

High-Performance Microcontrollers with CAN Module

High Performance RISC CPU:

- C-compiler optimized architecture instruction set
- Linear program memory addressing to 32 Kbytes
- Linear data memory addressing to 4 Kbytes

| Device | Program Memory | | | On-Chip RAM (bytes) |
|-----------|----------------|----------------------------|----------------------------|---------------------|
| | On-Chip | | Off-Chip | |
| | EPROM (bytes) | # Single Word Instructions | Maximum Addressing (bytes) | |
| PIC18C658 | 32 K | 16384 | N/A | 1536 |
| PIC18C858 | 32 K | 16384 | N/A | 1536 |

- Up to 10 MIPS operation:
 - DC - 40 MHz clock input
 - 4 MHz - 10 MHz osc./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 76 I/O with individual direction control
- Four external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time base for PWM)
- 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 6.25 ns
 - Compare is 16-bit, max resolution 100 ns (T_{CY})
 - PWM output: PWM resolution is 1- to 10-bit.
Max. PWM freq. @:8-bit resolution = 156 kHz
10-bit resolution = 39 kHz
- Master Synchronous Serial Port (MSSP) with two modes of operation:
 - 3-wire SPI™ (Supports all 4 SPI modes)
 - I²C™ 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 16 channels available
- Analog Comparator Module:
 - 2 Comparators
 - Programmable input and output multiplexing
- Comparator Voltage Reference Module
- Programmable Low Voltage Detection (LVD) module
 - Supports interrupt on low voltage detection
- Programmable Brown-out Reset (BOR)

CAN BUS Module Features:

- Message bit rates up to 1 Mbps
- Conforms to CAN 2.0B ACTIVE Spec with:
 - 29-bit Identifier Fields
 - 8 byte message length
- 3 Transmit Message Buffers with prioritization
- 2 Receive Message Buffers
- 6 full 29-bit Acceptance Filters
- Prioritization of Acceptance Filters
- Multiple Receive Buffers for High Priority Messages to prevent loss due to overflow
- Advanced Error Management Features

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
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options, including:
 - 4X Phase Lock Loop (of primary oscillator)
 - Secondary Oscillator (32 kHz) clock input
- In-Circuit Serial Programming (ICSP™) via two pins

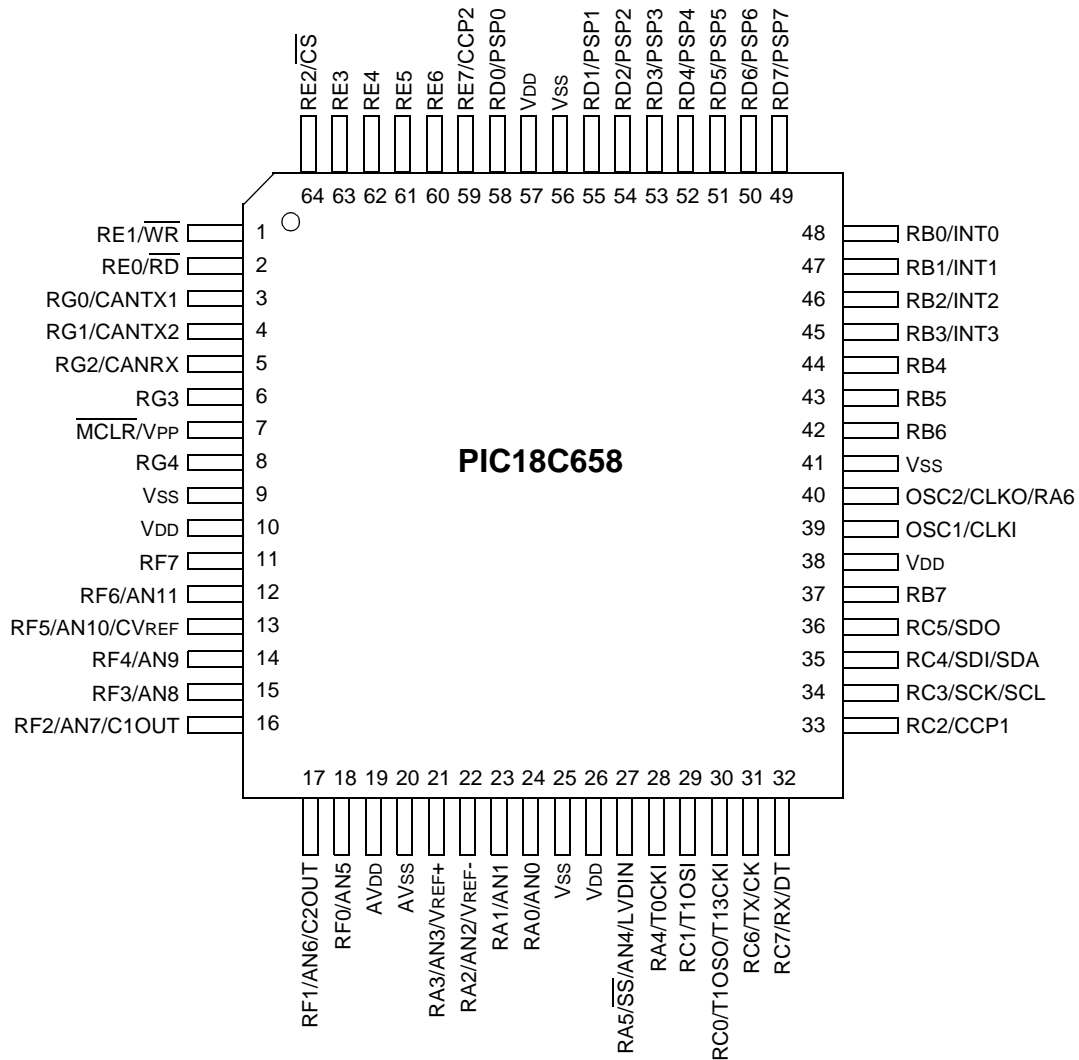
CMOS Technology:

- Low power, high speed EPROM technology
- Fully static design
- Wide operating voltage range (2.5V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption

PIC18CXX8

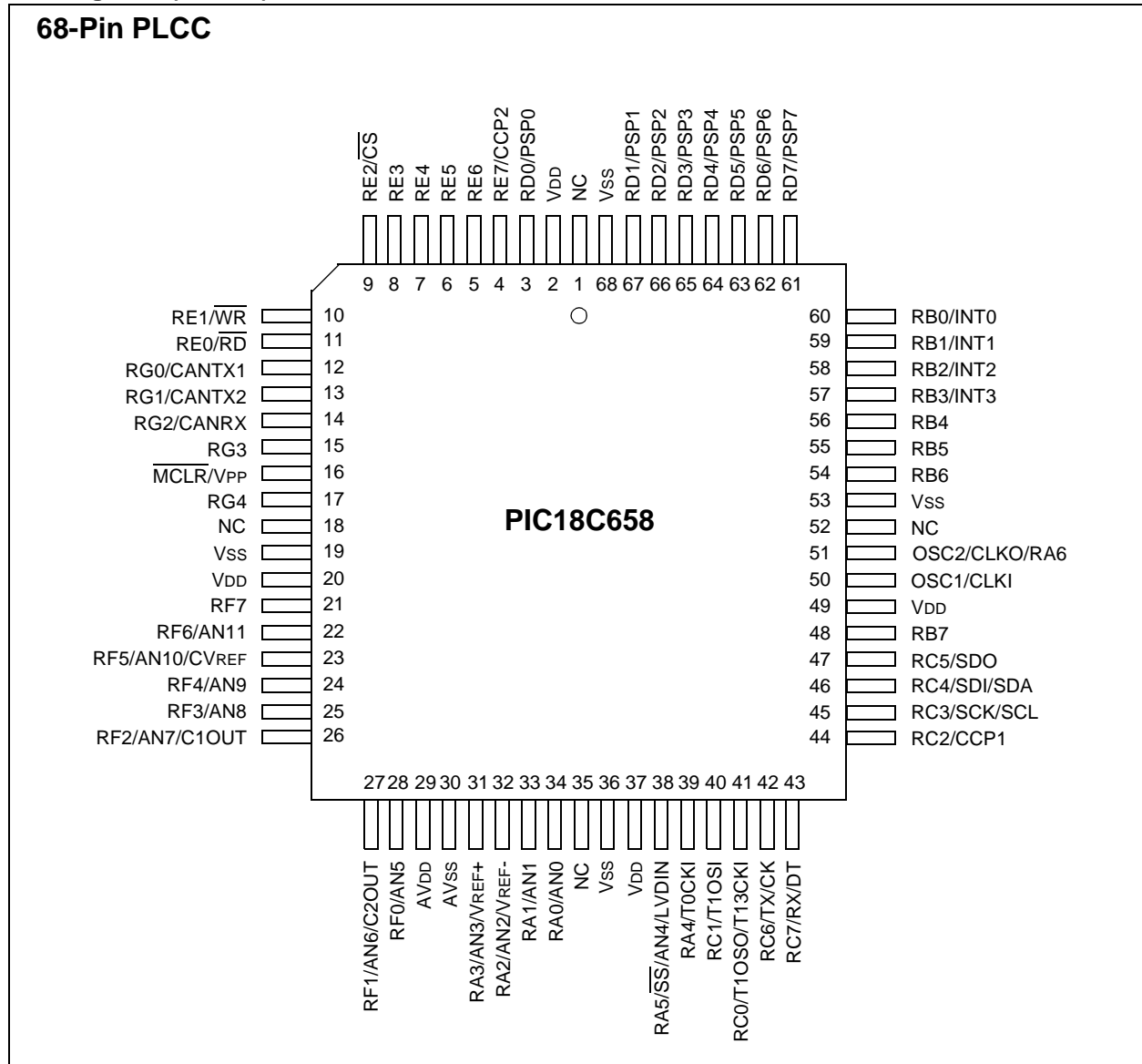
Pin Diagrams

64-Pin TQFP



Pin Diagrams (Cont.'d)

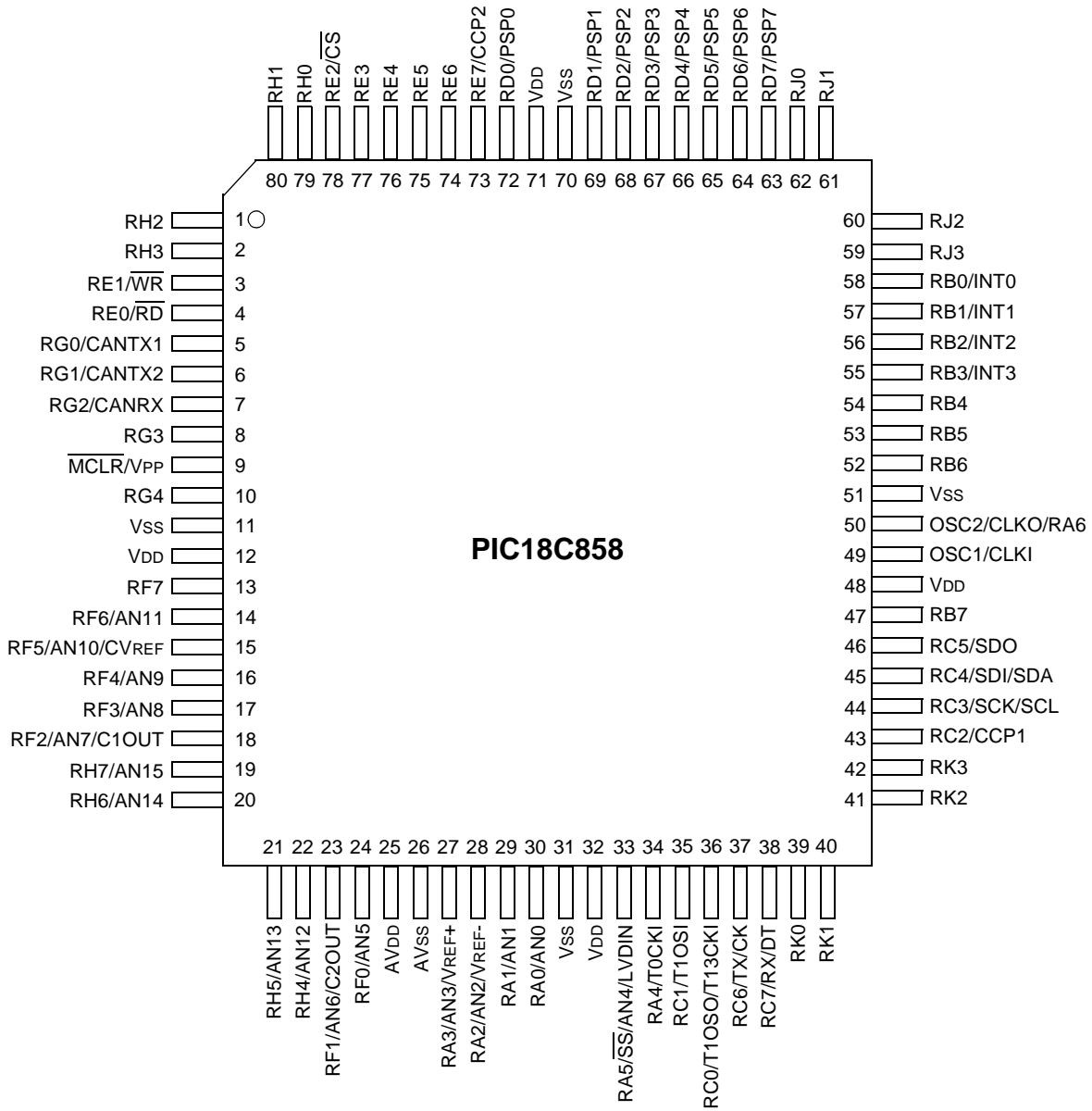
68-Pin PLCC



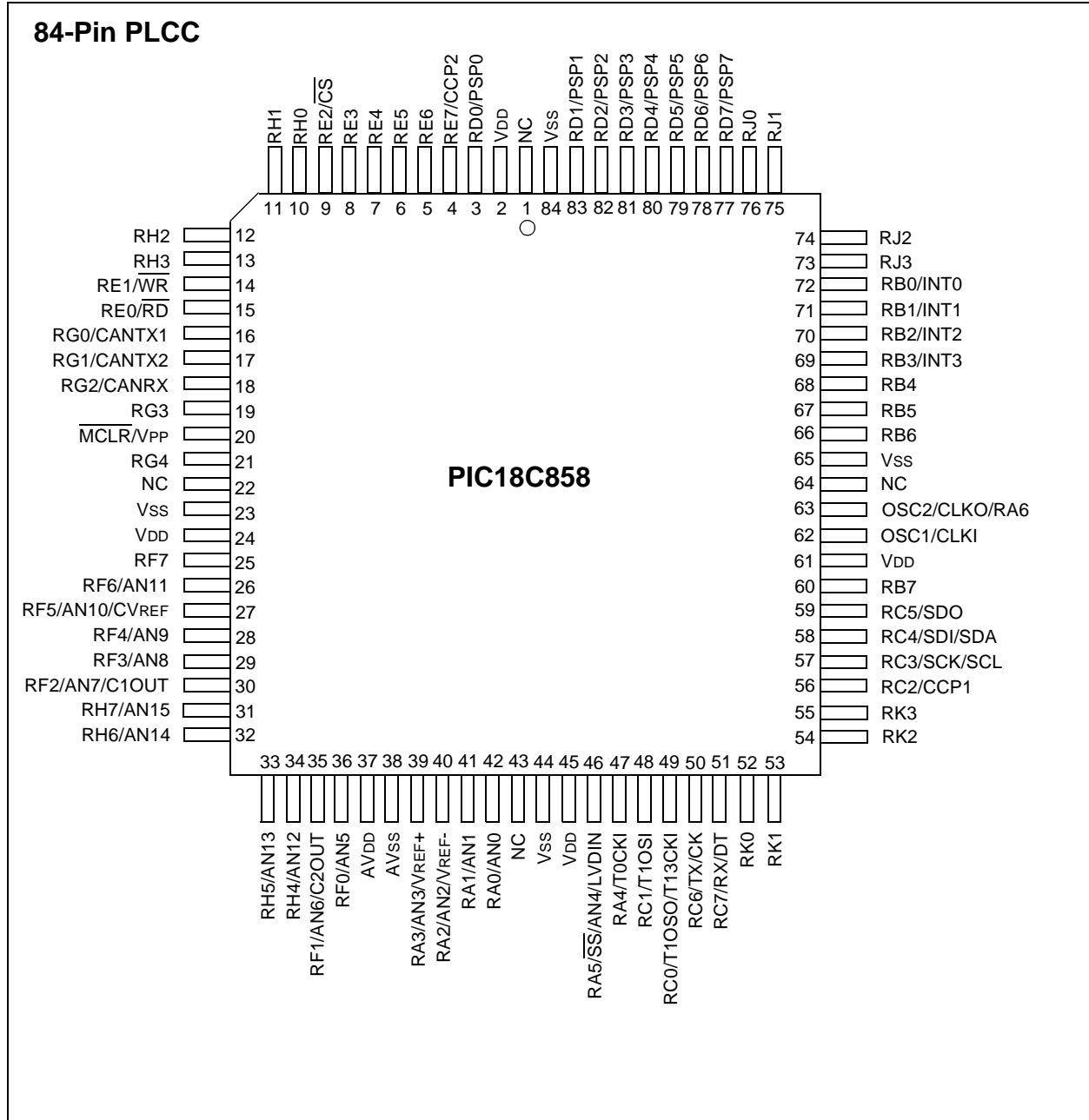
PIC18CXX8

Pin Diagrams (Cont.'d)

80-Pin TQFP



Pin Diagrams (Cont.'d)



PIC18CXX8

Table of Contents

| | | |
|-----------------------------------------------|----------------------------------------------------------------------------------|-----|
| 1.0 | Device Overview | 9 |
| 2.0 | Oscillator Configurations | 21 |
| 3.0 | Reset | 29 |
| 4.0 | Memory Organization | 41 |
| 5.0 | Table Reads/Table Writes | 65 |
| 6.0 | 8 X 8 Hardware Multiplier | 71 |
| 7.0 | Interrupts | 75 |
| 8.0 | I/O Ports | 89 |
| 9.0 | Parallel Slave Port | 109 |
| 10.0 | Timer0 Module | 113 |
| 11.0 | Timer1 Module | 117 |
| 12.0 | Timer2 Module | 121 |
| 13.0 | Timer3 Module | 123 |
| 14.0 | Capture/Compare/PWM (CCP) Modules | 127 |
| 15.0 | Master Synchronous Serial Port (MSSP) Module | 135 |
| 16.0 | Addressable Universal Synchronous Asynchronous Receiver Transmitter (USART)..... | 167 |
| 17.0 | CAN Module | 183 |
| 18.0 | 10-bit Analog-to-Digital Converter (A/D) Module | 227 |
| 19.0 | Comparator Module..... | 237 |
| 20.0 | Comparator Voltage Reference Module | 243 |
| 21.0 | Low Voltage Detect | 247 |
| 22.0 | Special Features of the CPU | 251 |
| 23.0 | Instruction Set Summary | 261 |
| 24.0 | Development Support..... | 305 |
| 25.0 | Electrical Characteristics | 311 |
| 26.0 | DC and AC Characteristics Graphs and Tables..... | 341 |
| 27.0 | Packaging Information..... | 343 |
| Appendix A: | Data Sheet Revision History | 349 |
| Appendix B: | Device Differences..... | 349 |
| Appendix C: | Device Migrations | 350 |
| Appendix D: | Migrating from other PICmicro Devices | 350 |
| Appendix E: | Development Tool Version Requirements | 351 |
| Index | | 353 |
| On-Line Support..... | | 361 |
| Reader Response | | 362 |
| PIC18CXX8 Product Identification System | | 363 |

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.

PIC18CXX8

NOTES:

1.0 DEVICE OVERVIEW

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

1. PIC18C658
2. PIC18C858

The PIC18C658 is available in 64-pin TQFP and 68-pin PLCC packages. The PIC18C858 is available in 80-pin TQFP and 84-pin PLCC packages.

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

The following two figures are device block diagrams sorted by pin count; 64/68-pin for Figure 1-1 and 80/84-pin for Figure 1-2. The 64/68-pin and 80/84-pin pinouts are listed in Table 1-2.

TABLE 1-1: DEVICE FEATURES

| Features | | | PIC18C658 | PIC18C858 |
|---------------------------------------|----------|-------------------------------|-------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| Operating Frequency | | | DC - 40 MHz | DC - 40 MHz |
| Program Memory | Internal | Bytes | 32 K | 32 K |
| | | # of Single word Instructions | 16384 | 16384 |
| Data Memory (Bytes) | | | 1536 | 1536 |
| Interrupt sources | | | 21 | 21 |
| I/O Ports | | | Ports A – G | Ports A – H, J, K |
| Timers | | | 4 | 4 |
| Capture/Compare/PWM modules | | | 2 | 2 |
| Serial Communications | | | MSSP, CAN Addressable USART | MSSP, CAN Addressable USART |
| Parallel Communications | | | PSP | PSP |
| 10-bit Analog-to-Digital Module | | | 12 input channels | 16 input channels |
| Analog Comparators | | | 2 | 2 |
| RESETS (and Delays) | | | POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST) | POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST) |
| Programmable Low Voltage Detect | | | Yes | Yes |
| Programmable Brown-out Reset | | | Yes | Yes |
| CAN Module | | | Yes | Yes |
| In-Circuit Serial Programming (ICSP™) | | | Yes | Yes |
| Instruction Set | | | 75 Instructions | 75 Instructions |
| Packages | | | 64-pin TQFP 68-pin CERQUAD (Windowed) 68-pin PLCC | 80-pin TQFP 84-pin CERQUAD (Windowed) 84-pin PLCC |

PIC18CXX8

FIGURE 1-1: PIC18C658 BLOCK DIAGRAM

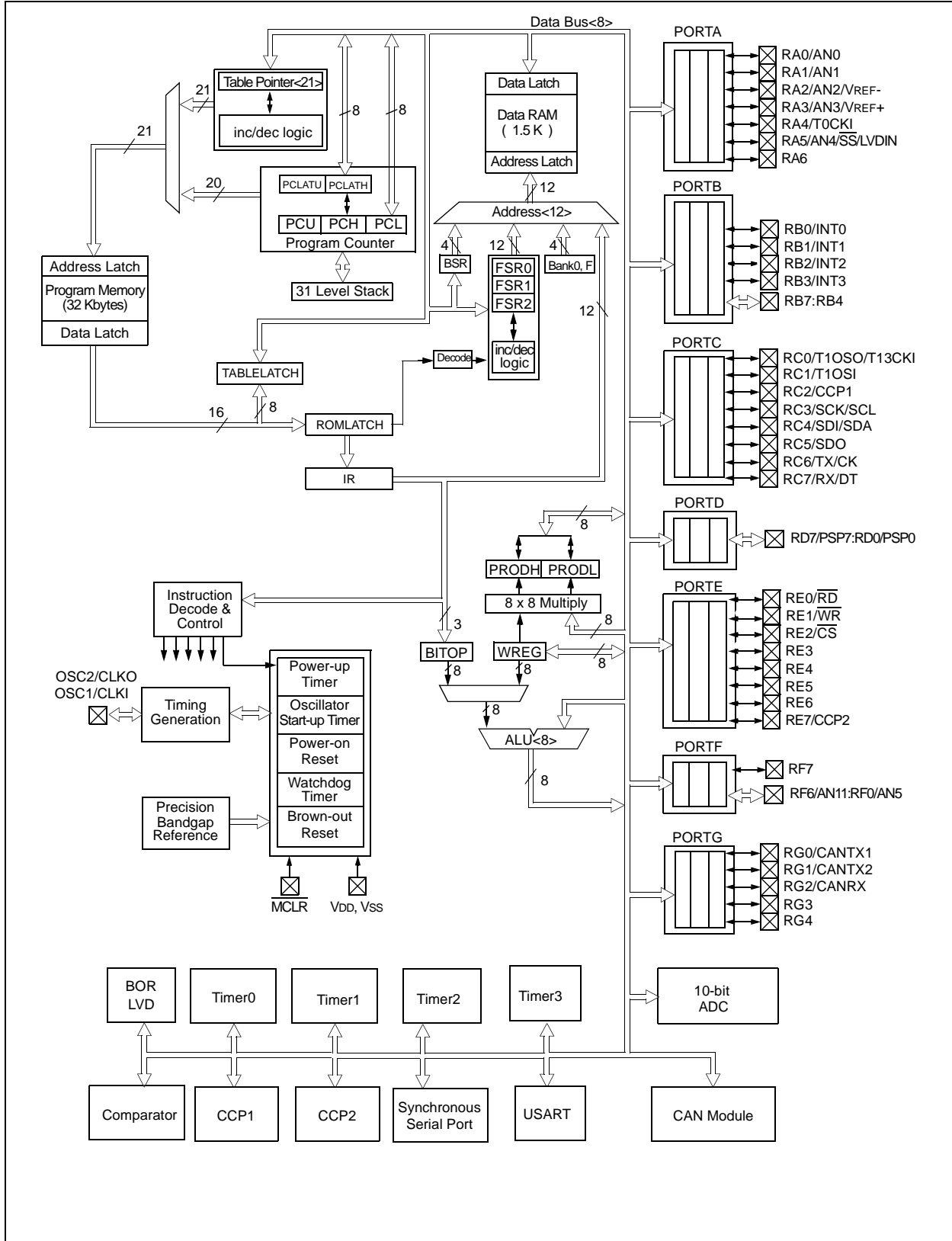
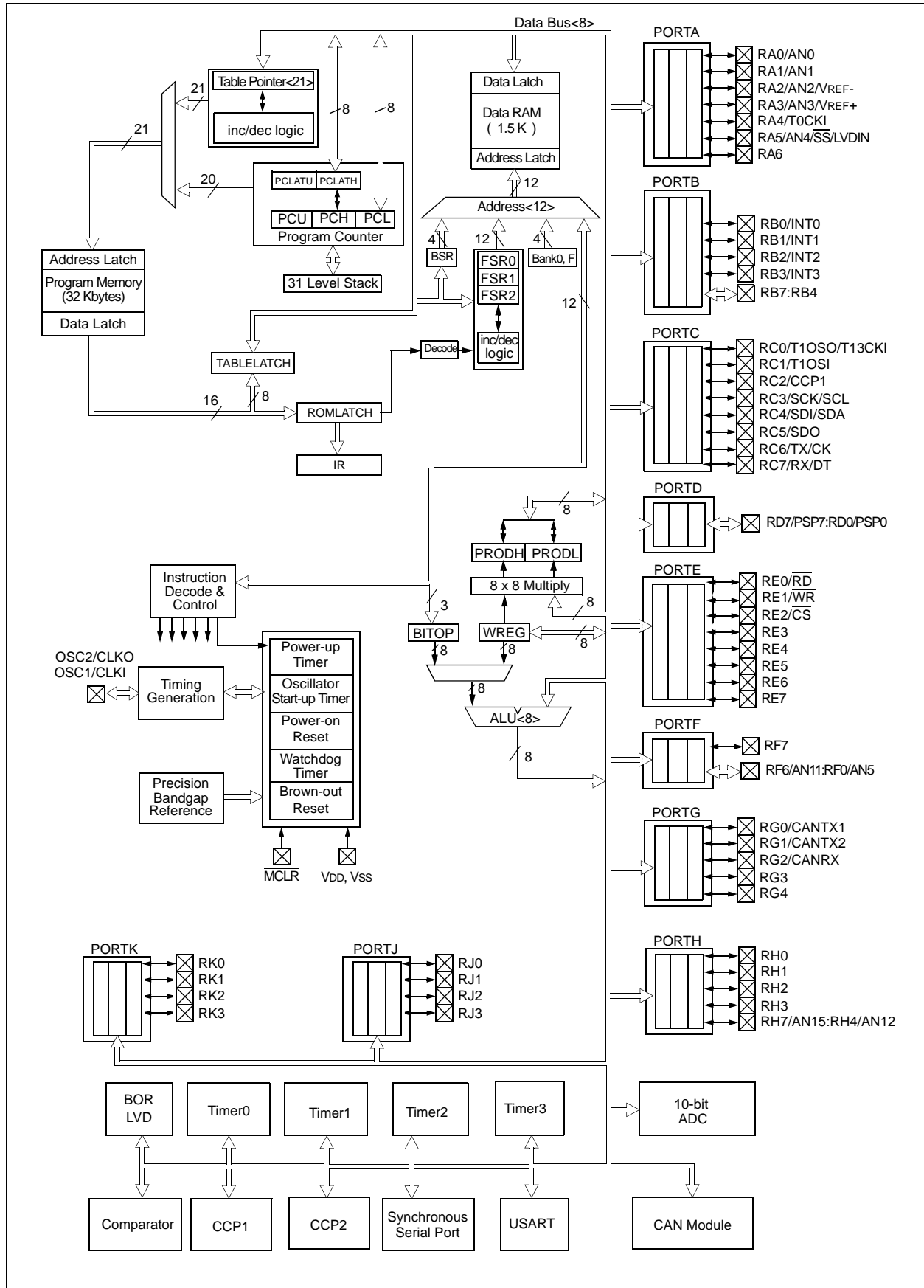


FIGURE 1-2: PIC18C858 BLOCK DIAGRAM



PIC18CXX8

TABLE 1-2: PINOUT I/O DESCRIPTIONS

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|----------------------------------------------|------------|------------------|-----------|------------------|-----------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | |
| MCLR/VPP MCLR VPP | 7 | 16 | 9 | 20 | I P | ST | Master clear (RESET) input. This pin is an active low RESET to the device. Programming voltage input |
| NC | — | 1, 18, 35, 52 | — | 1, 22, 43, 64 | — | — | These pins should be left unconnected |
| OSC1/CLKI OSC1 CLKI | 39 | 50 | 49 | 62 | I I | CMOS/ST CMOS | Oscillator crystal input or external clock source input. ST buffer when configured in RC mode. Otherwise CMOS. External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins). |
| OSC2/CLKO/RA6 OSC2 CLKO RA6 | 40 | 51 | 50 | 63 | O O I/O | — — TTL | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate General purpose I/O pin |

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 | |
|---------------------------------|------------|------|-----------|------|----------|-------------|-------------------------------------------------------|------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | | |
| RA0/AN0 | 24 | 34 | 30 | 42 | I/O | TTL | PORTA is a bi-directional I/O port Digital I/O | |
| RA0 AN0 | | | | | I | Analog | | Analog input 0 |
| RA1/AN1 | 23 | 33 | 29 | 41 | I/O | TTL | Digital I/O | |
| RA1 AN1 | | | | | I | Analog | | Analog input 1 |
| RA2/AN2/VREF- | 22 | 32 | 28 | 40 | I/O | TTL | Digital I/O | |
| RA2 AN2 | | | | | I | Analog | | Analog input 2 |
| VREF- | | | | | I | Analog | | A/D reference voltage (Low) input |
| RA3/AN3/VREF+ | 21 | 31 | 27 | 39 | I/O | TTL | Digital I/O | |
| RA3 AN3 | | | | | I | Analog | | Analog input 3 |
| VREF+ | | | | | I | Analog | | A/D reference voltage (High) input |
| RA4/T0CKI | 28 | 39 | 34 | 47 | I/O | ST/OD | Digital I/O – Open drain when configured as output | |
| RA4 T0CKI | | | | | I | ST | | Timer0 external clock input |
| RA5/AN4/ \overline{SS} /LVDIN | 27 | 38 | 33 | 46 | I/O | TTL | Digital I/O | |
| RA5 AN4 | | | | | I | Analog | | Analog input 4 |
| \overline{SS} | | | | | I | ST | | SPI slave select input |
| LVDIN | | | | | I | Analog | | Low voltage detect input |
| RA6 | | | | | | | | See the OSC2/CLKO/RA6 pin |

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)

PIC18CXX8

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

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|-------------------------|------------|------|-----------|------|------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | |
| RB0/INT0 RB0 INT0 | 48 | 60 | 58 | 72 | I/O I | TTL ST | PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. Digital I/O External interrupt 0 |
| RB1/INT1 RB1 INT1 | 47 | 59 | 57 | 71 | I/O I | TTL ST | Digital I/O External interrupt 1 |
| RB2/INT2 RB2 INT2 | 46 | 58 | 56 | 70 | I/O I | TTL ST | Digital I/O External interrupt 2 |
| RB3/INT3 RB3 INT3 | 45 | 57 | 55 | 69 | I/O I/O | TTL ST | Digital I/O External interrupt 3 |
| 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 |
| RB7 | 37 | 48 | 47 | 60 | I I/O | ST TTL | ICSP programming clock Digital I/O Interrupt-on-change pin ICSP programming 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 VDD)

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

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|--------------------------------------------|------------|------|-----------|------|-----------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | |
| RC0/T1OSO/T13CKI RC0 T1OSO T13CKI | 30 | 41 | 36 | 49 | I/O O I | ST — ST | PORTC is a bi-directional I/O port Digital I/O Timer1 oscillator output Timer1/Timer3 external clock input |
| RC1/T1OSI RC1 T1OSI | 29 | 40 | 35 | 48 | I/O I | ST CMOS | Digital I/O Timer1 oscillator input |
| RC2/CCP1 RC2 CCP1 | 33 | 44 | 43 | 56 | I/O I/O | ST ST | Digital I/O Capture1 input/Compare1 output/PWM1 output |
| RC3/SCK/SCL RC3 SCK SCL | 34 | 45 | 44 | 57 | I/O I/O I/O | ST ST ST | Digital I/O Synchronous serial clock input/output for SPI mode Synchronous serial clock input/output for I ² C mode |
| RC4/SDI/SDA RC4 SDI SDA | 35 | 46 | 45 | 58 | I/O I I/O | ST ST ST | Digital I/O SPI data in I ² C data I/O |
| RC5/SDO RC5 SDO | 36 | 47 | 46 | 59 | I/O O | ST — | Digital I/O SPI data out |
| RC6/TX/CK RC6 TX CK | 31 | 42 | 37 | 50 | I/O O I/O | ST — ST | Digital I/O USART asynchronous transmit USART synchronous clock (See RX/DT) |
| RC7/RX/DT RC7 RX DT | 32 | 43 | 38 | 51 | I/O I I/O | ST ST ST | Digital I/O USART asynchronous receive USART synchronous data (See TX/CK) |

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)

PIC18CXX8

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

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|-------------------------|------------|------|-----------|------|----------|-------------|-----------------------------------------------------------------------------------------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | |
| RD0/PSP0 | 58 | 3 | 72 | 3 | I/O | ST | PORTD is a bi-directional I/O port. These pins have TTL input buffers when external memory is enabled. Digital I/O |
| RD0/PSP0 RD0 PSP0 | | | | | I/O | TTL | |
| RD1/PSP1 | 55 | 67 | 69 | 83 | I/O | ST | Digital I/O |
| RD1/PSP1 RD1 PSP1 | | | | | I/O | TTL | |
| RD2/PSP2 | 54 | 66 | 68 | 82 | I/O | ST | Digital I/O |
| RD2/PSP2 RD2 PSP2 | | | | | I/O | TTL | |
| RD3/PSP3 | 53 | 65 | 67 | 81 | I/O | ST | Digital I/O |
| RD3/PSP3 RD3 PSP3 | | | | | I/O | TTL | |
| RD4/PSP4 | 52 | 64 | 66 | 80 | I/O | ST | Digital I/O |
| RD4/PSP4 RD4 PSP4 | | | | | I/O | TTL | |
| RD5/PSP5 | 51 | 63 | 65 | 79 | I/O | ST | Digital I/O |
| RD5/PSP5 RD5 PSP5 | | | | | I/O | TTL | |
| RD6/PSP6 | 50 | 62 | 64 | 78 | I/O | ST | Digital I/O |
| RD6/PSP6 RD6 PSP6 | | | | | I/O | TTL | |
| RD7/PSP7 | 49 | 61 | 63 | 77 | I/O | ST | Digital I/O |
| RD7/PSP7 RD7 PSP7 | | | | | I/O | TTL | |

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 |
|--------------------------------------------------------------|------------|------|-----------|------|------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | |
| RE0/ $\overline{\text{RD}}$ RE0 $\overline{\text{RD}}$ | 2 | 11 | 4 | 15 | I/O I | ST TTL | PORTE is a bi-directional I/O port Digital I/O Read control for parallel slave port (See $\overline{\text{WR}}$ and $\overline{\text{CS}}$ pins) |
| RE1/ $\overline{\text{WR}}$ RE1 $\overline{\text{WR}}$ | 1 | 10 | 3 | 14 | I/O I | ST TTL | Digital I/O Write control for parallel slave port (See $\overline{\text{CS}}$ and $\overline{\text{RD}}$ pins) |
| RE2/ $\overline{\text{CS}}$ RE2 $\overline{\text{CS}}$ | 64 | 9 | 78 | 9 | I/O I | ST TTL | Digital I/O Chip select control for parallel slave port (See $\overline{\text{RD}}$ and $\overline{\text{WR}}$) |
| RE3 | 63 | 8 | 77 | 8 | I/O | ST | Digital I/O |
| RE4 | 62 | 7 | 76 | 7 | I/O | ST | Digital I/O |
| RE5 | 61 | 6 | 75 | 6 | I/O | ST | Digital I/O |
| RE6 | 60 | 5 | 74 | 5 | I/O | ST | Digital I/O |
| RE7/CCP2 RE7 CCP2 | 59 | 4 | 73 | 4 | I/O I/O | ST ST | Digital I/O Capture2 input, Compare2 output, PWM2 output |

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)

PIC18CXX8

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

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|----------------|------------|------|-----------|------|----------|-------------|----------------------------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | |
| RF0/AN5 | 18 | 28 | 24 | 36 | I/O | ST | PORTF is a bi-directional I/O port Digital I/O |
| RF0 AN5 | | | | | I | Analog | |
| RF1/AN6/C2OUT | 17 | 27 | 23 | 35 | I/O | ST | Digital I/O Analog input 6 Comparator 2 output |
| RF1 AN6 | | | | | I | Analog | |
| C2OUT | | | | | O | ST | |
| RF2/AN7/C1OUT | 16 | 26 | 18 | 30 | I/O | ST | Digital I/O Analog input 7 Comparator 1 output |
| RF2 AN7 | | | | | I | Analog | |
| C1OUT | | | | | O | ST | |
| RF3/AN8 | 15 | 25 | 17 | 29 | I/O | ST | Digital I/O Analog input 8 |
| RF3 AN8 | | | | | I | Analog | |
| RF4/AN9 | 14 | 24 | 16 | 28 | I/O | ST | Digital I/O Analog input 9 |
| RF4 AN9 | | | | | I | Analog | |
| RF5/AN10/CVREF | 13 | 23 | 15 | 27 | I/O | ST | Digital I/O Analog input 10 Comparator VREF output |
| RF5 AN10 | | | | | I | Analog | |
| CVREF | | | | | O | Analog | |
| RF6/AN11 | 12 | 22 | 14 | 26 | I/O | ST | Digital I/O Analog input 11 |
| RF6 AN11 | | | | | I | Analog | |
| RF7 | 11 | 21 | 13 | 25 | I/O | ST | Digital I/O |

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 |
|-----------------------------|------------|------|-----------|------|----------|---------------|--------------------------------------------------------------------------|
| | PIC18C658 | | PIC18C858 | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | |
| RG0/CANTX1 RG0 CANTX1 | 3 | 12 | 5 | 16 | I/O O | ST CAN Bus | PORTG is a bi-directional I/O port Digital I/O CAN bus output |
| RG1/CANTX2 RG1 CANTX2 | 4 | 13 | 6 | 17 | I/O O | ST CAN Bus | Digital I/O Complimentary CAN bus output or CAN bus bit time clock |
| RG2/CANRX RG2 CANRX | 5 | 14 | 7 | 18 | I/O I | ST CAN Bus | Digital I/O CAN bus input |
| RG3 | 6 | 15 | 8 | 19 | I/O | ST | Digital I/O |
| RG4 | 8 | 17 | 10 | 21 | I/O | ST | Digital I/O |
| RH0 | — | — | 79 | 10 | I/O | ST | PORTH is a bi-directional I/O port. Digital I/O |
| RH1 | — | — | 80 | 11 | I/O | ST | Digital I/O |
| RH2 | — | — | 1 | 12 | I/O | ST | Digital I/O |
| RH3 | — | — | 2 | 13 | I/O | ST | Digital I/O |
| RH4/AN12 RH4 AN12 | — | — | 22 | 34 | I/O I | ST Analog | Digital I/O Analog input 12 |
| RH5/AN13 RH5 AN13 | — | — | 21 | 33 | I/O I | ST Analog | Digital I/O Analog input 13 |
| RH6/AN14 RH6 AN14 | — | — | 20 | 32 | I/O I | ST Analog | Digital I/O Analog input 14 |
| RH7/AN15 RH7 AN15 | — | — | 19 | 31 | I/O I | ST Analog | Digital I/O Analog input 15 |

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)

PIC18CXX8

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

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description | | |
|-------------------|-------------------|-------------------|-------------------|-------------------|----------|-------------|-------------------------------------------------------|----|-------------|
| | PIC18C658 | | PIC18C858 | | | | | | |
| | TQFP | PLCC | TQFP | PLCC | | | | | |
| RJ0 RJ0 RJ0 | — | — | 62 | 76 | I/O | ST | PORTJ is a bi-directional I/O port Digital I/O | | |
| RJ1 RJ1 RJ1 | — | — | 61 | 75 | | | I/O | ST | Digital I/O |
| RJ2 RJ2 RJ2 | — | — | 60 | 74 | | | | | I/O |
| RJ3 RJ3 RJ3 | — | — | 59 | 73 | I/O | ST | | | |
| RK0 | — | — | 39 | 52 | | | I/O | ST | |
| RK1 | — | — | 40 | 53 | | | I/O | ST | Digital I/O |
| RK2 | — | — | 41 | 54 | I/O | ST | Digital I/O | | |
| RK3 | — | — | 42 | 55 | I/O | ST | Digital I/O | | |
| 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

The PIC18CXX8 can be operated in one of eight oscillator modes, programmable by three configuration bits (FOSC2, FOSC1, and FOSC0).

1. LP Low Power Crystal
2. XT Crystal/Resonator
3. HS High Speed Crystal/Resonator
4. HS4 High Speed Crystal/Resonator with PLL enabled
5. RC External Resistor/Capacitor
6. RCIO External Resistor/Capacitor with I/O pin enabled
7. EC External Clock
8. ECIO External Clock with I/O pin enabled

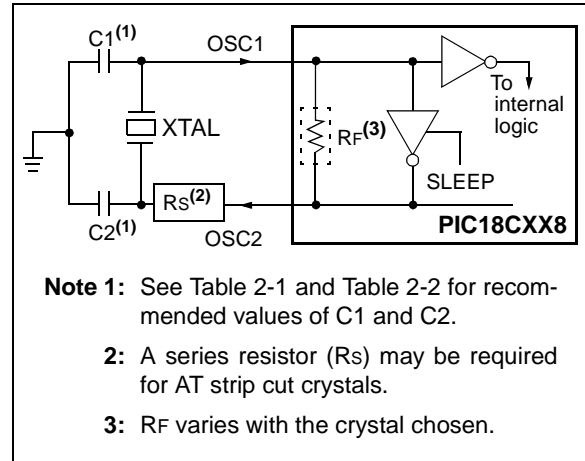
2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HS4 (PLL) 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.

The PIC18CXX8 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, XT OR LP OSC CONFIGURATION)



PIC18CXX8

TABLE 2-1: CERAMIC RESONATORS

| Ranges Tested: | | | |
|--------------------------------------------------------------------|------------------------|-------------|-------------|
| Mode | Freq | OSC1 | OSC2 |
| XT | 455 kHz | 68 - 100 pF | 68 - 100 pF |
| | 2.0 MHz | 15 - 68 pF | 15 - 68 pF |
| | 4.0 MHz | 15 - 68 pF | 15 - 68 pF |
| 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 |
| | 8.0 MHz | 10 - 68 pF | 10 - 68 pF |
| | 10.0 MHz | TBD | TBD |
| These values are for design guidance only. See notes on this page. | | | |
| Resonators Used: | | | |
| 455 kHz | Panasonic EFO-A455K04B | | ± 0.3% |
| 2.0 MHz | Murata Erie CSA2.00MG | | ± 0.5% |
| 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. | | | |

- 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, as well as XT mode, to avoid overdriving crystals with low drive level specification.

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

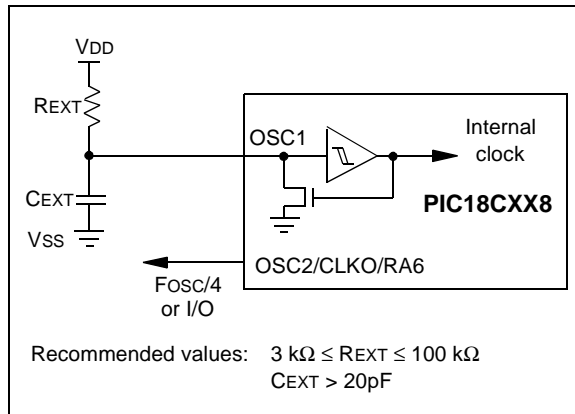
| Osc Type | Crystal Freq | Cap. Range C1 | Cap. Range C2 |
|--------------------------------------------------------------------|------------------------|---------------|---------------|
| LP | 32.0 kHz | 33 pF | 33 pF |
| | 200 kHz | 15 pF | 15 pF |
| XT | 200 kHz | 47-68 pF | 47-68 pF |
| | 1.0 MHz | 15 pF | 15 pF |
| | 4.0 MHz | 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 |
| | 8.0 MHz | 15-33 pF | 15-33 pF |
| | 10.0 MHz | TBD | TBD |
| 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 |

2.3 RC Oscillator

For timing insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (R_{EXT}) and capacitor (C_{EXT}) 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 C_{EXT} 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 R/C 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



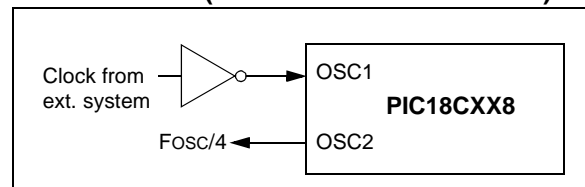
The RCIO oscillator mode functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

2.4 External Clock Input

The EC and ECIO oscillator modes require 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.

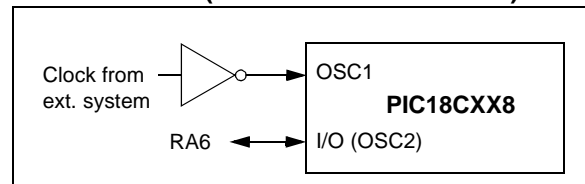
In the EC oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC oscillator mode.

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



The ECIO oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6). Figure 2-4 shows the pin connections for the ECIO oscillator mode.

FIGURE 2-4: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)



PIC18CXX8

2.5 HS4 (PLL)

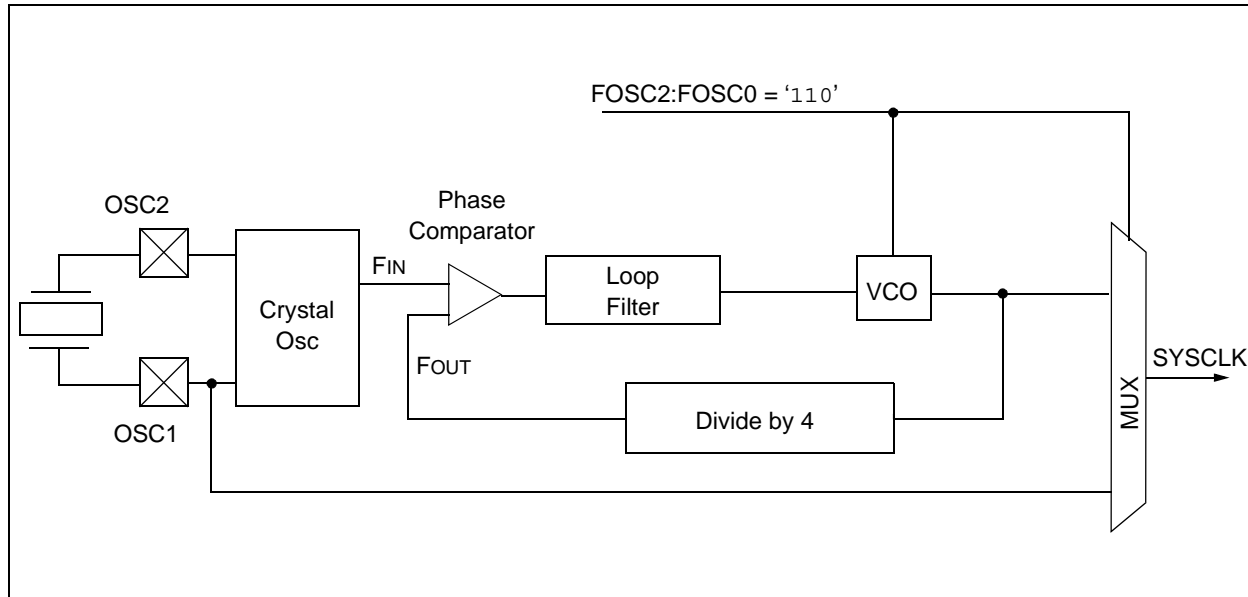
A Phase Locked Loop circuit is provided as a programmable option for users that want to multiply the frequency of the incoming crystal oscillator signal by 4. For an input clock frequency of 10 MHz, the internal clock frequency will be multiplied to 40 MHz. This is useful for customers who are concerned with EMI due to high frequency crystals.

The PLL can only be enabled when the oscillator configuration bits are programmed for HS mode. If they are programmed for any other mode, the PLL is not enabled and the system clock will come directly from OSC1.

The PLL is one of the modes of the FOSC2:FOSC0 configuration bits. The oscillator mode is specified during device programming.

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 2-5: PLL BLOCK DIAGRAM



2.6 Oscillator Switching Feature

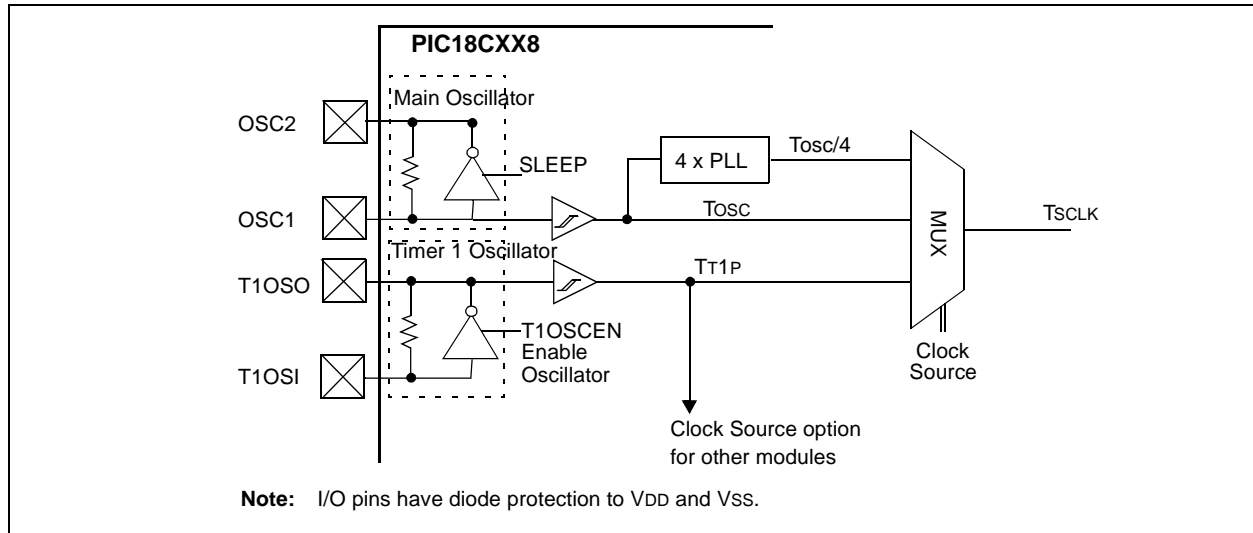
The PIC18CXX8 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low frequency clock source. For the PIC18CXX8 devices, this alternate clock source is the Timer1 oscillator. If a low frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a low power execution mode. Figure 2-6 shows a block diagram of the system clock sources. The clock switching feature is enabled by programming the Oscillator Switching Enable ($\overline{\text{OSCSEN}}$) bit in Configuration register CONFIG1H to a '0'. Clock switching is disabled in an erased device. See Section 9 for further details of the Timer1 oscillator. See Section 22.0 for Configuration Register details.

2.6.1 SYSTEM CLOCK SWITCH BIT

The system clock source switching is performed under software control. The system clock switch bit, SCS (OSCCON register), controls the clock switching. When the SCS bit is '0', the system clock source comes from the main oscillator selected by the FOSC2:FOSC0 configuration bits. When the SCS bit is set, the system clock source will come from the Timer1 oscillator. The SCS 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 SCS bit will be ignored (SCS bit forced cleared) and the main oscillator will continue to be the system clock source.

FIGURE 2-6: DEVICE CLOCK SOURCES



REGISTER 2-1: OSCCON REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|-----|-------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-1 |
| — | — | — | — | — | — | — | SCS |
| bit 7 | | | | | | | bit 0 |

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SCS:** System Clock Switch bit
when $\overline{\text{OSCSEN}}$ configuration bit = '0' and T1OSCEN bit is set:

- 1 = Switch to Timer1 Oscillator/Clock pin
- 0 = Use primary Oscillator/Clock input pin

when $\overline{\text{OSCSEN}}$ is clear or T1OSCEN is clear:
 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 |

PIC18CXX8

2.6.2 OSCILLATOR TRANSITIONS

The PIC18CXX8 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 2-7. The Timer1 oscillator is assumed to be running all the time. After the SCS 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, XT, LP), the transition will take place after an oscillator start-up time (T_{OST}) has occurred. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS, XT and LP modes is shown in Figure 2-8.

FIGURE 2-7: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR

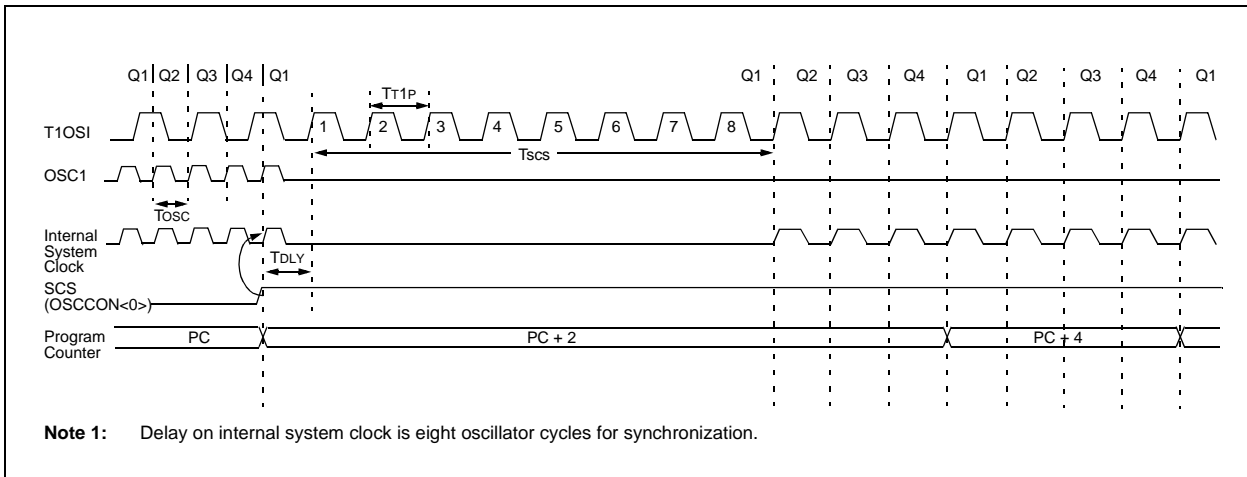
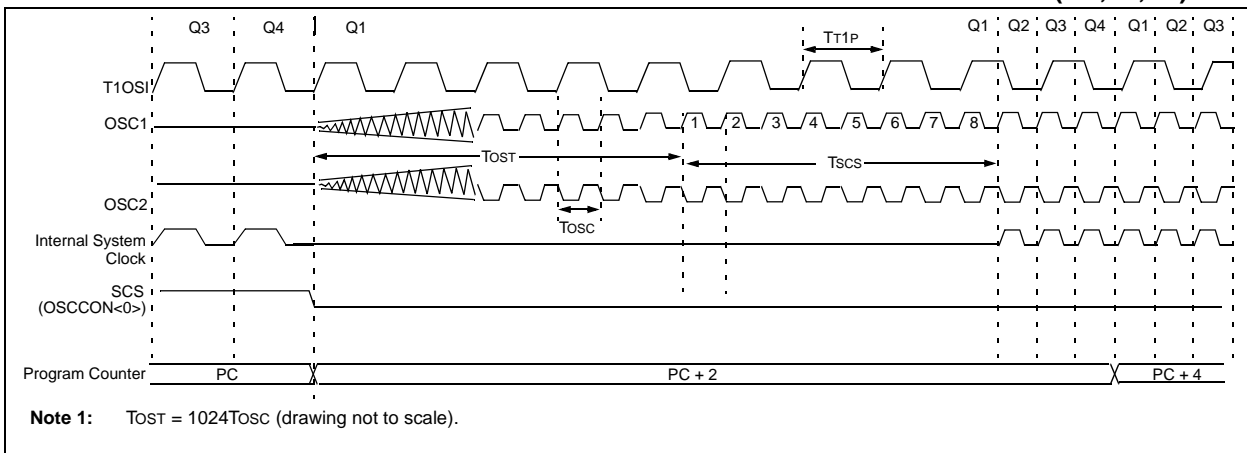


FIGURE 2-8: TIMING DIAGRAM FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS,XT,LP)



If the main oscillator is configured for HS4 (PLL) mode, an oscillator start-up time (T_{OST}) plus an additional PLL time-out (T_{PLL}) 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 2-9.

If the main oscillator is configured in the RC, RCIO, EC or ECIO 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, RCIO, EC and ECIO modes is shown in Figure 2-10.

FIGURE 2-9: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS WITH PLL)

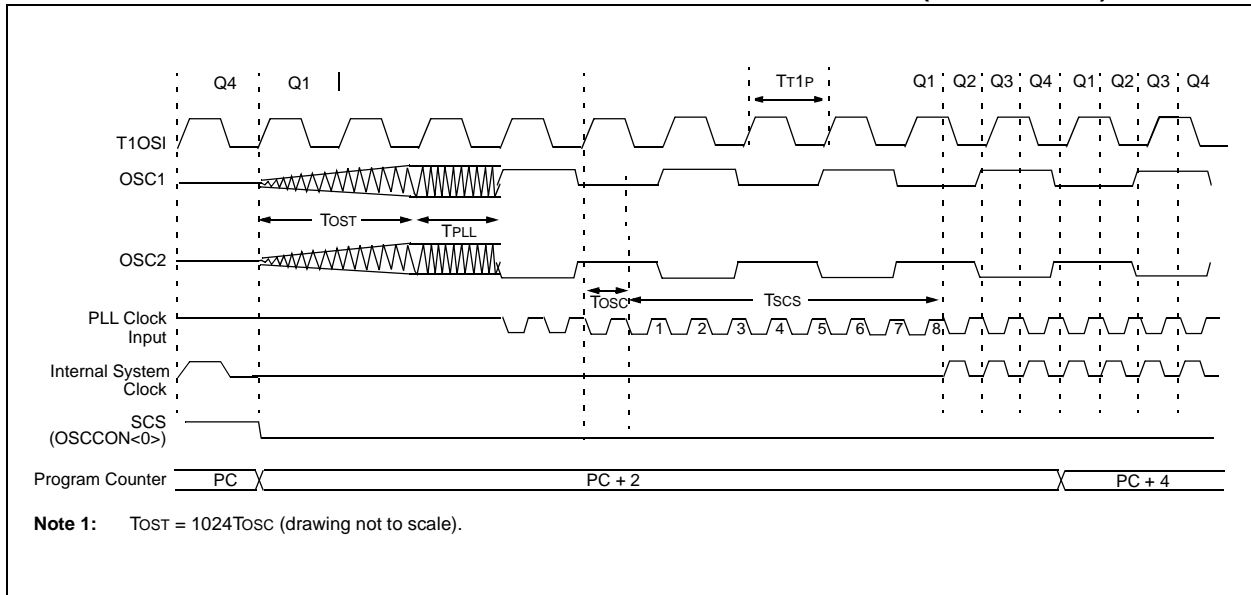
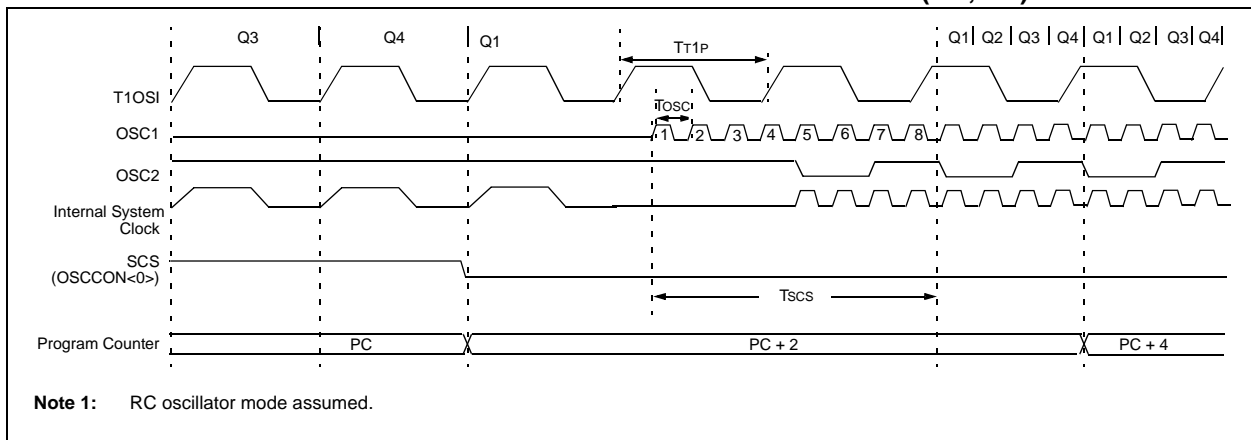


FIGURE 2-10: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (RC, EC)



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 (`POR` and `BOR`). The second timer is the Oscillator Start-up Timer (`OST`), intended to keep the chip in `RESET` until the crystal oscillator is stable.

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. This time is called `TPLL` (parameter #7) to allow the `PLL` ample time to lock to the incoming clock frequency.

TABLE 2-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE

| OSC Mode | OSC1 Pin | OSC2 Pin |
|----------------|--------------------------------------------------------|--------------------------------------------------------|
| RC | Floating, external resistor should pull high | At logic low |
| RCIO | Floating, external resistor should pull high | Configured as <code>PORTA</code> , bit 6 |
| ECIO | Floating | Configured as <code>PORTA</code> , bit 6 |
| EC | Floating | At logic low |
| LP, XT, 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

The PIC18CXX8 differentiates between various kinds of RESET:

- Power-on Reset (POR)
- $\overline{\text{MCLR}}$ Reset during normal operation
- $\overline{\text{MCLR}}$ Reset during SLEEP
- Watchdog Timer (WDT) Reset (during normal operation)
- Programmable Brown-out Reset (PBOR)
- RESET Instruction
- Stack Full Reset
- 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, Brown-out Reset, $\overline{\text{MCLR}}$ Reset during SLEEP and by the RESET instruction.

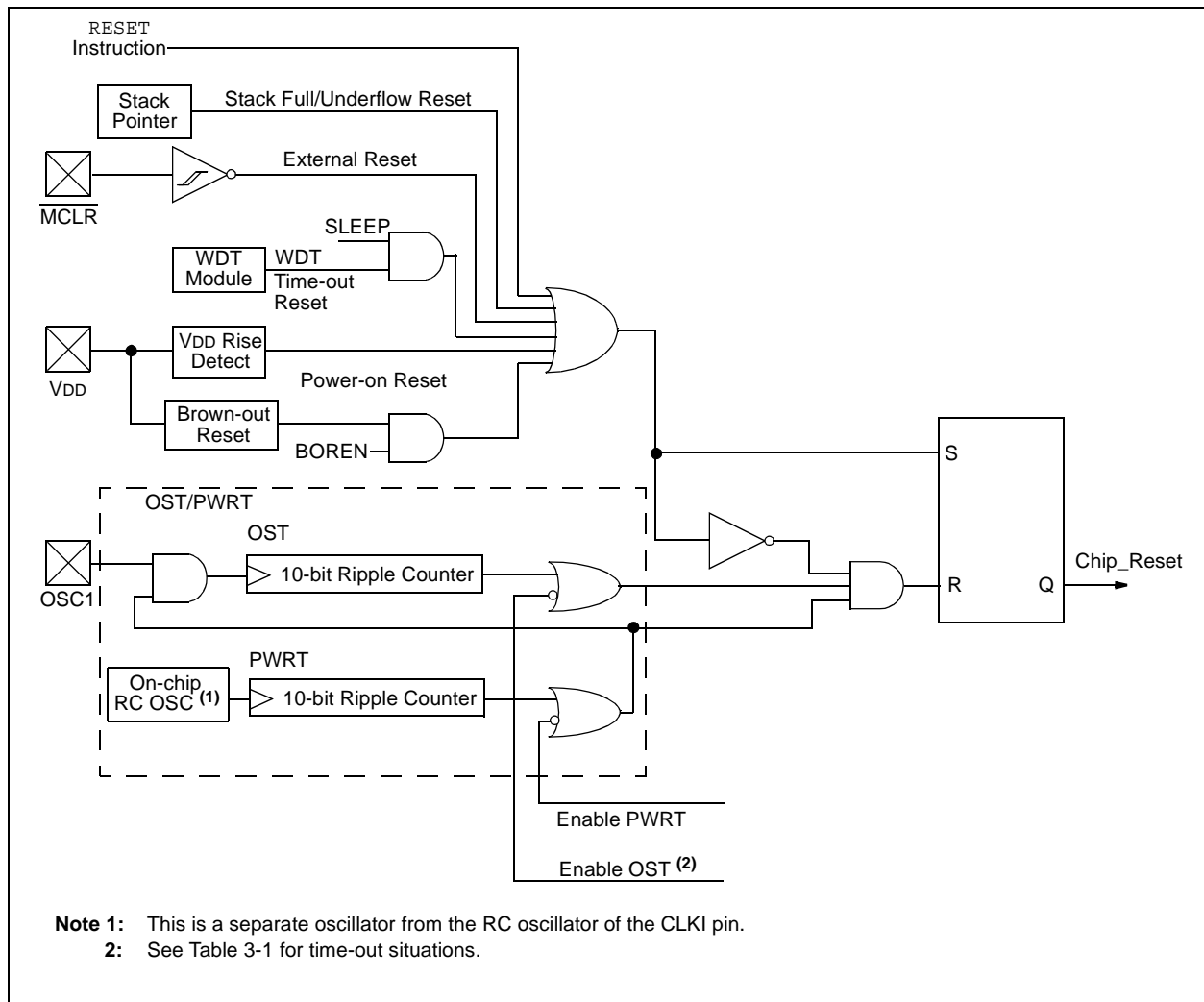
Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register, RI, TO, PD, POR and BOR are set or cleared differently in different RESET situations, as indicated in Table 3-2. These bits are used in software to determine 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.

The Enhanced MCU devices have 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 ON-CHIP RESET CIRCUIT

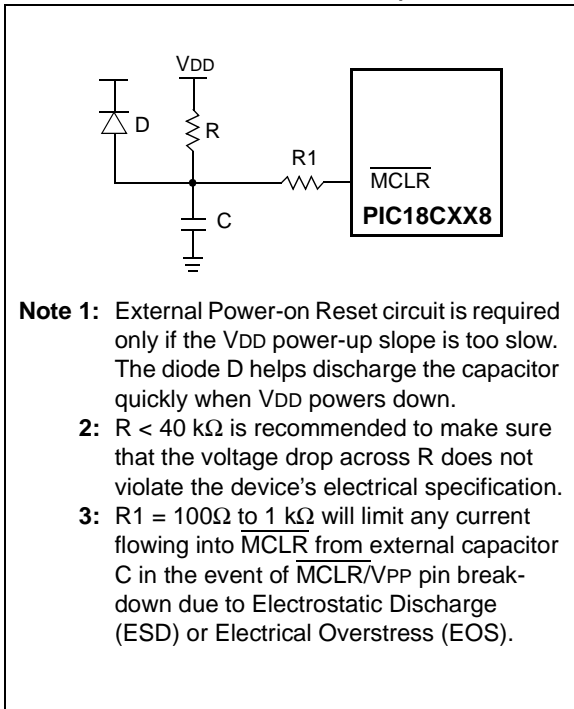


3.1 Power-on Reset (POR)

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,...) 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. Brown-out Reset may be used to meet the voltage start-up condition.

