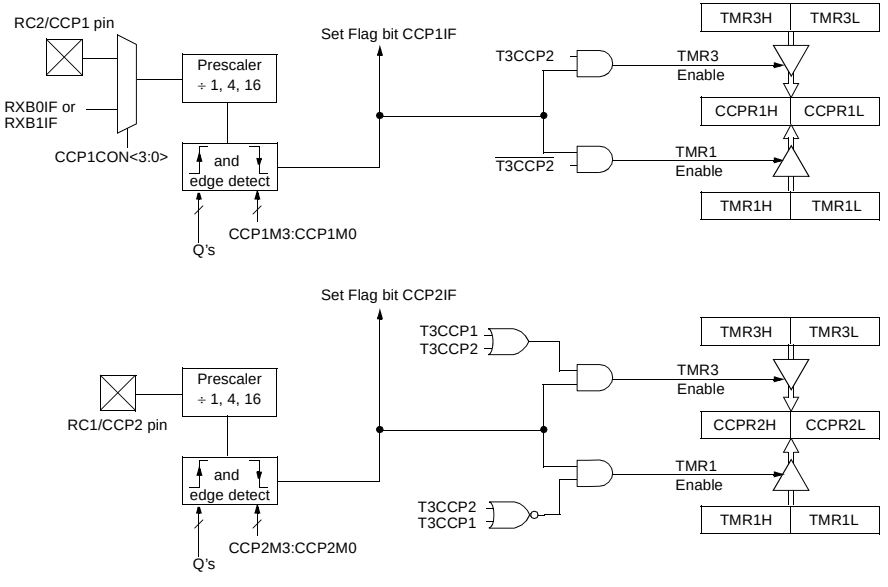
There are four prescaler settings, specified by bits CCP1M3:CCP1M0. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any RESET will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 14-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

EXAMPLE 14-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF   CCP1CON, F  ; Turn CCP module off
MOVLW  NEW_CAPT_PS ; Load WREG with the
                   ; new prescaler mode
                   ; value and CCP ON
MOVWF  CCP1CON     ; Load CCP1CON with
                   ; this value
```

FIGURE 14-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

14.4 Compare Mode

In Compare mode, the 16-bit CCPR1 (CCPR2) register value is constantly compared against either the TMR1 register pair value, or the TMR3 register pair value. When a match occurs, the RC2/CCP1 (RC1/CCP2) pin can have one of the following actions:

- Driven high
- Driven low
- Toggle output (high to low or low to high)
- Remains unchanged

The action on the pin is based on the value of control bits CCP1M3:CCP1M0 (CCP2M3:CCP2M0). At the same time, interrupt flag bit CCP1IF (CCP2IF) is set.

14.4.1 CCP PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the appropriate TRISC bit.

Note:	Clearing the CCP1CON register will force the RC2/CCP1 compare output latch to the default low level. This is not the data latch.

14.4.2 TIMER1/TIMER3 MODE SELECTION

Timer1 and/or Timer3 must be running in Timer mode, or Synchronized Counter mode, if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

14.4.3 SOFTWARE INTERRUPT MODE

When Generate Software Interrupt is chosen, the CCP1 pin is not affected. Only a CCP interrupt is generated (if enabled).

14.4.4 SPECIAL EVENT TRIGGER

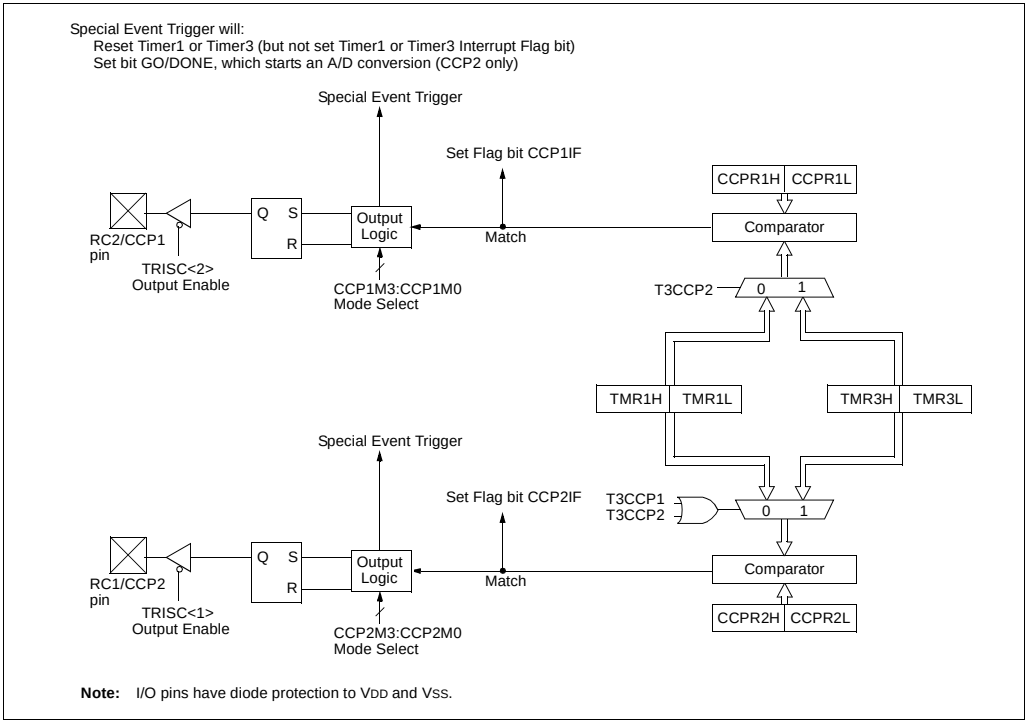
In this mode, an internal hardware trigger is generated, which may be used to initiate an action.

The special event trigger output of CCP1 resets the TMR1 register pair. This allows the CCPR1 register to effectively be a 16-bit programmable period register for Timer1.

The special trigger output of CCPx resets either the TMR1, or TMR3 register pair. Additionally, the CCP2 Special Event Trigger will start an A/D conversion, if the A/D module is enabled.

Note:	The special event trigger from the CCP2 module will not set the Timer1 or Timer3 interrupt flag bits.

FIGURE 14-2: COMPARE MODE OPERATION BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

TABLE 14-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

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
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
PIR2	—	—	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	---- 0000	---- 0000
PIE2	—	—	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	---- 0000	---- 0000
IPR2	—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP	---- 0000	---- 0000
TMR3L	Holding register for the Least Significant Byte of the 16-bit TMR3 register								xxxx xxxx	uuuu uuuu
TMR3H	Holding register for the Most Significant Byte of the 16-bit TMR3 register								xxxx xxxx	uuuu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.

14.5 PWM Mode

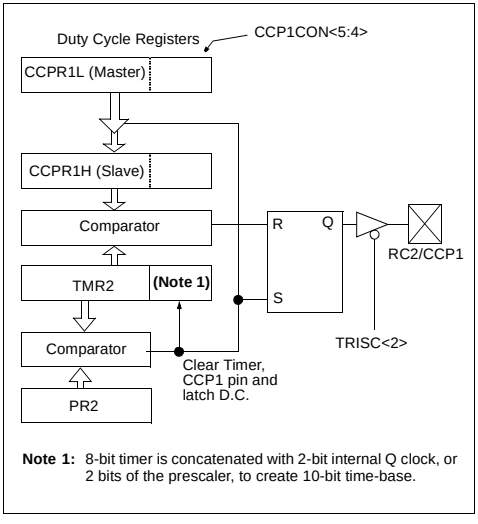
In Pulse Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note:	Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 14-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to setup the CCP module for PWM operation, see Section 14.5.3.

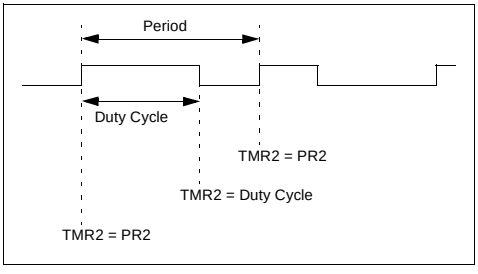
FIGURE 14-3: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create 10-bit time-base.

A PWM output (Figure 14-4) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 14-4: PWM OUTPUT



14.5.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated by the formula:

$$\text{PWM period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

where PWM frequency is defined as 1 / [PWM period].

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

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see Section 12.1) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

14.5.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the equation:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

14.5.3 SETUP FOR PWM OPERATION

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

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

TABLE 14-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 25 MHz

PWM Frequency	1.53 kHz	6.10 kHz	24.41 kHz	97.66kHz	195.31 kHz	260.42 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 14-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2

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
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TMR2	Timer2 Module's Register								0000 0000	0000 0000
PR2	Timer2 Module's Period Register								1111 1111	1111 1111
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
PIR2	—	—	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	---- 0000	---- 0000
PIE2	—	—	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	---- 0000	---- 0000
IPR2	—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP	---- 0000	---- 0000

Legend: x = unknown, u = unchanged, — = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.

NOTES:

15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

15.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) 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 MSSP module can operate in one of two modes:

- Serial Peripheral Interface™ (SPI)
- Inter-Integrated Circuit™ (I^2C)
  - Full Master mode
  - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

15.2 Control Registers

The MSSP module has three associated registers. These include a status register and two control registers.

Register 15-1 shows the MSSP Status Register (SSPSTAT), Register 15-2 shows the MSSP Control Register 1 (SSPCON1), and Register 15-3 shows the MSSP Control Register 2 (SSPCON2).

REGISTER 15-1: SSPSTAT REGISTER

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

bit 7 **SMP:** 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
CKP = 0:
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK
CKP = 1:
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK

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 MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a STOP bit has been detected last (this bit is '0' on RESET)
0 = STOP bit was not detected last

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

bit 2 **R/$\overline{W}$:** Read/Write bit Information (I^2C mode only)
This bit holds the R/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 $\overline{ACK}$ 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 MSSP is in IDLE mode.

bit 1 **UA:** Update Address (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 ACK and STOP bits), SSPBUF is empty

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 15-2: SSPCON1 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

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
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
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
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:
SCK 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 **SSPM3:SSPM0:** 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 = SPI Master mode, clock = TMR2 output/2
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))
1001 = Reserved
1010 = Reserved
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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 15-3: SSPCON2 REGISTER

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

bit 7 **GCEN:** General Call Enable bit (In I^2C Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (In I^2C Master mode only)
In Master Transmit mode:
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (In I^2C Master mode only)
In Master 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 Enabled 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 Enabled bit (In I^2C Master mode only)
1 = Initiate START condition on SDA and SCL pins. Automatically cleared by hardware.
0 = START condition idle

Note: 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).

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

15.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received, simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

- Serial Data Out (SDO) - RC5/SDO
- Serial Data In (SDI) - RC4/SDI/SDA
- Serial Clock (SCK) - RC3/SCK/SCL/LVOIN

Additionally, a fourth pin may be used when in any Slave mode of operation:

- Slave Select ($\overline{SS}$) - RA5/$\overline{SS}$/AN4

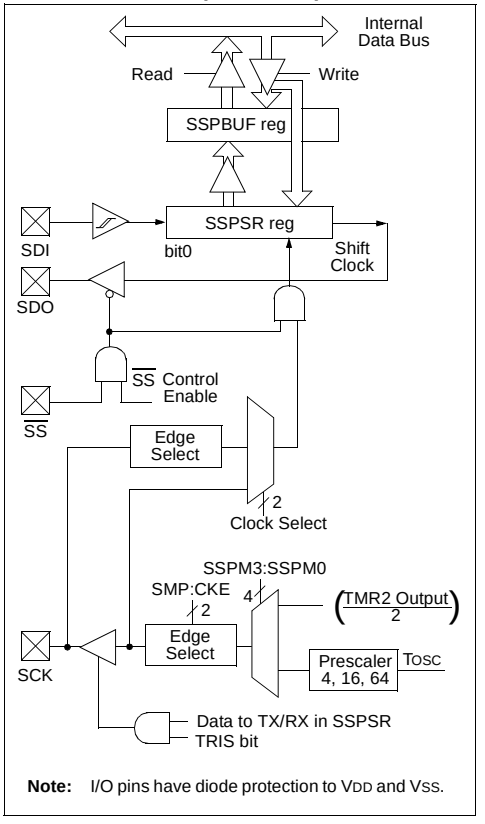
15.3.1 OPERATION

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)

Figure 15-1 shows the block diagram of the MSSP module, when in SPI mode.

FIGURE 15-1: MSSP BLOCK DIAGRAM (SPI MODE)



Note: I/O pins have diode protection to VDD and VSS.

The MSSP 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 8 bits of data have been received, that byte is moved to the SSPBUF register. Then the buffer full detect bit, BF (SSPSTAT register), and the interrupt flag bit, SSPIF (PIR registers), 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 (SSPCON1 register), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed 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 (BF) bit (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 MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. 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. Example 15-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable, and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP status register (SSPSTAT register) indicates the various status conditions.

15.3.2 ENABLING SPI I/O

To enable the serial port, SSP enable bit, SSPEN (SSPCON1 register), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPCON 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, corresponding pins must have their data direction bits (in the TRIS register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- RA5 must be configured as digital I/O using ADCON1 register
- $\overline{SS}$ must have TRISA<5> 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.

EXAMPLE 15-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
LOOP BTFSS SSPSTAT, BF    ;Has data been received (transmit complete)?
     BRA  LOOP            ;No
     MOVF  SSPBUF, W      ;WREG reg = contents of SSPBUF
     MOVWF RXDATA         ;Save in user RAM, if data is meaningful
     MOVF  TXDATA, W      ;W reg = contents of TXDATA
     MOVWF SSPBUF         ;New data to xmit
```


15.3.3 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave 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 a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a “line activity monitor” mode.

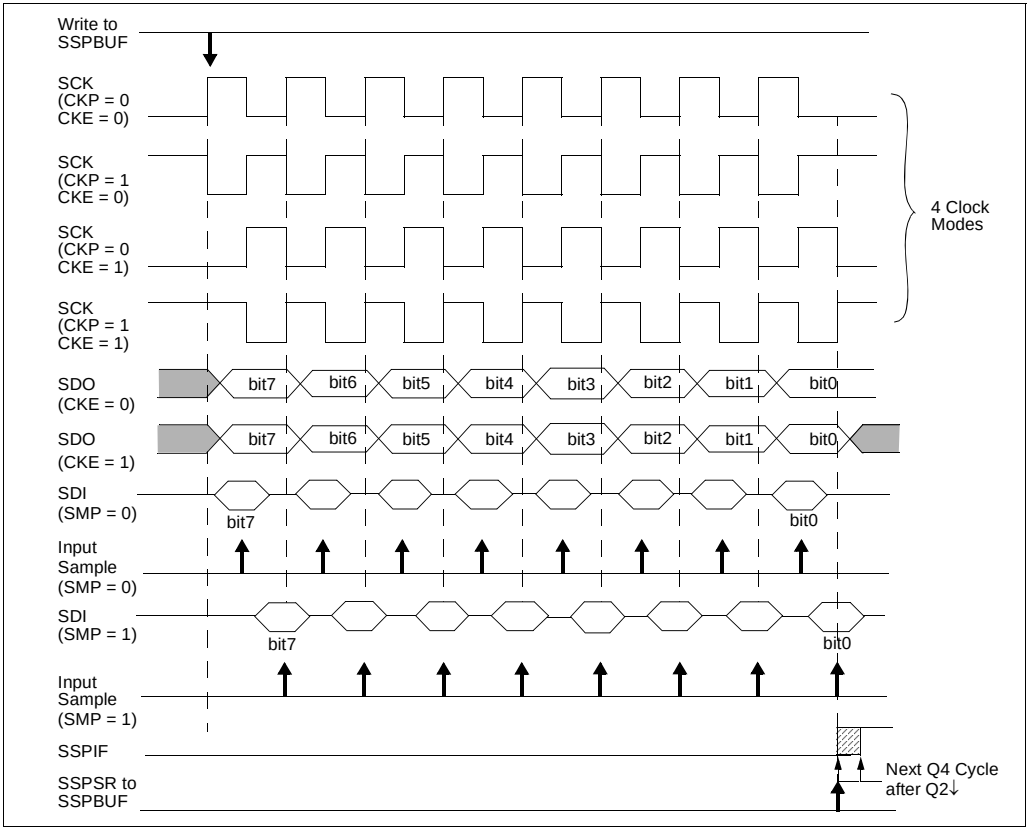
The clock polarity is selected by appropriately programming the CKP bit (SSPCON1 register). This, then, would give waveforms for SPI communication as shown in Figure 15-2, Figure 15-4, and Figure 15-5, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum data rate (at 25 MHz) of 6.25 Mbps.

Figure 15-2 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 15-2: SPI MODE WAVEFORM (MASTER MODE)



15.3.4 SLAVE MODE

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

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. When a byte is received, the device will wake-up from SLEEP.

15.3.5 SLAVE SELECT SYNCHRONIZATION

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> = 04h). The pin must not be driven low for the $\overline{SS}$ pin to function as an input. The data latch must be high. 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, (SSPCON<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to VDD.

2: If the SPI is used in Slave mode with CKE set, then the $\overline{SS}$ pin control must be enabled.

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.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function), since it cannot create a bus conflict.

FIGURE 15-3: SLAVE SYNCHRONIZATION WAVEFORM

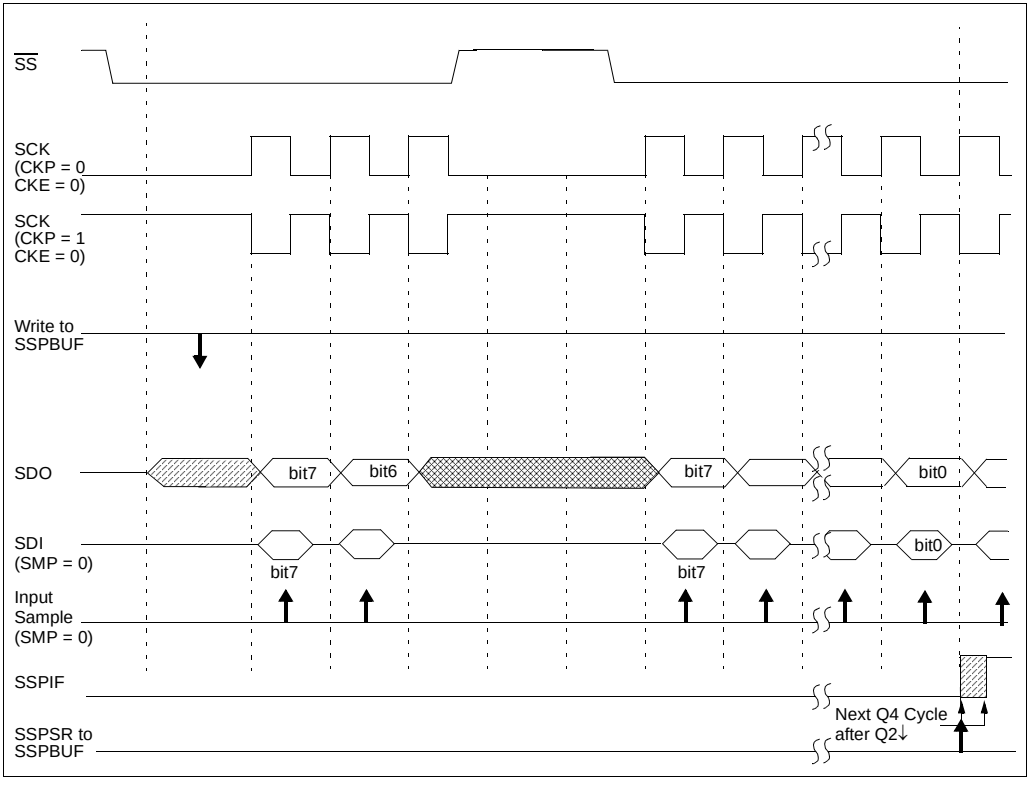


FIGURE 15-4: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

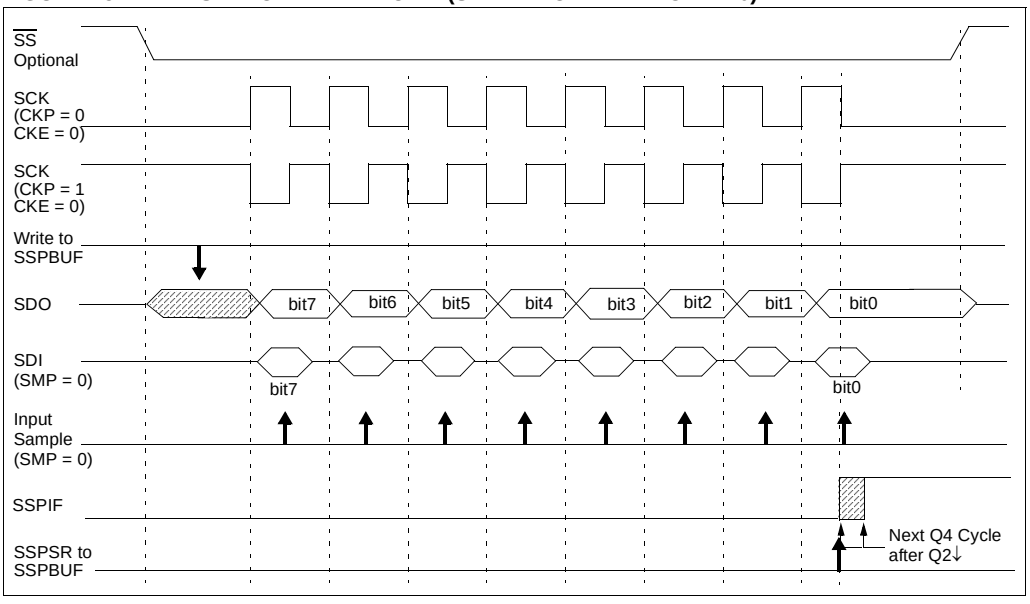
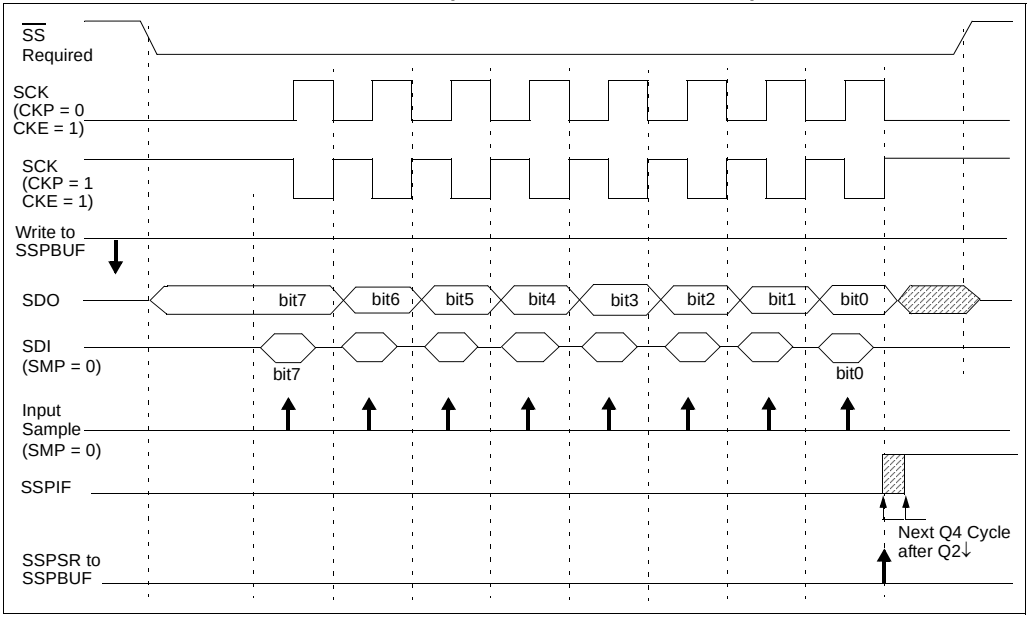


FIGURE 15-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



15.3.6 SLEEP OPERATION

In Master mode, all module clocks are halted, and the transmission/reception will remain in that state until the device wakes from SLEEP. After the device returns to normal mode, the module will continue to transmit/receive data.

In 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 MSSP interrupt flag bit will be set and, if enabled, will wake the device from SLEEP.

15.3.7 EFFECTS OF A RESET

A RESET disables the MSSP module and terminates the current transfer.

15.3.8 BUS MODE COMPATIBILITY

Table 15-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

TABLE 15-1: SPI BUS MODES

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

There is also a SMP bit that controls when the data will be sampled.

TABLE 15-2: REGISTERS ASSOCIATED WITH SPI OPERATION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
TRISA	—	PORTA Data Direction Register							--11 1111	--11 1111
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'.
Shaded cells are not used by the MSSP in SPI mode.

15.4 MSSP I²C Operation

The MSSP module in I²C mode, fully implements all master and slave functions (including general call support) and provides interrupts on START and STOP bits in hardware to determine a free bus (Multi-Master mode). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

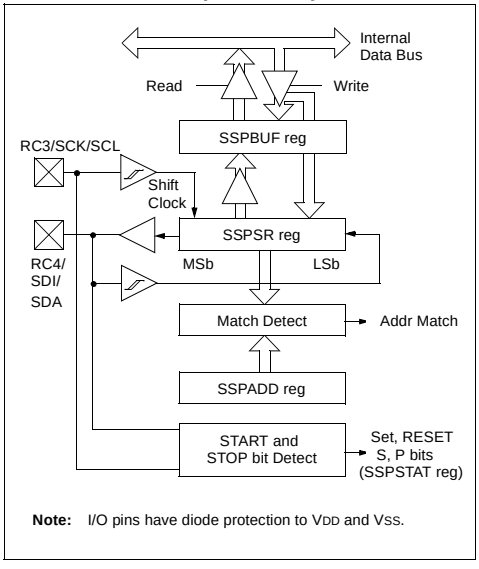
Two pins are used for data transfer. These are the RC3/SCK/SCL pin, which is the clock (SCL), and the RC4/SDI/SDA pin, which is the data (SDA). The user must configure these pins as inputs or outputs through the TRISC<4:3> bits.

The MSSP module functions are enabled by setting MSSP Enable bit SSPEN (SSPCON1 register).

The MSSP module has these six registers for I²C operation:

- MSSP Control Register1 (SSPCON1)
- MSSP Control Register2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- MSSP Shift Register (SSPSR) - Not directly accessible
- MSSP Address Register (SSPADD)

FIGURE 15-6: MSSP BLOCK DIAGRAM (I²C MODE)



Note: I/O pins have diode protection to VDD and VSS.

The SSPCON1 register allows control of the I²C operation. The SSPM3:SSPM0 mode selection bits (SSPCON1 register) allow one of the following I²C modes to be selected:

- I²C Master mode, clock = OSC/(4*(SSPADD +1))
- I²C Slave mode (7-bit address)
- I²C Slave mode (10-bit address)
- I²C Slave mode (7-bit address), with START and STOP bit interrupts enabled
- I²C Slave mode (10-bit address), with START and STOP bit interrupts enabled
- I²C firmware controlled master operation, slave is idle

Selection of any I²C mode with the SSPEN bit set, forces the SCL and SDA pins to be open drain, provided these pins are programmed to inputs by setting the appropriate TRISC bits.

15.4.1 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the acknowledge ($\overline{ACK}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

If either or both of the following conditions are true, the MSSP module will not give this $\overline{ACK}$ pulse:

a) The buffer full bit BF (SSPCON1 register) was set before the transfer was received.

b) The overflow bit SSPOV (SSPCON1 register) was set before the transfer was received.

In this event, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF (PIR registers) is set. The BF bit is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I²C specification, as well as the requirement of the MSSP module, is shown in timing parameter #100 and parameter #101.

15.4.1.1 Addressing

Once the MSSP module has been enabled, it waits for a START condition to occur. Following the START condition, the eight bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match, and the BF and SSPOV bits are clear, the following events occur:

a) The SSPSR register value is loaded into the SSPBUF register.
b) The buffer full bit BF is set.
c) An $\overline{ACK}$ pulse is generated.
d) MSSP interrupt flag bit SSPIF (PIR registers) is set on the falling edge of the ninth SCL pulse (interrupt is generated, if enabled).

In 10-bit Address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSb) of the first address byte, specify if this is a 10-bit address. The R/$\overline{W}$ bit (SSPSTAT register) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '1111 0 A9 A8 0', where A9 and A8 are the two MSb's of the address.

The sequence of events for 10-bit addressing is as follows, with steps 7- 9 for slave-transmitter:

1. Receive first (high) byte of address (the SSPIF, BF and UA bits (SSPSTAT register) are set).
2. Update the SSPADD register with second (low) byte of address (clears bit UA and releases the SCL line).
3. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
4. Receive second (low) byte of address (bits SSPIF, BF, and UA are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit UA.
6. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
7. Receive repeated START condition.
8. Receive first (high) byte of address (bits SSPIF and BF are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.

15.4.1.2 Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register.

When the address byte overflow condition exists, then no acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT register) is set or bit SSPOV (SSPCON1 register) is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR registers) must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

15.4.1.3 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. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then pin RC3/SCK/SCL should be enabled by setting bit CKP (SSPCON1 register). The master must monitor the SCL pin prior to asserting another clock pulse. The slave devices may be holding off the master by stretching the clock. 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 (Figure 15-8).

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in 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.

As a slave-transmitter, the $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{ACK}$), then the data transfer is complete. When the $\overline{ACK}$ is latched by the slave, the slave logic is reset (resets SSPSTAT register) and the slave monitors for another occurrence of the START bit. If the SDA line was low ($\overline{ACK}$), the transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Pin RC3/SCK/SCL should be enabled by setting bit CKP.

FIGURE 15-7: I²C SLAVE MODE WAVEFORMS FOR RECEPTION (7-BIT ADDRESS)

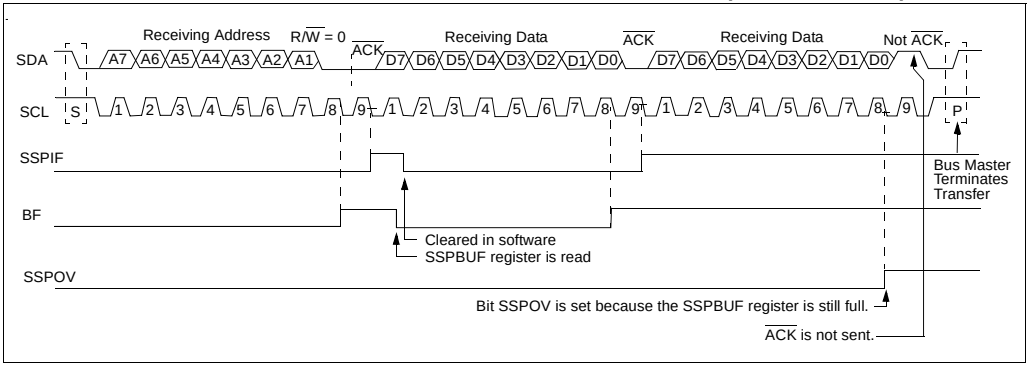
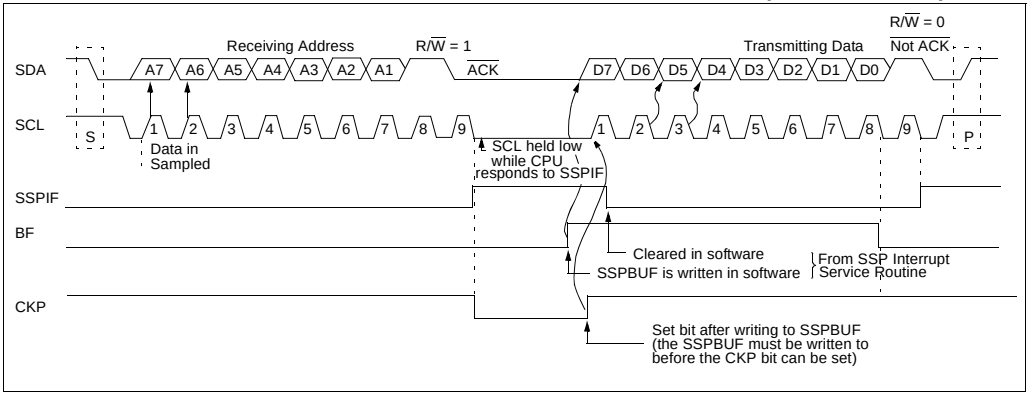


FIGURE 15-8: I²C SLAVE MODE WAVEFORMS FOR TRANSMISSION (7-BIT ADDRESS)



15.4.2 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. 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 one of eight addresses reserved for specific purposes by the I^2C protocol. It consists of all 0's with R/$\overline{W}$ = 0.