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



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's time delay allows VDD to rise to an acceptable level. A configuration bit (PWRTEN in CONFIG2L register) is provided to enable/disable the PWRT.

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 XT, 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 2 ms and follows the oscillator start-up time-out (OST).

3.5 Brown-out Reset (BOR)

A configuration bit, BOREN, can disable (if clear/programmed) or enable (if set) the Brown-out Reset circuitry. If VDD falls below parameter D005 for greater than parameter #35, the brown-out situation resets the chip. A RESET may not occur if VDD falls below parameter D005 for less than parameter #35. The chip will remain in Brown-out Reset until VDD rises above BVDD. The Power-up Timer will then be invoked and will keep the chip in RESET an additional time delay (parameter #33). If VDD drops below BVDD while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above BVDD, the Power-up Timer will execute the additional time delay.

3.6 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 $\overline{\text{MCLR}}$ is kept low long enough, the time-outs will expire. Bringing $\overline{\text{MCLR}}$ high will begin execution immediately (Figure 3-5). This is useful for testing purposes or to synchronize more than one PIC18CXX8 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⁾ | | Brown-out ⁽²⁾ | Wake-up from SLEEP or Oscillator Switch |
|------------------------------------|--------------------------------|--------------------------------|--------------------------|-----------------------------------------|
| | $\overline{\text{PWRTEN}} = 0$ | $\overline{\text{PWRTEN}} = 1$ | | |
| HS with PLL enabled ⁽¹⁾ | 72 ms + 1024Tosc + 2 ms | 1024Tosc + 2 ms | 72 ms + 1024Tosc + 2 ms | 1024Tosc + 2 ms |
| HS, XT, LP | 72 ms + 1024Tosc | 1024Tosc | 72 ms + 1024Tosc | 1024Tosc |
| EC | 72 ms | — | 72 ms | — |
| External RC | 72 ms | — | 72 ms | — |

Note 1: 2 ms = Nominal time required for the 4X PLL to lock.

2: 72 ms is the nominal power-up timer delay.

REGISTER 3-1: RCON REGISTER BITS AND POSITIONS

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

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

| Condition | Program Counter | RCON Register | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | STKFUL | STKUNF |
|--------------------------------------------------------|-----------------------|---------------|------------------------|------------------------|------------------------|-------------------------|-------------------------|--------|--------|
| Power-on Reset | 0000h | 00-1 1100 | 1 | 1 | 1 | 0 | 0 | u | u |
| $\overline{\text{MCLR}}$ Reset during normal operation | 0000h | 00-u uuuu | u | u | u | u | u | u | u |
| Software Reset during normal operation | 0000h | 0u-0 uuuu | 0 | u | u | u | u | u | u |
| Stack Full Reset during normal operation | 0000h | 0u-u uu11 | u | u | u | 1 | 1 | u | 1 |
| Stack Underflow Reset during normal operation | 0000h | 0u-u uu11 | u | u | u | 1 | 1 | 1 | u |
| $\overline{\text{MCLR}}$ Reset during SLEEP | 0000h | 00-u 10uu | u | 1 | 0 | u | u | u | u |
| WDT Reset | 0000h | 0u-u 01uu | u | 0 | 1 | u | u | u | u |
| WDT Wake-up | PC + 2 | uu-u 00uu | u | 0 | 0 | u | u | u | u |
| Brown-out Reset | 0000h | 0u-1 11u0 | 1 | 1 | 1 | u | 0 | u | u |
| Interrupt wake-up from SLEEP | PC + 2 ⁽¹⁾ | uu-u 00uu | u | 0 | 0 | u | u | u | u |

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '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 (0x000008h or 0x000018h).

PIC18CXX8

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

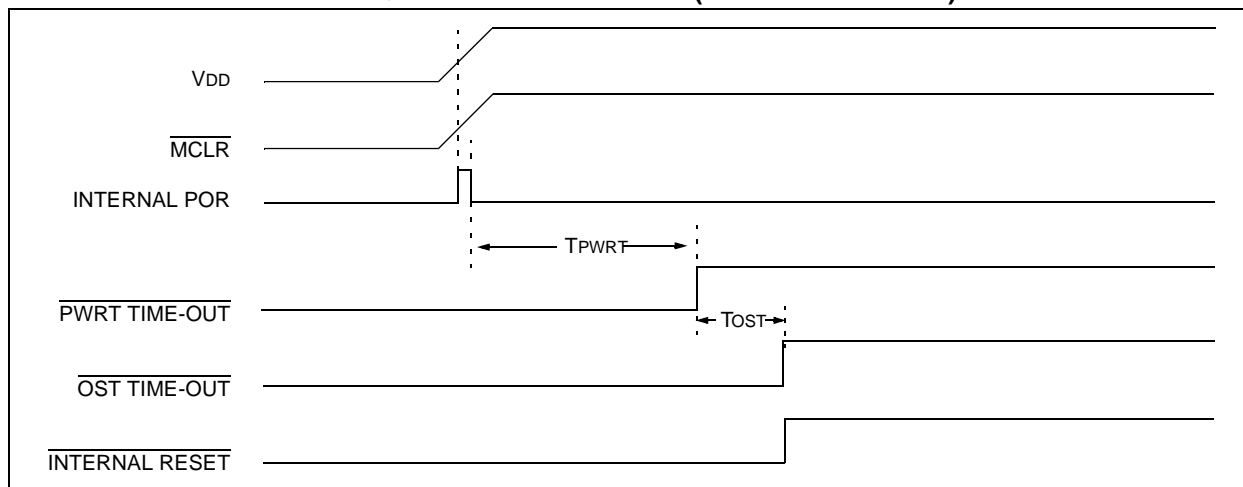


FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 1

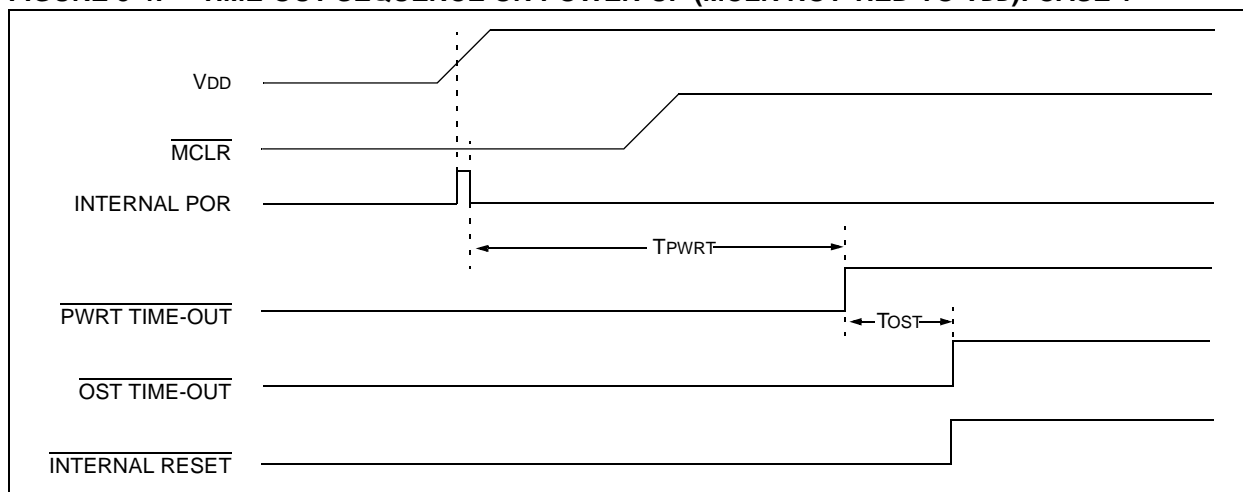


FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 2

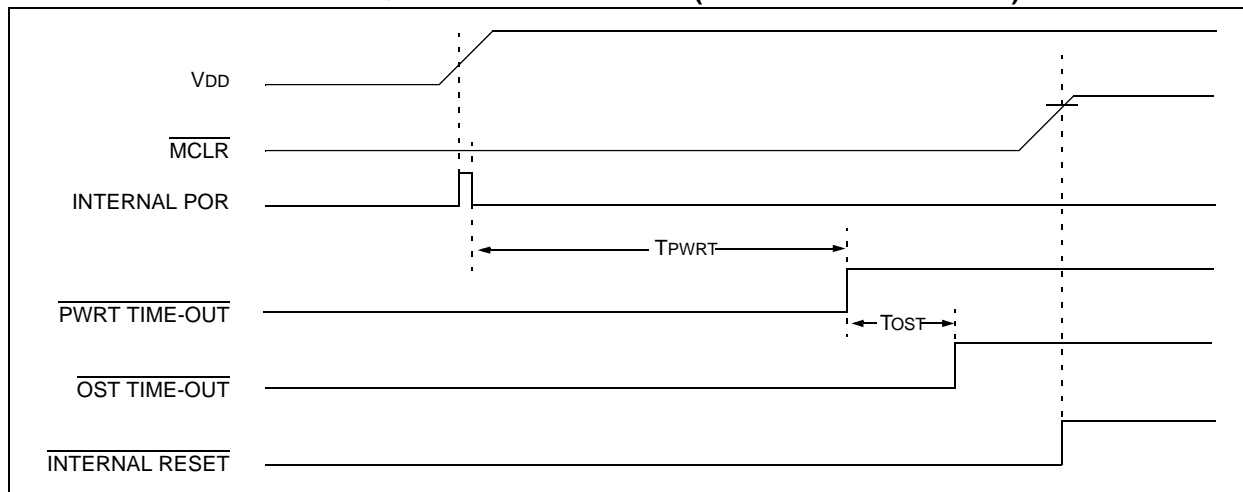


FIGURE 3-6: SLOW RISE TIME ($\overline{\text{MCLR}}$ TIED TO V_{DD})

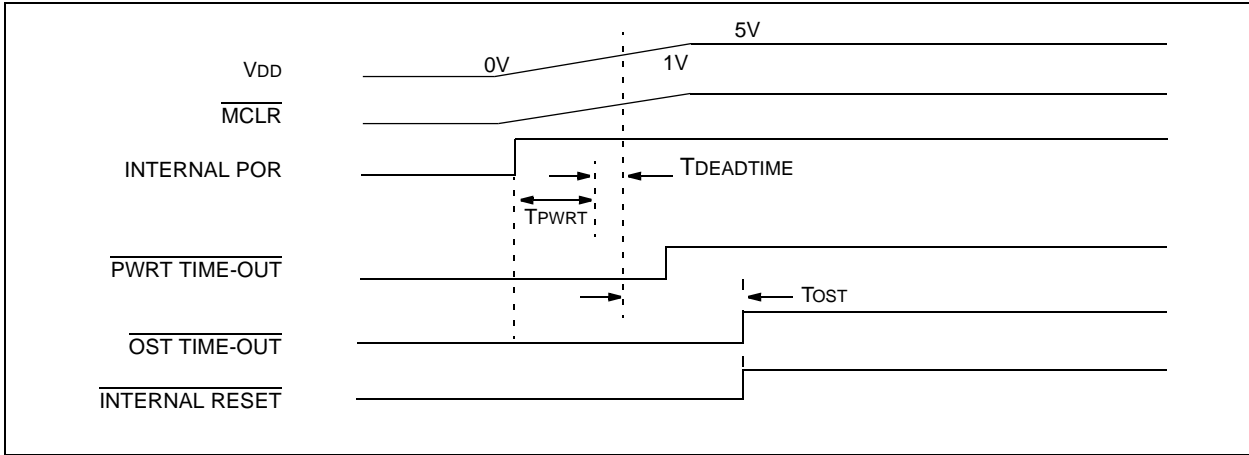
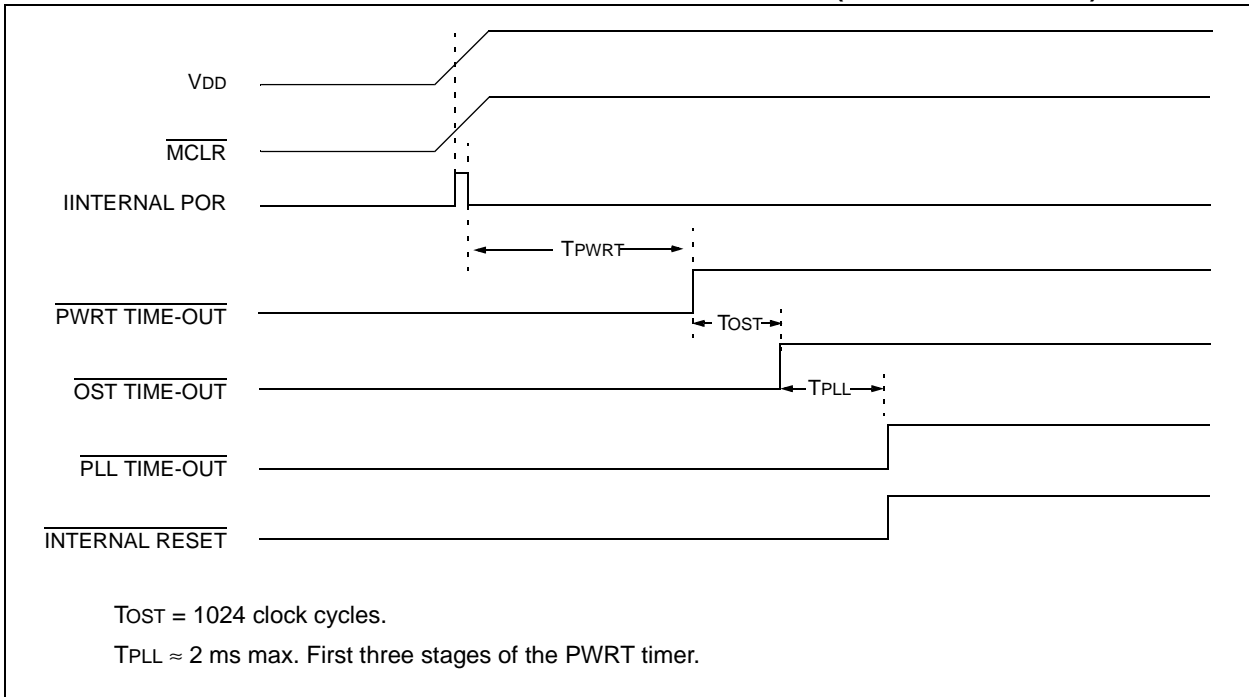


FIGURE 3-7: TIME-OUT SEQUENCE ON POR W/ PLL ENABLED ($\overline{\text{MCLR}}$ TIED TO V_{DD})



PIC18CXX8

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|----------|--------------------|-----|---------------------------------|--------------------------------------------------------------|------------------------------|
| | 658 | 858 | | | |
| TOSU | 658 | 858 | ---0 0000 | ---0 0000 | ---0 uuuu ⁽³⁾ |
| TOSH | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu ⁽³⁾ |
| TOSL | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu ⁽³⁾ |
| STKPTR | 658 | 858 | 00-0 0000 | 00-0 0000 | uu-u uuuu ⁽³⁾ |
| PCLATU | 658 | 858 | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | 658 | 858 | 0000 0000 | 0000 0000 | PC + 2 ⁽²⁾ |
| TBLPTRU | 658 | 858 | --00 0000 | --00 0000 | --uu uuuu |
| TBLPTRH | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRL | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TABLAT | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PRODH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INTCON | 658 | 858 | 0000 000x | 0000 000u | uuuu uuuu ⁽¹⁾ |
| INTCON2 | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu ⁽¹⁾ |
| INTCON3 | 658 | 858 | 1100 0000 | 1100 0000 | uuuu uuuu ⁽¹⁾ |
| INDF0 | 658 | 858 | N/A | N/A | N/A |
| POSTINC0 | 658 | 858 | N/A | N/A | N/A |
| POSTDEC0 | 658 | 858 | N/A | N/A | N/A |
| PREINC0 | 658 | 858 | N/A | N/A | N/A |
| PLUSW0 | 658 | 858 | N/A | N/A | N/A |
| FSR0H | 658 | 858 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR0L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF1 | 658 | 858 | N/A | N/A | N/A |
| POSTINC1 | 658 | 858 | N/A | N/A | N/A |
| POSTDEC1 | 658 | 858 | N/A | N/A | N/A |
| PREINC1 | 658 | 858 | N/A | N/A | N/A |
| PLUSW1 | 658 | 858 | N/A | N/A | N/A |

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

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 (0008h or 0018h).

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 STKPTR is modified to point to the next location in the hardware stack.

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

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: The long write enable is only reset on a POR or MCLR.

7: Available on PIC18C858 only.

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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|------------------------|--------------------|-----|---------------------------------|--------------------------------------------------------------|------------------------------|
| FSR1H | 658 | 858 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR1L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| BSR | 658 | 858 | ---- 0000 | ---- 0000 | ---- uuuu |
| INDF2 | 658 | 858 | N/A | N/A | N/A |
| POSTINC2 | 658 | 858 | N/A | N/A | N/A |
| POSTDEC2 | 658 | 858 | N/A | N/A | N/A |
| PREINC2 | 658 | 858 | N/A | N/A | N/A |
| PLUSW2 | 658 | 858 | N/A | N/A | N/A |
| FSR2H | 658 | 858 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR2L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| STATUS | 658 | 858 | ---x xxxx | ---u uuuu | ---u uuuu |
| TMR0H | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR0L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T0CON | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| OSCCON | 658 | 858 | ---- --0 | ---- --0 | ---- --u |
| LVDCON | 658 | 858 | --00 0101 | --00 0101 | --uu uuuu |
| WDTCON | 658 | 858 | ---- --0 | ---- --0 | ---- --u |
| RCON ^(4, 6) | 658 | 858 | 00-1 11q0 | 00-1 qquu | uu-u qquu |
| TMR1H | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR1L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T1CON | 658 | 858 | 0-00 0000 | u-uu uuuu | u-uu uuuu |
| TMR2 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PR2 | 658 | 858 | 1111 1111 | 1111 1111 | 1111 1111 |
| T2CON | 658 | 858 | -000 0000 | -000 0000 | -uuu uuuu |
| SSPBUF | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SSPADD | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPSTAT | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON1 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON2 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |

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

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 (0008h or 0018h).

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 STKPTR is modified to point to the next location in the hardware stack.

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

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: The long write enable is only reset on a POR or MCLR.

7: Available on PIC18C858 only.

PIC18CXX8

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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|----------|--------------------|-----|---------------------------------|--------------------------------------------------------------|------------------------------|
| ADRESH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADCON0 | 658 | 858 | --00 0000 | --00 0000 | --uu uuuu |
| ADCON1 | 658 | 858 | --00 0000 | --00 0000 | --uu uuuu |
| ADCON2 | 658 | 858 | 0--- -000 | 0--- -000 | u--- -uuu |
| CCPR1H | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR1L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP1CON | 658 | 858 | --00 0000 | --00 0000 | --uu uuuu |
| CCPR2H | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR2L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP2CON | 658 | 858 | --00 0000 | --00 0000 | --uu uuuu |
| CVRCON | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CMCON | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR3H | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T3CON | 658 | 858 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| PSPCON | 658 | 858 | 0000 ---- | 0000 ---- | uuuu ---- |
| SPBRG | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RCREG | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXREG | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXSTA | 658 | 858 | 0000 -01x | 0000 -01u | uuuu -uuu |
| RCSTA | 658 | 858 | 0000 000x | 0000 000u | uuuu uuuu |
| IPR3 | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR3 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE3 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IPR2 | 658 | 858 | -1-- 1111 | -1-- 1111 | -u-- uuuu |
| PIR2 | 658 | 858 | -0-- 0000 | -0-- 0000 | -u-- uuuu ⁽¹⁾ |
| PIE2 | 658 | 858 | -0-- 0000 | -0-- 0000 | -u-- uuuu |
| IPR1 | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| | 658 | 858 | -111 1111 | -111 1111 | -uuu uuuu |
| PIR1 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu ⁽¹⁾ |
| | 658 | 858 | -000 0000 | -000 0000 | -uuu uuuu ⁽¹⁾ |
| PIE1 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| | 658 | 858 | -000 0000 | -000 0000 | -uuu uuuu |

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

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 (0008h or 0018h).
- 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 STKPTR is modified to point to the next location in the hardware stack.
- 4:** See Table 3-2 for RESET value for specific condition.
- 5:** Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
- 6:** The long write enable is only reset on a POR or MCLR.
- 7:** Available on PIC18C858 only.

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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Reset | Wake-up via WDT or Interrupt |
|----------------------|--------------------|-----|---------------------------------|------------------------------------------------|------------------------------|
| | | | | WDT Reset RESET Instruction Stack Resets | |
| TRISJ ⁽⁷⁾ | - | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISH ⁽⁷⁾ | - | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISG | 658 | 858 | ---1 1111 | ---1 1111 | ---u uuuu |
| TRISF | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISE | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISD | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISC | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISB | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISA ⁽⁵⁾ | 658 | 858 | -111 1111 ⁽⁵⁾ | -111 1111 ⁽⁵⁾ | -uuu uuuu ⁽⁵⁾ |
| LATJ ⁽⁷⁾ | - | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATH ⁽⁷⁾ | - | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATG | 658 | 858 | ---x xxxx | ---u uuuu | ---u uuuu |
| LATF | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATE | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATD | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATC | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATB | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATA ⁽⁵⁾ | 658 | 858 | -xxx xxxx ⁽⁵⁾ | -uuu uuuu ⁽⁵⁾ | -uuu uuuu ⁽⁵⁾ |
| PORTJ ⁽⁷⁾ | - | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTH ⁽⁷⁾ | - | 858 | 0000 xxxx | 0000 uuuu | uuuu uuuu |
| PORTG | 658 | 858 | ---x xxxx | ---u uuuu | ---u uuuu |
| PORTF | 658 | 858 | x000 0000 | u000 0000 | uuuu uuuu |
| PORTE | 658 | 858 | --00 xxxx | uuuu u000 | uuuu uuuu |
| PORTD | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTC | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTB | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA ⁽⁵⁾ | 658 | 858 | -x0x 0000 ⁽⁵⁾ | -u0u 0000 ⁽⁵⁾ | -uuu uuuu ⁽⁵⁾ |
| TRISK | 658 | 858 | 1111 1111 | 1111 1111 | uuuu uuuu |
| LATK | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTK | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXERRCNT | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXERRCNT | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| COMSTAT | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CIOCON | 658 | 858 | 1000 ---- | 1000 ---- | uuuu ---- |
| BRGCON3 | 658 | 858 | -0-- -000 | -0-- -000 | -u-- -uuu |
| BRGCON2 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| BRGCON1 | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |

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

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 (0008h or 0018h).

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 STKPTR is modified to point to the next location in the hardware stack.

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

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: The long write enable is only reset on a POR or MCLR.

7: Available on PIC18C858 only.

PIC18CXX8

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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|----------|--------------------|-----|---------------------------------|--------------------------------------------------------------|------------------------------|
| CANCON | 658 | 858 | xxxx xxx- | uuuu uuu- | uuuu uuu- |
| CANSTAT | 658 | 858 | xxx- xxx- | uuu- uuu- | uuu- uuu- |
| RXB0D7 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D6 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D5 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D4 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D3 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D2 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D1 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D0 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0DLC | 658 | 858 | 0xxx xxxx | 0uuu uuuu | uuuu uuuu |
| RXB0EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0SIDL | 658 | 858 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| RXB0SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0CON | 658 | 858 | 000- 0000 | 000- 0000 | uuu- uuuu |
| RXB1D7 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D6 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D5 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D4 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D3 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D2 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D1 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D0 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1DLC | 658 | 858 | 0xxx xxxx | 0uuu uuuu | uuuu uuuu |
| RXB1EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1SIDL | 658 | 858 | xxxx x0xx | uuuu u0uu | uuuu uuuu |
| RXB1SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1CON | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXB0D7 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D6 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D5 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D4 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D3 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D2 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D1 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |

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

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 (0008h or 0018h).

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 STKPTR is modified to point to the next location in the hardware stack.

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

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: The long write enable is only reset on a POR or MCLR.

7: Available on PIC18C858 only.

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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|----------|--------------------|-----|---------------------------------|--------------------------------------------------------------|------------------------------|
| TXB0D0 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0DLC | 658 | 858 | 0x00 xxxx | 0u00 uuuu | uuuu uuuu |
| TXB0EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0SIDL | 658 | 858 | xxx0 x0xx | uuu0 u0uu | uuuu uuuu |
| TXB0SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0CON | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXB1D7 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D6 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D5 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D4 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D3 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D2 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D1 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D0 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1DLC | 658 | 858 | 0x00 xxxx | 0u00 uuuu | uuuu uuuu |
| TXB1EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1SIDL | 658 | 858 | xxx0 x0xx | uuu0 u0uu | uuuu uuuu |
| TXB1SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1CON | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXB2D7 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D6 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D5 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D4 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D3 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D2 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D1 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D0 | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2DLC | 658 | 858 | 0x00 xxxx | 0u00 uuuu | uuuu uuuu |
| TXB2EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2SIDL | 658 | 858 | xxx0 x0xx | uuu0 u0uu | uuuu uuuu |
| TXB2SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2CON | 658 | 858 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXM1EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM1EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |

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

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 (0008h or 0018h).
- 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 STKPTR is modified to point to the next location in the hardware stack.
- 4: See Table 3-2 for RESET value for specific condition.
- 5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
- 6: The long write enable is only reset on a POR or MCLR.
- 7: Available on PIC18C858 only.

PIC18CXX8

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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|----------|--------------------|-----|---------------------------------|-----------------------------------------------------------|------------------------------|
| RXM1SIDL | 658 | 858 | xxx- --xx | uuu- --uu | uuu- --uu |
| RXM1SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0SIDL | 658 | 858 | xxx- --xx | uuu- --uu | uuu- --uu |
| RXM0SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5SIDL | 658 | 858 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF5SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4SIDL | 658 | 858 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF4SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3SIDL | 658 | 858 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF3SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2SIDL | 658 | 858 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF2SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1SIDL | 658 | 858 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF1SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0EIDL | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0EIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0SIDL | 658 | 858 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF0SIDH | 658 | 858 | xxxx xxxx | uuuu uuuu | uuuu uuuu |

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

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 (0008h or 0018h).

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 STKPTR is modified to point to the next location in the hardware stack.

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

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: The long write enable is only reset on a POR or MCLR.

7: Available on PIC18C858 only.

4.0 MEMORY ORGANIZATION

There are two memory blocks in Enhanced MCU 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

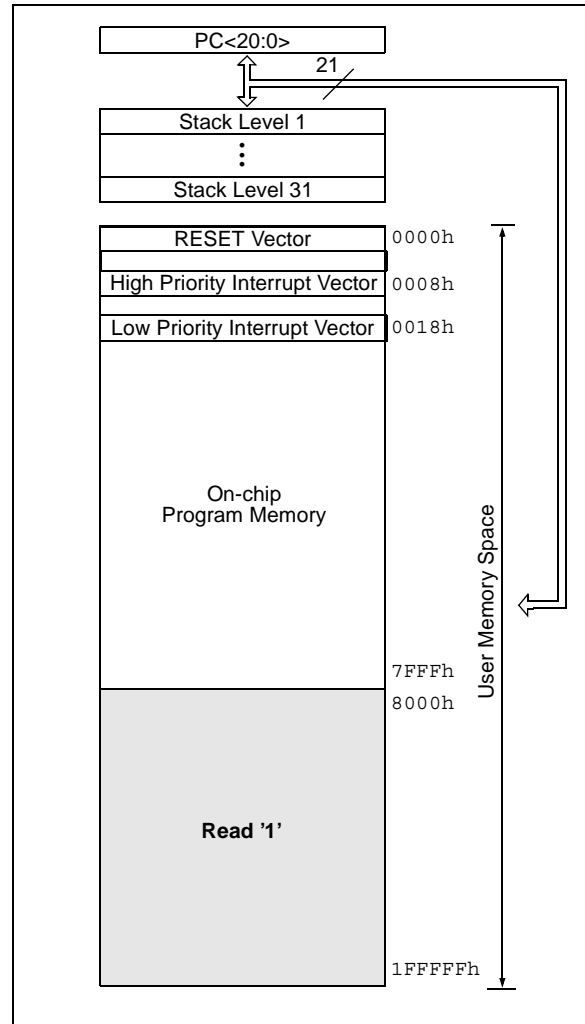
The PIC18CXX8 devices have a 21-bit program counter that is capable of addressing the 2 Mbyte program memory space.

The reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h. Figure 4-1 shows the diagram for program memory map and stack for the PIC18C658 and PIC18C858.

4.1.1 INTERNAL PROGRAM MEMORY OPERATION

All devices have 32 Kbytes of internal EPROM program memory. This means that the PIC18CXX8 devices can store up to 16K of single word instructions. Accessing a location between the physically implemented memory and the 2 Mbyte address will cause a read of all '0's (a NOP instruction).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR PIC18C658/858



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 5-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 indicate if the stack pointer is at or beyond the 31 levels provided.

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.

The action that takes place when the stack becomes full depends on the state of the `STVREN` (stack overflow RESET enable) configuration bit. Refer to Section 18 for a description of the device configuration bits. If `STVREN` is set (default) the 31st push will push the (`PC + 2`) value onto the stack, 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. The 32nd push will overwrite the 31st push (and so on), while `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'

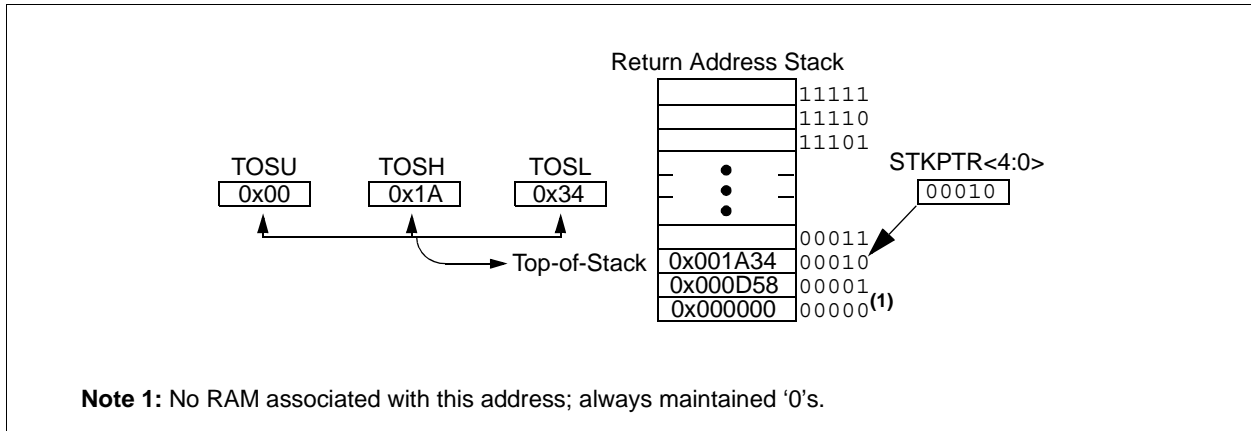
bit 4-0 **SP4:SP0:** Stack Pointer Location bits

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 4-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



Note 1: No RAM 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 pull 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 by programming the `STVREN` configuration bit. When the `STVREN` bit is disabled, a full or underflow condition will set the appropriate `STKFUL` or `STKUNF` bit, but not cause a device RESET. When the `STVREN` bit is enabled, a full or underflow will set the appropriate `STKFUL` or `STKUNF` bit and then cause a device 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 PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

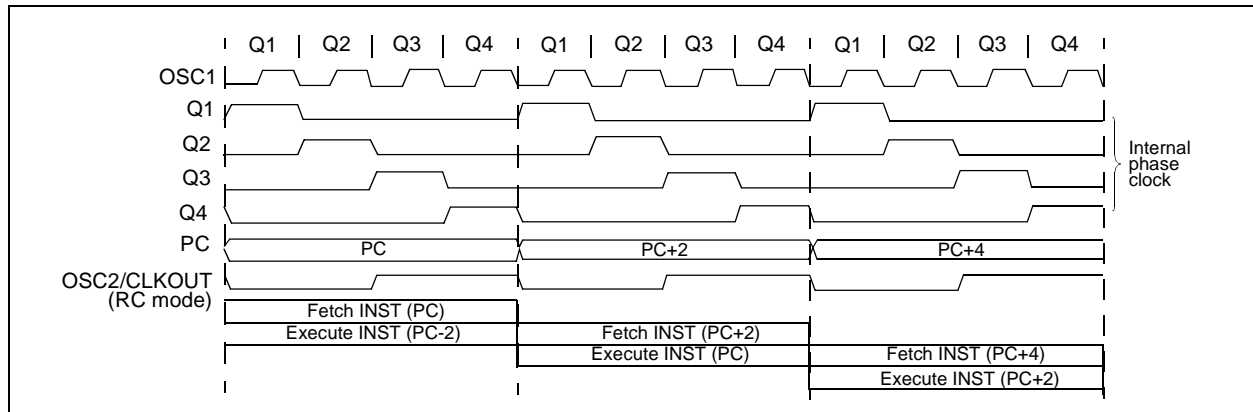
The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

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) 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 4-3.

FIGURE 4-3: CLOCK/INSTRUCTION CYCLE



PIC18CXX8

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 000006h” 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 23.0 provides further details of the instruction set.

EXAMPLE 4-2: INSTRUCTION PIPELINE FLOW

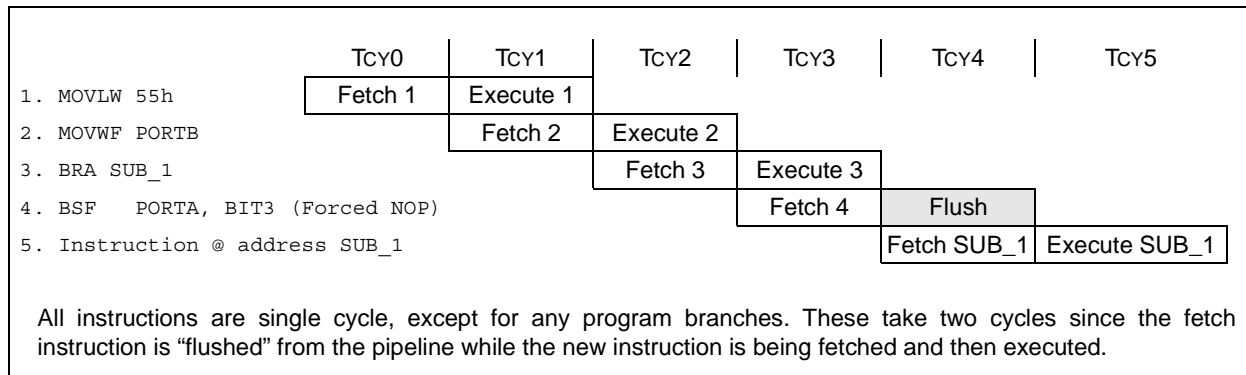


TABLE 4-1: INSTRUCTIONS IN PROGRAM MEMORY

| 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

The PIC18CXX8 devices have 4 two word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 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. 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. These are:

- 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 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 5.0.

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 becomes 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 4-4 shows the data memory organization for the PIC18CXX8 devices.

The data memory map is divided into as many as 16 banks that contain 256 bytes each. The lower 4 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 (0xFFF) 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.

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 the 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 (SFR's and select GPR's) 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 RAM. Section 4.10 provides a detailed description of the Access RAM.

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.

Enhanced MCU devices may have banked memory in the GPR area. GPR's 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 (0xF00 to 0xFFFF) contains SFR's. All other banks of data memory contain GPR registers starting with bank 0.

4.9.2 SPECIAL FUNCTION REGISTERS

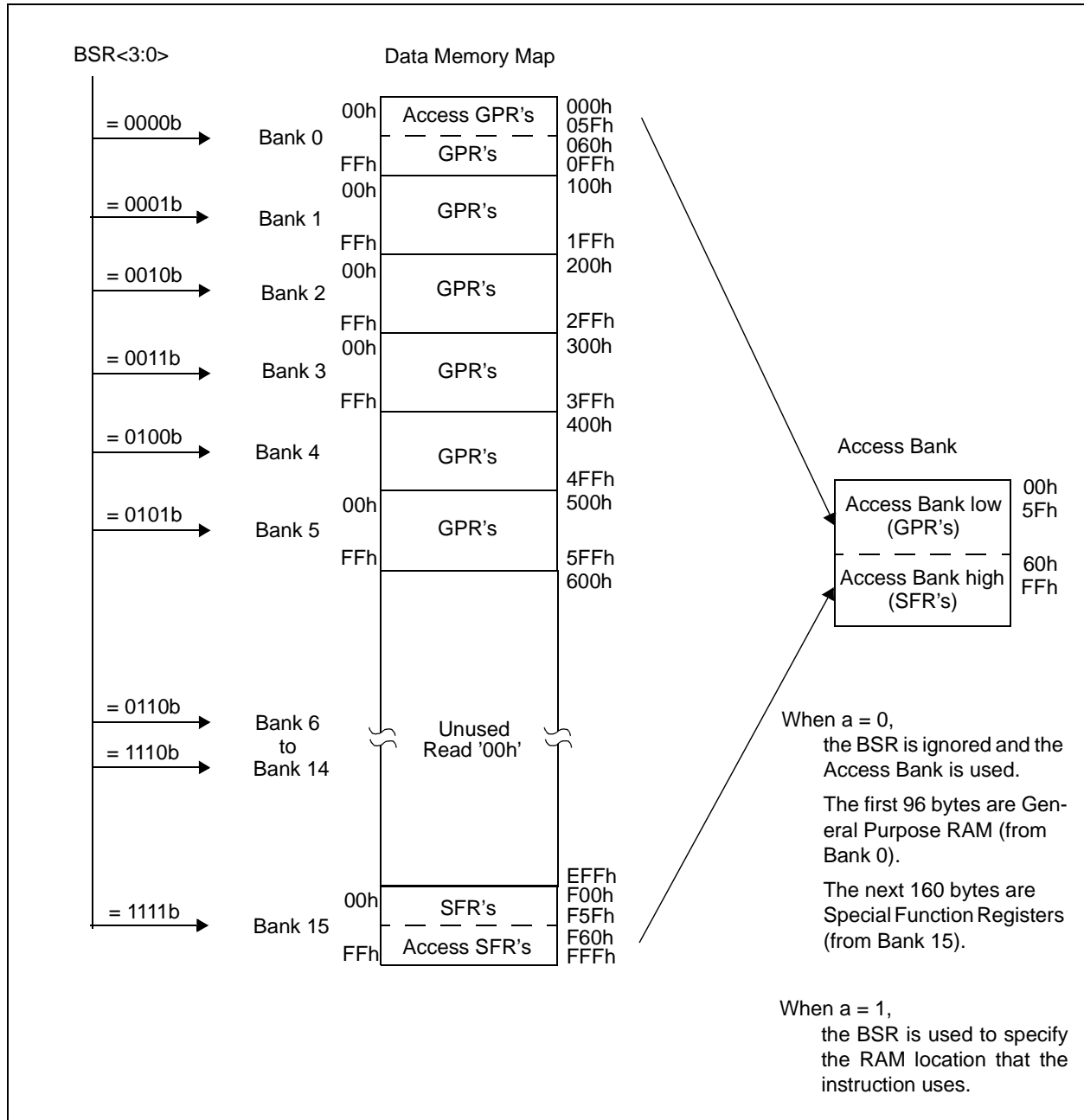
The Special Function Registers (SFR's) are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-2.

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 SFR's are typically distributed among the peripherals whose functions they control.

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

FIGURE 4-4: DATA MEMORY MAP FOR PIC18C658/858



PIC18CXX8

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP

| Address | Name | Address | Name | Address | Name | Address | Name |
|---------|-------------------------|---------|-------------------------|---------|---------|---------|----------------------|
| FFFh | TOSU | FDfH | INDF2 ⁽²⁾ | FBFh | CCPR1H | F9Fh | IPR1 |
| FFEh | TOSH | FDEh | POSTINC2 ⁽²⁾ | FBEh | CCPR1L | F9Eh | PIR1 |
| FFDh | TOSL | FDDh | POSTDEC2 ⁽²⁾ | FBDh | CCP1CON | F9Dh | PIE1 |
| FFCh | STKPTR | FDCh | PREINC2 ⁽²⁾ | FBCh | CCPR2H | F9Ch | — |
| FFBh | PCLATU | FDBh | PLUSW2 ⁽²⁾ | FBBh | CCPR2L | F9Bh | — |
| FFAh | PCLATH | FDAh | FSR2H | FBAh | CCP2CON | F9Ah | TRISJ ⁽⁵⁾ |
| FF9h | PCL | FD9h | FSR2L | FB9h | — | F99h | TRISH ⁽⁵⁾ |
| FF8h | TBLPTRU | FD8h | STATUS | FB8h | — | F98h | TRISG |
| FF7h | TBLPTRH | FD7h | TMR0H | FB7h | — | F97h | TRISF |
| FF6h | TBLPTRL | FD6h | TMR0L | FB6h | — | F96h | TRISE |
| FF5h | TABLAT | FD5h | T0CON | FB5h | CVRCON | F95h | TRISD |
| FF4h | PRODH | FD4h | — | FB4h | CMCON | 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 ⁽⁵⁾ |
| FF0h | INTCON3 | FD0h | RCON | FB0h | PSPCON | F90h | LATH ⁽⁵⁾ |
| FEFh | INDF0 ⁽²⁾ | FCFh | TMR1H | FAFh | SPBRG | F8Fh | LATG |
| FEEh | POSTINC0 ⁽²⁾ | FCEh | TMR1L | FAEh | RCREG | F8Eh | LATF |
| FEDh | POSTDEC0 ⁽²⁾ | FCDh | T1CON | FADh | TXREG | F8Dh | LATE |
| FECh | PREINC0 ⁽²⁾ | FCCh | TMR2 | FACH | TXSTA | F8Ch | LATD |
| FEBh | PLUSW0 ⁽²⁾ | 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 ⁽⁵⁾ |
| FE7h | INDF1 ⁽²⁾ | FC7h | SSPSTAT | FA7h | — | F87h | PORTH ⁽⁵⁾ |
| FE6h | POSTINC1 ⁽²⁾ | FC6h | SSPCON1 | FA6h | — | F86h | PORTG |
| FE5h | POSTDEC1 ⁽²⁾ | FC5h | SSPCON2 | FA5h | IPR3 | F85h | PORTF |
| FE4h | PREINC1 ⁽²⁾ | FC4h | ADRESH | FA4h | PIR3 | F84h | PORTE |
| FE3h | PLUSW1 ⁽²⁾ | FC3h | ADRESL | FA3h | PIE3 | 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 |

Note 1: Unimplemented registers are read as '0'.

2: This is not a physical register.

3: Contents of register is dependent on WIN2:WIN0 bits in CANCON register.

4: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the CANSTAT register due to the Microchip Header file requirement.

5: Available on PIC18C858 only.

| Address | Name | Address | Name | Address | Name | Address | Name |
|---------|-------------------------|---------|---------------------------|---------|---------------------------|---------|----------|
| F7Fh | TRISK ⁽⁵⁾ | F5Fh | — | F3Fh | — | F1Fh | RXM1EID0 |
| F7Eh | LATK ⁽⁵⁾ | F5Eh | CANSTATRO0 ⁽⁴⁾ | F3Eh | CANSTATRO2 ⁽⁴⁾ | F1Eh | RXM1EID8 |
| F7Dh | PORTK ⁽⁵⁾ | F5Dh | RXB1D7 | F3Dh | TXB1D7 | F1Dh | RXM1SIDL |
| F7Ch | — | F5Ch | RXB1D6 | F3Ch | TXB1D6 | F1Ch | RXM1SIDH |
| F7Bh | — | F5Bh | RXB1D5 | F3Bh | TXB1D5 | F1Bh | RXM0EID0 |
| F7Ah | — | F5Ah | RXB1D4 | F3Ah | TXB1D4 | F1Ah | RXM0EID8 |
| F79h | — | F59h | RXB1D3 | F39h | TXB1D3 | F19h | RXM0SIDL |
| F78h | — | F58h | RXB1D2 | F38h | TXB1D2 | F18h | RXM0SIDH |
| F77h | — | F57h | RXB1D1 | F37h | TXB1D1 | F17h | RXF5EID0 |
| F76h | TXERRCNT | F56h | RXB1D0 | F36h | TXB1D0 | F16h | RXF5EID8 |
| F75h | RXERRCNT | F55h | RXB1DLC | F35h | TXB1DLC | F15h | RXF5SIDL |
| F74h | COMSTAT | F54h | RXB1EIDL | F34h | TXB1EIDL | F14h | RXF5SIDH |
| F73h | CIOCON | F53h | RXB1EIDH | F33h | TXB1EIDH | F13h | RXF4EID0 |
| F72h | BRGCON3 | F52h | RXB1SIDL | F32h | TXB1SIDL | F12h | RXF4EID8 |
| F71h | BRGCON2 | F51h | RXB1SIDH | F31h | TXB1SIDH | F11h | RXF4SIDL |
| F70h | BRGCON1 | F50h | RXB1CON | F30h | TXB1CON | F10h | RXF4SIDH |
| F6Fh | CANCON | F4Fh | — | F2Fh | — | F0Fh | RXF3EID0 |
| F6Eh | CANSTAT | F4Eh | CANSTATRO1 ⁽⁴⁾ | F2Eh | CANSTATRO3 ⁽⁴⁾ | F0Eh | RXF3EID8 |
| F6Dh | RXB0D7 ⁽³⁾ | F4Dh | TXB0D7 | F2Dh | TXB2D7 | F0Dh | RXF3SIDL |
| F6Ch | RXB0D6 ⁽³⁾ | F4Ch | TXB0D6 | F2Ch | TXB2D6 | F0Ch | RXF3SIDH |
| F6Bh | RXB0D5 ⁽³⁾ | F4Bh | TXB0D5 | F2Bh | TXB2D5 | F0Bh | RXF2EID0 |
| F6Ah | RXB0D4 ⁽³⁾ | F4Ah | TXB0D4 | F2Ah | TXB2D4 | F0Ah | RXF2EID8 |
| F69h | RXB0D3 ⁽³⁾ | F49h | TXB0D3 | F29h | TXB2D3 | F09h | RXF2SIDL |
| F68h | RXB0D2 ⁽³⁾ | F48h | TXB0D2 | F28h | TXB2D2 | F08h | RXF2SIDH |
| F67h | RXB0D1 ⁽³⁾ | F47h | TXB0D1 | F27h | TXB2D1 | F07h | RXF1EID0 |
| F66h | RXB0D0 ⁽³⁾ | F46h | TXB0D0 | F26h | TXB2D0 | F06h | RXF1EID8 |
| F65h | RXB0DLC ⁽³⁾ | F45h | TXB0DLC | F25h | TXB2DLC | F05h | RXF1SIDL |
| F64h | RXB0EIDL ⁽³⁾ | F44h | TXB0EIDL | F24h | TXB2EIDL | F04h | RXF1SIDH |
| F63h | RXB0EIDH ⁽³⁾ | F43h | TXB0EIDH | F23h | TXB2EIDH | F03h | RXF0EIDL |
| F62h | RXB0SIDL ⁽³⁾ | F42h | TXB0SIDL | F22h | TXB2SIDL | F02h | RXF0EIDH |
| F61h | RXB0SIDH ⁽³⁾ | F41h | TXB0SIDH | F21h | TXB2SIDH | F01h | RXF0SIDL |
| F60h | RXB0CON ⁽³⁾ | F40h | TXB0CON | F20h | TXB2CON | F00h | RXF0SIDH |

Note: Shaded registers are available in Bank 15, while the rest are in Access Bank low.

Note 1: Unimplemented registers are read as '0'.

2: This is not a physical register.

3: Contents of register is dependent on WIN2:WIN0 bits in CANCON register.

4: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the CANSTAT register due to the Microchip Header file requirement.

5: Available on PIC18C858 only.

PIC18CXX8

TABLE 4-3: REGISTER FILE SUMMARY

| Filename | 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 ⁽³⁾ | | |
|----------|---------------------------------------------------------------------------------------------------------------------------------------|-----------|-----------------------|---------------------------------------------------------|---------------------------------------------|--------|--------|--------|-------------------|------------------------------------------|-----------|-----------|
| TOSU | — | — | — | Top-of-Stack upper Byte (TOS<20:16>) | | | | | ---0 0000 | ---0 0000 | | |
| TOSH | Top-of-Stack High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | 0000 0000 | | |
| TOSL | Top-of-Stack Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | 0000 0000 | | |
| STKPTR | STKFUL | STKUNF | — | Return Stack Pointer | | | | | 00-0 0000 | 00-0 0000 | | |
| PCLATU | — | — | bit 21 ⁽³⁾ | Holding Register for PC<20:16> | | | | | | | --00 0000 | --00 0000 |
| PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | 0000 0000 | | |
| PCL | PC Low Byte (PC<7:0>) | | | | | | | | 0000 0000 | 0000 0000 | | |
| TBLPTRU | — | — | bit 21 ⁽²⁾ | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | | | ---0 0000 | ---0 0000 |
| TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 0000 0000 | | |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 0000 0000 | | |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 0000 0000 | | |
| PRODH | Product Register High Byte | | | | | | | | xxxx xxxx | uuuu uuuu | | |
| PRODL | Product Register Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u | | |
| INTCON2 | RBPV | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP | 1111 1111 | 1111 1111 | | |
| INTCON3 | INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF | 1100 0000 | 1100 0000 | | |
| INDF0 | Uses contents of FSR0 to address data memory - value of FSR0 not changed (not a physical register) | | | | | | | | n/a | n/a | | |
| POSTINC0 | Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register) | | | | | | | | n/a | n/a | | |
| POSTDEC0 | Uses contents of FSR0 to address data memory - value of FSR0 post-decremented (not a physical register) | | | | | | | | n/a | n/a | | |
| PREINC0 | Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register) | | | | | | | | n/a | n/a | | |
| PLUSW0 | Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register) - value of FSR0 offset by WREG | | | | | | | | n/a | n/a | | |
| FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High | | | | | ---- 0000 | ---- 0000 | |
| FSR0L | Indirect Data Memory Address Pointer 0 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu | | |
| WREG | Working Register | | | | | | | | xxxx xxxx | uuuu uuuu | | |
| INDF1 | Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register) | | | | | | | | n/a | n/a | | |
| POSTINC1 | Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a physical register) | | | | | | | | n/a | n/a | | |
| POSTDEC1 | Uses contents of FSR1 to address data memory - value of FSR1 post-decremented (not a physical register) | | | | | | | | n/a | n/a | | |
| PREINC1 | Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register) | | | | | | | | n/a | n/a | | |
| PLUSW1 | Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register) - value of FSR1 offset by WREG | | | | | | | | n/a | n/a | | |
| FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High | | | | | ---- 0000 | ---- 0000 | |
| FSR1L | Indirect Data Memory Address Pointer 1 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu | | |
| BSR | — | — | — | — | Bank Select Register | | | | | ---- 0000 | ---- 0000 | |

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

Note 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: Other (non-power-up) RESETs include external RESET through MCLR and Watchdog Timer Reset.

4: These registers are reserved on PIC18C658.

PIC18CXX8

| Filename | 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 ⁽³⁾ |
|----------|---------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-------------------------|------------------------|---------------------------------------------|----------------------------|-----------------------------|-------------------------|-------------------|------------------------------------------|
| INDF2 | Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register) | | | | | | | | n/a | n/a |
| POSTINC2 | Uses contents of FSR2 to address data memory - value of FSR2 post-incremented (not a physical register) | | | | | | | | n/a | n/a |
| POSTDEC2 | Uses contents of FSR2 to address data memory - value of FSR2 post-decremented (not a physical register) | | | | | | | | n/a | n/a |
| PREINC2 | Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register) | | | | | | | | n/a | n/a |
| PLUSW2 | Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register) - value of FSR2 offset by WREG | | | | | | | | n/a | n/a |
| FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High | | | | ---- 0000 | ---- 0000 |
| FSR2L | Indirect Data Memory Address Pointer 2 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | ---u uuuu |
| TMR0H | Timer0 register high byte | | | | | | | | 0000 0000 | 0000 0000 |
| TMR0L | Timer0 register low byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | T0PS3 | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 1111 1111 |
| OSCCON | — | — | — | — | — | — | — | SCS | ---- --0 | ---- --0 |
| LVDCON | — | — | IRVST | LVDEN | LVDL3 | LVDL2 | LVDL1 | LVDL0 | --00 0101 | --00 0101 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | ---- --0 | ---- --0 |
| RCON | IPEN | $\overline{\text{LWRT}}$ | — | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 00-1 11qq | 00-q qquu |
| TMR1H | Timer1 Register High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR1L | Timer1 Register Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0-00 0000 | u-uu uuuu |
| TMR2 | Timer2 Register | | | | | | | | 0000 0000 | 0000 0000 |
| PR2 | Timer2 Period Register | | | | | | | | 1111 1111 | 1111 1111 |
| T2CON | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | -000 0000 |
| SSPBUF | SSP Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| SSPADD | SSP Address Register in I ² C Slave mode. SSP Baud Rate Reload Register in I ² C Master mode. | | | | | | | | 0000 0000 | 0000 0000 |
| SSPSTAT | SMP | CKE | $\overline{\text{D/A}}$ | P | S | $\overline{\text{R/W}}$ | UA | BF | 0000 0000 | 0000 0000 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 0000 0000 |
| SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 0000 0000 |
| ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| ADCON0 | — | — | CHS3 | CHS2 | CHS1 | CHS0 | $\overline{\text{GO/DONE}}$ | ADON | --00 0000 | --00 0000 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | -000 0000 | -000 0000 |
| ADCON2 | ADFM | — | — | — | — | ADCS2 | ADCS1 | ADCS0 | 0--- -000 | 0--- -000 |

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

- Note**
- 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
 - 2: Bit 21 of the TBLPTRU allows access to the device configuration bits.
 - 3: Other (non-power-up) RESETs include external RESET through MCLR and Watchdog Timer Reset.
 - 4: These registers are reserved on PIC18C658.

PIC18CXX8

| Filename | 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 ⁽³⁾ | |
|----------------------|-------------------------------------------|----------------------|-------------------------------------------|-------------------------------------------|--------|-----------------|--------|--------|-------------------|------------------------------------------|------|
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu | |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu | |
| CCP1CON | — | — | DC1B1 | DC1B0 | CCPM3 | CCP1M2 | CCP1M1 | CCP1M0 | --00 0000 | --00 0000 | |
| CCPR2H | Capture/Compare/PWM Register 2 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu | |
| CCPR2L | Capture/Compare/PWM Register 2 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu | |
| CCP2CON | — | — | DC2B1 | DC2B0 | CCPM3 | CCP2M2 | CCP2M1 | CCP2M0 | 0000 0000 | 0000 0000 | |
| VRCON | VREN | VROEN | VRR | VRSS | VR3 | VR2 | VR1 | VR0 | 0000 0000 | 0000 0000 | |
| CMCON | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 0000 0000 | 0000 0000 | |
| TMR3H | Timer3 Register High Byte | | | | | | | | xxxx xxxx | uuuu uuuu | |
| TMR3L | Timer3 Register Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu | |
| T3CON | RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | T3SYN \bar{C} | TMR3CS | TMR3ON | 0000 0000 | uuuu uuuu | |
| PSPCON | IBF | OBF | IBOV | PSPMODE | — | — | — | — | 0000 ---- | 0000 ---- | |
| SPBRG | USART Baud Rate Generator | | | | | | | | 0000 0000 | 0000 0000 | |
| RCREG | USART Receive Register | | | | | | | | 0000 0000 | 0000 0000 | |
| TXREG | USART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 | |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 | |
| RCSTA | SPEN | RX9 | SREN | CREN | ADEN | FERR | OERR | RX9D | 0000 000x | 0000 000x | |
| IPR3 | IRXIP | WAKIP | ERRIP | TXB2IP | TXB1IP | TXB0IP | RXB1IP | RXB0IP | 1111 1111 | 1111 1111 | |
| PIR3 | IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF | TXB0IF | RXB1IF | RXB0IF | 0000 0000 | 0000 0000 | |
| PIE3 | IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE | TXB0IE | RXB1IE | RXB0IE | 0000 0000 | 0000 0000 | |
| IPR2 | — | CMIP | — | — | BCLIP | LVDIP | TMR3IP | CCP2IP | -1-- 1111 | -1-- 1111 | |
| PIR2 | — | CMIF | — | — | BCLIF | LVDIF | TMR3IF | CCP2IF | -0-- 0000 | -0-- 0000 | |
| PIE2 | — | CMIE | — | — | BCLIE | LVDIE | TMR3IE | CCP2IE | -0-- 0000 | -0-- 0000 | |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 | |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 | |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 | |
| TRISJ ⁽⁴⁾ | Data Direction Control Register for PORTJ | | | | | | | | 1111 1111 | 1111 1111 | |
| TRISH ⁽⁴⁾ | Data Direction Control Register for PORTH | | | | | | | | 1111 1111 | 1111 1111 | |
| TRISG | — | — | — | Data Direction Control Register for PORTG | | | | --- | 1111 | --- | 1111 |
| TRISF | Data Direction Control Register for PORTF | | | | | | | | 1111 1111 | 1111 1111 | |
| TRISE | Data Direction Control Register for PORTE | | | | | | | | 1111 1111 | 1111 1111 | |
| TRISD | Data Direction Control Register for PORTD | | | | | | | | 1111 1111 | 1111 1111 | |
| TRISC | Data Direction Control Register for PORTC | | | | | | | | 1111 1111 | 1111 1111 | |
| TRISB | Data Direction Control Register for PORTB | | | | | | | | 1111 1111 | 1111 1111 | |
| TRISA | — | Bit 6 ⁽¹⁾ | Data Direction Control Register for PORTA | | | | | --- | 1111 | --- | 1111 |

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

- Note**
- 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
 - 2: Bit 21 of the TBLPTRU allows access to the device configuration bits.
 - 3: Other (non-power-up) RESETs include external RESET through MCLR and Watchdog Timer Reset.
 - 4: These registers are reserved on PIC18C658.

PIC18CXX8

| Filename | 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 ⁽³⁾ |
|----------------------|-----------------------------------------------|----------------------|-----------------------------------------------|-----------------------------------------------|---------|---------|---------|-----------|-------------------|------------------------------------------|
| LATJ ⁽⁴⁾ | Read PORTJ Data Latch, Write PORTJ Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATH ⁽⁴⁾ | Read PORTH Data Latch, Write PORTH Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATG | — | — | — | Read PORTG Data Latch, Write PORTG Data Latch | | | | | --x xxxx | --u uuuu |
| LATF | Read PORTF Data Latch, Write PORTF Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATE | Read PORTE Data Latch, Write PORTE Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATD | Read PORTD Data Latch, Write PORTD Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATC | Read PORTC Data Latch, Write PORTC Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATB | Read PORTB Data Latch, Write PORTB Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATA | — | Bit 6 ⁽¹⁾ | Read PORTA Data Latch, Write PORTA Data Latch | | | | | --xx xxxx | --uu uuuu | |
| PORTJ ⁽⁴⁾ | Read PORTJ pins, Write PORTJ Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| PORTH ⁽⁴⁾ | Read PORTH pins, Write PORTH Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| PORTG | — | — | — | Read PORTG pins, Write PORTG Data Latch | | | | | --x xxxx | uuuu uuuu |
| PORTF | Read PORTF pins, Write PORTF Data Latch | | | | | | | | 0000 0000 | 0000 0000 |
| PORTE | Read PORTE pins, Write PORTE Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| PORTD | Read PORTD pins, Write PORTD Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| PORTC | Read PORTC pins, Write PORTC Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| PORTB | Read PORTB pins, Write PORTB Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| PORTA | — | Bit 6 ⁽¹⁾ | Read PORTA pins, Write PORTA Data Latch | | | | | --0x 0000 | --0u 0000 | |
| TRISK ⁽⁴⁾ | Data Direction Control Register for PORTK | | | | | | | | 1111 1111 | 1111 1111 |
| LATK ⁽⁴⁾ | Read PORTK Data Latch, Write PORTK Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| PORTK ⁽⁴⁾ | Read PORTK pins, Write PORTK Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| TXERRCNT | TEC7 | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0 | 0000 0000 | 0000 0000 |
| RXERRCNT | REC7 | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0 | 0000 0000 | 0000 0000 |
| COMSTAT | RXB0OVFL | RXB1OVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN | 0000 0000 | 0000 0000 |
| CIOCON | TX1SRC | TX1EN | ENDRHI | CANCAP | — | — | — | — | 1000 ---- | 1000 ---- |
| BRGCON3 | — | WAKFIL | — | — | — | SEG2PH2 | SEG2PH1 | SEG2PH0 | -0-- -000 | -0-- -000 |
| BRGCON2 | SEG2PHTS | SAM | SEG1PH2 | SEG1PH1 | SEG1PH0 | PRSEG2 | PRSEG1 | PRSEG0 | 0000 0000 | 0000 0000 |
| BRGCON1 | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | 0000 0000 | 0000 0000 |
| CANCON | REQOP2 | REQOP1 | REQOP0 | ABAT | WIN2 | WIN1 | WIN0 | — | xxxx xxx- | uuuu uu- |
| CANSTAT | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICOED0 | — | xxx- xxx- | uu- uu- |

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

Note 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

Note 2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

Note 3: Other (non-power-up) RESEts include external RESET through MCLR and Watchdog Timer Reset.

Note 4: These registers are reserved on PIC18C658.

PIC18CXX8

| Filename | 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 ⁽³⁾ |
|----------|---------|---------|---------|---------|---------|----------|---------|---------|-------------------|------------------------------------------|
| RXB0D7 | RXB0D77 | RXB0D76 | RXB0D75 | RXB0D74 | RXB0D73 | RXB0D72 | RXB0D71 | RXB0D70 | xxxx xxxx | uuuu uuuu |
| RXB0D6 | RXB0D67 | RXB0D66 | RXB0D65 | RXB0D64 | RXB0D63 | RXB0D62 | RXB0D61 | RXB0D60 | xxxx xxxx | uuuu uuuu |
| RXB0D5 | RXB0D57 | RXB0D56 | RXB0D55 | RXB0D54 | RXB0D53 | RXB0D52 | RXB0D51 | RXB0D50 | xxxx xxxx | uuuu uuuu |
| RXB0D4 | RXB0D47 | RXB0D46 | RXB0D45 | RXB0D44 | RXB0D43 | RXB0D42 | RXB0D41 | RXB0D40 | xxxx xxxx | uuuu uuuu |
| RXB0D3 | RXB0D37 | RXB0D36 | RXB0D35 | RXB0D34 | RXB0D33 | RXB0D32 | RXB0D31 | RXB0D30 | xxxx xxxx | uuuu uuuu |
| RXB0D2 | RXB0D27 | RXB0D26 | RXB0D25 | RXB0D24 | RXB0D23 | RXB0D22 | RXB0D21 | RXB0D20 | xxxx xxxx | uuuu uuuu |
| RXB0D1 | RXB0D17 | RXB0D16 | RXB0D15 | RXB0D14 | RXB0D13 | RXB0D12 | RXB0D11 | RXB0D10 | xxxx xxxx | uuuu uuuu |
| RXB0D0 | RXB0D07 | RXB0D06 | RXB0D05 | RXB0D04 | RXB0D03 | RXB0D02 | RXB0D0? | RXB0D00 | xxxx xxxx | uuuu uuuu |
| RXB0DLC | — | RXRTR | RESB1 | RESB0 | DLC3 | DLC2 | DLC1 | DLC0 | 0xxx xxxx | 0uuu uuuu |
| RXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXB0SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | uuuu u-uu |
| RXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXB0CON | RXFUL | RXM1 | RXM0 | — | RXRTRRO | RXB0DBEN | JTOFF | FILHIT0 | 000- 0000 | 000- 0000 |
| CANSTAT | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | uuu- uuu- |
| RXB1D7 | RXB1D77 | RXB1D76 | RXB1D75 | RXB1D74 | RXB1D73 | RXB1D72 | RXB1D71 | RXB1D70 | xxxx xxxx | uuuu uuuu |
| RXB1D6 | RXB1D67 | RXB1D66 | RXB1D65 | RXB1D64 | RXB1D63 | RXB1D62 | RXB1D61 | RXB1D60 | xxxx xxxx | uuuu uuuu |
| RXB1D5 | RXB1D57 | RXB1D56 | RXB1D55 | RXB1D54 | RXB1D53 | RXB1D52 | RXB1D51 | RXB1D50 | xxxx xxxx | uuuu uuuu |
| RXB1D4 | RXB1D47 | RXB1D46 | RXB1D45 | RXB1D44 | RXB1D43 | RXB1D42 | RXB1D41 | RXB1D40 | xxxx xxxx | uuuu uuuu |
| RXB1D3 | RXB1D37 | RXB1D36 | RXB1D35 | RXB1D34 | RXB1D33 | RXB1D32 | RXB1D31 | RXB1D30 | xxxx xxxx | uuuu uuuu |
| RXB1D2 | RXB1D27 | RXB1D26 | RXB1D25 | RXB1D24 | RXB1D23 | RXB1D22 | RXB1D21 | RXB1D20 | xxxx xxxx | uuuu uuuu |
| RXB1D1 | RXB1D17 | RXB1D16 | RXB1D15 | RXB1D14 | RXB1D13 | RXB1D12 | RXB1D11 | RXB1D10 | xxxx xxxx | uuuu uuuu |
| RXB1D0 | RXB1D07 | RXB1D06 | RXB1D05 | RXB1D04 | RXB1D03 | RXB1D02 | RXB1D01 | RXB1D00 | xxxx xxxx | uuuu uuuu |
| RXB1DLC | — | RXRTR | RESB1 | RESB0 | DLC3 | DLC2 | DLC1 | DLC0 | 0xxx xxxx | 0uuu uuuu |
| RXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXB1SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x0xx | uuuu u0uu |
| RXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXB1CON | RXFUL | RXM1 | RXM0 | — | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 0000 0000 |
| CANSTAT | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | uuu- uuu- |

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

- Note** 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
 2: Bit 21 of the TBLPTRU allows access to the device configuration bits.
 3: Other (non-power-up) RESEts include external RESET through MCLR and Watchdog Timer Reset.
 4: These registers are reserved on PIC18C658.

| Filename | 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 ⁽³⁾ |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|-------------------|------------------------------------------|
| TXB0D7 | TXB0D77 | TXB0D76 | TXB0D75 | TXB0D74 | TXB0D73 | TXB0D72 | TXB0D71 | TXB0D70 | xxxx xxxx | uuuu uuuu |
| TXB0D6 | TXB0D67 | TXB0D66 | TXB0D65 | TXB0D64 | TXB0D63 | TXB0D62 | TXB0D61 | TXB0D60 | xxxx xxxx | uuuu uuuu |
| TXB0D5 | TXB0D57 | TXB0D56 | TXB0D55 | TXB0D54 | TXB0D53 | TXB0D52 | TXB0D51 | TXB0D50 | xxxx xxxx | uuuu uuuu |
| TXB0D4 | TXB0D47 | TXB0D46 | TXB0D45 | TXB0D44 | TXB0D43 | TXB0D42 | TXB0D41 | TXB0D40 | xxxx xxxx | uuuu uuuu |
| TXB0D3 | TXB0D37 | TXB0D36 | TXB0D35 | TXB0D34 | TXB0D33 | TXB0D32 | TXB0D31 | TXB0D30 | xxxx xxxx | uuuu uuuu |
| TXB0D2 | TXB0D27 | TXB0D26 | TXB0D25 | TXB0D24 | TXB0D23 | TXB0D22 | TXB0D21 | TXB0D20 | xxxx xxxx | uuuu uuuu |
| TXB0D1 | TXB0D17 | TXB0D16 | TXB0D15 | TXB0D14 | TXB0D13 | TXB0D12 | TXB0D11 | TXB0D10 | xxxx xxxx | uuuu uuuu |
| TXB0D0 | TXB0D07 | TXB0D06 | TXB0D05 | TXB0D04 | TXB0D03 | TXB0D02 | TXB0D01 | TXB0D00 | xxxx xxxx | uuuu uuuu |
| TXB0DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | 0x00 xxxx | 0u00 uuuu |
| TXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| TXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| TXB0SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx0 x0xx | uuu0 u0uu |
| TXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| TXB0CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0000 | 0000 0000 |
| CANSTAT | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | uuu- uuu- |
| TXB1D7 | TXB1D77 | TXB1D76 | TXB1D75 | TXB1D74 | TXB1D73 | TXB1D72 | TXB1D71 | TXB1D70 | xxxx xxxx | uuuu uuuu |
| TXB1D6 | TXB1D67 | TXB1D66 | TXB1D65 | TXB1D64 | TXB1D63 | TXB1D62 | TXB1D61 | TXB1D60 | xxxx xxxx | uuuu uuuu |
| TXB1D5 | TXB1D57 | TXB1D56 | TXB1D55 | TXB1D54 | TXB1D53 | TXB1D52 | TXB1D51 | TXB1D50 | xxxx xxxx | uuuu uuuu |
| TXB1D4 | TXB1D47 | TXB1D46 | TXB1D45 | TXB1D44 | TXB1D43 | TXB1D42 | TXB1D41 | TXB1D40 | xxxx xxxx | uuuu uuuu |
| TXB1D3 | TXB1D37 | TXB1D36 | TXB1D35 | TXB1D34 | TXB1D33 | TXB1D32 | TXB1D31 | TXB1D30 | xxxx xxxx | uuuu uuuu |
| TXB1D2 | TXB1D27 | TXB1D26 | TXB1D25 | TXB1D24 | TXB1D23 | TXB1D22 | TXB1D21 | TXB1D20 | xxxx xxxx | uuuu uuuu |
| TXB1D1 | TXB1D17 | TXB1D16 | TXB1D15 | TXB1D14 | TXB1D13 | TXB1D12 | TXB1D11 | TXB1D10 | xxxx xxxx | uuuu uuuu |
| TXB1D0 | TXB1D07 | TXB1D06 | TXB1D05 | TXB1D04 | TXB1D03 | TXB1D02 | TXB1D01 | TXB1D00 | xxxx xxxx | uuuu uuuu |
| TXB1DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | 0x00 xxxx | 0u00 uuuu |
| TXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| TXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| TXB1SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx0 x0xx | uuu0 u0uu |
| TXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| TXB1CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0000 | 0000 0000 |
| CANSTAT | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | uuu- uuu- |

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

- Note**
- 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
 - 2: Bit 21 of the TBLPTRU allows access to the device configuration bits.
 - 3: Other (non-power-up) RESETs include external RESET through MCLR and Watchdog Timer Reset.
 - 4: These registers are reserved on PIC18C658.

PIC18CXX8

| Filename | 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 ⁽³⁾ |
|----------|---------|---------|---------|---------|---------|---------|---------|---------|-------------------|------------------------------------------|
| TXB2D7 | TXB2D77 | TXB2D76 | TXB2D75 | TXB2D74 | TXB2D73 | TXB2D72 | TXB2D71 | TXB2D70 | xxxx xxxx | uuuu uuuu |
| TXB2D6 | TXB2D67 | TXB2D66 | TXB2D65 | TXB2D64 | TXB2D63 | TXB2D62 | TXB2D61 | TXB2D60 | xxxx xxxx | uuuu uuuu |
| TXB2D5 | TXB2D57 | TXB2D56 | TXB2D55 | TXB2D54 | TXB2D53 | TXB2D52 | TXB2D51 | TXB2D50 | xxxx xxxx | uuuu uuuu |
| TXB2D4 | TXB2D47 | TXB2D46 | TXB2D45 | TXB2D44 | TXB2D43 | TXB2D42 | TXB2D41 | TXB2D40 | xxxx xxxx | uuuu uuuu |
| TXB2D3 | TXB2D37 | TXB2D36 | TXB2D35 | TXB2D34 | TXB2D33 | TXB2D32 | TXB2D31 | TXB2D30 | xxxx xxxx | uuuu uuuu |
| TXB2D2 | TXB2D27 | TXB2D26 | TXB2D25 | TXB2D24 | TXB2D23 | TXB2D22 | TXB2D21 | TXB2D20 | xxxx xxxx | uuuu uuuu |
| TXB2D1 | TXB2D17 | TXB2D16 | TXB2D15 | TXB2D14 | TXB2D13 | TXB2D12 | TXB2D11 | TXB2D10 | xxxx xxxx | uuuu uuuu |
| TXB2D0 | TXB2D07 | TXB2D06 | TXB2D05 | TXB2D04 | TXB2D03 | TXB2D02 | TXB2D01 | TXB2D00 | xxxx xxxx | uuuu uuuu |
| TXB2DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | 0x00 xxxx | 0u00 uuuu |
| TXB2EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| TXB2EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| TXB2SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx0 x0xx | uuu0 u0uu |
| TXB2SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| TXB2CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0000 | 0000 0000 |
| RXM1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXM1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXM1SIDL | SID2 | SID1 | SID0 | — | — | — | EID17 | EID16 | xxx- --xx | uuu- --uu |
| RXM1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXM0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXM0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXM0SIDL | SID2 | SID1 | SID0 | — | — | — | EID17 | EID16 | xxx- --xx | uuu- --uu |
| RXM0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXF5EID0 | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXF5EID8 | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXF5SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | uuu- u-uu |
| RXF5SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXF4EID0 | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXF4EID8 | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXF4SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | uuu- u-uu |
| RXF4SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXF3EID0 | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXF3EID8 | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXF3SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | uuu- u-uu |
| RXF3SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |

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

- Note**
- 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
 - 2: Bit 21 of the TBLPTRU allows access to the device configuration bits.
 - 3: Other (non-power-up) RESETs include external RESET through MCLR and Watchdog Timer Reset.
 - 4: These registers are reserved on PIC18C658.

| Filename | 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 ⁽³⁾ |
|----------|-------|-------|-------|-------|--------|-------|-------|-------|-------------------|------------------------------------------|
| RXF2EID0 | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXF2EID8 | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXF2SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | uuu- u-uu |
| RXF2SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXF1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXF1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXF1SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | uuu- u-uu |
| RXF1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |
| RXF0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | uuuu uuuu |
| RXF0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | uuuu uuuu |
| RXF0SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | uuu- u-uu |
| RXF0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | uuuu uuuu |

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

- Note 1:** Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.
- 2:** Bit 21 of the TBLPTRU allows access to the device configuration bits.
- 3:** Other (non-power-up) RESETs include external RESET through MCLR and Watchdog Timer Reset.
- 4:** These registers are reserved on PIC18C658.

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 be 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 160 bytes in Bank 15 (SFR's) and the lower 96 bytes in Bank 0. These two sections will be referred to as Access Bank High and Access Bank Low, respectively. Figure 4-4 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 most of the 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. The data memory is partitioned into sixteen banks. 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.

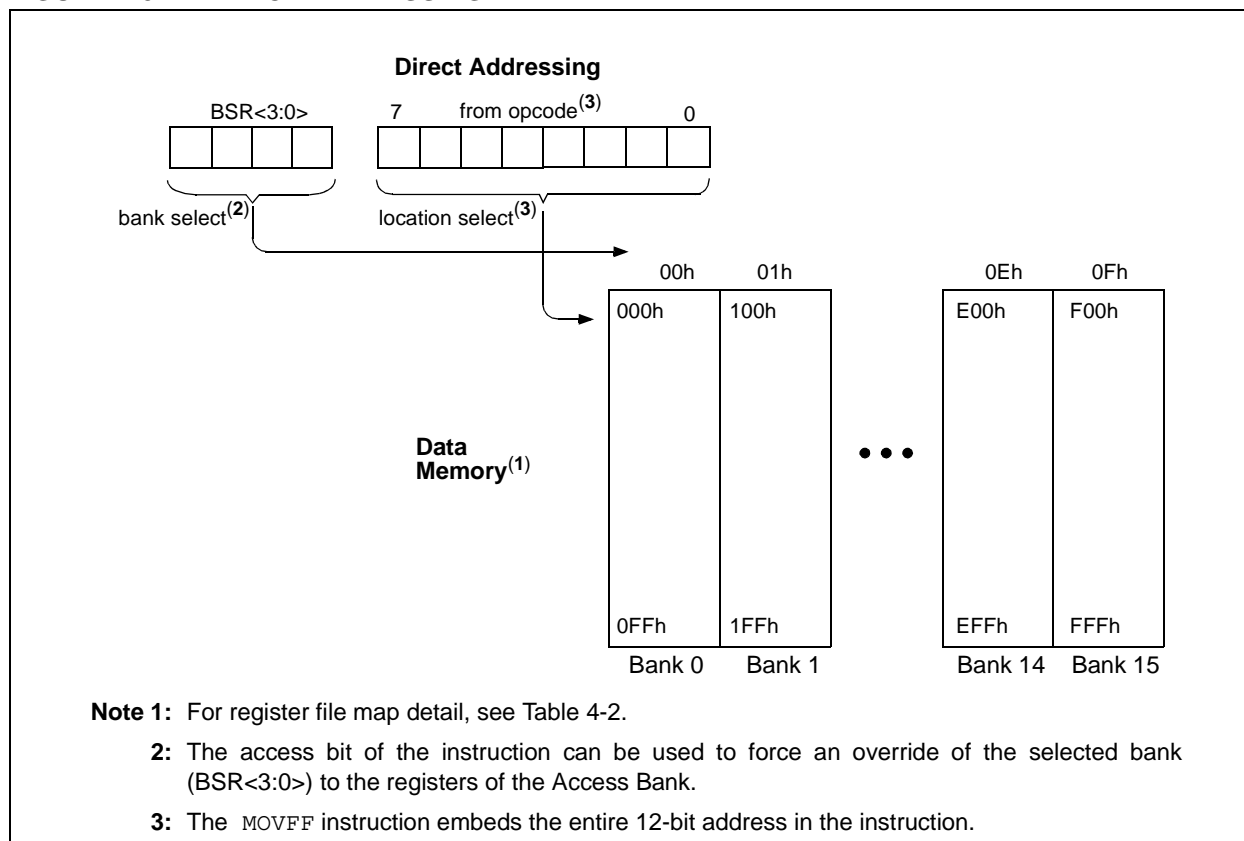
If the currently selected bank is not implemented, any read will return all '0's and all writes are ignored. The STATUS register bits will be set/cleared as appropriate for the instruction performed.

Each Bank extends up to FFh (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 4-5: DIRECT ADDRESSING



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 4-6 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 INDF registers. Any instruction using the INDF register actually accesses the register indicated by the File Select Register, FSR. Reading the INDF register itself indirectly (FSR = '0') will read 00h. Writing to the INDF register indirectly results in a no-operation. The FSR register contains a 12-bit address, which is shown in Figure 4-6.

The INDF_n (0 ≤ n ≤ 2) register is not a physical register. Addressing INDF_n actually addresses the register whose address is contained in the FSR_n register (FSR_n is a pointer). This is indirect addressing.

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

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

```

LFSR    FSR0, 0x100 ;
NEXT CLRFB    POSTINC0    ; Clear INDF
        ; register
        ; & inc pointer
        BTFSS    FSR0H, 1    ; All done
        ; w/ Bank1?
        GOTO     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. INDF_n 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 INDF_n locations, the address selected will configure the FSR_n register to:

- Do nothing to FSR_n after an indirect access (no change) - INDF_n
- Auto-decrement FSR_n after an indirect access (post-decrement) - POSTDEC_n
- Auto-increment FSR_n after an indirect access (post-increment) - POSTINC_n
- Auto-increment FSR_n before an indirect access (pre-increment) - PREINC_n
- Use the value in the WREG register as an offset to FSR_n. Do not modify the value of the WREG or the FSR_n register after an indirect access (no change) - PLUSW_n

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 FSR_nL overflows from an increment, FSR_nH will be incremented automatically.

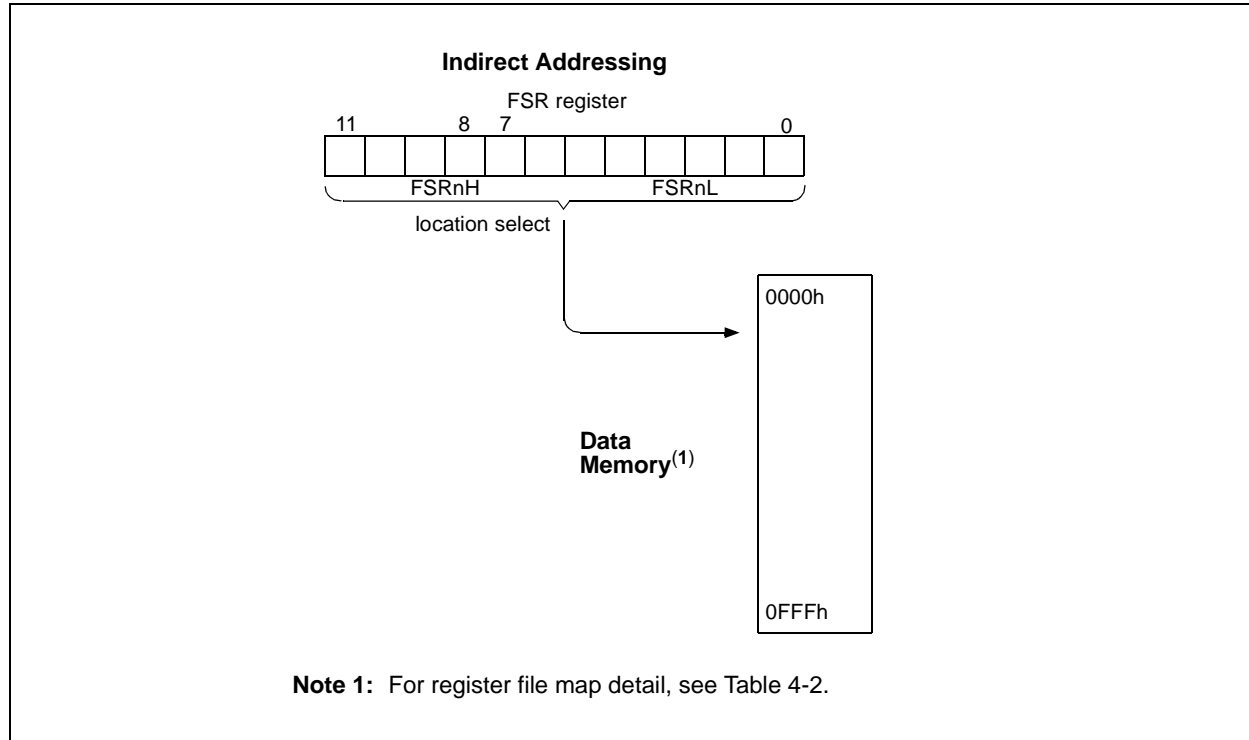
Adding these features allows the FSR_n to be used as a software stack pointer in addition to its uses for table operations in data memory.

Each FSR has an address associated with it that performs an indexed indirect access. When a data access to this INDF_n location (PLUSW_n) occurs, the FSR_n 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 FSR register contains a value that indicates one of the INDF_n, an indirect read will read 00h (zero bit is set), while an indirect write will be equivalent to a NOP (STATUS bits are not affected).

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

FIGURE 4-6: INDIRECT ADDRESSING



4.13 STATUS Register

The STATUS register, shown in Register 4-2, 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 the upper three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, `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 23-2.

Note: The C and DC bits operate as a borrow and digit borrow bit respectively, in subtraction.

REGISTER 4-2: 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/borrow bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` 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 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/borrow bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` 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 borrow, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high 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

PIC18CXX8

4.13.1 RCON REGISTER

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

Note 1: If the BOREN configuration bit is set, $\overline{\text{BOR}}$ is '1' on Power-on Reset. If the BOREN configuration bit is clear, $\overline{\text{BOR}}$ is unknown on Power-on Reset.

The $\overline{\text{BOR}}$ status bit is a "don't care" and is not necessarily predictable if the brown-out circuit is disabled (the BOREN configuration bit is clear). $\overline{\text{BOR}}$ must then be set by the user and checked on subsequent RESETs to see if it is clear, indicating a brown-out has occurred.

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

REGISTER 4-3: RCON REGISTER

| R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-0 | R/W-0 |
|--------------------------|--------------------------|-----|------------------------|------------------------|------------------------|-------------------------|-------------------------|
| $\overline{\text{IPEN}}$ | $\overline{\text{LWRT}}$ | — | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| 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 **LWRT:** Long Write Enable bit
 1 = Enable TBLWT to internal program memory
 Once this bit is set, it can only be cleared by a $\overline{\text{POR}}$ or $\overline{\text{MCLR}}$ Reset
 0 = Disable TBLWT to internal program memory; TBLWT only to external program memory
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **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 a Brown-out Reset occurs)
- bit 3 **TO:** Watchdog Time-out Flag bit
 1 = After power-up, CLRWDT instruction, or SLEEP instruction
 0 = A WDT time-out occurred
- bit 2 **PD:** Power-down Detection Flag bit
 1 = After power-up or by the CLRWDT instruction
 0 = By execution of the SLEEP instruction
- bit 1 **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 **BOR:** Brown-out Reset Status bit
 1 = A Brown-out Reset has not occurred
 0 = A Brown-out Reset occurred
 (must be set in software after a Brown-out Reset occurs)

| | | | |
|--------------------|------------------|------------------------------------|--------------------|
| 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 |

5.0 TABLE READS/TABLE WRITES

All PICmicro® devices have two memory spaces: the 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 program memory spaces are:

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

Table Read operations retrieve data from program memory and place it into the data memory space. Figure 5-1 shows the operation of a Table Read with program and data memory.

Table Write operations store data from the data memory space into program memory. Figure 5-2 shows the operation of a Table Write with 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 will need to be word aligned.

FIGURE 5-1: TABLE READ OPERATION

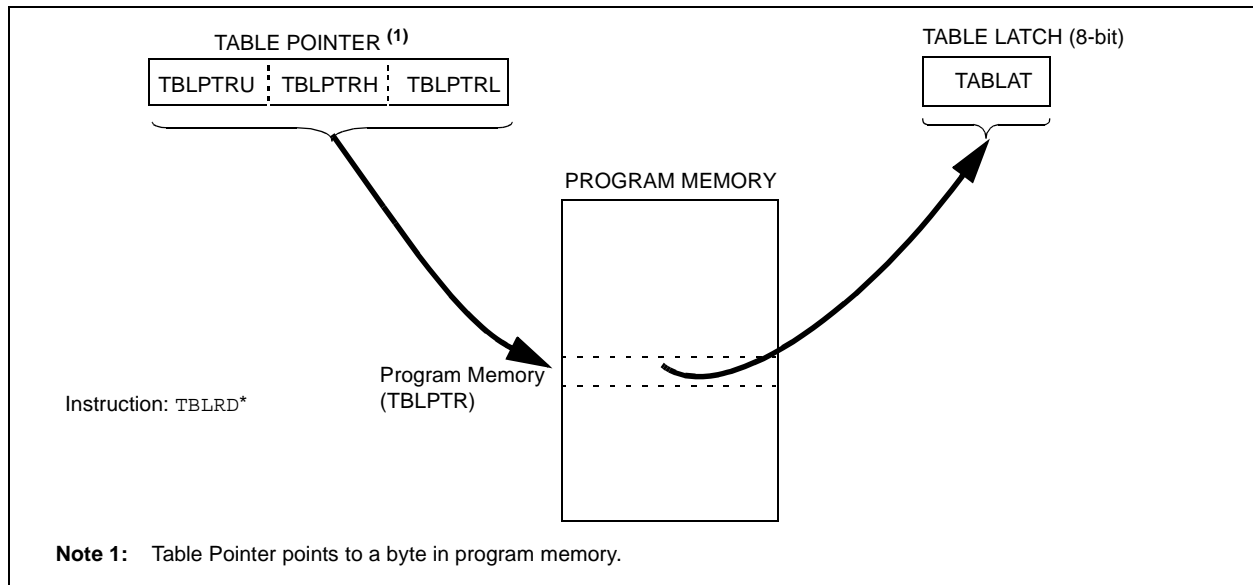
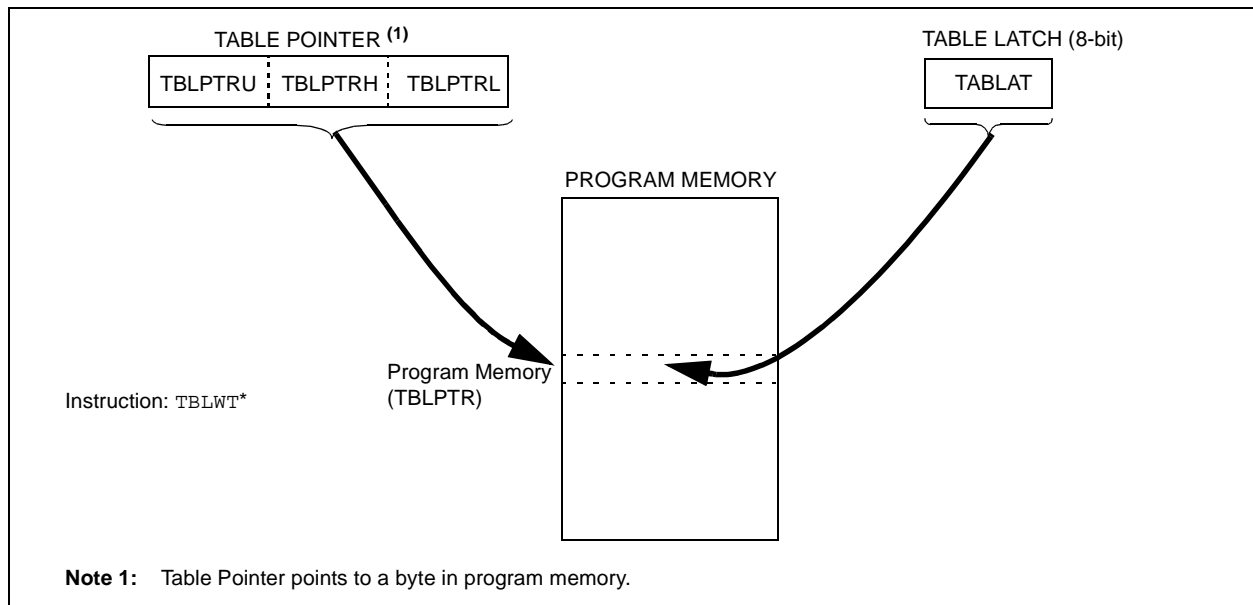


FIGURE 5-2: TABLE WRITE OPERATION



PIC18CXX8

5.1 Control Registers

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

- RCON register
- TABLAT register
- TBLPTR registers

5.1.1 RCON REGISTER

The LWRT bit specifies the operation of Table Writes to internal memory when the VPP voltage is applied to the MCLR pin. When the LWRT bit is set, the controller continues to execute user code, but long table writes are allowed (for programming internal program memory) from user mode. The LWRT bit can be cleared only by performing either a POR or MCLR Reset.

REGISTER 5-1: RCON REGISTER (ADDRESS: 0xFD0h)

| | | | | | | | |
|-------|-------|-----|-------|-------|-------|-------|-------|
| R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-0 | R/W-0 |
| IPEN | LWRT | — | RI | TO | PD | POR | BOR |
| | | | | | | bit 7 | bit 0 |

- bit 7 **IPEN:** Interrupt Priority Enable
 1 = Enable priority levels on interrupts
 0 = Disable priority levels on interrupts (16CXXX compatibility mode)
- bit 6 **LWRT:** Long Write Enable
 1 = Enable TBLWT to internal program memory
 0 = Disable TBLWT to internal program memory.
Note 1: Only cleared on a POR or MCLR reset.
 This bit has no effect on TBLWTs to external program memory.
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **RI:** RESET Instruction Flag bit
 1 = No RESET instruction occurred
 0 = A RESET instruction occurred
- bit 3 **TO:** Time-out bit
 1 = After power-up, CLRWDI instruction, or SLEEP instruction
 0 = A WDT time-out occurred
- bit 2 **PD:** Power-down bit
 1 = After power-up or by the CLRWDI instruction
 0 = By execution of the SLEEP instruction
- bit 1 **POR:** Power-on Reset Status bit
 1 = No Power-on Reset occurred
 0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)
- bit 0 **BOR:** Brown-out Reset Status bit
 1 = No Brown-out Reset nor POR Reset occurred
 0 = A Brown-out Reset or POR Reset occurred
 (must be set in software after a Brown-out Reset occurs)

| | | | |
|--------------------|------------------|------------------------------------|--------------------|
| 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 |

5.1.2 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.

5.1.3 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 22-bit wide pointer. The low order 21-bits allow the device to

address up to 2 Mbytes of program memory space. The 22nd bit allows read only access to the Device ID, the User ID and the Configuration bits.

The table pointer TBLPTR is used by the TBLRD and TBLWT instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 5-1. These operations on the TBLPTR only affect the low order 21-bits.

TABLE 5-1: 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 |

PIC18CXX8

5.2 Program Memory Read/Writes

5.2.1 TABLE READ OVERVIEW (TBLRD)

The TBLRD instructions are used to read data from program memory to data memory.

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

Table Reads from program memory are performed one byte at a time. The instruction will load TABLAT with the one byte from program memory pointed to by TBLPTR.

5.2.2 PROGRAM MEMORY WRITE BLOCK SIZE

The program memory of PIC18CXX8 devices is written in blocks. For PIC18CXX8 devices, the write block size is 2 bytes. Consequently, Table Write operations to program memory are performed in pairs, one byte at a time.

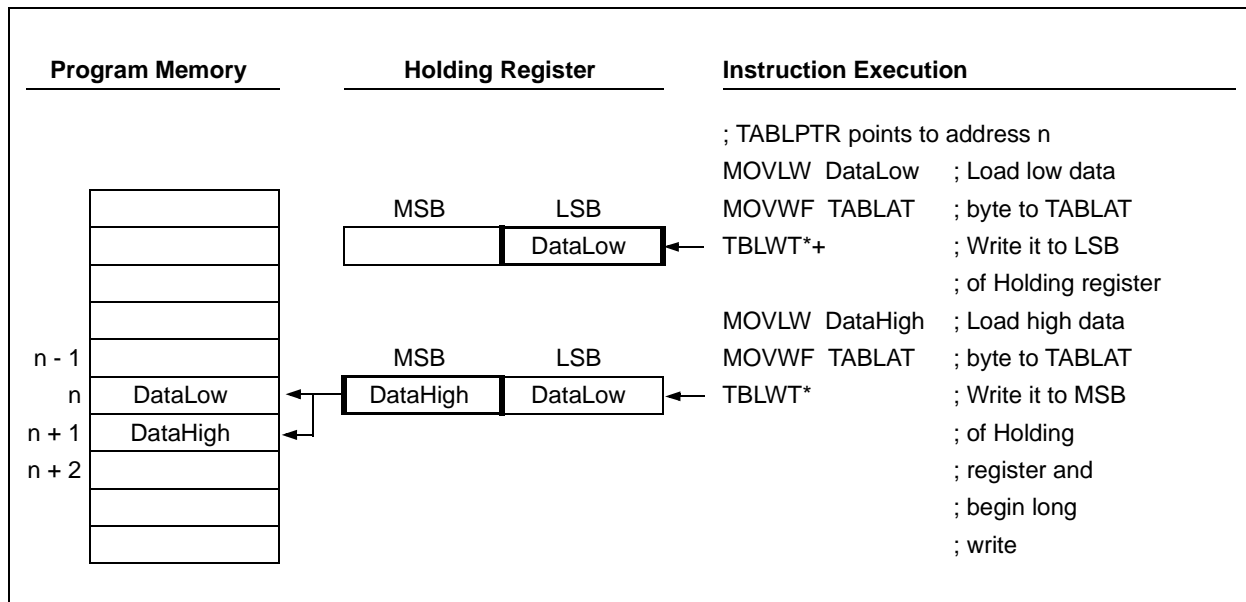
When a Table Write occurs to an even program memory address (TBLPTR<0> = 0), the contents of TABLAT are transferred to an internal holding register. This is performed as a short write and the program memory block is not actually programmed at this time. The holding register is not accessible by the user.

When a Table Write occurs to an odd program memory address (TBLPTR<0> = 1), a long write is started. During the long write, the contents of TABLAT are written to the high byte of the program memory block and the contents of the holding register are transferred to the low byte of the program memory block.

Figure 5-3 shows the holding register and the program memory write blocks.

If a single byte is to be programmed, the low (even) byte of the destination program word should be read using TBLRD*, modified or changed, if required, and written back to the same address using TBLWT*+. The high (odd) byte should be read using TBLRD*, modified or changed if required, and written back to the same address using TBLWT. The write to an odd address will cause a long write to begin. This process ensures that existing data in either byte will not be changed unless desired.

FIGURE 5-3: HOLDING REGISTER AND THE WRITE



EXAMPLE 5-1: TABLE READ CODE EXAMPLE

```

; Read a byte from location 0x0020
CLRF  TBLPTRU      ; Load upper 5 bits of
                  ; 0x0020
CLRF  TBLPTRH      ; Load higher 8 bits of
                  ; 0x0020
MOVLW 0x20         ; Load 0x20 into
MOVWF  TBLPTRL     ; TBLPTRL
MOVWF  TBLRD*      ; Data is in TABLAT
    
```

5.2.2.1 Long Write Operation

The long write is what actually programs words of data into the internal memory. When a $\overline{\text{TBLWT}}$ to the MSB of the write block occurs, instruction execution is halted. During this time, programming voltage and the data stored in internal latches is applied to program memory.

For a long write to occur:

1. $\overline{\text{MCLR/VPP}}$ pin must be at the programming voltage
2. LWRT bit must be set
3. $\overline{\text{TBLWT}}$ to the address of the MSB of the write block

If the LWRT bit is clear, a short write will occur and program memory will not be changed. If the $\overline{\text{TBLWT}}$ is not to the MSB of the write block, then the programming phase is not initiated.

Setting the LWRT bit enables long writes when the $\overline{\text{MCLR}}$ pin is taken to VPP voltage. Once the LWRT bit is set, it can be cleared only by performing a POR or $\overline{\text{MCLR}}$ Reset.

To ensure that the memory location has been well programmed, a minimum programming time is required. The long write can be terminated after the programming time has expired by a RESET or an interrupt. Having only one interrupt source enabled to terminate the long write, ensures that no unintended interrupts will prematurely terminate the long write.

5.2.2.2 Sequence of Events

The sequence of events for programming an internal program memory location should be:

1. Enable the interrupt that terminates the long write. Disable all other interrupts.
2. Clear the source interrupt flag.
3. If Interrupt Service Routine execution is desired when the device wakes, enable global interrupts.
4. Set LWRT bit in the RCON register.
5. Raise $\overline{\text{MCLR/VPP}}$ pin to the programming voltage, VPP.
6. Clear the WDT (if enabled).
7. Set the interrupt source to interrupt at the required time.
8. Execute the Table Write for the lower (even) byte. This will be a short write.
9. Execute the Table Write for the upper (odd) byte. This will be a long write. The controller will HALT while programming. The interrupt wakes the controller.
10. If GIE was set, service the interrupt request.
11. Go to 7 if more bytes to be programmed.
12. Lower $\overline{\text{MCLR/VPP}}$ pin to VDD.
13. Verify the memory location (table read).
14. Reset the device.

5.2.3 LONG WRITE INTERRUPTS

The long write must be terminated by a RESET or any interrupt.

The interrupt source must have its interrupt enable bit set. When the source sets its interrupt flag, programming will terminate. This will occur regardless of the settings of interrupt priority bits, the GIE/GIEH bit or the PIE/GIEL bit.

Depending on the states of interrupt priority bits, the GIE/GIEH bit or the PIE/GIEL bit, program execution can either be vectored to the high or low priority Interrupt Service Routine (ISR), or continue execution from where programming commenced.

In either case, the interrupt flag will not be cleared when programming is terminated and will need to be cleared by the software.

5.3 Unexpected Termination of Write Operations

If a write is terminated by an unplanned event such as loss of power, an unexpected RESET, or an interrupt that was not disabled, the memory location just programmed should be verified and reprogrammed if needed.

TABLE 5-2: SLEEP MODE, INTERRUPT ENABLE BITS AND INTERRUPT RESULTS

| GIE/GIEH | PIE/GIEL | Priority | Interrupt Enable | Interrupt Flag | Action |
|-------------|-------------|---------------------------|------------------|----------------|----------------------------------------------------------------------------------------------------------|
| X | X | X | 0 (default) | X | Long write continues even if interrupt flag becomes set during SLEEP. |
| X | X | X | 1 | 0 | Long write continues, will wake when the interrupt flag is set. |
| 0 (default) | 0 (default) | X | 1 | 1 | Terminates long write, executes next instruction. Interrupt flag not cleared. |
| 0 (default) | 1 | 1 high priority (default) | 1 | 1 | Terminates long write, executes next instruction. Interrupt flag not cleared. |
| 1 | 0 (default) | 0 low | 1 | 1 | Terminates long write, executes next instruction. Interrupt flag not cleared. |
| 0 (default) | 1 | 0 low | 1 | 1 | Terminates long write, branches to low priority interrupt vector. Interrupt flag can be cleared by ISR. |
| 1 | 0 (default) | 1 high priority (default) | 1 | 1 | Terminates long write, branches to high priority interrupt vector. Interrupt flag can be cleared by ISR. |

6.0 8 X 8 HARDWARE MULTIPLIER

An 8 x 8 hardware multiplier is included in the ALU of the PIC18CXX8 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 applications previously reserved for Digital Signal Processors.

Table 6-1 shows a performance comparison between enhanced devices using the single cycle hardware multiply, and performing the same function without the hardware multiply.

TABLE 6-1: PERFORMANCE COMPARISON

| Routine | Multiply Method | Program Memory (Words) | Cycles (Max) | Time | | |
|------------------|---------------------------|------------------------|--------------|--------------|---------------|-------------|
| | | | | @ 40 MHz | @ 10 MHz | @ 4 MHz |
| 8 x 8 unsigned | Without hardware multiply | 13 | 69 | 6.9 μ s | 27.6 μ s | 69 μ s |
| | Hardware multiply | 1 | 1 | 100 ns | 400 ns | 1 μ s |
| 8 x 8 signed | Without hardware multiply | 33 | 91 | 9.1 μ s | 36.4 μ s | 91 μ s |
| | Hardware multiply | 6 | 6 | 600 ns | 2.4 μ s | 6 μ s |
| 16 x 16 unsigned | Without hardware multiply | 21 | 242 | 24.2 μ s | 96.8 μ s | 242 μ s |
| | Hardware multiply | 24 | 24 | 2.4 μ s | 9.6 μ s | 24 μ s |
| 16 x 16 signed | Without hardware multiply | 52 | 254 | 25.4 μ s | 102.6 μ s | 254 μ s |
| | Hardware multiply | 36 | 36 | 3.6 μ s | 14.4 μ s | 36 μ s |

PIC18CXX8

6.1 Operation

Example 6-1 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 6-2 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 6-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

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

EXAMPLE 6-2: 8 x 8 SIGNED MULTIPLY ROUTINE

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

BTFSC ARG2, SB  ; Test Sign Bit
SUBWF PRODH, F  ; PRODH = PRODH
                ; - ARG1

MOVFF ARG2, WREG
BTFSC ARG1, SB  ; Test Sign Bit
SUBWF PRODH, F  ; PRODH = PRODH
                ; - ARG2
```

Example 6-3 shows the sequence to perform a 16 x 16 unsigned multiply. Equation 6-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers RES3:RES0.

EQUATION 6-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

```
RES3:RES0 = ARG1H:ARG1L • ARG2H:ARG2L
          = (ARG1H • ARG2H • 216) +
            (ARG1H • ARG2L • 28) +
            (ARG1L • ARG2H • 28) +
            (ARG1L • ARG2L)
```

EXAMPLE 6-3: 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, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;

MOVFF ARG1H, WREG ;
MULWF ARG2L      ; ARG1H * ARG2L ->
                ; PRODH:PRODL

MOVF PRODL, W    ;
ADDWF RES1, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
```


Example 6-4 shows the sequence to perform an 16 x 16 signed multiply. Equation 6-2 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 6-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

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

EXAMPLE 6-4: 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, F         ; Add cross
MOVF  PRODH, W        ;   products
ADDWFC RES2, F        ;
CLRF  WREG             ;
ADDWFC RES3, F        ;

;

MOVFF ARG1H, WREG    ;
MULWF ARG2L           ; ARG1H * ARG2L ->
                       ;   PRODH:PRODL

MOVF  PRODL, W        ;
ADDWF RES1, F         ; Add cross
MOVF  PRODH, W        ;   products
ADDWFC RES2, F        ;
CLRF  WREG             ;
ADDWFC RES3, F        ;

;

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
:

```

PIC18CXX8

NOTES:

7.0 INTERRUPTS

The PIC18CXX8 devices have multiple 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 13 registers that are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

It is recommended that the Microchip header files supplied with MPLAB 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 PICmicro® 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.

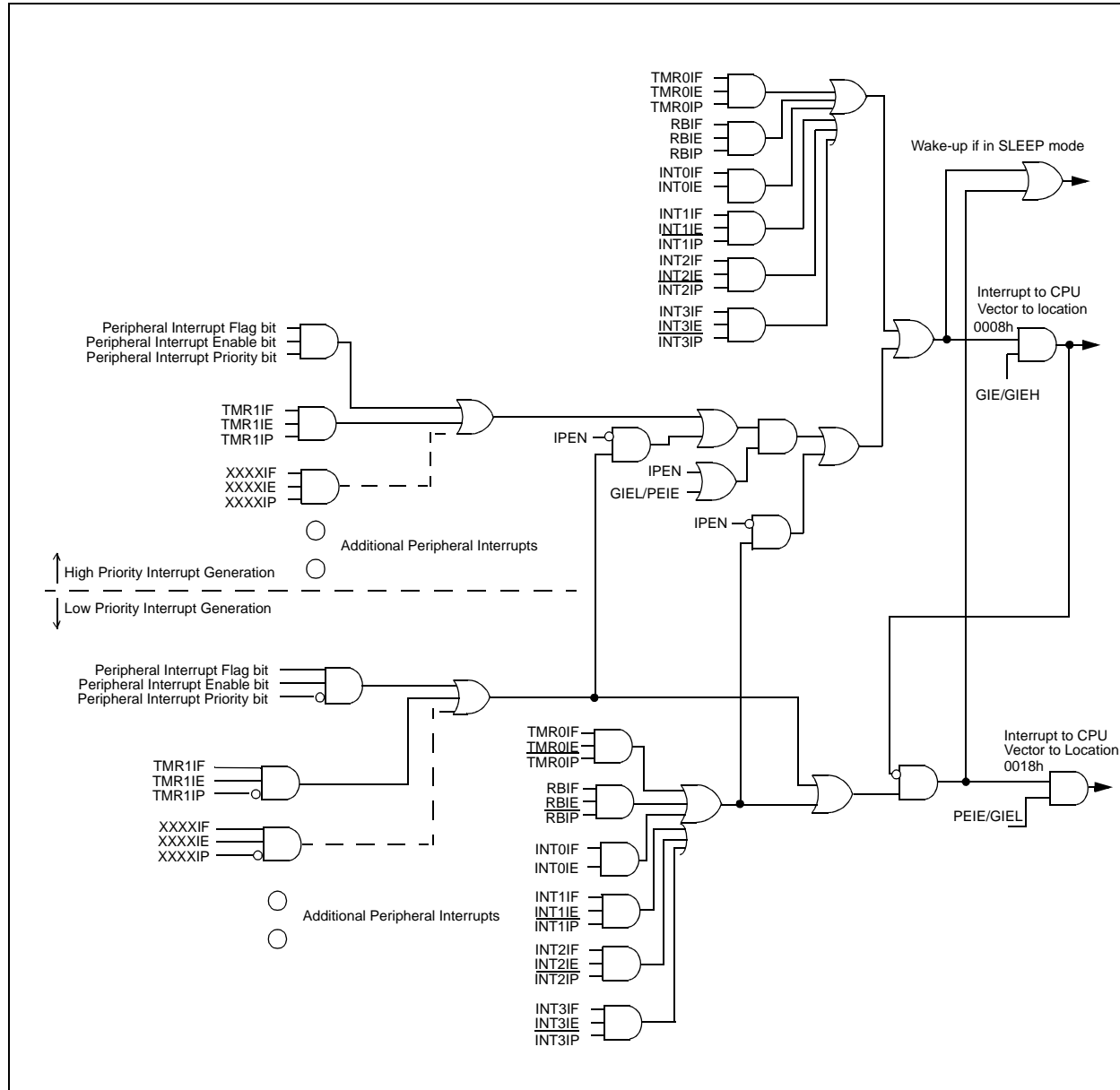
The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before 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.

PIC18CXX8

FIGURE 7-1: INTERRUPT LOGIC



7.1 Control Registers

This section contains the control and status registers.

7.1.1 INTCON REGISTERS

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

REGISTER 7-1: 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 un-masked 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 un-masked 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 by reading PORTB)
 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.

PIC18CXX8

REGISTER 7-2: INTCON2 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 |
| $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP |
| bit 7 | | | | | | | bit 0 |

- bit 7 **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 **INTEDG3**: External Interrupt 3 Edge Select bit
 1 = Interrupt on rising edge
 0 = Interrupt on falling edge
- bit 2 **TMR0IP**: TMR0 Overflow Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 1 **INT3IP**: INT3 External Interrupt Priority bit
 1 = High priority
 0 = Low priority
- 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 7-3: INTCON3 REGISTER

| R/W-1 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|--------|--------|--------|--------|--------|--------|--------|--------|
| INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | 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 **INT3IE:** INT3 External Interrupt Enable bit
 1 = Enables the INT3 external interrupt
 0 = Disables the INT3 external interrupt
- 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 **INT3IF:** INT3 External Interrupt Flag bit
 1 = The INT3 external interrupt occurred
 (must be cleared in software)
 0 = The INT3 external interrupt did not occur
- 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.

PIC18CXX8

7.1.2 PIR REGISTERS

The Peripheral Interrupt Request (PIR) registers contain the individual flag bits for the peripheral interrupts (Register 7-5). Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Request (Flag) registers (PIR1, PIR2, PIR3).

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.

7.1.3 PIE REGISTERS

The Peripheral Interrupt Enable (PIE) registers contain the individual enable bits for the peripheral interrupts (Register 7-5). Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2, PIE3). When IPEN is clear, the PEIE bit must be set to enable any of these peripheral interrupts.

7.1.4 IPR REGISTERS

The Interrupt Priority (IPR) registers contain the individual priority bits for the peripheral interrupts (Register 7-7). Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Priority registers (IPR1, IPR2, IPR3). The operation of the priority bits requires that the Interrupt Priority Enable bit (IPEN) be set.

7.1.5 RCON REGISTER

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

REGISTER 7-4: RCON REGISTER

| | | | | | | | | |
|-------|-------|-----|-----------------|-----------------|-----------------|------------------|------------------|-------|
| R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-0 | R/W-0 | |
| IPEN | LWRT | — | \overline{RI} | \overline{TO} | \overline{PD} | \overline{POR} | \overline{BOR} | |
| 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 **LWRT:** Long Write Enable
For details of bit operation see Register 4-3
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **\overline{RI} :** RESET Instruction Flag bit
For details of bit operation see Register 4-3
- bit 3 **\overline{TO} :** Watchdog Time-out Flag bit
For details of bit operation see Register 4-3
- bit 2 **\overline{PD} :** Power-down Detection Flag bit
For details of bit operation see Register 4-3
- bit 1 **\overline{POR} :** Power-on Reset Status bit
For details of bit operation see Register 4-3
- bit 0 **\overline{BOR} :** Brown-out Reset Status bit
For details of bit operation see Register 4-3

| | | | |
|--------------------|------------------|------------------------------------|--------------------|
| 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 7-5: PIR REGISTERS

| | | | | | | | | |
|-------------|-------|-------|------|------|-------|--------|--------|--------|
| | R/W-0 | R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF |
| | bit 7 | | | | | | | bit 0 |

| | | | | | | | | |
|-------------|-------|-------|-----|-----|-------|-------|--------|--------|
| | U-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| PIR2 | — | CMIF | — | — | BCLIF | LVDIF | TMR3IF | CCP2IF |
| | bit 7 | | | | | | | bit 0 |

| | | | | | | | | |
|-------------|-------|-------|-------|--------|--------|--------|--------|--------|
| | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| PIR3 | IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF | TXB0IF | RXB1IF | RXB0IF |
| | bit 7 | | | | | | | bit 0 |

| | | |
|-------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIR1 | bit 7 | PSPIF: Parallel Slave Port Read/Write Interrupt Flag bit 1 = A read or a write operation has taken place (must be cleared in software) 0 = No read or write has occurred |
| | 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 <u>Capture Mode</u> 1 = A TMR1 register capture occurred (must be cleared in software) 0 = No TMR1 register capture occurred <u>Compare Mode</u> 1 = A TMR1 register compare match occurred (must be cleared in software) 0 = No TMR1 register compare match occurred <u>PWM Mode</u> 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 |

PIC18CXX8

REGISTER 7-5: PIR REGISTERS (CONT'D)

| | | |
|------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIR2 | bit 7 | Unimplemented: Read as '0' |
| | bit 6 | CMIF: Comparator Interrupt Flag bit 1 = Comparator input has changed 0 = Comparator input has not changed |
| | bit 5-4 | 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 <u>Capture Mode</u> 1 = A TMR1 register capture occurred (must be cleared in software) 0 = No TMR1 register capture occurred <u>Compare Mode</u> 1 = A TMR1 register compare match occurred (must be cleared in software) 0 = No TMR1 register compare match occurred <u>PWM Mode</u> Unused in this mode |

REGISTER 7-5: PIR REGISTERS (CONT'D)

| | | |
|-------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIR3 | bit 7 | IRXIF: Invalid Message Received Interrupt Flag bit 1 = An invalid message has occurred on the CAN bus 0 = An invalid message has not occurred on the CAN bus |
| | bit 6 | WAKIF: Bus Activity Wake-up Interrupt Flag bit 1 = Activity on the CAN bus has occurred 0 = Activity on the CAN bus has not occurred |
| | bit 5 | ERRIF: CAN Bus Error Interrupt Flag bit 1 = An error has occurred in the CAN module (multiple sources) 0 = An error has not occurred in the CAN module |
| | bit 4 | TXB2IF: Transmit Buffer 2 Interrupt Flag bit 1 = Transmit Buffer 2 has completed transmission of a message, and may be reloaded 0 = Transmit Buffer 2 has not completed transmission of a message |
| | bit 3 | TXB1IF: Transmit Buffer 1 Interrupt Flag bit 1 = Transmit Buffer 1 has completed transmission of a message, and may be reloaded 0 = Transmit Buffer 1 has not completed transmission of a message |
| | bit 2 | TXB0IF: Transmit Buffer 0 Interrupt Flag bit 1 = Transmit Buffer 0 has completed transmission of a message, and may be reloaded 0 = Transmit Buffer 0 has not completed transmission of a message |
| | bit 1 | RXB1IF: Receive Buffer 1 Interrupt Flag bit 1 = Receive Buffer 1 has received a new message 0 = Receive Buffer 1 has not received a new message |
| | bit 0 | RXB0IF: Receive Buffer 0 Interrupt Flag bit 1 = Receive Buffer 0 has received a new message 0 = Receive Buffer 0 has not received a new message |

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 |

PIC18CXX8

REGISTER 7-6: PIE REGISTERS

| | | | | | | | | |
|-------------|-------|-------|-------|--------|--------|--------|--------|--------|
| | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |
| | bit 7 | | | | | | bit 0 | |
| | U-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| PIE2 | — | CMIE | — | — | BCLIE | LVDIE | TMR3IE | CCP2IE |
| | bit 7 | | | | | | bit 0 | |
| | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| PIE3 | IVRE | WAKIE | ERRIE | TXB2IE | TXB1IE | TXB0IE | RXB1IE | RXB0IE |
| | bit 7 | | | | | | bit 0 | |

- PIE1**
- bit 7 **PSPIE**: Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt
 - 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

REGISTER 7-6: PIE REGISTERS (CONT'D)

| | | |
|-------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIE2 | bit 7 | Unimplemented: Read as '0' |
| | bit 6 | CMIE: Comparator Interrupt Enable bit 1 = Enables the comparator interrupt 0 = Disables the comparator interrupt |
| | bit 5-4 | 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 |
| | PIE3 | bit 7 |
| bit 6 | | WAKIE: Bus Activity Wake-up Interrupt Enable bit 1 = Enables the Bus Activity Wake-Up Interrupt 0 = Disables the Bus Activity Wake-Up Interrupt |
| bit 5 | | ERRIE: CAN Bus Error Interrupt Enable bit 1 = Enables the CAN Bus Error Interrupt 0 = Disables the CAN Bus Error Interrupt |
| bit 4 | | TXB2IE: Transmit Buffer 2 Interrupt Enable bit 1 = Enables the Transmit Buffer 2 Interrupt 0 = Disables the Transmit Buffer 2 Interrupt |
| bit 3 | | TXB1IE: Transmit Buffer 1 Interrupt Enable bit 1 = Enables the Transmit Buffer 1 Interrupt 0 = Disables the Transmit Buffer 1 Interrupt |
| bit 2 | | TXB0IE: Transmit Buffer 0 Interrupt Enable bit 1 = Enables the Transmit Buffer 0 Interrupt 0 = Disables the Transmit Buffer 0 Interrupt |
| bit 1 | | RXB1IE: Receive Buffer 1 Interrupt Enable bit 1 = Enables the Receive Buffer 1 Interrupt 0 = Disables the Receive Buffer 1 Interrupt |
| bit 0 | | RXB0IE: Receive Buffer 0 Interrupt Enable bit 1 = Enables the Receive Buffer 0 Interrupt 0 = Disables the Receive Buffer 0 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 |

PIC18CXX8

REGISTER 7-7: IPR REGISTERS

| | | | | | | | | |
|-------------|-------|-------|-------|-------|-------|--------|--------|--------|
| | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP |
| | bit 7 | | | | | | bit 0 | |

| | | | | | | | | |
|-------------|-------|-------|-----|-----|-------|-------|--------|--------|
| | U-0 | R/W-1 | U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| IPR2 | — | CMIP | — | — | BCLIP | LVDIP | TMR3IP | CCP2IP |
| | bit 7 | | | | | | bit 0 | |

| | | | | | | | | |
|-------------|-------|-------|-------|--------|--------|--------|--------|--------|
| | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| IPR3 | IVRP | WAKIP | ERRIP | TXB2IP | TXB1IP | TXB0IP | RXB1IP | RXB0IP |
| | bit 7 | | | | | | bit 0 | |

- IPR1**
- bit 7 **PSPIP**: Parallel Slave Port Read/Write Interrupt Priority bit
1 = High priority
0 = Low priority
 - 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

REGISTER 7-7: IPR REGISTERS (CONT'D)

| | | |
|-------------|-------------|----------------------------------------------------------------------------------------------------|
| IPR2 | bit 7 | Unimplemented: Read as '0' |
| | bit 6 | CMIP: Comparator Interrupt Priority bit 1 = High priority 0 = Low priority |
| | bit 5-4 | 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 |
| | IPR3 | bit 7 |
| bit 6 | | WAKIP: Bus Activity Wake-up Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 5 | | ERRIP: CAN Bus Error Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 4 | | TXB2IP: Transmit Buffer 2 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 3 | | TXB1IP: Transmit Buffer 1 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 2 | | TXB0IP: Transmit Buffer 0 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 1 | | RXB1IP: Receive Buffer 1 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 0 | | RXB0IP: Receive Buffer 0 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 |

7.1.6 INT INTERRUPTS

External interrupts on the RB0/INT0, RB1/INT1, RB2/INT2, and RB3/INT3 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, INT2, and INT3) 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, INT2 and INT3 is determined by the value contained in the interrupt priority bits INT1IP (INTCON3 register), INT3IP (INTCON3 register), and INT2IP (INTCON2 register). There is no priority bit associated with INT0; it is always a high priority interrupt source.

7.1.7 TMR0 INTERRUPT

In 8-bit mode (which is the default), an overflow (FFh → 00h) in the TMR0 register will set flag bit TMR0IF. In 16-bit mode, an overflow (FFFFh → 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.

7.1.8 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).

7.2 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 7-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 7-1: 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
```


8.0 I/O PORTS

Depending on the device selected, there are up to eleven 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.

8.1 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 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'.

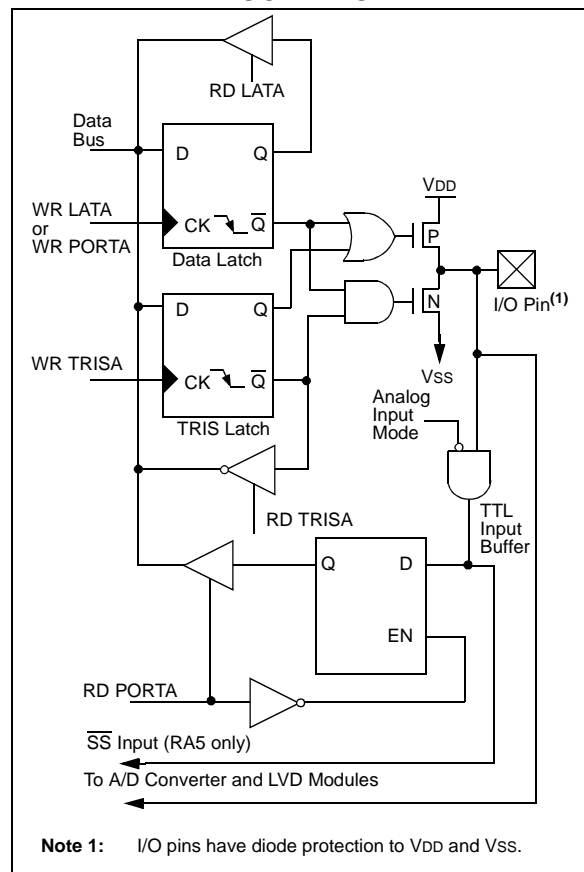
The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 8-1: INITIALIZING PORTA

```

CLRFB   PORTA    ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRFB   LATA     ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0x07     ; Configure A/D
MOVWF   ADCON1  ; for digital inputs
MOVLW   0xCF     ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISA    ; Set RA3:RA0 as inputs
                ; RA5:RA4 as outputs
    
```

FIGURE 8-1: RA3:RA0 AND RA5 PINS BLOCK DIAGRAM



PIC18CXX8

FIGURE 8-2: RA4/T0CKI PIN BLOCK DIAGRAM

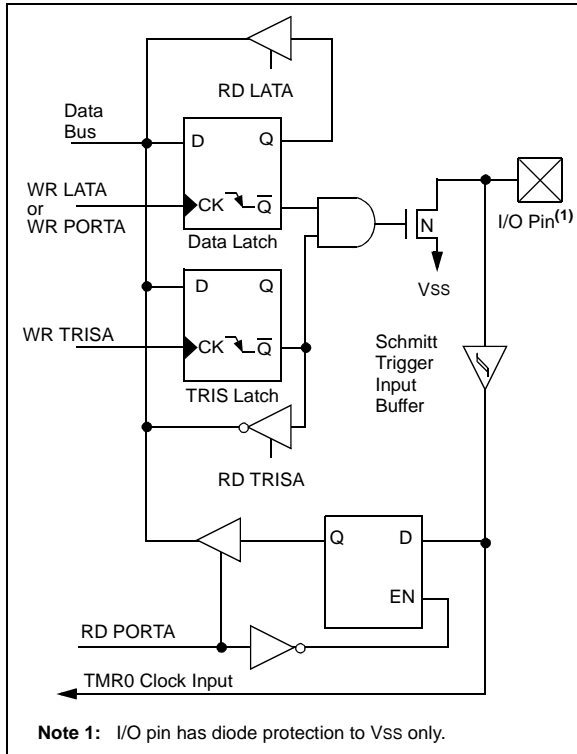


FIGURE 8-3: RA6 BLOCK DIAGRAM

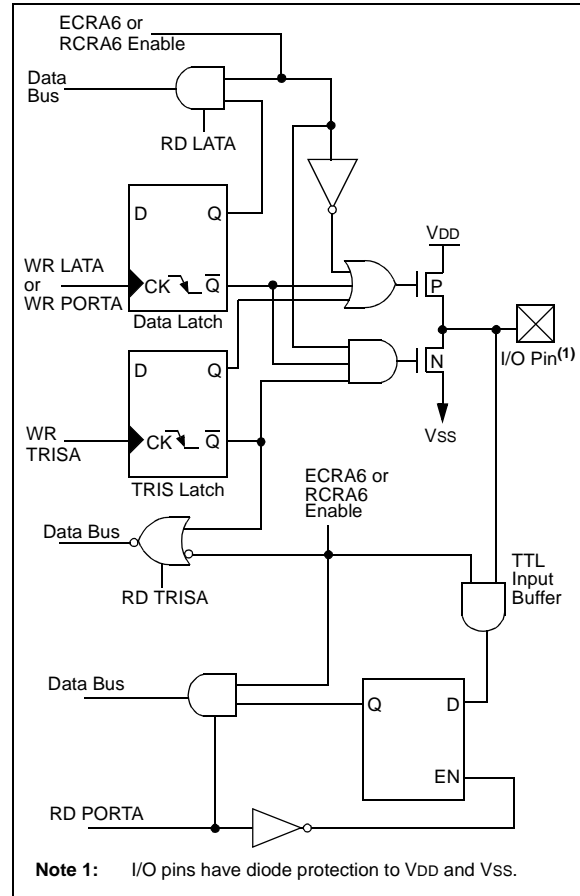


TABLE 8-1: PORTA FUNCTIONS

| 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/SS/AN4/LVDIN | bit5 | TTL | Input/output or slave select input for synchronous serial port or analog input, or low voltage detect input. |
| OSC2/CLKO/RA6 | bit6 | TTL | OSC2 or clock output or I/O pin. |

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

TABLE 8-2: 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 | — | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | -x0x 0000 | -uuu 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.

8.2 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).

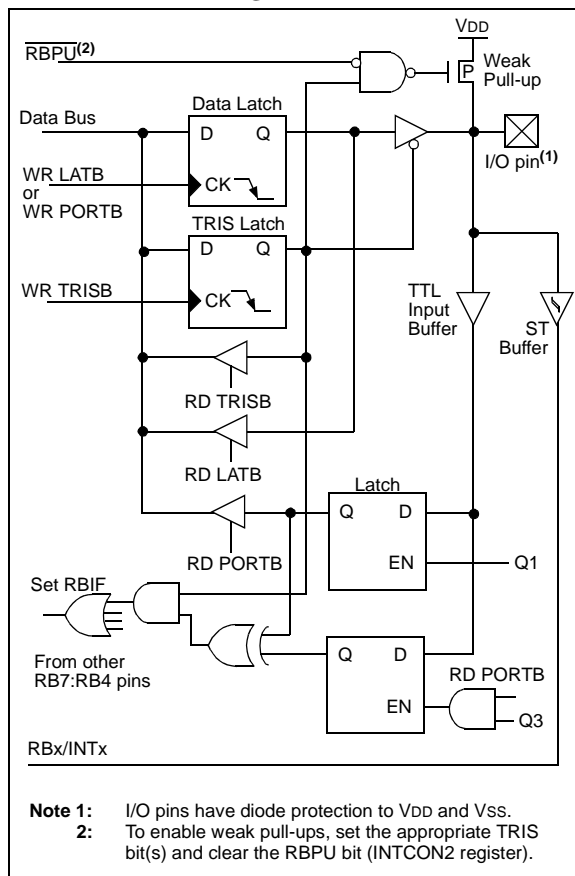
Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 8-2: INITIALIZING PORTB

```

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

FIGURE 8-4: RB7:RB4 PINS BLOCK DIAGRAM



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 RBPJ (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.

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).

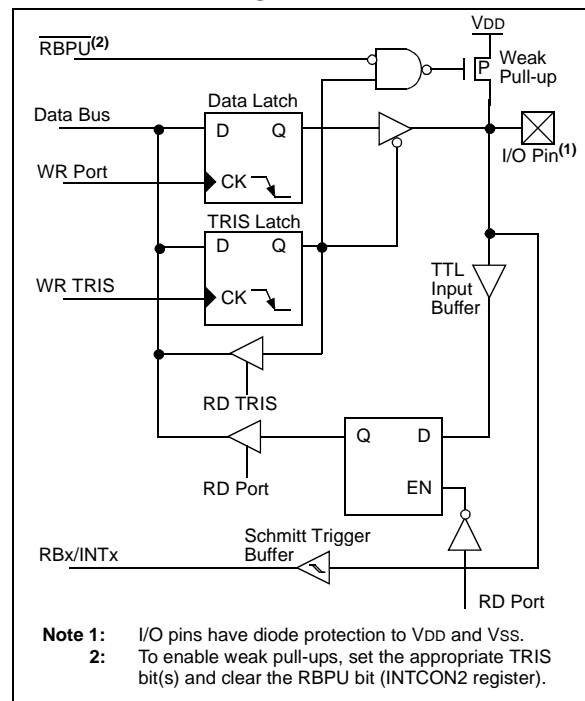
This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- Any read or write of PORTB (except with the MOVFF instruction). This will end the mismatch condition.
- 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 8-5: RB3:RB0 PINS BLOCK DIAGRAM



PIC18CXX8

TABLE 8-3: 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/INT3 | bit3 | TTL/ST ⁽¹⁾ | Input/output pin or external interrupt 3 input. 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 buffer is a Schmitt Trigger input when configured as the external interrupt.

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

TABLE 8-4: 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 | INT3IP | RBIP | 1111 1111 | 1111 1111 |
| INTCON3 | INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF | 1100 0000 | 1100 0000 |

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

8.3 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 8-5). 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 out-

put, 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.

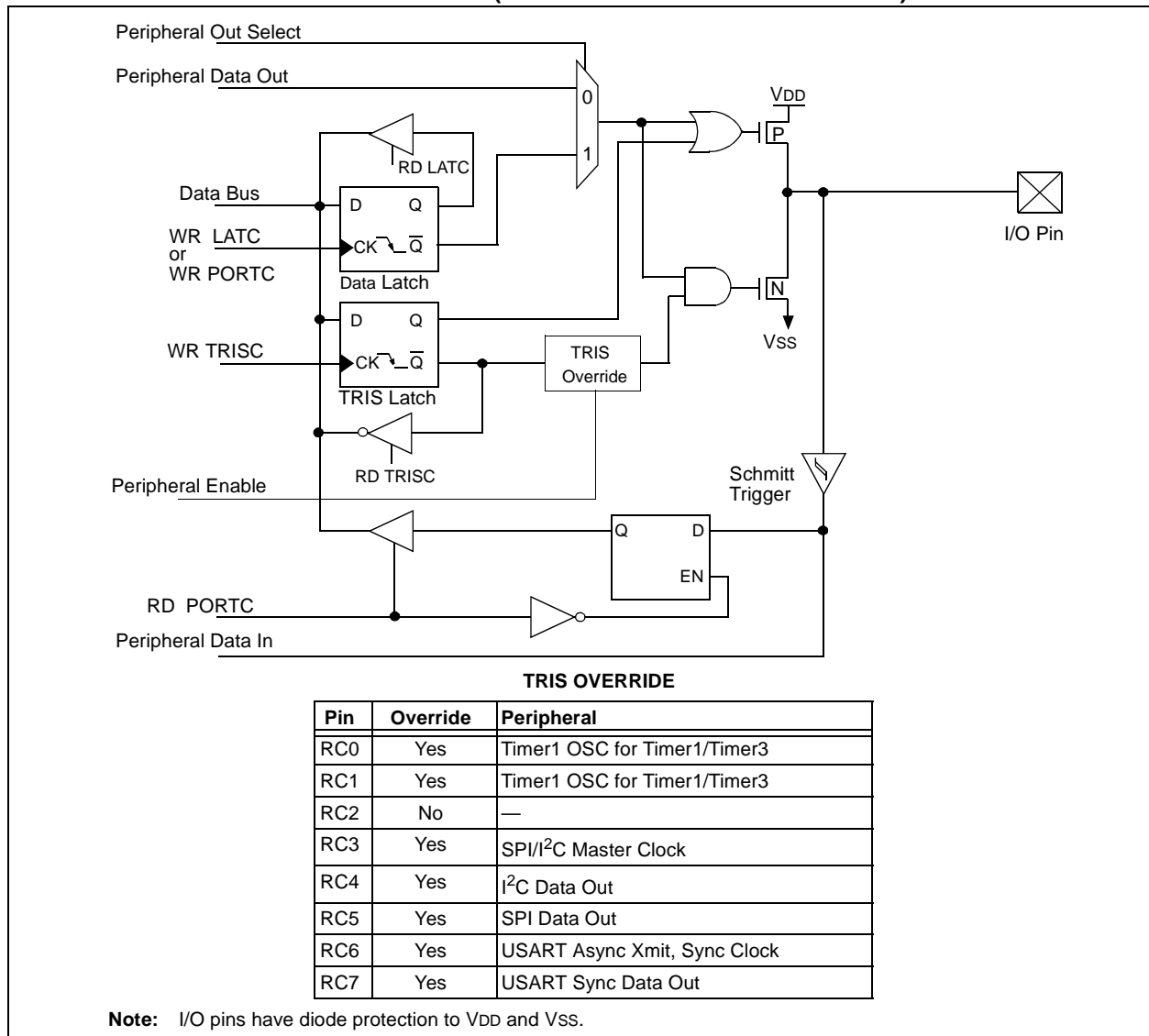
The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register, without concern due to peripheral overrides.

EXAMPLE 8-3: INITIALIZING PORTC

```

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

FIGURE 8-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE)



PIC18CXX8

TABLE 8-5: 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 or 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 ² C. |
| RC4/SDI/SDA | bit4 | ST | Input/output port pin or SPI Data in (SPI mode) or Data I/O (I ² C 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 8-6: 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

8.4 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.

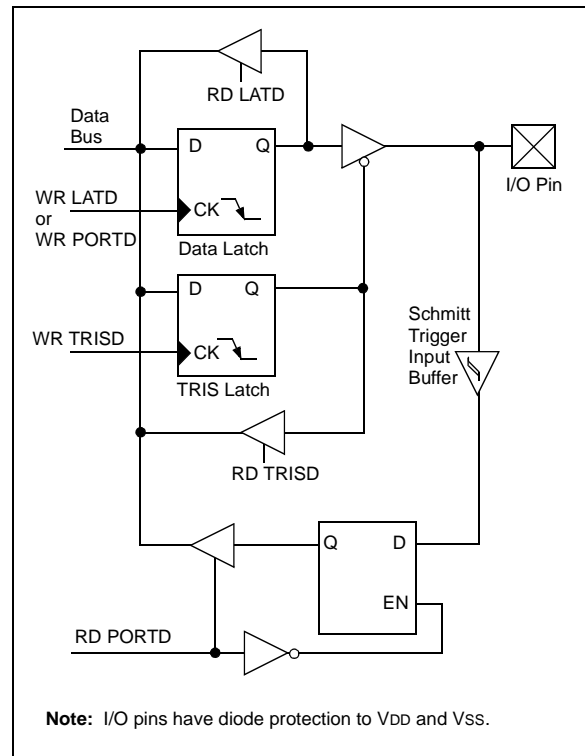
PORTD can be configured as an 8-bit wide micro-processor port (parallel slave port), by setting control bit PSPMODE (PSPCON register). In this mode, the input buffers are TTL. See Section 9.0 for additional information on the Parallel Slave Port (PSP).

EXAMPLE 8-4: INITIALIZING PORTD

```

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

FIGURE 8-7: PORTD BLOCK DIAGRAM IN I/O PORT MODE



PIC18CXX8

TABLE 8-7: PORTD FUNCTIONS

| Name | Bit# | Buffer Type | Function |
|----------|------|-----------------------|----------------------------------------------------|
| RD0/PSP0 | bit0 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit0. |
| RD1/PSP1 | bit1 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit1. |
| RD2/PSP2 | bit2 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit2. |
| RD3/PSP3 | bit3 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit3. |
| RD4/PSP4 | bit4 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit4. |
| RD5/PSP5 | bit5 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit5. |
| RD6/PSP6 | bit6 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit6. |
| RD7/PSP7 | bit7 | ST/TTL ⁽¹⁾ | Input/output port pin or parallel slave port bit7. |

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

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffer when in Parallel Slave Port mode.

TABLE 8-8: 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 |
| PSPCON | IBF | OBF | IBOV | PSPMODE | — | — | — | — | 0000 ---- | 0000 ---- |

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

8.5 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.