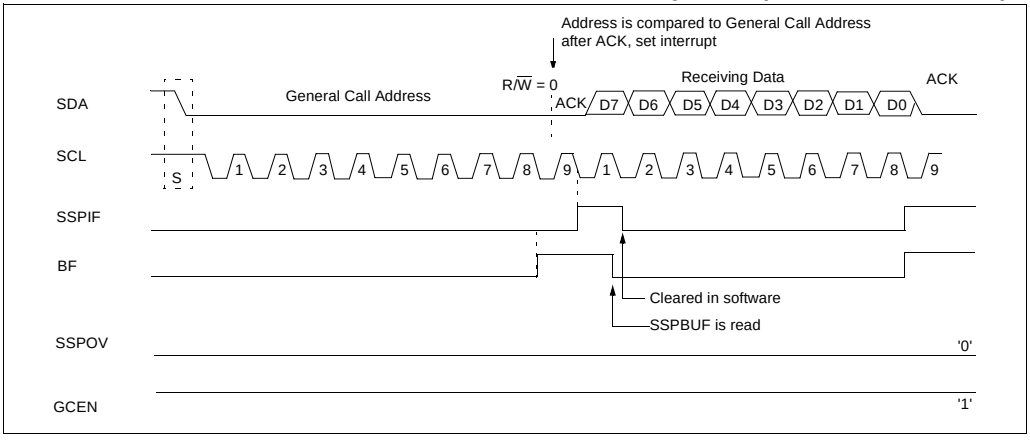
The general call address is recognized (enabled) when the General Call Enable (GCEN) bit is set (SSPCON2 register). Following a START bit detect, eight bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF bit is set (eighth bit), and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device specific or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match, and the UA bit is set (SSPSTAT register). If the general call address is sampled when the GCEN bit is set and while the slave is configured in 10-bit address mode, then the second half of the address is not necessary. The UA bit will not be set, and the slave will begin receiving data after the Acknowledge (Figure 15-9).

FIGURE 15-9: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESS)



15.4.3 MASTER MODE

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 MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set, or the bus is idle, with both the S and P bits clear.

In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

The following events will cause SSP Interrupt Flag bit, SSPIF, to be set (SSP Interrupt if enabled):

- START condition
- STOP condition
- Data transfer byte transmitted/received
- Acknowledge Transmit
- Repeated START condition

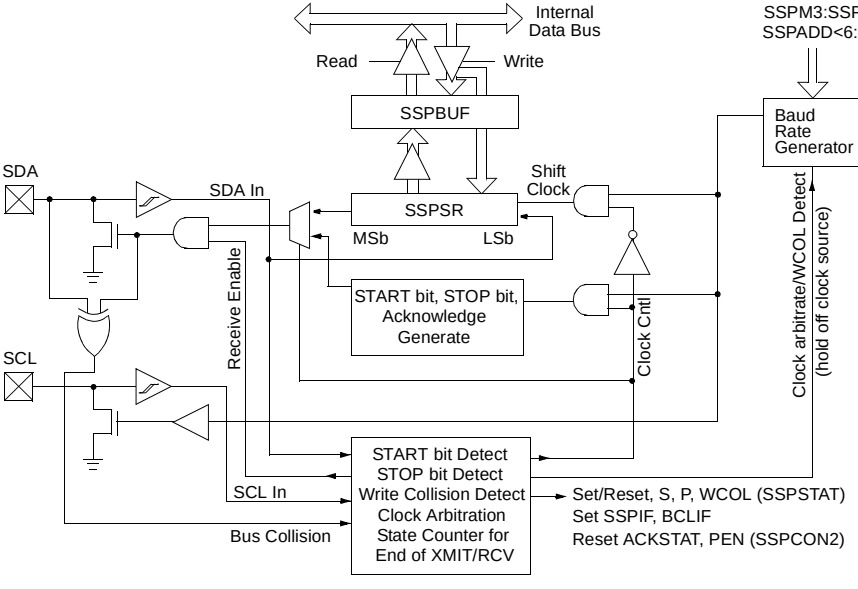
15.4.4 I^2C MASTER MODE SUPPORT

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. Once Master mode is enabled, the user has the following six options:

1. Assert a START condition on SDA and SCL.
2. Assert a Repeated START condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Generate a STOP condition on SDA and SCL.
5. Configure the I^2C port to receive data.
6. Generate an Acknowledge condition at the end of a received byte of data.

Note: The MSSP module, when configured in I^2C 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 imitate 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.

FIGURE 15-10: MSSP BLOCK DIAGRAM (I^2C MASTER MODE)



Note: I/O pins have diode protection to VDD and VSS.

15.4.4.1 I^2C 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 I^2C 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 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.

The baud rate generator used for the SPI mode operation is now used to set the SCL clock frequency for either 100 kHz, 400 kHz, or 1 MHz I^2C operation. The baud rate generator reload value is contained in the lower 7 bits of the SSPADD register. The baud rate generator will automatically begin counting on a write to the SSPBUF. Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

A typical transmit sequence would go as follows:

a) The user generates a START condition by setting the START enable (SEN) bit (SSPCON2 register).
b) SSPIF is set. The MSSP module will wait the required start time before any other operation takes place.
c) The user loads the SSPBUF with the address to transmit.
d) Address is shifted out the SDA pin until all eight bits are transmitted.
e) The MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit (SSPCON2 register).
f) The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
g) The user loads the SSPBUF with eight bits of data.
h) Data is shifted out the SDA pin until all eight bits are transmitted.
i) The MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit (SSPCON2 register).
j) The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
k) The user generates a STOP condition by setting the STOP enable bit PEN (SSPCON2 register).
l) Interrupt is generated once the STOP condition is complete.

15.4.5 BAUD RATE GENERATOR

In I^2C Master mode, the reload value for the BRG is located in the lower 7 bits of the SSPADD register (Figure 15-11). When the BRG is loaded with this value, the BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (T_{CY}) on the Q2 and Q4 clocks. In I^2C Master mode, the BRG is reloaded automatically. If clock arbitration is taking place, for instance, the BRG will be reloaded when the SCL pin is sampled high (Figure 15-12).

FIGURE 15-11: BAUD RATE GENERATOR BLOCK DIAGRAM

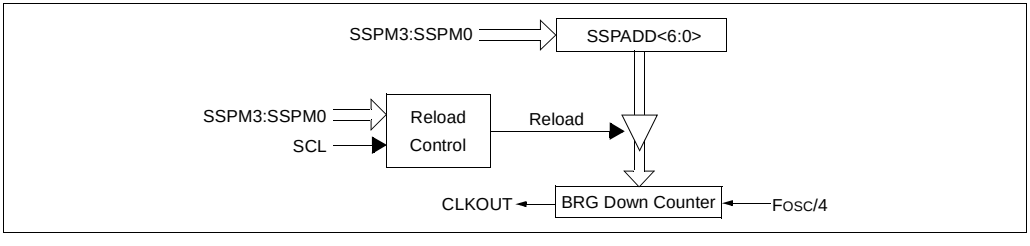
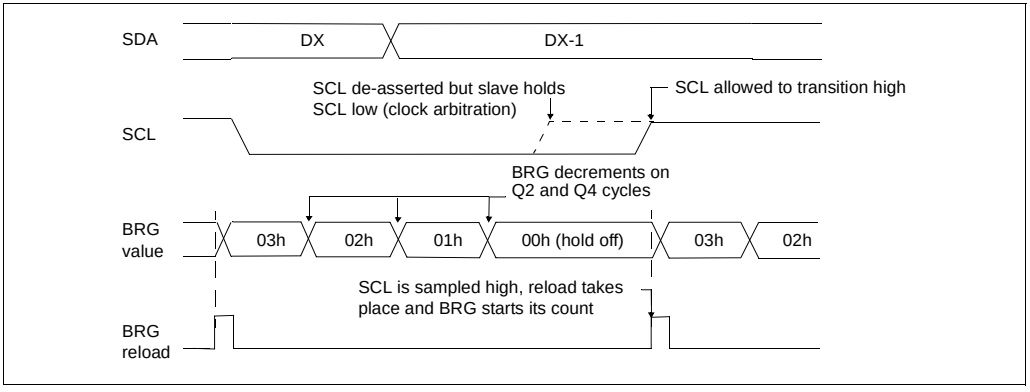


FIGURE 15-12: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



15.4.6 I²C MASTER MODE START CONDITION TIMING

To initiate a START condition, the user sets the START Condition Enable (SEN) bit (SSPCON2 register). If the SDA and SCL pins are sampled high, the baud rate generator is re-loaded with the contents of SSPADD<6:0> 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 (SSPSTAT register) to be set. Following this, the baud rate generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the baud rate generator times out (TBRG), the SEN bit (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.

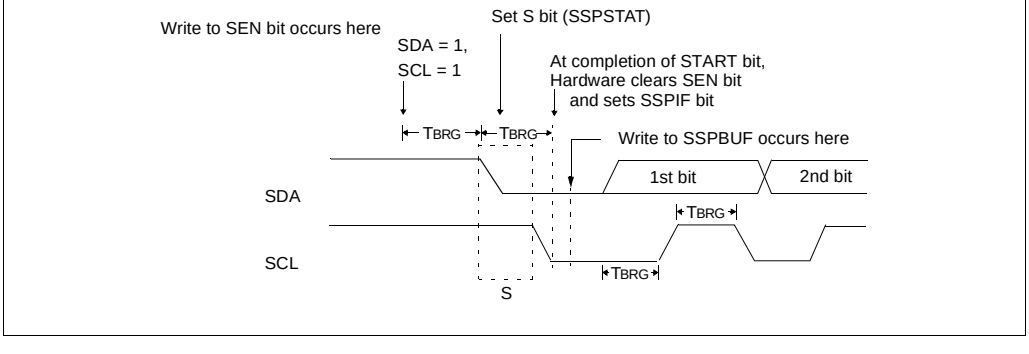
Note:	If at the beginning of the START condition, the SDA and SCL pins are already sampled low, or if during the START condition the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag BCLIF is set, the START condition is aborted, and the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

15.4.6.1 WCOL Status Flag

If the user writes the SSPBUF when a START sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

Note:	Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the START condition is complete.

FIGURE 15-13: FIRST START BIT TIMING



15.4.7 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated START condition occurs when the RSEN bit (SSPCON2 register) is programmed high and the I²C logic module is in the IDLE state. 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 with the contents of SSPADD<5:0> 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 de-asserted (brought high). When SCL is sampled high, the baud rate generator is re-loaded with the contents of SSPADD<6:0> 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. Following this, the RSEN bit (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 (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".

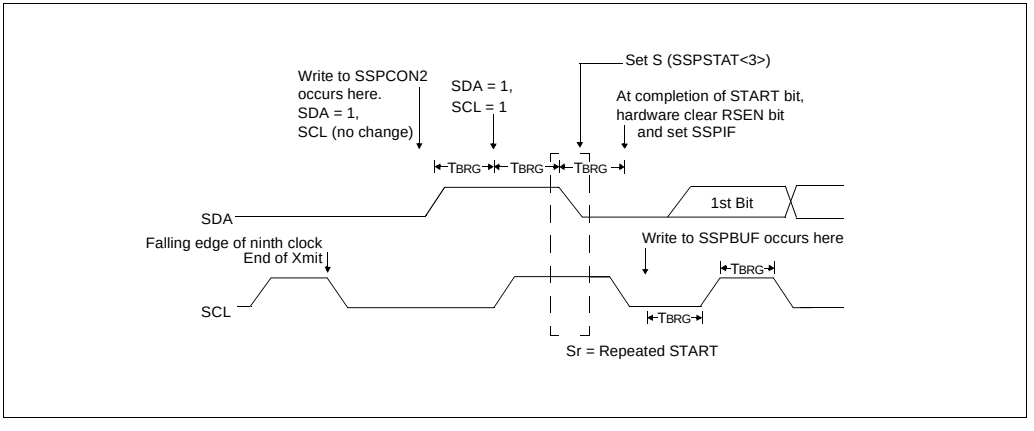
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

15.4.7.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated START sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

Note: Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated START condition is complete.

FIGURE 15-14: REPEATED START CONDITION WAVEFORM



15.4.8 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 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 (see data hold time specification parameter 106). SCL is held low for one baud rate generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification parameter 107). 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 bit is cleared and the master releases SDA, allowing the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time, if an address match occurs, or if data was received properly. The status of $\overline{ACK}$ is written into the ACKDT bit on the falling 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 15-15).

After the write to the SSPBUF, each bit of address will be shifted out on the falling edge of SCL, until all seven address bits and the $\overline{R/W}$ bit, are completed. On the falling edge of the eighth clock, the master will deassert 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 ACK bit is loaded into the ACKSTAT status bit (SSPCON2 register). Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF bit 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.

15.4.8.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT register) is set when the CPU writes to SSPBUF, and is cleared when all eight bits are shifted out.

15.4.8.2 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 is set and the contents of the buffer are unchanged (the write doesn't occur).

WCOL must be cleared in software.

15.4.8.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (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.

15.4.9 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPCON2 register).

Note:	The MSSP 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 RCEN bit is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF bit is set, the SSPIF flag bit is set and the baud rate generator is suspended from counting, holding SCL low. The MSSP is now in IDLE state, awaiting the next command. When the buffer is read by the CPU, the BF bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception, by setting the Acknowledge Sequence Enable bit ACKEN (SSPCON2 register).

15.4.9.1 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.

15.4.9.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when eight bits are received into the SSPSR and the BF bit is already set from a previous reception.

15.4.9.3 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 doesn't occur).

FIGURE 15-15: I^2C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

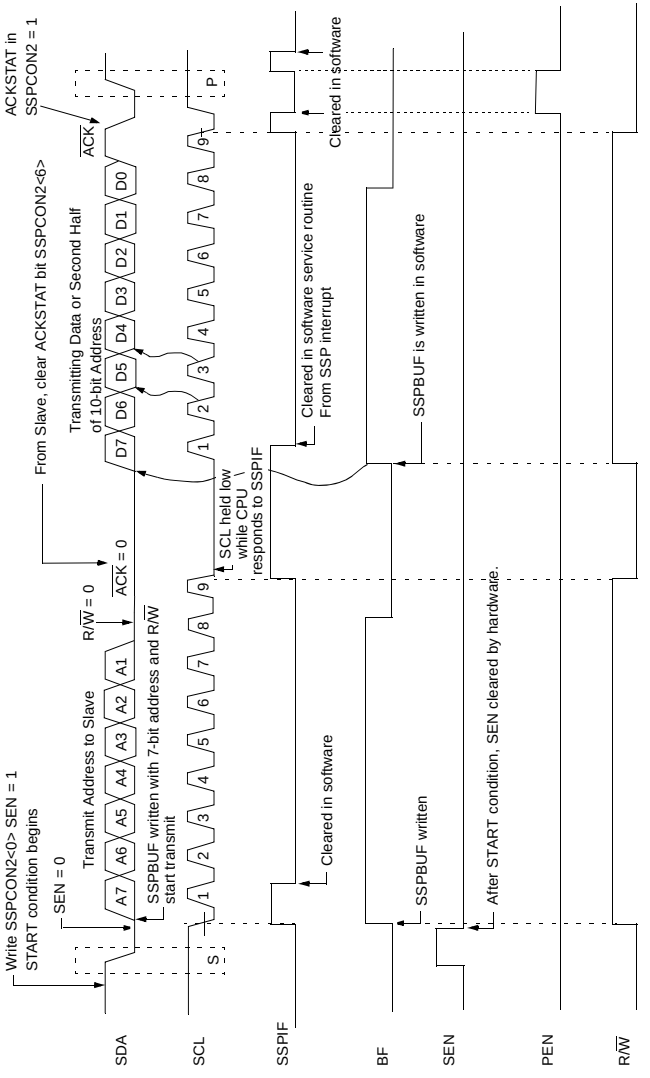
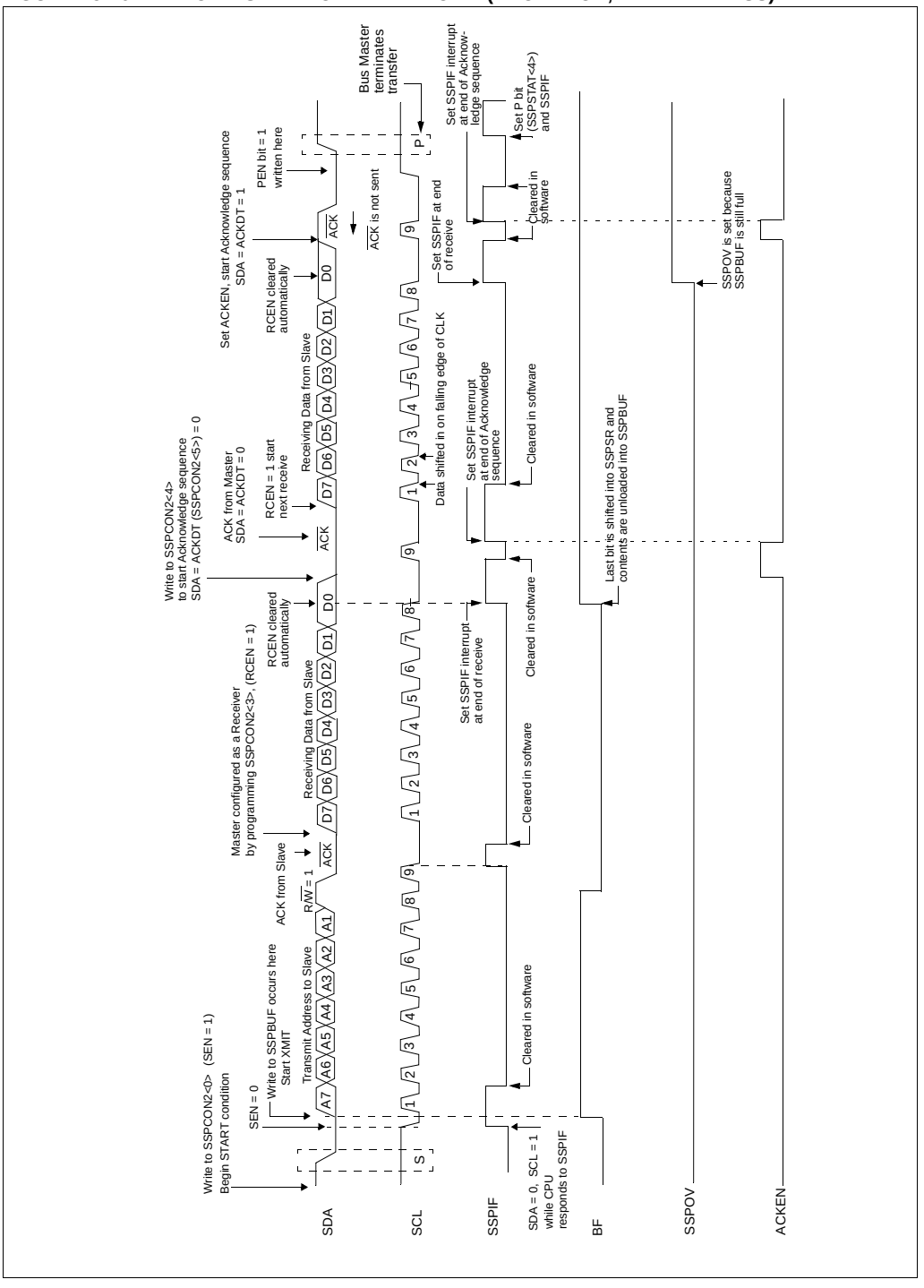


FIGURE 15-16: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



15.4.10 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence enable bit, ACKEN (SSPCON2 register). When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge Data bit (ACKDT) is 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 de-asserted (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 MSSP module then goes into IDLE mode (Figure 15-17).

15.4.10.1 WCOL Status Flag

If the user writes the SSPBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

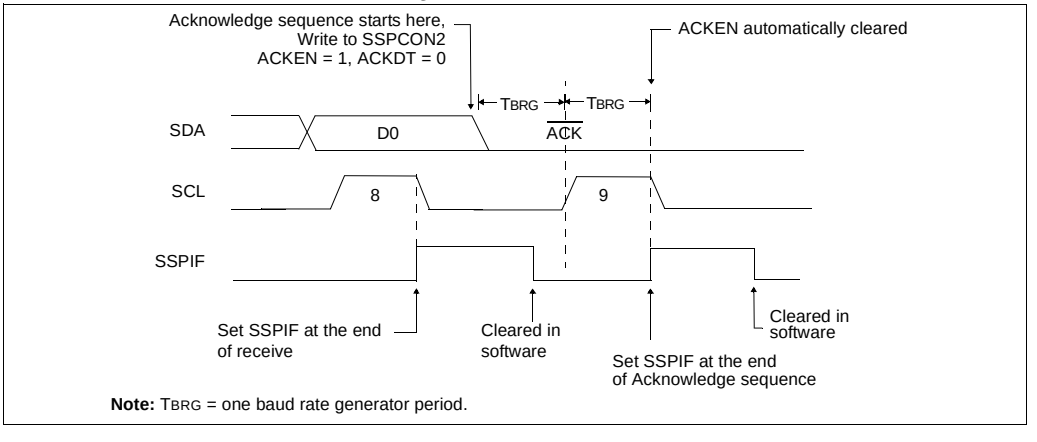
15.4.11 STOP CONDITION TIMING

A STOP bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN (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 de-asserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT register) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 15-18).

15.4.11.1 WCOL Status Flag

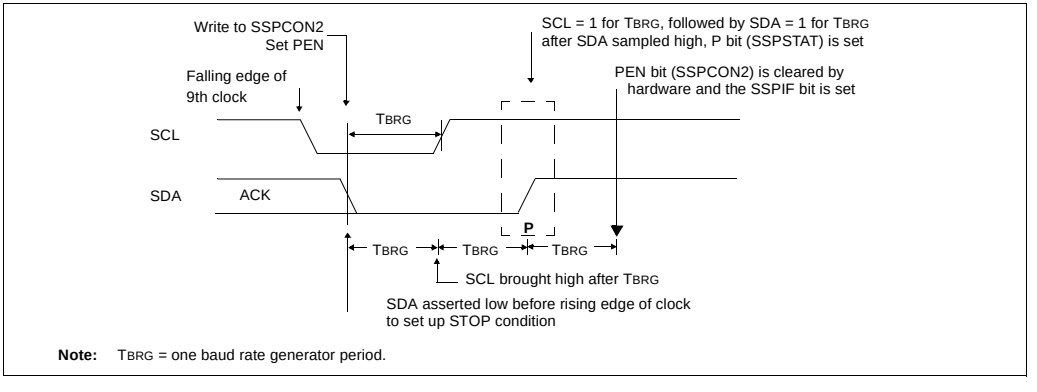
If the user writes the SSPBUF when a STOP sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 15-17: ACKNOWLEDGE SEQUENCE WAVEFORM



Note: TBRG = one baud rate generator period.

FIGURE 15-18: STOP CONDITION RECEIVE OR TRANSMIT MODE



Note: TBRG = one baud rate generator period.

15.4.12 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated START/STOP condition, de-asserts 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<6:0> 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 15-19).

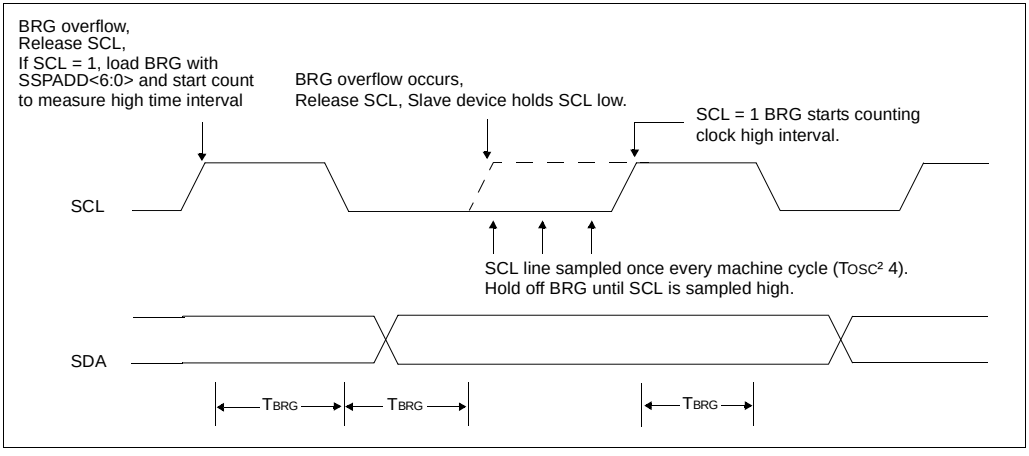
15.4.13 SLEEP OPERATION

While in SLEEP mode, the I^2C module can receive addresses or data, and when an address match or complete byte transfer occurs, wake the processor from SLEEP (if the MSSP interrupt is enabled).

15.4.14 EFFECT OF A RESET

A RESET disables the MSSP module and terminates the current transfer.

FIGURE 15-19: CLOCK ARBITRATION TIMING IN MASTER TRANSMIT MODE



15.4.15 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 MSSP module is disabled. Control of the I^2C bus may be taken when the P bit (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 in hardware, with the result placed in the BCLIF bit.

Arbitration can be lost in the following states:

- Address transfer
- Data transfer
- A START condition
- A Repeated START condition
- An Acknowledge condition

15.4.16 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 = '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 15-20).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF bit is cleared, the SDA and SCL lines are de-asserted, 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.

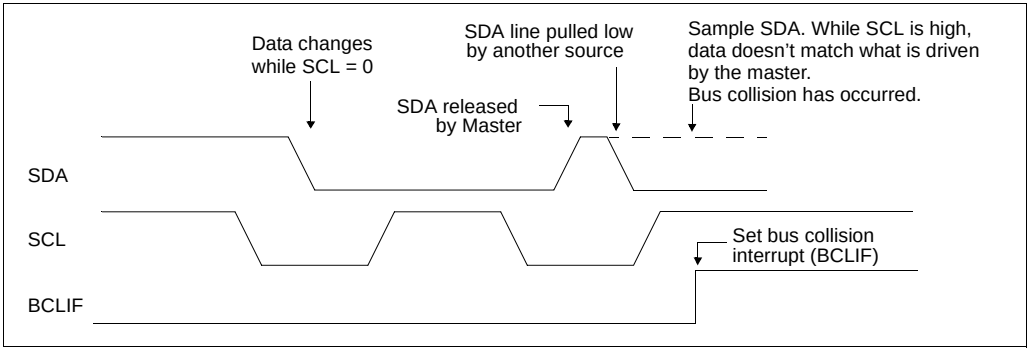
If a START, Repeated START, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted, and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service 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 15-20: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



15.4.16.1 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 15-21).

b) SCL is sampled low before SDA is asserted low (Figure 15-22).

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 MSSP module is reset to its IDLE state (Figure 15-21).

The START condition begins with the SDA and SCL pins de-asserted. When the SDA pin is sampled high, the baud rate generator is loaded from SSPADD<6:0> and counts down to 0. 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 15-23). 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 0, and during this time, if the SCL pin is sampled as '0', 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 15-21: BUS COLLISION DURING START CONDITION (SDA ONLY)

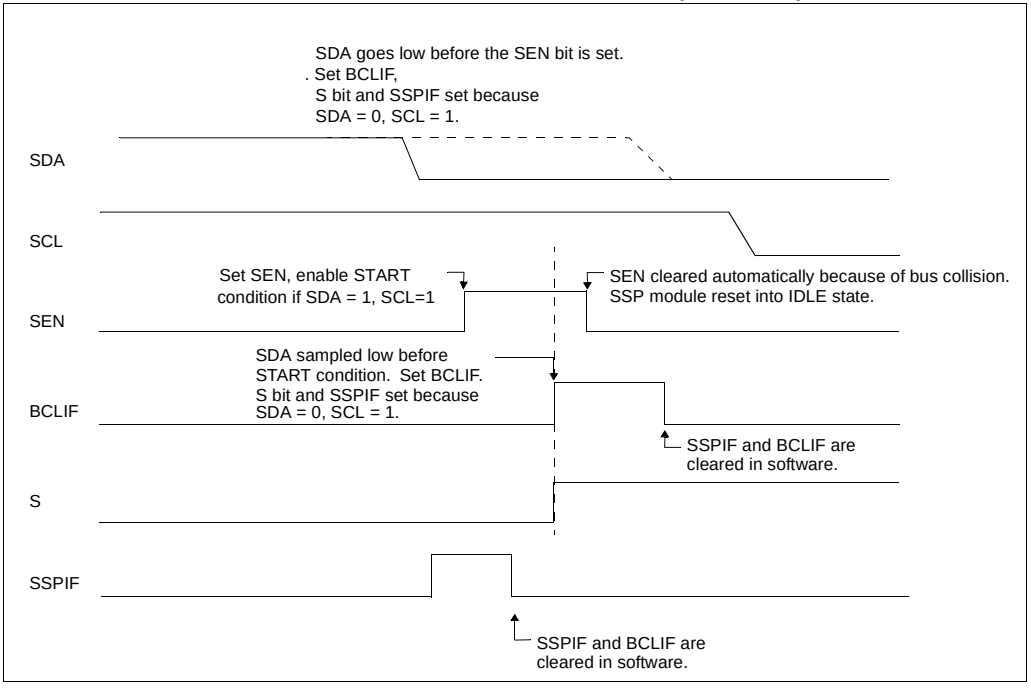


FIGURE 15-22: BUS COLLISION DURING START CONDITION (SCL = 0)

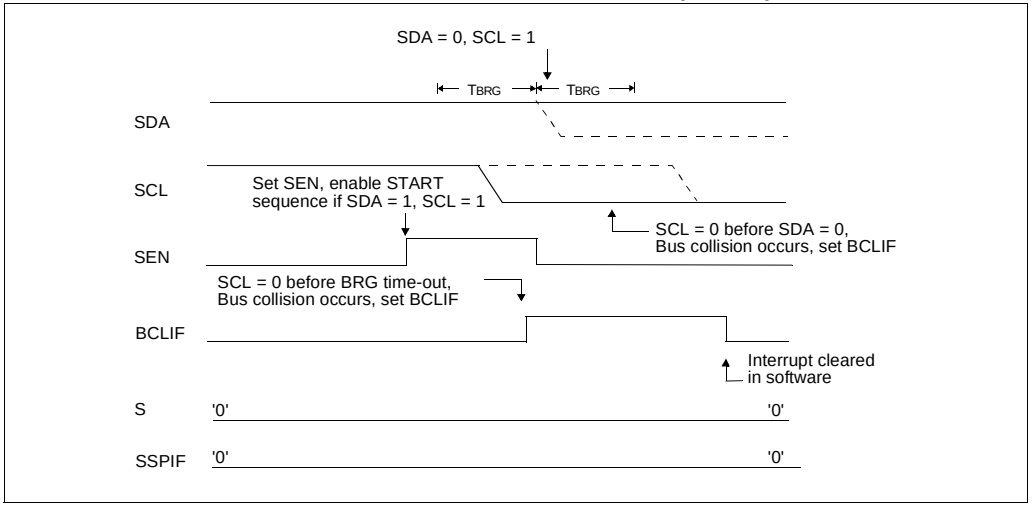
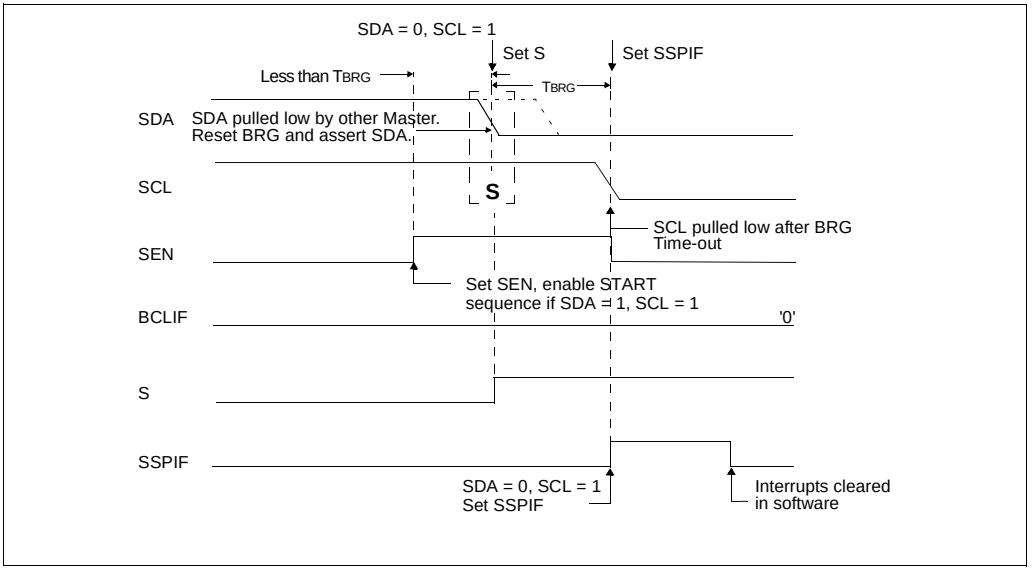


FIGURE 15-23: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



15.4.16.2 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.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user de-asserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then de-asserted 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', see Figure 15-24). 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 (Figure 15-25).