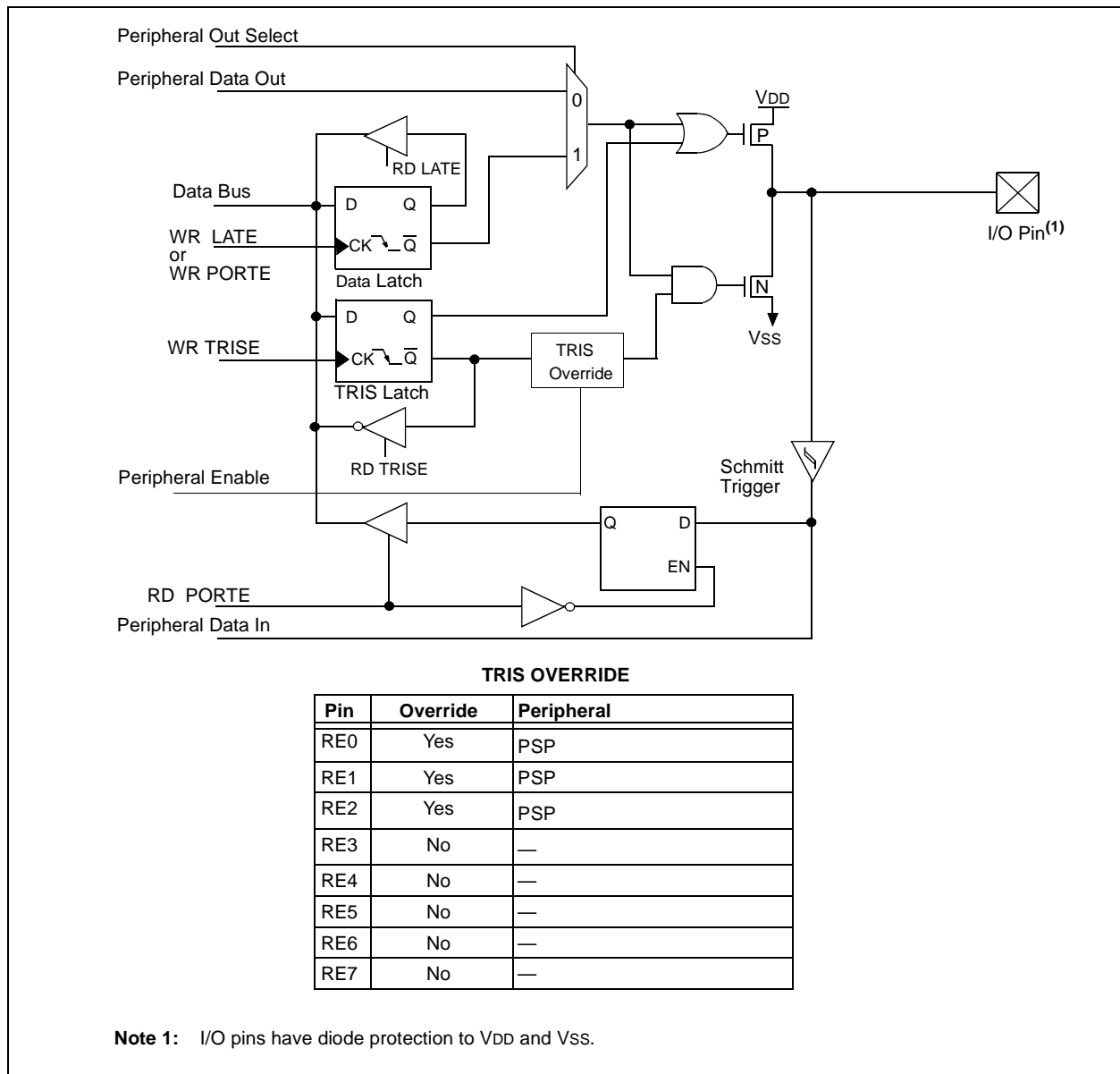
PORTE is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORTE is multiplexed with several peripheral functions (Table 8-9).

EXAMPLE 8-5: INITIALIZING PORTE

```

CLRFB  PORTE    ; Initialize PORTE by
              ; clearing output
              ; data latches
CLRFB  LATE     ; Alternate method
              ; to clear output
              ; data latches
MOVLW  0x03    ; Value used to
              ; initialize data
              ; direction
MOVWF  TRISE   ; Set RE1:RE0 as inputs
              ; RE7:RE2 as outputs
    
```

FIGURE 8-8: PORTE BLOCK DIAGRAM



PIC18CXX8

TABLE 8-9: PORTE FUNCTIONS

| Name | Bit# | Buffer Type | Function |
|----------------------|------|-----------------------|---------------------------------------------------------------------------------|
| RE0/ \overline{RD} | bit0 | ST/TTL ⁽¹⁾ | Input/output port pin or Read control input in Parallel Slave Port mode. |
| RE1/ \overline{WR} | bit1 | ST/TTL ⁽¹⁾ | Input/output port pin or Write control input in Parallel Slave Port mode. |
| RE2/ \overline{CS} | bit2 | ST/TTL ⁽¹⁾ | Input/output port pin or Chip Select control input in Parallel Slave Port mode. |
| RE3 | bit3 | ST | Input/output port pin. |
| RE4 | bit4 | ST | Input/output port pin. |
| RE5 | bit5 | ST | Input/output port pin. |
| RE6 | bit6 | ST | Input/output port pin. |
| RE7/CCP2 | bit7 | ST | Input/output port pin or Capture 2 input/Compare 2 output. |

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

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffer when in Parallel Slave Port mode.

TABLE 8-10: 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 |
| PSPCON | IBF | OBF | IBOV | PSPMODE | — | — | — | — | 0000 ---- | 0000 ---- |

Legend: x = unknown, u = unchanged

8.6 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 is multiplexed with several analog peripheral functions including the A/D converter inputs and comparator inputs, outputs, and voltage reference.

Note 1: On a Power-on Reset, the RF6:RF0 pins are configured as inputs and read as '0'.

2: To configure PORTF as digital I/O, turn off comparators and set ADCON1 value.

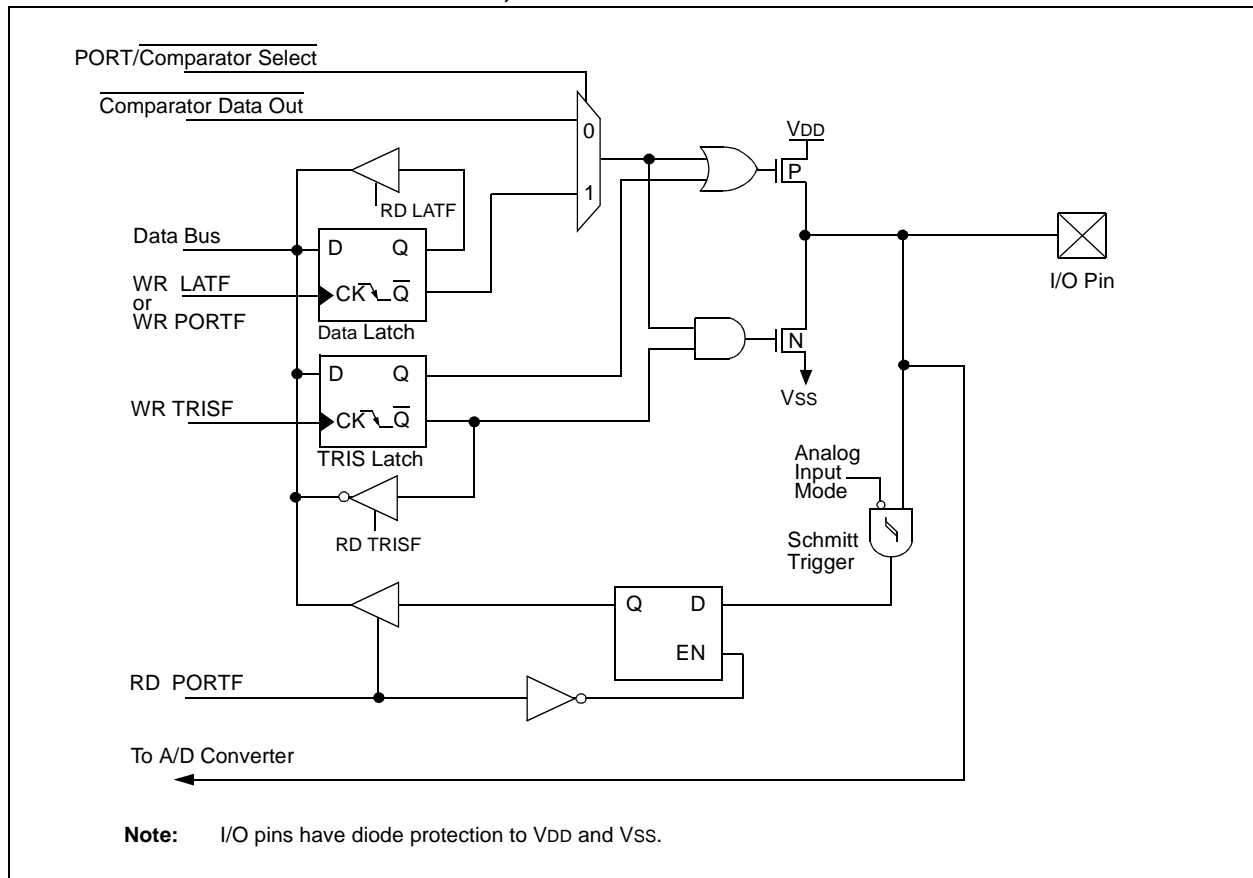
EXAMPLE 8-6: INITIALIZING PORTF

```

CLRFB   PORTF      ; Initialize PORTF by
                ; clearing output
                ; data latches
CLRFB   LATF       ; Alternate method
                ; to clear output
                ; data latches

MOVLW   0x07      ;
MOVWFB  CMCON      ; Turn off comparators
MOVLW   0x0F      ;
MOVWFB  ADCON1     ; Set PORTF as digital I/O
MOVLW   0xCF      ; Value used to
                ; initialize data
                ; direction
MOVWFB  TRISF      ; Set RF3:RF0 as inputs
                ; RF5:RF4 as outputs
                ; RF7:RF6 as inputs
    
```

FIGURE 8-9: PORTF RF1/AN6/C2OUT, RF2/AN5/C1OUT BLOCK DIAGRAM



PIC18CXX8

FIGURE 8-10: RF6:RF3 AND RF0 PINS BLOCK DIAGRAM

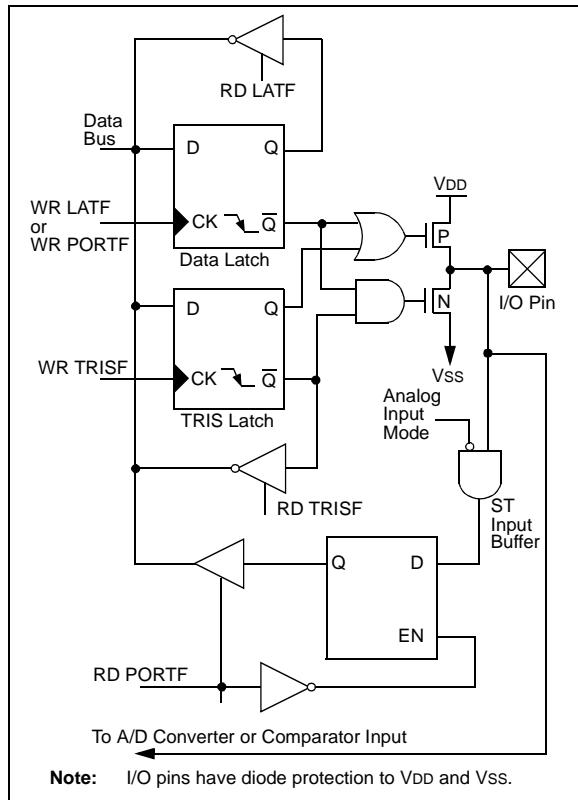


FIGURE 8-11: RF7 PIN BLOCK DIAGRAM

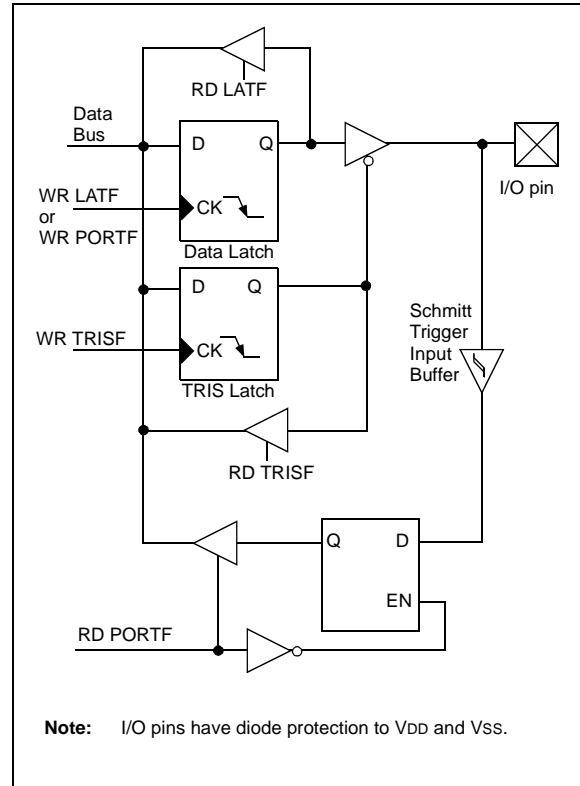


TABLE 8-11: PORTF FUNCTIONS

| Name | Bit# | Buffer Type | Function |
|--------------------|------|-------------|-------------------------------------------------------------------------------------------|
| RF0/AN5 | bit0 | ST | Input/output port pin or analog input. |
| RF1/AN6/C2OUT | bit1 | ST | Input/output port pin or analog input or comparator 2 output. |
| RF2/AN7/C1OUT | bit2 | ST | Input/output port pin or analog input or comparator 1 output. |
| RF3/AN8 | bit3 | ST | Input/output port pin or analog input or comparator input. |
| RF4/AN9 | bit4 | ST | Input/output port pin or analog input or comparator input. |
| RF5/AN10/ CVREF | bit5 | ST | Input/output port pin or analog input or comparator input or comparator reference output. |
| RF6/AN11 | bit6 | ST | Input/output port pin or analog input or comparator input. |
| RF7 | bit7 | ST | Input/output port pin. |

Legend: ST = Schmitt Trigger input

TABLE 8-12: 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 |
| CMCON | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 0000 0000 | 0000 0000 |

Legend: x = unknown, u = unchanged

8.7 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.

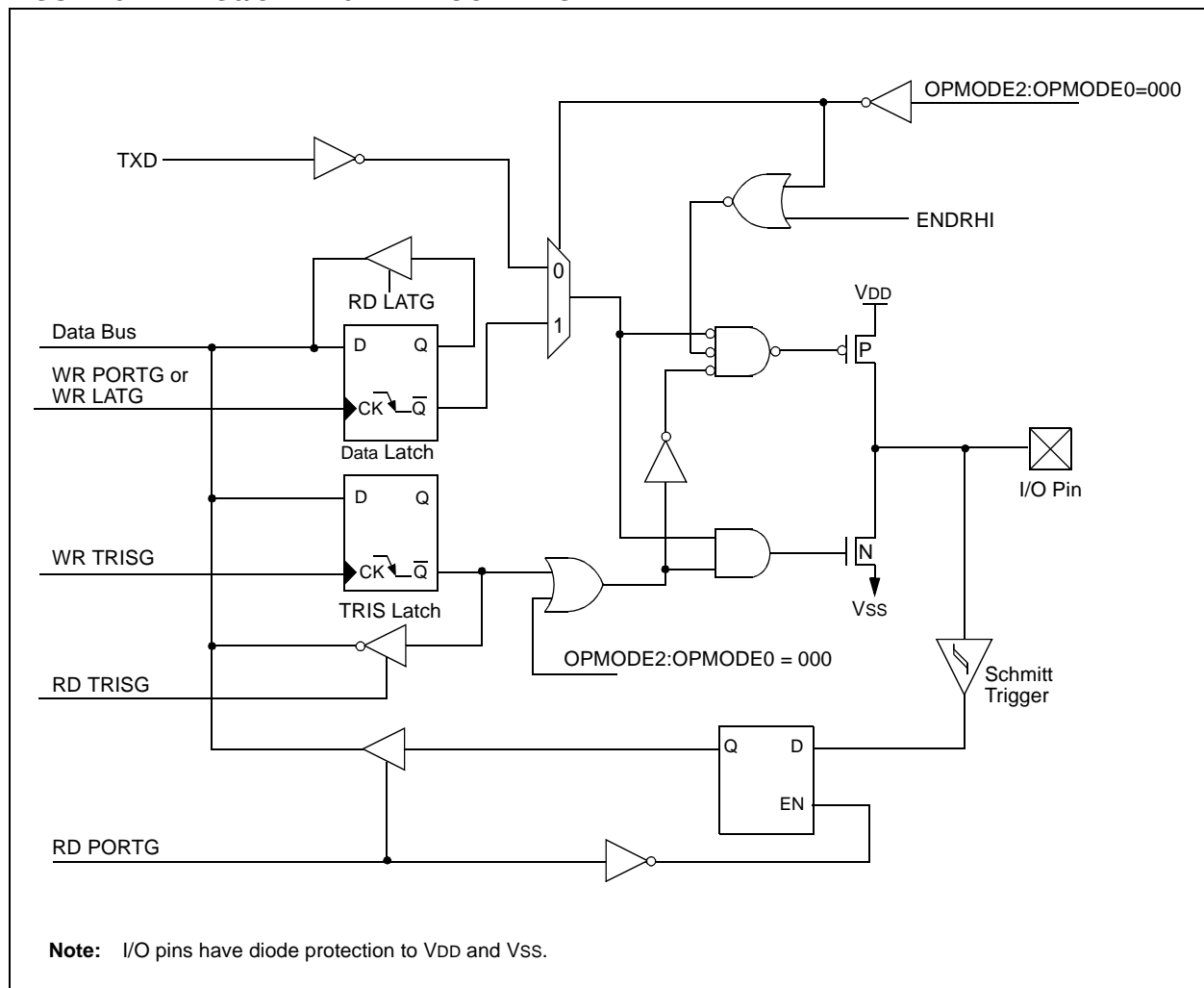
Pins RG0-RG2 on PORTG are multiplexed with the CAN peripheral. Refer to "CAN Module", Section 17.0 for proper settings of TRISG when CAN is enabled.

EXAMPLE 8-7: INITIALIZING PORTG

```

CLRF   PORTG   ; Initialize PORTG by
            ; clearing output
            ; data latches
CLRF   LATG    ; Alternate method
            ; to clear output
            ; data latches
MOVLW  0x04   ; Value used to
            ; initialize data
            ; direction
MOVWF  TRISG   ; Set RG1:RG0 as outputs
            ; RG2 as input
            ; RG4:RG3 as outputs
    
```

FIGURE 8-12: RG0/CANTX0 PIN BLOCK DIAGRAM



PIC18CXX8

FIGURE 8-13: RG1/CANTX1 PIN BLOCK DIAGRAM

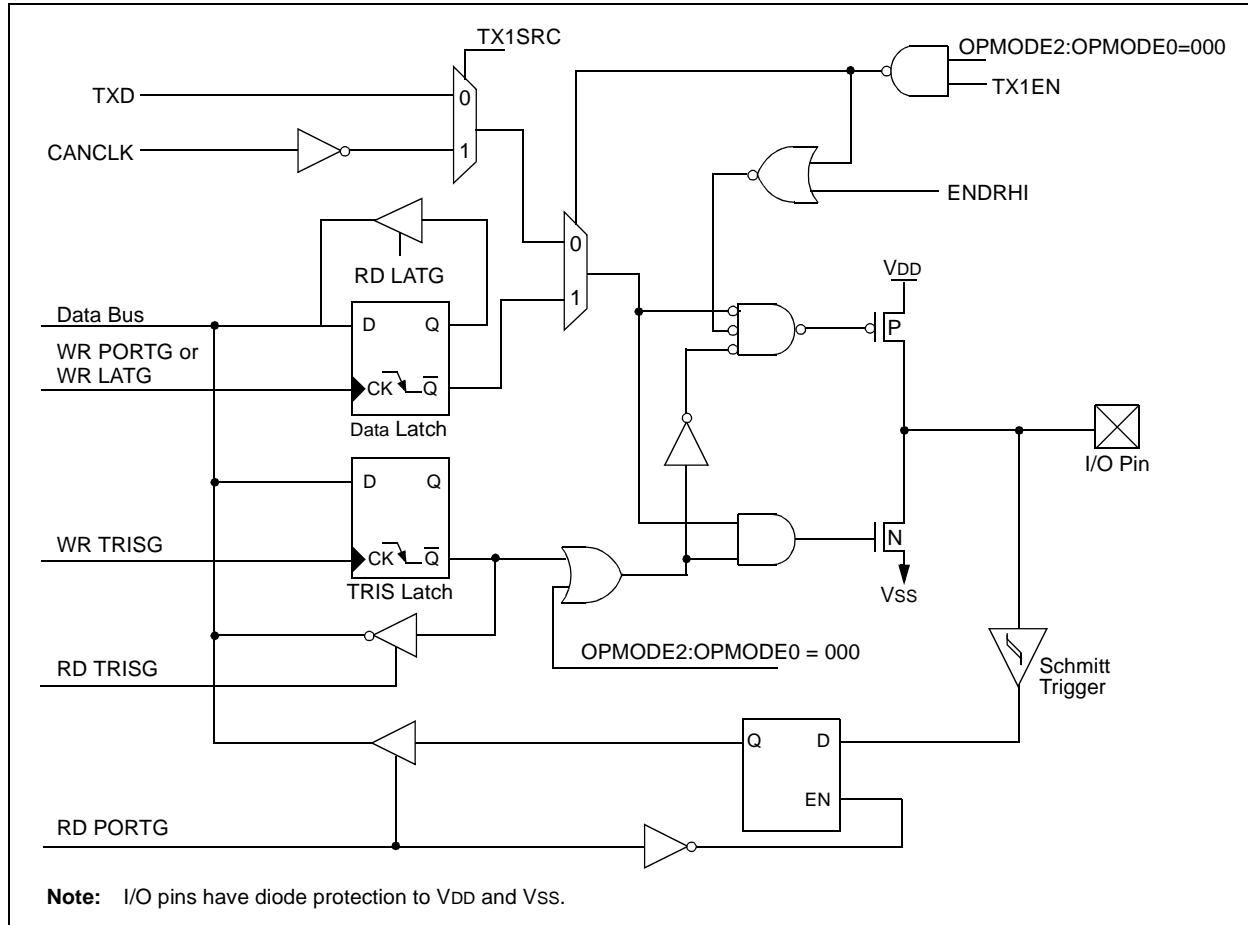


FIGURE 8-14: RG2/CANRX PIN BLOCK DIAGRAM

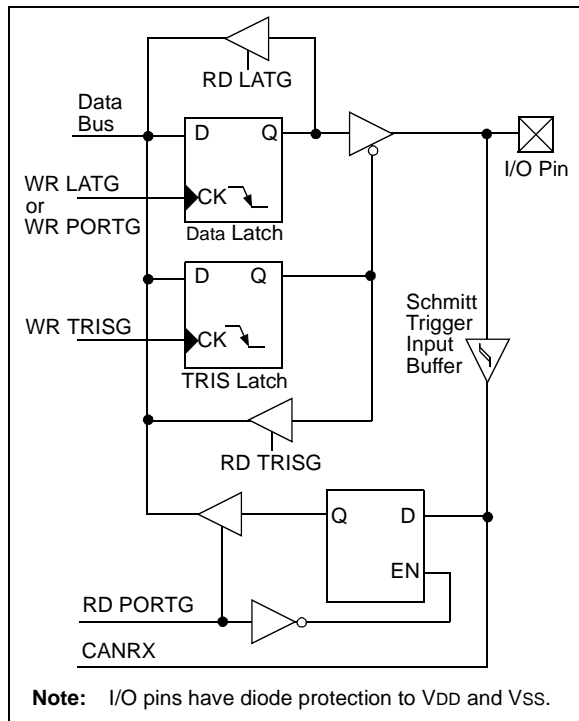


FIGURE 8-15: RG4:RG3 PINS BLOCK DIAGRAM

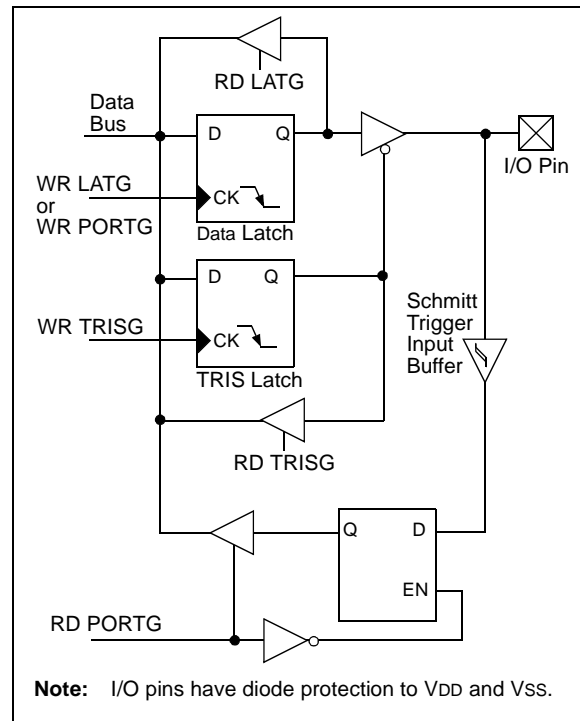


TABLE 8-13: PORTG FUNCTIONS

| Name | Bit# | Buffer Type | Function |
|------------|------|-------------|-------------------------------------------------------------------------------------------|
| RG0/CANTX0 | bit0 | ST | Input/output port pin or CAN bus transmit output. |
| RG1/CANTX1 | bit1 | ST | Input/output port pin or CAN bus complimentary transmit output or CAN bus bit time clock. |
| RG2/CANRX | bit2 | ST | Input/output port pin or CAN bus receive input. |
| RG3 | bit3 | ST | Input/output port pin. |
| RG4 | bit4 | ST | Input/output port pin. |

Legend: ST = Schmitt Trigger input

Note: Refer to "CAN Module", Section 17.0 for usage of CAN pin functions.

TABLE 8-14: 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 |
| CIOCON | TX1SRC | TX1EN | ENDRHI | CANCAP | — | — | — | — | 0000 ---- | 0000 ---- |

Legend: x = unknown, u = unchanged

PIC18CXX8

8.8 PORTH, LATH, and TRISH Registers

Note: This port is available on PIC18C858.

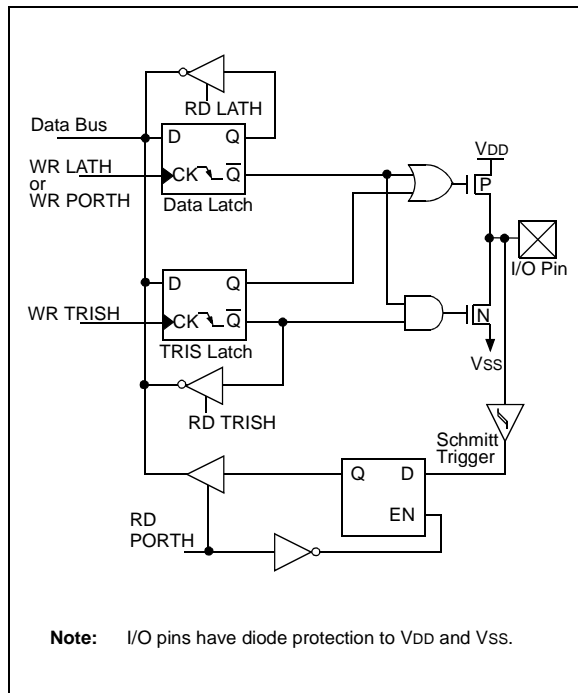
PORTH is a 5-bit wide, bi-directional port available only on the PIC18C858 devices. The corresponding Data Direction register is TRISH. Setting a TRISH bit (=1) will make the corresponding PORTH pin an input (i.e., put the corresponding output driver in a hi-impedance mode). Clearing a TRISH bit (=0) 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 RH0-RH3 on the PIC18C858 are bi-directional I/O pins with ST input buffers. Pins RH4-RH7 on all devices are multiplexed with A/D converter inputs.

Note: On a Power-on Reset, the RH7:RH4 pins are configured as inputs and read as '0'.

FIGURE 8-16: RH3:RH0 PINS BLOCK DIAGRAM



EXAMPLE 8-8: INITIALIZING PORTH

```

CLRF   PORTH   ; Initialize PORTH by
              ; clearing output
              ; data latches
CLRF   LATH    ; Alternate method
              ; to clear output
              ; data latches
MOVLW  0x0F   ;
MOVWF  ADCON1 ;
MOVLW  0xCF   ; Value used to
              ; initialize data
              ; direction
MOVWF  TRISH  ; Set RH3:RH0 as inputs
              ; RH5:RH4 as outputs
              ; RH7:RH6 as inputs
    
```

FIGURE 8-17: RH7:RH4 PINS BLOCK DIAGRAM

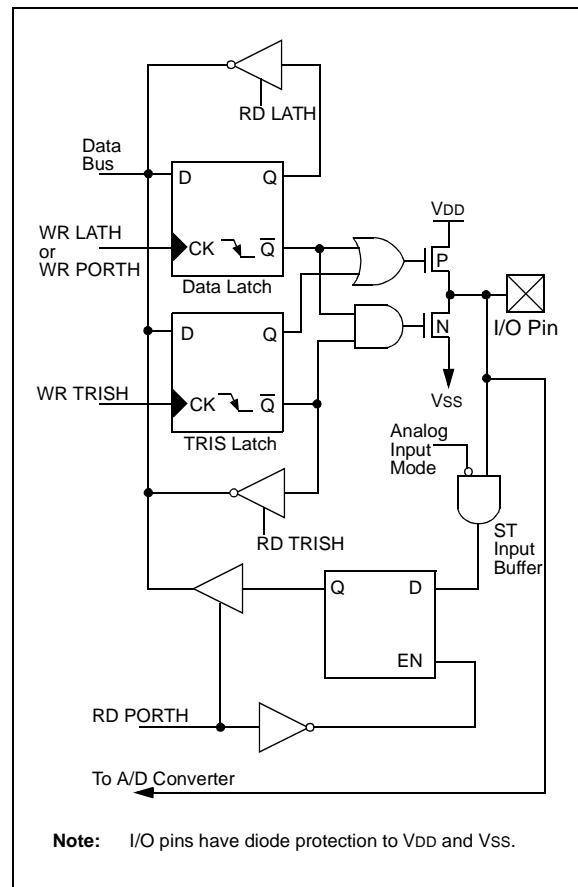


TABLE 8-15: PORTH FUNCTIONS

| Name | Bit# | Buffer Type | Function |
|----------|------|-------------|---------------------------------------------------|
| RH0 | bit0 | ST | Input/output port pin. |
| RH1 | bit1 | ST | Input/output port pin. |
| RH2 | bit2 | ST | Input/output port pin. |
| RH3 | bit3 | ST | Input/output port pin. |
| RH4/AN12 | bit4 | ST | Input/output port pin or analog input channel 12. |
| RH5/AN13 | bit5 | ST | Input/output port pin or analog input channel 13. |
| RH6/AN14 | bit6 | ST | Input/output port pin or analog input channel 14. |
| RH7/AN15 | bit7 | ST | Input/output port pin or analog input channel 15. |

Legend: ST = Schmitt Trigger input

TABLE 8-16: 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 |

Legend: x = unknown, u = unchanged, - = unimplemented

PIC18CXX8

8.9 PORTJ, LATJ, and TRISJ Registers

Note: This port is available on PIC18C858.

PORTJ is an 8-bit wide, bi-directional port available only on the PIC18C858 devices. 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 on the PIC18C858 is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

EXAMPLE 8-9: INITIALIZING PORTJ

```

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

FIGURE 8-18: PORTJ BLOCK DIAGRAM

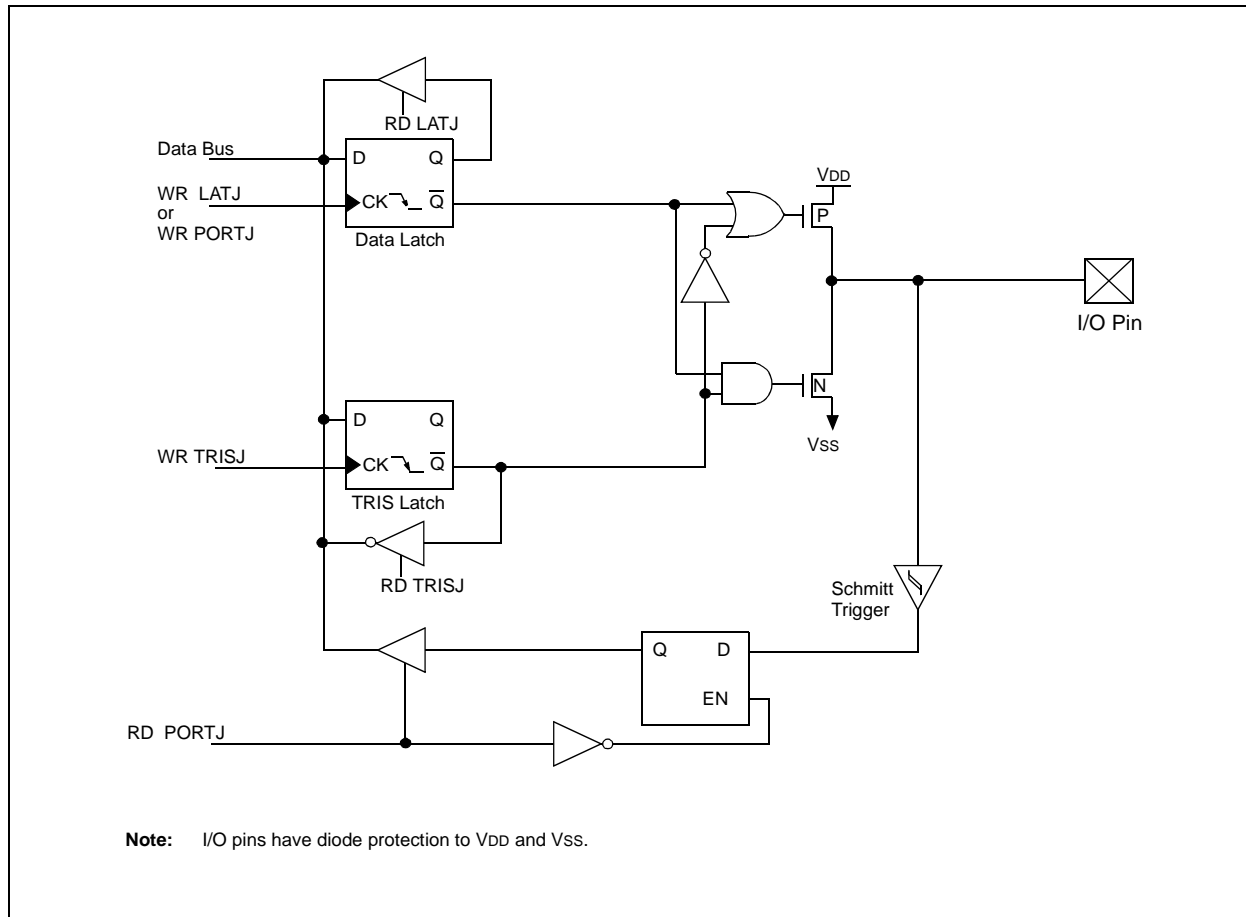


TABLE 8-17: PORTJ FUNCTIONS

| Name | Bit# | Buffer Type | Function |
|------|------|-------------|------------------------|
| RJ0 | bit0 | ST/TTL | Input/output port pin. |
| RJ1 | bit1 | ST/TTL | Input/output port pin. |
| RJ2 | bit2 | ST/TTL | Input/output port pin. |
| RJ3 | bit3 | ST/TTL | Input/output port pin. |
| RJ4 | bit4 | ST/TTL | Input/output port pin. |
| RJ5 | bit5 | ST/TTL | Input/output port pin. |
| RJ6 | bit6 | ST/TTL | Input/output port pin. |
| RJ7 | bit7 | ST/TTL | Input/output port pin. |

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

TABLE 8-18: 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 |

Legend: x = unknown, u = unchanged

PIC18CXX8

8.10 PORTK, LATA, and TRISA Registers

Note: This port is available on PIC18C858.

PORTK is an 8-bit wide, bi-directional port available only on the PIC18C858 devices. The corresponding Data Direction register is TRISK. Setting a TRISK bit (=1) will make the corresponding PORTK pin an input (i.e., put the corresponding output driver in a hi-impedance mode). Clearing a TRISK bit (=0) will make the corresponding PORTK pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATA register read and write the latched output value for PORTK.

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

EXAMPLE 8-10: INITIALIZING PORTK

```

CLRF   PORTK    ; Initialize PORTK by
                ; clearing output
                ; data latches
CLRF   LATA     ; Alternate method
                ; to clear output
                ; data latches
MOVLW  0xCF     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISK    ; Set RK3:RK0 as inputs
                ; RK5:RK4 as outputs
                ; RK7:RK6 as inputs
    