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 15-24: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

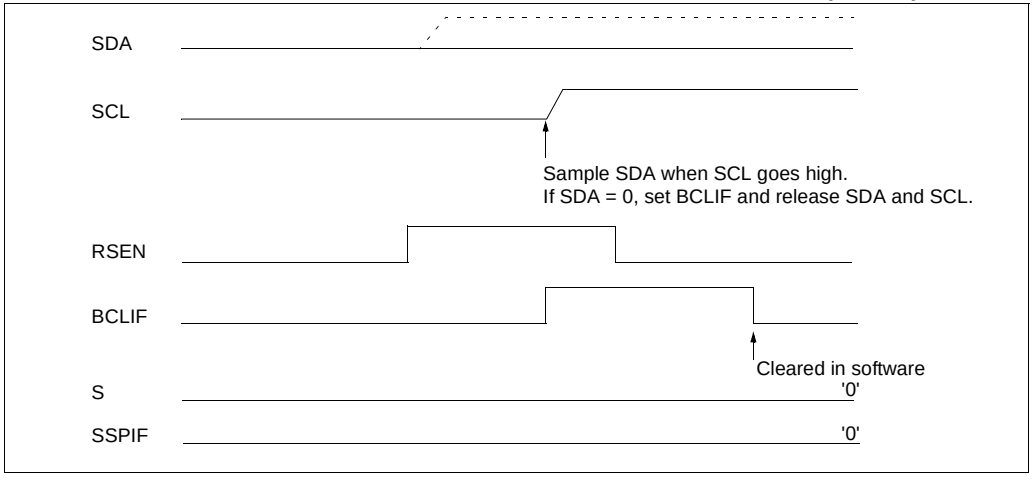
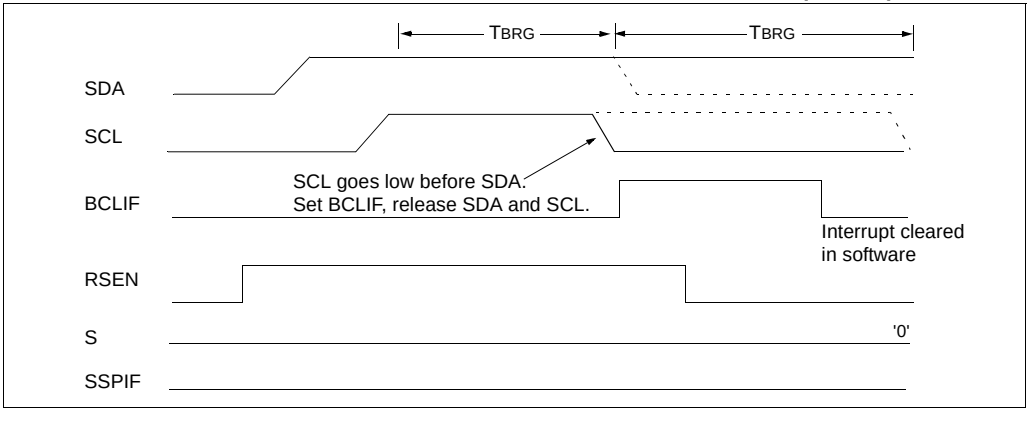


FIGURE 15-25: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



15.4.16.3 Bus Collision During a STOP Condition

Bus collision occurs during a STOP condition if:

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

b) After the SCL pin is de-asserted, SCL is sampled low before SDA goes high.

The STOP condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the baud rate generator is loaded with SSPADD<6:0> 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 15-26). 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 15-27).

FIGURE 15-26: BUS COLLISION DURING A STOP CONDITION (CASE 1)

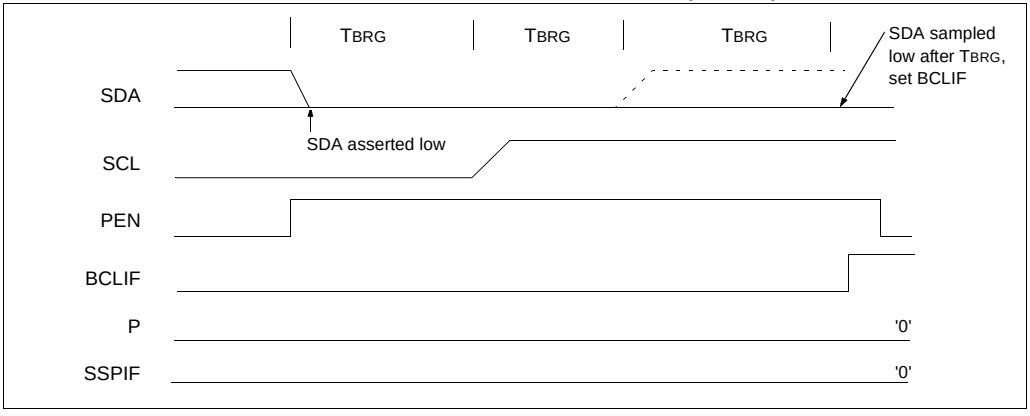
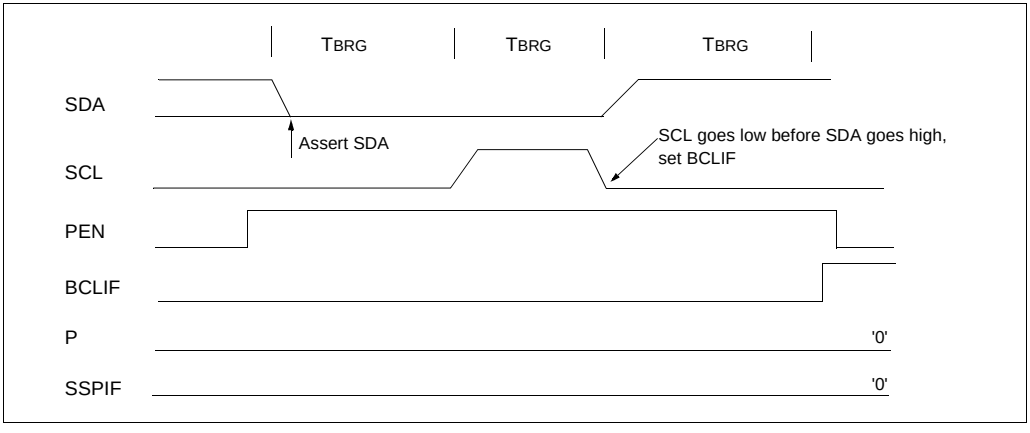


FIGURE 15-27: BUS COLLISION DURING A STOP CONDITION (CASE 2)



16.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

The SPEN (RCSTA register) and the TRISC<7> bits have to be set, and the TRISC<6> bit must be cleared, in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter.

Register 16-1 shows the Transmit Status and Control Register (TXSTA) and Register 16-2 shows the Receive Status and Control Register (RCSTA).

REGISTER 16-1: TXSTA REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (Clock generated internally from BRG)
0 = Slave mode (Clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

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

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data. Can be Address/Data bit or a parity bit.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 16-2: RCSTA REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (Configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - Slave:
Unused in this mode

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (Can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (Can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data. Can be Address/Data bit or a parity bit.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

16.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA register) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 16-1 shows the formula for computation of the baud rate for different USART modes, which only apply in Master mode (internal clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRG register can be calculated using the formula in Table 16-1. From this, the error in baud rate can be determined.

Example 16-1 shows the calculation of the baud rate error for the following conditions:

FOSC = 16 MHz
Desired Baud Rate = 9600
BRGH = 0
SYNC = 0

It may be advantageous to use the high baud rate (BRGH = 1), even for slower baud clocks. This is because the FOSC/(16(X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

16.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

EXAMPLE 16-1: CALCULATING BAUD RATE ERROR

Desired Baud Rate = FOSC / (64 (X + 1))

Solving for X:

X = ((FOSC / Desired Baud Rate) / 64) - 1
X = ((16000000 / 9600) / 64) - 1
X = [25.042] = 25

Calculated Baud Rate = 16000000 / (64 (25 + 1))
= 9615

Error = (Calculated Baud Rate - Desired Baud Rate) / Desired Baud Rate
= (9615 - 9600) / 9600
= 0.16%

TABLE 16-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FOSC/(64(X+1))	Baud Rate = FOSC/(16(X+1))
1	(Synchronous) Baud Rate = FOSC/(4(X+1))	NA

Legend: X = value in SPBRG (0 to 255)

TABLE 16-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

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
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 16-3: BAUD RATES FOR SYNCHRONOUS MODE

BAUD RATE (Kbps)	Fosc =25 MHz			20 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-
19.2	NA	-	-	NA	-	-
76.8	77.16	+0.47	80	76.92	+0.16	64
96	96.15	+0.16	64	96.15	+0.16	51
300	297.62	-0.79	20	294.12	-1.96	16
500	480.77	-3.85	12	500	0	9
HIGH	6250	-	0	5000	-	0
LOW	24.41	-	255	19.53	-	255

BAUD RATE (Kbps)	Fosc = 16 MHz			10 MHz			7.15909 MHz			5.0688 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	9.62	+0.23	185	9.60	0	131
19.2	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92	19.20	0	65
76.8	76.92	+0.16	51	75.76	-1.36	32	77.82	+1.32	22	74.54	-2.94	16
96	95.24	-0.79	41	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	307.70	+2.56	12	312.50	+4.17	7	298.35	-0.57	5	NA	-	-
500	500	0	7	500	0	4	NA	-	-	NA	-	-
HIGH	4000	-	0	2500	-	0	1789.80	-	0	1267.20	-	0
LOW	15.63	-	255	9.77	-	255	6.99	-	255	4.95	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-		NA	-		NA	-		0.30	+1.14	
1.2	NA	-	-	NA	-	-	1.20	+0.16	207	1.17	-2.48	6
2.4	NA	-	-	NA	-	-	2.40	+0.16	103	NA	-	-
9.6	9.62	+0.16	103	9.62	+0.23	92	9.62	+0.16	25	NA	-	-
19.2	19.23	+0.16	51	19.04	-0.83	46	19.23	+0.16	12	NA	-	-
76.8	76.92	+0.16	12	74.57	-2.90	11	NA	-	-	NA	-	-
96	1000	+4.17	9	99.43	+3.57	8	NA	-	-	NA	-	-
300	NA	-	-	298.30	-0.57	2	NA	-	-	NA	-	-
500	500	0	1	NA	-	-	NA	-	-	NA	-	-
HIGH	1000	-	0	894.89	-	0	250	-	0	8.20	-	0
LOW	3.91	-	255	3.50	-	255	0.98	-	255	0.03	-	255

TABLE 16-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (Kbps)	Fosc = 25 MHz			20 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-		NA	-	
1.2	NA	-	-	NA	-	-
2.4	2.40	-0.15	162	2.40	+0.16	129
9.6	9.53	-0.76	40	9.47	-1.36	32
19.2	19.53	+1.73	19	19.53	+1.73	15
76.8	78.13	+1.73	4	78.13	+1.73	3
96	97.66	+1.73	3	NA	-	-
300	NA	-	-	312.50	+4.17	0
500	NA	-	-	NA	-	-
HIGH	390.63	-	0	312.50	-	0
LOW	1.53	-	255	1.22	-	255

BAUD RATE (Kbps)	Fosc = 16 MHz			10 MHz			7.15909 MHz			5.0688 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-		NA	-		NA	-		NA	-	
1.2	1.20	+0.16	207	1.20	+0.16	129	1.20	+0.23	92	1.20	0	65
2.4	2.40	+0.16	103	2.40	+0.16	64	2.38	-0.83	46	2.40	0	32
9.6	9.62	+0.16	25	9.77	+1.73	15	9.32	-2.90	11	9.90	+3.13	7
19.2	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5	19.80	+3.13	3
76.8	NA	-	-	78.13	+1.73	1	NA	-	-	79.20	+3.13	0
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	250	-	0	156.25	-	0	111.86	-	0	79.20	-	0
LOW	0.98	-	255	0.61	-	255	0.44	-	255	0.31	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.30	-0.16		0.30	+0.23		0.30	+0.16		NA	-	
1.2	1.20	+1.67	51	1.19	-0.83	46	1.20	+0.16	12	NA	-	-
2.4	2.40	+1.67	25	2.43	+1.32	22	NA	-	-	NA	-	-
9.6	NA	-	-	9.32	-2.90	5	NA	-	-	NA	-	-
19.2	NA	-	-	18.64	-2.90	2	NA	-	-	NA	-	-
76.8	NA	-	-	NA	-	-	NA	-	-	NA	-	-
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	62.50	-	0	55.93	-	0	15.63	-	0	0.51	-	0
LOW	0.24	-	255	0.22	-	255	0.06	-	255	0.002	-	255

TABLE 16-5: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (Kbps)	FOSC = 25 MHz			20 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-		NA	-	
1.2	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-
9.6	9.59	-0.15	162	9.62	+0.16	129
19.2	19.30	+0.47	80	19.23	+0.16	64
76.8	78.13	+1.73	19	78.13	+1.73	15
96	97.66	+1.73	15	96.15	+0.16	12
300	312.50	+4.17	4	312.50	+4.17	3
500	520.83	+4.17	2	NA	-	-
HIGH	1562.50	-	0	1250	-	0
LOW	6.10	-	255	4.88	-	255

BAUD RATE (Kbps)	FOSC = 16 MHz			10 MHz			7.15909 MHz			5.0688 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-		NA	-		NA	-		NA	-	
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	2.41	+0.23	185	2.40	0	131
9.6	9.62	+0.16	103	9.62	+0.16	64	9.52	-0.83	46	9.60	0	32
19.2	19.23	+0.16	51	18.94	-1.36	32	19.45	+1.32	22	18.64	-2.94	16
76.8	76.92	+0.16	12	78.13	+1.73	7	74.57	-2.90	5	79.20	+3.13	3
96	100	+4.17	9	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	312.50	+4.17	1	NA	-	-	NA	-	-
500	500	0	1	NA	-	-	NA	-	-	NA	-	-
HIGH	1000	-	0	625	-	0	447.44	-	0	316.80	-	0
LOW	3.91	-	255	2.44	-	255	1.75	-	255	1.24	-	255

BAUD RATE (Kbps)	FOSC = 4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	0.30	+0.16	207	0.29	-2.48	6
1.2	1.20	+0.16	207	1.20	+0.23	185	1.20	+0.16	51	NA	-	-
2.4	2.40	+0.16	103	2.41	+0.23	92	2.40	+0.16	25	NA	-	-
9.6	9.62	+0.16	25	9.73	+1.32	22	NA	-	-	NA	-	-
19.2	19.23	+0.16	12	18.64	-2.90	11	NA	-	-	NA	-	-
76.8	NA	-	-	74.57	-2.90	2	NA	-	-	NA	-	-
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	250	-	0	55.93	-	0	62.50	-	0	2.05	-	0
LOW	0.98	-	255	0.22	-	255	0.24	-	255	0.008	-	255

16.2 USART Asynchronous Mode

In this mode, data is transmitted in non-return-to-zero (NRZ) format. Data consists of one START bit, eight or nine data bits and one STOP bit. Data is transmitted in serial fashion with LSb first. An on-chip 8-bit baud rate generator can be programmed to generate the desired baud rate. The baud rate generator produces a clock, either x16 or x64 of the bit shift rate, depending on the BRGH bit (TXSTA register). USART does not automatically calculate the parity bit for the given data byte. If parity is to be transmitted, USART must be programmed to transmit nine bits and software must set/clear ninth data bit as parity bit. Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing the SYNC bit (TXSTA register).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

16.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 16-1. The heart of the transmitter is the Transmit (serial) Shift Register (TSR). The TSR register obtains its data from the Read/Write Transmit Buffer register (TXREG). The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR registers) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE registers). Flag bit TXIF will be set, regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicated the status of the TXREG register, another bit TRMT (TXSTA register) shows the status of the TSR register. Status bit TRMT is a read only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

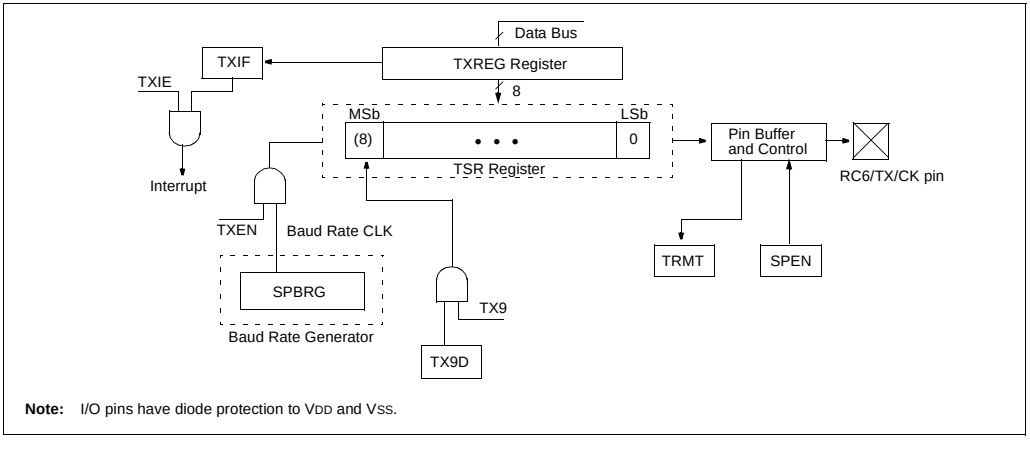
Note 1: The TSR register is not mapped in data memory, so it is not available to the user.

2: Flag bit TXIF is set when enable bit TXEN is set.

Steps to follow when setting up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 16.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

FIGURE 16-1: USART TRANSMIT BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

FIGURE 16-2: ASYNCHRONOUS TRANSMISSION

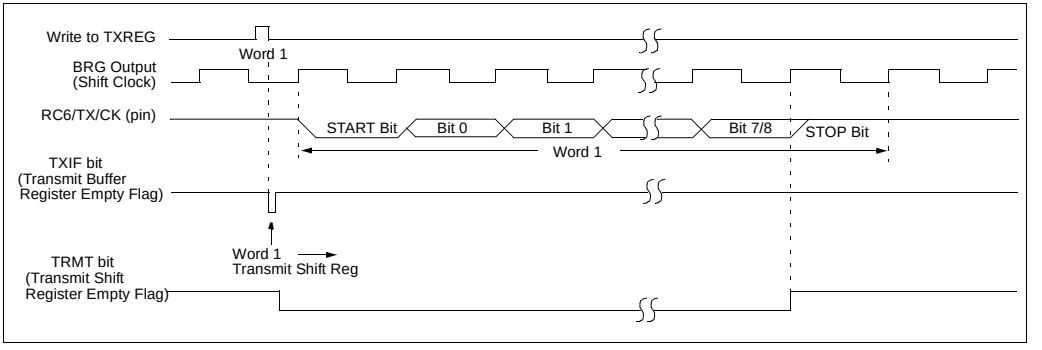
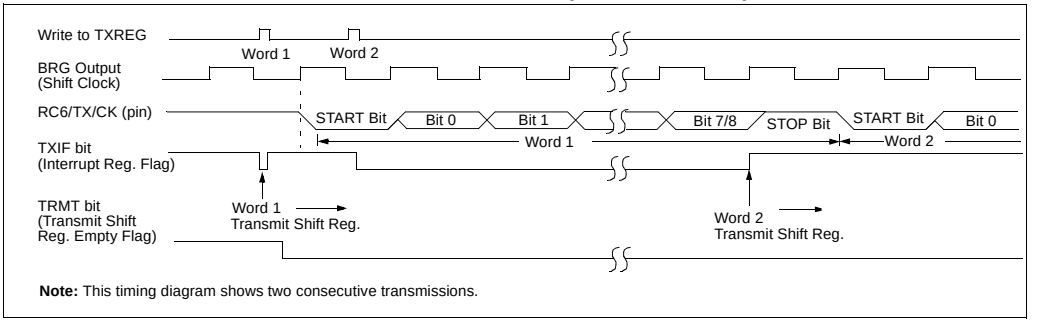


FIGURE 16-3: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 16-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	ADDEN	BRGH	TRMT	TX9D	0000 0010	0000 0010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Asynchronous Transmission.

16.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 16-4. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter, operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

Steps to follow when setting up an Asynchronous Reception:

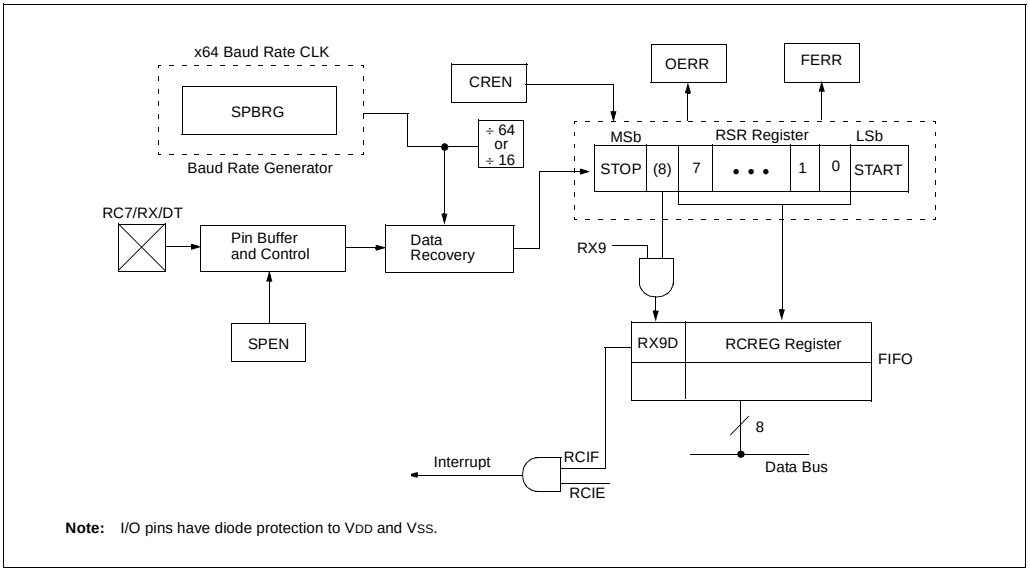
1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 16.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCIE.
4. If 9-bit reception is desired, set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.

16.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. Steps to follow when setting up an Asynchronous Reception with Address Detect Enable:

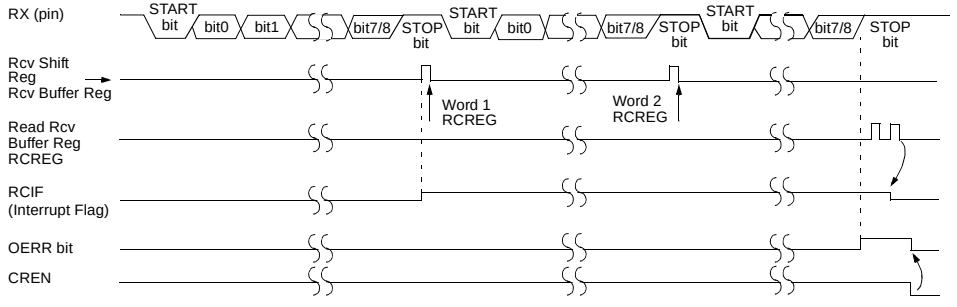
1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is required, set the BRGH bit.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCIF bit will be set when reception is complete. The interrupt will be acknowledged if the RCIE and GIE bits are set.
8. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 16-4: USART RECEIVE BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

FIGURE 16-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

TABLE 16-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	ADDEN	BRGH	TRMT	TX9D	0000 0010	0000 0010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for Asynchronous Reception.

16.3 USART Synchronous Master Mode

In Synchronous Master mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTA register). In addition, enable bit SPEN (RCSTA register) is set, in order to configure the RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting bit CSRC (TXSTA register).

16.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 16-1. The heart of the transmitter is the Transmit (serial) Shift register (TSR). The shift register obtains its data from the Read/Write Transmit Buffer register (TXREG). The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG is empty and interrupt bit TXIF (PIR registers) is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE registers). Flag bit TXIF will be set, regardless of the state of enable bit TXIE, and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA register) shows the status of the TSR register. TRMT is a read only bit, which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory, so it is not available to the user.

Steps to follow when setting up a Synchronous Master Transmission:

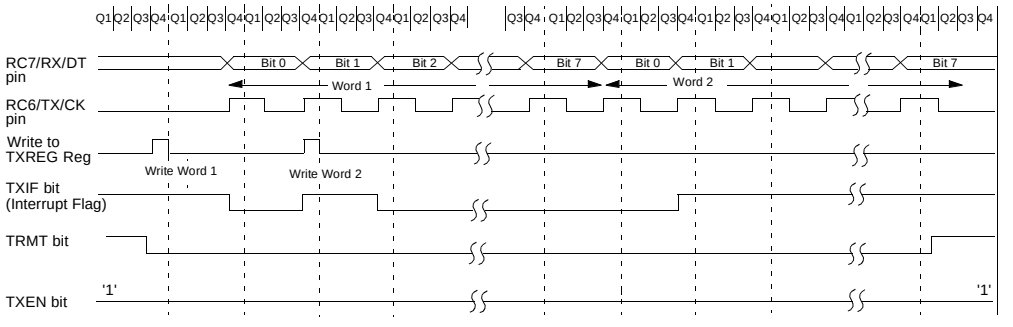
1. Initialize the SPBRG register for the appropriate baud rate (Section 16.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.

TABLE 16-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	ADDEN	BRGH	TRMT	TX9D	0000 0010	0000 0010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

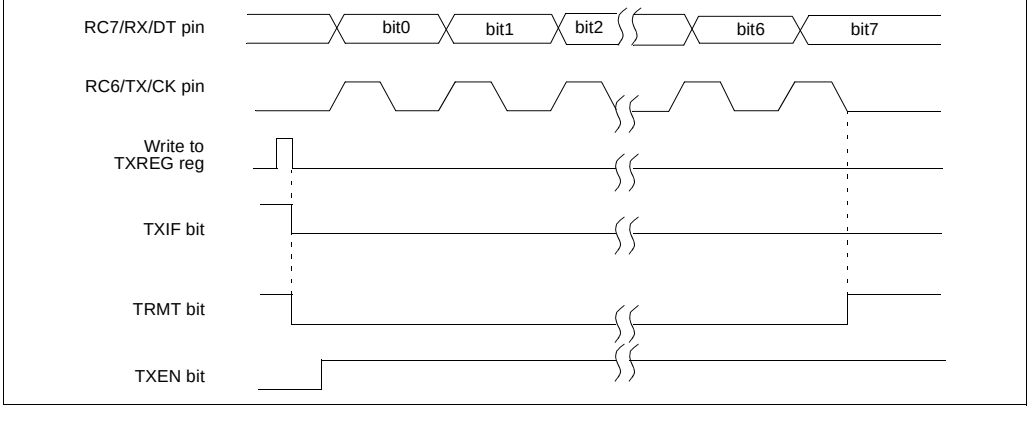
Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for Synchronous Master Transmission.

FIGURE 16-6: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode; SPBRG = '0'; continuous transmission of two 8-bit words.

FIGURE 16-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



16.3.2 USART SYNCHRONOUS MASTER RECEPTION

Once Synchronous Master mode is selected, reception is enabled by setting either enable bit SREN (RCSTA register), or enable bit CREN (RCSTA register). Data is sampled on the RC7/RX/DT pin on the falling edge of the clock. If enable bit SREN is set, only a single word is received. If enable bit CREN is set, the reception is continuous until CREN is cleared. If both bits are set, then CREN takes precedence.

When setting up a Synchronous Master reception, follow these steps:

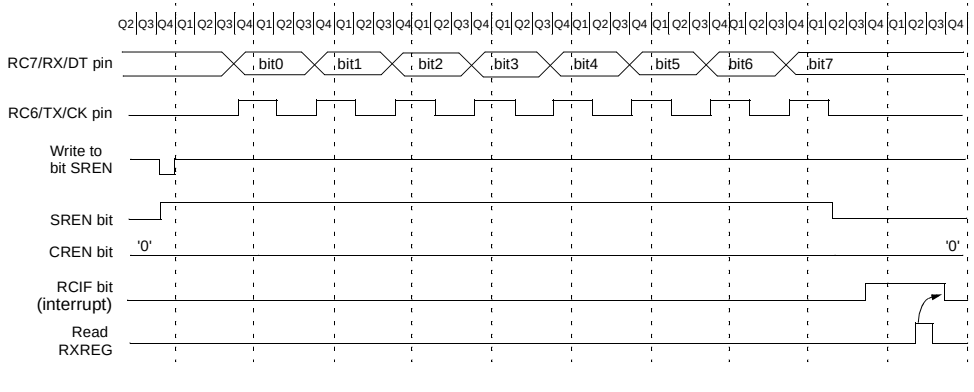
1. Initialize the SPBRG register for the appropriate baud rate (Section 16.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, set enable bit RCIE.
5. If 9-bit reception is desired, set bit RX9.
6. If a single reception is required, set bit SREN. For continuous reception, set bit CREN.
7. Interrupt flag bit RCIF will be set when reception is complete and an interrupt will be generated if the enable bit RCIE was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If any error occurred, clear the error by clearing bit CREN.

TABLE 16-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	ADDEN	BRGH	TRMT	TX9D	0000 0010	0000 0010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for Synchronous Master Reception.

FIGURE 16-8: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



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

16.4 USART Synchronous Slave Mode

Synchronous Slave mode differs from the Master mode, in that the shift clock is supplied externally at the RC6/TX/CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in SLEEP mode. Slave mode is entered by clearing bit CSRC (TXSTA register).

16.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical, except in the case of the SLEEP mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in TXREG register.
c) Flag bit TXIF will not be set.
d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit TXIF will be set.
e) If enable bit TXIE is set, the interrupt will wake the chip from SLEEP. If the global interrupt is enabled, the program will branch to the interrupt vector.

When setting up a Synchronous Slave Transmission, follow these steps:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.

16.4.2 USART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of the SLEEP mode and bit SREN, which is a "don't care" in Slave mode.

If receive is enabled by setting bit CREN prior to the SLEEP instruction, then a word may be received during SLEEP. On completely receiving the word, the RSR register will transfer the data to the RCREG register, and if enable bit RCIE bit is set, the interrupt generated will wake the chip from SLEEP. If the global interrupt is enabled, the program will branch to the interrupt vector.

When setting up a Synchronous Slave Reception, follow these steps:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit RCIF will be set when reception is complete. An interrupt will be generated if enable bit RCIE was set.
6. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.

TABLE 16-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	ADDEN	BRGH	TRMT	TX9D	0000 0010	0000 0010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for Synchronous Slave Transmission.

TABLE 16-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	ADDEN	BRGH	TRMT	TX9D	0000 0010	0000 0010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for Synchronous Slave Reception.

NOTES:

17.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The analog-to-digital (A/D) converter module has 8 inputs for the PIC18C601 devices and 12 for the PIC18C801 devices. This module has the ADCON0, ADCON1, and ADCON2 registers.

The A/D allows conversion of an analog input signal to a corresponding 10-bit digital number.

The A/D module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 17-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 17-2, configures the functions of the port pins. The ADCON2, shown in Register 16-3, configures the A/D clock source and justification.

REGISTER 17-1: ADCON0 REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

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

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits

0000 = channel 00, (AN0)
0001 = channel 01, (AN1)
0010 = channel 02, (AN2)
0011 = channel 03, (AN3)
0100 = channel 04, (AN4)
0101 = channel 05, (AN5)
0110 = channel 06, (AN6)
0111 = channel 07, (AN7)
1000 = channel 08, (AN8)**(1)**
1001 = channel 09, (AN9)**(1)**
1010 = channel 10, (AN10)**(1)**
1011 = channel 11, (AN11)**(1)**
1100 = Reserved
1101 = Reserved
1110 = Reserved
1111 = Reserved

These channels are not available on the PIC18C601 devices.

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

When ADON = 1:
1 = A/D conversion in progress. Setting this bit starts an A/D conversion cycle. This bit is automatically cleared by hardware when the A/D conversion is complete.
0 = A/D conversion not in progress

bit 0 **ADON:** A/D On bit

1 = A/D converter module is operating
0 = A/D converter module is shut-off and consumes no operating current

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 17-2: ADCON1 REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

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

bit 5-4 **VCFG1:VCFG0:** Voltage Reference Configuration bits

	A/D VREF+	**A/D VREF-**
00	AVDD	AVSS
01	External VREF+	AVSS
10	AVDD	External VREF-
11	External VREF+	External VREF-

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

	AN11	**AN10**	**AN9**	**AN8**	**AN7**	**AN6**	**AN5**	**AN4**	**AN3**	**AN2**	**AN1**	**AN0**
0000	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A
0011	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Shaded cells = Additional A/D channels available on PIC18C801 devices.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 17-3: ADCON2 REGISTER

R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	—	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6-3 **Unimplemented:** Read as '0'

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
011 = FRC (clock derived from an internal RC oscillator = 1 MHz max)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64
111 = FRC (clock derived from an internal RC oscillator = 1 MHz max)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (V_{DD} and V_{SS}), or the voltage level on the RA3/AN3/V_{REF}+ pin and RA2/AN2/V_{REF}-.

The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode. To operate in SLEEP, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

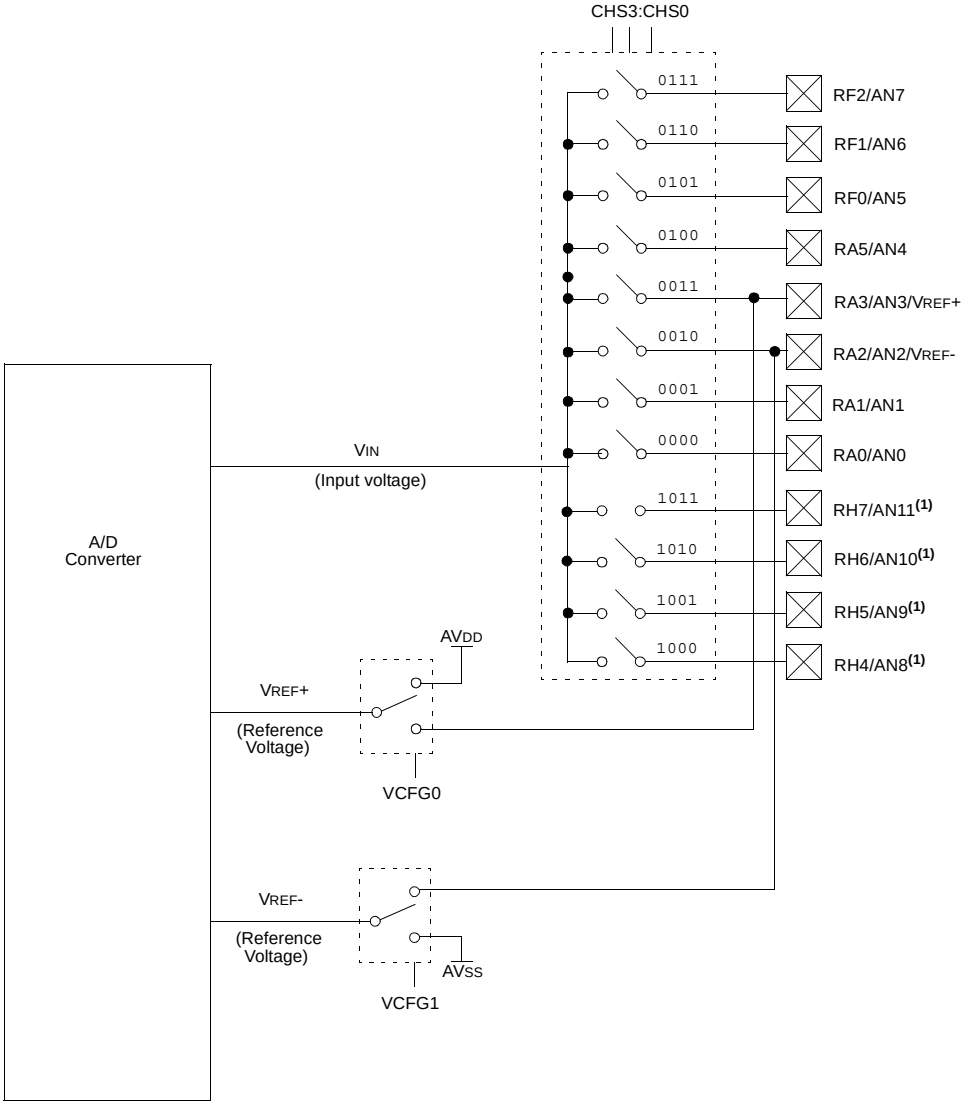
The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off and any conversion is aborted.

Each port pin associated with the A/D converter can be configured as an analog input (RA3 can also be a voltage reference), or as a digital I/O.

The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH/ADRESL registers, the GO/$\overline{\text{DONE}}$ bit (ADCON0 register) is cleared, and A/D interrupt flag bit ADIF is set. The block diagram of the A/D module is shown in Figure 17-1.

FIGURE 17-1: A/D BLOCK DIAGRAM



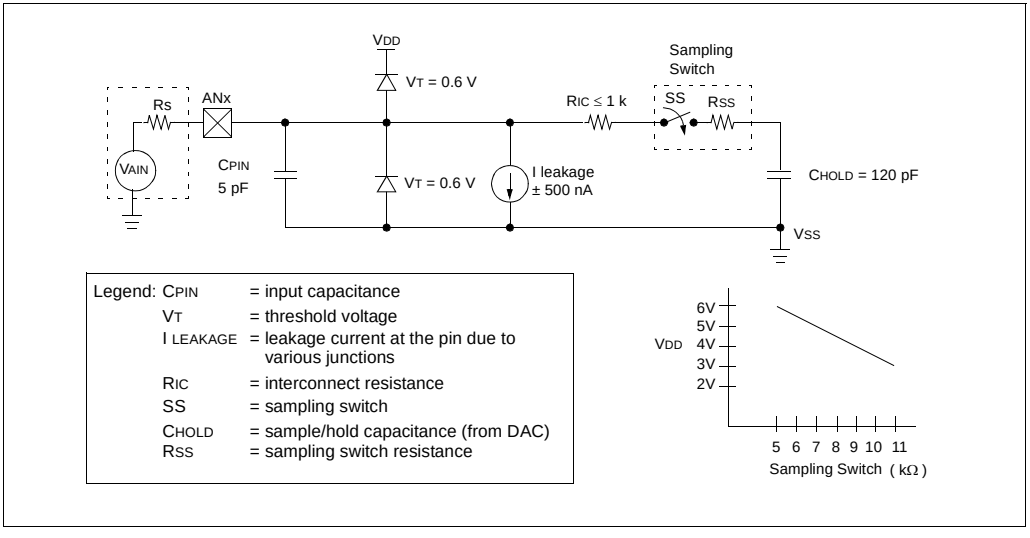
Note 1: These channels are not available on the PIC18C601 devices.

The value in the ADRESH/ADRESL registers is not modified for a Power-on Reset. The ADRESH/ADRESL registers will contain unknown data after a Power-on Reset.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see Section 17.1. After this acquisition time has elapsed, the A/D conversion can be started. The following steps should be followed to do an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared, OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2TAD is required before next acquisition starts.

FIGURE 17-2: ANALOG INPUT MODEL



17.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 17-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD). The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is 2.5kΩ**. After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

Note:	When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 17-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Example 17-1 shows the calculation of the minimum required acquisition time TACQ. This calculation is based on the following application system assumptions:

CHOLD	=	120 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSb
VDD	=	5V → Rss = 7 kΩ
Temperature	=	50°C (system max.)
VHOLD	=	0V @ time = 0

EQUATION 17-1: ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

EQUATION 17-2: A/D MINIMUM CHARGING TIME

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(120\ pF)(1\ k\Omega + R_{SS} + R_S) \ln(1/2047)$$

EXAMPLE 17-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

Temperature coefficient is only required for temperatures > 25°C.

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

$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S) \ln(1/2047)$
$-120\ pF\ (1\ k\Omega + 7\ k\Omega + 2.5\ k\Omega) \ln(0.0004885)$
$-120\ pF\ (10.5\ k\Omega) \ln(0.0004885)$
$-1.26\ \mu s\ (-7.6241)$
$9.61\ \mu s$

$T_{ACQ} = 2\ \mu s + 9.61\ \mu s + [(50°C - 25°C)(0.05\ \mu s/°C)]$
$11.61\ \mu s + 1.25\ \mu s$
$12.86\ \mu s$

17.2 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 12 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. There are seven possible options for TAD:

- 2TOSC
- 4TOSC
- 8TOSC
- 16TOSC
- 32TOSC
- 64TOSC
- Internal RC oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6 μs.

Table 17-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

17.3 Configuring Analog Port Pins

The ADCON1, TRISA, TRISF and TRISH registers control the operation of the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS3:CHS0 bits and the TRIS bits.

Note 1: When reading the port register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.

2: Analog levels on any pin defined as a digital input may cause the input buffer to consume current out of the device's specification limits.

TABLE 17-1: TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	**ADCS2:ADCS0**	**PIC18C601/801**	**PIC18LC601/801(5)**
2TOSC	000	1.25 MHz	666 kHz
4TOSC	100	2.50 MHz	1.33 MHz
8TOSC	001	5.00 MHz	2.67 MHz
16TOSC	101	10.0 MHz	5.33 MHz
32TOSC	010	20.0 MHz	10.67 MHz
64TOSC	110	—	—
RC	x11	—	—

Note 1: The RC source has a typical TAD time of 4 μs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion or the A/D accuracy may be out of specification.
5: This column is for the **LC** devices only.

17.4 A/D Conversions

Figure 17-3 shows the operation of the A/D converter after the GO bit has been set. Clearing the GO/DONE bit during a conversion will abort the current conversion. The A/D result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, acquisition on the selected channel is automatically started.

Note:	The GO/DONE bit should **NOT** be set in the same instruction that turns on the A/D.

17.5 Use of the CCP2 Trigger

An A/D conversion can be started by the "special event trigger" of the CCP2 module. This requires that the CCP2M3:CCP2M0 bits (CCP2CON<3:0>) be programmed as 1011, and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/DONE bit will be set, starting the A/D conversion and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition done before the "special event trigger" sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the "special event trigger" will be ignored by the A/D module, but will still reset the Timer1 (or Timer3) counter.

FIGURE 17-3: A/D CONVERSION TAD CYCLES

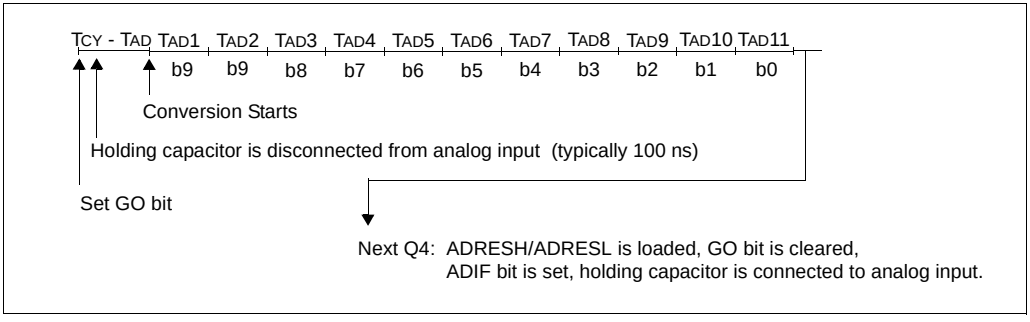


TABLE 17-2: SUMMARY OF A/D REGISTERS

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 0000
PIR2	—	—	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-- 0000	-0-- 0000
PIE2	—	—	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	---- 0000	---- 0000
IPR2	—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP	---- 0000	---- 0000
ADRESH	A/D Result Register								xxxx xxxx	uuuu uuuu
ADRESL	A/D Result Register								xxxx xxxx	uuuu uuuu
ADCON0	—	—	CHS3	CHS3	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	0000 00-0	0000 00-0
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	---- -000	---- -000
ADCON2	ADFM	—	—	—	—	ADCS2	ADCS1	ADCS0	0--- -000	0--- -000
PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
TRISA	—	PORTA Data Direction Register							--11 1111	--11 1111
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	x000 0000	u000 0000
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	xxxx xxxx	uuuu uuuu
TRISF	PORTF Data Direction Control Register								1111 1111	1111 1111
PORTH(1)	RH7	RH6	RH5	RH4	RH3	RH2	RH1	RH0	0000 xxxx	0000 xxxx
LATH(1)	LATH7	LATH6	LATH5	LATH4	LATH3	LATH2	LATH1	LATH0	xxxx xxxx	uuuu uuuu
TRISH(1)	PORTH Data Direction Control Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.
Note 1: Only available on PIC18C801 devices.

NOTES:

18.0 LOW VOLTAGE DETECT

In many applications, the ability to determine if the device voltage (V_{DD}) is below a specified voltage level is a desirable feature. A window of operation for the application can be created, where the application software can do "housekeeping tasks", before the device voltage exits the valid operating range. This can be done using the Low Voltage Detect module.

This module is software programmable circuitry, where a device voltage trip point can be specified (internal reference voltage or external voltage input). When the voltage of the device becomes lower than the specified point, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to that interrupt source.

The Low Voltage Detect circuitry is completely under software control. This allows the circuitry to be "turned off" by the software, which minimizes the current consumption for the device.

Figure 18-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage V_A, the LVD logic generates an interrupt. This occurs at time T_A. The application software then has the time, until the device voltage is no longer in valid operating range, to shut-down the system. Voltage point V_B is the minimum valid operating voltage specification. This occurs at time T_B. T_B - T_A is the total time for shut-down.

FIGURE 18-1: TYPICAL LOW VOLTAGE DETECT APPLICATION

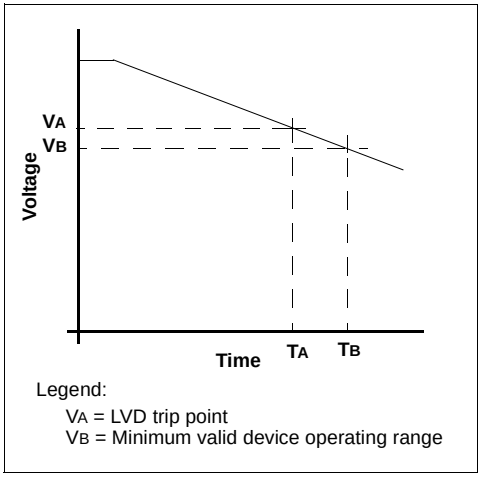
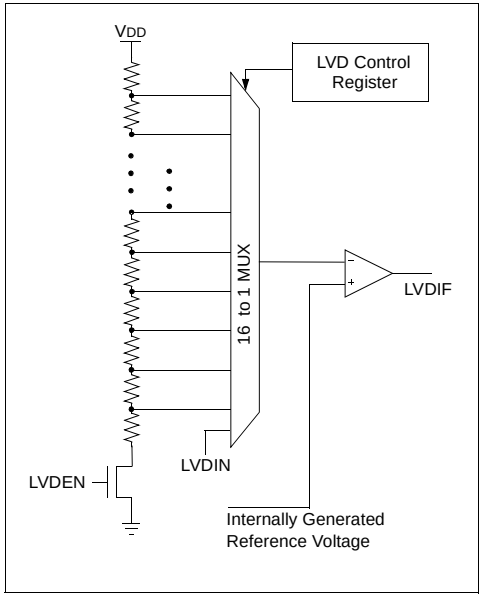


Figure 18-2 shows the block diagram for the LVD module. A comparator uses an internally generated reference voltage as the set point. When the selected tap output of the device voltage crosses the set point (is lower than), the LVDIF bit (PIR registers) is set.

Each node in the resister divider represents a “trip point” voltage. The “trip point” voltage is the minimum supply voltage level at which the device can operate, before the LVD module asserts an interrupt. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array (or external LVDIN input pin) is equal to the voltage generated by the internal voltage reference module. The comparator then generates an interrupt signal setting the LVDIF bit. This voltage is software programmable to any one of 16 values (see Figure 18-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

FIGURE 18-2: LOW VOLTAGE DETECT (LVD) BLOCK DIAGRAM



18.1 Control Register

The Low Voltage Detect Control register (Register 18-1) controls the operation of the Low Voltage Detect circuitry.

REGISTER 18-1: LVDCON REGISTER

U-0	U-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0
bit 7							bit 0

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

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit

1 = Indicates that the Low Voltage Detect logic will generate the interrupt flag at the specified voltage range

0 = Indicates that the Low Voltage Detect logic will not generate the interrupt flag at the specified voltage range and the LVD interrupt should not be enabled

bit 4 **LVDEN:** Low Voltage Detect Power Enable bit

1 = Enables LVD, powers up LVD circuit
0 = Disables LVD, powers down LVD circuit

bit 3-0 **LVDL3:LVDL0:** Low Voltage Detection Limit bits

1111 = External analog input is used (input comes from the LVDIN pin)
1110 = 4.5V
1101 = 4.2V
1100 = 4.0V - Reserved on PIC18C601/801
1011 = 3.8V - Reserved on PIC18C601/801
1010 = 3.6V - Reserved on PIC18C601/801
1001 = 3.5V - Reserved on PIC18C601/801
1000 = 3.3V - Reserved on PIC18C601/801
0111 = 3.0V - Reserved on PIC18C601/801
0110 = 2.8V - Reserved on PIC18C601/801
0101 = 2.7V - Reserved on PIC18C601/801
0100 = 2.5V - Reserved on PIC18C601/801
0011 = 2.4V - Reserved on PIC18C601/801
0010 = 2.2V - Reserved on PIC18C601/801
0001 = 2.0V - Reserved on PIC18C601/801
0000 = Reserved on PIC18C601/801 and PIC18LC801/601

LVDL3:LVDL0 modes which result in a trip point below the valid operating voltage of the device are not tested.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

18.2 Operation

Depending on the power source for the device voltage, the voltage normally decreases relatively slowly. This means that the LVD module does not need to be constantly operating. To decrease current consumption, the LVD circuitry only needs to be enabled for short periods, where the voltage is checked. After doing the check, the LVD module may be disabled.

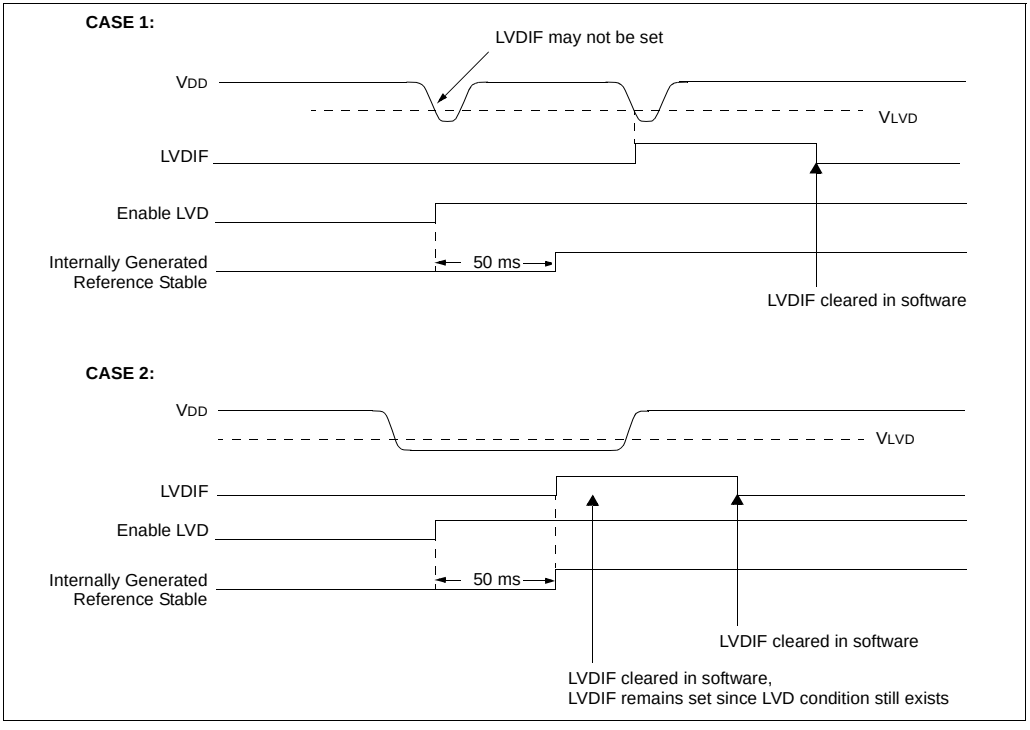
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

The following steps are needed to setup the LVD module:

1. Write the value to the LVDL3:LVDL0 bits (LVDCON register), which selects the desired LVD trip point.
2. Ensure that LVD interrupts are disabled (the LVDIE bit is cleared or the GIE bit is cleared).
3. Enable the LVD module (set the LVDEN bit in the LVDCON register).
4. Wait for the LVD module to stabilize (the IRVST bit to become set).
5. Clear the LVD interrupt flag, which may have falsely become set, until the LVD module has stabilized (clear the LVDIF bit).
6. Enable the LVD interrupt (set the LVDIE and the GIE bits).

Figure 18-3 shows typical waveforms that the LVD module may be used to detect.

FIGURE 18-3: LOW VOLTAGE DETECT WAVEFORMS



18.2.1 REFERENCE VOLTAGE SET POINT

The Internal Reference Voltage of the LVD module may be used by other internal circuitry (the programmable Brown-out Reset). If these circuits are disabled (lower current consumption), the reference voltage circuit requires time to become stable before a low voltage condition can be reliably detected. This time is invariant of system clock speed. This start-up time is specified in electrical specification parameter #36. The low voltage interrupt flag will not be enabled until a stable reference voltage is reached. Refer to the waveform in Figure 18-3.

18.2.2 CURRENT CONSUMPTION

When the module is enabled, the LVD comparator and voltage divider are enabled and will consume static current. The voltage divider can be tapped from multiple places in the resistor array. Total current consumption, when enabled, is specified in electrical specification parameter #D022B.

18.3 External Analog Voltage Input

The LVD module has an additional feature that allows the user to supply the trip point voltage to the module from an external source (the LVDIN pin). The LVDIN pin is used as the trip point when the LVDL3:LVDL0 bits equal '1111'. This state connects the LVDIN pin voltage to the comparator. The other comparator input is connected to an internal reference voltage source.

18.4 Operation During SLEEP

When enabled, the LVD circuitry continues to operate during SLEEP. If the device voltage crosses the trip point, the LVDIF bit will be set and the device will wake-up from SLEEP. Device execution will continue from the interrupt vector address, if interrupts have been globally enabled.

18.5 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the LVD module to be turned off.

19.0 SPECIAL FEATURES OF THE CPU

There are several features intended to maximize system reliability, minimize cost through elimination of external components and provide power saving operating modes:

- OSC Selection
- RESET
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- ID Locations

PIC18C601/801 devices have a Watchdog Timer, which can be permanently enabled/disabled via the configuration bits, or it can be software controlled. By default, the Watchdog Timer is disabled to allow software control. It runs off its own RC oscillator for cost reduction. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay on power-up only, designed to keep the part in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

SLEEP mode is designed to offer a very low current Power-down mode. The user can wake-up from SLEEP through external RESET, Watchdog Timer Wake-up or through an interrupt. Several oscillator options are also available to allow the part to fit the application. The RC oscillator option saves system cost, while the LP crystal option saves power. By default, HS oscillator mode is selected. There are two main modes of operations for external memory interface: 8-bit and 16-bit (default). A set of configuration bits are used to select various options.

19.1 Configuration Bits

The configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h - 3FFFFFh), which can only be accessed using table reads and table writes.

TABLE 19-1: CONFIGURATION BITS AND DEVICE IDs

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300001h	CONFIG1H	—	—	—	—	—	—	FOSC1	FOSC0	---- --11
300002h	CONFIG2L	—	BW	—	—	—	—	—	$\overline{\text{PWRTEN}}$	-1-- ---1
300003h	CONFIG2H	—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN	---- 1110
300006h	CONFIG4L	r	—	—	—	—	—	—	STVREN	1--- ---1
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	0000 0000
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved, maintain '1'.
Shaded cells are unimplemented, read as '0'.

REGISTER 19-1: CONFIGURATION REGISTER 1 HIGH (CONFIG1H: BYTE ADDRESS 0300001h)

U-0	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
—	—	—	—	—	—	FOSC1	FOSC0
bit 7							bit 0

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

bit 2-0 **FOSC1:FOSC0**: Oscillator Selection bits

11 = RC oscillator
10 = HS oscillator
01 = EC oscillator
00 = LP oscillator

Legend:		
r = Reserved		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 19-2: CONFIGURATION REGISTER 2 LOW (CONFIG2L: BYTE ADDRESS 300002h)

U-0	R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1
—	BW	—	—	—	—	—	$\overline{\text{PWRTEN}}$
bit 7							bit 0

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

bit 6 **BW:** External Bus Data Width bit

1 = 16-bit external bus mode
0 = 8-bit external bus mode

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

bit 0 **$\overline{\text{PWRTEN}}$:** Power-up Timer Enable bit

1 = PWRT disabled
0 = PWRT enabled

Legend:		
r = Reserved		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 19-3: CONFIGURATION REGISTER 2 HIGH (CONFIG2H: BYTE ADDRESS 300003H)

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3-1 **WDTPS2:WDTPS0:** Watchdog Timer Postscale Select bits

000 =1:128
001 =1:64
010 =1:32
011 =1:16
100 =1:8
101 =1:4
110 =1:2
111 =1:1

bit 0 **WDTEN:** Watchdog Timer Enable bit

1 = WDT enabled
0 = WDT disabled (control is placed on the SWDTEN bit)

Legend:		
r = Reserved		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 19-4: CONFIGURATION REGISTER 4 LOW (CONFIG4L: BYTE ADDRESS 300006H)

R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
r	—	—	—	—	—	—	STVREN
bit 7							bit 0

bit 7 **Reserved:** Maintain as '1'

bit 6-1 **Unimplemented:** Read as '0'

bit 0 **STVREN:** Stack Full/Underflow RESET Enable bit

1 = Stack Full/Underflow will cause RESET
0 = Stack Full/Underflow will not cause RESET

Legend:		
r = Reserved		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

19.2 Watchdog Timer (WDT)

The Watchdog Timer is a free running on-chip RC oscillator, which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKI pin. That means that the WDT will run, even if the clock on the OSC1/CLKI and OSC2/CLKO pins of the device has been stopped; for example, by execution of a SLEEP instruction.

During normal operation, a WDT time-out generates a device RESET (Watchdog Timer Reset). If the device is in SLEEP mode, a WDT time-out causes the device to wake-up and continue with normal operation (Watchdog Timer Wake-up). The $\overline{TO}$ bit in the RCON register will be cleared upon a WDT time-out.

By default, the Watchdog Timer is disabled by configuration to allow software control over Watchdog Timer operation. If the WDT is enabled by configuration, software execution may not disable this function. When the Watchdog Timer is disabled by configuration, the SWDTEN bit in the WDTCON register enables/disables the operation of the WDT.

The WDT time-out period values may be found in the Electrical Specifications section under parameter #31. Values for the WDT postscaler may be assigned by using configuration bits WDPS<3:1> in CONFIG2H register. If the Watchdog Timer is disabled by configuration, values for the WDT postscaler may be assigned using the SWDPS bits in the WDTCON register.

Note 1:	The CLRWDT and SLEEP instructions clear the WDT and the postscaler, if assigned to the WDT, and prevent it from timing out and generating a device RESET condition.
2:	When a CLRWDT instruction is executed and the prescaler is assigned to the WDT, the prescaler count will be cleared, but the prescaler assignment is not changed.

19.2.1 CONTROL REGISTER

Register 19-5 shows the WDTCON register. This is a readable and writable register. It contains control bits to control the Watchdog Timer from user software. If the Watchdog Timer is enabled by configuration, this register setting is ignored.

REGISTER 19-5: WDTCON REGISTER

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	SWDPS2	SWDPS1	SWDPS0	SWDTEN
bit 7							bit 0

bit 7-4 **Unimplemented**: Read as '0'

bit 3-1 **SWDPS2:SWDPS0:** Software Watchdog Timer Postscale Select bits
111 = 1:128
110 = 1:64
101 = 1:32
100 = 1:16
011 = 1:8
010 = 1:4
001 = 1:2
000 = 1:1

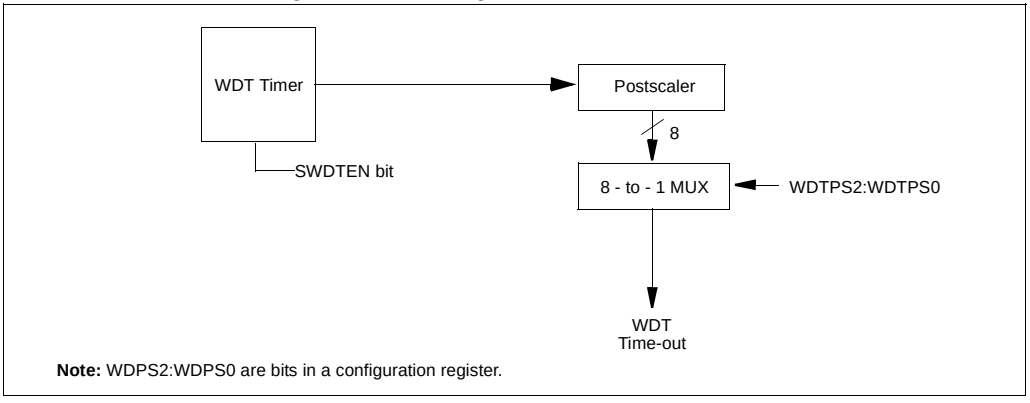
bit 0 **SWDTEN:** Software Controlled Watchdog Timer Enable bit
1 = Watchdog Timer is on
0 = Watchdog Timer is turned off if it is not disabled

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

19.2.2 WDT POSTSCALER

The WDT has a postscaler that can extend the WDT Reset period. The postscaler may be programmed by the user software or is selected by configuration bits WDTPS<2:0> in the CONFIG2H register. If the device has the Watchdog Timer enabled by configuration bits, the device will use predefined set postscaler value. If the device has the Watchdog Timer disabled by configuration bits, user software can set desired postscaler value. When the device has the Watchdog Timer enabled by configuration bits, by default, Watchdog postscaler of 1:128 is selected.

FIGURE 19-1: Watchdog Timer Block Diagram



Note: WDPS2:WDPS0 are bits in a configuration register.

TABLE 19-2: SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG2H	—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN
RCON	IPEN	r	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	r
WDTCON	—	—	—	—	SWDPS2	SWDPS1	SWDPS0	SWDTEN

Legend: Shaded cells are not used by the Watchdog Timer.

19.3 Power-down Mode (SLEEP)

Power-down mode is entered by executing a SLEEP instruction.