```

FIGURE 8-19: PORTK BLOCK DIAGRAM

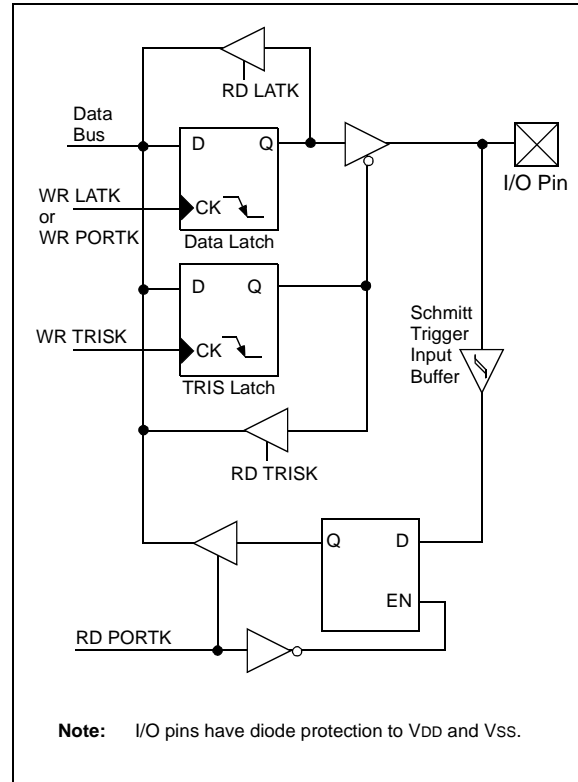


TABLE 8-19: PORTK FUNCTIONS

| Name | Bit# | Buffer Type | Function |
|------|------|-------------|------------------------|
| RK0 | bit0 | ST | Input/output port pin. |
| RK1 | bit1 | ST | Input/output port pin. |
| RK2 | bit2 | ST | Input/output port pin. |
| RK3 | bit3 | ST | Input/output port pin. |
| RK4 | bit4 | ST | Input/output port pin. |
| RK5 | bit5 | ST | Input/output port pin. |
| RK6 | bit6 | ST | Input/output port pin. |
| RK7 | bit7 | ST | Input/output port pin. |

Legend: ST = Schmitt Trigger input

TABLE 8-20: SUMMARY OF REGISTERS ASSOCIATED WITH PORTK

| 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 |
|-------|----------------------------------------------|-------|-------|-------|-------|-------|-------|-------|--------------------|---------------------------|
| TRISK | PORTK Data Direction Control Register | | | | | | | | 1111 1111 | 1111 1111 |
| PORTK | Read PORTK pin / Write PORTK Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATA | Read PORTK Data Latch/Write PORTK Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |

Legend: x = unknown, u = unchanged

9.0 PARALLEL SLAVE PORT

The Parallel Slave Port is an 8-bit parallel interface for transferring data between the PIC18CXX8 device and an external device.

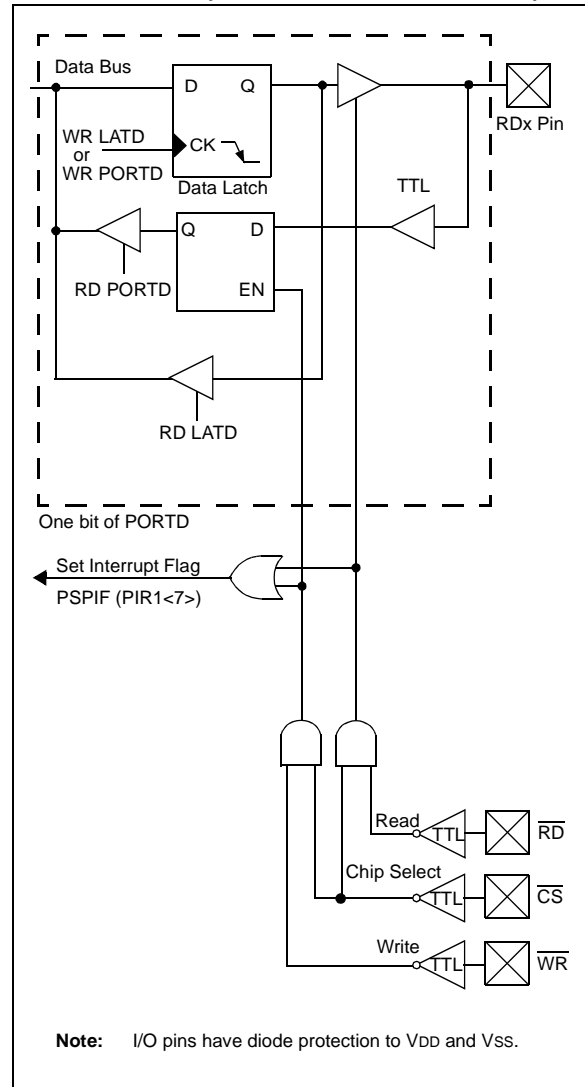
PORTD operates as an 8-bit wide Parallel Slave Port, or microprocessor port when control bit PSPMODE (PSPCON register) is set. In Slave mode, it is asynchronously readable and writable by the external world through \overline{RD} control input pin RE0/ \overline{RD} and \overline{WR} control input pin RE1/ \overline{WR} .

It can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting bit PSPMODE enables port pin RE0/ \overline{RD} to be the \overline{RD} input, RE1/ \overline{WR} to be the \overline{WR} input and RE2/ \overline{CS} to be the \overline{CS} (chip select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set).

A write to the PSP occurs when both the \overline{CS} and \overline{WR} lines are first detected low. A read from the PSP occurs when both the \overline{CS} and \overline{RD} lines are first detected low.

The PORTE I/O pins become control inputs for the microprocessor port when bit PSPMODE (PSPCON Register) is set. In this mode, the user must make sure that the TRISE<2:0> bits are set (pins are configured as digital inputs). In this mode, the input buffers are TTL.

FIGURE 9-1: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)



PIC18CXX8

REGISTER 9-1: PSPCON REGISTER

| | | | | | | | |
|-------|-----|-------|---------|-------|-----|-----|-----|
| R-0 | R-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
| IBF | OBF | IBOV | PSPMODE | — | — | — | — |
| bit 7 | | | | bit 0 | | | |

- bit 7 **IBF**: Input Buffer Full Status bit
1 = A word has been received and waiting to be read by the CPU
0 = No word has been received
- bit 6 **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read
- bit 5 **IBOV**: Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read
(must be cleared in software)
0 = No overflow occurred
- bit 4 **PSPMODE**: Parallel Slave Port Mode Select bit
1 = Parallel Slave Port mode
0 = General purpose I/O mode
- bit 3-0 **Unimplemented**: Read 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 |

FIGURE 9-2: PARALLEL SLAVE PORT WRITE WAVEFORMS

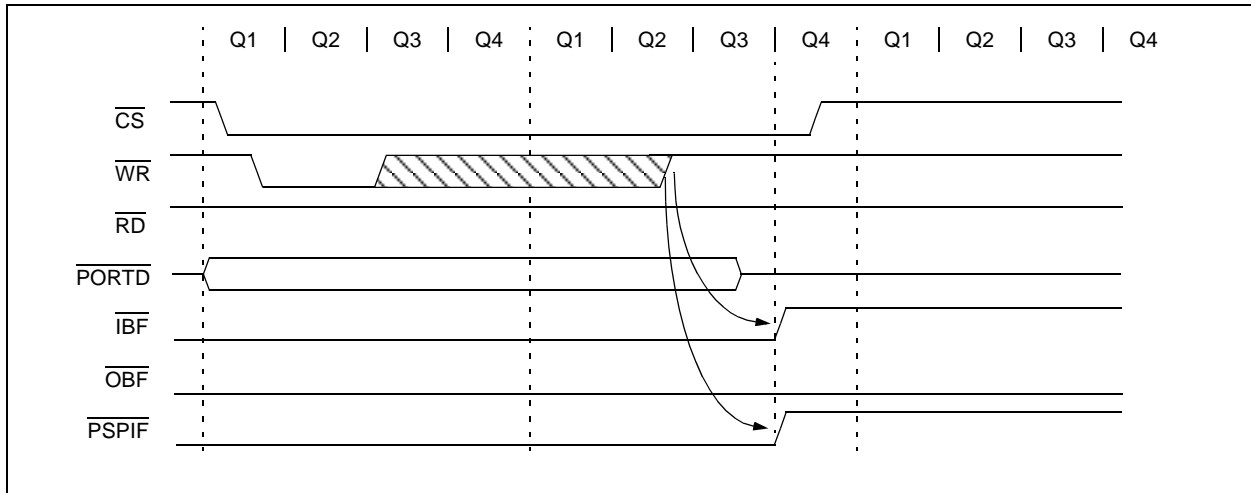


FIGURE 9-3: PARALLEL SLAVE PORT READ WAVEFORMS

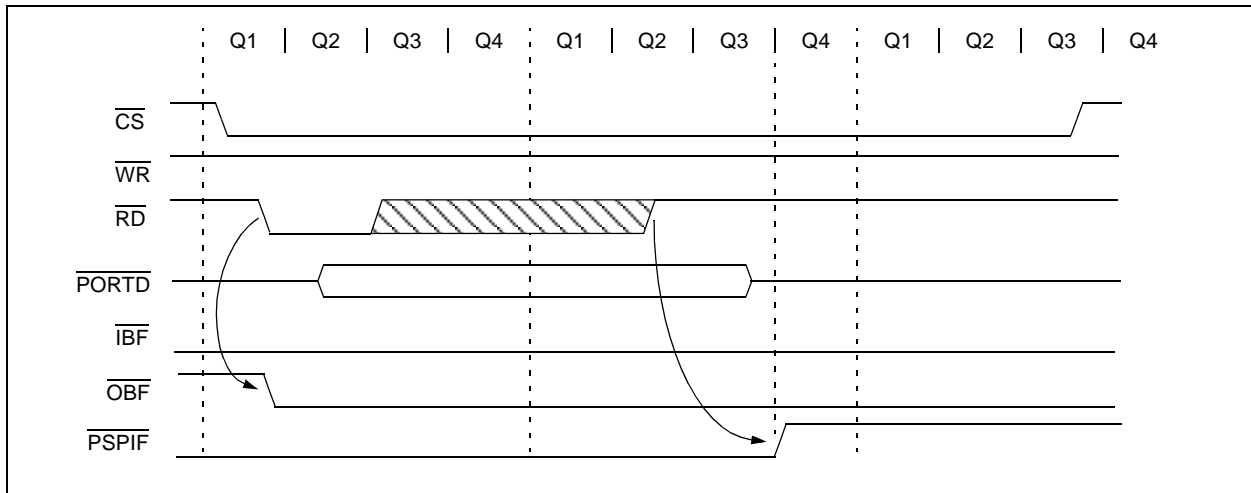


TABLE 9-1: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT

| 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 | Port data latch when written; port pins when read | | | | | | | | xxxx xxxx | uuuu uuuu |
| LATD | LATD Data Output Bits | | | | | | | | xxxx xxxx | uuuu uuuu |
| TRISD | PORTD Data Direction Bits | | | | | | | | 1111 1111 | 1111 1111 |
| PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 | 0000 0000 | 0000 0000 |
| LATE | LATE Data Output Bits | | | | | | | | xxxx xxxx | uuuu uuuu |
| TRISE | PORTE Data Direction Bits | | | | | | | | 1111 1111 | 1111 1111 |
| INTCON | GIE/ GIEH | PEIE/ GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 0000 |

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

PIC18CXX8

NOTES:

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 10-1 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 | 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 |

PIC18CXX8

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

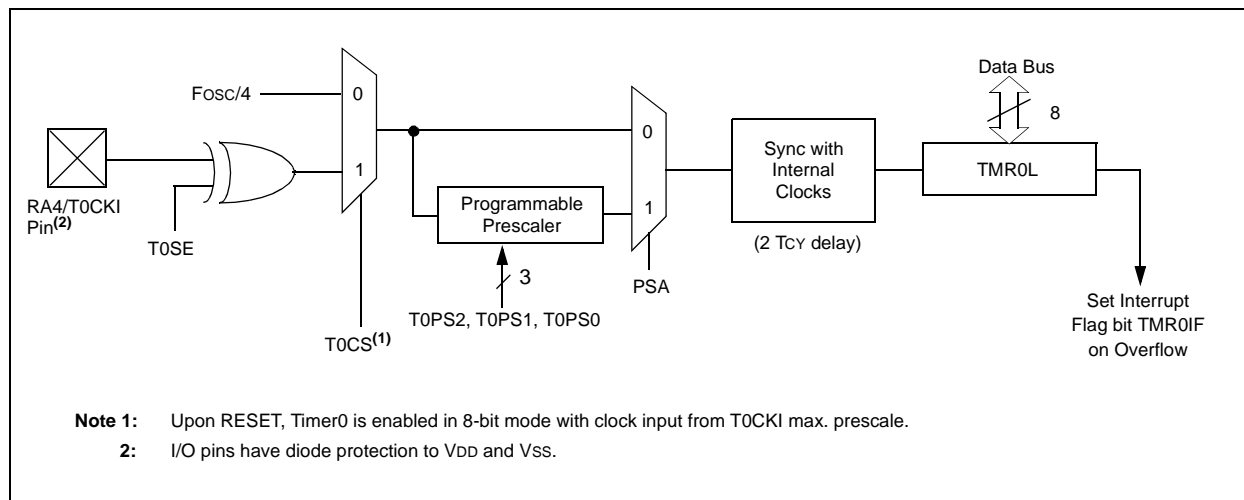
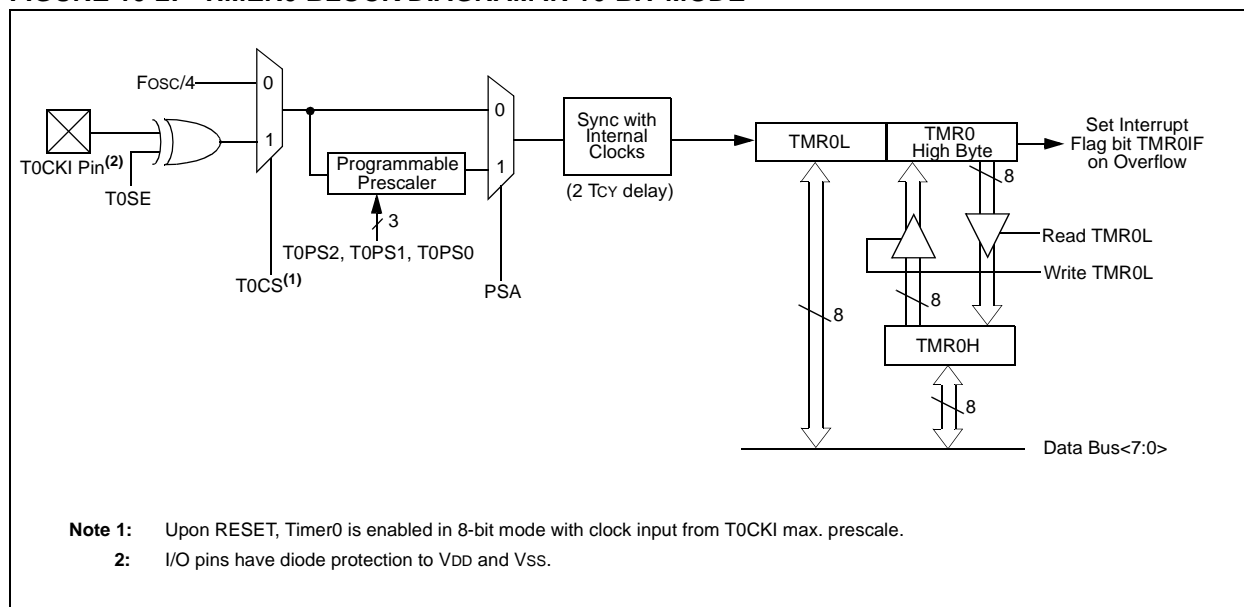


FIGURE 10-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE



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).

PIC18CXX8

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 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 ⁽¹⁾ | | | | | | | --11 1111 | --11 1111 |

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

Note 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read as '0'.

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 | T1SYNC | TMR1CS | TMR1ON |
| | | | | | | | bit 0 |
| bit 7 | | | | | | | |

- 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 **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 |

PIC18CXX8

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).

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 (Section 14.0).

FIGURE 11-1: TIMER1 BLOCK DIAGRAM

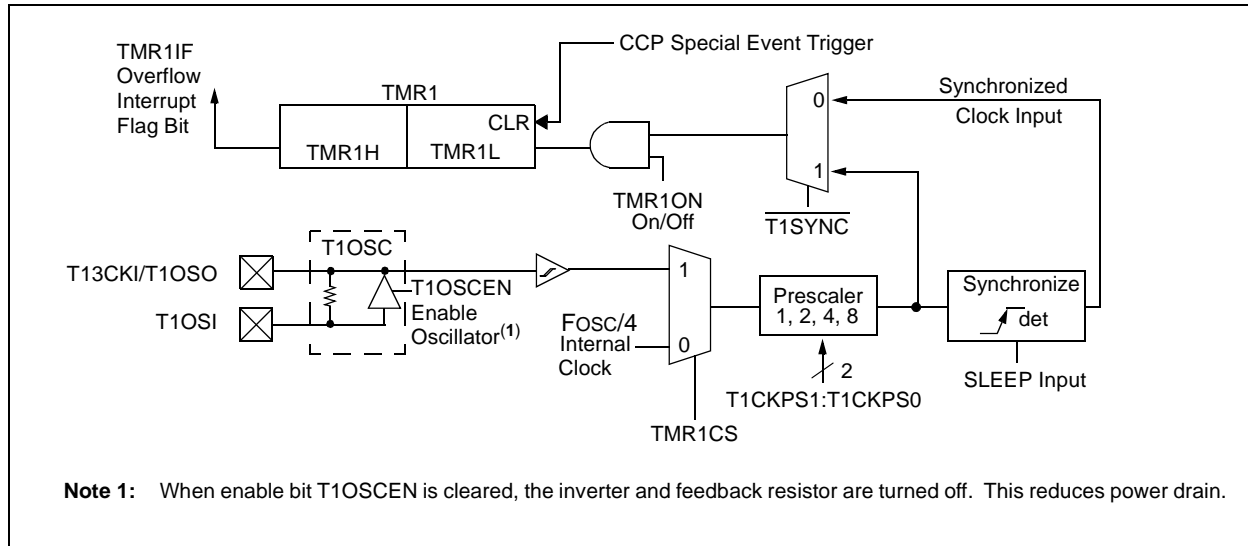
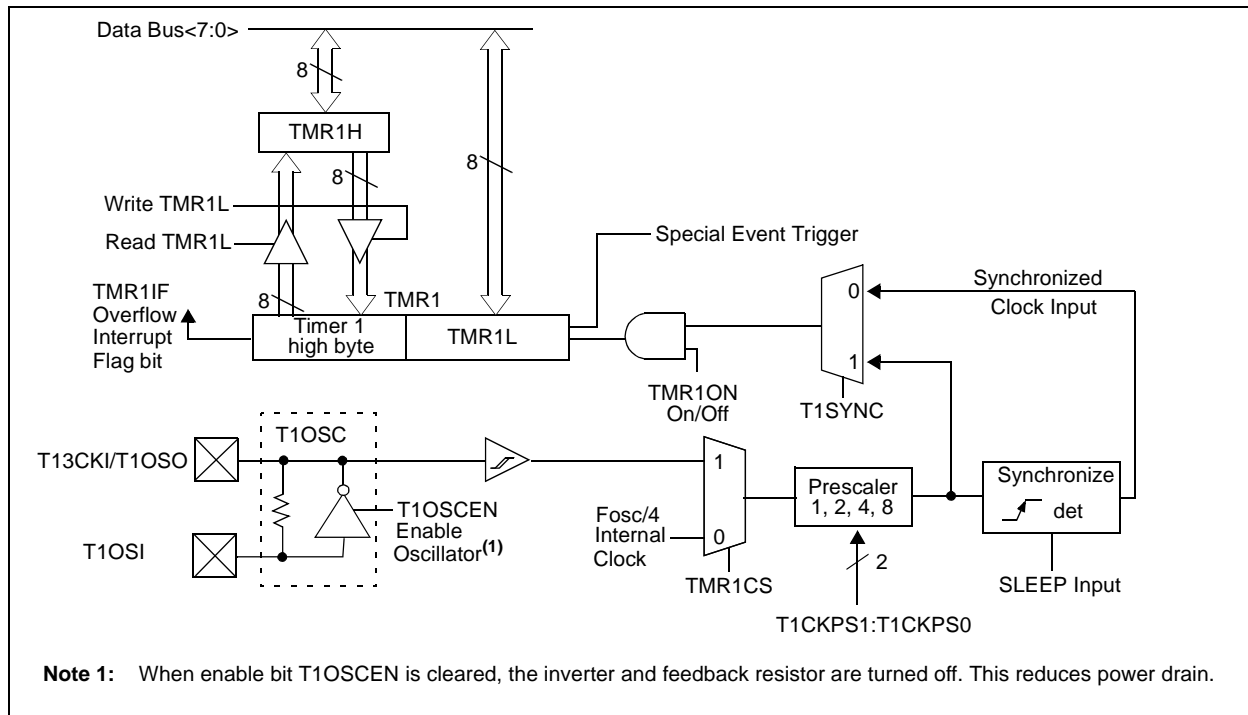


FIGURE 11-2: TIMER1 BLOCK DIAGRAM: 16-BIT READ/WRITE MODE



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 |
| <p>Note 1: Microchip suggests 33 pF as a starting point in validating the oscillator circuit.</p> <p>2: Higher capacitance increases the stability of the oscillator but also increases the start-up time.</p> <p>3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.</p> <p>4: Capacitor values are for design guidance only.</p> | |

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.

PIC18CXX8

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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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 | 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 ($F_{osc}/4$) 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{MCLR} Reset, Watchdog Timer Reset, or Brown-out 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 | |
| bit 7 | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 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

PIC18CXX8

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

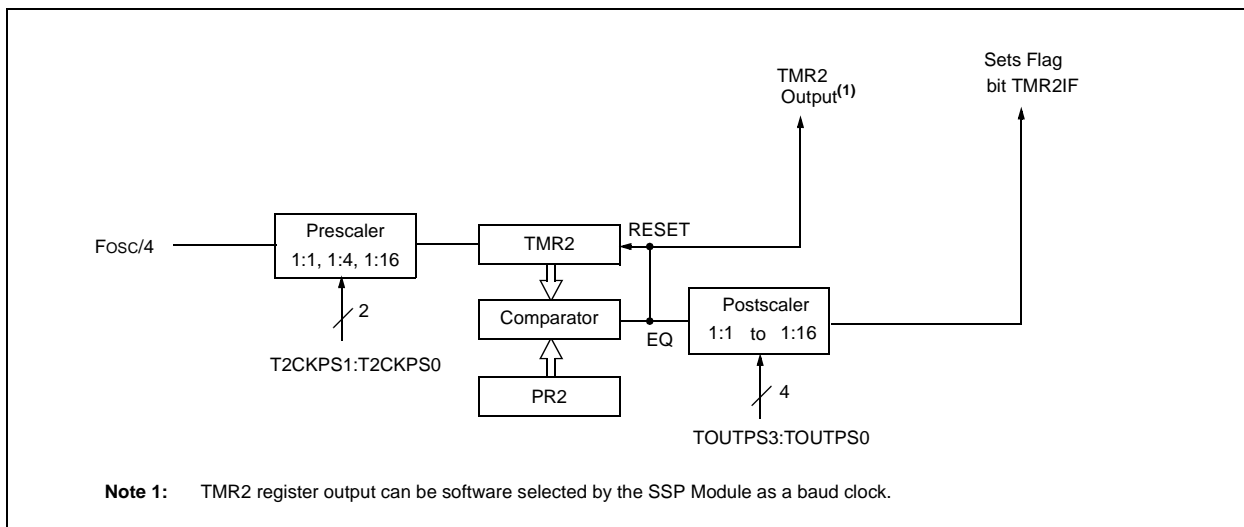


TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

| 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 | RBF | 0000 000x | 0000 000u |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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 | $\overline{T3SYNC}$ | 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 **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 |

PIC18CXX8

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

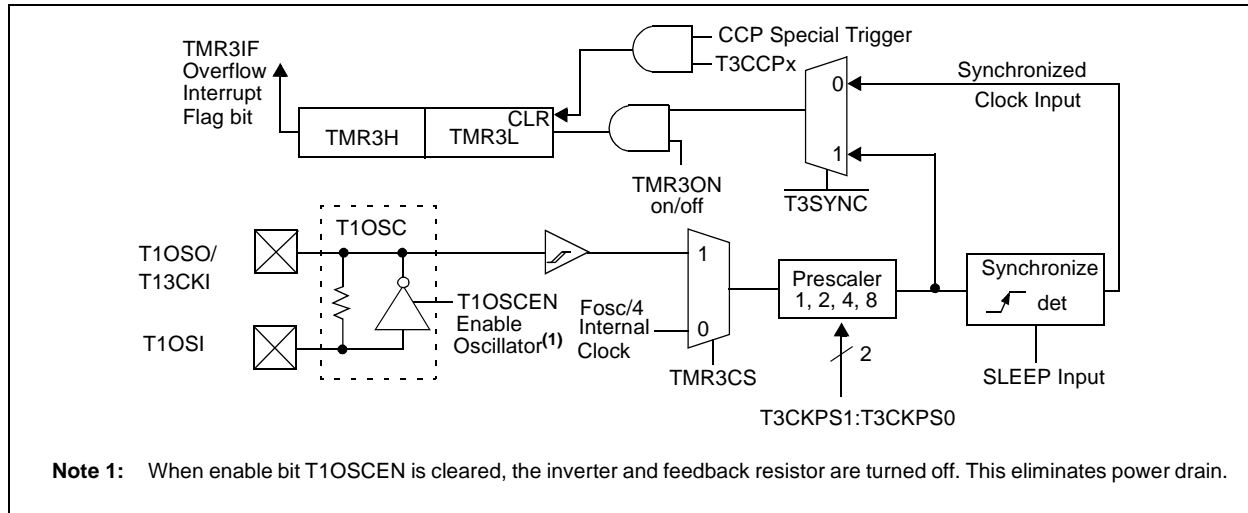
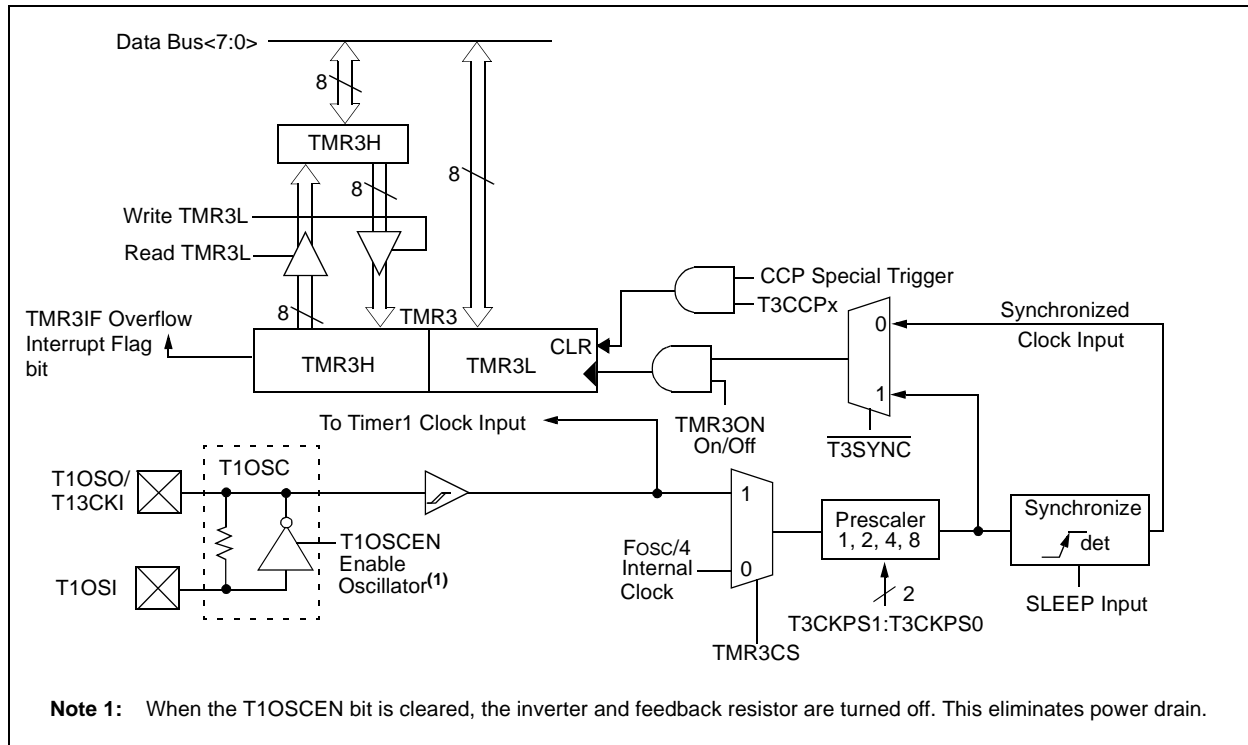


FIGURE 13-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE



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 (PIR 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 “Capture/Compare/PWM (CCP) Modules”, Section 14.0 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 | — | CMIF | — | — | BCLIF | LVDIF | TMR3IF | CCP2IF | -0-- 0000 | -0-- 0000 |
| PIE2 | — | CMIE | — | — | BCLIE | LVDIE | TMR3IE | CCP2IE | -0-- 0000 | -0-- 0000 |
| IPR2 | — | CMIP | — | — | BCLIP | LVDIP | TMR3IP | CCP2IP | -0-- 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 | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 0-00 0000 | u-uu uuuu |
| T3CON | RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 0000 0000 | uuuu uuuu |

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

PIC18CXX8

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 and the CAN message timestamp received. (Refer to "CAN Module",

Section 17.0 for CAN operation.) 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 = Capture mode, CAN message received (CCP1 only)

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 Compare PWM | Timer1 or Timer3 Timer1 or Timer3 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.

14.3.5 CAN MESSAGE RECEIVED

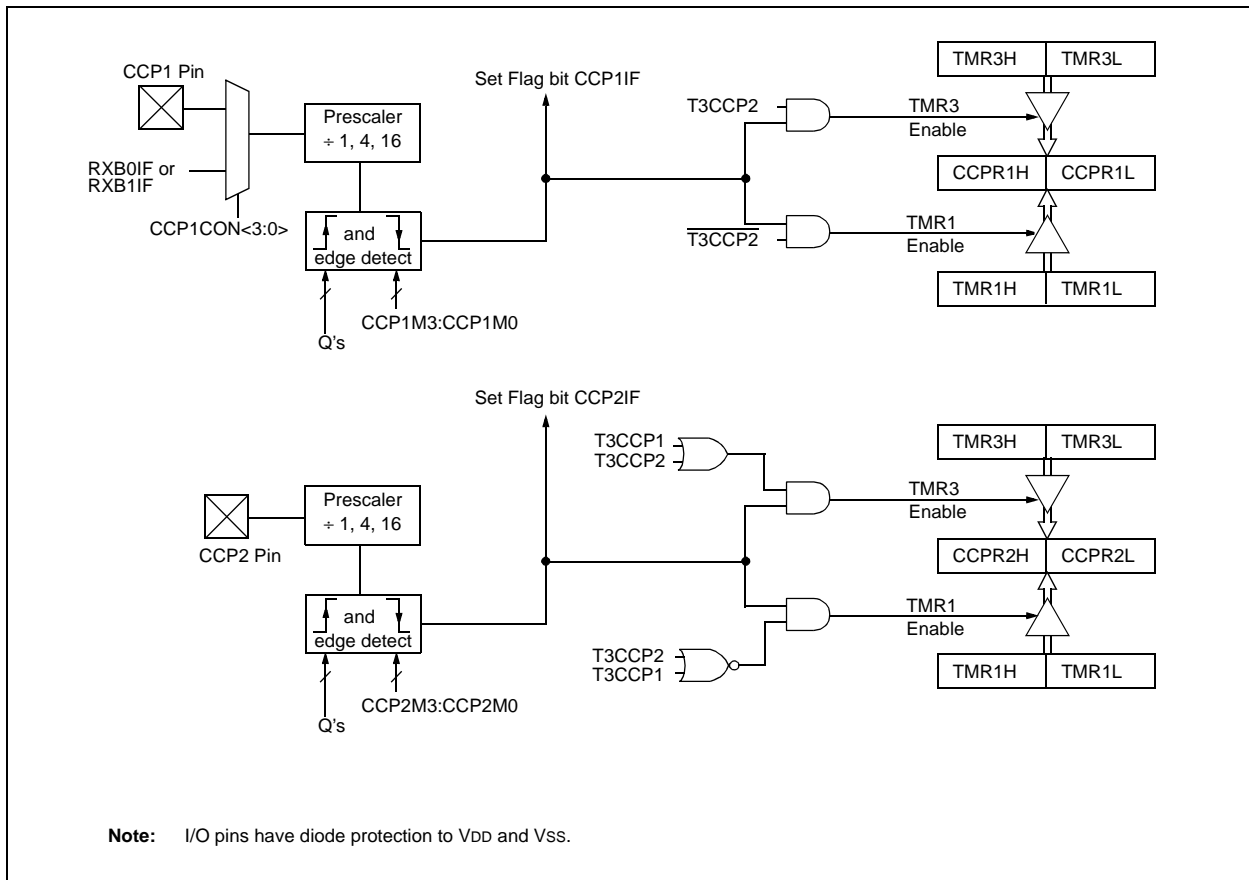
The CAN capture event occurs when a message is received in either receive buffer. The CAN module provides a rising edge to the CCP module to cause a capture event. This feature is provided to time-stamp the received CAN messages.

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



PIC18CXX8

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

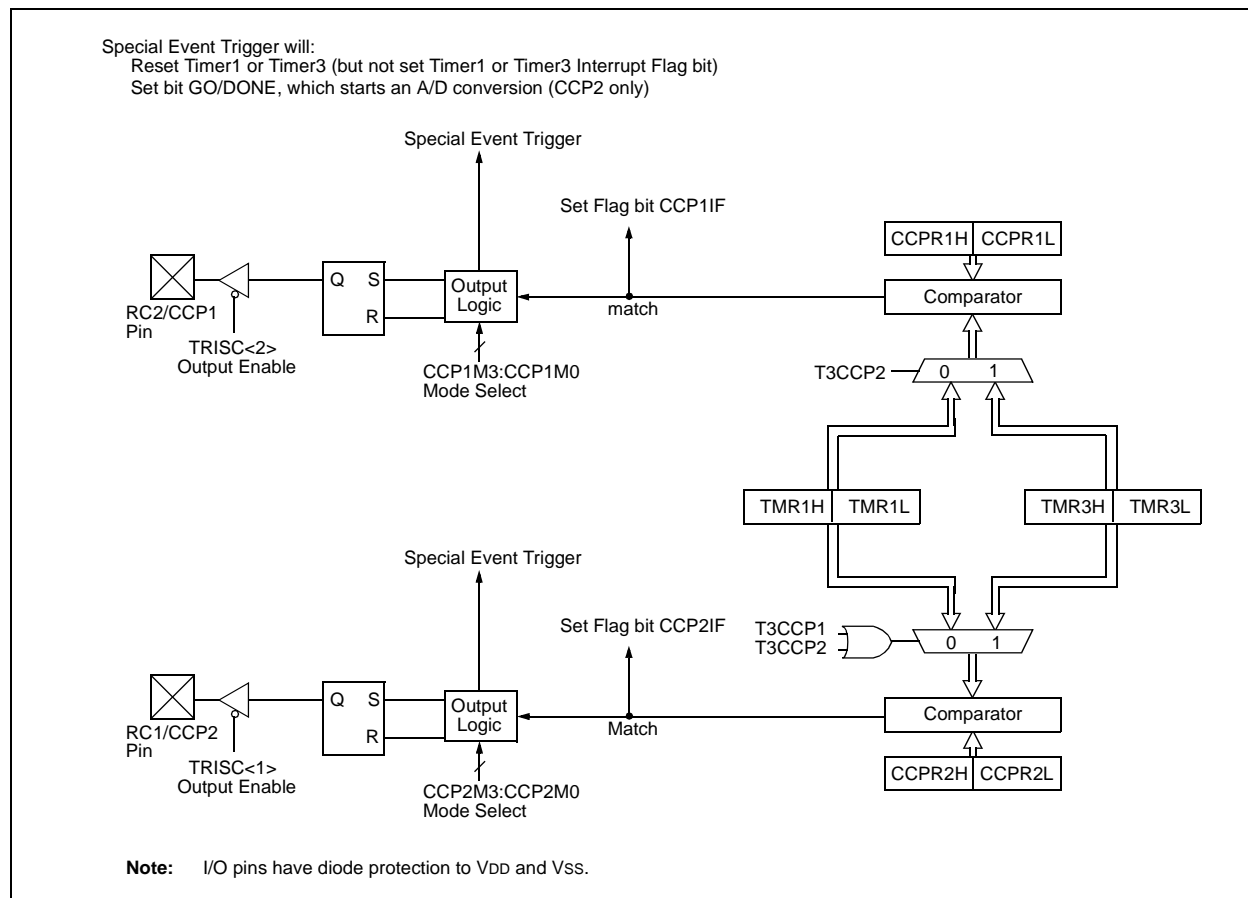


TABLE 14-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

| 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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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 | $\overline{T1SYNC}$ | 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 | — | CMIF | — | — | BCLIF | LVDIF | TMR3IF | CCP2IF | -0-- 0000 | -0-- 0000 |
| PIE2 | — | CMIE | — | — | BCLIE | LVDIE | TMR3IE | CCP2IE | -0-- 0000 | -0-- 0000 |
| IPR2 | — | CMIP | — | — | BCLIP | LVDIP | TMR3IP | CCP2IP | -0-- 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 |
| T3CON | RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 0000 0000 | uuuu uuuu |

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.

PIC18CXX8

14.5 PWM Mode

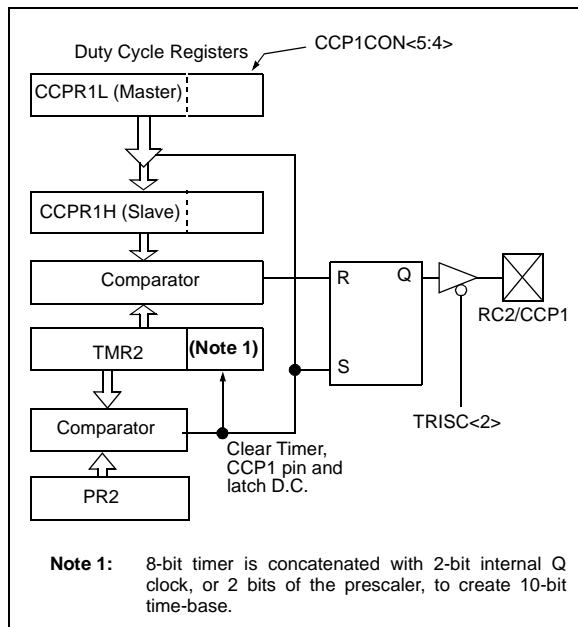
In Pulse Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

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.

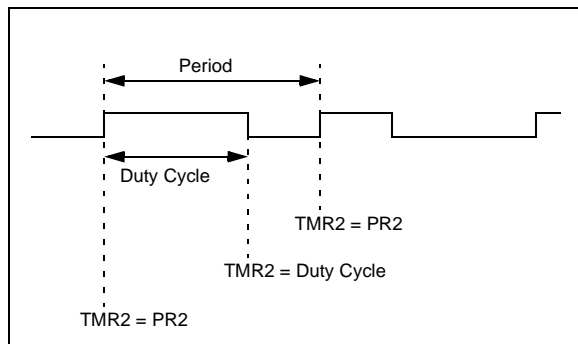
For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 14.5.3.

FIGURE 14-3: SIMPLIFIED PWM BLOCK DIAGRAM



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 using the following formula:

$$\text{PWM period} = \frac{[(PR2) + 1] \cdot 4 \cdot T_{osc}}{(\text{TMR2 prescale value})}$$

PWM frequency is defined as $1 / [\text{PWM period}]$.

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 12.0) 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:

$$\text{PWM duty cycle} = \frac{(\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{osc}}{(\text{TMR2 prescale value})}$$

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.

Maximum PWM resolution (bits) for a given PWM frequency:

$$= \frac{\log\left(\frac{F_{osc}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

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 40 MHz

| PWM Frequency | 2.44 kHz | 9.76 kHz | 39.06 kHz | 156.3 kHz | 312.5 kHz | 416.6 kHz |
|----------------------------|----------|----------|-----------|-----------|-----------|-----------|
| Timer Prescaler (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 5.5 |

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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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 | — | CMIF | — | — | BCLIF | LVDIF | TMR3IF | CCP2IF | -0-- 0000 | -0-- 0000 |
| PIE2 | — | CMIE | — | — | BCLIE | LVDIE | TMR3IE | CCP2IE | -0-- 0000 | -0-- 0000 |
| IPR2 | — | CMIP | — | — | BCLIP | LVDIP | TMR3IP | CCP2IP | -0-- 0000 | -0-- 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 (I²C)
 - Full Master mode
 - Slave mode (with general address call)

The I²C interface supports the following modes in hardware:

- Master mode
- Multi-master mode
- Slave mode

PIC18CXX8

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/A | P | S | R/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 I²C 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/A:** Data/Address bit (I²C 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
 (I²C 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
 (I²C 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/W:** Read/Write bit information (I²C 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 $\overline{\text{ACK}}$ bit.
In I²C Slave mode:
 1 = Read
 0 = Write
In I²C 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 I²C 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 I²C modes)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
Transmit (I²C mode only)
1 = Data Transmit in progress (does not include the \overline{ACK} and STOP bits), SSPBUF is full
0 = Data Transmit complete (does not include the \overline{ACK} and STOP bits), SSPBUF is empty

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 I²C 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 I²C 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 I²C 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 I²C Slave mode:

SCK release control

1 = Enable clock

0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In I²C 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 = I²C Slave mode, 7-bit address
0111 = I²C Slave mode, 10-bit address
1000 = I²C Master mode, clock = Fosc / (4 * (SSPADD+1))
1001 = Reserved
1010 = Reserved
1011 = I²C firmware controlled Master mode (Slave idle)
1100 = Reserved
1101 = Reserved
1110 = I²C Slave mode, 7-bit address with START and STOP bit interrupts enabled
1111 = I²C 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 |

PIC18CXX8

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 I²C 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 I²C 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 I²C 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 I²C 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 I²C Master mode only)
 1 = Enables Receive mode for I²C
 0 = Receive idle
- bit 2 **PEN:** STOP Condition Enable bit (In I²C 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 I²C 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 I²C 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 I²C 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

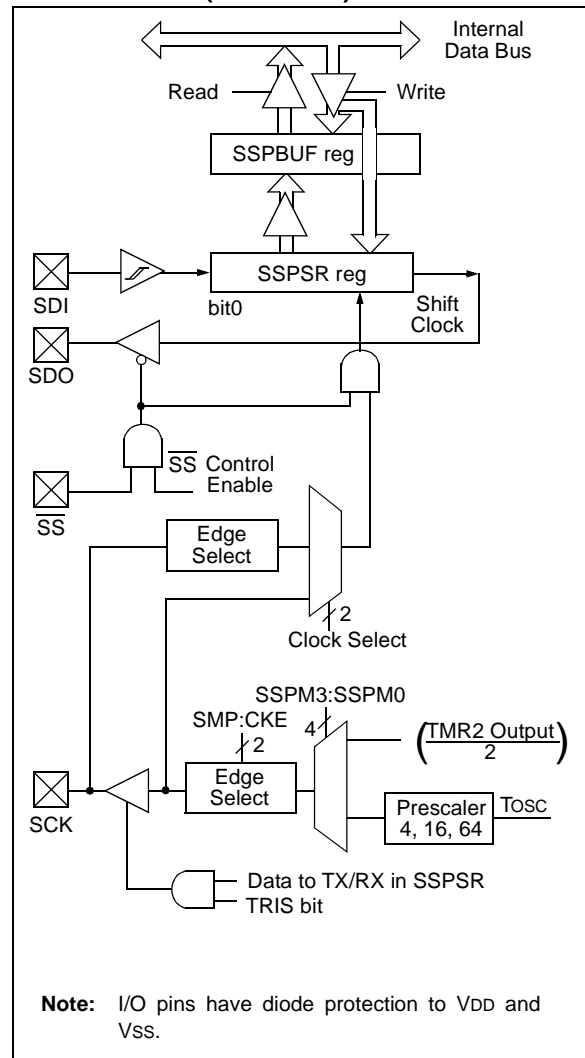
15.3.1 OPERATION

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)



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, some 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
- \overline{SS} must have TRISC<4> 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)? |
| | GOTO | 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

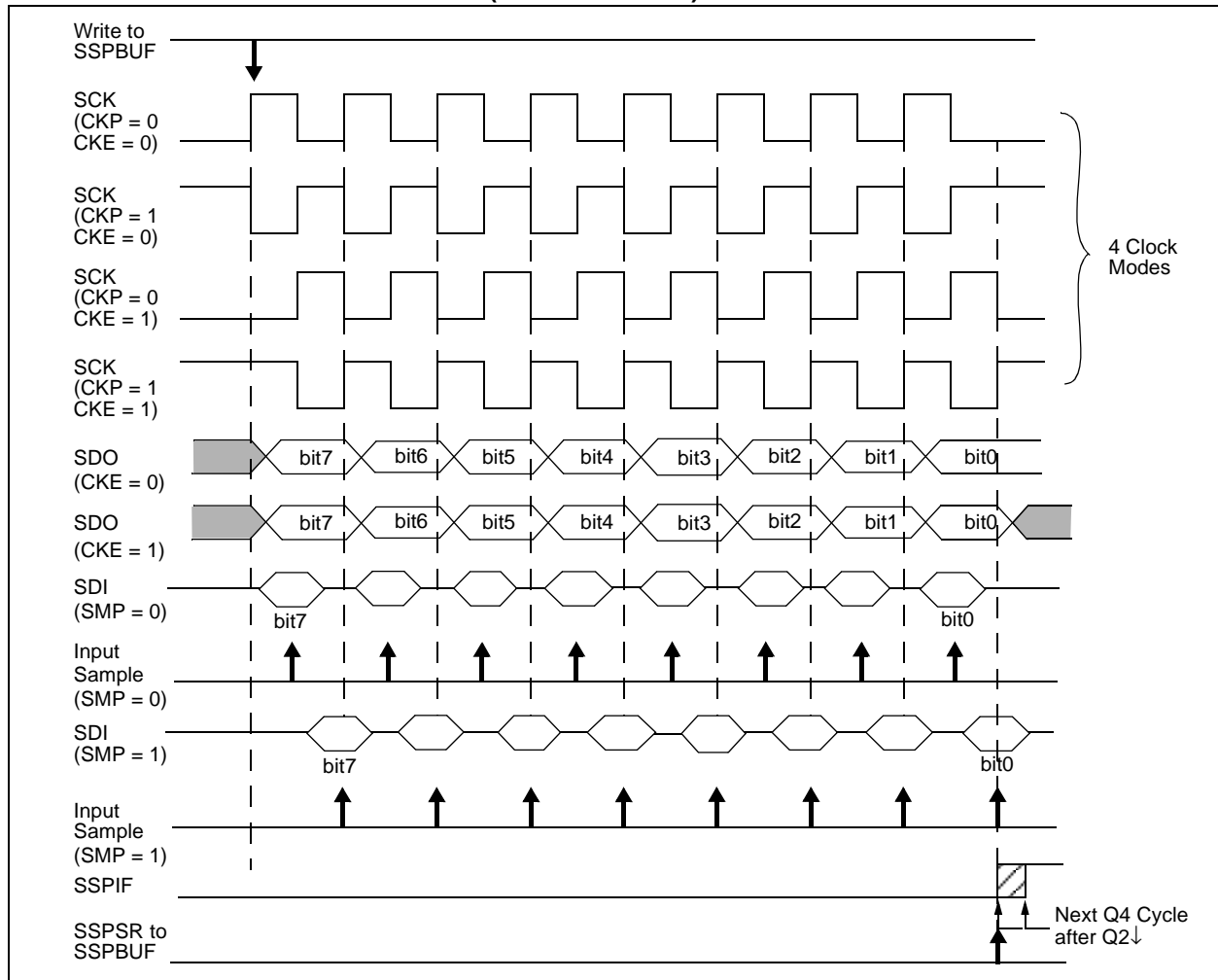
shown in Figure 15-2, Figure 15-4, and Figure 15-5, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- $F_{osc}/4$ (or T_{CY})
- $F_{osc}/16$ (or $4 \cdot T_{CY}$)
- $F_{osc}/64$ (or $16 \cdot T_{CY}$)
- $\text{Timer2 output}/2$

This allows a maximum data rate (at 40 MHz) of 10.00 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)



PIC18CXX8

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 ($SSPCON1\langle 3:0 \rangle = 04h$). 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, ($SSPCON\langle 3:0 \rangle = 0100$) 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

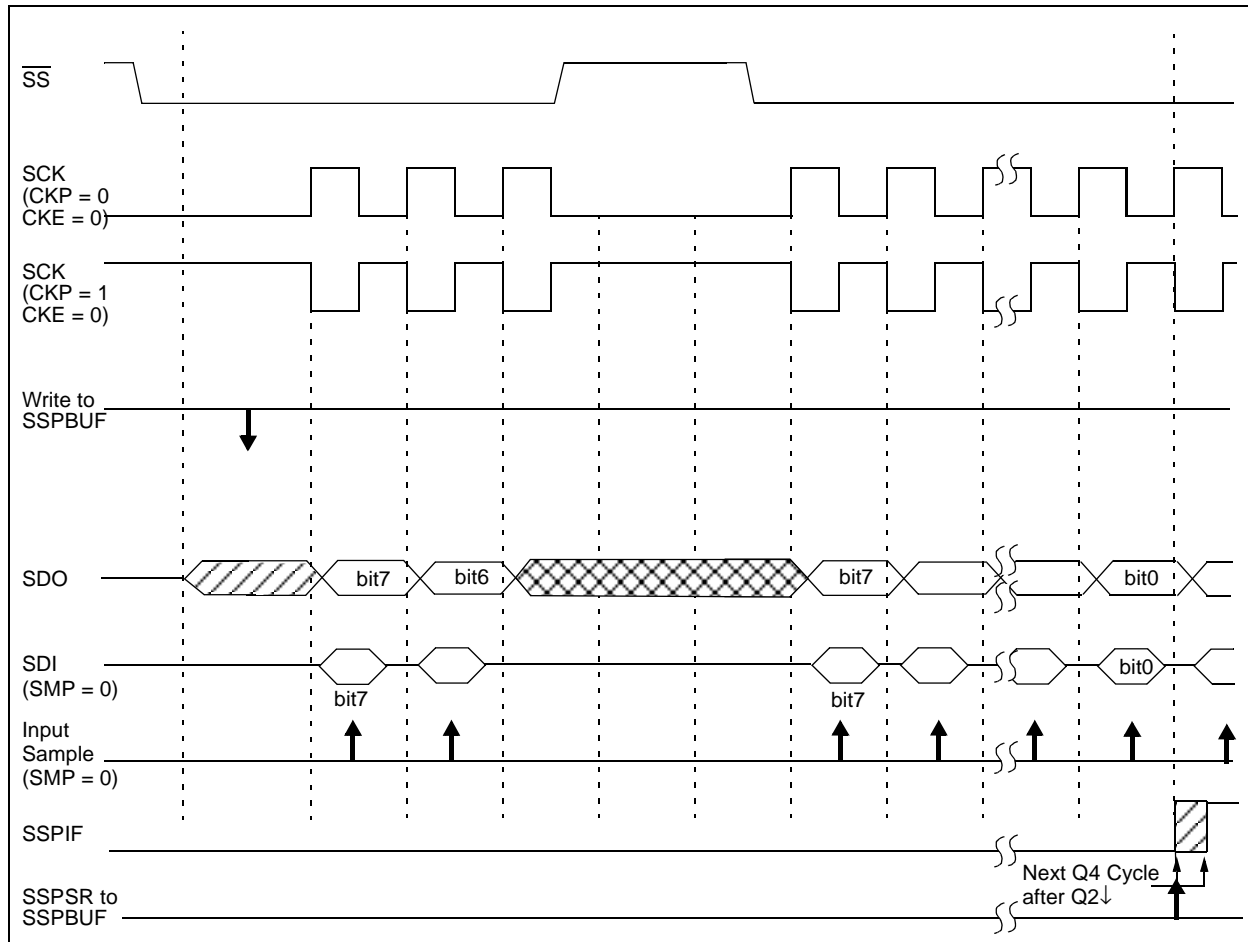


FIGURE 15-4: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

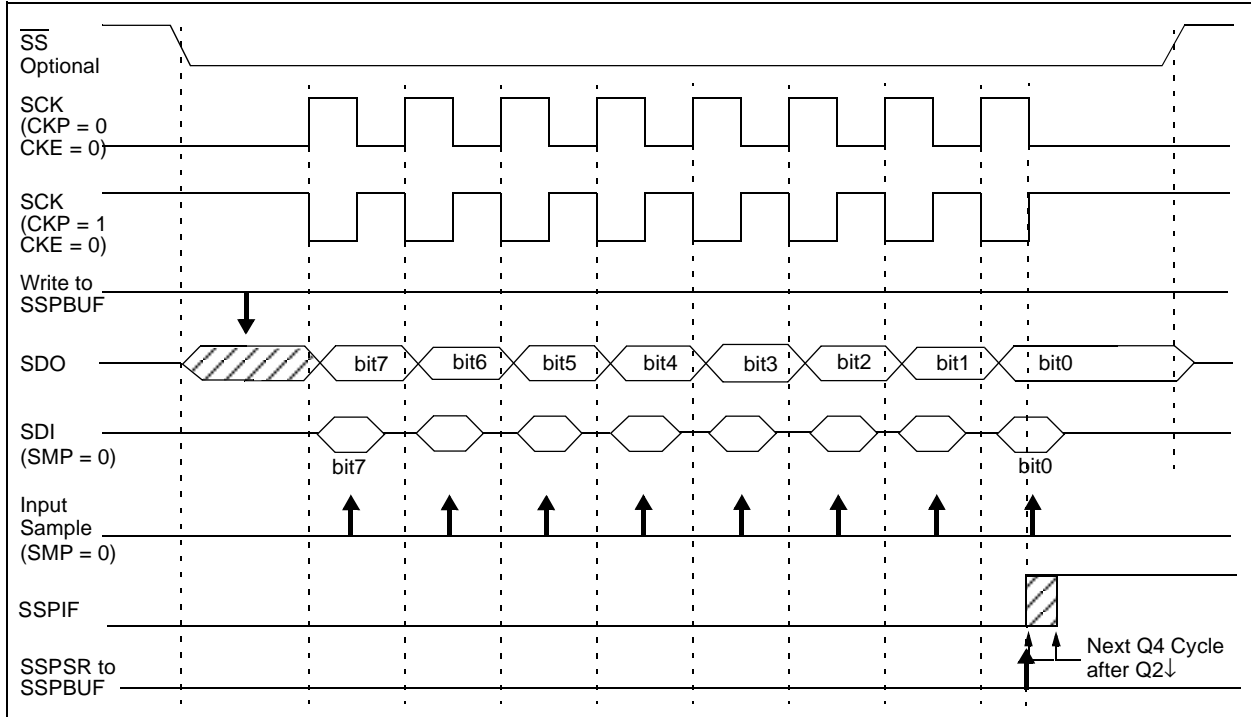
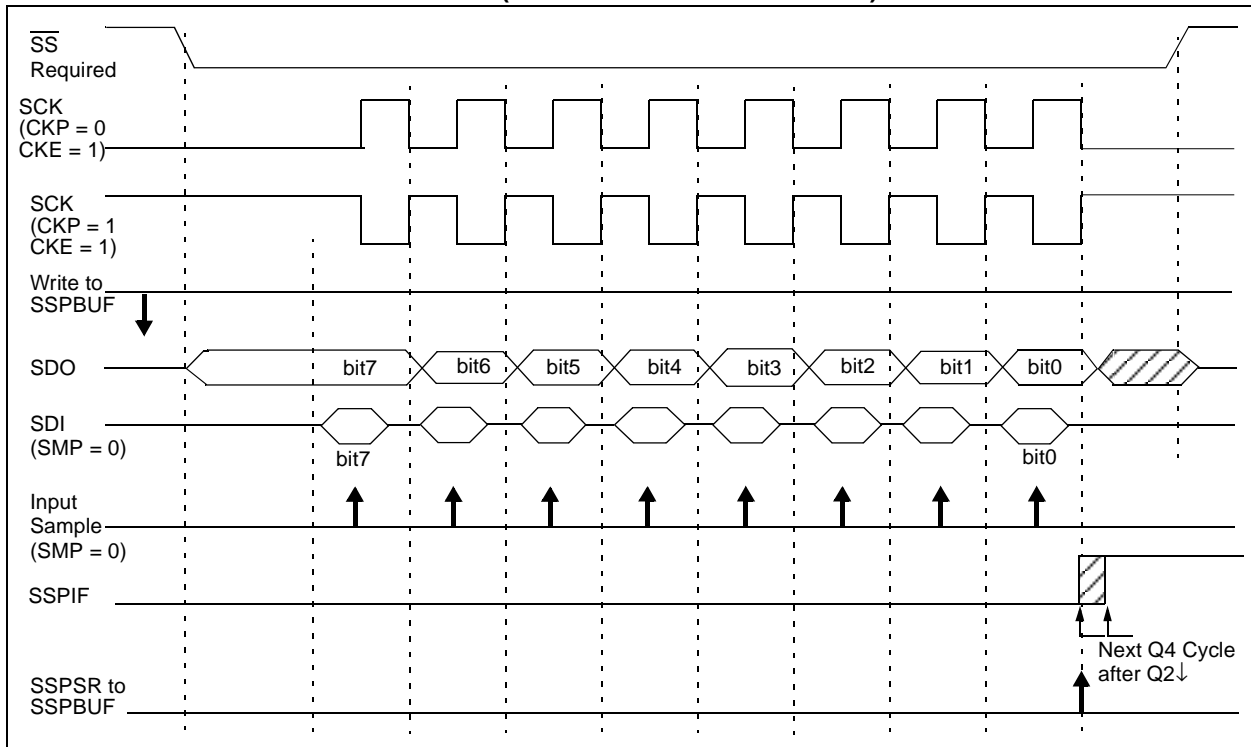


FIGURE 15-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



PIC18CXX8

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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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 ⁽¹⁾ | | | | | | | --11 1111 | --11 1111 |
| SSPSTAT | SMP | CKE | D/Ā | P | S | R/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.

Note 1: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

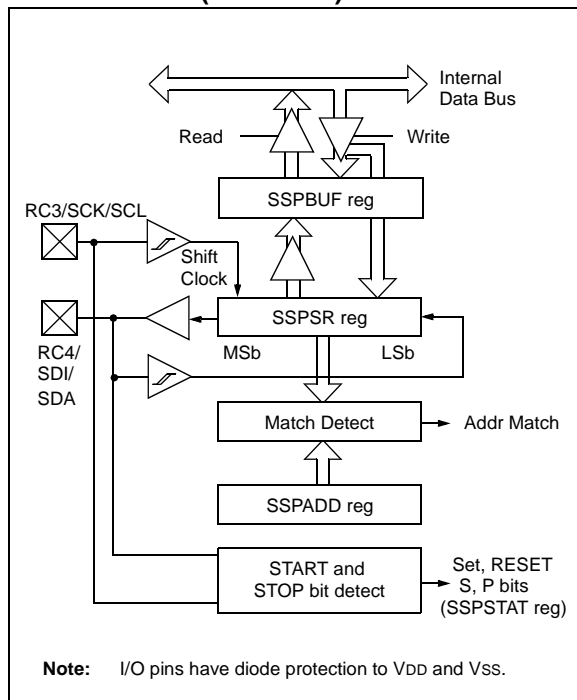
15.4 MSSP I²C Operation

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).

FIGURE 15-6: MSSP BLOCK DIAGRAM (I²C MODE)



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)

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 ($\overline{\text{ACK}}$) 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 $\overline{\text{ACK}}$ pulse:

- The buffer full bit BF (SSPCON1 register) was set before the transfer was received.
- 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/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 $\overline{R/W}$ bit of the address byte is clear and an address match occurs, the $\overline{R/W}$ 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 $\overline{R/W}$ bit of the incoming address byte is set and an address match occurs, the $\overline{R/W}$ 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)

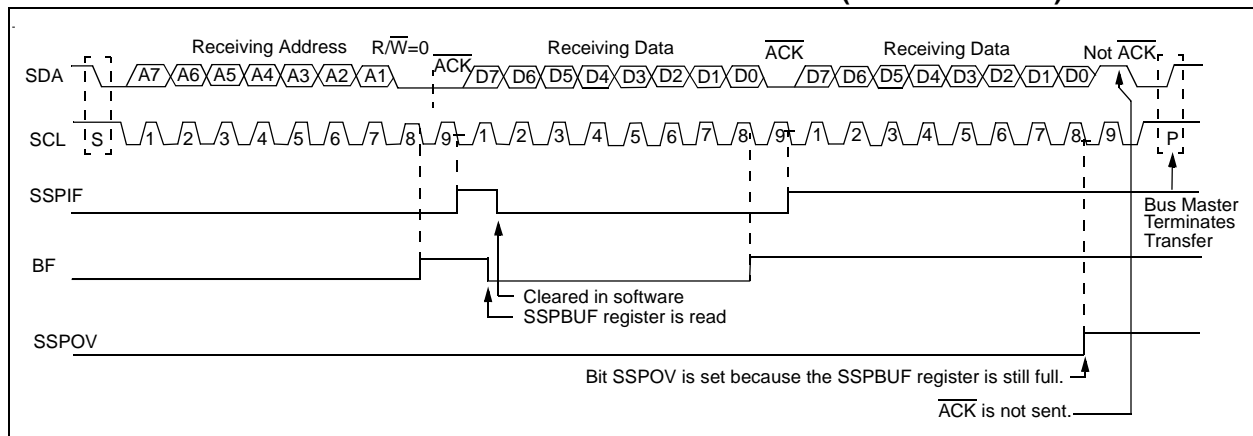
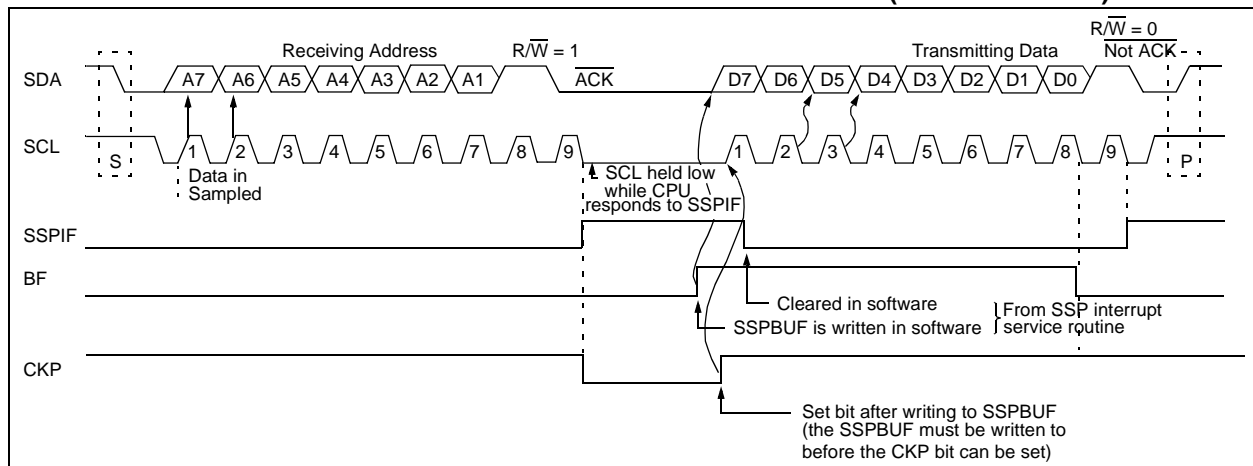


FIGURE 15-8: I²C SLAVE MODE WAVEFORMS FOR TRANSMISSION (7-BIT ADDRESS)



15.4.2 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I²C 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 I²C protocol. It consists of all 0's with R/W = 0.

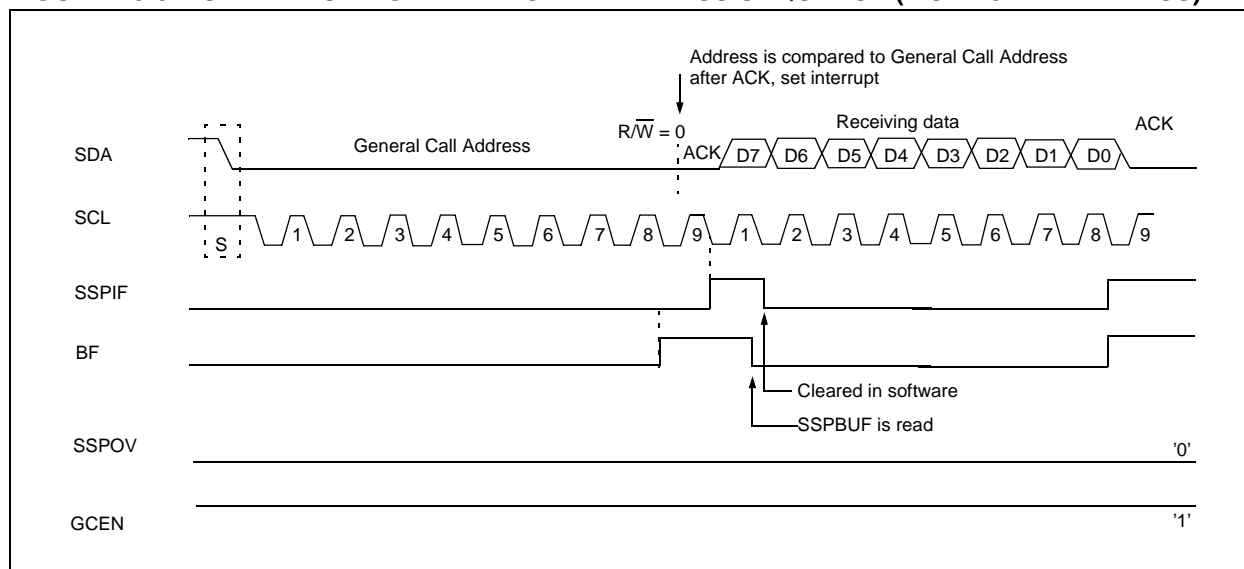
The general call address is recognized (enabled) when the General Call Enable (GCEN) bit is set (SSPCON2 register). Following a START bit detect, eight bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

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 (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 I²C 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

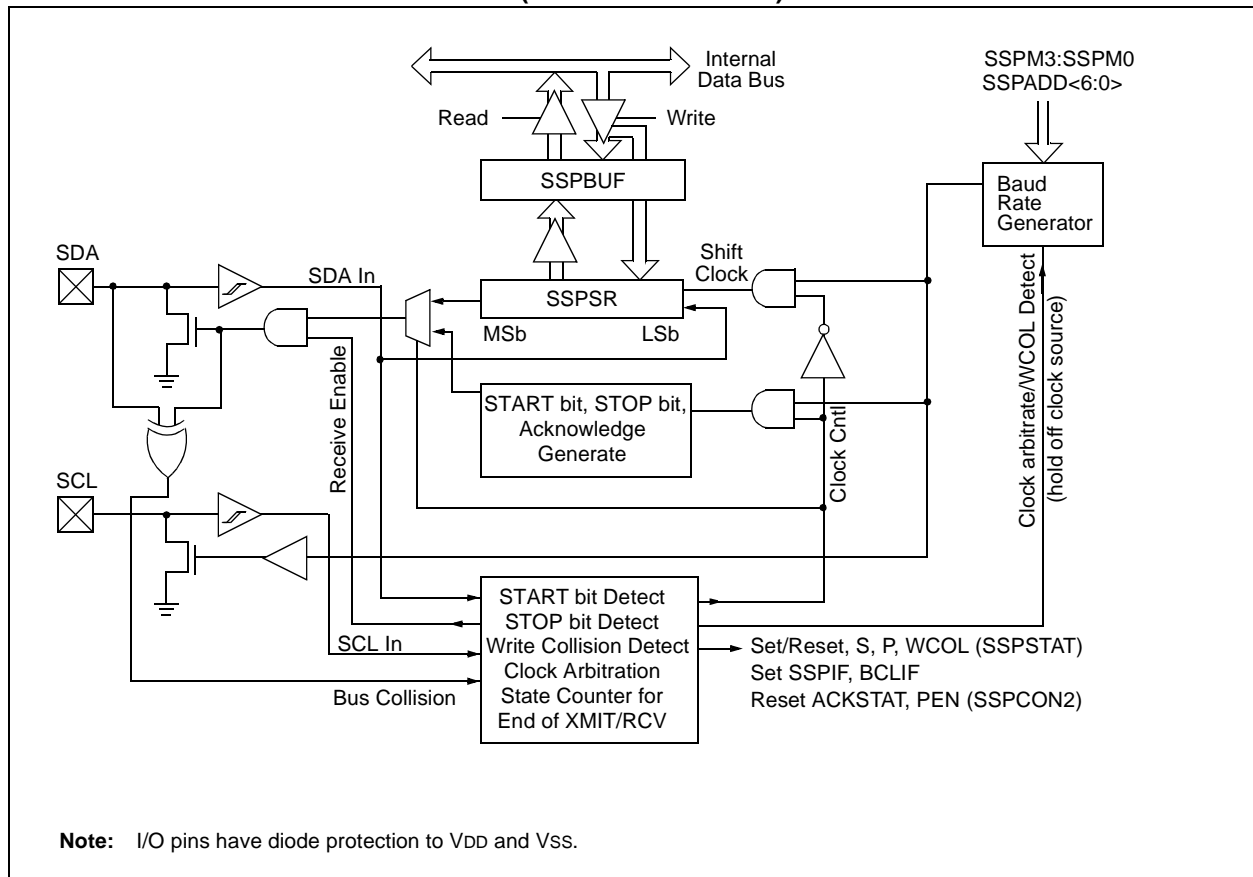
15.4.4 I²C MASTER MODE SUPPORT

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 I²C 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 I²C Master mode, does not allow queuing 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 (I²C MASTER MODE)



15.4.4.1 I²C Master Mode Operation

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a Repeated START condition. Since the Repeated START condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In this case, the R/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/W bit. In this case, the R/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 I²C 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 I²C 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 dec-

remented twice per instruction cycle (T_{cy}) on the Q2 and Q4 clocks. In I²C 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

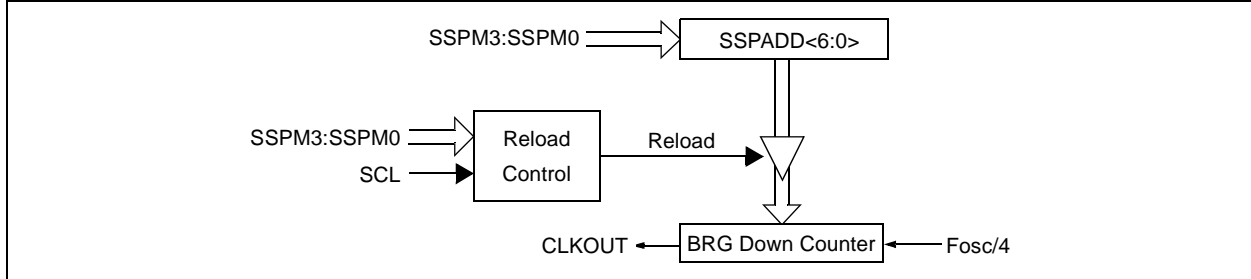
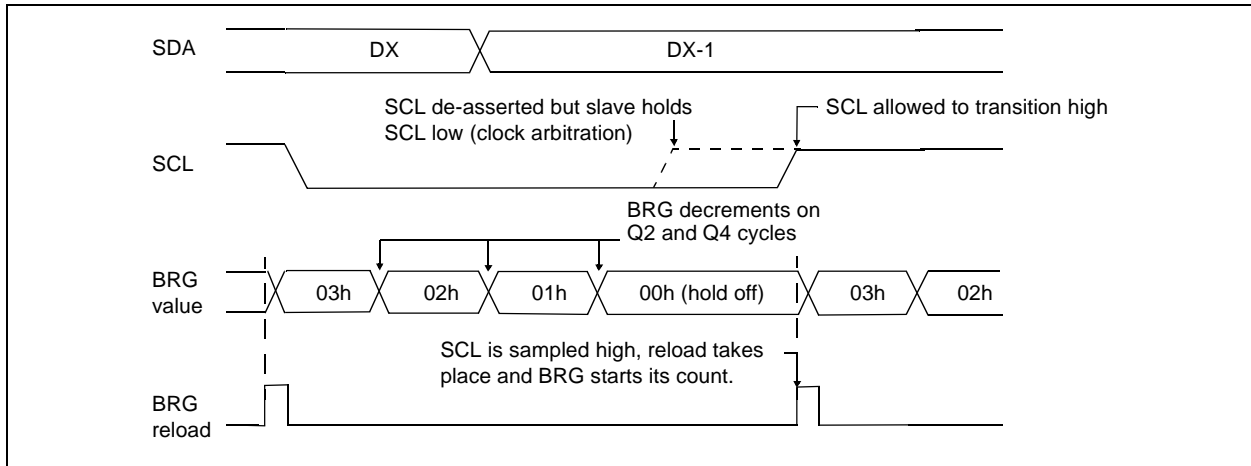


FIGURE 15-12: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



PIC18CXX8

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.

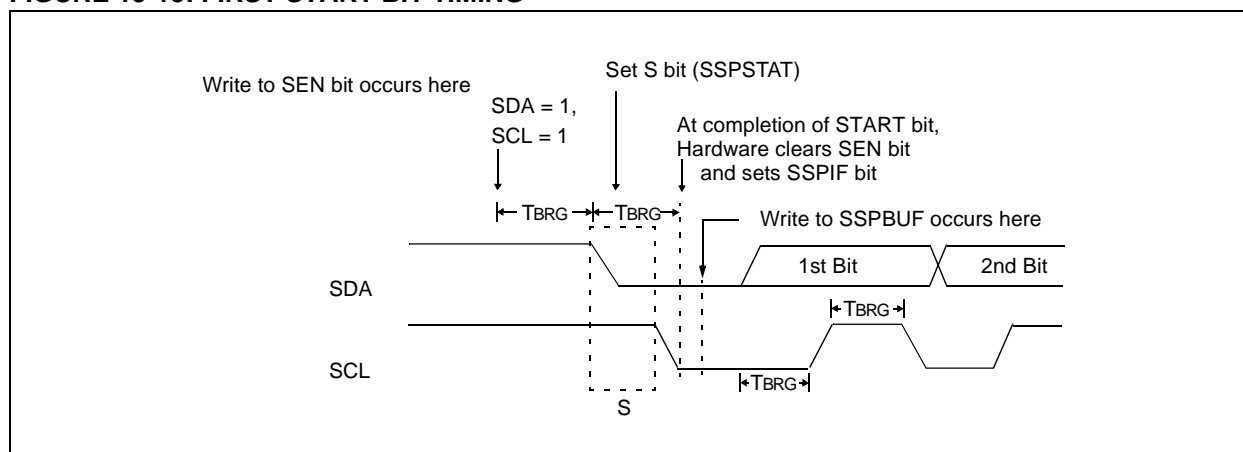
Note: If at the beginning of the START condition, the SDA and SCL pins are already sampled low, or if during the START condition the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag BCLIF is set, the START condition is aborted, and the 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 queuing 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".

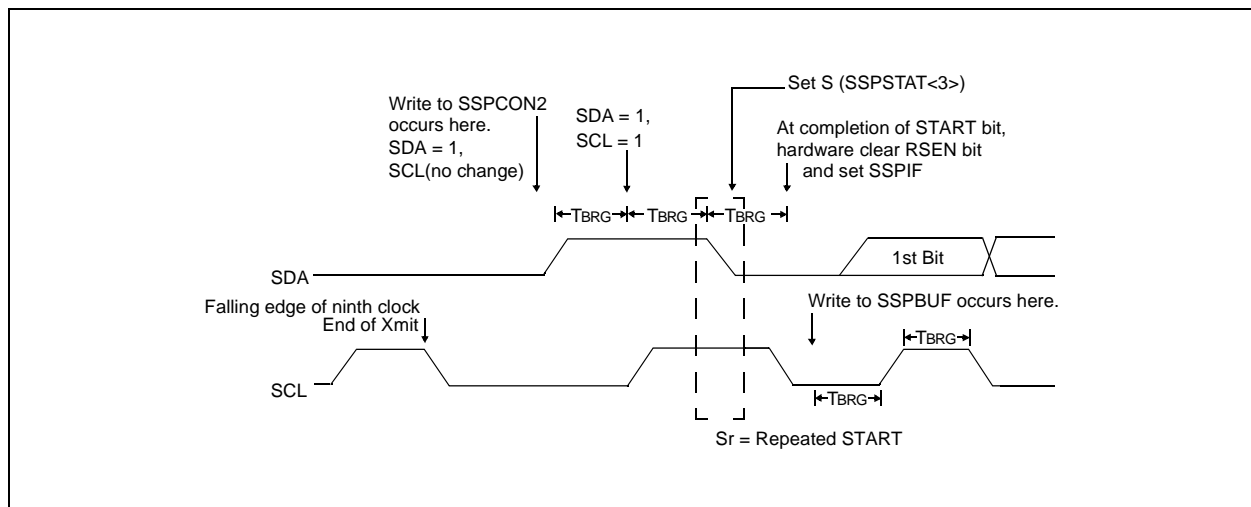
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) 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: REPEAT 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 roll over 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 $\overline{\text{ACK}}$ bit during the ninth bit time if an address match occurs, or if data was received properly. The status of $\overline{\text{ACK}}$ 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 R/W bit, are completed. On the falling edge of the eighth clock, the master will de-assert 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 ($\overline{\text{ACK}} = 0$), and is set when the slave does not acknowledge ($\overline{\text{ACK}} = 1$). 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: I²C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

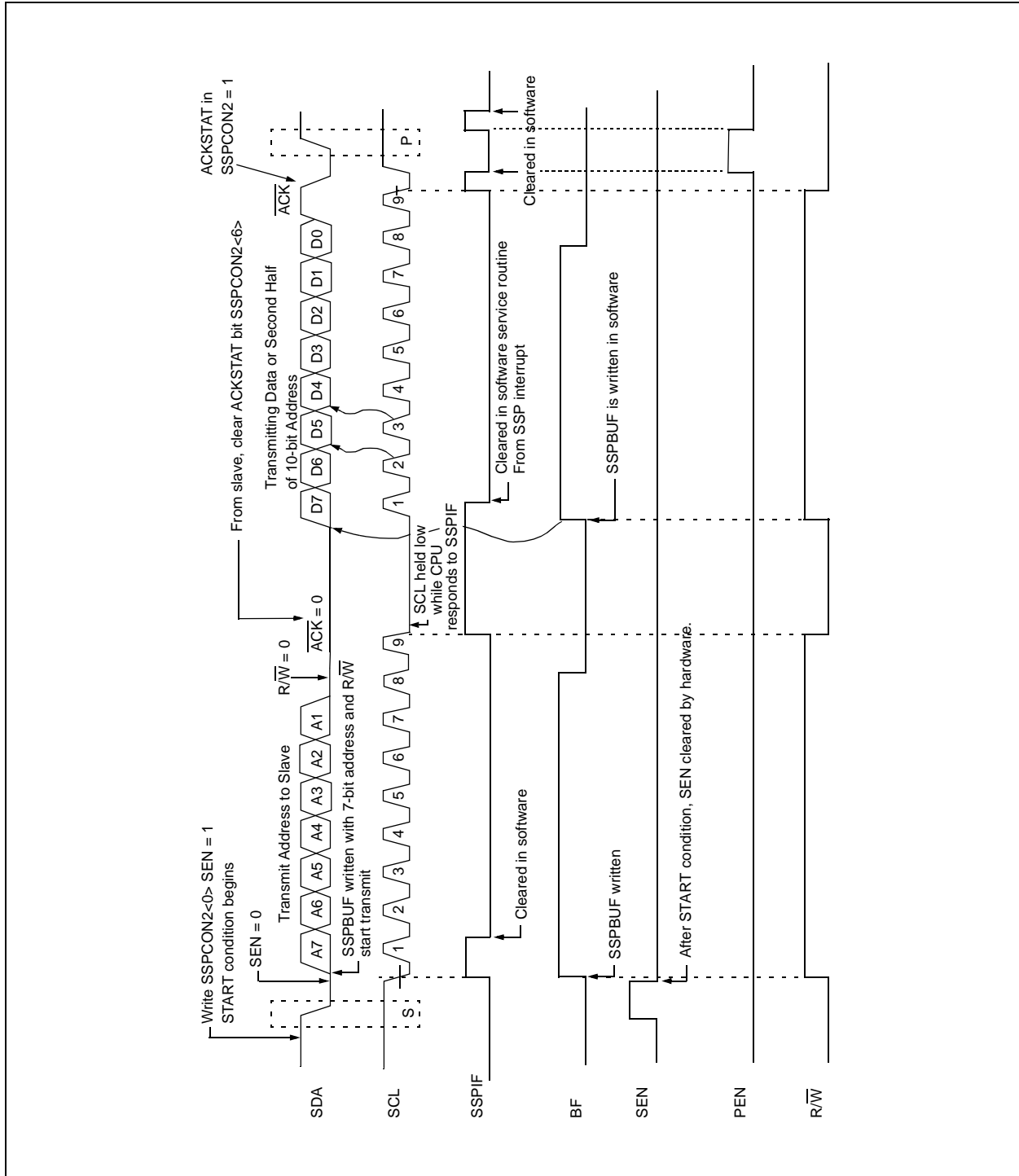
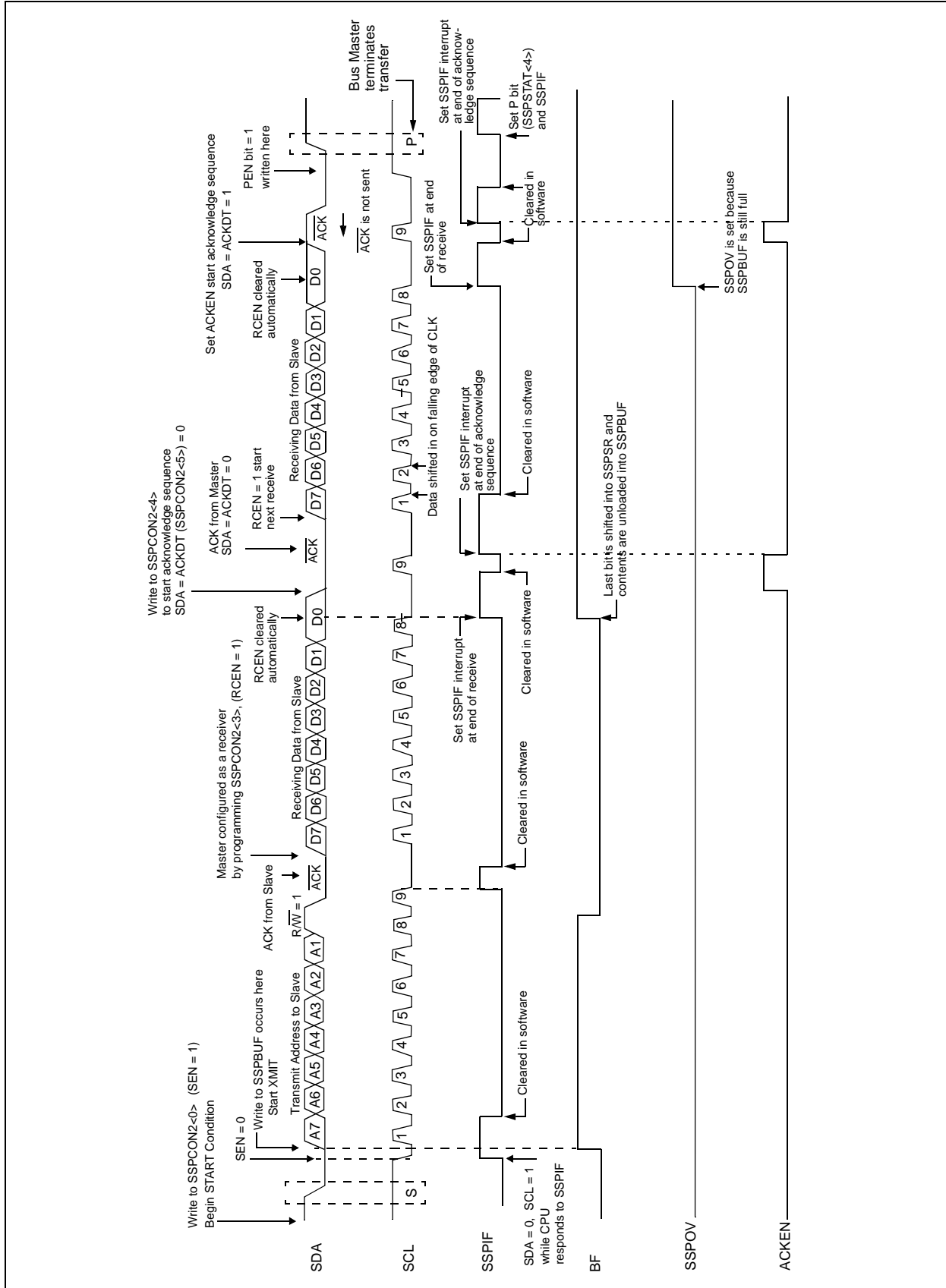


FIGURE 15-16: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



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

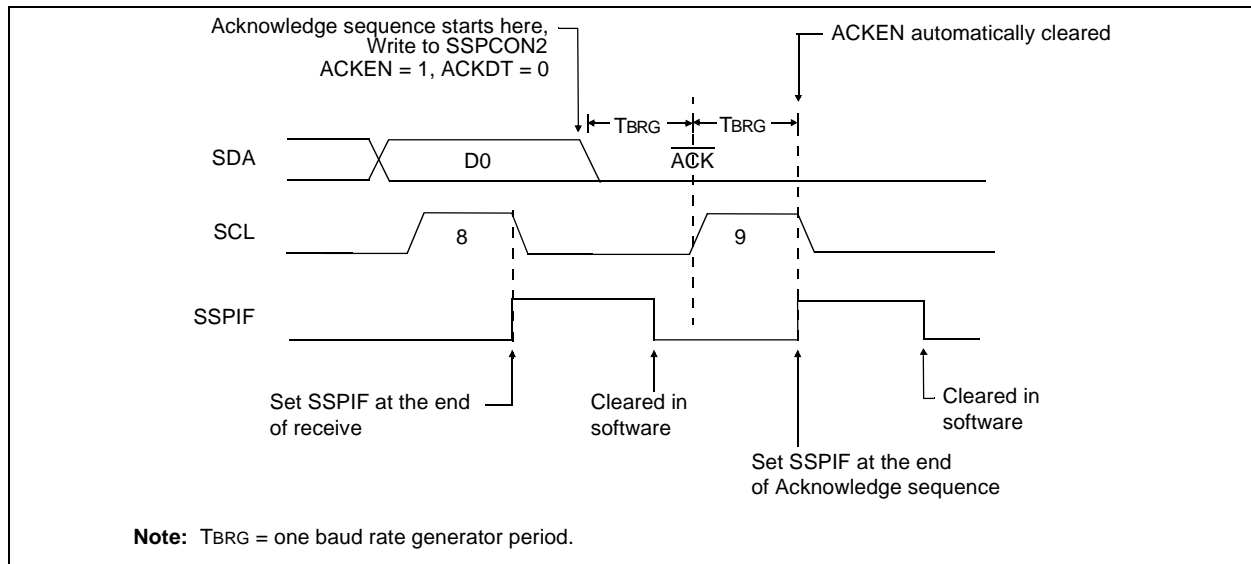
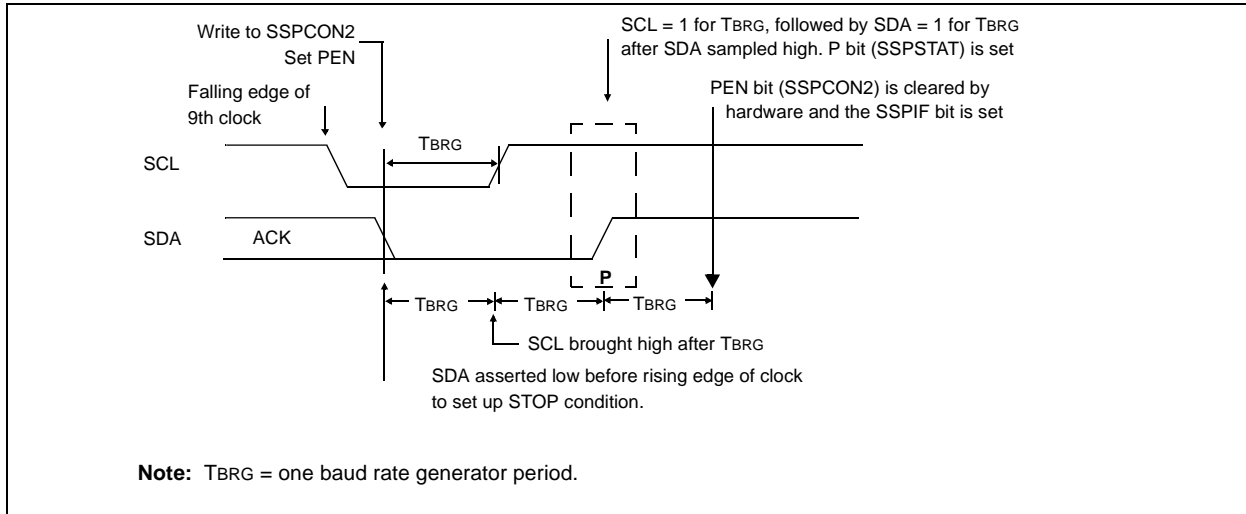


FIGURE 15-18: STOP CONDITION RECEIVE OR TRANSMIT MODE



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).

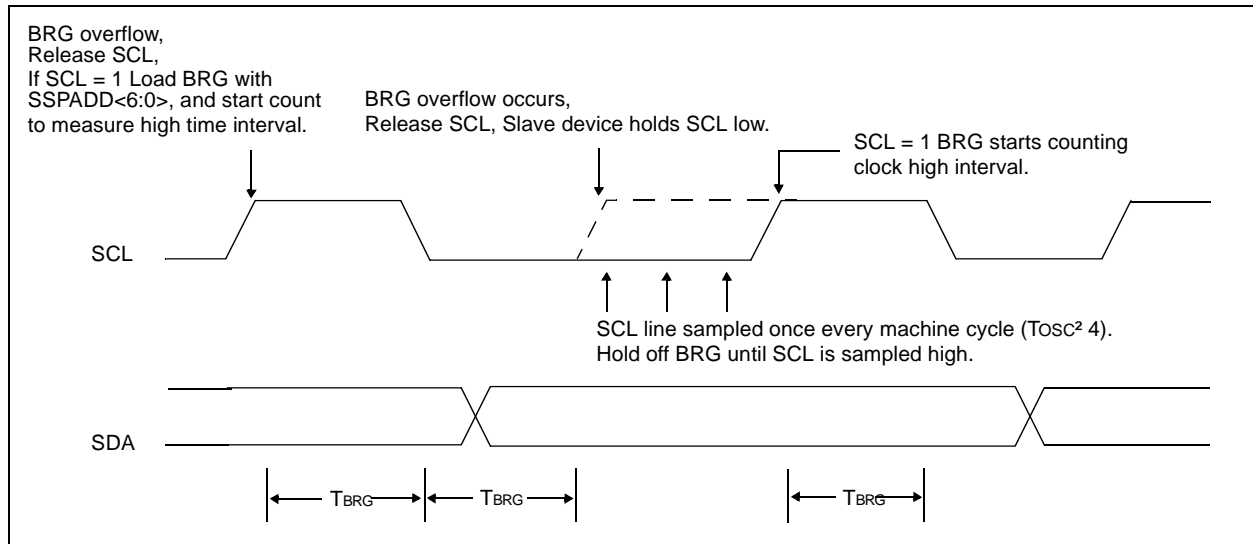
15.4.13 SLEEP OPERATION

While in SLEEP mode, the I²C 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 I²C 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 I²C 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 I²C bus is free, the user can resume communication by asserting a START condition.

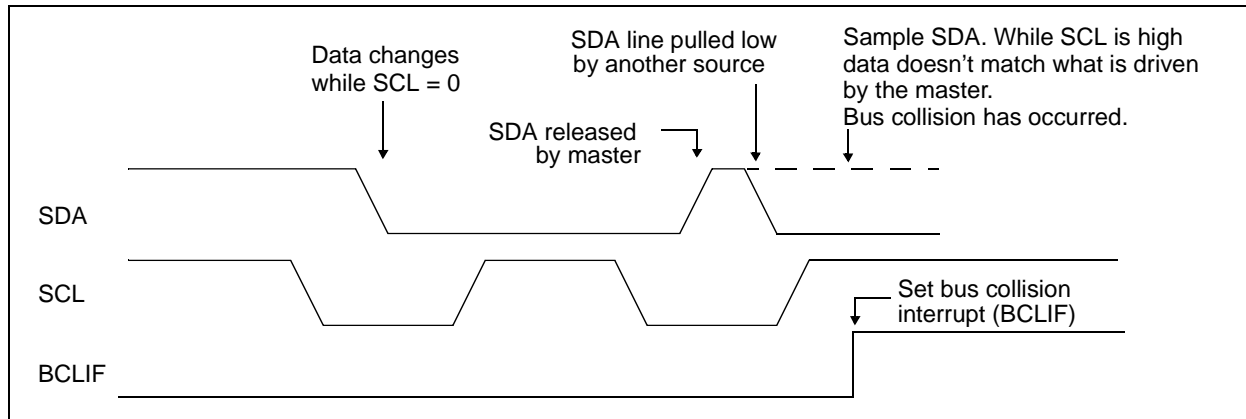
If a START, Repeated START, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted, and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision interrupt service routine, and if the I²C 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 I²C 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:

- SDA or SCL are sampled low at the beginning of the START condition (Figure 15-21).
- 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:

the START condition is aborted,
and 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)

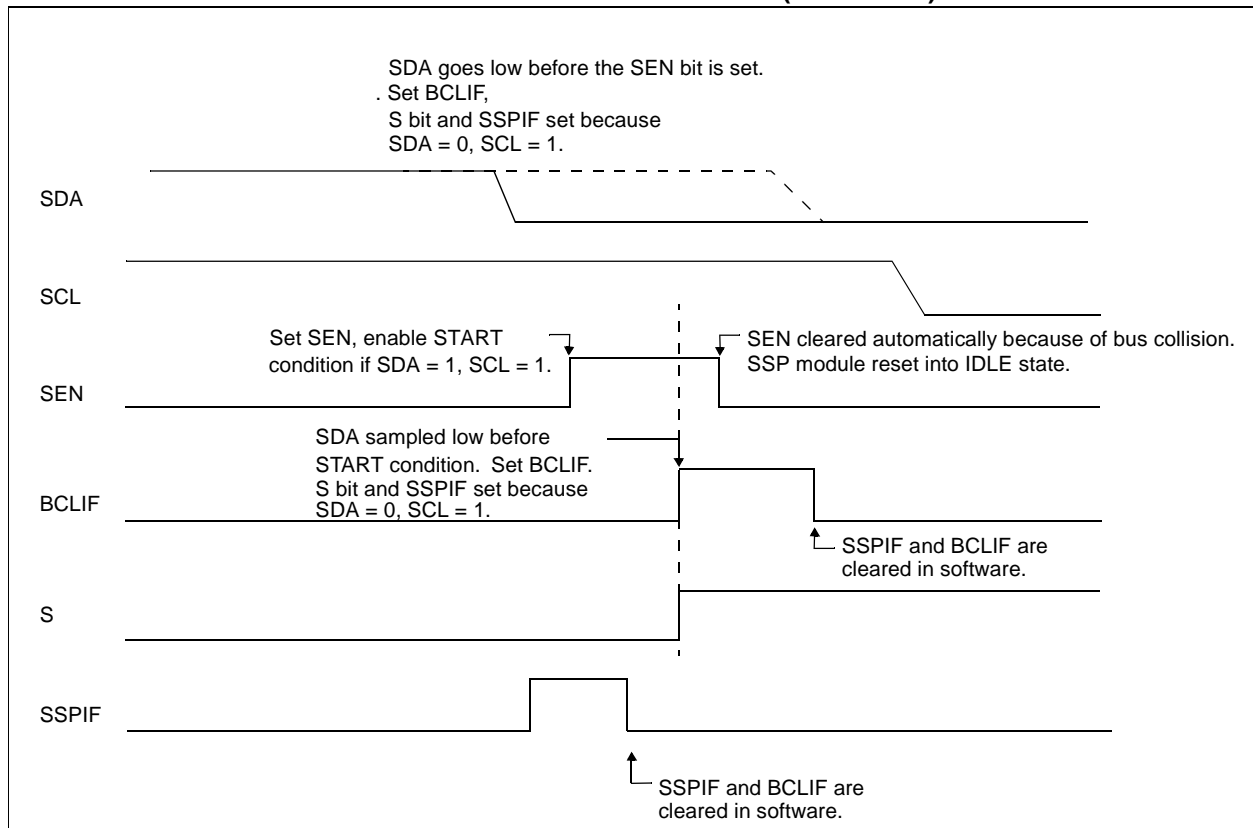


FIGURE 15-22: BUS COLLISION DURING START CONDITION (SCL = 0)

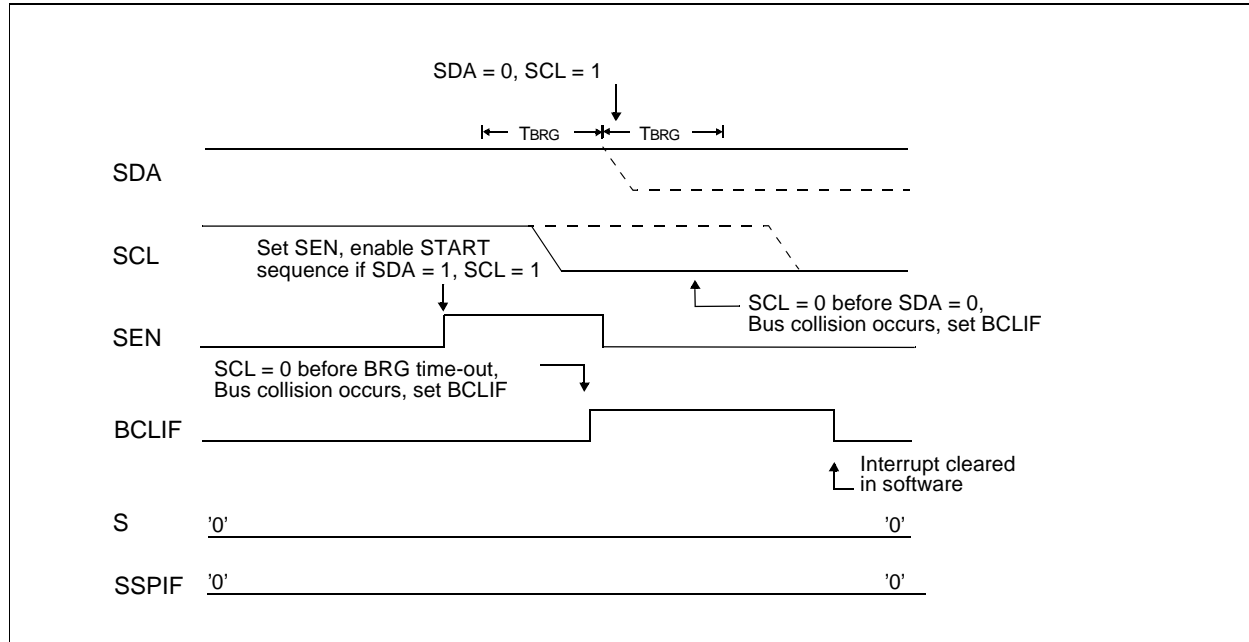
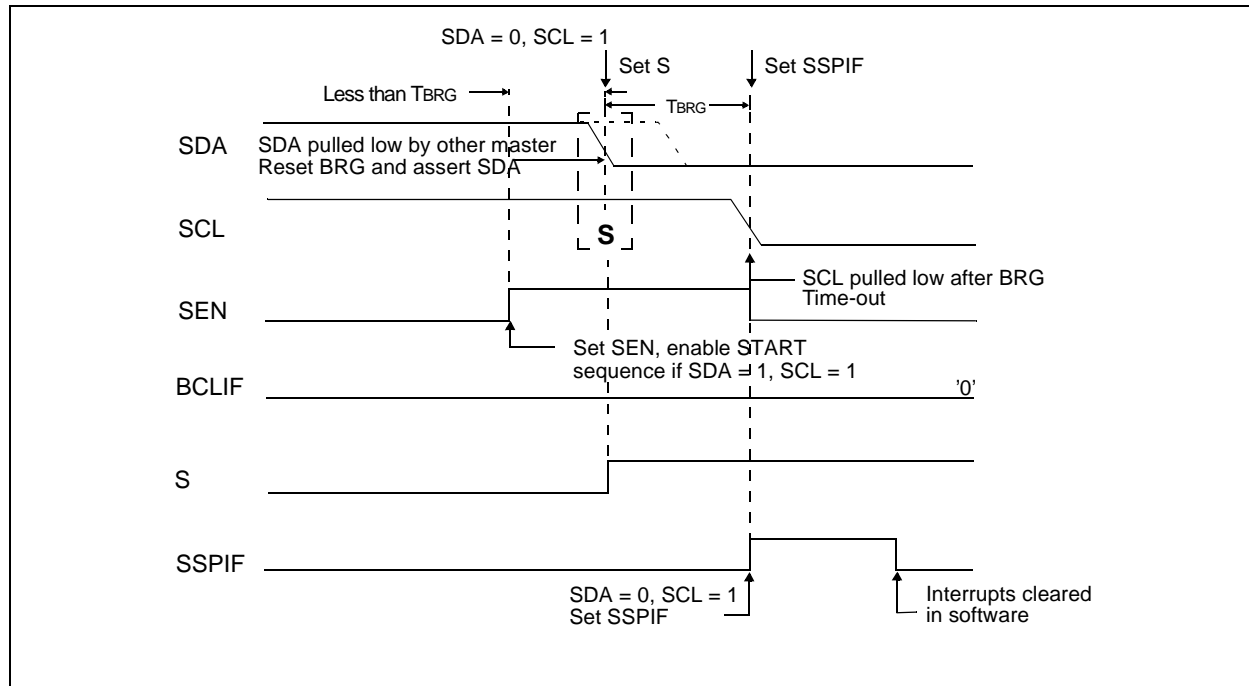


FIGURE 15-23: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



15.4.16.2 Bus Collision During a Repeated START Condition

During a Repeated START condition, a bus collision occurs if:

- A low level is sampled on SDA when SCL goes from low level to high level.
- 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)

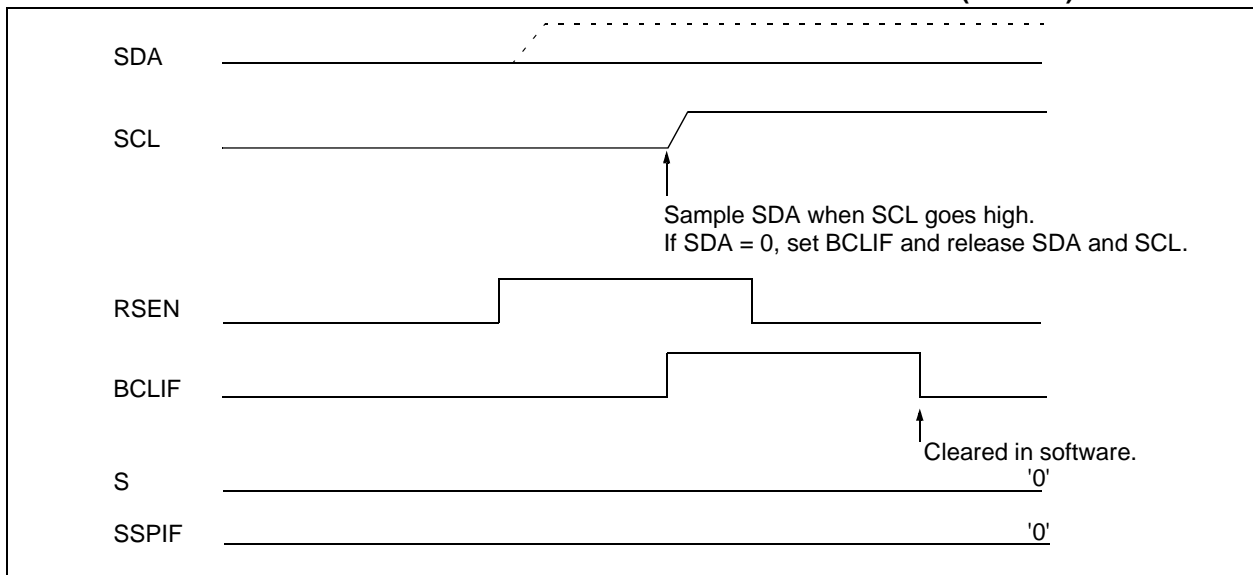
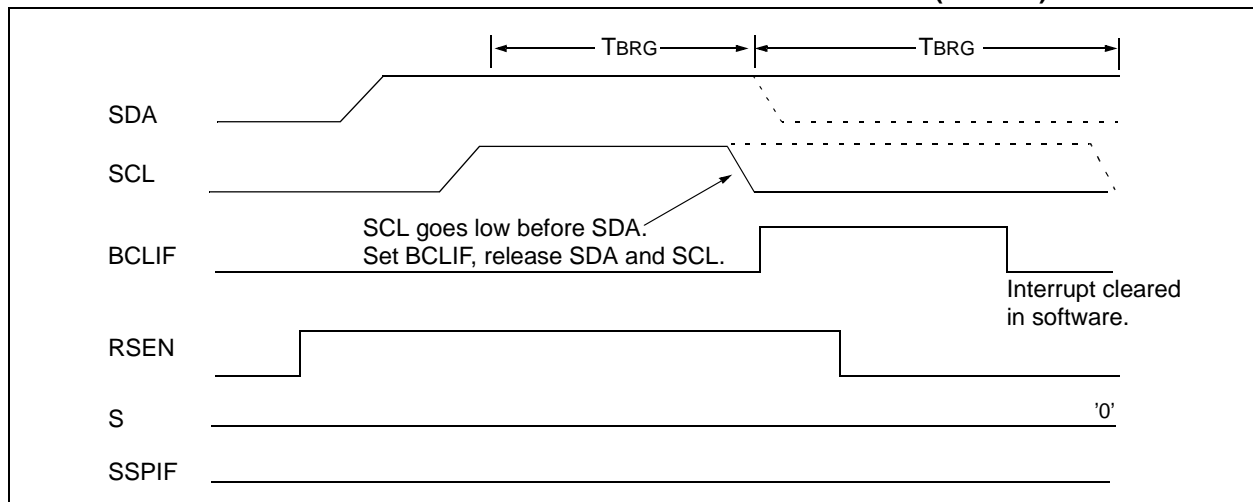


FIGURE 15-25: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



PIC18CXX8

15.4.16.3 Bus Collision During a STOP Condition

Bus collision occurs during a STOP condition if:

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

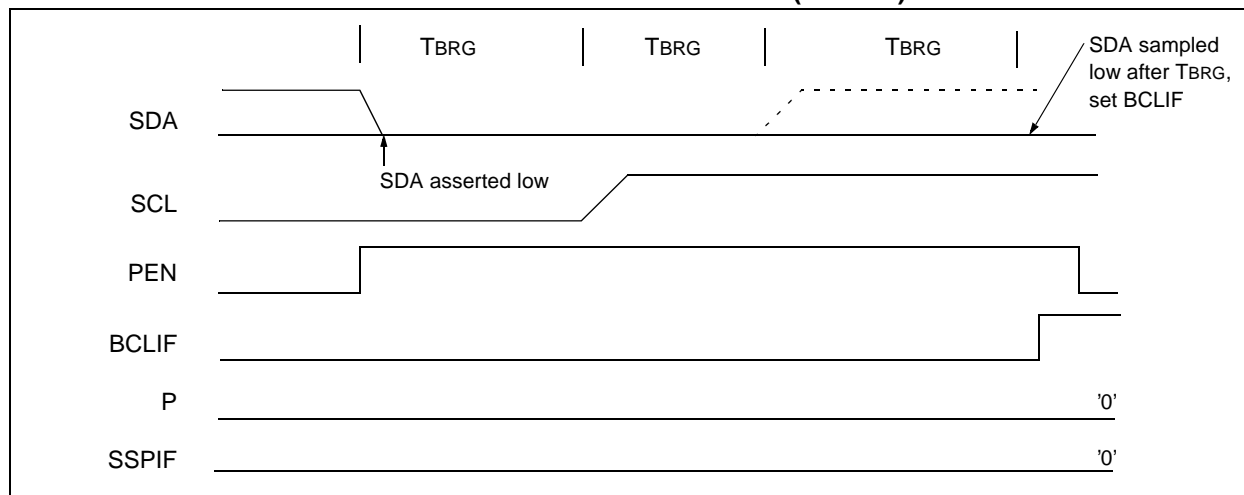
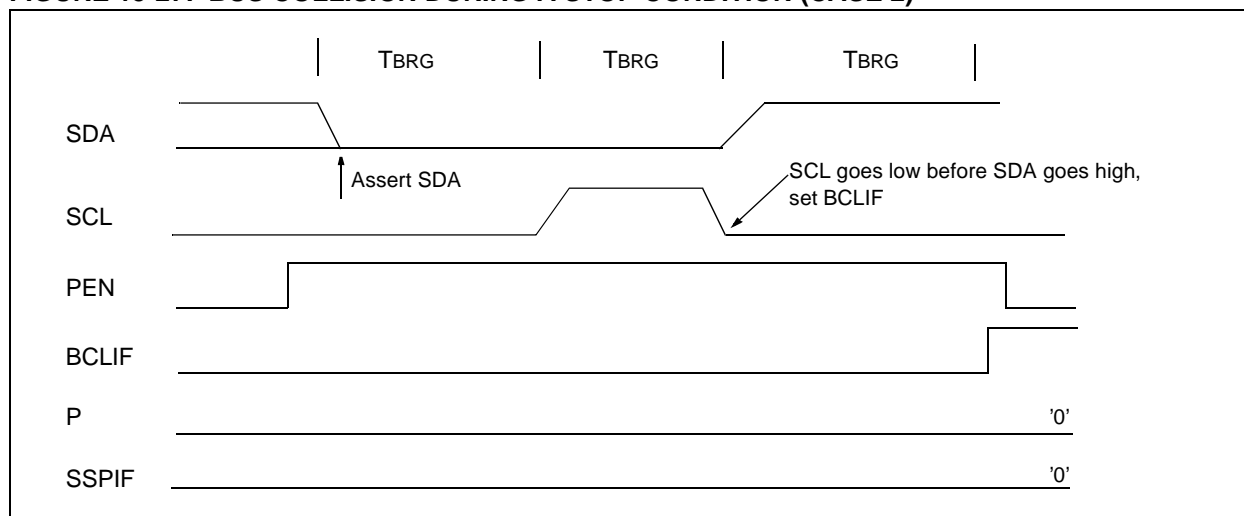


FIGURE 15-27: BUS COLLISION DURING A STOP CONDITION (CASE 2)



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 (RXSTA).

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 | <p>CSRC: Clock Source Select bit</p> <p><u>Asynchronous mode</u> Don't care</p> <p><u>Synchronous mode</u> 1 = Master mode (Clock generated internally from BRG) 0 = Slave mode (Clock from external source)</p> |
| bit 6 | <p>TX9: 9-bit Transmit Enable bit</p> <p>1 = Selects 9-bit transmission 0 = Selects 8-bit transmission</p> |
| bit 5 | <p>TXEN: Transmit Enable bit</p> <p>1 = Transmit enabled 0 = Transmit disabled</p> <p>Note: SREN/CREN overrides TXEN in SYNC mode.</p> |
| bit 4 | <p>SYNC: USART Mode Select bit</p> <p>1 = Synchronous mode 0 = Asynchronous mode</p> |
| bit 3 | <p>Unimplemented: Read as '0'</p> |
| bit 2 | <p>BRGH: High Baud Rate Select bit</p> <p><u>Asynchronous mode</u> 1 = High speed 0 = Low speed</p> <p><u>Synchronous mode</u> Unused in this mode</p> |
| bit 1 | <p>TRMT: Transmit Shift Register Status bit</p> <p>1 = TSR empty 0 = TSR full</p> |
| bit 0 | <p>TX9D: 9th bit of transmit data. Can be Address/Data bit or a parity bit.</p> |

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 |

PIC18CXX8

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 $F_{osc}/(16(X + 1))$ 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 | = | $F_{osc} / (64 (X + 1))$ |
| Solving for X: | | |
| X | = | $((F_{osc} / \text{Desired Baud Rate}) / 64) - 1$ |
| X | = | $((16000000 / 9600) / 64) - 1$ |
| X | = | $[25.042] = 25$ |
| Calculated Baud Rate | = | $16000000 / (64 (25 + 1))$ |
| | = | 9615 |
| Error | = | $\frac{(\text{Calculated Baud Rate} - \text{Desired Baud Rate})}{\text{Desired Baud Rate}}$ |
| | = | $(9615 - 9600) / 9600$ |
| | = | 0.16% |

TABLE 16-1: BAUD RATE FORMULA

| SYNC | BRGH = 0 (Low Speed) | BRGH = 1 (High Speed) |
|------|------------------------------------------------|---------------------------------|
| 0 | (Asynchronous) Baud Rate = $F_{osc}/(64(X+1))$ | Baud Rate = $F_{osc}/(16(X+1))$ |
| 1 | (Synchronous) Baud Rate = $F_{osc}/(4(X+1))$ | 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.

PIC18CXX8

TABLE 16-3: BAUD RATES FOR SYNCHRONOUS MODE

| BAUD RATE (Kbps) | Fosc = 40 MHz | | | 33 MHz | | | 25 MHz | | | 20 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 | - | - | NA | - | - | NA | - | - |
| 19.2 | NA | - | - | NA | - | - | NA | - | - | NA | - | - |
| 76.8 | 76.92 | +0.16 | 129 | 77.10 | +0.39 | 106 | 77.16 | +0.47 | 80 | 76.92 | +0.16 | 64 |
| 96 | 96.15 | +0.16 | 103 | 95.93 | -0.07 | 85 | 96.15 | +0.16 | 64 | 96.15 | +0.16 | 51 |
| 300 | 303.03 | +1.01 | 32 | 294.64 | -1.79 | 27 | 297.62 | -0.79 | 20 | 294.12 | -1.96 | 16 |
| 500 | 500 | 0 | 19 | 485.30 | -2.94 | 16 | 480.77 | -3.85 | 12 | 500 | 0 | 9 |
| HIGH | 10000 | - | 0 | 8250 | - | 0 | 6250 | - | 0 | 5000 | - | 0 |
| LOW | 39.06 | - | 255 | 32.23 | - | 255 | 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 | 316.80 | +5.60 | 3 |
| 500 | 500 | 0 | 7 | 500 | 0 | 4 | 447.44 | -10.51 | 3 | 422.40 | -15.52 | 2 |
| 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 | 26 |
| 1.2 | NA | - | - | NA | - | - | 1.20 | +0.16 | 207 | 1.17 | -2.48 | 6 |
| 2.4 | NA | - | - | NA | - | - | 2.40 | +0.16 | 103 | 2.73 | +13.78 | 2 |
| 9.6 | 9.62 | +0.16 | 103 | 9.62 | +0.23 | 92 | 9.62 | +0.16 | 25 | 8.20 | -14.67 | 0 |
| 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 | 83.33 | +8.51 | 2 | NA | - | - |
| 96 | 1000 | +4.17 | 9 | 99.43 | +3.57 | 8 | 83.33 | -13.19 | 2 | NA | - | - |
| 300 | 333.33 | +11.11 | 2 | 298.30 | -0.57 | 2 | 250 | -16.67 | 0 | NA | - | - |
| 500 | 500 | 0 | 1 | 447.44 | -10.51 | 1 | 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 = 40 MHz | | | 33 MHz | | | 25 MHz | | | 20 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 | - | - | 2.40 | -0.07 | 214 | 2.40 | -0.15 | 162 | 2.40 | +0.16 | 129 |
| 9.6 | 9.62 | +0.16 | 64 | 9.55 | -0.54 | 53 | 9.53 | -0.76 | 40 | 9.47 | -1.36 | 32 |
| 19.2 | 18.94 | -1.36 | 32 | 19.10 | -0.54 | 26 | 19.53 | +1.73 | 19 | 19.53 | +1.73 | 15 |
| 76.8 | 78.13 | +1.73 | 7 | 73.66 | -4.09 | 6 | 78.13 | +1.73 | 4 | 78.13 | +1.73 | 3 |
| 96 | 89.29 | -6.99 | 6 | 103.13 | +7.42 | 4 | 97.66 | +1.73 | 3 | 104.17 | +8.51 | 2 |
| 300 | 312.50 | +4.17 | 1 | 257.81 | -14.06 | 1 | NA | - | - | 312.50 | +4.17 | 0 |
| 500 | 625 | +25.00 | 0 | NA | - | - | NA | - | - | NA | - | - |
| HIGH | 625 | - | 0 | 515.63 | - | 0 | 390.63 | - | 0 | 312.50 | - | 0 |
| LOW | 2.44 | - | 255 | 2.01 | - | 255 | 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 | 83.33 | +8.51 | 2 | 78.13 | +1.73 | 1 | 111.86 | +45.65 | 0 | 79.20 | +3.13 | 0 |
| 96 | 83.33 | -13.19 | 2 | 78.13 | -18.62 | 1 | NA | - | - | NA | - | - |
| 300 | 250 | -16.67 | 0 | 156.25 | -47.92 | 0 | 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 | 207 | 0.30 | +0.23 | 185 | 0.30 | +0.16 | 51 | 0.26 | -14.67 | 1 |
| 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 | 2.23 | -6.99 | 6 | NA | - | - |
| 9.6 | 8.93 | -6.99 | 6 | 9.32 | -2.90 | 5 | 7.81 | -18.62 | 1 | NA | - | - |
| 19.2 | 20.83 | +8.51 | 2 | 18.64 | -2.90 | 2 | 15.63 | -18.62 | 0 | NA | - | - |
| 76.8 | 62.50 | -18.62 | 0 | 55.93 | -27.17 | 0 | 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 |

PIC18CXX8

TABLE 16-5: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

| BAUD RATE (Kbps) | Fosc = 40 MHz | | | 33 MHz | | | 25 MHz | | | 20 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 | - | - | 9.60 | -0.07 | 214 | 9.59 | -0.15 | 162 | 9.62 | +0.16 | 129 |
| 19.2 | 19.23 | +0.16 | 129 | 19.28 | +0.39 | 106 | 19.30 | +0.47 | 80 | 19.23 | +0.16 | 64 |
| 76.8 | 75.76 | -1.36 | 32 | 76.39 | -0.54 | 26 | 78.13 | +1.73 | 19 | 78.13 | +1.73 | 15 |
| 96 | 96.15 | +0.16 | 25 | 98.21 | +2.31 | 20 | 97.66 | +1.73 | 15 | 96.15 | +0.16 | 12 |
| 300 | 312.50 | +4.17 | 7 | 294.64 | -1.79 | 6 | 312.50 | +4.17 | 4 | 312.50 | +4.17 | 3 |
| 500 | 500 | 0 | 4 | 515.63 | +3.13 | 3 | 520.83 | +4.17 | 2 | 416.67 | -16.67 | 2 |
| HIGH | 2500 | - | 0 | 2062.50 | - | 0 | 1562.50 | - | 0 | 1250 | - | 0 |
| LOW | 9.77 | - | 255 | 8.06 | - | 255 | 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 | 89.29 | -6.99 | 6 | 89.49 | -6.78 | 4 | 105.60 | +10.00 | 2 |
| 300 | 333.33 | +11.11 | 2 | 312.50 | +4.17 | 1 | 447.44 | +49.15 | 0 | 316.80 | +5.60 | 0 |
| 500 | 500 | 0 | 1 | 625 | +25.00 | 0 | 447.44 | -10.51 | 0 | 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 | 1.02 | -14.67 | 1 |
| 2.4 | 2.40 | +0.16 | 103 | 2.41 | +0.23 | 92 | 2.40 | +0.16 | 25 | 2.05 | -14.67 | 0 |
| 9.6 | 9.62 | +0.16 | 25 | 9.73 | +1.32 | 22 | 8.93 | -6.99 | 6 | NA | - | - |
| 19.2 | 19.23 | +0.16 | 12 | 18.64 | -2.90 | 11 | 20.83 | +8.51 | 2 | NA | - | - |
| 76.8 | NA | - | - | 74.57 | -2.90 | 2 | 62.50 | -18.62 | 0 | NA | - | - |
| 96 | NA | - | - | 111.86 | +16.52 | 1 | NA | - | - | NA | - | - |
| 300 | NA | - | - | 223.72 | -25.43 | 0 | 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, the USART uses standard non-return-to-zero (NRZ) format (one START bit, eight or nine data bits and one STOP bit). The most common data format is 8-bits. An on-chip dedicated 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The USART's transmitter and receiver are functionally independent, but use the same data format and 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). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data 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 T_{CY}), 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.

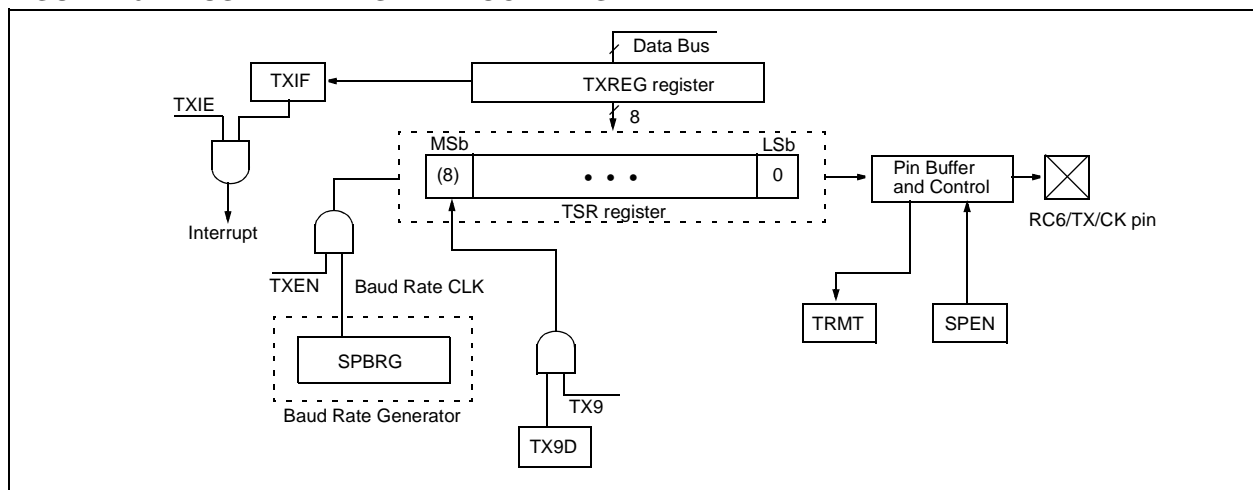
Note 1: The TSR register is not mapped in data memory, so it is not available to the user.

Note 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



PIC18CXX8

FIGURE 16-2: ASYNCHRONOUS TRANSMISSION

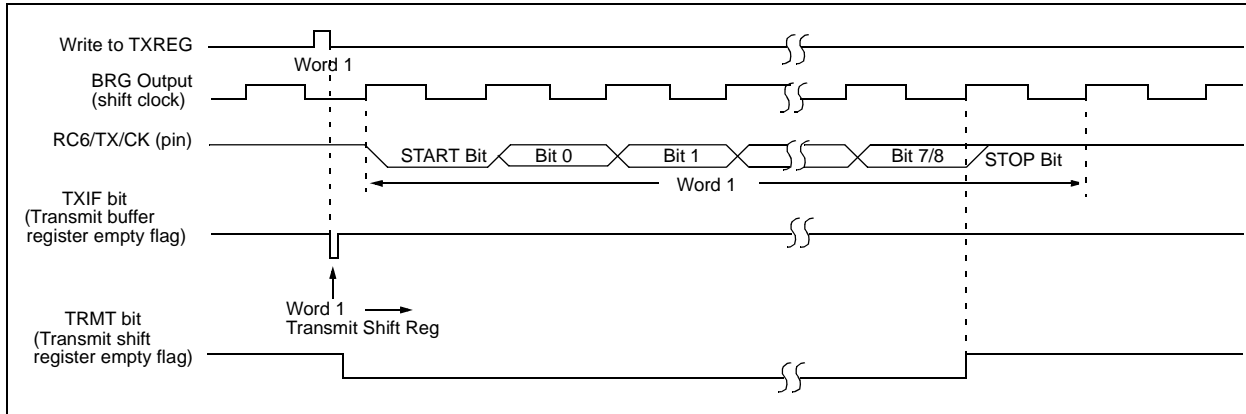


FIGURE 16-3: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)

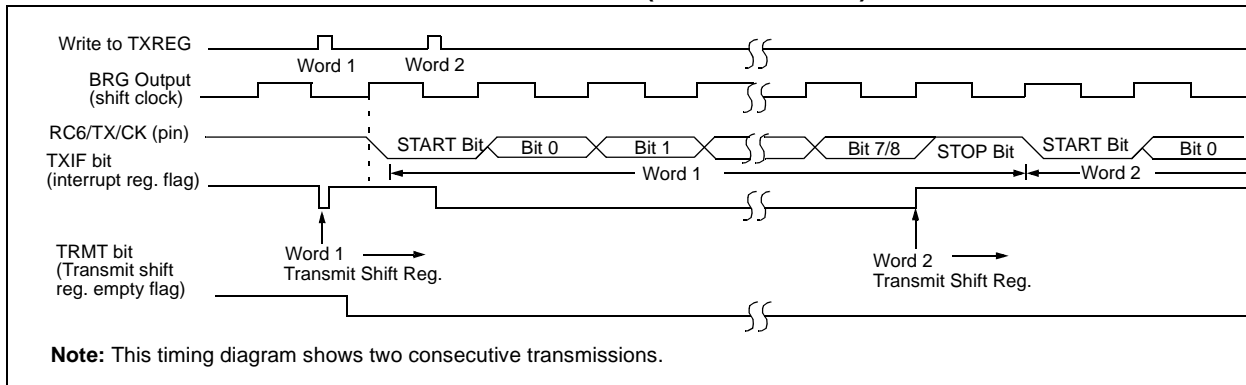


TABLE 16-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

| 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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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

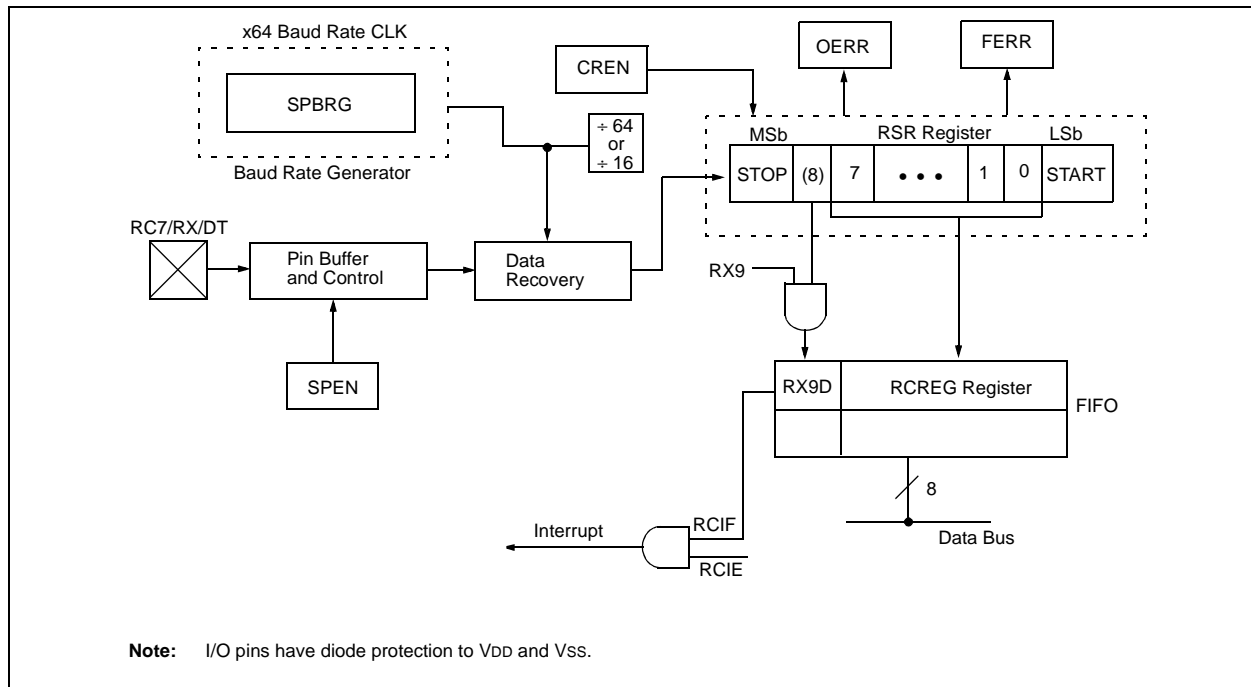


FIGURE 16-5: ASYNCHRONOUS RECEPTION

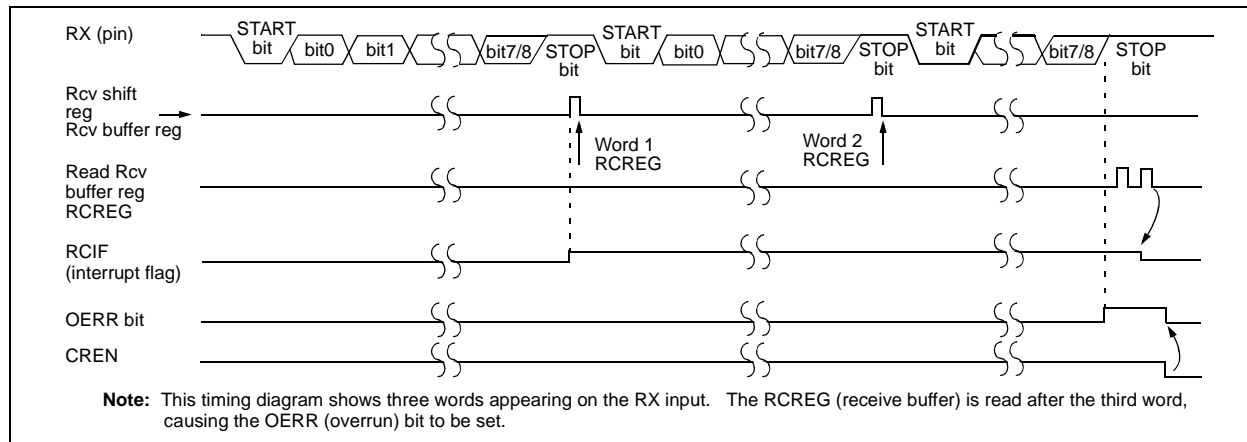


TABLE 16-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

| 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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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 | RBF | 0000 000x | 0000 000u |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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

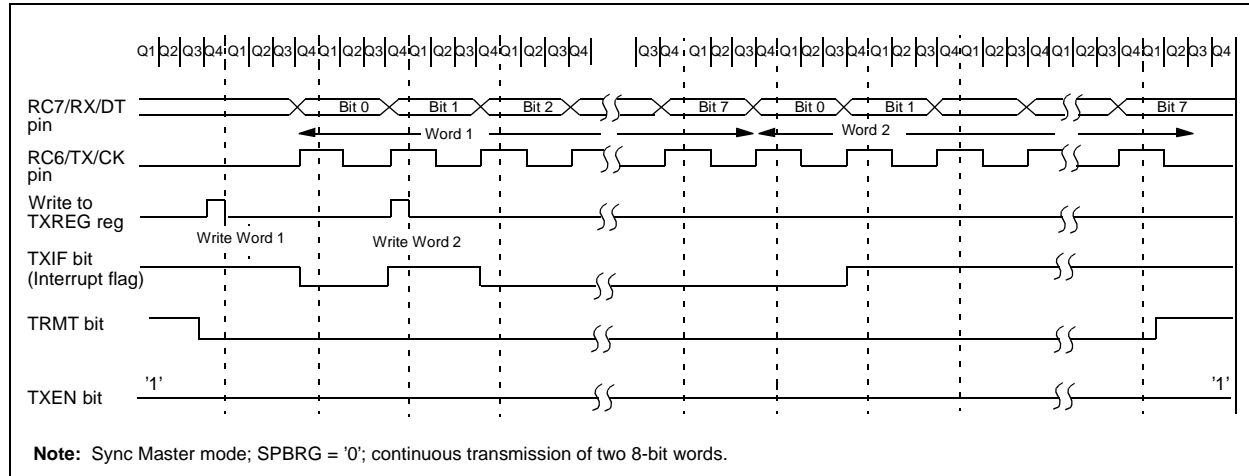
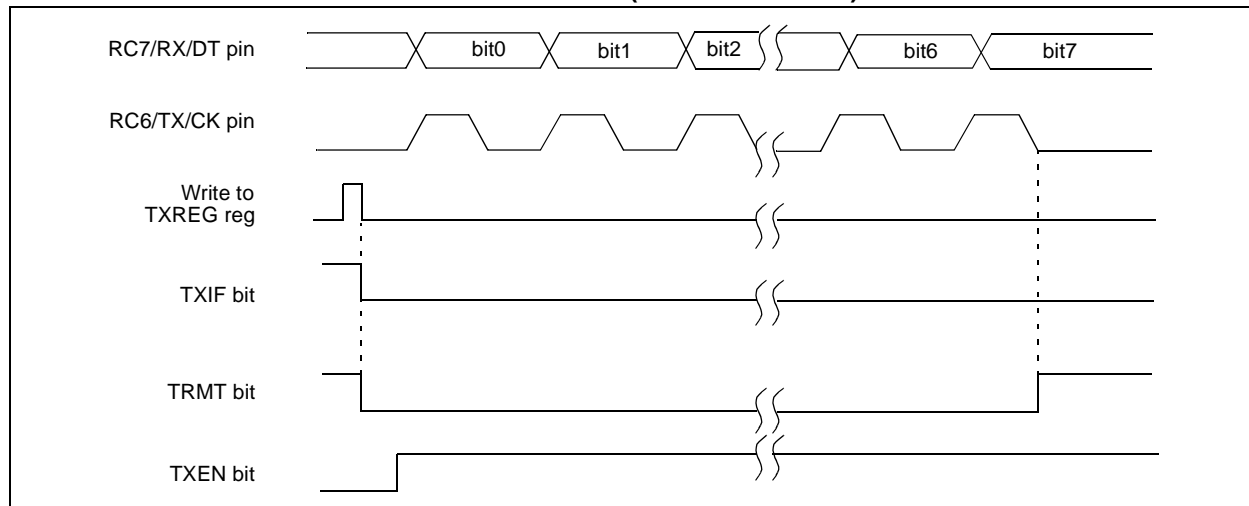


FIGURE 16-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



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.

Steps to follow when setting up a Synchronous Master Reception:

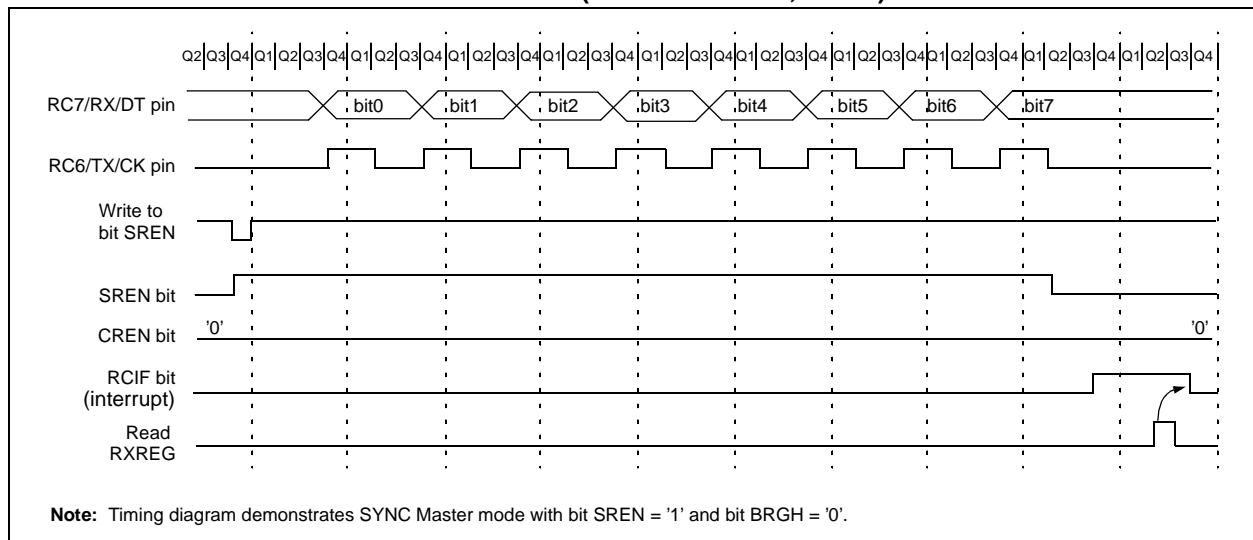
1. Initialize the SPBRG register for the appropriate baud rate (Section 16.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, set enable bit RCIE.
5. If 9-bit reception is desired, set bit RX9.
6. 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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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)



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.

Steps to follow when setting up a Synchronous Slave Transmission:

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.

Steps to follow when setting up a Synchronous Slave Reception:

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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 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.

PIC18CXX8

NOTES:

17.0 CAN MODULE

17.1 Overview

The Controller Area Network (CAN) module is a serial interface, useful for communicating with other peripherals or microcontroller devices. This interface/protocol was designed to allow communications within noisy environments.

The CAN module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN2.0B Passive, and CAN 2.0B Active versions of the protocol. The module implementation is a Full CAN system. The CAN specification is not covered within this data sheet. The reader may refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol CAN1.2, CAN2.0A and CAN2.0B
- Standard and extended data frames
- 0 - 8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Support for remote frames
- Double buffered receiver with two prioritized received message storage buffers
- 6 full (standard/extended identifier) acceptance filters, 2 associated with the high priority receive buffer, and 4 associated with the low priority receive buffer
- 2 full acceptance filter masks, one each associated with the high and low priority receive buffers
- Three transmit buffers with application specified prioritization and abort capability
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low power SLEEP mode

17.1.1 OVERVIEW OF THE MODULE

The CAN bus module consists of a Protocol Engine and message buffering and control. The CAN protocol engine handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the 2 receive registers.

The CAN Module supports the following Frame types:

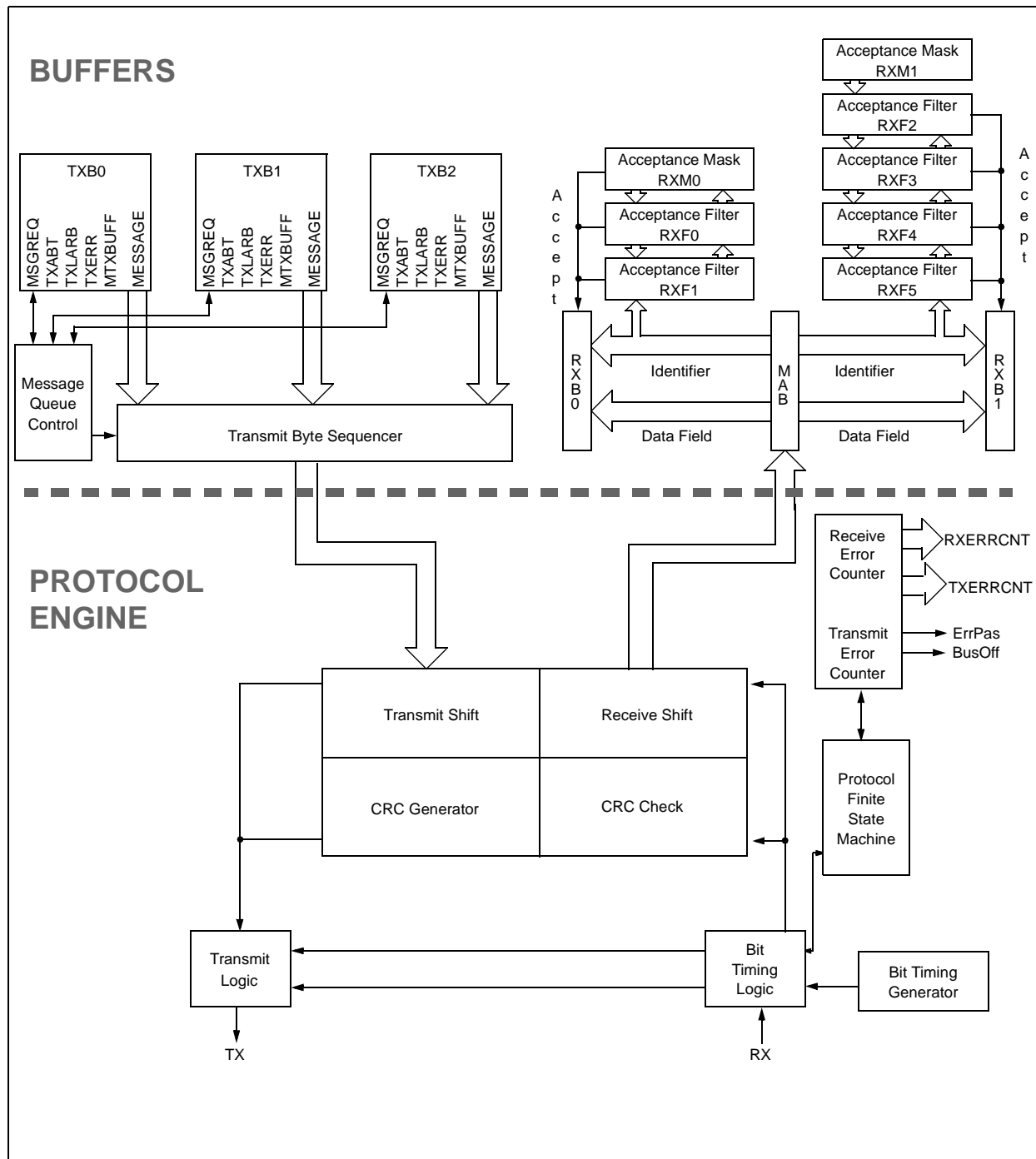
- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception
- Interframe Space

PIC18CXX8

17.1.2 TRANSMIT/RECEIVE BUFFERS

The PIC18CXX8 has three transmit and two receive buffers, two acceptance masks (one for each receive buffer), and a total of six acceptance filters. Figure 17-1 is a block diagram of these buffers and their connection to the protocol engine.

FIGURE 17-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM



17.2 Control Registers for the CAN Module

17.2.1 CAN CONTROL AND STATUS REGISTERS

Note: Not all CAN registers are available in the access bank.

This section shows the CAN Control and Status registers.

There are many registers associated with the CAN module. Descriptions of these registers are grouped into sections. These sections are:

- Control and Status Registers
- Transmit Buffer Registers
- Receive Buffer Registers
- Baud Rate Control Registers
- Interrupt Status and Control Registers

REGISTER 17-1: CANCON – CAN CONTROL REGISTER

| | | | | | | | |
|--------|--------|--------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
| REQOP2 | REQOP1 | REQOP0 | ABAT | WIN2 | WIN1 | WIN0 | — |
| | | | | | | | bit 0 |
| | | | | | | | bit 7 |

bit 7-5 **REQOP2:REQOP0:** Request CAN Operation Mode bits

1xx = Request Configuration mode
 011 = Request Listen Only mode
 010 = Request Loopback mode
 001 = Request Disable mode
 000 = Request Normal mode

bit 4 **ABAT:** Abort All Pending Transmissions bit
 1 = Abort all pending transmissions (in all transmit buffers)
 0 = Transmissions proceeding as normal

bit 3-1 **WIN2:WIN0:** Window Address bits

This selects which of the CAN buffers to switch into the access bank area. This allows access to the buffer registers from any data memory bank. After a frame has caused an interrupt, the ICODE2:ICODE0 bits can be copied to the WIN2:WIN0 bits to select the correct buffer. See Example 17-1 for code example.

111 = Receive Buffer 0
 110 = Receive Buffer 0
 101 = Receive Buffer 1
 100 = Transmit Buffer 0
 011 = Transmit Buffer 1
 010 = Transmit Buffer 2
 001 = Receive Buffer 0
 000 = Receive Buffer 0

bit 0 **Unimplemented:** Read 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 |

PIC18CXX8

REGISTER 17-2: CANSTAT – CAN STATUS REGISTER

| | | | | | | | |
|---------|---------|---------|-----|--------|--------|--------|-------|
| R-1 | R-0 | R-0 | U-0 | R-0 | R-0 | R-0 | U-0 |
| OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — |
| bit 7 | | | | | | | bit 0 |

bit 7-5 **OPMODE2:OPMODE0:** Operation Mode Status bits

- 111 = Reserved
- 110 = Reserved
- 101 = Reserved
- 100 = Configuration mode
- 011 = Listen Only mode
- 010 = Loopback mode
- 001 = Disable mode
- 000 = Normal mode

Note: Before the device goes into SLEEP mode, select Disable mode.

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

bit 3-1 **ICODE2:ICODE0:** Interrupt Code bits

When an interrupt occurs, a prioritized coded interrupt value will be present in the ICODE2:ICODE0 bits. These codes indicate the source of the interrupt. The ICODE2:ICODE0 bits can be copied to the WIN2:WIN0 bits to select the correct buffer to map into the Access Bank area. See Example 17-1 for code example.

- 111 = Wake-up on Interrupt
- 110 = RXB0 Interrupt
- 101 = RXB1 Interrupt
- 100 = TXB0 Interrupt
- 011 = TXB1 Interrupt
- 010 = TXB2 Interrupt
- 001 = Error Interrupt
- 000 = No Interrupt

bit 0 **Unimplemented:** Read 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 |

EXAMPLE 17-1: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS

```

; Save application required context.
; Poll interrupt flags and determine source of interrupt
; This was found to be CAN interrupt
; TempCANCON and TempCANSTAT are variables defined in Access Bank low
movff  CANCON, TempCANCON          ; Save CANCON.WIN bits
                                        ; This is required to prevent CANCON
                                        ; from corrupting CAN buffer access
                                        ; in-progress while this interrupt
                                        ; occurred

movff  CANSTAT, TempCANSTAT        ; Save CANSTAT register
                                        ; This is required to make sure that
                                        ; we use same CANSTAT value rather
                                        ; than one changed by another CAN
                                        ; interrupt.

movf   TempCANSTAT, W              ; Retrieve ICODE bits
andlw  b'00001110'
addwf  PCL, F                     ; Perform computed GOTO
                                        ; to corresponding interrupt cause

bra    NoInterrupt                ; 000 = No interrupt
bra    ErrorInterrupt            ; 001 = Error interrupt
bra    TXB2Interrupt            ; 010 = TXB2 interrupt
bra    TXB1Interrupt            ; 011 = TXB1 interrupt
bra    TXB0Interrupt            ; 100 = TXB0 interrupt
bra    RXB1Interrupt            ; 101 = RXB1 interrupt
bra    RXB0Interrupt            ; 110 = RXB0 interrupt
                                        ; 111 = Wake-up on interrupt

WakeupInterrupt
  bcf   PIR3, WAKIF              ; Clear the interrupt flag
  ;
  ; User code to handle wake-up procedure
  ;
  ;
  ; Continue checking for other interrupt source or return from here
  ...

NoInterrupt
  ...                            ; PC should never vector here. User may
                                        ; place a trap such as infinite loop or pin/port
                                        ; indication to catch this error.

ErrorInterrupt
  bcf   PIR3, ERRIF              ; Clear the interrupt flag
  ...                            ; Handle error.
  retfie

TXB2Interrupt
  bcf   PIR3, TXB2IF            ; Clear the interrupt flag
  goto  AccessBuffer

TXB1Interrupt
  bcf   PIR3, TXB1IF            ; Clear the interrupt flag
  goto  AccessBuffer

TXB0Interrupt
  bcf   PIR3, TXB0IF            ; Clear the interrupt flag
  goto  AccessBuffer

RXB1Interrupt
  bcf   PIR3, RXB1IF            ; Clear the interrupt flag
  goto  Accessbuffer

```

PIC18CXX8

```
RXB0Interrupt
    bcf    PIR3, RXB0IF          ; Clear the interrupt flag
    goto  AccessBuffer

AccessBuffer                          ; This is either TX or RX interrupt
; Copy CANCON.ICODE bits to CANSTAT.WIN bits
    movf  TempCANCON, W          ; Clear CANCON.WIN bits before copying
                                      ; new ones.
    andlw b'11110001'           ; Use previously saved CANCON value to
                                      ; make sure same value.
    movwf TempCANCON            ; Copy masked value back to TempCANCON

    movf  TempCANSTAT, W        ; Retrieve ICODE bits
    andlw b'00001110'         ; Use previously saved CANSTAT value
                                      ; to make sure same value.

    iorwf TempCANCON            ; Copy ICODE bits to WIN bits.
    movff TempCANCON, CANCON    ; Copy the result to actual CANCON

; Access current buffer...
; Your code

; Restore CANCON.WIN bits
    movf  CANCON, W             ; Preserve current non WIN bits
    andlw b'11110001'
    iorwf TempCANCON          ; Restore original WIN bits