Upon entering into Power-down mode, the following actions are performed:

1. Watchdog Timer is cleared and kept running.
2. $\overline{PD}$ bit in RCON register is cleared.
3. $\overline{TO}$ bit in RCON register is set.
4. Oscillator driver is turned off.
5. I/O ports maintain the status they had before the SLEEP instruction was executed.

To achieve lowest current consumption, follow these steps before switching to Power-down mode:

1. Place all I/O pins at either VDD or VSS and ensure no external circuitry is drawing current from I/O pin.
2. Power-down A/D and external clocks.
3. Pull all hi-impedance inputs to high or low, externally.
4. Place T0CKI at VSS or VDD.
5. Current consumption by PORTB on-chip pull-ups should be taken into account and disabled, if necessary.

The $\overline{MCLR}$ pin must be at a logic high level (VIHMC).

19.3.1 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.
2. Watchdog Timer Wake-up (if WDT was enabled).
3. Interrupt from INT pin, RB port change, or a peripheral interrupt.

The following peripheral interrupts can wake the device from SLEEP:

4. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
5. TMR3 interrupt. Timer3 must be operating as an asynchronous counter.
6. CCP Capture mode interrupt.
7. Special event trigger (Timer1 in Asynchronous mode using an external clock).
8. MSSP (START/STOP) bit detect interrupt.
9. MSSP transmit or receive in Slave mode (SPI/I^2C).
10. USART RX or TX (Synchronous Slave mode).
11. A/D conversion (when A/D clock source is RC).

Other peripherals cannot generate interrupts, since during SLEEP, no on-chip clocks are present.

External $\overline{MCLR}$ Reset will cause a device RESET. All other events are considered a continuation of program execution and will cause a "wake-up". The $\overline{TO}$ and $\overline{PD}$ bits in the RCON register can be used to determine the cause of the device RESET. The $\overline{PD}$ bit, which is set on power-up, is cleared when SLEEP is invoked. The $\overline{TO}$ bit is cleared, if a WDT time-out occurred (and caused wake-up).

When the SLEEP instruction is being executed, the next instruction (PC + 2) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is set (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt address. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

19.3.2 WAKE-UP USING INTERRUPTS

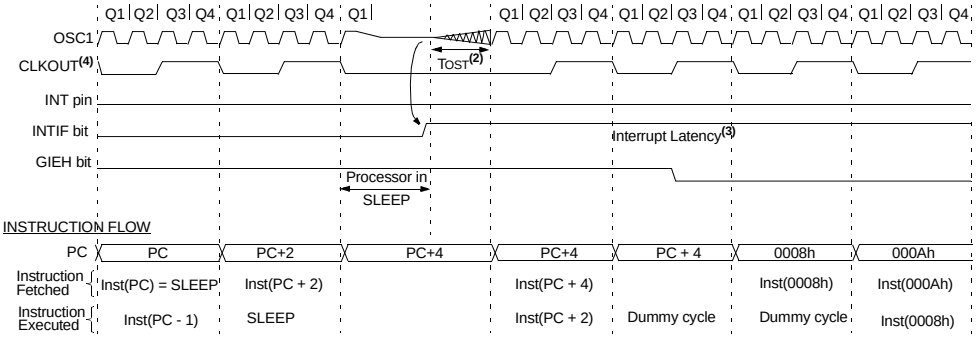
When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If an interrupt condition (interrupt flag bit and interrupt enable bits are set) occurs **before** the execution of a SLEEP instruction, the SLEEP instruction will complete as a NOP. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bits will not be cleared.
- If the interrupt condition occurs **during or after** the execution of a SLEEP instruction, the device will immediately wake-up from sleep. The SLEEP instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit 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.

To ensure that the WDT is cleared, a CLRWDT instruction should be executed before a SLEEP instruction.

FIGURE 19-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT(1,2)



Note 1: HS or LP oscillator mode assumed.
2: GIE set is assumed. In this case, after wake- up, the processor jumps to the interrupt routine. If GIE is cleared, execution will continue in-line.
3: TOST = 1024TOSC (drawing not to scale). This delay will not occur for RC and EC osc modes.
4: CLKOUT is not available in these oscillator modes, but shown here for timing reference.

NOTES:

20.0 INSTRUCTION SET SUMMARY

The PIC18C601/801 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration path from them.

With few exceptions, instructions are a single program memory word (16-bits). Each single word instruction is divided into an OPCODE, which specifies the instruction type, and one or more operands which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18C601/801 instruction set summary in Table 20-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 20-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (represented by 'f')
2. The destination of the result (represented by 'd')
3. The accessed memory (represented by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction.

The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (represented by 'f')
2. The bit in the file register (represented by 'b')
3. The accessed memory (represented by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (represented by 'k')
- The desired FSR register to load the literal value into (represented by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (represented by 'n')
- The mode of the Call or Return instructions (represented by 's')
- The mode of the Table Read and Table Write instructions (represented by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for four double word instructions. These four instructions were made double word instructions so that all the required information is available in these 32 bits. In the second word, the 4 MSbs are 1's. If this second word is executed as an instruction (by itself), it will execute as a `NOP`.

All single word instructions are executed in a single instruction cycle, unless a conditional test is true, or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a `NOP`.The double word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 µs. Two word branch instructions (if true) would take 3 µs.

Figure 20-1 shows the general formats that the instructions can have. All examples use the format `'nnh'` to represent a hexadecimal number, where `'h'` signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 20-2, lists the instructions recognized by the Microchip assembler (MPASM™).

Section 20.1 provides a description of each instruction.

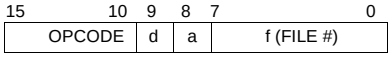
TABLE 20-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
ACCESS	ACCESS = 0: RAM access bit symbol
BANKED	BANKED = 1: RAM access bit symbol
bbb	Bit address within an 8-bit file register (0 to 7)
BSR	Bank Select Register. Used to select the current RAM bank.
d	Destination select bit; d = 0: store result in WREG, d = 1: store result in file register f.
dest	Destination either the WREG register or the specified register file location
f	8-bit Register file address (00h to FFh)
f_s	12-bit Register file address (000h to FFFh). This is the source address.
f_d	12-bit Register file address (000h to FFFh). This is the destination address.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value)
label	Label name
mm	The mode of the TBLPTR register for the Table Read and Table Write instructions Only used with Table Read and Table Write instructions:
*	No change to register (such as TBLPTR with Table reads and writes)
*+	Post-Increment register (such as TBLPTR with Table reads and writes)
*-	Post-Decrement register (such as TBLPTR with Table reads and writes)
+*	Pre-Increment register (such as TBLPTR with Table reads and writes)
n	The relative address (2's complement number) for relative branch instructions, or the direct address for Call/Branch and Return instructions
PRODH	Product of Multiply high byte (Register at address FF4h)
PRODL	Product of Multiply low byte (Register at address FF3h)
s	Fast Call / Return mode select bit. s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
u	Unused or Unchanged (Register at address FE8h)
W	W = 0: Destination select bit symbol
WREG	Working register (accumulator) (Register at address FE8h)
x	Don't care (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.
TBLPTR	21-bit Table Pointer (points to a Program Memory location) (Register at address FF6h)
TABLAT	8-bit Table Latch (Register at address FF5h)
TOS	Top-of-Stack
PC	Program Counter
PCL	Program Counter Low Byte (Register at address FF9h)
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch (Register at address FFAh)
PCLATU	Program Counter Upper Byte Latch (Register at address FFBh)
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
TO	Time-out bit
PD	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

FIGURE 20-1: GENERAL FORMAT FOR INSTRUCTIONS

Byte-oriented file register operations

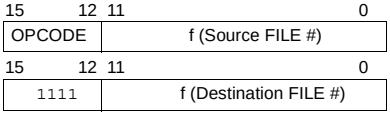
Example Instruction



ADDWF MYREG, W

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select Bank
f = 8-bit file register address

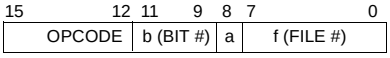
Byte to Byte move operations (2-word)



MOVFF MYREG1, MYREG2

f = 12-bit file register address

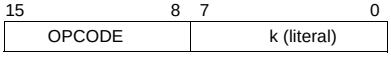
Bit-oriented file register operations



BSF MYREG, bit

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select Bank
f = 8-bit file register address

Literal operations

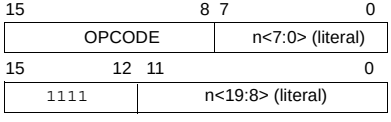


MOVLW 7Fh

k = 8-bit immediate value

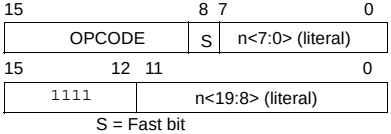
Control operations

CALL, GOTO and Branch operations



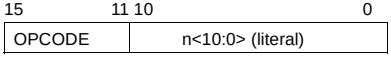
GOTO Label

n = 20-bit immediate value

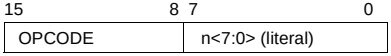


CALL MYFUNC

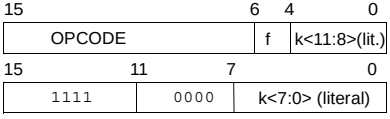
S = Fast bit



BRA MYFUNC



BC MYFUNC



LFSR FSR0, 100h

TABLE 20-2: PIC18C601/801 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f [,d [,a]]	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 6
ADDWFC	f [,d [,a]]	Add WREG and Carry bit to f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 6
ANDWF	f [,d [,a]]	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2, 6
CLRF	f [,a]	Clear f	1	0110	101a	ffff	ffff	Z	2, 6
COMF	f [,d [,a]]	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2, 6
CPFSEQ	f [,a]	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4, 6
CPFSGT	f [,a]	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4, 6
CPFSLT	f [,a]	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2, 6
DECF	f [,d [,a]]	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4, 6
DECFSZ	f [,d [,a]]	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4, 6
DCFSNZ	f [,d [,a]]	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2, 6
INCF	f [,d [,a]]	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4, 6
INCFSZ	f [,d [,a]]	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4, 6
INFSNZ	f [,d [,a]]	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2, 6
IORWF	f [,d [,a]]	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2, 6
MOVF	f [,d [,a]]	Move f	1	0101	00da	ffff	ffff	Z, N	1, 6
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination)2nd word	2	1100	ffff	ffff	ffff	None	
				1111	ffff	ffff	ffff		
MOVWF	f [,a]	Move WREG to f	1	0110	111a	ffff	ffff	None	6
MULWF	f [,a]	Multiply WREG with f	1	0000	001a	ffff	ffff	None	6
NEGF	f [,a]	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	1, 2, 6
RLCF	f [,d [,a]]	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	6
RLNCF	f [,d [,a]]	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	1, 2, 6
RRCF	f [,d [,a]]	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	6
RRNCF	f [,d [,a]]	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	6
SETF	f [,a]	Set f	1	0110	100a	ffff	ffff	None	6
SUBFWB	f [,d [,a]]	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 6
SUBWF	f [,d [,a]]	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	6
SUBWFB	f [,d [,a]]	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 6
SWAPF	f [,d [,a]]	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4, 6
TSTFSZ	f [,a]	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2, 6
XORWF	f [,d [,a]]	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	6
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b [,a]	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2, 6
BSF	f, b [,a]	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2, 6
BTFSC	f, b [,a]	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4, 6
BTFSS	f, b [,a]	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4, 6
BTG	f [,d [,a]]	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2, 6

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

6: Microchip's MPASM™ Assembler automatically defaults destination bit 'd' to '1', while access bit 'a' defaults to '1' or '0', according to address of register being used.

TABLE 20-2: PIC18C601/801 INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb LSb	Status Affected	Notes
CONTROL OPERATIONS						
BC	n	Branch if Carry	1 (2)	1110 0010 nnnn nnnn	None	
BN	n	Branch if Negative	1 (2)	1110 0110 nnnn nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110 0011 nnnn nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110 0111 nnnn nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110 0101 nnnn nnnn	None	
BNZ	n	Branch if Not Zero	2	1110 0001 nnnn nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110 0100 nnnn nnnn	None	
BRA	n	Branch Unconditionally	1 (2)	1101 0nnn nnnn nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110 0000 nnnn nnnn	None	
CALL	n, s	Call subroutine1st word	2	1110 110s kkkk kkkk	None	
		2nd word		1111 kkkk kkkk kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000 0000 0000 0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000 0000 0000 0111	C	
GOTO	n	Go to address1st word	2	1110 1111 kkkk kkkk	None	
		2nd word		1111 kkkk kkkk kkkk		
NOP	—	No Operation	1	0000 0000 0000 0000	None	
NOP	—	No Operation (Note 4)	1	1111 xxxx xxxx xxxx	None	
POP	—	Pop top of return stack (TOS)	1	0000 0000 0000 0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000 0000 0000 0101	None	
RCALL	n	Relative Call	2	1101 1nnn nnnn nnnn	None	
RESET		Software device RESET	1	0000 0000 1111 1111	All	
RETFIE	s	Return from interrupt enable	2	0000 0000 0001 000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000 1100 kkkk kkkk	None	
RETURN	s	Return from Subroutine	2	0000 0000 0001 001s	None	
SLEEP	—	Go into Standby mode	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

6: Microchip's MPASM™ Assembler automatically defaults destination bit 'd' to '1', while access bit 'a' defaults to '1' or '0', according to address of register being used.

TABLE 20-2: PIC18C601/801 INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Load FSR (f) with a 12-bit	2	1110	1110	00ff	kkkk	None	
		literal (k)		1111	0000	kkkk	kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2 (5)	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

6: Microchip's MPASM™ Assembler automatically defaults destination bit 'd' to '1', while access bit 'a' defaults to '1' or '0', according to address of register being used.

20.1 Instruction Set

ADDLW **ADD literal to WREG**

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) + k → WREG

Status Affected: N,OV, C, DC, Z

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of WREG are added to the 8-bit literal 'k' and the result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: `ADDLW 15h`

Before Instruction
- WREG = 10h
- N = ?
- OV = ?
- C = ?
- DC = ?
- Z = ?

After Instruction
- WREG = 25h
- N = 0
- OV = 0
- C = 0
- DC = 0
- Z = 0

ADDWF **ADD WREG to f**

Syntax: [*label*] ADDWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (WREG) + (f) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

0010	01da	ffff	ffff

Description: Add WREG to register 'f'. If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWF REG, W`

Before Instruction
- WREG = 17h
- REG = 0C2h
- N = ?
- OV = ?
- C = ?
- DC = ?
- Z = ?

After Instruction
- WREG = 0D9h
- REG = 0C2h
- N = 1
- OV = 0
- C = 0
- DC = 0
- Z = 0

ADDWFC **ADD WREG and Carry bit to f**

Syntax: [*label*] ADDWFC f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (WREG) + (f) + (C) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

0010	00da	ffff	ffff

Description: Add WREG, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWFC REG, W`

Before Instruction
C = 1
REG = 02h
WREG = 4Dh
N = ?
OV = ?
DC = ?
Z = ?

After Instruction
C = 0
REG = 02h
WREG = 50h
N = 0
OV = 0
DC = 0
Z = 0

ANDLW **AND literal with WREG**

Syntax: [*label*] ANDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .AND. k → WREG

Status Affected: N,Z

Encoding:

0000	1011	kkkk	kkkk

Description: The contents of WREG are AND'ed with the 8-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: `ANDLW 5Fh`

Before Instruction
WREG = 0A3h
N = ?
Z = ?

After Instruction
WREG = 03h
N = 0
Z = 0

ANDWF **AND WREG with f**

Syntax: [*label*] ANDWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (WREG) .AND. (f) → dest

Status Affected: N,Z

Encoding:

0001	01da	ffff	ffff

Description: The contents of WREG are AND'ed with register 'f'. If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ANDWF REG, W`

Before Instruction
WREG = 17h
REG = 0C2h
N = ?
Z = ?

After Instruction
WREG = 02h
REG = 0C2h
N = 0
Z = 0

BC **Branch if Carry**

Syntax: [*label*] BC n

Operands: $-128 \le n \le 127$

Operation: if carry bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0010	nnnn	nnnn

Description: If the Carry bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BC 5`

Before Instruction
PC = address `(HERE)`

After Instruction
If Carry = 1;
PC = address `(HERE+12)`
If Carry = 0;
PC = address `(HERE+2)`

BCF **Bit Clear f**

Syntax: [*label*] BCF f, b [,a]

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow$ f<b>

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BCF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0C7h

After Instruction
FLAG_REG = 47h

BN **Branch if Negative**

Syntax: [*label*] BN n

Operands: $-128 \le n \le 127$

Operation: if negative bit is '1'
(PC) + 2 + 2n $\rightarrow$ PC

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the Negative bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BN Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If Negative = 1;
PC = address `(Jump)`
If Negative = 0;
PC = address `(HERE+2)`

BNC **Branch if Not Carry**

Syntax: [*label*] BNC n

Operands: -128 ≤ n ≤ 127

Operation: if carry bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0011	nnnn	nnnn

Description: If the Carry bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNC Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If Carry = 0;
PC = address `(Jump)`
If Carry = 1;
PC = address `(HERE+2)`

BNN **Branch if Not Negative**

Syntax: [*label*] BNN n

Operands: -128 ≤ n ≤ 127

Operation: if negative bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0111	nnnn	nnnn

Description: If the Negative bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNN Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If Negative = 0;
PC = address `(Jump)`
If Negative = 1;
PC = address `(HERE+2)`

BNOV **Branch if Not Overflow**

Syntax: [*label*] BNOV n

Operands: -128 ≤ n ≤ 127

Operation: if overflow bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0101	nnnn	nnnn

Description: If the Overflow bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNOV Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If Overflow = 0;
PC = address `(Jump)`
If Overflow = 1;
PC = address `(HERE+2)`

BNZ **Branch if Not Zero**

Syntax: [*label*] BNZ n

Operands: -128 ≤ n ≤ 127

Operation: if zero bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0001	nnnn	nnnn

Description: If the Zero bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNZ Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If Zero = 0;
PC = address `(Jump)`
If Zero = 1;
PC = address `(HERE+2)`

BRA **Unconditional Branch**

Syntax: [*label*] BRA n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1101	0nnn	nnnn	nnnn

Description: Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE BRA Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
PC = address `(Jump)`

BSF **Bit Set f**

Syntax: [*label*] BSF f, b [,a]

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 1 → f<b>

Status Affected: None

Encoding:

1000	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is set. If 'a' is 0 Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BSF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0Ah

After Instruction
FLAG_REG = 8Ah

BTFSC **Bit Test File, Skip if Clear**

Syntax: [*label*] BTFSC f, b [,a]

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

1011	bbba	ffff	ffff

Description: If bit 'b' in register 'f' is 0, then the next instruction is skipped.
If bit 'b' is 0, then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    BTFSC   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

BTFSS **Bit Test File, Skip if Set**

Syntax: [*label*] BTFSS f, b [,a]

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

1010	bbba	ffff	ffff

Description: If bit 'b' in register 'f' is 1 then the next instruction is skipped.
If bit 'b' is 1, then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    BTFSS   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

BTG **Bit Toggle f**

Syntax: [*label*] BTG f, b [,a]

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff

Description: Bit 'b' in data memory location 'f' is inverted. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BTG PORTC, 4`

Before Instruction:
PORTC = 0111 0101 [75h]

After Instruction:
PORTC = 0110 0101 [65h]

BOV **Branch if Overflow**

Syntax: [*label*] BOV n

Operands: $-128 \le n \le 127$

Operation: if overflow bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn

Description: If the Overflow bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BOV Jump`

Before Instruction
PC = address (HERE)

After Instruction
If Overflow = 1;
PC = address (Jump)
If Overflow = 0;
PC = address (HERE+2)

BZ **Branch if Zero**

Syntax: [*label*] BZ n

Operands: $-128 \le n \le 127$

Operation: if Zero bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0000	nnnn	nnnn

Description: If the Zero bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BZ Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If Zero = 1;
PC = address `(Jump)`
If Zero = 0;
PC = address `(HERE+2)`

CALL **Subroutine Call**

Syntax: [*label*] CALL k [,s]

Operands: $0 \le k \le 1048575$
$s \in [0,1]$

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(WREG) → WS,
(STATUS) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	110s	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description: Subroutine call of entire 2M byte memory range. First, return address (PC+ 4) is pushed onto the return stack. If 's' = 1, the WREG, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then the 20-bit value 'k' is loaded into PC<20:1>. `CALL` is a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	Push PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE, FAST`

Before Instruction
PC = Address `(HERE)`

After Instruction
PC = Address `(THERE)`
TOS = Address `(HERE + 4)`
WS = WREGREG
BSRS = BSR
STATUSS = STATUS

CLRF **Clear f**

Syntax: [*label*] CLRF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: 000h → f
1 → Z

Status Affected: Z

Encoding:

0110	101a	ffff	ffff

Description: Clears the contents of the specified register. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: CLRF FLAG_REG

Before Instruction
FLAG_REG = 5Ah
Z = ?

After Instruction
FLAG_REG = 00h
Z = 0

CLRWDT **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

Operation: 000h → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0100

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: CLRWDT

Before Instruction
WDT counter = ?
WDT postscaler = ?
$\overline{TO}$ = ?
$\overline{PD}$ = ?

After Instruction
WDT counter = 00h
WDT postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF **Complement f**

Syntax: [*label*] COMF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ dest

Status Affected: N,Z

Encoding:

0001	11da	ffff	ffff

Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `COMF REG`

Before Instruction
REG = 13h
N = ?
Z = ?

After Instruction
REG = 13h
WREG = 0ECh
N = 1
Z = 0

CPFSEQ **Compare f with WREG, skip if f = WREG**

Syntax: [*label*] CPFSEQ f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (WREG),
skip if (f) = (WREG)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction.

If 'f' = WREG, then the fetched instruction is discarded and a `NOP` is executed instead making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = `HERE`
WREG = ?
REG = ?
After Instruction
If REG = WREG;
PC = Address `(EQUAL)`
If REG ≠ WREG;
PC = Address `(NEQUAL)`

CPFSGT — **Compare f with WREG, skip if f > WREG**

Syntax: [*label*] CPFSGT f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (WREG),
skip if (f) > (WREG)
(unsigned comparison)

Status Affected: None

Encoding:

0110	010a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of the WREG by performing an unsigned subtraction.

If the contents of 'f' are greater than the contents of **,** then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE      CPFSGT REG
NGREATER  :
GREATER   :
```
Before Instruction
PC = Address (HERE)
WREG = ?
After Instruction
If REG > WREG;
PC = Address (GREATER)
If REG ≤ WREG;
PC = Address (NGREATER)

CPFSLT — **Compare f with WREG, skip if f < WREG**

Syntax: [*label*] CPFSLT f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (WREG),
skip if (f) < (WREG)
(unsigned comparison)

Status Affected: None

Encoding:

0110	000a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction.

If the contents of 'f' are less than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSLT REG
NLESS   :
LESS    :
```
Before Instruction
PC = Address (HERE)
WREG = ?
After Instruction
If REG < WREG;
PC = Address (LESS)
If REG ≥ WREG;
PC = Address (NLESS)

DAW **Decimal Adjust WREG Register**

Syntax: [*label*] DAW

Operands: None

Operation: If [WREG<3:0> >9] or [DC = 1] then
(WREG<3:0>) + 6 → W<3:0>;
else
(WREG<3:0>) → W<3:0>;

If [WREG<7:4> >9] or [C = 1] then
(WREG<7:4>) + 6 → WREG<7:4>;
else
(WREG<7:4>) → WREG<7:4>;

Status Affected: C

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the eight-bit value in WREG resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register WREG	Process Data	Write WREG

Example1: DAW

Before Instruction
WREG = 0A5h
C = 0
DC = 0

After Instruction
WREG = 05h
C = 1
DC = 0

Example 2:

Before Instruction
WREG = 0CEh
C = 0
DC = 0

After Instruction
WREG = 34h
C = 1
DC = 0

DECF **Decrement f**

Syntax: [*label*] DECF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – 1 → dest

Status Affected: C,DC,N,OV,Z

Encoding:

0000	01da	ffff	ffff

Description: Decrement register 'f'. If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: DECF CNT

Before Instruction
CNT = 01h
Z = 0

After Instruction
CNT = 00h
Z = 1

DECFSZ **Decrement f, skip if 0**

Syntax: [*label*] DECFSZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0010	11da	ffff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default).
If the result is 0, the next instruction, which is already fetched, is discarded, and a NOP is executed instead, making it a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    DECFSZ  CNT
        GOTO    LOOP
CONTINUE
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE+2)

DCFSNZ **Decrement f, skip if not 0**

Syntax: [*label*] DCFSNZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

0100	11da	ffff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default).
If the result is not 0, the next instruction, which is already fetched, is discarded, and a NOP is executed instead, making it a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    DCFSNZ  TEMP
ZERO    :
NZERO   :
```

Before Instruction
TEMP = ?

After Instruction
TEMP = TEMP - 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP ≠ 0;
PC = Address (NZERO)

GOTO **Unconditional Branch**

Syntax: [*label*] GOTO k

Operands: $0 \le k \le 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	1111	k_7kkk	$kkkk_0$
2nd word(k<19:8>)	1111	$k_{19}kkk$	kkkk	$kkkk_8$

Description: GOTO allows an unconditional branch anywhere within entire 2M byte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: GOTO THERE

After Instruction
PC = Address (THERE)

INCF **Increment f**

Syntax: [*label*] INCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) + 1 → dest

Status Affected: C,DC,N,OV,Z

Encoding:

0010	10da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: INCF CNT

Before Instruction
CNT = 0FFh
Z = 0
C = ?
DC = ?

After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1

INCFSZ **Increment f, skip if 0**

Syntax: [*label*] INCFSZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) + 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0011	11da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default).
If the result is 0, the next instruction, which is already fetched, is discarded, and a NOP is executed instead, making it a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    INCFSZ  CNT
NZERO   :
ZERO    :
```
Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

INFSNZ **Increment f, skip if not 0**

Syntax: [*label*] INFSNZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) + 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

0100	10da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default).
If the result is not 0, the next instruction, which is already fetched, is discarded, and a NOP is executed instead, making it a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    INFSNZ  REG
ZERO
NZERO
```
Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

IORLW **Inclusive OR literal with WREG**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .OR. k → WREG

Status Affected: N,Z

Encoding:

0000	1001	kkkk	kkkk

Description: The contents of WREG are OR'ed with the eight bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: `IORLW 35h`

Before Instruction
WREG = 9Ah
N = ?
Z = ?

After Instruction
WREG = 0BFh
N = 1
Z = 0

IORWF **Inclusive OR WREG with f**

Syntax: [*label*] IORWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (WREG) .OR. (f) → dest

Status Affected: N,Z

Encoding:

0001	00da	ffff	ffff

Description: Inclusive OR WREG with register 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `IORWF RESULT, W`

Before Instruction
RESULT = 13h
WREG = 91h
N = ?
Z = ?

After Instruction
RESULT = 13h
WREG = 93h
N = 1
Z = 0

LFSR **Load FSR**

Syntax: [*label*] LFSR f,k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk

Description: The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL

Example: `LFSR FSR2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = 0ABh

MOVF **Move f**

Syntax: [*label*] MOVF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $f \rightarrow dest$

Status Affected: N,Z

Encoding:

0101	00da	ffff	ffff

Description: The contents of register 'f' is moved to a destination dependent upon the status of 'd'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte Bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write WREG

Example: `MOVF REG, W`

Before Instruction
REG = 22h
WREG = 0FFh
N = ?
Z = ?

After Instruction
REG = 22h
WREG = 22h
N = 0
Z = 0

MOVFF **Move f to f**

Syntax: [*label*] MOVFF f_s,f_d

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	1100	ffff	ffff	$ffff_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination 'f_d' can also be anywhere from 000h to FFFh.

Either source or destination can be WREG (a useful special situation).

MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: MOVFF REG1, REG2

Before Instruction
REG1 = 33h
REG2 = 11h

After Instruction
REG1 = 33h,
REG2 = 33h

MOVLB **Move literal to low nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \le k \le 255$

Operation: k → BSR

Status Affected: None

Encoding:

0000	0001	kkkk	kkkk

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR

Example: MOVLB 05h

Before Instruction
BSR register = 02h

After Instruction
BSR register = 05h

MOVLW **Move literal to WREG**

Syntax: [*label*] MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow WREG$

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: `MOVLW 5Ah`

After Instruction
WREG = 0x5A

MOVWF **Move WREG to f**

Syntax: [*label*] MOVWF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(WREG) \rightarrow f$

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from WREG to register 'f'. Location 'f' can be anywhere in the 256 byte Bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `MOVWF REG`

Before Instruction
WREG = 4Fh
REG = 0FFh

After Instruction
WREG = 4Fh
REG = 4Fh

MULLW **Multiply Literal with WREG**

Syntax: [*label*] MULLW k

Operands: $0 \le k \le 255$

Operation: (WREG) x k → PRODH:PRODL

Status Affected: None

Encoding:

0000	1101	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of WREG and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte.

WREG is unchanged.

None of the status flags are affected.

Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: `MULLW C4h`

Before Instruction
- WREG = 0E2h
- PRODH = ?
- PRODL = ?

After Instruction
- WREG = 0E2h
- PRODH = 0ADh
- PRODL = 08h

MULWF **Multiply WREG with f**

Syntax: [*label*] MULWF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (WREG) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

0000	001a	ffff	ffff

Description: An unsigned multiplication is carried out between the contents of WREG and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte.

Both WREG and 'f' are unchanged.

None of the status flags are affected.

Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example: `MULWF REG`

Before Instruction
- WREG = 0C4h
- REG = 0B5h
- PRODH = ?
- PRODL = ?

After Instruction
- WREG = 0C4h
- REG = 0B5h
- PRODH = 8Ah
- PRODL = 94h

NEGF **Negate f**

Syntax: [*label*] NEGF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(\overline{f}) + 1 \rightarrow f$

Status Affected: N,OV, C, DC, Z

Encoding:

0110	110a	ffff	ffff

Description: Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `NEGF REG`

Before Instruction
REG = 0011 1010 [3Ah]
N = ?
OV = ?
C = ?
DC = ?
Z = ?

After Instruction
REG = 1100 0110 [0C6h]
N = 1
OV = 0
C = 0
DC = 0
Z = 0

NOP **No Operation**

Syntax: [*label*] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

0000 1111	0000 xxxx	0000 xxxx	0000 xxxx

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

POP **Pop Top of Return Stack**

Syntax: [*label*] POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

0000	0000	0000	0110

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack.

This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Pop TOS value	No operation

Example:

```
POP
GOTO    NEW
```

Before Instruction
- TOS = 0031A2h
- Stack (1 level down) = 014332h

After Instruction
- TOS = 014332h
- PC = NEW

PUSH **Push Top of Return Stack**

Syntax: [*label*] PUSH

Operands: None

Operation: (PC+2) → TOS

Status Affected: None

Encoding:

0000	0000	0000	0101

Description: The PC+2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack.

This instruction allows implementing a software stack by modifying TOS, and then push it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Push PC+2 onto return stack	No operation	No operation

Example:

```
PUSH
```

Before Instruction
- TOS = 00345Ah
- PC = 000124h

After Instruction
- PC = 000126h
- TOS = 000126h
- Stack (1 level down) = 00345Ah

RCALL **Relative Call**

Syntax: [*label*] RCALL n

Operands: $-1024 \le n \le 1023$

Operation: $(PC) + 2 \rightarrow TOS$,
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1101	1nnn	nnnn	nnnn

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE RCALL` Jump

Before Instruction
PC = Address `(HERE)`

After Instruction
PC = Address `(Jump)`
TOS = Address `(HERE+2)`

RESET **Reset**

Syntax: [*label*] RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{MCLR}$ Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a $\overline{MCLR}$ Reset in software.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start reset	No operation	No operation

Example: `RESET`

After Instruction
Registers = Reset Value
Flags* = Reset Value

RETFIE **Return from Interrupt**

Syntax: [*label*] RETFIE [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if s = 1
(WS) → WREG,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: None

Encoding:

0000	0000	0001	000s

Description: Return from Interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting the either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, WREG, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	Pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example: `RETFIE 1`

After Interrupt
PC = TOS
WREG = WS
BSR = BSRS
STATUS = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETLW **Return Literal to WREG**

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	1100	kkkk	kkkk

Description: W is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Pop PC from stack, write to WREG
No operation	No operation	No operation	No operation

Example:

```
   CALL TABLE  ; WREG contains table
               ;  offset value
               ;  WREG now has
               ;  table value
   :
TABLE
   ADDWF PCL   ; WREG = offset
   RETLW k0    ; Begin table
   RETLW k1    ;
   :
   :
   RETLW kn    ; End of table
```

Before Instruction
WREG = 07h

After Instruction
WREG = value of kn

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	0000	0001	001s

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, WREG, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Pop PC from stack
No operation	No operation	No operation	No operation

Example: `RETURN`

After Call
PC = TOS

`RETURN FAST`

Before Instruction
WRG = 04h
STATUS = 00h
BSR = 00h

After Instruction
WREG = 04h
STATUS = 00h
BSR = 00h
PC = TOS

RLCF **Rotate Left f through Carry**

Syntax: [*label*] RLCF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

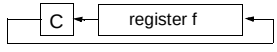
Operation: (f<n>) → dest<n+1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C,N,Z

Encoding:

0011	01da	ffff	ffff

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 WREG. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RLCF REG, W`

Before Instruction
REG = 1110 0110
C = 0
N = ?
Z = ?

After Instruction
REG = 1110 0110
WREG = 1100 1100
C = 1
N = 1
Z = 0

RLNCF **Rotate Left f (no carry)**

Syntax: [*label*] RLNCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

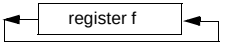
Operation: (f<n>) → dest<n+1>,
(f<7>) → dest<0>

Status Affected: N,Z

Encoding:

0100	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is 0 the result is placed in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLNCF REG

Before Instruction
REG = 1010 1011
N = ?
Z = ?

After Instruction
REG = 0101 0111
N = 0
Z = 0

RRCF **Rotate Right f through Carry**

Syntax: [*label*] RRCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

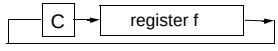
Operation: (f<n>) → dest<n-1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C,N,Z

Encoding:

0011	00da	ffff	ffff

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 WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RRCF REG, W

Before Instruction
REG = 1110 0110
C = 0
N = ?
Z = ?

After Instruction
REG = 1110 0110
WREG = 0111 0011
C = 0
N = 0
Z = 0

RRNCF **Rotate Right f (no carry)**

Syntax: [*label*] RRNCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

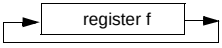
Operation: (f<n>) → dest<n-1>,
(f<0>) → dest<7>

Status Affected: N,Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: `RRNCF REG`

Before Instruction
REG = 1101 0111
N = ?
Z = ?

After Instruction
REG = 1110 1011
N = 1
Z = 0

Example 2: `RRNCF REG, 0, 0`

Before Instruction
WREG = ?
REG = 1101 0111
N = ?
Z = ?

After Instruction
WREG = 1110 1011
REG = 1101 0111
N = 1
Z = 0

SETF **Set f**

Syntax: [*label*] SETF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `SETF REG`

Before Instruction
REG = 5Ah

After Instruction
REG = 0FFh

SLEEP **Enter SLEEP mode**

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT postscaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0011

Description: The power-down status bit ($\overline{PD}$) is cleared. The time-out status bit ($\overline{TO}$) is set. Watchdog Timer and its postscaler are cleared.

The processor is put into SLEEP mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to sleep

Example: SLEEP

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?

After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared.

SUBFWB **Subtract f from WREG with borrow**

Syntax: [*label*] SUBFWB f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (WREG) – (f) – ($\overline{C}$) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

0101	01da	ffff	ffff

Description: Subtract register 'f' and carry flag (borrow) from WREG (2's complement method). If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored in register 'f' (default) . If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

SUBFWB (Cont.)

Example 1: `SUBFWB REG`

Before Instruction
REG = 3
WREG = 2
C = 1

After Instruction
REG = 0FFh
WREG = 2
C = 0
Z = 0
N = 1 ; result is negative

Example 2: `SUBFWB REG`

Before Instruction
REG = 2
WREG = 5
C = 1

After Instruction
REG = 2
WREG = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3: `SUBFWB REG`

Before Instruction
REG = 1
WREG = 2
C = 0

After Instruction
REG = 0
WREG = 2
C = 1
Z = 1 ; result is zero
N = 0

SUBLW Subtract WREG from literal

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (WREG) \rightarrow WREG$

Status Affected: N,OV, C, DC, Z

Encoding:

0000	1000	kkkk	kkkk

Description: WREG is subtracted from the eight bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example 1: `SUBLW 02h`

Before Instruction
WREG = 1
C = ?

After Instruction
WREG = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBLW 02h`

Before Instruction
WREG = 2
C = ?

After Instruction
WREG = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBLW 02h`

Before Instruction
WREG = 3
C = ?

After Instruction
WREG = 0FFh ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF **Subtract WREG from f**

Syntax: [*label*] SUBWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – (WREG) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

0101	11da	ffff	ffff

Description: Subtract WREG from register 'f' (2's complement method). If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

SUBWF (Cont.)

Example 1:
```
SUBWF   REG
```
Before Instruction
```
REG  =  3
WREG =  2
C    =  ?
```
After Instruction
```
REG  =  1
WREG =  2
C    =  1   ; result is positive
Z    =  0
N    =  0
```
Example 2:
```
SUBWF   REG, W
```
Before Instruction
```
REG  =  2
WREG =  2
C    =  ?
```
After Instruction
```
REG  =  2
WREG =  0
C    =  1   ; result is zero
Z    =  1
N    =  0
```
Example 3:
```
SUBWF   REG
```
Before Instruction
```
REG  =  1
WREG =  2
C    =  ?
```
After Instruction
```
REG  =  0FFh  ;(2's complement)
WREG =  2
C    =  0   ; result is negative
Z    =  0
N    =  1
```


SUBWFB **Subtract WREG from f with Borrow**

Syntax: [*label*] SUBWFB f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – (WREG) – ($\overline{C}$) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

0101	10da	ffff	ffff

Description: Subtract WREG and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

SUBWFB (Cont.)

Example 1: `SUBWFB REG`

Before Instruction
REG = 19h (0001 1001)
WREG = 0Dh (0000 1101)
C = 1

After Instruction
REG = 0Ch (0000 1011)
WREG = 0Dh (0000 1101)
C = 1
Z = 0
N = 0 ; result is positive

Example 2: `SUBWFB REG, W`

Before Instruction
REG = 1Bh (0001 1011)
WREG = 1Ah (0001 1010)
C = 0

After Instruction
REG = 1Bh (0001 1011)
WREG = 00h
C = 1
Z = 1 ; result is zero
N = 0

Example 3: `SUBWFB REG`

Before Instruction
REG = 03h (0000 0011)
WREG = 0Eh (0000 1101)
C = 1

After Instruction
REG = 0F5h (1111 0100) [2's comp]
WREG = 0Eh (0000 1101)
C = 0
Z = 0
N = 1 ; result is negative

SWAPF	Swap nibbles in f
Syntax:	[*label*] SWAPF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>
Status Affected:	None

Encoding:

0011	10da	ffff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `SWAPF REG`

Before Instruction
REG = 53h

After Instruction
REG = 35h

TBLRD **Table Read**

Syntax: [*label*] TBLRD (*; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT;
TBLPTR - No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) +1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) -1 → TBLPTR;
if TBLRD +*,
(TBLPTR) +1 → TBLPTR;
(Prog Mem (TBLPTR)) → TABLAT;

Status Affected: None

Encoding:

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2 Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD (Cont.)

Example 1: `TBLRD *+ ;`

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 34h

After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example 2: `TBLRD +* ;`

Before Instruction
TABLAT = 0AAh
TBLPTR = 01A357h
MEMORY(01A357h) = 12h
MEMORY(01A358h) = 34h

After Instruction
TABLAT = 34h
TBLPTR = 01A358h

TBLWT **Table Write**

Syntax: [*label*] TBLWT (*; *+; *-; +*)

Operands: None

Operation: if TBLWT*,
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;
TBLPTR - No Change;
if TBLWT*+,
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;
(TBLPTR) +1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;
(TBLPTR) -1 → TBLPTR;
if TBLWT+*,
(TBLPTR) +1 → TBLPTR;
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;

Status Affected: None

Encoding:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction is used to program the contents of Program Memory (P.M.).

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2 MByte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2 (many if long write is to on-chip EPROM program memory)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register or Memory)

TBLWT (Cont.)

Example 1: `TBLWT *+;`

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 0FFh

After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
MEMORY(00A356h) = 55h

Example 2: `TBLWT +*;`

Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
MEMORY(01389Ah) = 0FFh
MEMORY(01389Bh) = 0FFh

After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
MEMORY(01389Ah) = 0FFh
MEMORY(01389Bh) = 34h

TSTFSZ **Test f, skip if 0**

Syntax: [*label*] TSTFSZ f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: skip if f = 0

Status Affected: None

Encoding:

0110	011a	ffff	ffff

Description: If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    TSTFSZ  CNT
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)

After Instruction
If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)

XORLW **Exclusive OR literal with WREG**

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .XOR. k → WREG

Status Affected: N,Z

Encoding:

0000	1010	kkkk	kkkk

Description: The contents of WREG are XOR'ed with the 8-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: XORLW 0AFh

Before Instruction
WREG = 0B5h
N = ?
Z = ?

After Instruction
WREG = 1Ah
N = 0
Z = 0

XORWF	Exclusive OR WREG with f
Syntax:	[*label*] XORWF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(WREG) .XOR. (f) → dest
Status Affected:	N,Z

Encoding:

0001	10da	ffff	ffff

Description: Exclusive OR the contents of WREG with register 'f'. If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in the register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `XORWF REG`

Before Instruction
- REG = 0AFh
- WREG = 0B5h
- N = ?
- Z = ?

After Instruction
- REG = 1Ah
- WREG = 0B5h
- N = 0
- Z = 0

21.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

21.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or ‘C’)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

21.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PIC MCUs.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. 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, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

21.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI ‘C’ compilers for Microchip’s PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

21.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can also link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian is a librarian for pre-compiled code to be used with the MPLINK object linker. When a routine from a library is called from another 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 MPLIB object librarian manages the creation and modification of library files.

The MPLINK object linker features include:

- Integration with MPASM assembler and MPLAB C17 and MPLAB C18 C compilers.
- Allows all memory areas to be defined as sections to provide link-time flexibility.

The MPLIB object librarian features include:

- Easier linking because single libraries can be included instead of many smaller files.
- Helps keep code maintainable by grouping related modules together.
- Allows libraries to be created and modules to be added, listed, replaced, deleted or extracted.

21.5 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment by simulating the PIC series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user-defined key press, to any of the pins. The execution can be performed in single step, execute until break, or trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and the MPLAB C18 C compilers and the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent multi-project software development tool.

21.6 MPLAB ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB ICE universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers (MCUs). Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE in-circuit emulator system has been designed as a real-time emulation system, with advanced features that are generally found on more expensive development tools. The PC platform and Microsoft® Windows environment were chosen to best make these features available to you, the end user.

21.7 ICEPIC In-Circuit Emulator

The ICEPIC low cost, in-circuit emulator is a solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X and PIC16CXXX families of 8-bit One-Time-Programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules, or daughter boards. The emulator is capable of emulating without target application circuitry being present.

21.8 MPLAB ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD, is a powerful, low cost, run-time development tool. This tool is based on the FLASH PIC16F87X and can be used to develop for this and other PIC microcontrollers from the PIC16CXXX family. The MPLAB ICD utilizes the in-circuit debugging capability built into the PIC16F87X. This feature, along with Microchip's In-Circuit Serial Programming™ protocol, offers cost-effective in-circuit FLASH debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time.

21.9 PRO MATE II Universal Device Programmer

The PRO MATE II universal device programmer is a full-featured programmer, capable of operating in stand-alone mode, as well as PC-hosted mode. The PRO MATE II device programmer is CE compliant.

The PRO MATE II device programmer has programmable V_{DD} and V_{PP} supplies, which allow it to verify programmed memory at V_{DD} min and V_{DD} max for maximum reliability. It has an LCD display for instructions and error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode, the PRO MATE II device programmer can read, verify, or program PIC devices. It can also set code protection in this mode.

21.10 PICSTART Plus Entry Level Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

The PICSTART Plus development programmer supports all PIC devices with up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

21.11 PICDEM 1 Low Cost PIC MCU Demonstration Board

The PICDEM 1 demonstration board is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The user can program the sample microcontrollers provided with the PICDEM 1 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The user can also connect the PICDEM 1 demonstration board to the MPLAB ICE in-circuit emulator and download the firmware to the emulator for testing. A prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs connected to PORTB.

21.12 PICDEM 2 Low Cost PIC16CXX Demonstration Board

The PICDEM 2 demonstration board is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 2 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a serial EEPROM to demonstrate usage of the I^2C™ bus and separate headers for connection to an LCD module and a keypad.

21.13 PICDEM 3 Low Cost PIC16CXXX Demonstration Board

The PICDEM 3 demonstration board is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with an LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 3 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer with an adapter socket, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 3 demonstration board to test firmware. A prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM 3 demonstration board is a LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM 3 demonstration board provides an additional RS-232 interface and Windows software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

21.14 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included and the user may erase it and program it with the other sample programs using the PRO MATE II device programmer, or the PICSTART Plus development programmer, and easily debug and test the sample code. In addition, the PICDEM 17 demonstration board supports downloading of programs to and executing out of external FLASH memory on board. The PICDEM 17 demonstration board is also usable with the MPLAB ICE in-circuit emulator, or the PICMASTER emulator and all of the sample programs can be run and modified using either emulator. Additionally, a generous prototype area is available for user hardware.

21.15 KEELOQ Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchip's HCS Secure Data Products. The HCS evaluation kit includes a LCD display to show changing codes, a decoder to decode transmissions and a programming interface to program test transmitters.

TABLE 21-1: DEVELOPMENT TOOLS FROM MICROCHIP

		PIC12CXXX	PIC14000	PIC16C5X	PIC16C6X	PIC16CXXX	PIC16F62X	PIC16C7X	PIC16C7XX	PIC16C8X	PIC16F8XX	PIC16C9XX	PIC17C4X	PIC17C7XX	PIC18CXX2	24CXX/ 25CXX/ 93CXX	HCSXXX	MCRFXXX	MCP2510
Software Tools	MPLAB® Integrated Development Environment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	MPLAB® C17 C Compiler								✓				✓	✓					
	MPLAB® C18 C Compiler														✓				
	MPASM™ Assembler/ MPLINK™ Object Linker	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Emulators	MPLAB® ICE In-Circuit Emulator	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓				
	ICEPIC™ In-Circuit Emulator	✓		✓	✓	✓		✓	✓	✓		✓							
Debugger	MPLAB® ICD In-Circuit Debugger				✓*			✓*			✓								
Programmers	PICSTART® Plus Entry Level Development Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓				
	PRO MATE® II Universal Device Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Demo Boards and Eval Kits	PICDEM™ 1 Demonstration Board			✓		✓		✓†		✓									
	PICDEM™ 2 Demonstration Board				✓†			✓†							✓				
	PICDEM™ 3 Demonstration Board											✓							
	PICDEM™ 14A Demonstration Board		✓																
	PICDEM™ 17 Demonstration Board													✓					
	KEELOQ® Evaluation Kit																✓		
	KEELOQ® Transponder Kit																✓		
	microID™ Programmer's Kit																	✓	
	125 kHz microID™ Developer's Kit																	✓	
	125 kHz Anticollision microID™ Developer's Kit																	✓	
	13.56 MHz Anticollision microID™ Developer's Kit																	✓	
	MCP2510 CAN Developer's Kit																		✓

* Contact the Microchip Technology Inc. web site at www.microchip.com for information on how to use the MPLAB® ICD In-Circuit Debugger (DV164001) with PIC16C62, 63, 64, 65, 72, 73, 74, 76, 77.

** Contact Microchip Technology Inc. for availability date.

† Development tool is available on select devices.

NOTES:

22.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -55°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD}, $\overline{MCLR}$, and RA4) ..... -0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS} ..... -0.3V to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} **(Note 2)** ..... 0V to +13.25V
Voltage on RA4 with respect to V_{SS} ..... 0V to +8.5V
Total power dissipation **(Note 1)** 1.0W
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ±20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > 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
Maximum current sunk by all ports (combined) ..... 200 mA
Maximum current sourced by all ports (combined) ..... 200 mA

Note 1: Power dissipation is calculated as follows:
$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

2: Voltage spikes below V_{SS} at the $\overline{MCLR}/V_{PP}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}/V_{PP}$ pin, rather than pulling this pin directly to V_{SS}.

† 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 to maximum rating conditions for extended periods may affect device reliability.

FIGURE 22-1: PIC18C601/801 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

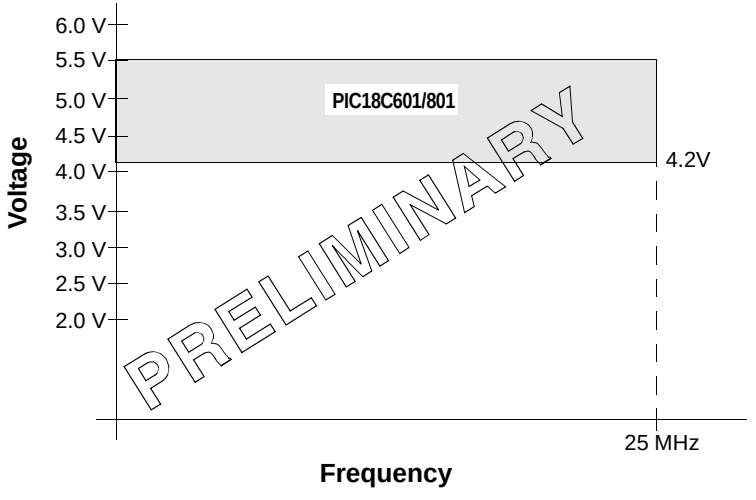
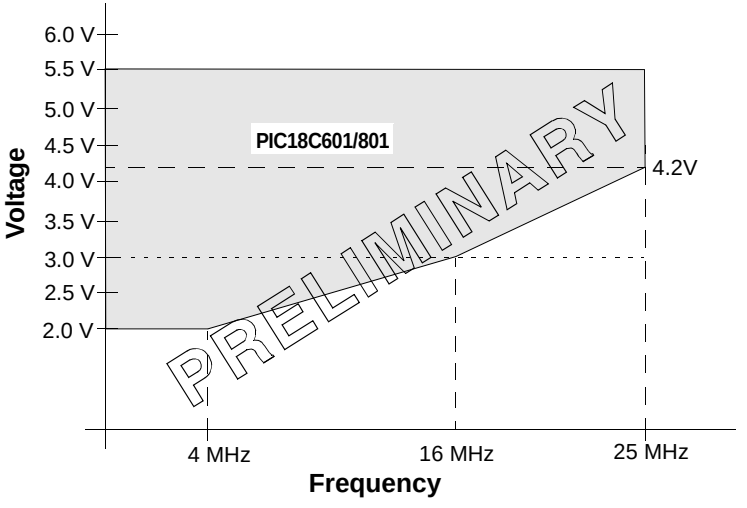