; Do not need to restore CANSTAT - it is read-only register.
; Return from interrupt or check for another module interrupt source
```

REGISTER 17-3: COMSTAT – COMMUNICATION STATUS REGISTER

| R/C-0 | R/C-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|----------|----------|------|------|------|--------|--------|-------|
| RXB0OVFL | RXB1OVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| bit 7 | | | | | | | bit 0 |

- bit 7 **RXB0OVFL:** Receive Buffer 0 Overflow bit
 1 = Receive Buffer 0 overflowed
 0 = Receive Buffer 0 has not overflowed
- bit 6 **RXB1OVFL:** Receive Buffer 1 Overflow bit
 1 = Receive Buffer 1 overflowed
 0 = Receive Buffer 1 has not overflowed
- bit 5 **TXBO:** Transmitter Bus Off bit
 1 = Transmit Error Counter >255
 0 = Transmit Error Counter ≤ 255
- bit 4 **TXBP:** Transmitter Bus Passive bit
 1 = Transmission Error Counter >127
 0 = Transmission Error Counter ≤127
- bit 3 **RXBP:** Receiver Bus Passive bit
 1 = Receive Error Counter >127
 0 = Receive Error Counter ≤127
- bit 2 **TXWARN:** Transmitter Warning bit
 1 = Transmit Error Counter >95
 0 = Transmit Error Counter ≤95
- bit 1 **RXWARN:** Receiver Warning bit
 1 = Receive Error Counter >95
 0 = Receive Error Counter ≤ 95
- bit 0 **EWARN:** Error Warning bit
 This bit is a flag of the RXWARN and TXWARN bits
 1 = The RXWARN or the TXWARN bits are set
 0 = Neither the RXWARN or the TXWARN bits are set

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 |

PIC18CXX8

17.2.2 CAN TRANSMIT BUFFER REGISTERS

This section describes the CAN Transmit Buffer Register and the associated Transmit Buffer Control Registers.

REGISTER 17-4: TXBnCON – TRANSMIT BUFFER n CONTROL REGISTER

| | | | | | | | |
|-----|-------|--------|-------|-------|-----|--------|--------|
| U-0 | R-0 | R-0 | R-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
| — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 |

bit 7

bit 0

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

bit 6 **TXABT:** Transmission Aborted Status bit

1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit

1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit

1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQ:** Transmit Request Status bit

1 = Requests sending a message. Clears the TXABT, TLARB, and TXERR bits
0 = Automatically cleared when the message is successfully sent

Note: Clearing this bit in software, while the bit is set, will request a message abort.

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

bit 1-0 **TXPRI1:TXPRI0:** Transmit Priority bits

11 = Priority Level 3 (Highest Priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (Lowest Priority)

Note: These bits set the order in which Transmit buffer will be transferred. They do not alter CAN message identifier.

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 17-5: TXBnSIDH: TRANSMIT BUFFER n STANDARD IDENTIFIER HIGH BYTE REGISTER

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |

bit 7

bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits, if EXIDE = 0 (TXBnSID Register).

Extended Identifier bits EID28:EID21, if EXIDE = 1.

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 17-6: TXBnSIDL – TRANSMIT BUFFER n STANDARD IDENTIFIER LOW BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | |
| bit 7 | | | | | | | | bit 0 |

- bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXIDE = 0.
Extended Identifier bits EID20:EID18, if EXIDE = 1.
- bit 4 **Unimplemented:** Read as '0'
- bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Message will transmit Extended ID, SID10:SID0 becomes EID28:EID18
0 = Message will transmit Standard ID, EID17:EID0 are ignored
- bit 2 **Unimplemented:** Read as '0'
- bit 1-0 **EID17:EID16:** Extended Identifier 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 |

REGISTER 17-7: TXBnEIDH – TRANSMIT BUFFER n EXTENDED IDENTIFIER HIGH BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | |
| bit 7 | | | | | | | | bit 0 |

- bit 7-0 **EID15:EID8:** Extended Identifier 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 |

REGISTER 17-8: TXBnEIDL – TRANSMIT BUFFER n EXTENDED IDENTIFIER LOW BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | |
| bit 7 | | | | | | | | bit 0 |

- bit 7-0 **EID7:EID0:** Extended Identifier 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 |

REGISTER 17-9: TXBnDm – TRANSMIT BUFFER n DATA FIELD BYTE m REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TXBnDm7 | TXBnDm6 | TXBnDm5 | TXBnDm4 | TXBnDm3 | TXBnDm2 | TXBnDm1 | TXBnDm0 |
| bit 7 | | | | | | | bit 0 |

bit 1-0 **TXBnDm7:TXBnDm0:** Transmit Buffer n Data Field Byte m bits (where $0 \leq n < 3$ and $0 < m < 8$)
 Each Transmit Buffer has an array of registers. For example, Transmit buffer 0 has 7 registers: TXB0D0 to TXB0D7.

| | | | |
|--------------------|------------------|------------------------------------|--------------------|
| 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 17-10: TXBnDLC – TRANSMIT BUFFER n DATA LENGTH CODE REGISTER

| U-0 | R/W-x | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-----|-----|-------|-------|-------|-------|
| — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 |
| bit 7 | | | | bit 0 | | | |

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

bit 6 **TXRTR:** Transmission Frame Remote Transmission Request bit
 1 = Transmitted message will have TXRTR bit set
 0 = Transmitted message will have TXRTR bit cleared.

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

bit 3-0 **DLC3:DLC0:** Data Length Code bits
 1111 = Reserved
 1110 = Reserved
 1101 = Reserved
 1100 = Reserved
 1011 = Reserved
 1010 = Reserved
 1001 = Reserved
 1000 = Data Length = 8 bytes
 0111 = Data Length = 7 bytes
 0110 = Data Length = 6 bytes
 0101 = Data Length = 5 bytes
 0100 = Data Length = 4 bytes
 0011 = Data Length = 3 bytes
 0010 = Data Length = 2 bytes
 0001 = Data Length = 1 bytes
 0000 = Data Length = 0 bytes

| | | | |
|--------------------|------------------|------------------------------------|--------------------|
| 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 17-11: TXERRCNT – TRANSMIT ERROR COUNT REGISTER

| | | | | | | | |
|-------|------|------|------|------|------|-------|------|
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| TEC7 | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0 |
| bit 7 | | | | | | bit 0 | |

bit 7-0

TEC7:TEC0: Transmit Error Counter bits

This register contains a value which is derived from the rate at which errors occur. When the error count overflows, the bus off state occurs. When the bus has 128 occurrences of 11 consecutive recessive bits, the counter value is cleared.

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

PIC18CXX8

17.2.3 CAN RECEIVE BUFFER REGISTERS

This section shows the Receive Buffer registers with its associated control registers.

REGISTER 17-12: RXB0CON – RECEIVE BUFFER 0 CONTROL REGISTER

| | | | | | | | |
|-------|-------|-------|-----|---------|----------|-------|---------|
| R/C-0 | R/W-0 | R/W-0 | U-0 | R-0 | R/W-0 | R-0 | R/W-0 |
| RXFUL | RXM1 | RXM0 | — | RXRTRRO | RXB0DBEN | JTOFF | FILHIT0 |

bit 7

bit 0

bit 7 **RXFUL:** Receive Full Status bit
 1 = Receive buffer contains a received message
 0 = Receive buffer is open to receive a new message

Note: This bit is set by the CAN module and should be cleared by software after the buffer is read.

bit 6-5 **RXM1:RXM0:** Receive Buffer Mode bits
 11 = Receive all messages (including those with errors)
 10 = Receive only valid messages with extended identifier
 01 = Receive only valid messages with standard identifier
 00 = Receive all valid messages

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

bit 3 **RXRTRRO:** Receive Remote Transfer Request Read Only bit
 1 = Remote transfer request
 0 = No remote transfer request

bit 2 **RXB0DBEN:** Receive Buffer 0 Double Buffer Enable bit
 1 = Receive Buffer 0 overflow will write to Receive Buffer 1
 0 = No Receive Buffer 0 overflow to Receive Buffer 1

bit 1 **JTOFF:** Jump Table Offset bit (read only copy of RX0DBEN)
 1 = Allows Jump Table offset between 6 and 7
 0 = Allows Jump Table offset between 1 and 0

Note: This bit allows same filter jump table for both RXB0CON and RXB1CON.

bit 0 **FILHIT0:** Filter Hit bit
 This bit indicates which acceptance filter enabled the message reception into receive buffer 0
 1 = Acceptance Filter 1 (RXF1)
 0 = Acceptance Filter 0 (RXF0)

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 17-13: RXB1CON – RECEIVE BUFFER 1 CONTROL REGISTER

| | | | | | | | |
|-------|-------|-------|-----|---------|---------|---------|---------|
| R/C-0 | R/W-0 | R/W-0 | U-0 | R-0 | R-0 | R-0 | R-0 |
| RXFUL | RXM1 | RXM0 | — | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 |
| bit 7 | | | | bit 0 | | | |

bit 7 **RXFUL:** Receive Full Status bit
 1 = Receive buffer contains a received message
 0 = Receive buffer is open to receive a new message

Note: This bit is set by the CAN module and should be cleared by software after the buffer is read.

bit 6-5 **RXM1:RXM0:** Receive Buffer Mode bits
 11 = Receive all messages (including those with errors)
 10 = Receive only valid messages with extended identifier
 01 = Receive only valid messages with standard identifier
 00 = Receive all valid messages

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

bit 3 **RXRTRRO:** Receive Remote Transfer Request bit (read only)
 1 = Remote transfer request
 0 = No remote transfer request

bit 2-0 **FILHIT2:FILHIT0:** Filter Hit bits
 These bits indicate which acceptance filter enabled the last message reception into Receive Buffer 1.
 111 = Reserved
 110 = Reserved
 101 = Acceptance Filter 5 (RXF5)
 100 = Acceptance Filter 4 (RXF4)
 011 = Acceptance Filter 3 (RXF3)
 010 = Acceptance Filter 2 (RXF2)
 001 = Acceptance Filter 1 (RXF1) only possible when RXB0DBEN bit is set
 000 = Acceptance Filter 0 (RXF0) only possible when RXB0DBEN bit is set

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 17-14: RXBnSIDH – RECEIVE BUFFER n STANDARD IDENTIFIER HIGH BYTE REGISTER

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | bit 0 | | | |

bit 7-0 **SID10:SID3:** Standard Identifier bits, if EXID = 0 (RXBnSIDL Register).
 Extended Identifier bits EID28:EID21, if EXID = 1.

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

PIC18CXX8

REGISTER 17-15: RXBnSIDL – RECEIVE BUFFER n STANDARD IDENTIFIER LOW BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | U-0 | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-----|-------|-------|
| SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 |

bit 7 bit 0

- bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXID = 0.
Extended Identifier bits EID20:EID18, if EXID = 1.
- bit 4 **SRR:** Substitute Remove Request bit (only when EXID = '1')
1 = Remote transfer request occurred
0 = No remote transfer request occurred
- bit 3 **EXID:** Extended Identifier bit
1 = Received message is an Extended Data Frame, SID10:SID0 are EID28:EID18
0 = Received message is a Standard Data Frame
- bit 2 **Unimplemented:** Read as '0'
- bit 1-0 **EID17:EID16:** Extended Identifier 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 |

REGISTER 17-16: RXBnEIDH – RECEIVE BUFFER n EXTENDED IDENTIFIER HIGH BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |

bit 7 bit 0

- bit 7-0 **EID15:EID8:** Extended Identifier 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 |

REGISTER 17-17: RXBnEIDL – RECEIVE BUFFER n EXTENDED IDENTIFIER LOW BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |

bit 7 bit 0

- bit 7-0 **EID7:EID0:** Extended Identifier 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 |

REGISTER 17-18: RXBnDLC – RECEIVE BUFFER n DATA LENGTH CODE REGISTER

| U-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 |
| bit 7 | | | | | | | bit 0 |

- bit 7 **Unimplemented:** Read as '0'
- bit 6 **RXRTR:** Receiver Remote Transmission Request bit
1 = Remote transfer request
0 = No remote transfer request
- bit 5 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as '0'
- bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as '0'
- bit 3-0 **DLC3:DLC0:** Data Length Code bits
1111 = Invalid
1110 = Invalid
1101 = Invalid
1100 = Invalid
1011 = Invalid
1010 = Invalid
1001 = Invalid
1000 = Data Length = 8 bytes
0111 = Data Length = 7 bytes
0110 = Data Length = 6 bytes
0101 = Data Length = 5 bytes
0100 = Data Length = 4 bytes
0011 = Data Length = 3 bytes
0010 = Data Length = 2 bytes
0001 = Data Length = 1 bytes
0000 = Data Length = 0 bytes

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 17-19: RXBnDm – RECEIVE BUFFER n DATA FIELD BYTE m REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---------|---------|---------|---------|---------|---------|---------|---------|
| RXBnDm7 | RXBnDm6 | RXBnDm5 | RXBnDm4 | RXBnDm3 | RXBnDm2 | RXBnDm1 | RXBnDm0 |
| bit 7 | | | | | | | bit 0 |

- bit 7-0 **RXBnDm7:RXBnDm0:** Receive Buffer n Data Field Byte m bits (where $0 \leq n < 1$ and $0 < m < 7$)
Each Receive Buffer has an array of registers. For example, Receive buffer 0 has 8 registers: RXB0D0 to RXB0D7.

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 17-20: RXERRCNT – RECEIVE ERROR COUNT REGISTER

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| REC7 | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0 |

bit 7 bit 0

bit 7-0

REC7:REC0: Receive Error Counter bits

This register contains the Receive Error value as defined by the CAN specifications.

When RXERRCNT > 127, the module will go into an error passive state. RXERRCNT does not have the ability to put the module in “Bus Off” 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

17.2.4 MESSAGE ACCEPTANCE FILTERS

This subsection describes the Message Acceptance filters.

REGISTER 17-21: RXFnSIDH – RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER HIGH BYTE

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | | | | bit 0 |

bit 7-0 **SID10:SID3:** Standard Identifier Filter bits, if EXIDEN = 0.
Extended Identifier Filter bits EID28:EID21, if EXIDEN = 1,

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 17-22: RXFnSIDL – RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER LOW BYTE

| R/W-x | R/W-x | R/W-x | U-0 | R/W-x | U-0 | R/W-x | R/W-x |
|-------|-------|-------|-----|--------|-----|-------|-------|
| SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 |
| bit 7 | | | | | | | bit 0 |

bit 7-5 **SID2:SID0:** Standard Identifier Filter bits, if EXIDEN = 0.
Extended Identifier Filter bits EID20:EID18, if EXIDEN = 0.

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

bit 3 **EXIDEN:** Extended Identifier Filter Enable bit
1 = Filter will only accept Extended ID messages
0 = Filter will only accept Standard ID messages

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

bit 1-0 **EID17:EID16:** Extended Identifier Filter 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

REGISTER 17-23: RXFnEIDH – RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER HIGH BYTE

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

bit 7-0 **EID15:EID8:** Extended Identifier Filter 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

PIC18CXX8

REGISTER 17-24: RXFnEIDL – RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER LOW BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |

bit 7 bit 0

bit 7-0 **EID7:EID0:** Extended Identifier Filter 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 | | | | |

REGISTER 17-25: RXMnSIDH – RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK HIGH BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |

bit 7 bit 0

bit 7-0 **SID10:SID3:** Standard Identifier Mask bits, or Extended Identifier Mask bits EID28:EID21

| | | | | | | | |
|--------------------|------------------|------------------------------------|--------------------|--|--|--|--|
| 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 17-26: RXMnSIDL – RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK LOW BYTE REGISTER

| R/W-x | R/W-x | R/W-x | U-0 | U-0 | U-0 | R/W-x | R/W-x |
|-------|-------|-------|-----|-----|-----|-------|-------|
| SID2 | SID1 | SID0 | — | — | — | EID17 | EID16 |

bit 7 bit 0

bit 7-5 **SID2:SID0:** Standard Identifier Mask bits, or Extended Identifier Mask bits EID20:EID18

bit 4-2 **Unimplemented:** Read as '0'

bit 1-0 **EID17:EID16:** Extended Identifier Mask 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 | | | | |

REGISTER 17-27: RXMnEIDH – RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK HIGH BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

bit 1-0 **EID15:EID8:** Extended Identifier Mask 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 |

REGISTER 17-28: RXMnEIDL – RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK LOW BYTE REGISTER

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |
| bit 7 | | | | | | | bit 0 |

bit 1-0 **EID7:EID0:** Extended Identifier Mask 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 |

PIC18CXX8

17.2.5 CAN BAUD RATE REGISTERS

This subsection describes the CAN Baud Rate registers.

REGISTER 17-29: BRGCON1 – BAUD RATE CONTROL REGISTER 1

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |

bit 7

bit 0

bit 7-6 **SJW1:SJW0:** Synchronized Jump Width bits
11 = Synchronization Jump Width Time = 4 x T_Q
10 = Synchronization Jump Width Time = 3 x T_Q
01 = Synchronization Jump Width Time = 2 x T_Q
00 = Synchronization Jump Width Time = 1 x T_Q

bit 5-0 **BRP5:BRP0:** Baud Rate Prescaler bits
111111 = T_Q = (2 x 64)/F_{OSC}
111110 = T_Q = (2 x 63)/F_{OSC}
:
:
000001 = T_Q = (2 x 2)/F_{OSC}
000000 = T_Q = (2 x 1)/F_{OSC}

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: This register is only accessible in Configuration mode.

REGISTER 17-30: BRGCON2 – BAUD RATE CONTROL REGISTER 2

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|----------|-------|---------|---------|---------|--------|--------|--------|
| SEG2PHTS | SAM | SEG1PH2 | SEG1PH1 | SEG1PH0 | PRSEG2 | PRSEG1 | PRSEG0 |
| bit 7 | | | | | | bit 0 | |

- bit 7 **SEG2PHTS:** Phase Segment 2 Time Select bit
 1 = Freely programmable
 0 = Maximum of PHEG1 or Information Processing Time (IPT), whichever is greater
- bit 6 **SAM:** Sample of the CAN Bus Line bit
 1 = Bus line is sampled three times prior to the sample point
 0 = Bus line is sampled once at the sample point
- bit 5-3 **SEG1PH2:SEG1PH0:** Phase Segment 1 bits
 111 = Phase Segment 1 Time = 8 x T_Q
 110 = Phase Segment 1 Time = 7 x T_Q
 101 = Phase Segment 1 Time = 6 x T_Q
 100 = Phase Segment 1 Time = 5 x T_Q
 011 = Phase Segment 1 Time = 4 x T_Q
 010 = Phase Segment 1 Time = 3 x T_Q
 001 = Phase Segment 1 Time = 2 x T_Q
 000 = Phase Segment 1 Time = 1 x T_Q
- bit 2-0 **PRSEG2:PRSEG0:** Propagation Time Select bits
 111 = Propagation Time = 8 x T_Q
 110 = Propagation Time = 7 x T_Q
 101 = Propagation Time = 6 x T_Q
 100 = Propagation Time = 5 x T_Q
 011 = Propagation Time = 4 x T_Q
 010 = Propagation Time = 3 x T_Q
 001 = Propagation Time = 2 x T_Q
 000 = Propagation Time = 1 x T_Q

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: This register is only accessible in Configuration mode.

PIC18CXX8

REGISTER 17-31: BRGCON3 – BAUD RATE CONTROL REGISTER 3

| | | | | | | | |
|-------|--------|-----|-----|-------|---------|---------|---------|
| U-0 | R/W-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
| — | WAKFIL | — | — | — | SEG2PH2 | SEG2PH1 | SEG2PH0 |
| bit 7 | | | | bit 0 | | | |

- bit 7 **Unimplemented:** Read as '0'
- bit 6 **WAKFIL:** Selects CAN Bus Line Filter for Wake-up bit
 1 = Use CAN bus line filter for wake-up
 0 = CAN bus line filter is not used for wake-up
- bit 5-3 **Unimplemented:** Read as '0'
- bit 2-0 **SEG2PH2:SEG2PH0:** Phase Segment 2 Time Select bits
 111 = Phase Segment 2 Time = 8 x T_Q
 110 = Phase Segment 2 Time = 7 x T_Q
 101 = Phase Segment 2 Time = 6 x T_Q
 100 = Phase Segment 2 Time = 5 x T_Q
 011 = Phase Segment 2 Time = 4 x T_Q
 010 = Phase Segment 2 Time = 3 x T_Q
 001 = Phase Segment 2 Time = 2 x T_Q
 000 = Phase Segment 2 Time = 1 x T_Q

Note: Ignored if SEG2PHTS bit is 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

17.2.6 CAN MODULE I/O CONTROL REGISTER

This subsection describes the CAN Module I/O Control register.

REGISTER 17-32: CIOCON – CAN I/O CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|--------|-------|--------|--------|-----|-----|-----|-----|
| TX1SRC | TX1EN | ENDRHI | CANCAP | — | — | — | — |

bit 7

bit 0

- bit 7 **TX1SRC:** CAN TX1 Pin Data Source
 1 = CAN TX1 pin will output the CAN clock
 0 = CAN TX1 pin will output TXD
- bit 6 **TX1EN:** CAN TX1 Pin Enable
 1 = CAN TX1 pin will output TXD or CAN clock
 0 = CAN TX1 pin will have digital I/O function
- bit 5 **ENDRHI:** Enable Drive High
 1 = CAN TX0, CAN TX1 pins will drive VDD when recessive
 0 = CAN TX0, CAN TX1 pins will tri-state when recessive
- bit 4 **CANCAP:** CAN Message Receive Capture Enable
 1 = Enable CAN capture
 0 = Disable CAN capture
- bit 3-0 **Unimplemented:** Read 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 |

PIC18CXX8

17.2.7 CAN INTERRUPT REGISTERS

REGISTER 17-33: PIR3 – PERIPHERAL INTERRUPT FLAG 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 |
|-------|-------|-------|--------|--------|--------|--------|--------|
| IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF | TXB0IF | RXB1IF | RXB0IF |

bit 7 bit 0

- bit 7 **IRXIF:** CAN Invalid Received Message Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus
- bit 6 **WAKIF:** CAN Bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus
- bit 5 **ERRIF:** CAN Bus Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources)
0 = No CAN module errors
- bit 4 **TXB2IF:** CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message, and may be re-loaded
0 = Transmit Buffer 2 has not completed transmission of a message
- bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit
1 = Transmit Buffer 1 has completed transmission of a message, and may be re-loaded
0 = Transmit Buffer 1 has not completed transmission of a message
- bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit
1 = Transmit Buffer 0 has completed transmission of a message, and may be re-loaded
0 = Transmit Buffer 0 has not completed transmission of a message
- bit 1 **RXB1IF:** CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message
- bit 0 **RXB0IF:** CAN Receive Buffer 0 Interrupt Flag bit
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message

| | | | |
|--------------------|------------------|------------------------------------|--------------------|
| 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 17-34: PIE3 – PERIPHERAL INTERRUPT ENABLE 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 |
|-------|-------|-------|--------|--------|--------|--------|--------|
| IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE | TXB0IE | RXB1IE | RXB0IE |
| bit 7 | | | | | | bit 0 | |

- bit 7 **IRXIE:** CAN Invalid Received Message Interrupt Enable bit
 1 = Enable invalid message received interrupt
 0 = Disable invalid message received interrupt
- bit 6 **WAKIE:** CAN Bus Activity Wake-up Interrupt Enable bit
 1 = Enable bus activity wake-up interrupt
 0 = Disable bus activity wake-up interrupt
- bit 5 **ERRIE:** CAN Bus Error Interrupt Enable bit
 1 = Enable CAN bus error interrupt
 0 = Disable CAN bus error interrupt
- bit 4 **TXB2IE:** CAN Transmit Buffer 2 Interrupt Enable bit
 1 = Enable Transmit Buffer 2 interrupt
 0 = Disable Transmit Buffer 2 interrupt
- bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit
 1 = Enable Transmit Buffer 1 interrupt
 0 = Disable Transmit Buffer 1 interrupt
- bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit
 1 = Enable Transmit Buffer 0 interrupt
 0 = Disable Transmit Buffer 0 interrupt
- bit 1 **RXB1IE:** CAN Receive Buffer 1 Interrupt Enable bit
 1 = Enable Receive Buffer 1 interrupt
 0 = Disable Receive Buffer 1 interrupt
- bit 0 **RXB0IE:** CAN Receive Buffer 0 Interrupt Enable bit
 1 = Enable Receive Buffer 0 interrupt
 0 = Disable Receive Buffer 0 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 |

PIC18CXX8

REGISTER 17-35: IPR3 – PERIPHERAL INTERRUPT PRIORITY 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 |
|-------|-------|-------|--------|--------|--------|--------|--------|
| IRXIP | WAKIP | ERRIP | TXB2IP | TXB1IP | TXB0IP | RXB1IP | RXB0IP |

bit 7

bit 0

- bit 7 **IRXIP:** CAN Invalid Received Message Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 6 **WAKIP:** CAN Bus Activity Wake-up Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 5 **ERRIP:** CAN bus Error Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 4 **TXB2IP:** CAN Transmit Buffer 2 Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 3 **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 2 **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 1 **RXB1IP:** CAN Receive Buffer 1 Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 0 **RXB0IP:** CAN Receive Buffer 0 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 |

TABLE 17-1: CAN CONTROLLER REGISTER MAP

| Address | Name | Address | Name | Address | Name | Address | Name |
|---------|----------|---------|----------|---------|----------|---------|----------|
| F7Fh | | F5Fh | | F3Fh | | F1Fh | RXM1EIDL |
| F7Eh | | F5Eh | CANSTAT | F3Eh | CANSTAT | F1Eh | RXM1EIDH |
| F7Dh | | F5Dh | RXB1D7 | F3Dh | TXB1D7 | F1Dh | RXM1SIDL |
| F7Ch | | F5Ch | RXB1D6 | F3Ch | TXB1D6 | F1Ch | RXM1SIDH |
| F7Bh | | F5Bh | RXB1D5 | F3Bh | TXB1D5 | F1Bh | RXM0EIDL |
| F7Ah | | F5Ah | RXB1D4 | F3Ah | TXB1D4 | F1Ah | RXM0EIDH |
| F79h | | F59h | RXB1D3 | F39h | TXB1D3 | F19h | RXM0SIDL |
| F78h | | F58h | RXB1D2 | F38h | TXB1D2 | F18h | RXM0SIDH |
| F77h | | F57h | RXB1D1 | F37h | TXB1D1 | F17h | RXF5EIDL |
| F76h | TXERRCNT | F56h | RXB1D0 | F36h | TXB1D0 | F16h | RXF5EIDH |
| F75h | RXERRCNT | F55h | RXB1DLC | F35h | TXB1DLC | F15h | RXF5SIDL |
| F74h | COMSTAT | F54h | RXB1EIDL | F34h | TXB1EIDL | F14h | RXF5SIDH |
| F73h | CIOCON | F53h | RXB1EIDH | F33h | TXB1EIDH | F13h | RXF4EIDL |
| F72h | BRGCON3 | F52h | RXB1SIDL | F32h | TXB1SIDL | F12h | RXF4EIDH |
| F71h | BRGCON2 | F51h | RXB1SIDH | F31h | TXB1SIDH | F11h | RXF4SIDL |
| F70h | BRGCON1 | F50h | RXB1CON | F30h | TXB1CON | F10h | RXF4SIDH |
| F6Fh | CANCON | F4Fh | | F2Fh | | F0Fh | RXF3EIDL |
| F6Eh | CANSTAT | F4Eh | CANSTAT | F2Eh | CANSTAT | F0Eh | RXF3EIDH |
| F6Dh | RXB0D7 | F4Dh | TXB0D7 | F2Dh | TXB2D7 | F0Dh | RXF3SIDL |
| F6Ch | RXB0D6 | F4Ch | TXB0D6 | F2Ch | TXB2D6 | F0Ch | RXF3SIDH |
| F6Bh | RXB0D5 | F4Bh | TXB0D5 | F2Bh | TXB2D5 | F0Bh | RXF2EIDL |
| F6Ah | RXB0D4 | F4Ah | TXB0D4 | F2Ah | TXB2D4 | F0Ah | RXF2EIDH |
| F69h | RXB0D3 | F49h | TXB0D3 | F29h | TXB2D3 | F09h | RXF2SIDL |
| F68h | RXB0D2 | F48h | TXB0D2 | F28h | TXB2D2 | F08h | RXF2SIDH |
| F67h | RXB0D1 | F47h | TXB0D1 | F27h | TXB2D1 | F07h | RXF1EIDL |
| F66h | RXB0D0 | F46h | TXB0D0 | F26h | TXB2D0 | F06h | RXF1EIDH |
| F65h | RXB0DLC | F45h | TXB0DLC | F25h | TXB2DLC | F05h | RXF1SIDL |
| F64h | RXB0EIDL | F44h | TXB0EIDL | F24h | TXB2EIDL | F04h | RXF1SIDH |
| F63h | RXB0EIDH | F43h | TXB0EIDH | F23h | TXB2EIDH | F03h | RXF0EIDL |
| F62h | RXB0SIDL | F42h | TXB0SIDL | F22h | TXB2SIDL | F02h | RXF0EIDL |
| F61h | RXB0SIDH | F41h | TXB0SIDH | F21h | TXB2SIDH | F01h | RXF0SIDL |
| F60h | RXB0CON | F40h | TXB0CON | F20h | TXB2CON | F00h | RXF0SIDH |

Note: Shaded registers are available in Access Bank Low area while the rest are available in Bank 15.

17.3 CAN Modes of Operation

The PIC18CXX8 has the following modes of operation. These modes are:

- Configuration mode
- Disable mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode (selected through CANRXM bits)

Modes are requested by setting the REQOP bits, except the Error Recognition mode, which is requested through the CANRXM bits. Entry into a mode is acknowledged by monitoring the OPMODE bits.

When changing modes, the mode will not actually change until all pending message transmissions are complete. Because of this, the user must verify that the device has actually changed into the requested mode before further operations are executed.

17.3.1 CONFIGURATION MODE

The CAN module has to be initialized before the activation. This is only possible if the module is in the Configuration mode. The Configuration mode is requested by setting REQOP2 bit. Only when the status bit OPMODE2 has a high level, the initialization can be performed. Afterwards, the configuration registers and the acceptance mask registers and the acceptance filter registers can be written. The module is activated by setting the control bits CFGREQ to zero.

The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is on-line. The CAN module will not be allowed to enter the Configuration mode while a transmission is taking place. The CONFIG bit serves as a lock to protect the following registers.

- Configuration registers
- Bus Timing registers
- Identifier Acceptance Filter registers
- Identifier Acceptance Mask registers

In the Configuration mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to configuration registers that are access restricted in other modes.

17.3.2 DISABLE MODE

In Disable mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity, however, any pending interrupts will remain and the error counters will retain their value.

If REQOP<2:0> is set to 001, the module will enter the module Disable mode. This mode is similar to disabling other peripheral modules by turning off the module enables. This causes the module internal clock to stop unless the module is active (i.e., receiving or transmitting a message). If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an idle bus, then accept the module disable command. OPMODE<2:0>=001 indicates whether the module successfully went into module Disable mode

The WAKIF interrupt is the only module interrupt that is still active in the module Disable mode. If the WAKIE is set, the processor will receive an interrupt whenever the CAN bus detects a dominant state, as occurs with a SOF.

The I/O pins will revert to normal I/O function when the module is in the module Disable mode.

17.3.3 NORMAL MODE

This is the standard operating mode of the PIC18CXX8. In this mode, the device actively monitors all bus messages and generates acknowledge bits, error frames, etc. This is also the only mode in which the PIC18CXX8 will transmit messages over the CAN bus.

17.3.4 LISTEN ONLY MODE

Listen Only mode provides a means for the PIC18CXX8 to receive all messages, including messages with errors. This mode can be used for bus monitor applications, or for detecting the baud rate in 'hot plugging' situations. For auto-baud detection, it is necessary that there are at least two other nodes which are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received. The Listen Only mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or acknowledge signals. The filters and masks can be used to allow only particular messages to be loaded into the receive registers, or the filter masks can be set to all zeros to allow a message with any identifier to pass. The error counters are reset and deactivated in this state. The Listen Only mode is activated by setting the mode request bits in the CANCON register.

17.3.5 LOOPBACK MODE

This mode will allow internal transmission of messages from the transmit buffers to the receive buffers, without actually transmitting messages on the CAN bus. This mode can be used in system development and testing. In this mode, the ACK bit is ignored and the device will allow incoming messages from itself just as if they were coming from another node. The Loopback mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or acknowledge signals. The TXCAN pin will revert to port I/O while the device is in this mode. The filters and masks can be used to allow only particular messages to be loaded into the receive registers. The masks can be set to all zeros to provide a mode that accepts all messages. The Loopback mode is activated by setting the mode request bits in the CANCON register.

17.3.6 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. The Error Recognition mode is activated by setting the RXM<1:0> bits in the RXBnCON registers to 11. In this mode, the data which is in the message assembly buffer until the error time, is copied in the receive buffer and can be read via the CPU interface. In addition, the data which was on the internal sampling of the CAN bus at the error time and the state vector of the protocol state machine and the bit counter CntCan, are stored in registers and can be read.

17.4 CAN Message Transmission

17.4.1 TRANSMIT BUFFERS

The PIC18CXX8 implements three Transmit Buffers. Each of these buffers occupies 14 bytes of SRAM and are mapped into the device memory maps.

For the MCU to have write access to the message buffer, the TXREQ bit must be clear, indicating that the message buffer is clear of any pending message to be transmitted. At a minimum, the TXBnSIDH, TXBnSIDL, and TXBnDLC registers must be loaded. If data bytes are present in the message, the TXBnDm registers must also be loaded. If the message is to use extended identifiers, the TXBnEIDm registers must also be loaded and the EXIDE bit set.

Prior to sending the message, the MCU must initialize the TXINE bit to enable or disable the generation of an interrupt when the message is sent. The MCU must also initialize the TXP priority bits (see Section 17.4.2).

17.4.2 TRANSMIT PRIORITY

Transmit priority is a prioritization, within the PIC18CXX8, of the pending transmittable messages. This is independent from, and not related to, any prioritization implicit in the message arbitration scheme built into the CAN protocol. Prior to sending the SOF, the priority of all buffers that are queued for transmission is compared. The transmit buffer with the highest priority will be sent first. If two buffers have the same priority setting, the buffer with the highest buffer number will be sent first. There are four levels of transmit priority. If TXP bits for a particular message buffer are set to 11, that buffer has the highest possible priority. If TXP bits for a particular message buffer are 00, that buffer has the lowest possible priority.

17.4.3 INITIATING TRANSMISSION

To initiate message transmission, the TXREQ bit must be set for each buffer to be transmitted.

When TXREQ is set, the TXABT, TXLARB and TXERR bits will be cleared.

Setting the TXREQ bit does not initiate a message transmission, it merely flags a message buffer as ready for transmission. Transmission will start when the device detects that the bus is available. The device will then begin transmission of the highest priority message that is ready.

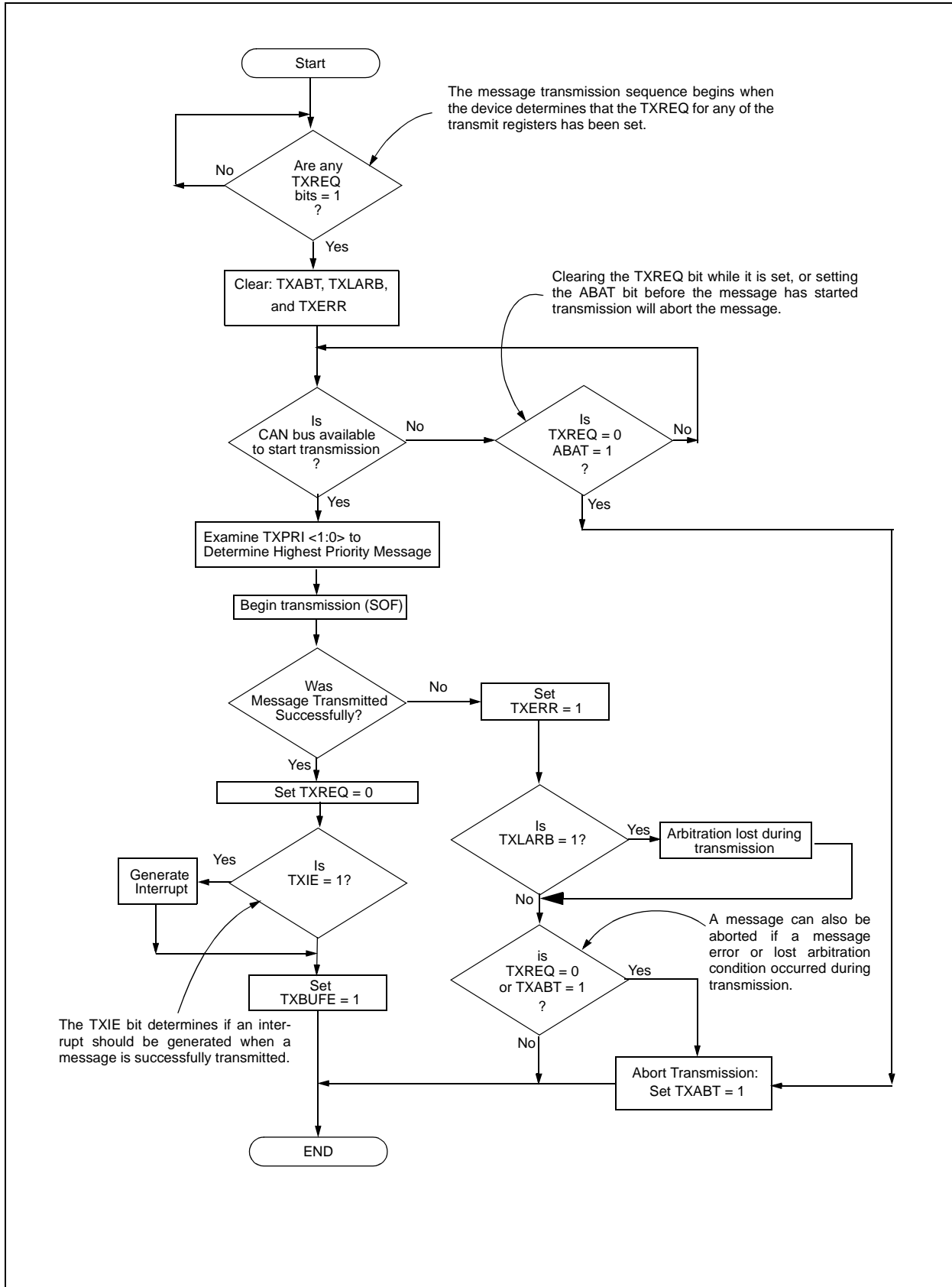
When the transmission has completed successfully, the TXREQ bit will be cleared, the TXBnIF bit will be set, and an interrupt will be generated if the TXBnIE bit is set.

If the message transmission fails, the TXREQ will remain set indicating that the message is still pending for transmission and one of the following condition flags will be set. If the message started to transmit but encountered an error condition, the TXERR and the IRXIF bits will be set and an interrupt will be generated. If the message lost arbitration, the TXLARB bit will be set.

17.4.4 ABORTING TRANSMISSION

The MCU can request to abort a message by clearing the TXBnCON.TXREQ bit associated with the corresponding message buffer. Setting CANCON.ABAT will request an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets TXBnCON.ABTF bits. If the message has started to transmit, it will attempt to transmit the current message fully. If the current message is transmitted fully and is not lost to arbitration or an error, the ABTF bit will not be set, because the message was transmitted successfully. Likewise, if a message is being transmitted during an abort request and the message is lost to arbitration or an error, the message will not be re-transmitted and the ABTF bit will be set, indicating that the message was successfully aborted.

FIGURE 17-2: TRANSMIT MESSAGE FLOWCHART



17.5 Message Reception

17.5.1 RECEIVE MESSAGE BUFFERING

The PIC18CXX8 includes two full receive buffers with multiple acceptance filters for each. There is also a separate Message Assembly Buffer (MAB), which acts as a third receive buffer (see Figure 17-3).

17.5.2 RECEIVE BUFFERS

Of the three receive buffers, the MAB is always committed to receiving the next message from the bus. The remaining two receive buffers are called RXB0 and RXB1 and can receive a complete message from the protocol engine. The MCU can access one buffer while the other buffer is available for message reception, or holding a previously received message.

The MAB assembles all messages received. These messages will be transferred to the RXBN buffers, only if the acceptance filter criteria are met.

Note: The entire contents of the MAB is moved into the receive buffer once a message is accepted. This means that regardless of the type of identifier (standard or extended) and the number of data bytes received, the entire receive buffer is overwritten with the MAB contents. Therefore, the contents of all registers in the buffer must be assumed to have been modified when any message is received.

When a message is moved into either of the receive buffers, the appropriate RXBnIF bit is set. This bit must be cleared by the MCU when it has completed processing the message in the buffer, in order to allow a new message to be received into the buffer. This bit provides a positive lockout to ensure that the MCU has finished with the message before the PIC18CXX8 attempts to load a new message into the receive buffer. If the RXBnIE bit is set, an interrupt will be generated to indicate that a valid message has been received.

17.5.3 RECEIVE PRIORITY

RXB0 is the higher priority buffer and has two message acceptance filters associated with it. RXB1 is the lower priority buffer and has four acceptance filters associated with it. The lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer. Additionally, the RXB0CON register can be configured such that if RXB0 contains a valid message, and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1, regardless of the acceptance criteria of RXB1. There are also two programmable acceptance filter masks available, one for each receive buffer (see Section 4.5).

When a message is received, bits <3:0> of the RXBNCON register will indicate the acceptance filter number that enabled reception, and whether the received message is a remote transfer request.

The RXM bits set special receive modes. Normally, these bits are set to 00 to enable reception of all valid messages, as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the acceptance filter register. If the RXM bits are set to 01 or 10, the receiver will accept only messages with standard or extended identifiers, respectively. If an acceptance filter has the EXIDE bit set such that it does not correspond with the RXM mode, that acceptance filter is rendered useless. These two modes of RXM bits can be used in systems where it is known that only standard or extended messages will be on the bus. If the RXM bits are set to 11, the buffer will receive all messages, regardless of the values of the acceptance filters. Also, if a message has an error before the end of frame, that portion of the message assembled in the MAB before the error frame, will be loaded into the buffer. This mode has some value in debugging a CAN system and would not be used in an actual system environment.

PIC18CXX8

FIGURE 17-3: RECEIVE BUFFER BLOCK DIAGRAM

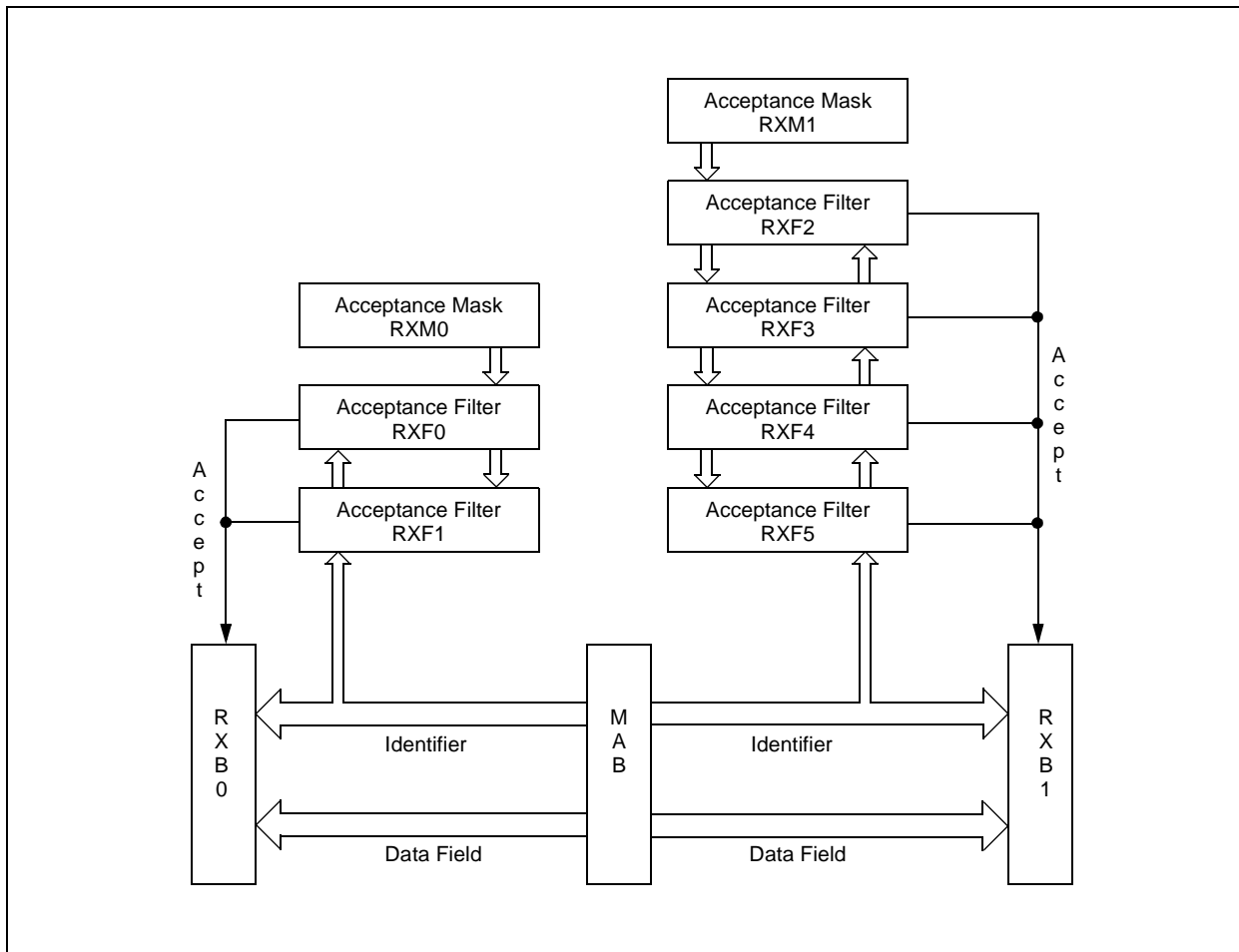
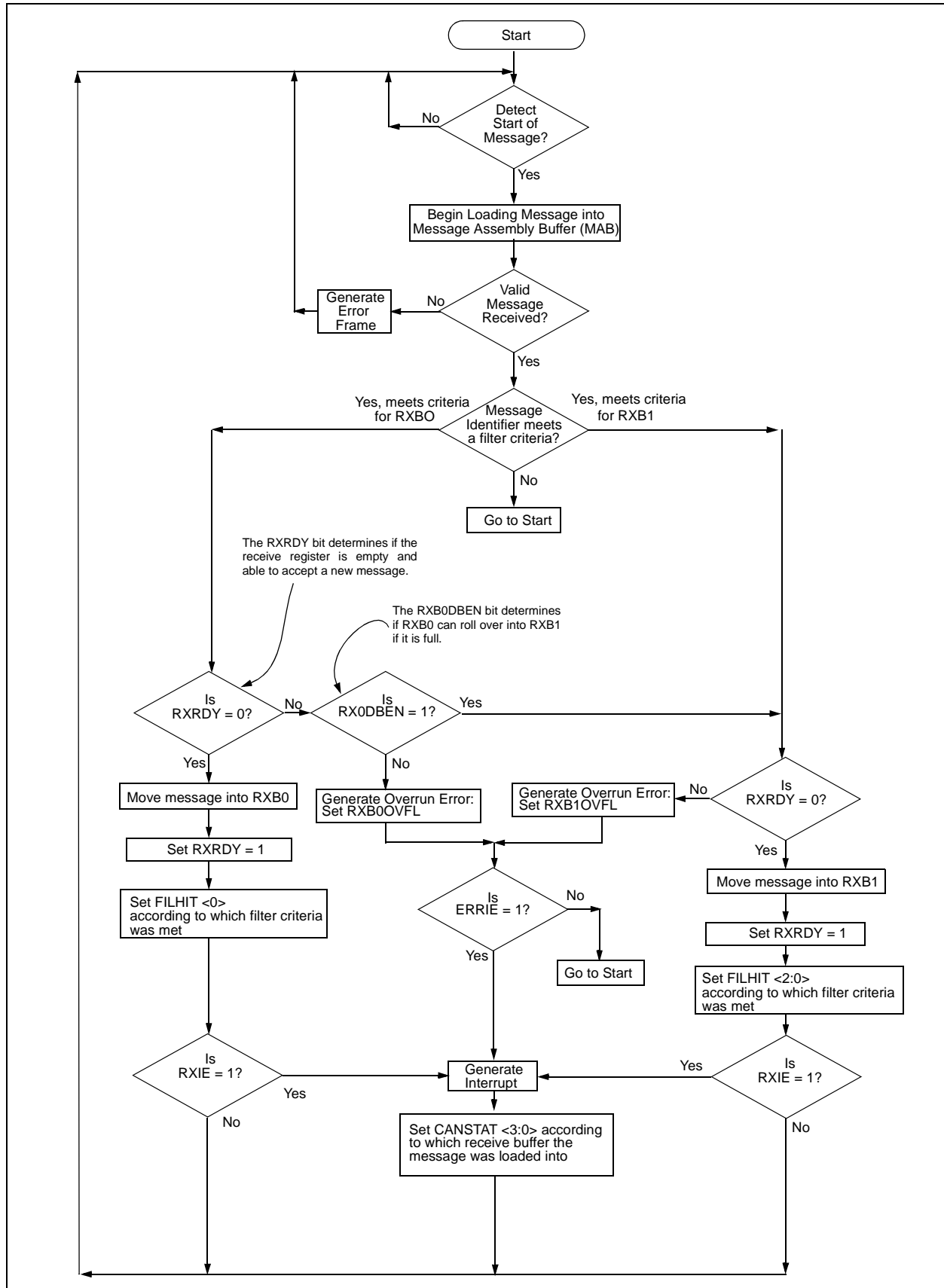


FIGURE 17-4: MESSAGE RECEPTION FLOWCHART



17.6 Message Acceptance Filters and Masks

The Message Acceptance Filters and Masks are used to determine if a message in the message assembly buffer should be loaded into either of the receive buffers. Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifier are examined with the filters. A truth table is shown below in Table 17-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if the message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, then that bit will automatically be accepted, regardless of the filter bit.

TABLE 17-2: FILTER/MASK TRUTH TABLE

| Mask bit n | Filter bit n | Message Identifier bit n001 | Accept or Reject bit n |
|------------|--------------|-----------------------------|------------------------|
| 0 | X | X | Accept |
| 1 | 0 | 0 | Accept |
| 1 | 0 | 1 | Reject |
| 1 | 1 | 0 | Reject |
| 1 | 1 | 1 | Accept |

Legend: X = don't care

As shown in the Receive Buffers Block Diagram (Figure 17-3), acceptance filters RXF0 and RXF1, and filter mask RXM0 are associated with RXB0. Filters RXF2, RXF3, RXF4, and RXF5 and mask RXM1 are associated with RXB1. When a filter matches and a message is loaded into the receive buffer, the filter number that enabled the message reception is loaded into the FILHIT bit(s). For RXB1, the RXB1CON register contains the FILHIT<2:0> bits. They are coded as follows:

- 101 = Acceptance Filter 5 (RXF5)
- 100 = Acceptance Filter 4 (RXF4)
- 011 = Acceptance Filter 3 (RXF3)
- 010 = Acceptance Filter 2 (RXF2)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

Note: 000 and 001 can only occur if the RXB0DBEN bit is set in the RXB0CON register, allowing RXB0 messages to roll over into RXB1.

The coding of the RXB0DBEN bit enables these three bits to be used similarly to the FILHIT bits and to distinguish a hit on filter RXF0 and RXF1, in either RXB0, or after a roll over into RXB1.

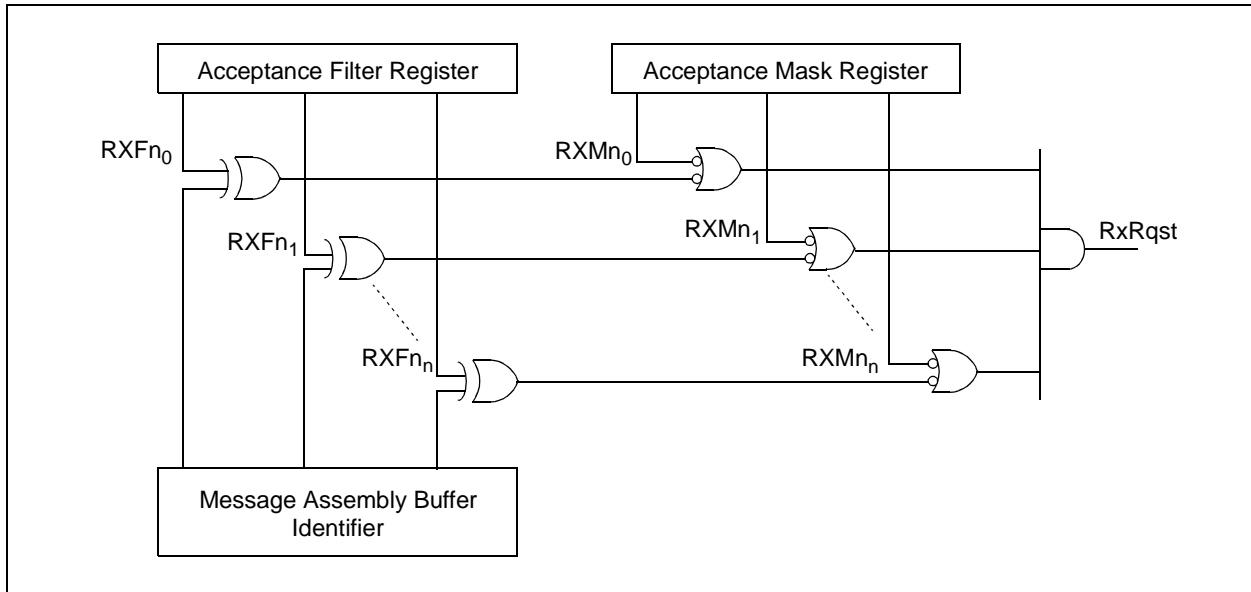
- 111 = Acceptance Filter 1 (RXF1)
- 110 = Acceptance Filter 0 (RXF0)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0

If the RXB0DBEN bit is clear, there are six codes corresponding to the six filters. If the RXB0DBEN bit is set, there are six codes corresponding to the six filters, plus two additional codes corresponding to RXF0 and RXF1 filters that roll over into RXB1.

If more than one acceptance filter matches, the FILHIT bits will encode the binary value of the lowest numbered filter that matched. In other words, if filter RXF2 and filter RXF4 match, FILHIT will be loaded with the value for RXF2. This essentially prioritizes the acceptance filters with a lower number filter having higher priority. Messages are compared to filters in ascending order of filter number.

The mask and filter registers can only be modified when the PIC18CXX8 is in Configuration mode. The mask and filter registers cannot be read outside of Configuration mode. When outside of Configuration mode, all mask and filter registers will be read as '0'.

FIGURE 17-5: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION



17.7 Baud Rate Setting

All nodes on a given CAN bus must have the same nominal bit rate. The CAN protocol uses Non-Return-to-Zero (NRZ) coding, which does not encode a clock within the data stream. Therefore, the receive clock must be recovered by the receiving nodes and synchronized to the transmitters clock.

As oscillators and transmission time may vary from node to node, the receiver must have some type of Phase Lock Loop (PLL) synchronized to data transmission edges, to synchronize and maintain the receiver clock. Since the data is NRZ coded, it is necessary to include bit stuffing to ensure that an edge occurs at least every six bit times, to maintain the Digital Phase Lock Loop (DPLL) synchronization.

The bit timing of the PIC18CXX8 is implemented using a DPLL that is configured to synchronize to the incoming data, and provide the nominal timing for the transmitted data. The DPLL breaks each bit time into multiple segments, made up of minimal periods of time called the time quanta (T_Q).

Bus timing functions executed within the bit time frame, such as synchronization to the local oscillator, network transmission delay compensation, and sample point positioning, are defined by the programmable bit timing logic of the DPLL.

All devices on the CAN bus must use the same bit rate. However, all devices are not required to have the same master oscillator clock frequency. For the different clock frequencies of the individual devices, the bit rate has to be adjusted by appropriately setting the baud rate prescaler and number of time quanta in each segment.

The nominal bit rate is the number of bits transmitted per second assuming an ideal transmitter with an ideal oscillator, in the absence of resynchronization. The nominal bit rate is defined to be a maximum of 1Mb/s.

Nominal Bit Time is defined as:

$$T_{BIT} = 1 / \text{NOMINAL BIT RATE}$$

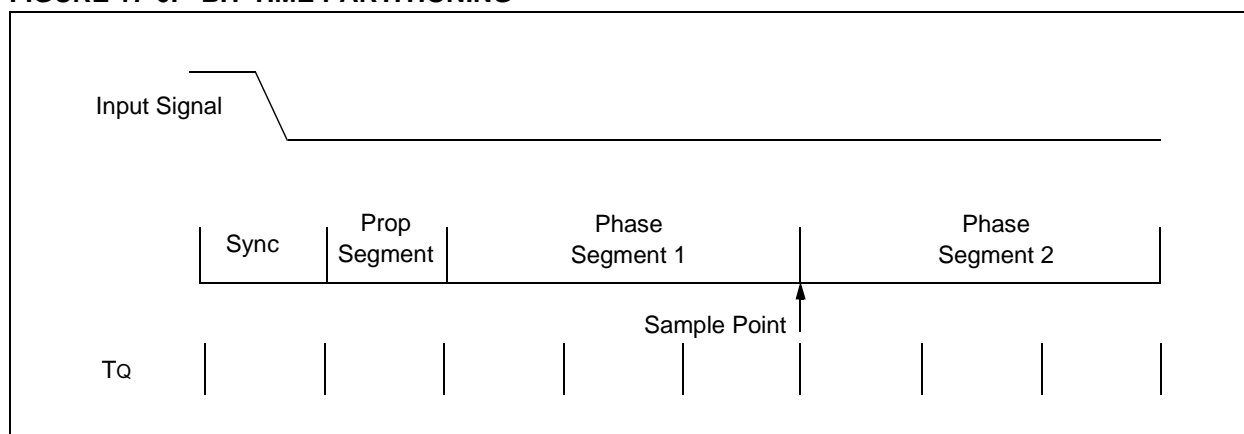
The nominal bit time can be thought of as being divided into separate non-overlapping time segments. These segments are shown in Figure 17-6.

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

$$\text{Nominal Bit Time} = T_Q * (\text{Sync_Seg} + \text{Prop_Seg} + \text{Phase_Seg1} + \text{Phase_Seg2})$$

The time segments and also, the nominal bit time, are made up of integer units of time called time quanta or T_Q (see Figure 17-6). By definition, the nominal bit time is programmable from a minimum of 8 T_Q to a maximum of 25 T_Q. Also by definition, the minimum nominal bit time is 1 μs, corresponding to a maximum 1 Mb/s rate.

FIGURE 17-6: BIT TIME PARTITIONING



17.7.1 TIME QUANTA

The Time Quanta is a fixed unit of time derived from the oscillator period. There is a programmable baud rate prescaler, with integral values ranging from 1 to 64, in addition to a fixed divide by two for clock generation.

EXAMPLE 17-2: CALCULATION FOR

Fosc = 16MHz

If Fosc = 16 MHz, BRP<5:0> = 00h, and Nominal Bit Time = 8 TQ; then TQ = 125 nsec and Nominal Bit Rate = 1 Mb/s

EXAMPLE 17-3: CALCULATION FOR

Fosc = 20MHz

If Fosc = 20 MHz, BRP<5:0> = 01h, and Nominal Bit Time = 8 TQ; then TQ = 200nsec and Nominal Bit Rate = 625 Kb/s

EXAMPLE 17-4: CALCULATION FOR

Fosc = 25MHz

If Fosc = 25 MHz, BRP<5:0> = 3Fh, and Nominal Bit Time = 25 TQ; then TQ = 5.12 usec and Nominal Bit Rate = 7.8 Kb/s

The frequencies of the oscillators in the different nodes must be coordinated in order to provide a system-wide specified nominal bit time. This means that all oscillators must have a TOSC that is a integral divisor of TQ. It should also be noted that although the number of TQ is programmable from 4 to 25, the usable minimum is 8 TQ. A bit time of less than 8 TQ in length is not guaranteed to operate correctly.

17.7.2 SYNCHRONIZATION SEGMENT

This part of the bit time is used to synchronize the various CAN nodes on the bus. The edge of the input signal is expected to occur during the sync segment. The duration is 1 TQ.

17.7.3 PROPAGATION SEGMENT

This part of the bit time is used to compensate for physical delay times within the network. These delay times consist of the signal propagation time on the bus line and the internal delay time of the nodes. The length of the Propagation Segment can be programmed from 1 TQ to 8 TQ by setting the PRSEG2:PRSEG0 bits.

17.7.4 PHASE BUFFER SEGMENTS

The Phase Buffer Segments are used to optimally locate the sampling point of the received bit, within the nominal bit time. The sampling point occurs between phase segment 1 and phase segment 2. These segments can be lengthened or shortened by the resynchronization process. The end of phase segment 1 determines the sampling point within a bit time. Phase segment 1 is programmable from 1 TQ to 8 TQ in duration. Phase segment 2 provides delay before the next transmitted data transition and is also programmable from 1 TQ to 8 TQ in duration (however, due to IPT requirements the actual minimum length of phase segment 2 is 2 TQ, or it may be defined to be equal to the greater of phase segment 1 or the Information Processing Time (IPT)).

17.7.5 SAMPLE POINT

The Sample Point is the point of time at which the bus level is read and value of the received bit is determined. The sampling point occurs at the end of phase segment 1. If the bit timing is slow and contains many TQ, it is possible to specify multiple sampling of the bus line at the sample point. The value of the received bit is determined to be the value of the majority decision of three values. The three samples are taken at the sample point, and twice before with a time of TQ/2 between each sample.

17.7.6 INFORMATION PROCESSING TIME

The Information Processing Time (IPT) is the time segment, starting at the sample point, that is reserved for calculation of the subsequent bit level. The CAN specification defines this time to be less than or equal to 2 TQ. The PIC18CXX8 defines this time to be 2 TQ. Thus, phase segment 2 must be at least 2 TQ long.

17.8 Synchronization

To compensate for phase shifts between the oscillator frequencies of each of the nodes on the bus, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (Sync Seg). The circuit will then adjust the values of phase segment 1 and phase segment 2, as necessary. There are two mechanisms used for synchronization.

17.8.1 HARD SYNCHRONIZATION

Hard Synchronization is only done when there is a recessive to dominant edge during a BUS IDLE condition, indicating the start of a message. After hard synchronization, the bit time counters are restarted with Sync Seg. Hard synchronization forces the edge, which has occurred to lie within the synchronization segment of the restarted bit time. Due to the rules of synchronization, if a hard synchronization occurs, there will not be a resynchronization within that bit time.

17.8.2 RESYNCHRONIZATION

As a result of Resynchronization, phase segment 1 may be lengthened, or phase segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to phase segment 1 (see Figure 17-7), or subtracted from phase segment 2 (see Figure 17-8). The SJW is programmable between 1 T_Q and 4 T_Q.

Clocking information will only be derived from recessive to dominant transitions. The property that only a fixed maximum number of successive bits have the same value, ensures resynchronization to the bit stream during a frame.

The phase error of an edge is given by the position of the edge relative to Sync Seg, measured in T_Q. The phase error is defined in magnitude of T_Q as follows:

- $e = 0$ if the edge lies within SYNCSEGE.
- $e > 0$ if the edge lies before the SAMPLE POINT.
- $e < 0$ if the edge lies after the SAMPLE POINT of the previous bit.

If the magnitude of the phase error is less than, or equal to, the programmed value of the synchronization jump width, the effect of a resynchronization is the same as that of a hard synchronization.

If the magnitude of the phase error is larger than the synchronization jump width, and if the phase error is positive, then phase segment 1 is lengthened by an amount equal to the synchronization jump width.

If the magnitude of the phase error is larger than the resynchronization jump width, and if the phase error is negative, then phase segment 2 is shortened by an amount equal to the synchronization jump width.

17.8.3 SYNCHRONIZATION RULES

- Only one synchronization within one bit time is allowed.
- An edge will be used for synchronization only if the value detected at the previous sample point (previously read bus value) differs from the bus value immediately after the edge.
- All other recessive to dominant edges, fulfilling rules 1 and 2, will be used for resynchronization with the exception that a node transmitting a dominant bit will not perform a resynchronization, as a result of a recessive to dominant edge with a positive phase error.

FIGURE 17-7: LENGTHENING A BIT PERIOD

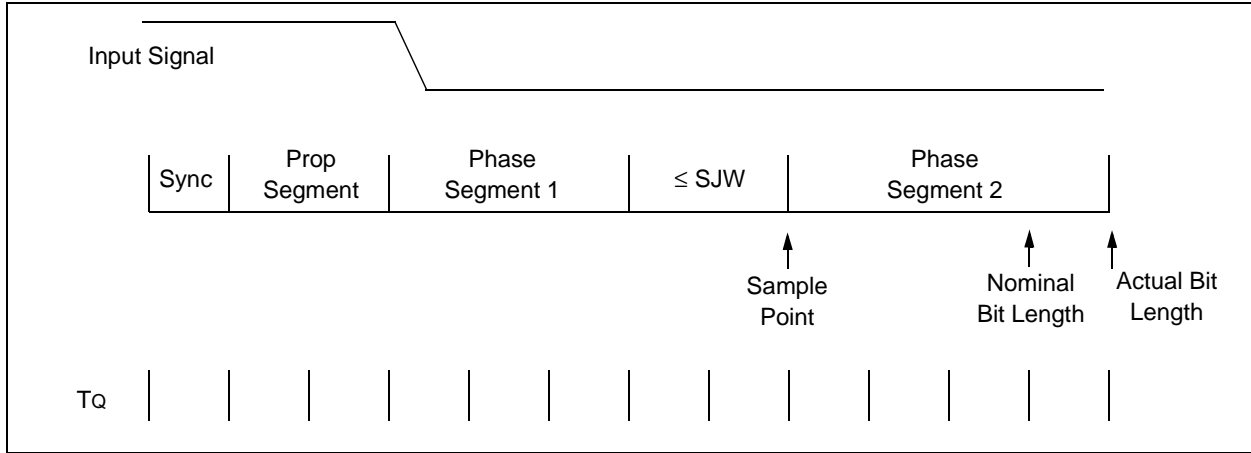
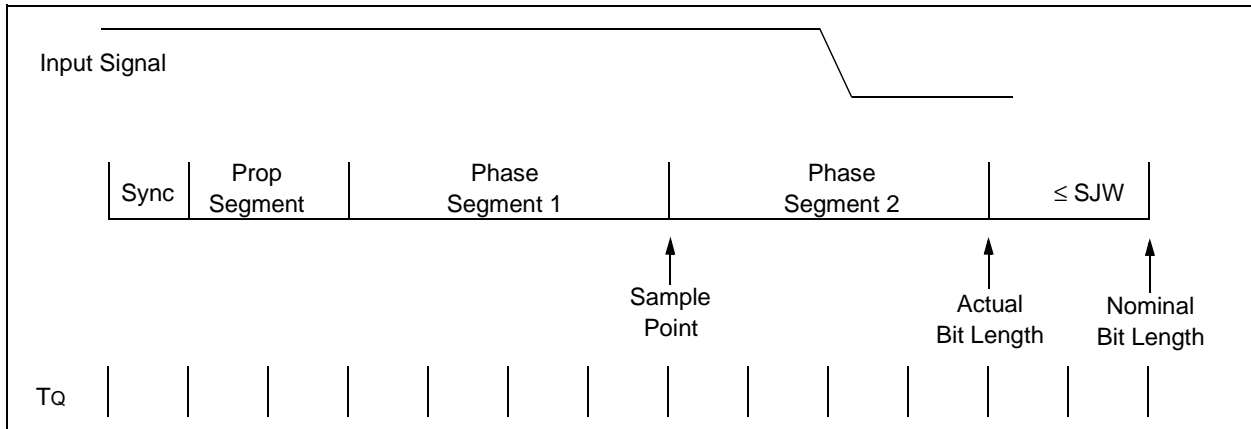


FIGURE 17-8: SHORTENING A BIT PERIOD



17.9 Programming Time Segments

Some requirements for programming of the time segments:

- Prop Seg + Phase Seg 1 \geq Phase Seg 2
- Phase Seg 2 \geq Sync Jump Width

For example, assuming that a 125 kHz CAN baud rate with $F_{OSC} = 20$ MHz is desired:

$T_{OSC} = 50$ nsec, choose $BRP_{<5:0>} = 04h$, then $T_Q = 500$ nsec. To obtain 125 kHz, the bit time must be 16 T_Q .

Sync Seg = 1 T_Q ; Prop Seg = 2 T_Q ; So, setting Phase Seg 1 = 7 T_Q would place the sample at 10 T_Q after the transition. This would leave 6 T_Q for Phase Seg 2.

Since Phase Seg 2 is 6, by the rules, SJW could be the maximum of 4 T_Q . However, normally a large SJW is only necessary when the clock generation of the different nodes is inaccurate or unstable, such as using ceramic resonators. So an SJW of 1 is typically enough.

17.10 Oscillator Tolerance

The bit timing requirements allow ceramic resonators to be used in applications with transmission rates of up to 125 kbit/sec, as a rule of thumb. For the full bus speed range of the CAN protocol, a quartz oscillator is required. A maximum node-to-node oscillator variation of 1.7% is allowed.

17.11 Bit Timing Configuration Registers

The configuration registers (BRGCON1, BRGCON2, BRGCON3) control the bit timing for the CAN bus interface. These registers can only be modified when the PIC18CXX8 is in Configuration mode.

17.11.1 BRGCON1

The BRP bits control the baud rate prescaler. The SJW<1:0> bits select the synchronization jump width in terms of number of T_Q 's.

17.11.2 BRGCON2

The PRSEG bits set the length, in T_Q 's, of the propagation segment. The SEG1PH bits set the length, in T_Q 's, of phase segment 1. The SAM bit controls how many times the RXCAN pin is sampled. Setting this bit to a '1' causes the bus to be sampled three times; twice at $T_Q/2$ before the sample point, and once at the normal sample point (which is at the end of phase segment 1). The value of the bus is determined to be the value read during at least two of the samples. If the SAM bit is set to a '0', then the RXCAN pin is sampled only once at the sample point. The SEG2PHTS bit controls how the length of phase segment 2 is determined. If this bit is set to a '1', then the length of phase segment 2 is determined by the SEG2PH bits of BRGCON3. If the SEG2PHTS bit is set to a '0', then the length of phase segment 2 is the greater of phase segment 1 and the information processing time (which is fixed at 2 T_Q for the PIC18CXX8).

17.11.3 BRGCON3

The PHSEG2<2:0> bits set the length, in T_Q 's, of phase segment 2, if the SEG2PHTS bit is set to a '1'. If the SEG2PHTS bit is set to a '0', then the PHSEG2<2:0> bits have no effect.

17.12 Error Detection

The CAN protocol provides sophisticated error detection mechanisms. The following errors can be detected.

17.12.1 CRC ERROR

With the Cyclic Redundancy Check (CRC), the transmitter calculates special check bits for the bit sequence, from the start of a frame until the end of the data field. This CRC sequence is transmitted in the CRC Field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an error frame is generated. The message is repeated.

17.12.2 ACKNOWLEDGE ERROR

In the acknowledge field of a message, the transmitter checks if the acknowledge slot (which has sent out as a recessive bit) contains a dominant bit. If not, no other node has received the frame correctly. An acknowledge error has occurred; an error frame is generated and the message will have to be repeated.

17.12.3 FORM ERROR

If a node detects a dominant bit in one of the four segments, including end of frame, interframe space, acknowledge delimiter, or CRC delimiter, then a form error has occurred and an error frame is generated. The message is repeated.

17.12.4 BIT ERROR

A Bit Error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit, when monitoring the actual bus level and comparing it to the just transmitted bit. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the arbitration field and the acknowledge slot, no bit error is generated because normal arbitration is occurring.

17.12.5 STUFF BIT ERROR

If, between the start of frame and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit stuffing rule has been violated. A Stuff Bit Error occurs and an error frame is generated. The message is repeated.

17.12.6 ERROR STATES

Detected errors are made public to all other nodes via error frames. The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states "error-active", "error-passive" or "bus-off" according to the value of the internal error counters. The error-active state is the usual state, where the bus node can transmit messages and active error frames (made of dominant bits), without any restrictions. In the error-passive state, messages and passive error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the station to participate in the bus communication. During this state, messages can neither be received nor transmitted.

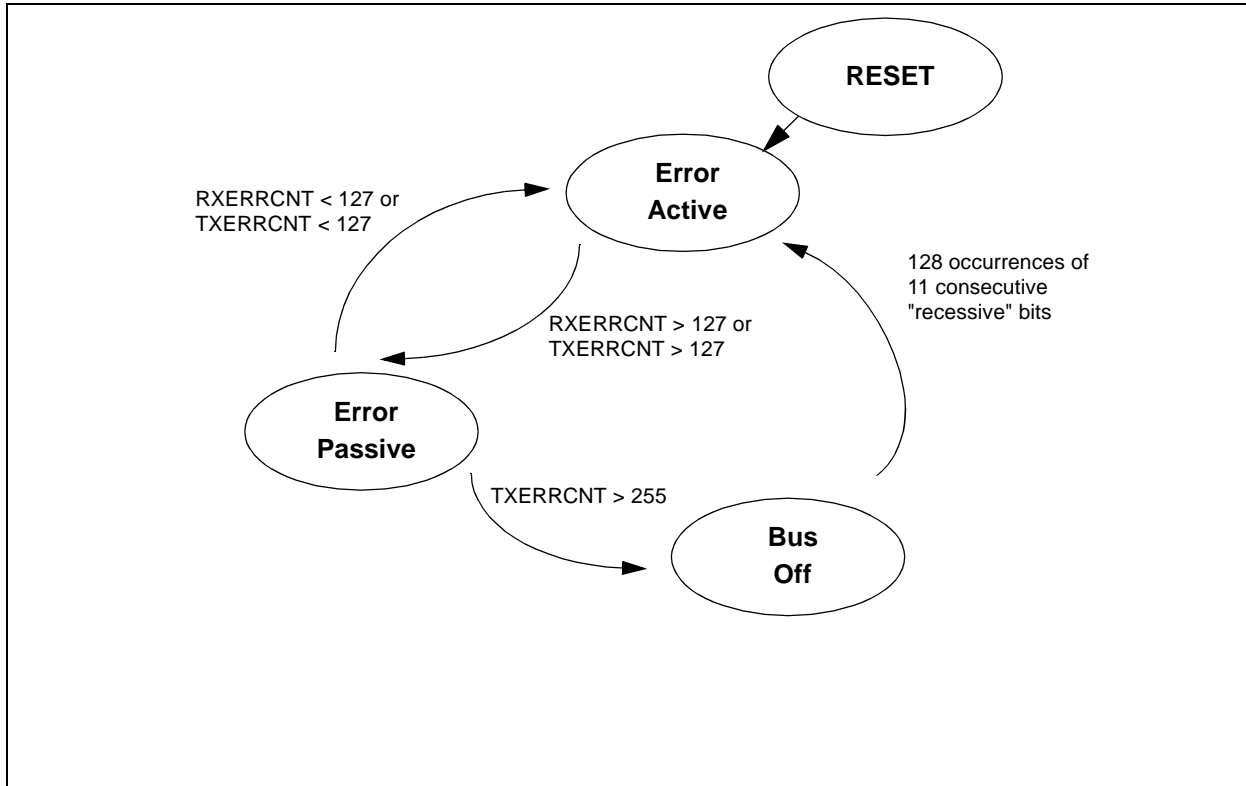
17.12.7 ERROR MODES AND ERROR COUNTERS

The PIC18CXX8 contains two error counters: the Receive Error Counter (RXERRCNT), and the Transmit Error Counter (TXERRCNT). The values of both counters can be read by the MCU. These counters are incremented or decremented in accordance with the CAN bus specification.