FIGURE 22-2: PIC18C601/801 VOLTAGE-FREQUENCY GRAPH (EXTENDED)



FMAX = (12.0 MHz/V) (VDDAPPMIN - 2.0 V) + 4 MHz where VDDAPPMIN ≤ 3
FMAX = (7.5 MHz/V) (VDDAPPMIN - 3.0 V) + 16 MHz where VDDAPPMIN > 3
Note: VDDAPP is the minimum voltage of the PIC® device in the application.

22.1 DC Characteristics

PIC18LC601/801 (Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC18C601/801 (Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
D001	VDD	**Supply Voltage**					
		PIC18LC601/801	2.0	—	5.5	V	
D001		PIC18C601/801	4.2	—	5.5	V	
D002	VDR	**RAM Data Retention Voltage(1)**	1.5	—	—	V	
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	—	0.7	V	See section on Power-on Reset for details
D004	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See section on Power-on Reset for details

Legend: Rows with industrial-extended data are shaded for improved readability.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode, or during a device RESET, without losing RAM data.

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.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD
MCLR = VDD; WDT enabled/disabled as specified.

3: 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 hi-impedance state and tied to VDD or VSS, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, ...).

4: For RC osc option, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kOhm.

22.1 DC Characteristics (Continued)

PIC18LC601/801 (Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC18C601/801 (Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
D010	IDD	**Supply Current(2,4)**					
		PIC18LC601/801	—	TBD	TBD	mA	RC osc option FOSC = 4 MHz, VDD = 2.5V
D010		PIC18C601/801	—	TBD	TBD	mA	RC osc options FOSC = 4 MHz, VDD = 4.2V
D010A		PIC18LC601/801	—	TBD	TBD	µA	LP osc option FOSC = 32 kHz, VDD = 2.5V
D010A		PIC18C601/801	—	TBD	TBD	µA	LP osc option FOSC = 32 kHz, VDD = 4.2V
D010C		PIC18LC601/801	—	TBD	45	mA	EC osc option, FOSC = 25 MHz, VDD = 5.5V
D010C		PIC18C601/801	—	—	45	mA	EC osc option, FOSC = 25 MHz, VDD = 5.5V
D013		PIC18LC601/801	— — —	— — —	TBD 50 50	mA mA mA	HS osc options FOSC = 6 MHz, VDD = 2.5V FOSC = 25 MHz, VDD = 5.5V HS + PLL osc option FOSC = 10 MHz, VDD = 5.5V
D013		PIC18C601/801	— —	— —	50 50	mA mA	HS osc option FOSC = 25 MHz, VDD = 5.5V HS + PLL osc option FOSC = 10 MHz, VDD = 5.5V
D014		PIC18LC601/801	— —	— —	48 TBD	µA µA	Timer1 osc option FOSC = 32 kHz, VDD = 2.5V FOSC = 32 kHz, VDD = 2.5V, 25°C
D014		PIC18C601/801	— —	— —	TBD TBD	µA µA	OSCB osc option FOSC = 32 kHz, VDD = 4.2V FOSC = 32 kHz, VDD = 4.2V, 25°C

Legend: Rows with industrial-extended data are shaded for improved readability.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode, or during a device RESET, without losing RAM data.

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.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD
MCLR = VDD; WDT enabled/disabled as specified.

3: 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 hi-impedance state and tied to VDD or VSS, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, ...).

4: For RC osc option, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kOhm.

22.1 DC Characteristics (Continued)

PIC18LC601/801 (Industrial)			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC18C601/801 (Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
D020	IPD	**Power-down Current(3)**					
		PIC18LC601/801	—	TBD	5	µA	VDD = 2.5V, -40°C to +85°C
			—	—	36	µA	VDD = 5.5V, -40°C to +85°C
			—	—	TBD	µA	VDD = 2.5V, 25°C
D020		PIC18C601/801	—	TBD	TBD	µA	VDD = 4.2V, -40°C to +85°C
			—	—	36	µA	VDD = 5.5V, -40°C to +85°C
D020A			—	—	TBD	µA	VDD = 4.2V, 25°C
D021B			—	TBD	TBD	µA	VDD = 4.2V, -40°C to +125°C
			—	—	42		VDD = 5.5V, -40°C to +125°C
D022	ΔIWDT	**Module Differential Current**					
		PIC18LC801/601	—	TBD	TBD	µA	VDD = 2.5V
		Watchdog Timer	—	6.5	12	µA	VDD = 3.0V
			—	—	TBD	µA	VDD = 5.5V
			—	—	TBD	µA	VDD = 2.5V, 25°C
D022		PIC18C601/801	—	—	TBD	µA	VDD = 5.5V, -40°C to +85°C
		Watchdog Timer	—	—	TBD	µA	VDD = 5.5V, -40°C to +125°C
			—	—	TBD	µA	VDD = 4.2V, 25°C
D022B	ΔILVD	PIC18LC801/601	—	—	50	µA	VDD = 2.5V
		Low Voltage Detect	—	—	TBD	µA	VDD = 2.5V, 25°C
D022B		PIC18C601/801	—	—	TBD	µA	VDD = 4.2V, -40°C to +85°C
		Low Voltage Detect	—	—	TBD	µA	VDD = 4.2V, -40°C to +125°C
			—	—	TBD	µA	VDD = 4.2V, 25°C
D025	ΔIOSCB	PIC18LC801/601	—	—	3	µA	VDD = 2.5V
		Timer1 Oscillator	—	—	TBD	µA	VDD = 2.5V, 25°C
D025		PIC18C601/801	—	—	TBD	µA	VDD = 4.2V, -40°C to +85°C
		Timer1 Oscillator	—	—	TBD	µA	VDD = 4.2V, -40°C to +125°C
			—	—	TBD	µA	VDD = 4.2V, 25°C

Legend: Rows with industrial-extended data are shaded for improved readability.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode, or during a device RESET, without losing RAM data.

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.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD
MCLR = VDD; WDT enabled/disabled as specified.

3: 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 hi-impedance state and tied to VDD or VSS, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, ...).

4: For RC osc option, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kOhm.

22.2 DC Characteristics: PIC18C801 (Industrial, Extended) PIC18LC601/801 (Industrial)

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.	**Symbol**	**Characteristic/ Device**	**Min**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**				
		I/O ports:				
D030		with TTL buffer	VSS	0.15VDD	V	VDD < 4.5V
D030A			—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D031		with Schmitt Trigger buffer	VSS	0.2 VDD	V	
		RC3 and RC4	VSS	0.3VDD	V	
D032		$\overline{MCLR}$	VSS	0.2 VDD	V	
D032A		OSC1 (in XT, HS and LP modes) and T1OSI	VSS	0.3VDD	V	
D033		OSC1(in RC mode)**(1)**	VSS	0.2 VDD	V	
	VIH	**Input High Voltage**				
		I/O ports:				
D040		with TTL buffer	0.25VDD + 0.8V	VDD	V	VDD < 4.5V
D040A			2.0	VDD	V	4.5V ≤ VDD ≤ 5.5V
D041		with Schmitt Trigger buffer	0.8VDD	VDD	V	
		RC3 and RC4	0.7VDD	VDD	V	
D042		$\overline{MCLR}$	0.8VDD	VDD	V	
D042A		OSC1 (in HS and LP modes) and T1OSI	0.7VDD	VDD	V	
D043		OSC1 (RC mode)**(1)**	0.9VDD	VDD	V	
	VHYS	**Hysteresis of Schmitt Trigger Inputs**				
D050			TBD	TBD	V	
	IIL	**Input Leakage Current(2,3)**				
D060		I/O ports	—	±1	µA	VSS ≤ VPIN ≤ VDD, Pin at hi-impedance
D061		$\overline{MCLR}$	—	±5	µA	VSS ≤ VPIN ≤ VDD
D063		OSC1	—	±5	µA	VSS ≤ VPIN ≤ VDD
	IPU	**Weak Pull-up Current**				
D070	IPURB	PORTB weak pull-up current	50	400	µA	VDD = 5V, VPIN = VSS

Note 1: In RC oscillator option, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC device be driven with an external clock while in RC mode.

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

22.2 DC Characteristics: PIC18C801 (Industrial, Extended) PIC18LC601/801 (Industrial) (Continued)

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.	**Symbol**	**Characteristic/ Device**	**Min**	**Max**	**Units**	**Conditions**
	VOL	**Output Low Voltage**				
D080		I/O ports	—	0.6	V	IOL = 8.5 mA, VDD = 4.5V, -40°C to +85°C
D080A			—	0.6	V	IOL = 7.0 mA, VDD = 4.5V, -40°C to +125°C
D083		OSC2/CLKO (RC mode)	—	0.6	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
D083A			—	0.6	V	IOL = 1.2 mA, VDD = 4.5V, -40°C to +125°C
D084		System Bus mode	—	TBD	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
D084A			—	TBD	V	IOL = 1.2 mA, VDD = 4.5V, -40°C to +125°C
D085		Control Signals	—	TBD	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
D085A			—	TBD	V	IOL = 1.2 mA, VDD = 4.5V, -40°C to +125°C
	VOH	**Output High Voltage(3)**				
D090		I/O ports	VDD - 0.7	—	V	IOH = -3.0 mA, VDD = 4.5V, -40°C to +85°C
D090A			VDD - 0.7	—	V	IOH = -2.5 mA, VDD = 4.5V, -40°C to +125°C
D092		OSC2/CLKO (RC mode)	VDD - 0.7	—	V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
D092A			VDD - 0.7	—	V	IOH = -1.0 mA, VDD = 4.5V, -40°C to +125°C
D093		System Bus mode	TBD	—	V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
D093A			TBD	—	V	IOH = -1.0 mA, VDD = 4.5V, -40°C to +125°C
D094		Control Signals	TBD	—	V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
D094A			TBD	—	V	IOH = -1.0 mA, VDD = 4.5V, -40°C to +125°C
	VOD	**Open-drain High Voltage**				
D150			—	7.5	V	RA4 pin
		Capacitive Loading Specs on Output Pins				
D101	CIO	All I/O pins and OSC2 (in RC mode)	—	50	pF	To meet the AC Timing Specifications
D102	CB	SCL, SDA	—	400	pF	In I^2C mode

Note 1: In RC oscillator option, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC device be driven with an external clock while in RC mode.

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

FIGURE 22-3: LOW-VOLTAGE DETECT CHARACTERISTICS

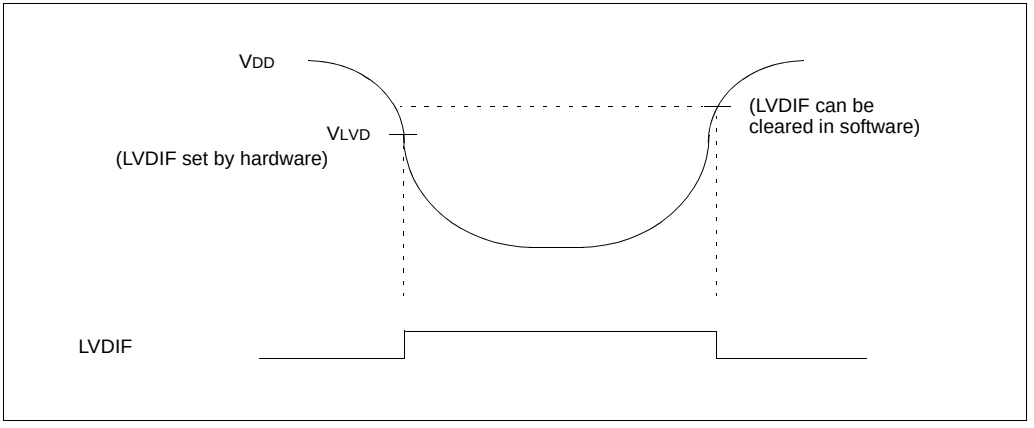


TABLE 22-1: LOW VOLTAGE DETECT CHARACTERISTICS

			VCC = 2.0V to 5.5V Commercial (C): TAMB = 0°C to +70°C Industrial (I): TAMB = -40°C to +85°C					
Param No.	**Characteristic**		**Symbol**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D420	LVD Voltage on VDD Transition High to Low	LVV = 0001	VLVD	2.0	2.06	2.12	V	
		LVV = 0010		2.2	2.27	2.34	V	
		LVV = 0011		2.4	2.47	2.54	V	
		LVV = 0100		2.5	2.58	2.66	V	
		LVV = 0101		2.7	2.78	2.86	V	
		LVV = 0110		2.8	2.89	2.98	V	
		LVV = 0111		3.0	3.1	3.2	V	
		LVV = 1000		3.3	3.41	3.52	V	
		LVV = 1001		3.5	3.61	3.72	V	
		LVV = 1010		3.6	3.72	3.84	V	
		LVV = 1011		3.8	3.92	4.04	V	
		LVV = 1100		4.0	4.13	4.26	V	
		LVV = 1101		4.2	4.33	4.46	V	
		LVV = 1110		4.5	4.64	4.78	V	
D421	LVD Voltage Drift Temperature Coefficient		TCVOUT	—	15	50	ppm/°C	
D422	Bandgap Voltage Drift with respect to VDD Regulation		ΔVBG/ ΔVDD	—	—	50	µV/V	
D423	Bandgap Reference Voltage Value		VBG	—	1.22		V	

Note: Production tested at TAMB = 25°C. Specifications over temperature limits guaranteed by characterization.

22.3 AC (Timing) Characteristics

22.3.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
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 (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^2C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	STOP condition
STA	START condition		

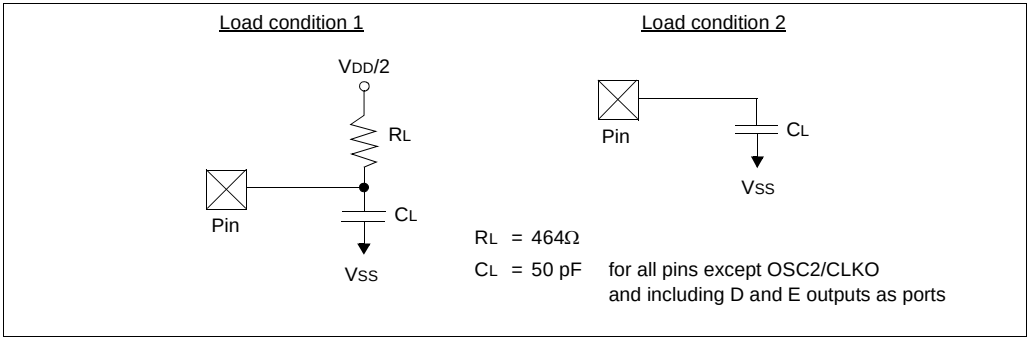
22.3.2 TIMING CONDITIONS

The temperature and voltages specified in Table 22-2 apply to all timing specifications, unless otherwise noted. Figure 22-4 specifies the load conditions for the timing specifications.

TABLE 22-2: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Section 22.1. LC parts operate for industrial temperatures only.

FIGURE 22-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



22.3.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 22-5: EXTERNAL CLOCK TIMING

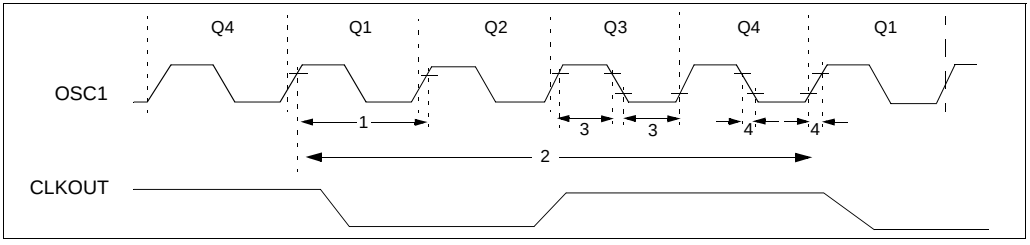


TABLE 22-3: EXTERNAL CLOCK TIMING REQUIREMENTS

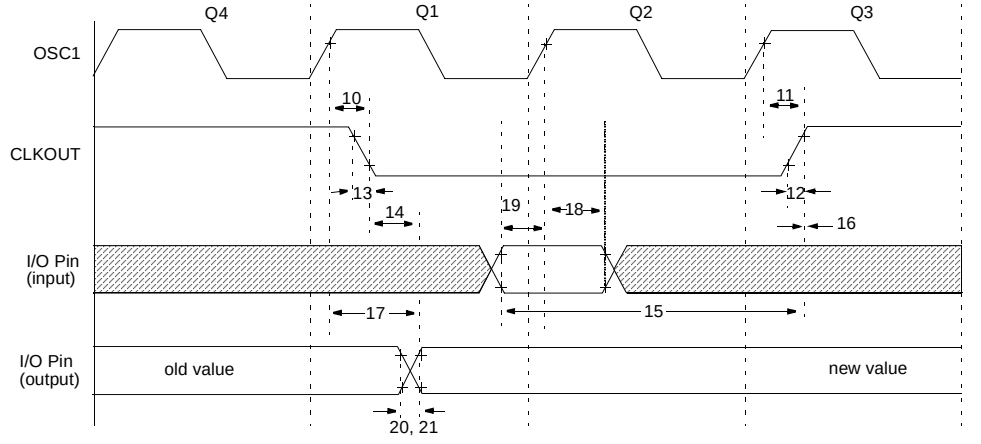
Param No.	Symbol	Characteristic	Min	Typ	Max	Units
	FOSC	**External CLKI Frequency (Note 1)**	DC	—	4	MHz
			DC	—	25	MHz
			4	—	6.25	MHz
			DC	—	25	MHz
			DC	—	200	kHz
		Oscillator Frequency (Note 1)	DC	—	4	MHz
			4	—	25	MHz
			4	—	6.25	MHz
			5	—	200	kHz
1	TOSC	**External CLKI Period (Note 1)**	250	—	—	ns
			40	—	—	ns
			40	—	—	ns
			160	—	—	ns
			5	—	—	µs
		Oscillator Period (Note 1)	250	—	—	ns
			40	—	100	ns
			160	—	100	ns
			5	—	—	µs
2	TCY	**Instruction Cycle Time (Note 1)**	160	TCY	DC	ns
3	TOSL, TOSH	**External Clock in (OSC1) High or Low Time**	2.5	—	—	µs
			10	—	—	ns
4	TOSR, TOSF	**External Clock in (OSC1) Rise or Fall Time**	—	—	50	ns
			—	—	5	ns

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 the OSC1/CLKI pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.

TABLE 22-4: PLL CLOCK TIMING SPECIFICATION (VDD = 4.2V - 5.5V)

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
7	TPLL	PLL Start-up Time (Lock Time)	—	2	ms	
	ΔCLK	CLKOUT Stability (Jitter) using PLL	-2	+2	%	

FIGURE 22-6: CLKOUT AND I/O TIMING



Note: Refer to Figure 22-4 for load conditions.

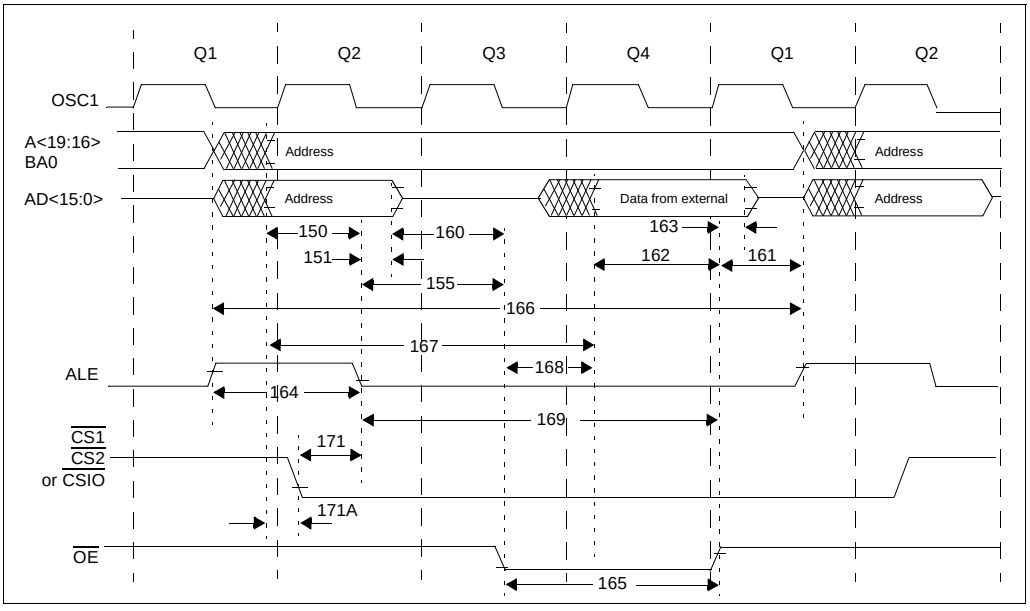
TABLE 22-5: CLKOUT AND I/O TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
10	TosH2ckL	OSC1↑ to CLKOUT↓		—	75	200	ns	(1)
11	TosH2ckH	OSC1↑ to CLKOUT↑		—	75	200	ns	(1)
12	TckR	CLKOUT rise time		—	35	100	ns	(1)
13	TckF	CLKOUT fall time		—	35	100	ns	(1)
14	TckL2ioV	CLKOUT ↓ to Port out valid		—	—	0.5TCY + 20	ns	(1)
15	TioV2ckH	Port in valid before CLKOUT ↑		0.25TCY + 25	—	—	ns	(1)
16	TckH2ioI	Port in hold after CLKOUT ↑		0	—	—	ns	(1)
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid			50	150	ns	
18	TosH2ioI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	PIC18C601/801	100	—	—	ns	
18A			PIC18**LC**601/801	200	—	—	ns	
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)		0	—	—	ns	
20	TioR	Port output rise time	PIC18C601/801	—	10	25	ns	
20A			PIC18**LC**601/801	—	—	60	ns	
21	TioF	Port output fall time	PIC18C601/801	—	10	25	ns	
21A			PIC18**LC**601/801	—	—	60	ns	
22††	TINP	INT pin high or low time		TCY	—	—	ns	
23††	TRBP	RB7:RB4 change INT high or low time		TCY	—	—	ns	

†† These parameters are asynchronous events, not related to any internal clock edges.

Note 1: Measurements are taken in RC mode, where CLKO pin output is 4 x TOSC.

FIGURE 22-7: PROGRAM MEMORY READ TIMING DIAGRAM

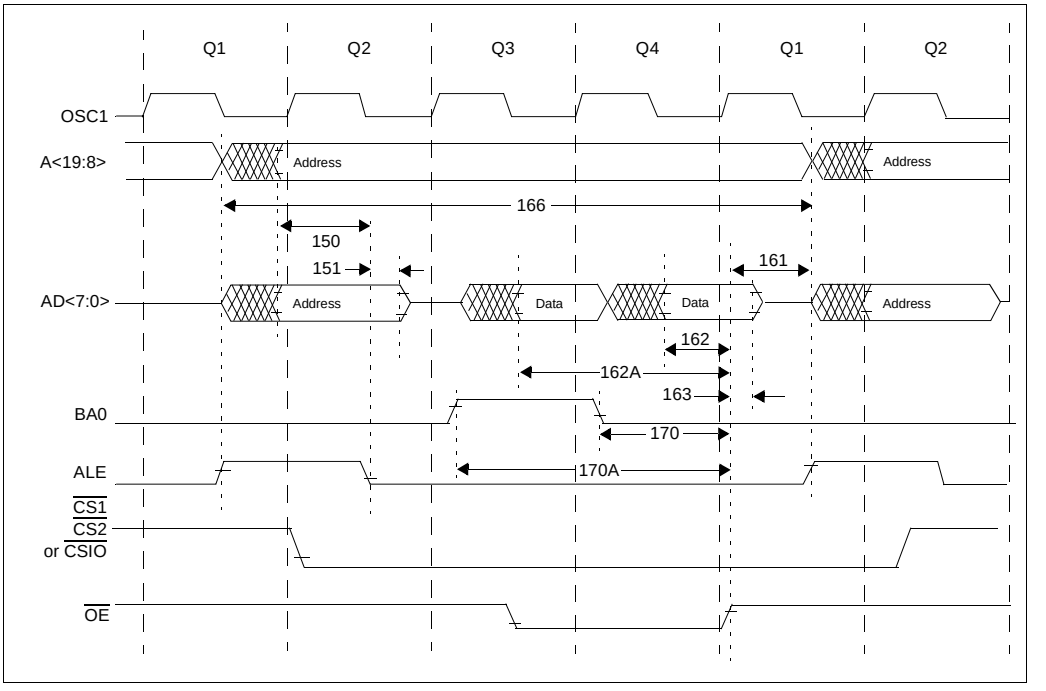


Operating Conditions: 2.0V <VCC <5.5V, -40°C <TA <125°C, unless otherwise stated.

TABLE 22-6: CLKOUT AND I/O TIMING REQUIREMENTS

Param No.	Symbol	Characteristics	Min	Typ	Max	Units
150	TadV2alL	Address out valid to ALE↓ (address setup time)	0.25TCY-10	—	—	ns
151	TalL2adI	ALE↓ to address out invalid (address hold time)	5	—	—	ns
155	TalL2oeL	ALE ↓ to $\overline{OE}$ ↓	10	0.125TCY	—	ns
160	TadZ2oeL	AD high-Z to $\overline{OE}$ ↓ (bus release to $\overline{OE}$)	0	—	—	ns
161	ToeH2adD	$\overline{OE}$ ↑ to AD driven	0.125TCY-5	—	—	ns
162	TadV2oeH	LS data valid before $\overline{OE}$ ↑ (data setup time)	20	—	—	ns
163	ToeH2adI	$\overline{OE}$ ↑ to data in invalid (data hold time)	0	—	—	ns
164	TalH2alL	ALE pulse width	—	TCY	—	ns
165	ToeL2oeH	$\overline{OE}$ pulse width	0.5TCY-5	0.5TCY	—	ns
166	TalH2alH	ALE↑ to ALE↑ (cycle time)	—	0.25TCY	—	ns
167	Tacc	Address valid to data valid	0.75TCY-25	—	—	ns
168	Toe	$\overline{OE}$ ↓ to data valid		—	0.5TCY-25	ns
169	TalL2oeH	ALE ↓ to $\overline{OE}$ ↑	0.625TCY-10	—	0.625TCY+10	ns
171	TalH2csL	Chip select active to ALE ↓	0.25TCY-20	—	—	ns
171A	TubL2oeH	AD valid to chip select active	—	—	10	ns

FIGURE 22-8: 8-BIT PROGRAM MEMORY FETCH TIMING DIAGRAM

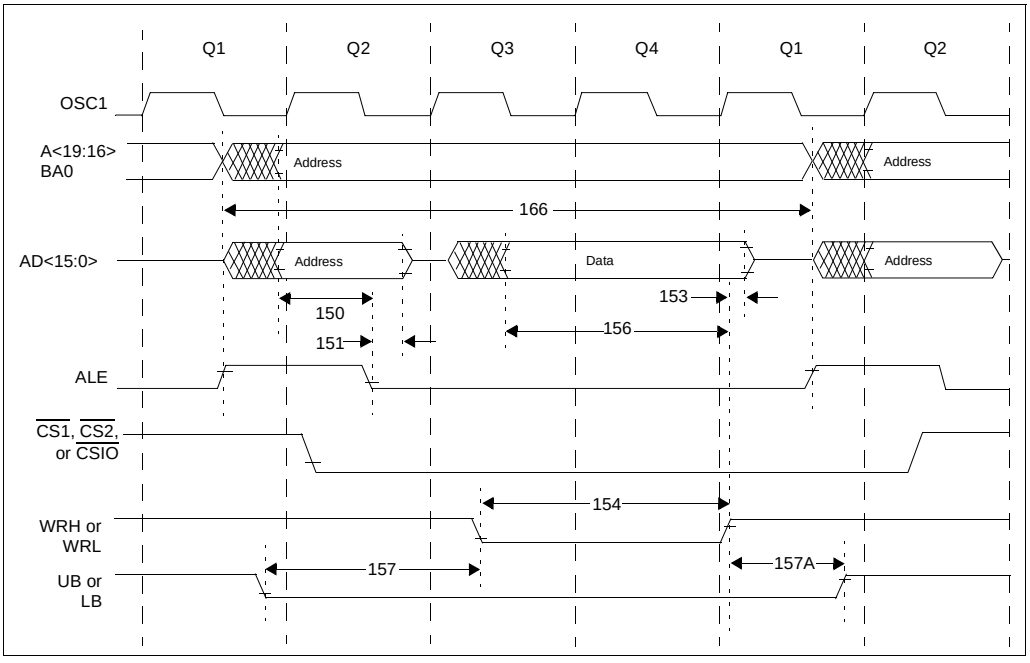


Operating Conditions: 2.0V <VCC <5.5V, -40°C <TA <125°C, FOSC max = 25MHz, unless otherwise stated.

TABLE 22-7: 8-BIT PROGRAM MEMORY FETCH TIMING REQUIREMENTS

Param No.	Symbol	Characteristics	Min	Typ	Max	Units
150	TadV2alL	Address out valid to ALE↓ (address setup time)	0.25TCY-10	—		ns
151	TalL2adI	ALE↓ to address out invalid (address hold time)	5	—	—	ns
161	ToeH2adD	$\overline{OE}$ ↑ to AD driven	0.125TCY-5	—	—	ns
162	TadV2oeH	LS data valid before $\overline{OE}$ ↑ (data setup time)	20	—	—	ns
162A	TadV2oeH	MS data valid before $\overline{OE}$ ↑ (data setup time)	0.25TCY+20	—	—	ns
163	ToeH2adI	$\overline{OE}$ ↑ to data in invalid (data hold time)	0	—	—	ns
166	TalH2alH	ALE↑ to ALE↑ (cycle time)	—	0.25TCY	—	ns
170	TubH2oeH	BA0 = 0 valid before $\overline{OE}$ ↑	0.25TCY-10	—	—	ns
170A	TubL2oeH	BA0 = 1 valid before $\overline{OE}$ ↑	0.5TCY-10	—	—	ns

FIGURE 22-9: PROGRAM MEMORY WRITE TIMING DIAGRAM

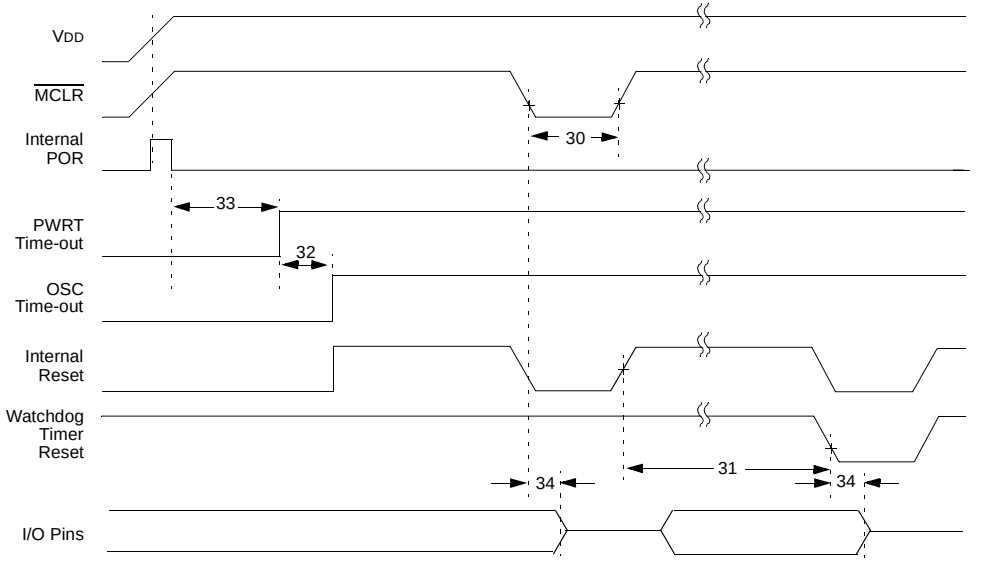


Operating Conditions: 2.0V <VCC <5.5V, -40°C <TA <125°C unless otherwise stated.

TABLE 22-8: PROGRAM MEMORY WRITE TIMING REQUIREMENTS

Param No.	Symbol	Characteristics	Min	Typ	Max	Units
150	TadV2alL	Address out valid to ALE↓ (address setup time)	0.25TCY-10	—	—	ns
151	TalL2adl	ALE↓ to address out invalid (address hold time)	5	—	—	ns
153	TwrH2adl	WRn ↑ to data out invalid (data hold time)	5	—	—	ns
154	TwrL	WRn pulse width	0.5TCY-5	0.5TCY	—	ns
156	TadV2wrH	Data valid before WRn ↑ (data setup time)	0.5TCY-10	—	—	ns
157	TbsV2wrL	Byte select valid before WRn ↓ (byte select setup time)	0.25TCY	—	—	ns
157A	TwrH2bsl	WRn ↑ to byte select invalid (byte select hold time)	0.125TCY-5	—	—	ns
166	TalH2alH	ALE↑ to ALE↑ (cycle time)	—	0.25TCY	—	ns
36	TIVRST	Time for Internal Reference Voltage to become stable	—	20	50	µs

FIGURE 22-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

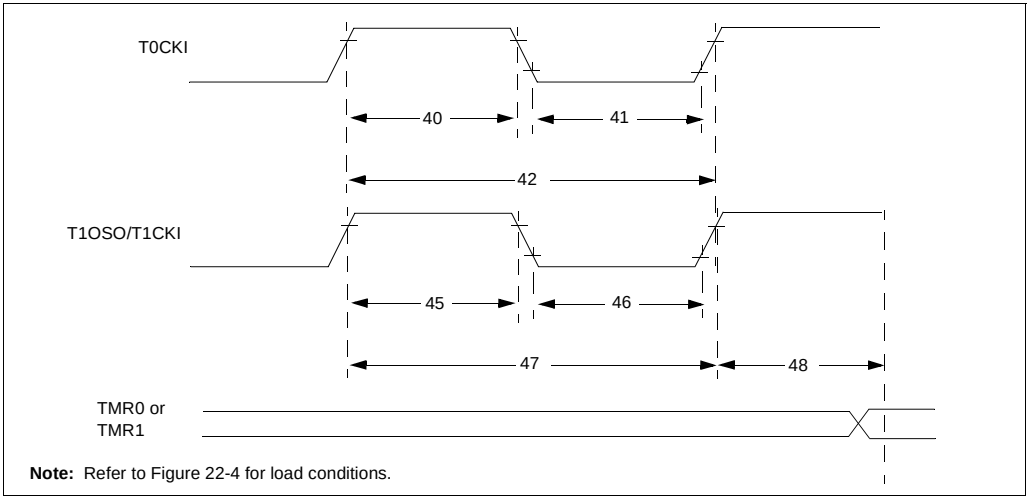


Note: Refer to Figure 22-4 for load conditions.

TABLE 22-9: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)	2	—	—	µs	
31	TWDT	Watchdog Timer Time-out Period (No Prescaler)	7	18	33	ms	
32	TOST	Oscillation Start-up Timer Period	—	—	1024TOSC	—	TOSC = OSC1 period
33	TPWRT	Power up Timer Period	28	72	132	ms	
34	TIOZ	I/O Hi-Impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset	—	2	—	µs	

FIGURE 22-11: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

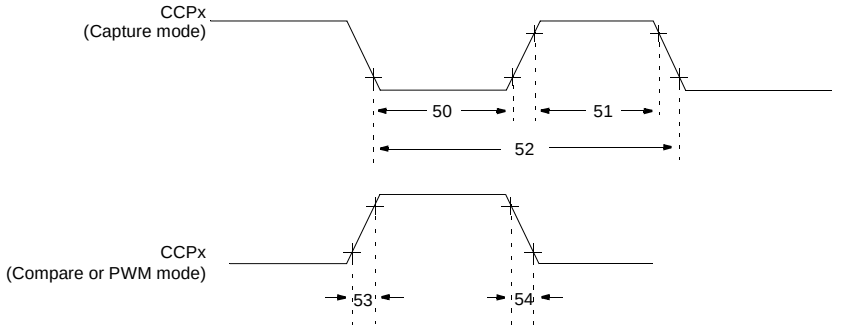


Note: Refer to Figure 22-4 for load conditions.

TABLE 22-10: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Symbol	Characteristic			Min	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width		No Prescaler	0.5TCY + 20	—	ns	
				With Prescaler	10	—	ns	
41	Tt0L	T0CKI Low Pulse Width		No Prescaler	0.5TCY + 20	—	ns	
				With Prescaler	10	—	ns	
42	Tt0P	T0CKI Period		No Prescaler	TCY + 10	—	ns	
				With Prescaler	Greater of: 20 ns or $\frac{T_{CY} + 40}{N}$	—	ns	N = prescale value (1, 2, 4,..., 256)
45	Tt1H	T1CKI High Time	Synchronous, no prescaler		0.5TCY + 20	—	ns	
			Synchronous, with prescaler	PIC18C601/801	10	—	ns	
				PIC18**LC**601/801	25	—	ns	
			Asynchronous	PIC18C601/801	30	—	ns	
				PIC18**LC**601/801	50	—	ns	
46	Tt1L	T1CKI Low Time	Synchronous, no prescaler		0.5TCY + 5	—	ns	
			Synchronous, with prescaler	PIC18C601/801	10	—	ns	
				PIC18**LC**601/801	25	—	ns	
			Asynchronous	PIC18C601/801	30	—	ns	
				PIC18**LC**601/801	TBD	TBD	ns	
47	Tt1P	T1CKI input period	Synchronous		Greater of: 20 ns or $\frac{T_{CY} + 40}{N}$	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous		60	—	ns	
	Ft1	T1CKI oscillator input frequency range			DC	50	kHz	
48	Tcke2tmrl	Delay from external T1CKI clock edge to timer increment			2TOSC	7TOSC	—	

FIGURE 22-12: CAPTURE/COMPARE/PWM TIMINGS (CCP1 AND CCP2)

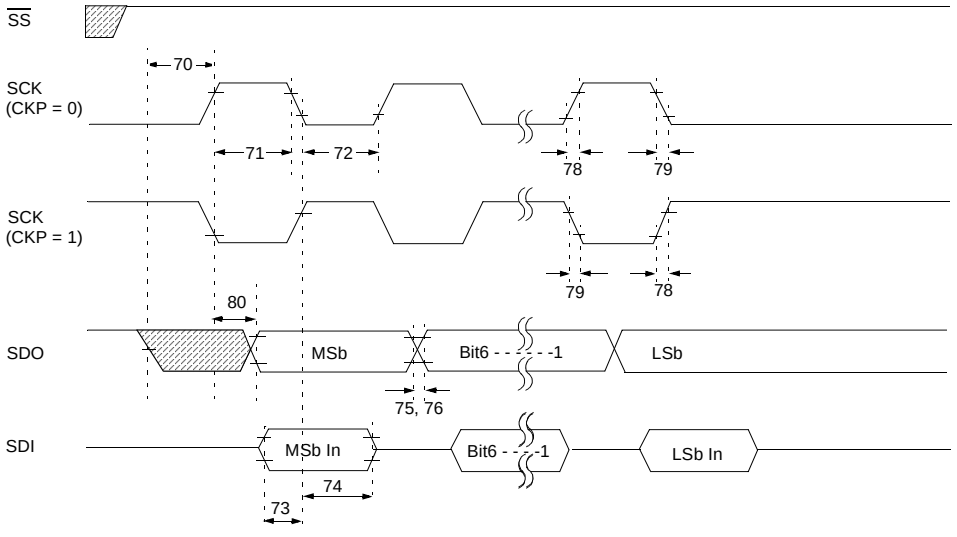


Note: Refer to Figure 22-4 for load conditions.

TABLE 22-11: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP1 AND CCP2)

Param. No.	Symbol	Characteristic			Min	Max	Units	Conditions
50	TccL	CCPx input low time	No Prescaler		0.5TCY + 20	—	ns	
			With Prescaler	PIC18**C**601/801	10	—	ns	
				PIC18**LC**601/801	20	—	ns	
51	TccH	CCPx input high time	No Prescaler		0.5TCY + 20	—	ns	
			With Prescaler	PIC18**C**601/801	10	—	ns	
				PIC18**LC**601/801	20	—	ns	
52	TccP	CCPx input period			$\frac{3T_{CY} + 40}{N}$	—	ns	N = prescale value (1, 4 or 16)
53	TccR	CCPx output fall time		PIC18**C**601/801	—	25	ns	
				PIC18**LC**601/801	—	45	ns	
54	TccF	CCPx output fall time		PIC18**C**601/801	—	25	ns	
				PIC18**LC**601/801	—	45	ns	

FIGURE 22-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)



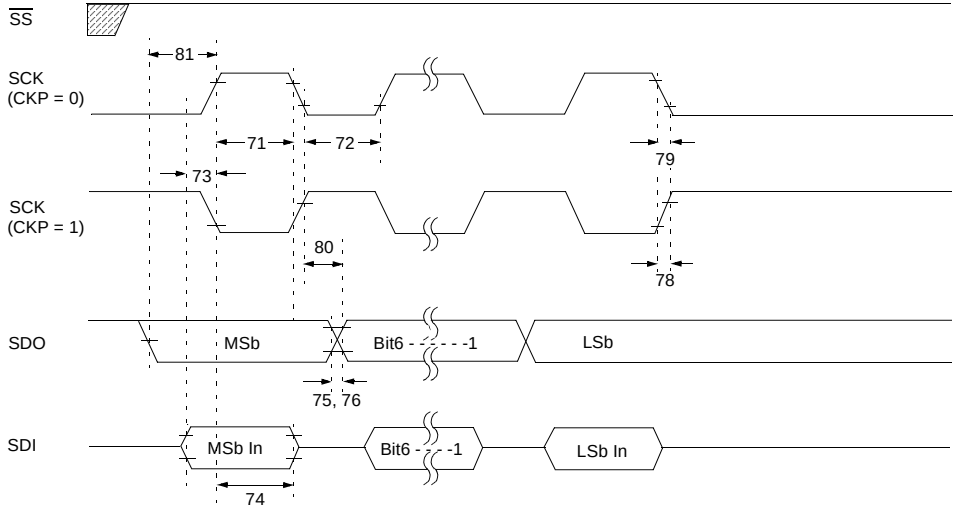
Note: Refer to Figure 22-4 for load conditions.

TABLE 22-12: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}\downarrow$ to SCK$\downarrow$ or SCK$\uparrow$ input		TCY	—	ns	
71	TscH	SCK input high time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TscL	SCK input low time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	ns	
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5TCY + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801	—	45	ns	
76	TdoF	SDO data output fall time		—	25	ns	
78	TscR	SCK output rise time (Master mode)	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801	—	45	ns	
79	TscF	SCK output fall time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18**C**601/801	—	50	ns	
			PIC18**LC**601/801	—	100	ns	

Note 1: Requires the use of parameter # 73A.
2: Only if parameter #s 71A and 72A are used.

FIGURE 22-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)



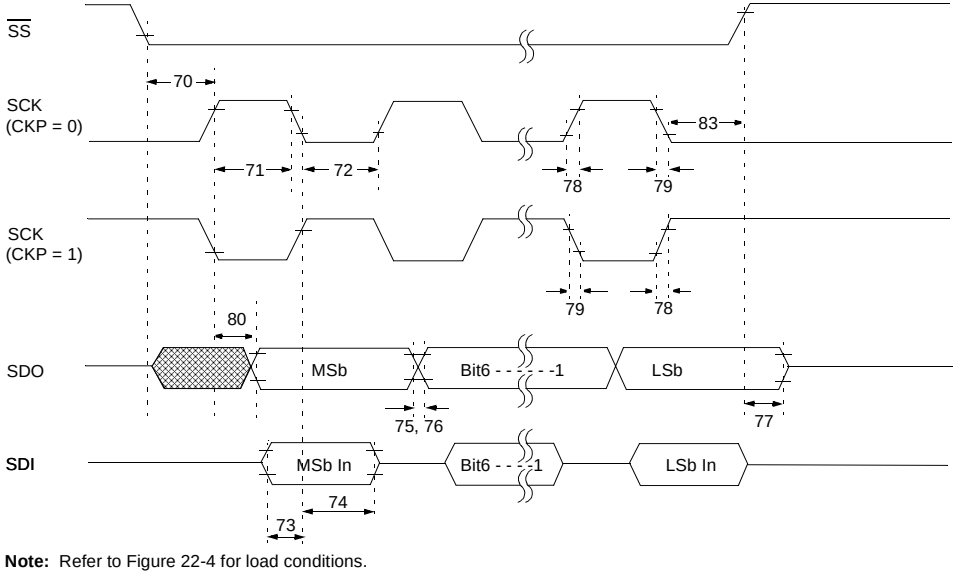
Note: Refer to Figure 22-4 for load conditions.

TABLE 22-13: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
71	TscH	SCK input high time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TscL	SCK input low time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	ns	
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5TCY + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801	—	45	ns	
76	TdoF	SDO data output fall time		—	25	ns	
78	TscR	SCK output rise time (Master mode)	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801	—	45	ns	
79	TscF	SCK output fall time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18**C**601/801	—	50	ns	
			PIC18**LC**601/801	—	100	ns	
81	TdoV2scH, TdoV2scL	SDO data output setup to SCK edge		TCY	—	ns	

Note 1: Requires the use of parameter # 73A.
2: Only if parameter #s 71A and 72A are used.

FIGURE 22-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)



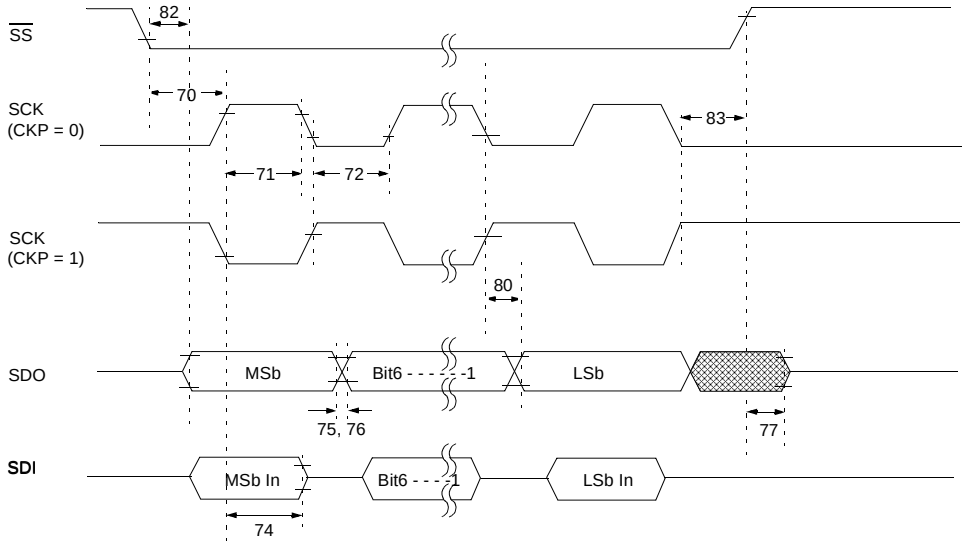
Note: Refer to Figure 22-4 for load conditions.

TABLE 22-14: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING (CKE = 0))

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}\downarrow$ to SCK$\downarrow$ or SCK$\uparrow$ input		TCY	—	ns	
71	TscH	SCK input high time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TscL	SCK input low time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	ns	
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5TCY + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801		45	ns	
76	TdoF	SDO data output fall time		—	25	ns	
77	TssH2doZ	$\overline{SS}\uparrow$ to SDO output hi-impedance		10	50	ns	
78	TscR	SCK output rise time (Master mode)	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801		45	ns	
79	TscF	SCK output fall time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18**C**601/801	—	50	ns	
			PIC18**LC**601/801		100	ns	
83	TscH2ssH, TscL2ssH	$\overline{SS}\uparrow$ after SCK edge		1.5TCY + 40	—	ns	

Note 1: Requires the use of parameter # 73A.
2: Only if parameter #s 71A and 72A are used.

FIGURE 22-16: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)



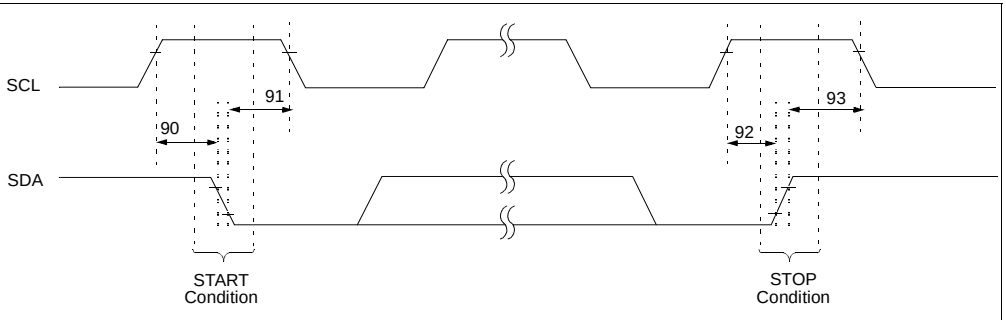
Note: Refer to Figure 22-4 for load conditions.

TABLE 22-15: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}\downarrow$ to SCK↓ or SCK↑ input		TCY	—	ns	
71	TscH	SCK input high time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TscL	SCK input low time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5TCY + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801	—	45	ns	
76	TdoF	SDO data output fall time		—	25	ns	
77	TssH2doZ	$\overline{SS}\uparrow$ to SDO output hi-impedance		10	50	ns	
78	TscR	SCK output rise time (Master mode)	PIC18**C**601/801	—	25	ns	
			PIC18**LC**601/801	—	45	ns	
79	TscF	SCK output fall time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18**C**601/801	—	50	ns	
			PIC18**LC**601/801	—	100	ns	
82	TssL2doV	SDO data output valid after $\overline{SS}\downarrow$ edge	PIC18**C**601/801	—	50	ns	
			PIC18**LC**601/901	—	100	ns	
83	TscH2ssH, TscL2ssH	$\overline{SS}\uparrow$ after SCK edge		1.5TCY + 40	—	ns	

Note 1: Requires the use of parameter # 73A.
2: Only if parameter #s 71A and 72A are used.

FIGURE 22-17: I²C BUS START/STOP BITS TIMING

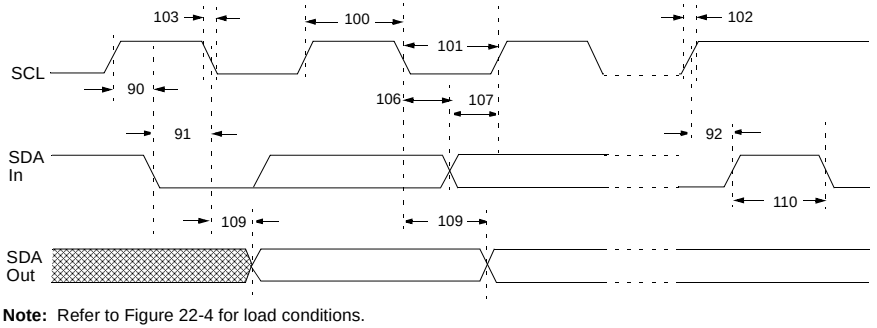


Note: Refer to Figure 22-4 for load conditions.

TABLE 22-16: I²C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	START condition Setup time	100 kHz mode	4700	—	ns	Only relevant for Repeated START condition
			400 kHz mode	600	—		
91	THD:STA	START condition Hold time	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—		
92	TSU:STO	STOP condition Setup time	100 kHz mode	4700	—	ns	
			400 kHz mode	600	—		
93	THD:STO	STOP condition Hold time	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—		

FIGURE 22-18: I²C BUS DATA TIMING



Note: Refer to Figure 22-4 for load conditions.

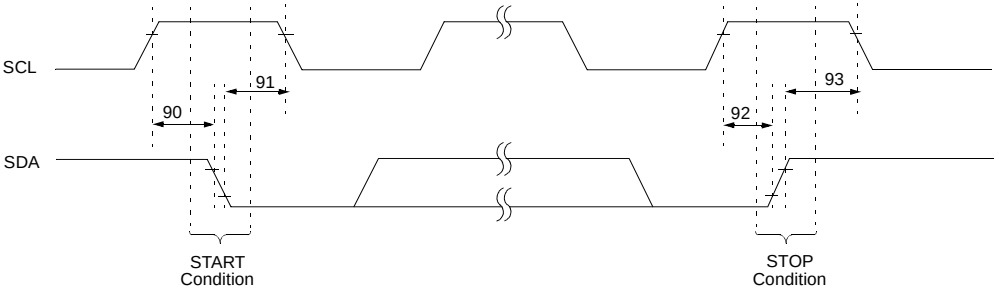
TABLE 22-17: I²C BUS DATA REQUIREMENTS (SLAVE MODE)

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock high time	100 kHz mode	4.0	—	µs	PIC18C601/801 must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	µs	PIC18C601/801 must operate at a minimum of 10 MHz
			SSP Module	1.5TCY	—		
101	TLOW	Clock low time	100 kHz mode	4.7	—	µs	PIC18C601/801 must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	µs	PIC18C601/801 must operate at a minimum of 10 MHz
			SSP module	1.5TCY	—	ns	
102	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10 to 400 pF
103	TF	SDA and SCL fall time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10 to 400 pF
90	TSU:STA	START condition setup time	100 kHz mode	4.7	—	µs	Only relevant for Repeated START condition
			400 kHz mode	0.6	—	µs	
91	THD:STA	START condition hold time	100 kHz mode	4.0	—	µs	After this period the first clock pulse is generated
			400 kHz mode	0.6	—	µs	
106	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	µs	
107	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
92	TSU:STO	STOP condition setup time	100 kHz mode	4.7	—	µs	
			400 kHz mode	0.6	—	µs	
109	TAA	Output valid from clock	100 kHz mode	—	3500	ns	**(Note 1)**
			400 kHz mode	—	—	ns	
110	TBUF	Bus free time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
D102	Cb	Bus capacitive loading		—	400	pF	

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of START or STOP conditions.

2: A fast mode I²C bus device can be used in a standard mode I²C bus system, but the requirement tsu;DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line.
Before the SCL line is released, TR max. + tsu;DAT = 1000 + 250 = 1250 ns (according to the standard mode I²C bus specification).

FIGURE 22-19: MASTER SSP I²C BUS START/STOP BITS TIMING WAVEFORMS



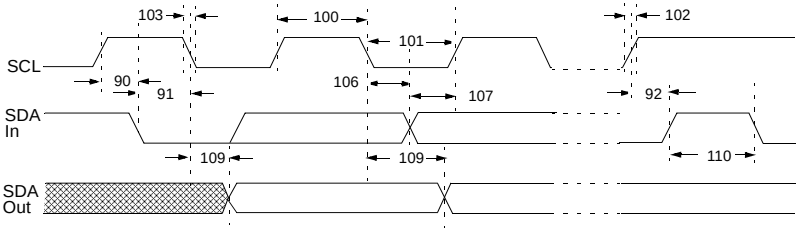
Note: Refer to Figure 22-4 for load conditions.

TABLE 22-18: MASTER SSP I²C BUS START/STOP BITS REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	START condition Setup time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns	Only relevant for Repeated START condition
			400 kHz mode	2(Tosc)(BRG + 1)	—		
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—		
91	THD:STA	START condition Hold time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	2(Tosc)(BRG + 1)	—		
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—		
92	TSU:STO	STOP condition Setup time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns	
			400 kHz mode	2(Tosc)(BRG + 1)	—		
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—		
93	THD:STO	STOP condition Hold time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns	
			400 kHz mode	2(Tosc)(BRG + 1)	—		
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—		

Note 1: Maximum pin capacitance = 10 pF for all I²C pins.

FIGURE 22-20: MASTER SSP I²C BUS DATA TIMING



Note: Refer to Figure 22-4 for load conditions.

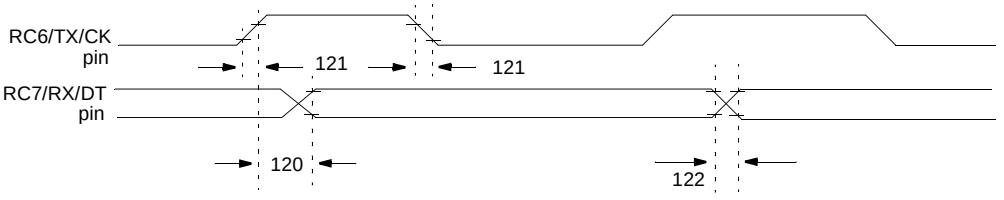
TABLE 22-19: MASTER SSP I²C BUS DATA REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock high time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—	ms	
101	TLOW	Clock low time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—	ms	
102	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1Cb	300	ns	
			1 MHz mode(1)	—	300	ns	
103	TF	SDA and SCL fall time	100 kHz mode	—	300	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1Cb	300	ns	
			1 MHz mode(1)	—	100	ns	
90	TSU:STA	START condition setup time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	Only relevant for Repeated START condition
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—	ms	
91	THD:STA	START condition hold time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	After this period, the first clock pulse is generated
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—	ms	
106	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	ms	
			1 MHz mode(1)	TBD	—	ns	
107	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
			1 MHz mode(1)	TBD	—	ns	
92	TSU:STO	STOP condition setup time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(Tosc)(BRG + 1)	—	ms	
109	TAA	Output valid from clock	100 kHz mode	—	3500	ns	
			400 kHz mode	—	1000	ns	
			1 MHz mode(1)	—	—	ns	
110	TBUF	Bus free time	100 kHz mode	4.7	—	ms	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	ms	
			1 MHz mode(1)	TBD	—	ms	
D102	Cb	Bus capacitive loading		—	400	pF	

Note 1: Maximum pin capacitance = 10 pF for all I²C pins.

2: A fast mode I²C bus device can be used in a standard mode I²C bus system, but parameter #107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line. Before the SCL line is released, parameter #102 + parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode).

FIGURE 22-21: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

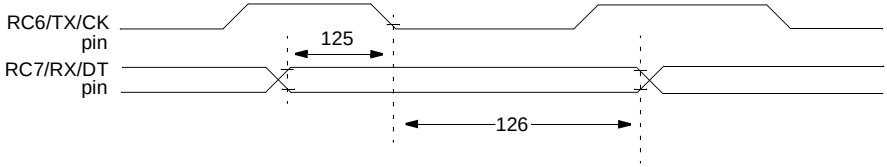


Note: Refer to Figure 22-4 for load conditions.

TABLE 22-20: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
120	TckH2dtV	SYNC XMIT (Master & Slave) Clock high to data-out valid	PIC18**C**601/801	—	40	ns	
			PIC18**LC**601/801	—	100	ns	
121	Tckrf	Clock out rise time and fall time (Master mode)	PIC18**C**601/801	—	20	ns	
			PIC18**LC**601/801	—	50	ns	
122	Tdtrf	Data-out rise time and fall time	PIC18**C**601/801	—	20	ns	
			PIC18**LC**601/801	—	50	ns	

FIGURE 22-22: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 22-4 for load conditions.

TABLE 22-21: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TdtV2ckl	SYNC RCV (Master & Slave) Data-hold before CK ↓ (DT hold time)	10	—	ns	
126	TckL2dtl	Data-hold after CK ↓ (DT hold time)	15	—	ns	

TABLE 22-22: A/D CONVERTER CHARACTERISTICS: PIC18C601/801 (INDUSTRIAL, EXTENDED) PIC18LC601/801 (INDUSTRIAL)

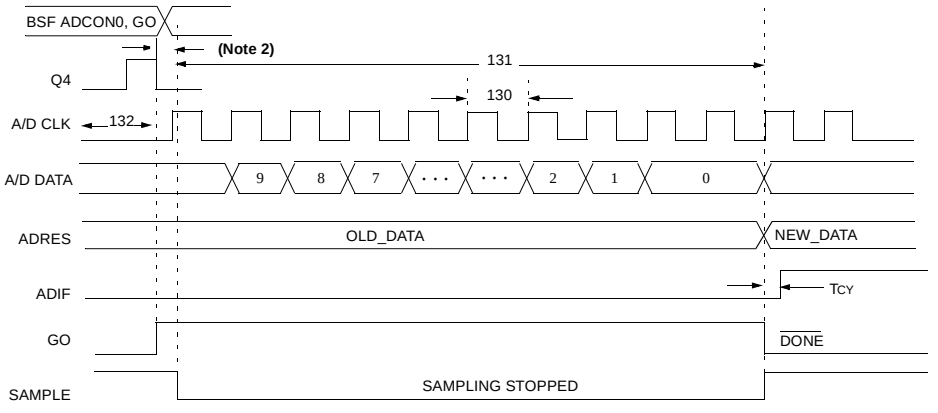
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	—	—	10	bit	VREF = VDD ≥ 3.0V
			—	—	TBD	bit	VREF = VDD < 3.0V
A03	EIL	Integral linearity error	—	—	<±1	LSb	VREF = VDD ≥ 3.0V
			—	—	TBD	LSb	VREF = VDD < 3.0V
A04	EDL	Differential linearity error	—	—	<±1	LSb	VREF = VDD ≥ 3.0V
			—	—	TBD	LSb	VREF = VDD < 3.0V
A05	EFS	Full scale error	—	—	<±1	LSb	VREF = VDD ≥ 3.0V
			—	—	TBD	LSb	VREF = VDD < 3.0V
A06	EOFF	Offset error	—	—	<±1	LSb	VREF = VDD ≥ 3.0V
			—	—	TBD	LSb	VREF = VDD < 3.0V
A10	—	Monotonicity	guaranteed(3)			—	VSS ≤ VAIN ≤ VREF
A20	VREF	Reference voltage	0	—	—	V	
A20A		(VREFH - VREFL)	3	—	—	V	For 10-bit resolution
A21	VREFH	Reference voltage High	AVSS	—	AVDD + 0.3 V	V	
A22	VREFL	Reference voltage Low	AVSS - 0.3 V	—	AVDD	V	
A25	VAIN	Analog input voltage	AVSS - 0.3 V	—	VREF + 0.3 V	V	
A30	ZAIN	Recommended impedance of analog voltage source	—	—	10.0	kΩ	
A40	IAD	A/D conversion current (VDD)	PIC18**C**601/801 —	180	—	μA	Average current consumption when A/D is on(1)
			PIC18**LC**601/801 —	90	—	μA	
A50	IREF	VREF input current(2)	10	—	1000	μA	During VAIN acquisition. Based on differential of VHOLD to VAIN. To charge CHOLD, see Section 17.0.
			—	—	10	μA	During A/D conversion cycle.

Note 1: When A/D is off, it will not consume any current other than minor leakage current. The power-down current spec includes any such leakage from the A/D module.
VREF current is from RA2/AN2/VREF- and RA3/AN3/VREF+ pins or AVDD and AVSS pins, whichever is selected as reference input.

2: VSS ≤ VAIN ≤ VREF

3: The A/D conversion result either increases or remains constant as the analog input increases.

FIGURE 22-23: A/D CONVERSION TIMING



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

2: This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.

TABLE 22-23: A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D clock period	PIC18**C**601/801	1.6	20(5)	µs	TOSC based, VREF ≥ 3.0V
			PIC18**LC**601/801	3.0	20(5)	µs	TOSC based, VREF full range
			PIC18**C**601/801	2.0	6.0	µs	A/D RC mode
			PIC18**LC**601/801	3.0	9.0	µs	A/D RC mode
131	TCNV	Conversion time (not including acquisition time)(1)		11	12	TAD	
132	TACQ	Acquisition time(3)		15 10	— —	µs µs	-40°C ≤ Temp ≤ 125°C 0°C ≤ Temp ≤ 125°C
135	TSWC	Switching time from convert → sample		—	(Note 4)		
136	TAMP	Amplifier settling time(2)		1	—	µs	This may be used if the "new" input voltage has not changed by more than 1 LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).

Note 1: ADRES register may be read on the following TCY cycle.

2: See Section 17.0 for minimum conditions, when input voltage has changed more than 1 LSb.

3: The time for the holding capacitor to acquire the "New" input voltage, when the voltage changes full scale after the conversion (AVDD to AVSS, or AVSS to AVDD). The source impedance (*RS*) on the input channels is 50Ω.

4: On the next Q4 cycle of the device clock.

5: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

NOTES:

23.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

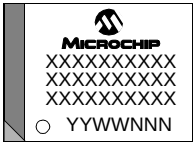
Graphs and Tables are not available at this time.

NOTES:

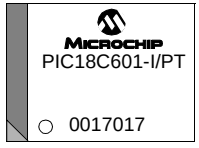
24.0 PACKAGING INFORMATION

24.1 Package Marking Information

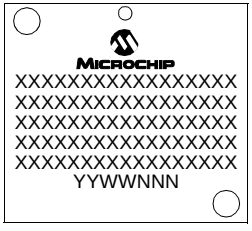
64-Lead TQFP



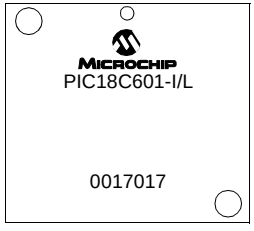
Example



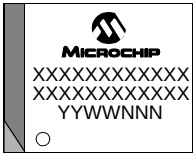
68-Lead PLCC



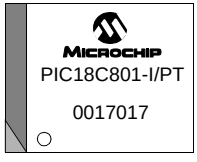
Example



80-Lead TQFP



Example



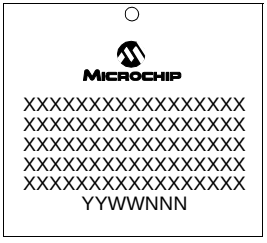
Legend:	XX...X	Customer specific information*
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code

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.

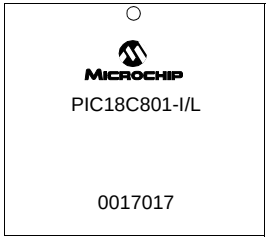
***** Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

Package Marking Information (Cont'd)

84-Lead PLCC

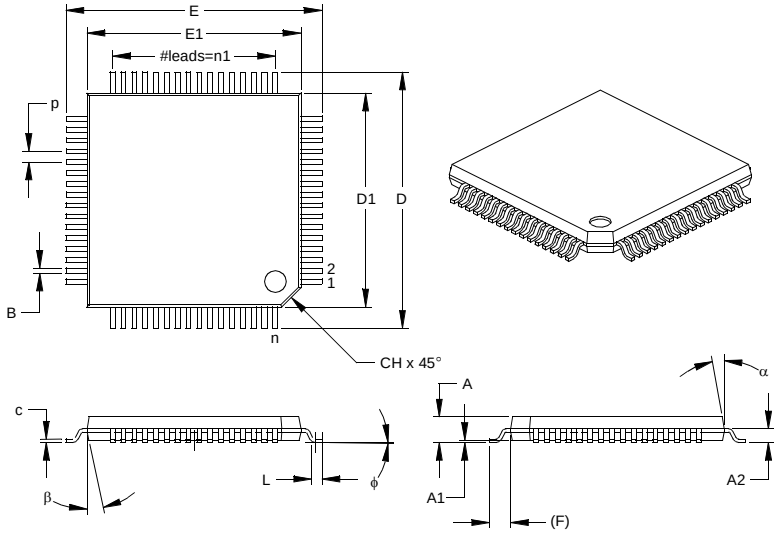


Example



64-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

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



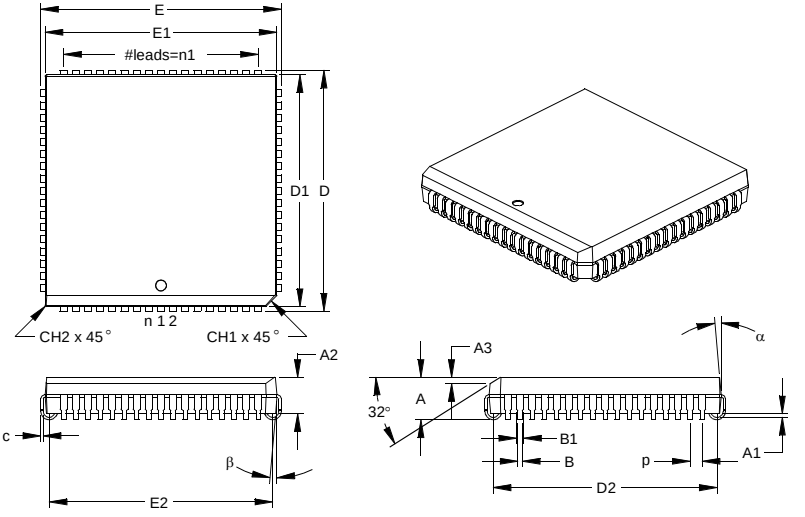
Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		64			64	
Pitch	p		.020			0.50	
Pins per Side	n1		16			16	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.006	.010	0.05	0.15	0.25
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039			1.00	
Foot Angle	φ	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.005	.007	.009	0.13	0.18	0.23
Lead Width	B	.007	.009	.011	0.17	0.22	0.27
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-085

68-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)

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



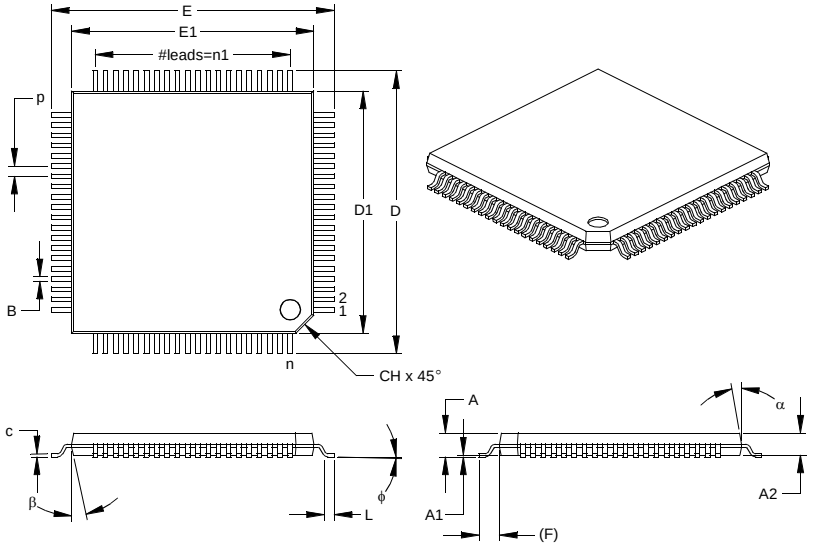
Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		68			68	
Pitch	p		.050			1.27	
Pins per Side	n1		17			17	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.985	.990	.995	25.02	25.15	25.27
Overall Length	D	.985	.990	.995	25.02	25.15	25.27
Molded Package Width	E1	.950	.954	.958	24.13	24.23	24.33
Molded Package Length	D1	.950	.954	.958	24.13	24.23	24.33
Footprint Width	E2	.890	.920	.930	22.61	23.37	23.62
Footprint Length	D2	.890	.920	.930	22.61	23.37	23.62
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-049

80-Lead Plastic Thin Quad Flatpack (PT) 12x12x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

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



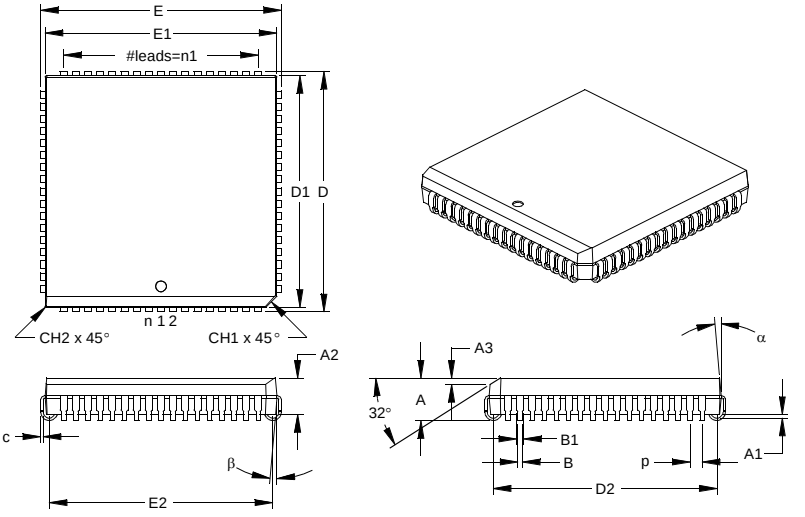
Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		80			80	
Pitch	p		.020			0.50	
Pins per Side	n1		20			20	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039			1.00	
Foot Angle	ϕ	0	3.5	7	0	3.5	7
Overall Width	E	.541	.551	.561	13.75	14.00	14.25
Overall Length	D	.541	.551	.561	13.75	14.00	14.25
Molded Package Width	E1	.463	.472	.482	11.75	12.00	12.25
Molded Package Length	D1	.463	.472	.482	11.75	12.00	12.25
Lead Thickness	c	.004	.006	.008	0.09	0.15	0.20
Lead Width	B	.007	.009	.011	0.17	0.22	0.27
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-092

84-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)

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



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		68			68	
Pitch	p		.050			1.27	
Pins per Side	n1		17			17	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.985	.990	.995	25.02	25.15	25.27
Overall Length	D	.985	.990	.995	25.02	25.15	25.27
Molded Package Width	E1	.950	.954	.958	24.13	24.23	24.33
Molded Package Length	D1	.950	.954	.958	24.13	24.23	24.33
Footprint Width	E2	.890	.920	.930	22.61	23.37	23.62
Footprint Length	D2	.890	.920	.930	22.61	23.37	23.62
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-093

APPENDIX A: DATA SHEET REVISION HISTORY

Revision A

This is a new data sheet.

Revision B (January 2013)

Added a note to each package outline drawing.

APPENDIX B: DEVICE DIFFERENCES

The differences between the PIC18C601/801 devices listed in this data sheet are shown in Table B-1.

TABLE B-1: DEVICE DIFFERENCES

Feature		PIC18C601	PIC18C801
Maximum External Program Memory (Bytes)		256K	2M
Data Memory (Bytes)		1.5K	1.5K
A/D Channels		8	12
Package Types	TQFP	64-pin	80-pin
	PLCC	68-pin	84-pin

APPENDIX C: DEVICE MIGRATIONS

This section is intended to describe the functional and electrical specification differences when migrating between functionally similar devices (such as from a PIC16C74A to a PIC16C74B).

Not Applicable

APPENDIX D: MIGRATING FROM OTHER PIC DEVICES

This discusses some of the issues in migrating from other PIC devices to the PIC18CXXX family of devices.

D.1 PIC16CXXX to PIC18CXXX

See application note AN716.

D.2 PIC17CXXX to PIC18CXXX

See application note AN726.

APPENDIX E: DEVELOPMENT TOOL VERSION REQUIREMENTS

This lists the minimum requirements (software/ firmware) of the specified development tool to support the devices listed in this data sheet.

MPLAB® IDE: TBD

MPLAB® SIMULATOR: TBD

MPLAB® ICE 3000:

PIC18C601/801 Processor Module:
Part Number - TBD

PIC18C601/801 Device Adapter:

Socket	Part Number
64-pin TQFP	TBD
68-pin PLCC	TBD
80-pin TQFP	TBD
84-pin PLCC	TBD

MPLAB® ICD: TBD

PRO MATE® II: TBD

PICSTART® Plus: TBD

MPASM™ Assembler: TBD

MPLAB® C18 C Compiler: TBD

Note:	Please read all associated README.TXT files that are supplied with the development tools. These "read me" files will discuss product support and any known limitations.

NOTES:

INDEX

T

U

W

X

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site 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 web site 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 web site at: http://microchip.com/support

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

TO: Technical Publications Manager
RE: Reader Response

Total Pages Sent _____

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____ FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: PIC16xxxxxx family Literature Number: DS39541B

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?

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	/XX	XXX
Device	Temperature Range	Package	Pattern

Device	PIC18C601/801[1], PIC18C601/801T[2]: VDD range, 4.2V to 5.5V PIC18LC601/801[1], PIC18LC601/801T[2] VDD range, 2.5V to 5.5V
Temperature Range	I = -40°C to +70°C (Industrial) E = -40°C to +125°C (Extended)
Package	PT = TQFP L = PLCC
Pattern	QTP, SQTP, ROM Code (factory specified) or Special Requirements. Blank for OTP and Windowed devices.

Examples:

a) PIC18LC601 - I/L = Industrial temp., PLCC package, Extended VDD limits, 16-bit data bus.

b) PIC18LC801 - E/PT = Extended temp., TQFP package, Extended VDD limits, 16-bit data bus.

Note 1: C = Standard Voltage Range
LC = Wide Voltage Range

2: T = In tape and reel (both PLCC and TQFP packages)

SALES AND SUPPORT

Data Sheets

Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Worldwide Site (www.microchip.com)

NOTES:

NOTES:

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.

Trademarks


Printed on recycled paper.

ISBN: 9781620769331

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

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.

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

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
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
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: 852-2401-1200
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 - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

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

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

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: 91-20-2566-1512
Fax: 91-20-2566-1513

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
Fax: 886-7-330-9305

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 - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/12

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Microchip:

PIC18LC601-I/L PIC18LC801-I/L PIC18C801T-I/L PIC18C601T-I/L PIC18C801-I/PT PIC18C601-I/PT PIC18LC801-I/PT PIC18LC601-I/PT PIC18C601T-I/PT PIC18C601-I/L PIC18LC601T-I/L PIC18LC801T-I/L PIC18C801-I/L PIC18C801T-I/PT PIC18LC801T-I/PT PIC18LC601T-I/PT