The PIC18CXX8 is error-active if both error counters are below the error-passive limit of 128. It is error-passive if at least one of the error counters equals or exceeds 128. It goes to bus-off if the transmit error counter equals or exceeds the bus-off limit of 256. The device remains in this state, until the bus-off recovery sequence is received. The bus-off recovery sequence consists of 128 occurrences of 11 consecutive recessive bits (see Figure 17-9). Note that the CAN module, after going bus-off, will recover back to error-active, without any intervention by the MCU, if the bus remains idle for 128 X 11 bit times. If this is not desired, the error interrupt service routine should address this. The current error mode of the CAN module can be read by the MCU via the COMSTAT register.

Additionally, there is an error state warning flag bit, EWARN, which is set if at least one of the error counters equals or exceeds the error warning limit of 96. EWARN is reset if both error counters are less than the error warning limit.

FIGURE 17-9: ERROR MODES STATE DIAGRAM



17.13 CAN Interrupts

The module has several sources of interrupts. Each of these interrupts can be individually enabled or disabled. The CANINTF register contains interrupt flags. The CANINTE register contains the enables for the 8 main interrupts. A special set of read only bits in the CANSTAT register (ICODE bits) can be used in combination with a jump table for efficient handling of interrupts.

All interrupts have one source, with the exception of the Error Interrupt. Any of the Error Interrupt sources can set the Error Interrupt Flag. The source of the Error Interrupt can be determined by reading the Communication Status register COMSTAT.

The interrupts can be broken up into two categories: receive and transmit interrupts.

The receive related interrupts are:

- Receive Interrupts
- Wake-up Interrupt
- Receiver Overrun Interrupt
- Receiver Warning Interrupt
- Receiver Error Passive Interrupt

The Transmit related interrupts are

- Transmit Interrupts
- Transmitter Warning Interrupt
- Transmitter Error Passive Interrupt
- Bus Off Interrupt

17.13.1 INTERRUPT CODE BITS

The source of a pending interrupt is indicated in the ICODE (interrupt code) bits. Interrupts are internally prioritized, such that the lower the ICODE value, the higher the interrupt priority. Once the highest priority interrupt condition has been cleared, the code for the next highest priority interrupt that is pending (if any), will be reflected by the ICODE bits (see Table 17-3). Note that only those interrupt sources that have their associated CANINTE enable bit set will be reflected in the ICODE bits.

TABLE 17-3: ICODE<2:0> DECODE

| ICODE<2:0> | Boolean Expression |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 000 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \overline{\text{RX1}}$ |
| 001 | ERR |
| 010 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}}$ |
| 011 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}}$ |
| 100 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}}$ |
| 101 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}}$ |
| 110 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}}$ |
| 111 | $\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \overline{\text{RX1}}$ |

17.13.2 TRANSMIT INTERRUPT

When the Transmit Interrupt is enabled, an interrupt will be generated when the associated transmit buffer becomes empty and is ready to be loaded with a new message. The TXBnIF bit will be set to indicate the source of the interrupt. The interrupt is cleared by the MCU resetting the TXBnIF bit to a '0'.

17.13.3 RECEIVE INTERRUPT

When the Receive Interrupt is enabled, an interrupt will be generated when a message has been successfully received and loaded into the associated receive buffer. This interrupt is activated immediately after receiving the EOF field. The RXBnIF bit will be set to indicate the source of the interrupt. The interrupt is cleared by the MCU resetting the RXBnIF bit to a '0'.

17.13.4 MESSAGE ERROR INTERRUPT

When an error occurs during transmission or reception of a message, the message error flag IRXIF will be set and, if the IRXIE bit is set, an interrupt will be generated. This is intended to be used to facilitate baud rate determination when used in conjunction with Listen Only mode.

17.13.5 BUS ACTIVITY WAKE-UP INTERRUPT

When the PIC18CXX8 is in SLEEP mode and the bus activity wake-up interrupt is enabled, an interrupt will be generated, and the WAKIF bit will be set, when activity is detected on the CAN bus. This interrupt causes the PIC18CXX8 to exit SLEEP mode. The interrupt is reset by the MCU clearing the WAKIF bit.

17.13.6 ERROR INTERRUPT

When the error interrupt is enabled, an interrupt is generated if an overflow condition occurs, or if the error state of transmitter or receiver has changed. The Error Flags in COMSTAT will indicate one of the following conditions.

17.13.6.1 Receiver Overflow

An overflow condition occurs when the MAB has assembled a valid received message (the message meets the criteria of the acceptance filters) and the receive buffer associated with the filter is not available for loading of a new message. The associated COMSTAT.RXNOVFL bit will be set to indicate the overflow condition. This bit must be cleared by the MCU.

17.13.6.2 Receiver Warning

The receive error counter has reached the MCU warning limit of 96.

17.13.6.3 Transmitter Warning

The transmit error counter has reached the MCU warning limit of 96.

17.13.6.4 Receiver Bus-Passive

The receive error counter has exceeded the error-passive limit of 127 and the device has gone to error-passive state.

17.13.6.5 Transmitter Bus-Passive

The transmit error counter has exceeded the error-passive limit of 127 and the device has gone to error-passive state.

17.13.6.6 Bus-Off

The transmit error counter has exceeded 255 and the device has gone to bus-off state.

17.13.7 INTERRUPT ACKNOWLEDGE

Interrupts are directly associated with one or more status flags in the PIF register. Interrupts are pending as long as one of the flags is set. Once an interrupt flag is set by the device, the flag can not be reset by the MCU until the interrupt condition is removed.

18.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The analog-to-digital (A/D) converter module has twelve inputs for the PIC18C658 devices and sixteen for the PIC18C858 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 18-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 18-2, configures the functions of the port pins. The ADCON2, shown in Register 16-3, configures the A/D clock source and justification.

REGISTER 18-1: 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/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)
 1001 = channel 09, (AN9)
 1010 = channel 10, (AN10)
 1011 = channel 11, (AN11)
 1100 = channel 12, (AN12)⁽¹⁾
 1101 = channel 13, (AN13)⁽¹⁾
 1110 = channel 14, (AN14)⁽¹⁾
 1111 = channel 15, (AN15)⁽¹⁾

Note 1: These channels are not available on the PIC18C658 devices.

bit 1 **GO/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 |

PIC18CXX8

REGISTER 18-2: 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

| | AN15 | AN14 | AN13 | AN12 | 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 | A | A | A | A |
| 0001 | D | D | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| 0010 | D | D | D | A | A | A | A | A | A | A | A | A | A | A | A | A |
| 0011 | D | D | D | D | A | A | A | A | A | A | A | A | A | A | A | A |
| 0100 | D | D | D | D | D | A | A | A | A | A | A | A | A | A | A | A |
| 0101 | D | D | D | D | D | D | A | A | A | A | A | A | A | A | A | A |
| 0110 | D | D | D | D | D | D | D | A | A | A | A | A | A | A | A | A |
| 0111 | D | D | D | D | D | D | D | D | A | A | A | A | A | A | A | A |
| 1000 | D | D | D | D | D | D | D | D | D | A | A | A | A | A | A | A |
| 1001 | D | D | D | D | D | D | D | D | D | D | A | A | A | A | A | A |
| 1010 | D | D | D | D | D | D | D | D | D | D | D | A | A | A | A | A |
| 1011 | D | D | D | D | D | D | D | D | D | D | D | D | A | A | A | A |
| 1100 | D | D | D | D | D | D | D | D | D | D | D | D | D | A | A | A |
| 1101 | D | D | D | D | D | D | D | D | D | D | D | D | D | D | A | A |
| 1110 | D | D | D | D | 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 | D | D | D | D |

A = Analog input D = Digital I/O

Shaded cells = additional A/D channels available on the PIC18C858 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

Note: Channels AN15 through AN12 are not available on the 68-pin devices.

REGISTER 18-3: 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 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
 000 = Fosc/2
 001 = Fosc/8
 010 = Fosc/32
 011 = FRC (clock derived from an RC oscillator = 1 MHz max)
 100 = FOSC/4
 101 = FOSC/16
 110 = Fosc/64
 111 = FRC (clock derived from an 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 |

PIC18CXX8

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (VDD and VSS), or the voltage level on the RA3/AN3/VREF+ pin and RA2/AN2/VREF-.

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.

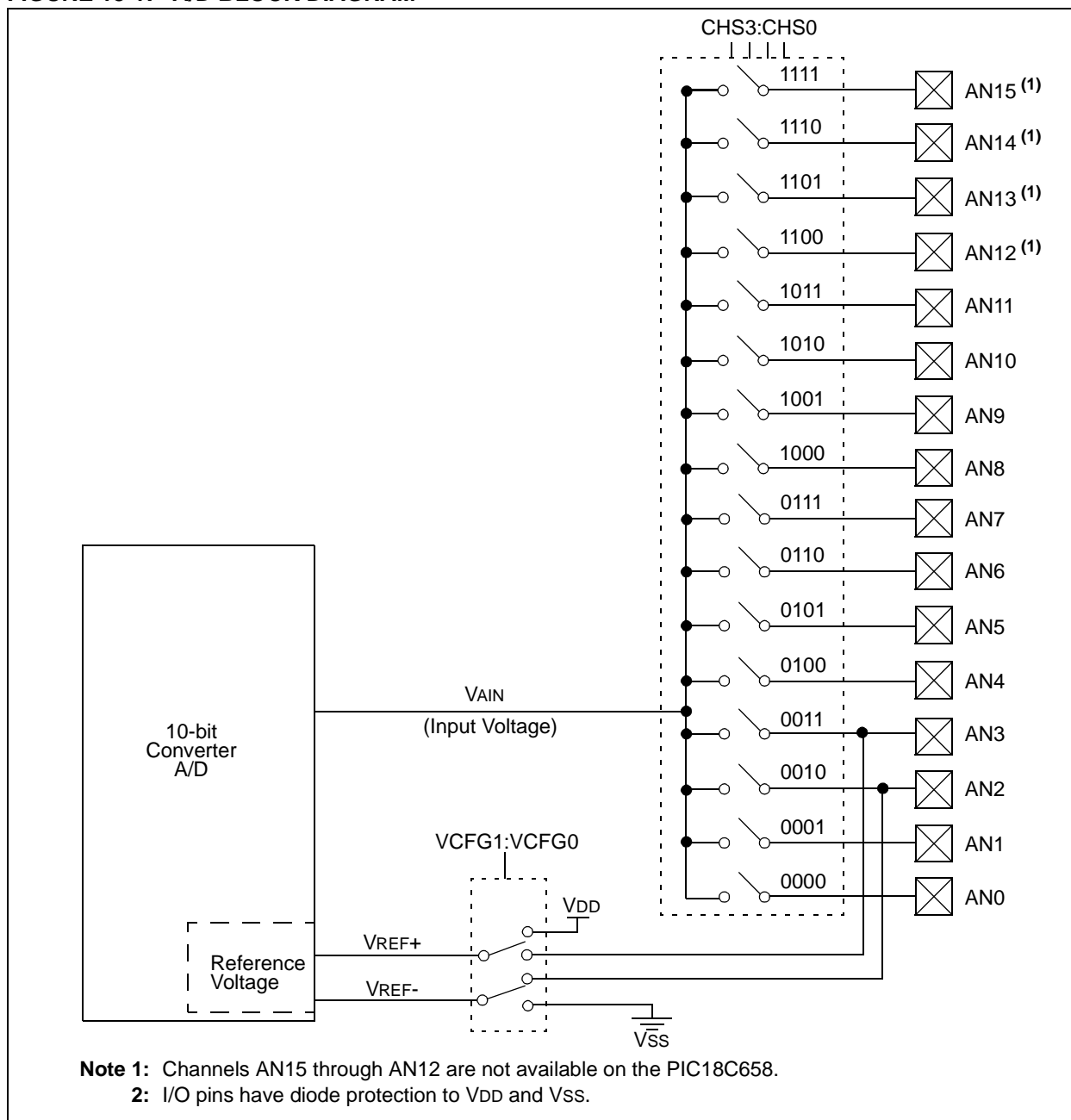
The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

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/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 18-1.

FIGURE 18-1: A/D BLOCK DIAGRAM

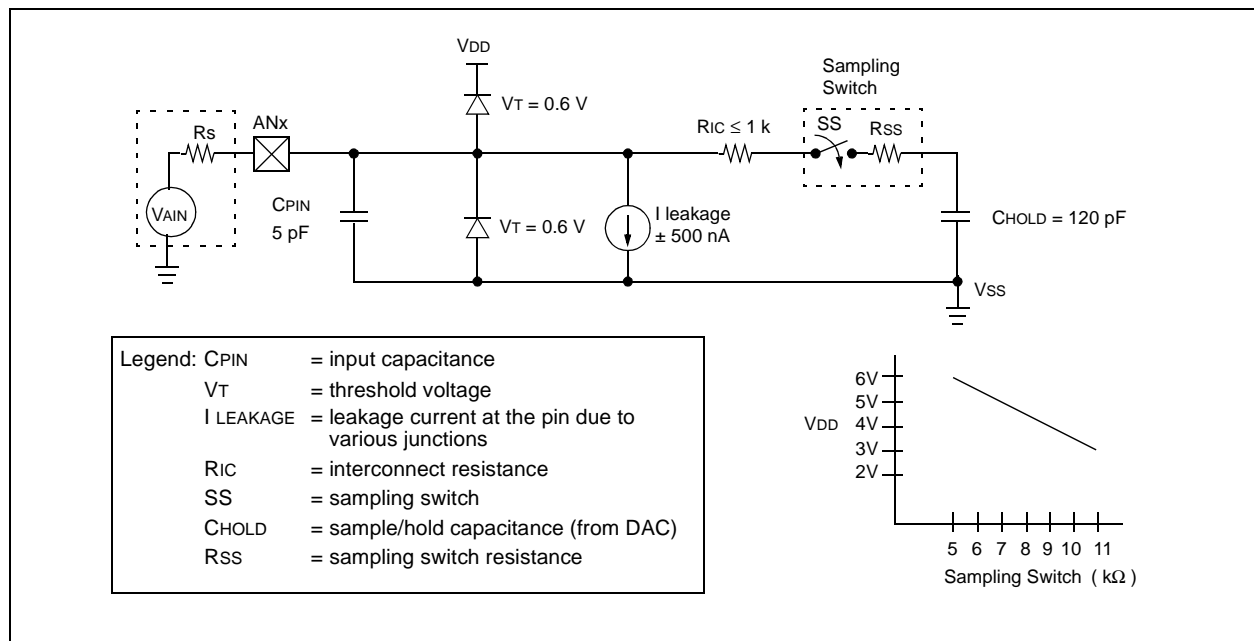


The value in the ADRESH/ADRESL registers is not modified for a Power-on Reset. The ADRESH/ADRESL registers will contain unknown data after a Power-on Reset.

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 18.1. 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/DONE bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
 - Polling for the GO/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 18-2: ANALOG INPUT MODEL



PIC18CXX8

18.1 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 18-2. 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 18-1 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 18-1 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 18-1: ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{Amplifier Settling Time} + \\ &\quad \text{Holding Capacitor Charging Time} + \\ &\quad \text{Temperature Coefficient} \\ &= \text{TAMP} + \text{Tc} + \text{TCOFF} \end{aligned}$$

EQUATION 18-2: A/D MINIMUM CHARGING TIME

$$\begin{aligned} \text{VHOLD} &= (\text{VREF} - (\text{VREF}/2048)) \cdot (1 - e^{(-\text{Tc}/\text{CHOLD}(\text{RIC} + \text{Rss} + \text{Rs}))}) \\ \text{or} \\ \text{Tc} &= -(120 \text{ pF})(1 \text{ k}\Omega + \text{Rss} + \text{Rs}) \ln(1/2047) \end{aligned}$$

EXAMPLE 18-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{TAMP} + \text{Tc} + \text{TCOFF} \\ \text{Temperature coefficient is only required for temperatures} &> 25^\circ\text{C}. \\ \text{TACQ} &= 2 \mu\text{s} + \text{Tc} + [(50^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\ \text{Tc} &= -\text{CHOLD} (\text{RIC} + \text{Rss} + \text{Rs}) \ln(1/2047) \\ &= -120 \text{ pF} (1 \text{ k}\Omega + 7 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(0.0004885) \\ &= -120 \text{ pF} (10.5 \text{ k}\Omega) \ln(0.0004885) \\ &= -1.26 \mu\text{s} (-7.6241) \\ &= 9.61 \mu\text{s} \\ \text{TACQ} &= 2 \mu\text{s} + 9.61 \mu\text{s} + [(50^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\ &= 11.61 \mu\text{s} + 1.25 \mu\text{s} \\ &= 12.86 \mu\text{s} \end{aligned}$$

18.2 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 μ s.

Table 18-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

18.3 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 18-1: TAD vs. DEVICE OPERATING FREQUENCIES

| AD Clock Source (TAD) | | Maximum Device Frequency | |
|-----------------------|-------------|--------------------------|---------------------------|
| Operation | ADCS2:ADCS0 | PIC18CXX8 | PIC18LCXX8 ⁽⁶⁾ |
| 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 | 40.0 MHz | 21.33 MHz |
| RC | x11 | — | — |

- Note 1:** The RC source has a typical TAD time of 4 ms.
2: The RC source has a typical TAD time of 6 ms.
3: These values violate the minimum required TAD time.
4: For faster conversion times, the selection of another clock source is recommended.
5: 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.
6: This column is for the LC devices only.

18.4 A/D Conversions

Figure 18-3 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.

18.5 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 18-3: A/D CONVERSION TAD CYCLES

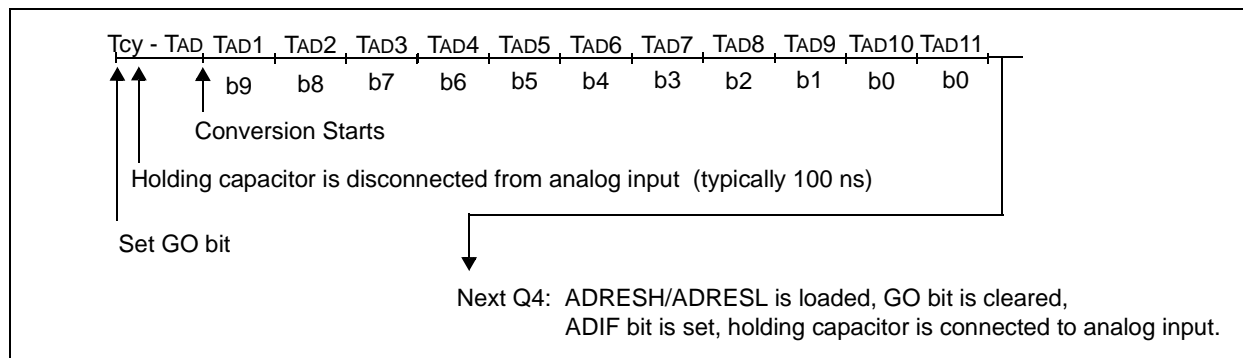


TABLE 18-2: SUMMARY OF A/D REGISTERS

| 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 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 0000 0000 | 0000 0000 |
| PIR2 | — | CMIF | — | — | BCLIF | LVDIF | TMR3IF | CCP2IF | -0-- 0000 | -0-- 0000 |
| PIE2 | — | CMIE | — | — | BCLIE | LVDIE | TMR3IE | CCP2IE | -0-- 0000 | -0-- 0000 |
| IPR2 | — | CMIP | — | — | BCLIP | LVDIP | TMR3IP | CCP2IP | -0-- 0000 | -0-- 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/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 | — | RA6 | 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 PIC18C858 devices.

PIC18CXX8

NOTES:

19.0 COMPARATOR MODULE

The comparator module contains two analog comparators. The inputs to the comparators are multiplexed with the RF1 through RF6 pins. The on-chip Voltage Reference (Section 20.0) can also be an input to the comparators.

The CMCON register, shown as Register 19-1, controls the comparator input and output multiplexers. A block diagram of the comparator is shown in Figure 19-1.

REGISTER 19-1: CMCON REGISTER

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 |
| bit 7 | | | | | | bit 0 | |

bit 7 **C2OUT**: Comparator 2 Output

When C2INV = 0:

1 = C2 VIN+ > C2 VIN-

0 = C2 VIN+ < C2 VIN-

When C2INV = 1:

1 = C2 VIN+ < C2 VIN-

0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**: Comparator 1 Output

When C1INV = 0:

1 = C1 VIN+ > C1 VIN-

0 = C1 VIN+ < C1 VIN-

When C1INV = 1:

1 = C1 VIN+ < C1 VIN-

0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**: Comparator 2 Output Inversion

1 = C2 output inverted

0 = C2 output not inverted

bit 4 **C1INV**: Comparator 1 Output Inversion

1 = C1 Output inverted

0 = C1 Output not inverted

bit 3 **CIS**: Comparator Input Switch

When CM2:CM0 = 110:

1 = C1 VIN- connects to RF5/AN10

C2 VIN- connects to RF3/AN8

0 = C1 VIN- connects to RF6/AN11

C2 VIN- connects to RF4/AN9

bit 2-0 **CM2:CM0**: Comparator Mode

Figure 19-1 shows the Comparator modes and CM2:CM0 bit settings

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

PIC18CXX8

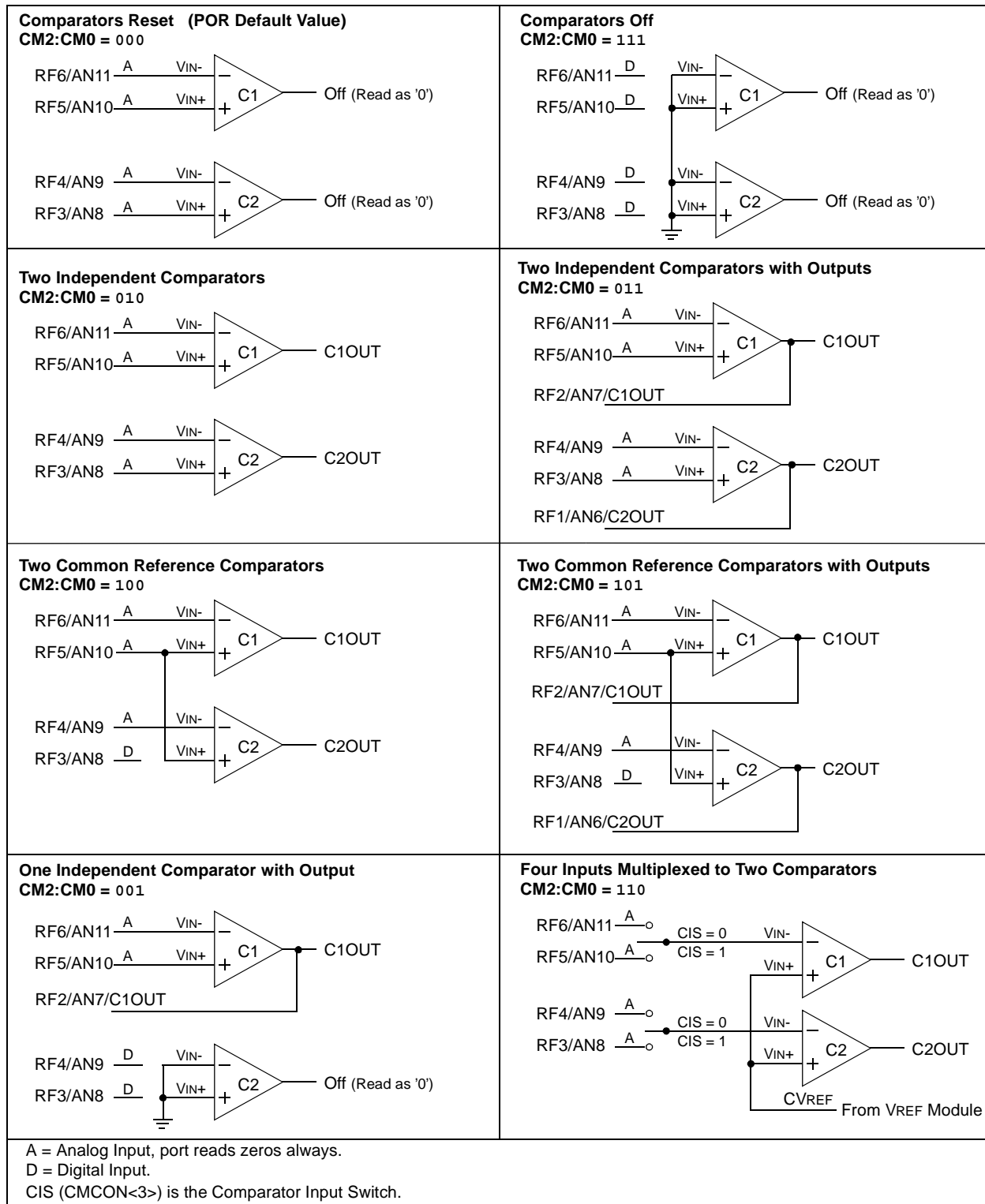
19.1 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select these modes. Figure 19-1 shows the eight possible modes. The TRISF register controls the data direction of the comparator pins for each mode. If the Comparator

mode is changed, the comparator output level may not be valid for the specified mode change delay shown in Electrical Specifications (Section 25.0).

Note: Comparator interrupts should be disabled during a Comparator mode change. Otherwise, a false interrupt may occur.

FIGURE 19-1: COMPARATOR I/O OPERATING MODES



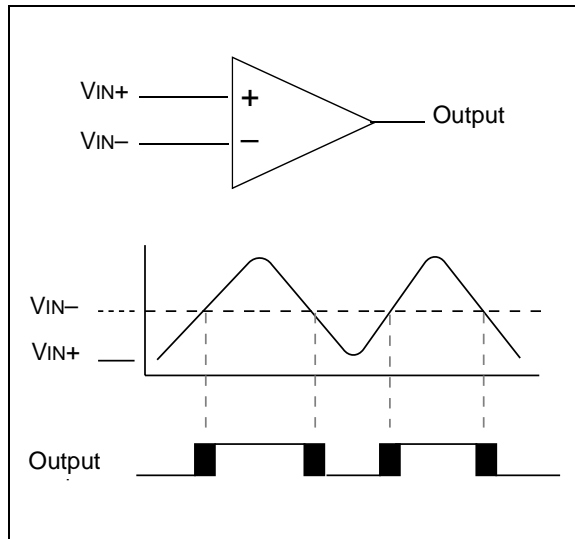
19.2 Comparator Operation

A single comparator is shown in Figure 19-2 along with the relationship between the analog input levels and the digital output. When the analog input at V_{IN+} is less than the analog input V_{IN-} , the output of the comparator is a digital low level. When the analog input at V_{IN+} is greater than the analog input V_{IN-} , the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 19-2 represent the uncertainty due to input offsets and response time.

19.3 Comparator Reference

An external or internal reference signal may be used depending on the comparator operating mode. The analog signal present at V_{IN-} is compared to the signal at V_{IN+} , and the digital output of the comparator is adjusted accordingly (Figure 19-2).

FIGURE 19-2: SINGLE COMPARATOR



19.3.1 EXTERNAL REFERENCE SIGNAL

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same, or different reference sources. However, threshold detector applications may require the same reference. The reference signal must be between V_{SS} and V_{DD} , and can be applied to either pin of the comparator(s).

19.3.2 INTERNAL REFERENCE SIGNAL

The comparator module also allows the selection of an internally generated voltage reference for the comparators. Section 20.0 contains a detailed description of the Comparator Voltage Reference Module that provides this signal. The internal reference signal is used when comparators are in mode $CM\langle 2:0 \rangle = 110$ (Figure 19-1). In this mode, the internal voltage reference is applied to the V_{IN+} pin of both comparators.

19.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise the maximum delay of the comparators should be used (Section 25.0).

19.5 Comparator Outputs

The comparator outputs are read through the CMCON Register. These bits are read-only. The comparator outputs may also be directly output to the RF1 and RF2 I/O pins. When enabled, multiplexors in the output path of the RF1 and RF2 pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 19-3 shows the comparator output block diagram.

The TRISA bits will still function as an output enable/disable for the RF1 and RF2 pins while in this mode.

The polarity of the comparator outputs can be changed using the C2INV and C1INV bits ($CMCON\langle 4:5 \rangle$).

Note 1: When reading the PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input, according to the Schmitt Trigger input specification.

2: Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

19.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that occurred. The CMIF bit (PIR registers) is the comparator interrupt flag. The CMIF bit must be RESET by clearing '0'. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

The CMIE bit (PIE registers) and the PEIE bit (INTCON register) must be set to enable the interrupt. In addition, the GIE bit must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

Note: If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR registers) interrupt flag may not get set.

The user, in the interrupt service routine, can clear the interrupt in the following manner:

- a) Any read or write of CMCON will end the mismatch condition.
- b) Clear flag bit CMIF.

A mismatch condition will continue to set flag bit CMIF. Reading CMCON will end the mismatch condition, and allow flag bit CMIF to be cleared.

19.7 Comparator Operation During SLEEP

When a comparator is active and the device is placed in SLEEP mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake-up the device from SLEEP mode, when enabled. While the comparator is powered up, higher SLEEP currents than shown in the power-down current specification will occur. Each operational comparator will consume additional current, as shown in the comparator specifications. To minimize power consumption while in SLEEP mode, turn off the comparators, CM<2:0> = 111, before entering SLEEP. If the device wakes up from SLEEP, the contents of the CMCON register are not affected.

19.8 Effects of a RESET

A device RESET forces the CMCON register to its RESET state, causing the comparator module to be in the comparator RESET mode, CM<2:0> = 000. This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at RESET time. The comparators will be powered down during the RESET interval.

PIC18CXX8

19.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 19-4. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6 V in either direction, one of the diodes is forward biased and a latch-up condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 19-4: ANALOG INPUT MODEL

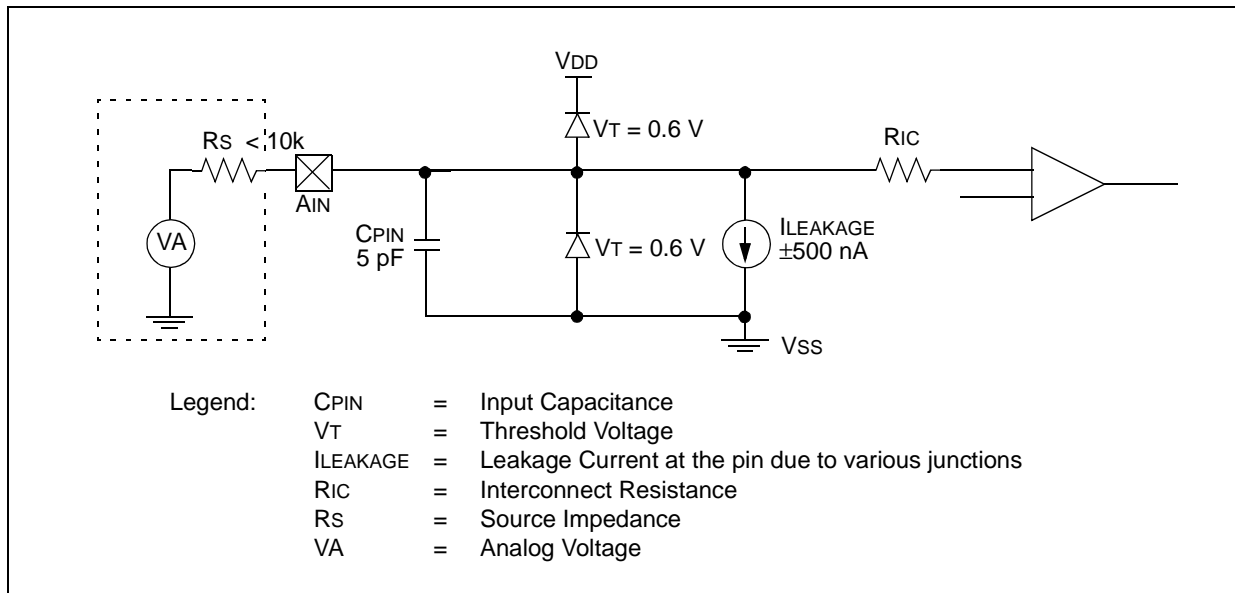


TABLE 19-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

| 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 |
|--------|-------------------------------|---------------|--------|-------|-------|--------|--------|--------|--------------|---------------------------|
| CMCON | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 0000 0000 | 0000 0000 |
| VRCON | VREN | VROE | VRR | VRSS | VR3 | VR2 | VR1 | VR0 | 0000 0000 | 0000 0000 |
| INTCON | GIE/ GIEH | PEIE/ GIEL | TMR0IE | INTIE | RBIE | TMR0IF | INTIF | RBIF | 0000 000x | 0000 000u |
| PIR2 | — | CMIF | — | — | BCLIF | LVDIF | TMR3IF | CCP2IF | -0-- 0000 | -0-- 0000 |
| PIE2 | — | CMIE | — | — | BCLIE | LVDIE | TMR3IE | CCP2IE | -0-- 0000 | -0-- 0000 |
| IPR2 | — | CMIP | — | — | BCLIP | LVDIP | TMR3IP | CCP2IP | -1-- 1111 | -1-- 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 Register | | | | | | | | 1111 1111 | 1111 1111 |

Legend: x = unknown, u = unchanged, - = unimplemented, read as "0"

20.0 COMPARATOR VOLTAGE REFERENCE MODULE

The Comparator Voltage Reference is a 16-tap resistor ladder network that provides a selectable voltage reference. The resistor ladder is segmented to provide two ranges of CVREF values and has a power-down function to conserve power when the reference is not being used. The CVRCON register controls the operation of the reference as shown in Register 20-1. The block diagram is given in Figure 20-1.

The comparator reference supply voltage can come from either VDD or VSS, or the external VREF+ and VREF- that are multiplexed with RA3 and RA2. The comparator reference supply voltage is controlled by the CVRSS bit.

20.1 Configuring the Comparator Voltage Reference

The Comparator Voltage Reference can output 16 distinct voltage levels for each range. The equations used to calculate the output of the Comparator Voltage Reference are as follows:

If CVRR = 1:

$$CVREF = (CVR<3:0>/24) \times CVRSRC$$

If CVRR = 0:

$$CVREF = (CVDD \times 1/4) + (CVR<3:0>/32) \times CVRSRC$$

The settling time of the Comparator Voltage Reference must be considered when changing the CVREF output (Section 25.0).

REGISTER 20-1: VRCON 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 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| VREN | VROE | VRR | VRSS | VR3 | VR2 | VR1 | VR0 |
| | | | | | | | bit 0 |
| bit 7 | | | | | | | |

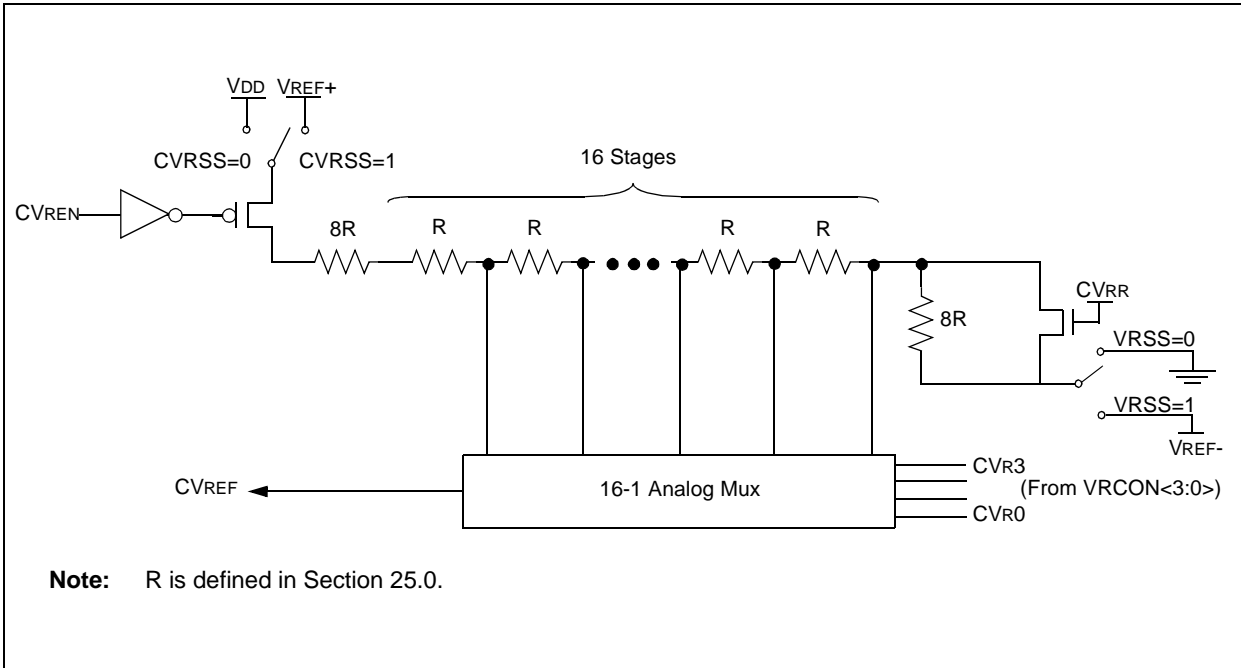
- bit 7 **VREN:** Comparator Voltage Reference Enable
1 = CVREF circuit powered on
0 = CVREF circuit powered down
- bit 6 **VROE:** Comparator VREF Output Enable
1 = CVREF voltage level is also output on the RF5/AN10/CVREF pin
0 = CVREF voltage is disconnected from the RF5/AN10/CVREF pin
- bit 5 **VRR:** Comparator VREF Range Selection
1 = 0.00 CVRSRC to 0.75 CVRSRC, with CVRSRC/24 step size
0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size
- bit 4 **VRSS:** Comparator VREF Source Selection
1 = Comparator reference source CVRSRC = VREF+-VREF-
0 = Comparator reference source CVRSRC = VDD-VSS
- bit 3-0 **VR3:VR0:** Comparator VREF Value Selection $0 \leq VR3:VR0 \leq 15$
When VRR = 1:
 $CVREF = (VR<3:0>/24) \cdot (CVRSRC)$
When VRR = 0:
 $CVREF = 1/4 \cdot (CVRSRC) + (VR3:VR0/32) \cdot (CVRSRC)$

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 |

PIC18CXX8

FIGURE 20-1: VOLTAGE REFERENCE BLOCK DIAGRAM



20.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 20-1) keep VREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the VREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in Section 25.0.

20.3 Operation During SLEEP

When the device wakes up from SLEEP through an interrupt or a Watchdog Timer time-out, the contents of the VRCON register are not affected. To minimize current consumption in SLEEP mode, the voltage reference should be disabled.

20.4 Effects of a RESET

A device RESET disables the voltage reference by clearing bit VREN (VRCON register). This RESET also disconnects the reference from the RA2 pin by clearing bit VROE (VRCON register) and selects the high voltage range by clearing bit CVRR (VRCON register). The VRSS value select bits, CVRCON<3:0>, are also cleared.

20.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RF5 pin if the TRISF<5> bit is set and the VROE bit (VRCON register) is set. Enabling the voltage reference output onto the RF5 pin, with an input signal present, will increase current consumption. Connecting RF5 as a digital output with VRSS enabled will also increase current consumption.

The RF5 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 20-2 shows an example buffering technique.

FIGURE 20-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE

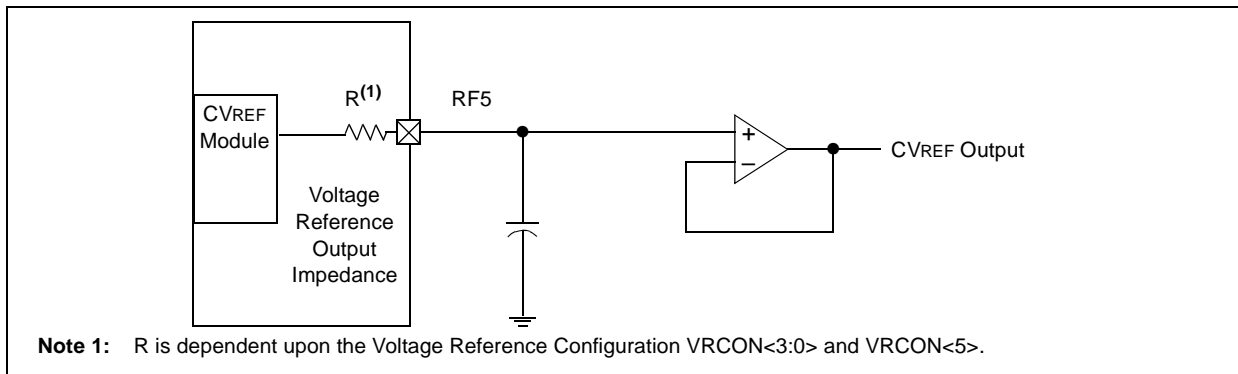


TABLE 20-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

| 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 |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------------|---------------------------|
| VRCON | VREN | VROE | VRR | VRSS | VR3 | VR2 | VR1 | VR0 | 0000 0000 | 0000 0000 |
| CMCON | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 0000 0000 | 0000 0000 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | TRISF0 | 1111 1111 | 1111 1111 |

PIC18CXX8

NOTES:

21.0 LOW VOLTAGE DETECT

In many applications, the ability to determine if the device voltage (VDD) 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 21-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage VA, the LVD logic generates an interrupt. This occurs at time TA. 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 VB is the minimum valid operating voltage specification. This occurs at time TB. TB - TA is the total time for shutdown.

FIGURE 21-1: TYPICAL LOW VOLTAGE DETECT APPLICATION

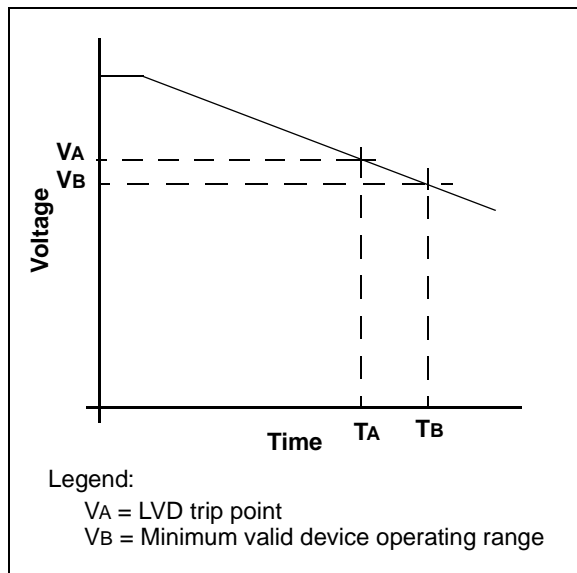
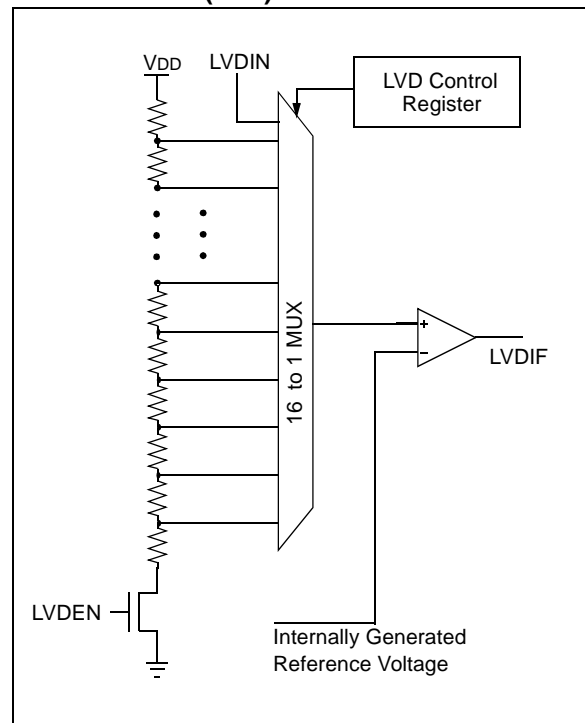


Figure 21-2 shows the block diagram for the LVD module. A comparator uses an internally generated reference voltage as the set point. When the selected tap output of the device voltage crosses the set point (is lower than), the LVDIF bit (PIR registers) is set.

Each node in the resistor 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 21-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

FIGURE 21-2: LOW VOLTAGE DETECT (LVD) BLOCK DIAGRAM



PIC18CXX8

21.1 Control Register

The Low Voltage Detect Control register (Register 21-1) controls the operation of the Low Voltage Detect circuitry.

REGISTER 21-1: 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 min - 4.77V max.

1101 = 4.2V min - 4.45V max.

1100 = 4.0V min - 4.24V max.; Reserved on PIC18CXX8

1011 = 3.8V min - 4.03V max.; Reserved on PIC18CXX8

1010 = 3.6V min - 3.82V max.; Reserved on PIC18CXX8

1001 = 3.5V min - 3.71V max.; Reserved on PIC18CXX8

1000 = 3.3V min - 3.50V max.; Reserved on PIC18CXX8

0111 = 3.0V min - 3.18V max.; Reserved on PIC18CXX8

0110 = 2.8V min - 2.97V max.; Reserved on PIC18CXX8

0101 = 2.7V min - 2.86V max.; Reserved on PIC18CXX8

0100 = 2.5V min - 2.65V max.; Reserved on PIC18CXX8

0011 = Reserved on PIC18CXX8 and PIC18LCXX8

0010 = Reserved on PIC18CXX8 and PIC18LCXX8

0001 = Reserved on PIC18CXX8 and PIC18LCXX8

0000 = Reserved on PIC18CXX8 and PIC18LCXX8

Note: 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

21.2 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.

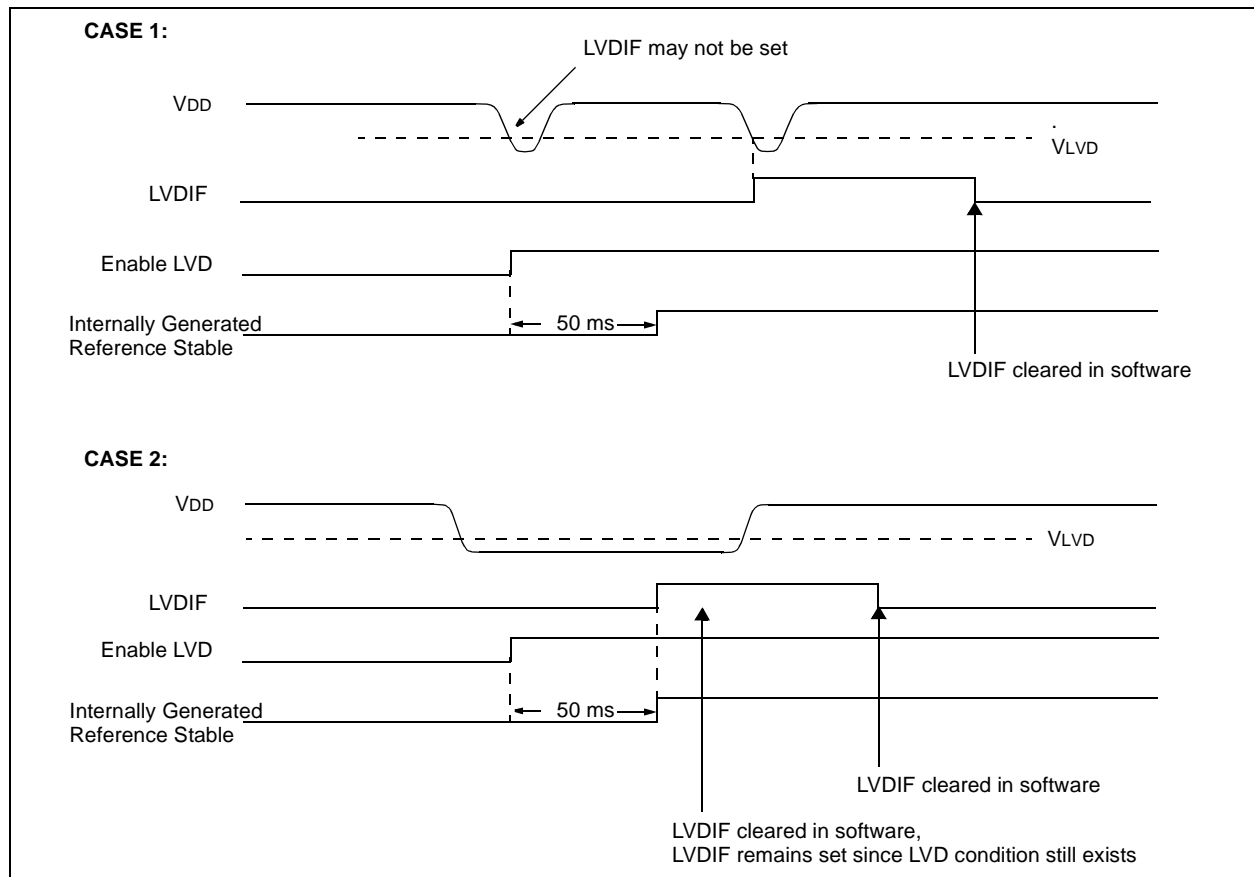
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

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 21-3 shows typical waveforms that the LVD module may be used to detect.

FIGURE 21-3: LOW VOLTAGE DETECT WAVEFORMS



21.2.1 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 21-3.

21.2.2 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.

21.3 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 = '1111'. This state connects the LVDIN pin voltage to the comparator. The other comparator input is connected to an internal reference voltage source.

21.4 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.

21.5 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the LVD module to be turned off.

22.0 SPECIAL FEATURES OF THE CPU

There are several features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection:

- OSC Selection
- RESET
 - Power-on Reset (POR)
 - Power-up Timer (PWRT)
 - Oscillator Start-up Timer (OST)
 - Programmable Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code Protection
- ID Locations
- In-circuit Serial Programming

PIC18CXX8 devices have a Watchdog Timer, which is permanently enabled via the configuration bits or it can be software-controlled. It runs off its own RC oscillator for added reliability. 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 made available to allow the part to fit the application. The RC oscillator option saves system cost, while the LP crystal option saves power. A set of configuration bits are used to select various options.

22.1 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 - 3FFFFh), which can only be accessed using table reads and table writes.

TABLE 22-1: CONFIGURATION BITS AND DEVICE ID'S

| Filename | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Default/ Unprogrammed Value |
|----------|----------|-------|-------|----------------------------|-------|--------|--------|--------|----------------------------|-----------------------------------|
| 300000h | CONFIG1L | CP | CP | CP | CP | CP | CP | CP | CP | 1111 1111 |
| 300001h | CONFIG1H | r | r | $\overline{\text{OSCSEN}}$ | — | — | FOSC2 | FOSC1 | FOSC0 | 111- -111 |
| 300002h | CONFIG2L | — | — | — | — | BORV1 | BORV0 | BODEN | $\overline{\text{PWRTEN}}$ | ---- 1111 |
| 300003h | CONFIG2H | — | — | — | — | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN | ---- 1111 |
| 300006h | CONFIG4L | — | — | — | — | — | — | r | STVREN | ---- --11 |
| 3FFFEh | DEVID1 | DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | 1111 1111 |
| 3FFFFh | DEVID2 | DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | 1111 1111 |

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

PIC18CXX8

REGISTER 22-1: CONFIGURATION REGISTER 1 LOW (CONFIG1L: BYTE ADDRESS 0x300000)

| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| CP | CP | CP | CP | CP | CP | CP | CP |

bit 7 bit 0

bit 7-0 **CP:** Code Protection bits (apply when in Code Protected Microcontroller mode)
 1 = Program memory code protection off
 0 = All of program memory code protected

| | | | | | | | |
|-----------------------------------------|----------------------|------------------------------------|--|-------------------------------------|--|--|--|
| Legend: | | | | | | | |
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' | | | | | |
| - n = Value when device is unprogrammed | | | | u = Unchanged from programmed state | | | |

REGISTER 22-2: CONFIGURATION REGISTER 1 HIGH (CONFIG1H: BYTE ADDRESS 0x300001)

| R/P-1 | R/P-1 | R/P-1 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 |
|----------|----------|--------|-----|-----|-------|-------|-------|
| Reserved | Reserved | OSCSEN | — | — | FOSC2 | FOSC1 | FOSC0 |

bit 7 bit 0

bit 7-6 **Reserved:** Maintain this bit set

bit 5 **OSCSEN:** Oscillator System Clock Switch Enable bit
 1 = Oscillator system clock switch option is disabled (Main oscillator is source)
 0 = Oscillator system clock switch option is enabled (Oscillator switching is enabled)

bit 4-3 **Unimplemented:** Read as '0'

bit 2-0 **FOSC2:FOSC0:** Oscillator Selection bits
 111 = RC oscillator w/ OSC2 configured as RA6
 110 = HS4 oscillator with PLL enabled/Clock frequency = (4 x Fosc)
 101 = EC oscillator w/ OSC2 configured as RA6
 100 = EC oscillator w/ OSC2 configured as divide by 4 clock output
 011 = RC oscillator
 010 = HS oscillator
 001 = XT oscillator
 000 = LP oscillator

| | | | | | | | |
|-----------------------------------------|----------------------|------------------------------------|--|-------------------------------------|--|--|--|
| Legend: | | | | | | | |
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' | | | | | |
| - n = Value when device is unprogrammed | | | | u = Unchanged from programmed state | | | |

REGISTER 22-3: CONFIGURATION REGISTER 2 LOW (CONFIG2L: BYTE ADDRESS 0x300002)

| U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|-------|-----|-----|-----|-------|-------|-------|--------|
| — | — | — | — | BORV1 | BORV0 | BOREN | PWRTEN |
| bit 7 | | | | bit 0 | | | |

- bit 7-4 **Unimplemented:** Read as '0'
- bit 3-2 **BORV1:BORV0:** Brown-out Reset Voltage bits
 11 = VBOR set to 2.5V
 10 = VBOR set to 2.7V
 01 = VBOR set to 4.2V
 00 = VBOR set to 4.5V
- bit 1 **BOREN:** Brown-out Reset Enable bit⁽¹⁾
 1 = Brown-out Reset enabled
 0 = Brown-out Reset disabled
- bit 0 **PWRTEN:** Power-up Timer Enable bit⁽¹⁾
 1 = PWRT disabled
 0 = PWRT enabled

Note 1: Enabling Brown-out Reset automatically enables the Power-up Timer (PWRT), regardless of the value of bit PWRTEN. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.

| | | |
|-----------------------------------------|-------------------------------------|------------------------------------|
| Legend: | | |
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| - n = Value when device is unprogrammed | u = Unchanged from programmed state | |

REGISTER 22-4: CONFIGURATION REGISTER 2 HIGH (CONFIG2H: BYTE ADDRESS 0x300003)

| 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 = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| - n = Value when device is unprogrammed | u = Unchanged from programmed state | |

PIC18CXX8

REGISTER 22-5: CONFIGURATION REGISTER 4 LOW (CONFIG4L: BYTE ADDRESS 0x300006)

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|----------|--------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 |
| — | — | — | — | — | — | Reserved | STVREN |
| bit 7 | | | | | | bit 0 | |

- bit 7-2 **Unimplemented:** Read as '0'
- bit 1 **Reserved:** Maintain this bit set
- 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 = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| - n = Value when device is unprogrammed | | u = Unchanged from programmed state |

22.2 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/RA6 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.

The Watchdog Timer is enabled/disabled by a device configuration bit. If the WDT is enabled, software execution may not disable this function. When the WDTEEN configuration bit is cleared, the SWDTEN bit 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 using the configuration bits.

Note: 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.

Note: 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.

22.2.1 CONTROL REGISTER

Register 22-6 shows the WDTCON register. This is a readable and writable register, which contains a control bit that allows software to override the WDT enable configuration bit, only when the configuration bit has disabled the WDT.

REGISTER 22-6: WDTCON REGISTER

| | | | | | | | | |
|-------|-----|-----|-----|-----|-----|-----|-----|--------|
| | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
| | — | — | — | — | — | — | — | SWDTEN |
| bit 7 | | | | | | | | bit 0 |

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SWDTEN:** Software Controlled Watchdog Timer Enable bit
 1 = Watchdog Timer is on
 0 = Watchdog Timer is turned off if the WDTEEN configuration bit in the configuration register = '0'

Legend:

R = Readable bit W = Writable bit
 U = Unimplemented bit, read as '0' - n = Value at POR

PIC18CXX8

22.2.2 WDT POSTSCALER

The WDT has a postscaler that can extend the WDT Reset period. The postscaler is selected at the time of the device programming, by the value written to the CONFIG2H configuration register.

FIGURE 22-1: WATCHDOG TIMER BLOCK DIAGRAM

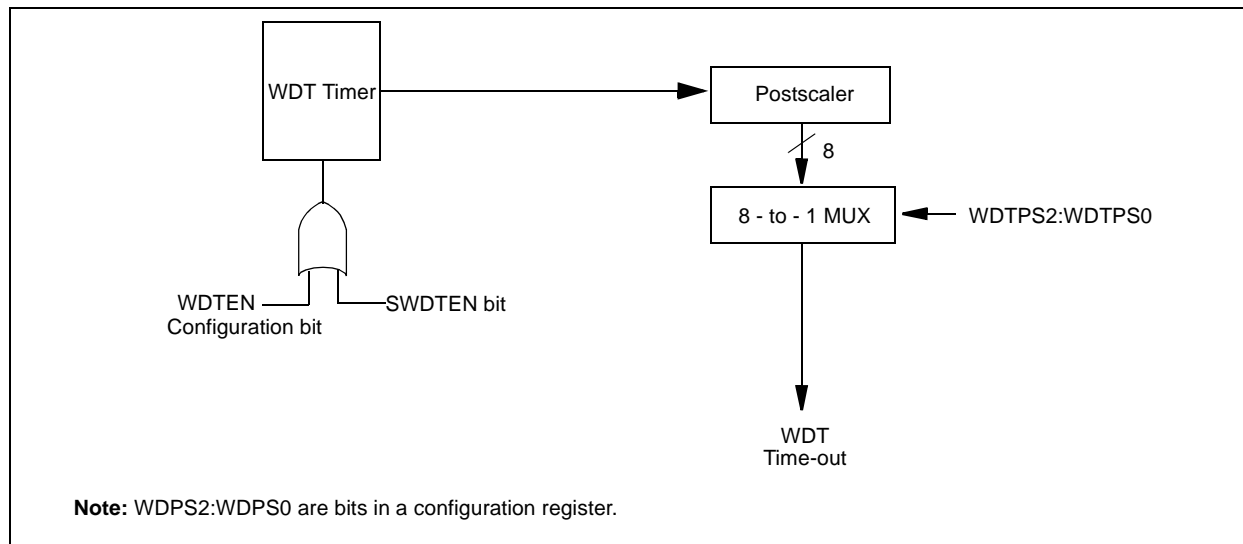


TABLE 22-2: SUMMARY OF WATCHDOG TIMER REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|----------|-------|-------|-------|-----------------|-----------------|-----------------|------------------|------------------|
| CONFIG2H | — | — | — | — | WDTPS2 | WDTPS2 | WDTPS0 | WDTEN |
| RCON | IPEN | LWRT | — | \overline{RI} | \overline{TO} | \overline{PD} | \overline{POR} | \overline{BOR} |
| WDTCON | — | — | — | — | — | — | — | SWDTEN |

Legend: Shaded cells are not used by the Watchdog Timer.

22.3 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 V_{DD} or V_{SS} 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 V_{SS} or V_{DD} .
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 (V_{IHMC}).

22.3.1 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:

1. PSP read or write.
2. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
3. TMR3 interrupt. Timer3 must be operating as an asynchronous counter.
4. CCP Capture mode interrupt.
5. Special event trigger (Timer1 in Asynchronous mode using an external clock).
6. MSSP (START/STOP) bit detect interrupt.
7. MSSP transmit or receive in Slave mode (SPI/I²C).
8. USART RX or TX (Synchronous Slave mode).
9. A/D conversion (when A/D clock source is RC).
10. Activity on CAN bus receive line.

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.

22.3.2 WAKE-UP USING INTERRUPTS

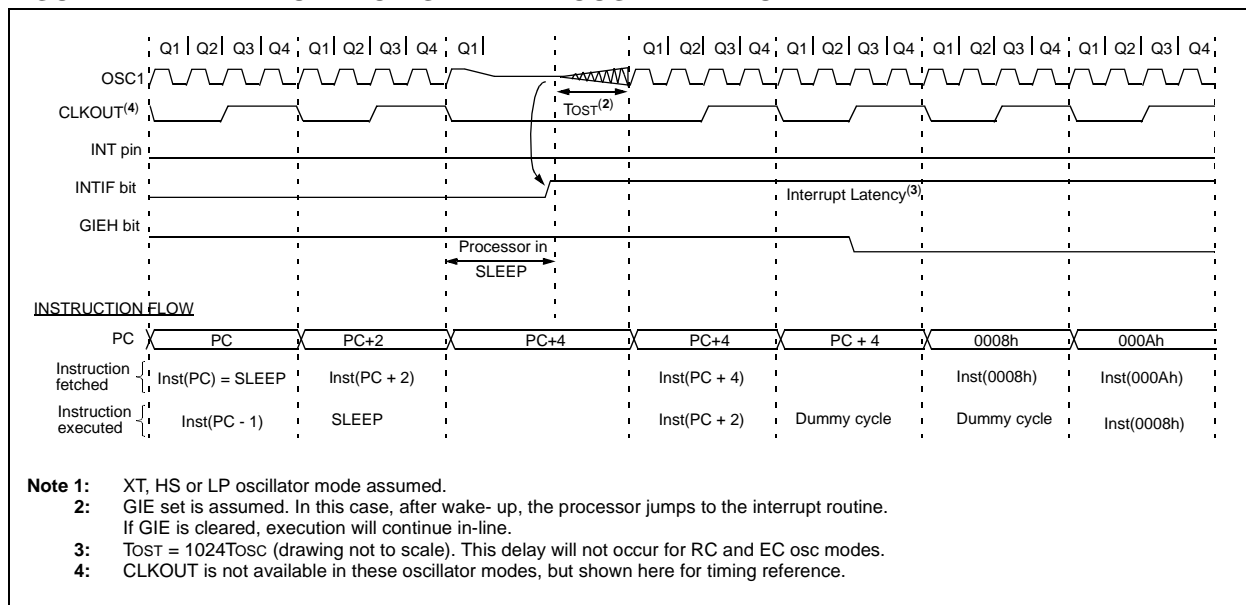
When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- 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 22-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT^(1,2)



22.4 Program Verification/Code Protection

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

Note: Microchip Technology does not recommend code protecting windowed devices.

22.5 ID Locations

Five memory locations (200000h - 200004h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are accessible during normal execution through the TBLRD instruction, or during program/verify. The ID locations can be read when the device is code protected.

22.6 In-Circuit Serial Programming

PIC18CXX8 microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

22.7 Device ID Bits

Device ID bits are located in program memory at 3FFFEh and 3FFFFh. The Device ID bits are used by programmers to retrieve part number and revision information about a device. These registers may also be accessed using a TBLRD instruction (Register 22-8 and Register 22-7).

REGISTER 22-7: DEVID1 ID REGISTER FOR THE PIC18CXX8 DEVICE (0x3FFFE)

| | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
| DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | |
| bit 7 | | | | | | | bit 0 | |

- bit 7-5 **DEV2:DEV0:** Device ID bits
These bits are used with the DEV10:DEV3 bits in the Device ID register 2 to identify the part number
- bit 4-0 **REV4:REV0:** Revision ID bits
These bits are used to indicate the revision of the device

Legend:

| | |
|------------------------------------|-------------------------------------------|
| R = Readable bit | P = Programmable bit |
| U = Unimplemented bit, read as '0' | - n = Unprogrammed Value (x = unknown) |

REGISTER 22-8: DEVID2 ID REGISTER FOR THE PIC18CXX8 DEVICE (0x3FFFF)

| | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
| DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | |
| bit 7 | | | | | | | bit 0 | |

- bit 7-0 **DEV10:DEV3:** Device ID bits
These bits are used with the DEV2:DEV0 bits in the Device ID register 1 to identify the part number

Legend:

| | |
|------------------------------------|-------------------------------------------|
| R = Readable bit | P = Programmable bit |
| U = Unimplemented bit, read as '0' | - n = Unprogrammed Value (x = unknown) |

PIC18CXX8

NOTES:

23.0 INSTRUCTION SET SUMMARY

The PIC18CXX8 instruction set adds many enhancements to the previous PICmicro[®] instruction sets, while maintaining an easy migration from these PICmicro instruction sets.

Most instructions are a single program memory word (16-bits), but there are three instructions that require two program memory locations.

Each single word instruction is a 16-bit word 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 PIC18CXX8 instruction set summary in Table 23-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 23-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by the value of 'f')
2. The destination of the result (specified by the value of 'd')
3. The accessed memory (specified by the value of 'a')

'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator 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 (specified by the value of 'f')
2. The bit in the file register (specified by the value of 'b')
3. The accessed memory (specified by the value of 'a')

'b' represents a bit field designator which selects the number of the bit affected by the operation, while '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 (specified by the value of 'k')
- The desired FSR register to load the literal value into (specified by the value of 'f')
- No operand required (specified by the value of '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by the value of 'n')
- The mode of the Call or Return instructions (specified by the value of 's')
- The mode of the Table Read and Table Write instructions (specified by the value of 'm')
- No operand required (specified by the value of '—')

All instructions are a single word, except for four double word instructions. These three instructions were made double word instructions so that all the required information is available in these 32-bits. In the second word, the 4-MSb's 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 μ s. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 μ s. Two word branch instructions (if true) would take 3 μ s.

Figure 23-1 shows the general formats that the instructions can have.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 23-2, lists the instructions recognized by the Microchip assembler (MPASM[™]).

Section 23.1 provides a description of each instruction.

PIC18CXX8

TABLE 23-1: 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 (0x00 to 0xFF) |
| f _s | 12-bit Register file address (0x000 to 0xFFF). This is the source address. |
| f _d | 12-bit Register file address (0x000 to 0xFFF). 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 0xFF4) |
| PRODL | Product of Multiply low byte (Register at address 0xFF3) |
| 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 0xFE8) |
| W | W = 0: Destination select bit symbol |
| WREG | Working register (accumulator) (Register at address 0xFE8) |
| 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 0xFF6) |
| TABLAT | 8-bit Table Latch (Register at address 0xFF5) |
| TOS | Top-of-Stack |
| PC | Program Counter |
| PCL | Program Counter Low Byte (Register at address 0xFF9) |
| PCH | Program Counter High Byte |
| PCLATH | Program Counter High Byte Latch (Register at address 0xFFA) |
| PCLATU | Program Counter Upper Byte Latch (Register at address 0xFFB) |
| GIE | Global Interrupt Enable bit |
| WDT | Watchdog Timer |
| T \bar{O} | Time-out bit |
| P \bar{D} | 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 |
| <i>italics</i> | User defined term (font is courier) |

FIGURE 23-1: GENERAL FORMAT FOR INSTRUCTIONS

| Byte-oriented file register operations | Example Instruction | | | | | | | | | | | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|------------------|---|--------|-------------------|------------------|-------------------|-------|---|------|------------------------|-------------|----------------------|
| <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">10 9 8 7</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td style="border: 1px solid black; padding: 2px;">d a</td> <td style="border: 1px solid black; padding: 2px;">f (FILE #)</td> </tr> </table> <p>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</p> | 15 | 10 9 8 7 | 0 | OPCODE | d a | f (FILE #) | ADDWF MYREG, W, B | | | | | | |
| 15 | 10 9 8 7 | 0 | | | | | | | | | | | |
| OPCODE | d a | f (FILE #) | | | | | | | | | | | |
| <p>Byte to Byte move operations (2-word)</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">12 11</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">f (Source FILE #)</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">12 11</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">1111</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">f (Destination FILE #)</td> </tr> </table> <p>f = 12-bit file register address</p> | 15 | 12 11 | 0 | OPCODE | f (Source FILE #) | | 15 | 12 11 | 0 | 1111 | f (Destination FILE #) | | MOVFF MYREG1, MYREG2 |
| 15 | 12 11 | 0 | | | | | | | | | | | |
| OPCODE | f (Source FILE #) | | | | | | | | | | | | |
| 15 | 12 11 | 0 | | | | | | | | | | | |
| 1111 | f (Destination FILE #) | | | | | | | | | | | | |
| <p>Bit-oriented file register operations</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">12 11 9 8 7</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td style="border: 1px solid black; padding: 2px;">b (BIT #) a</td> <td style="border: 1px solid black; padding: 2px;">f (FILE #)</td> </tr> </table> <p>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</p> | 15 | 12 11 9 8 7 | 0 | OPCODE | b (BIT #) a | f (FILE #) | BSF MYREG, bit, B | | | | | | |
| 15 | 12 11 9 8 7 | 0 | | | | | | | | | | | |
| OPCODE | b (BIT #) a | f (FILE #) | | | | | | | | | | | |
| <p>Literal operations</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">8 7</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">k (literal)</td> </tr> </table> <p>k = 8-bit immediate value</p> | 15 | 8 7 | 0 | OPCODE | k (literal) | | MOVLW 0x7F | | | | | | |
| 15 | 8 7 | 0 | | | | | | | | | | | |
| OPCODE | k (literal) | | | | | | | | | | | | |
| <p>Control operations</p> <p>CALL, GOTO and Branch operations</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">8 7</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">n<7:0> (literal)</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">12 11</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">1111</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">n<19:8> (literal)</td> </tr> </table> <p>n = 20-bit immediate value</p> | 15 | 8 7 | 0 | OPCODE | n<7:0> (literal) | | 15 | 12 11 | 0 | 1111 | n<19:8> (literal) | | GOTO Label |
| 15 | 8 7 | 0 | | | | | | | | | | | |
| OPCODE | n<7:0> (literal) | | | | | | | | | | | | |
| 15 | 12 11 | 0 | | | | | | | | | | | |
| 1111 | n<19:8> (literal) | | | | | | | | | | | | |
| <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">8 7</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td style="border: 1px solid black; padding: 2px;">S</td> <td style="border: 1px solid black; padding: 2px;">n<7:0> (literal)</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">12 11</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">1111</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">n<19:8> (literal)</td> </tr> </table> <p>S = Fast bit</p> | 15 | 8 7 | 0 | OPCODE | S | n<7:0> (literal) | 15 | 12 11 | 0 | 1111 | n<19:8> (literal) | | CALL MYFUNC |
| 15 | 8 7 | 0 | | | | | | | | | | | |
| OPCODE | S | n<7:0> (literal) | | | | | | | | | | | |
| 15 | 12 11 | 0 | | | | | | | | | | | |
| 1111 | n<19:8> (literal) | | | | | | | | | | | | |
| <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">11 10</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">n<10:0> (literal)</td> </tr> </table> | 15 | 11 10 | 0 | OPCODE | n<10:0> (literal) | | BRA MYFUNC | | | | | | |
| 15 | 11 10 | 0 | | | | | | | | | | | |
| OPCODE | n<10:0> (literal) | | | | | | | | | | | | |
| <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">8 7</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">n<7:0> (literal)</td> </tr> </table> | 15 | 8 7 | 0 | OPCODE | n<7:0> (literal) | | BC MYFUNC | | | | | | |
| 15 | 8 7 | 0 | | | | | | | | | | | |
| OPCODE | n<7:0> (literal) | | | | | | | | | | | | |
| <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">6 4</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td> <td style="border: 1px solid black; padding: 2px;">f</td> <td style="border: 1px solid black; padding: 2px;">k (literal)</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">11 7</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">1111</td> <td style="border: 1px solid black; padding: 2px;">0000</td> <td style="border: 1px solid black; padding: 2px;">k (literal)</td> </tr> </table> | 15 | 6 4 | 0 | OPCODE | f | k (literal) | 15 | 11 7 | 0 | 1111 | 0000 | k (literal) | LFSR FSR0, 0x100 |
| 15 | 6 4 | 0 | | | | | | | | | | | |
| OPCODE | f | k (literal) | | | | | | | | | | | |
| 15 | 11 7 | 0 | | | | | | | | | | | |
| 1111 | 0000 | k (literal) | | | | | | | | | | | |

PIC18CXX8

TABLE 23-2: PIC18CXX8 INSTRUCTION SET

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|-----------------------------------------------|----------------------------------------------------------------------------------|------------|-------------------------|------|------|------|--------------------|---------------|
| | | | MSb | | | LSb | | |
| 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 | 00da | 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 | |
| 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 |
| BTFSZ f, b [,a] | Bit Test f, Skip if Clear | 1 (2 or 3) | 1011 | bbba | ffff | ffff | None | 3, 4, 6 |
| BTFSZ 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 Assembler MASM automatically defaults destination bit 'd' to '1', while access bit 'a' defaults to '1' or '0' according to address of register being used.

TABLE 23-2: PIC18CXX8 INSTRUCTION SET (CONTINUED)

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---------------------------|-------------|--------------------------------|-------------------------|------|------|------|--------------------|-----------------------------------|
| | | | MSb | | | LSb | | |
| 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 Assembler MASM automatically defaults destination bit 'd' to '1', while access bit 'a' defaults to '1' or '0' according to address of register being used.

PIC18CXX8

TABLE 23-2: PIC18CXX8 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 literal (k) | 2 | 1110 | 1110 | 00ff | kkkk | None | |
| 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 `NOOP`.
- 4: Some instructions are 2 word instructions. The second word of these instructions will be executed as a `NOOP`, 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 Assembler MASM automatically defaults destination bit 'd' to '1', while access bit 'a' defaults to '1' or '0' according to address of register being used.

23.1 Instruction Set

ADDLW ADD literal to W

Syntax: [*label*] ADDLW *k*
 Operands: $0 \leq k \leq 255$
 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 W |

Example: ADDLW 0x15

Before Instruction

WREG = 0x10
 N = ?
 OV = ?
 C = ?
 DC = ?
 Z = ?

After Instruction

WREG = 0x25
 N = 0
 OV = 0
 C = 0
 DC = 0
 Z = 0

ADDWF ADD W to f

Syntax: [*label*] ADDWF *f* [,d] [,a]
 Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$
 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 = 0x17
 REG = 0xC2
 N = ?
 OV = ?
 C = ?
 DC = ?
 Z = ?

After Instruction

WREG = 0xD9
 REG = 0xC2
 N = 1
 OV = 0
 C = 0
 DC = 0
 Z = 0

PIC18CXX8

ADDWFC ADD WREG and Carry bit to f

Syntax: [*label*] ADDWFC f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

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 = 0x02
WREG = 0x4D
N = ?
OV = ?
DC = ?
Z = ?

After Instruction

C = 0
REG = 0x02
WREG = 0x50
N = 0
OV = 0
DC = 0
Z = 0

ANDLW AND literal with WREG

Syntax: [*label*] ANDLW k

Operands: $0 \leq k \leq 255$

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 W |

Example: ANDLW 0x5F

Before Instruction

WREG = 0xA3
N = ?
Z = ?

After Instruction

WREG = 0x03
N = 0
Z = 0

ANDWF AND WREG with f

Syntax: [*label*] ANDWF f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

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 = 0x17
 REG = 0xC2
 N = ?
 Z = ?

After Instruction

WREG = 0x02
 REG = 0xC2
 N = 0
 Z = 0

BC Branch if Carry

Syntax: [*label*] BC n

Operands: $-128 \leq n \leq 127$

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)

PIC18CXX8

BCF Bit Clear f

Syntax: [*label*] BCF f, b [,a]

Operands: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$

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 = 0xC7

After Instruction
 FLAG_REG = 0x47

BN Branch if Negative

Syntax: [*label*] BN n

Operands: $-128 \leq n \leq 127$

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: $-128 \leq n \leq 127$

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: $-128 \leq n \leq 127$

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)

PIC18CXX8

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: $-1024 \leq n \leq 1023$

Operation: $(PC) + 2 + 2n \rightarrow PC$

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: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$

Operation: $1 \rightarrow f < b >$

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, 1

Before Instruction
FLAG_REG = 0x0A

After Instruction
FLAG_REG = 0x8A

PIC18CXX8

BTFSK Bit Test File, Skip if Clear

Syntax: [*label*] BTFSK f, b [,a]

Operands: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$

Operation: skip if (f) = 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 BTFSK FLAG, 1, ACCESS
 FALSE :
 TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;
 PC = address (TRUE)
 If FLAG<1> = 1;
 PC = address (FALSE)

BTFSK Bit Test File, Skip if Set

Syntax: [*label*] BTFSK f, b [,a]

Operands: $0 \leq f \leq 255$
 $0 \leq b < 7$
 $a \in [0,1]$

Operation: skip if (f) = 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 BTFSK FLAG, 1, ACCESS
 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: $0 \leq f \leq 255$
 $0 \leq b < 7$
 $a \in [0,1]$

Operation: $(\overline{f}) \rightarrow f$

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 [0x75]

After Instruction:
 PORTC = 0110 0101 [0x65]

BOV Branch if Overflow

Syntax: [*label*] BOV n

Operands: $-128 \leq n \leq 127$

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)

PIC18CXX8

BZ Branch if Zero

Syntax: [*label*] BZ n
 Operands: $-128 \leq n \leq 127$
 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: $0 \leq k \leq 1048575$
 $s \in [0,1]$

Operation: (PC) + 4 → TOS,
 k → PC<20:1>,
 if s = 1
 (WREG) → WS,
 (STATUS) → STATUSS,
 (BSR) → BSRS

Status Affected: None

Encoding:

| | | | |
|------|---------------------|--------------------|-------------------|
| 1110 | 110s | k ₇ kkk | kkkk ₀ |
| 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 | 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 = WREG
 BSRS = BSR
 STATUSS = STATUS

CLRF **Clear f**

Syntax: *[label]* CLRF f [,a]

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $000h \rightarrow f$
 $1 \rightarrow Z$

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 = 0x5A

Z = ?

After Instruction

FLAG_REG = 0x00

Z = 0

CLRWDT **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

Operation: $000h \rightarrow$ WDT,
 $000h \rightarrow$ WDT postscaler,
 $1 \rightarrow \overline{TO}$,
 $1 \rightarrow \overline{PD}$

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 = 0x00

WDT postscaler = 0

\overline{TO} = 1

\overline{PD} = 1

PIC18CXX8

COMF **Complement f**

Syntax: [*label*] COMF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
 d ∈ [0,1]
 a ∈ [0,1]

Operation: $\overline{f} \rightarrow \text{dest}$

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 W. 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: COMP REG

Before Instruction

REG = 0x13

N = ?

Z = ?

After Instruction

REG = 0x13

WREG = 0xEC

N = 1

Z = 0

CPFSEQ **Compare f with WREG, skip if f = WREG**

Syntax: [*label*] CPFSEQ f [,a]

Operands: 0 ≤ f ≤ 255
 a ∈ [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 W by performing an unsigned subtraction.
 If 'f' = WREG, then the fetched instruction is discarded and an 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: $0 \leq f \leq 255$
 $a \in [0,1]$

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: $0 \leq f \leq 255$
 $a \in [0,1]$

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 W 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)

PIC18CXX8

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 |

Example 1: DAW

Before Instruction
 WREG = 0xA5
 C = 0
 DC = 0

After Instruction
 WREG = 0x05
 C = 1
 DC = 0

Example 2:

Before Instruction
 WREG = 0xCE
 C = 0
 DC = 0

After Instruction
 WREG = 0x34
 C = 1
 DC = 0

DECF Decrement f

Syntax: [label] DECF f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - 1 \rightarrow \text{dest}$

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 = 0x01
 Z = 0

After Instruction
 CNT = 0x00
 Z = 1

DECFSZ **Decrement f, skip if 0**

Syntax: [*label*] DECFSZ f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - 1 \rightarrow \text{dest}$,
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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - 1 \rightarrow \text{dest}$,
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)
    
```

PIC18CXX8

GOTO Unconditional Branch

Syntax: [*label*] GOTO *k*

Operands: $0 \leq k \leq 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

1st word ($k<7:0>$)

| | | | |
|------|------|-----------|----------|
| 1110 | 1111 | k_7 kkk | $kkkk_0$ |
|------|------|-----------|----------|

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: $0 \leq f \leq 255$

$d \in [0,1]$

$a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$

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 = 0xFF

Z = 0

C = ?

DC = ?

After Instruction

CNT = 0x00

Z = 1

C = 1

DC = 1

INCFSZ **Increment f, skip if 0**

Syntax: [label] INCFSZ f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$,
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 \neq 0;

PC = Address (NZERO)

INFSNZ **Increment f, skip if not 0**

Syntax: [label] INFSNZ f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$,
skip if result \neq 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 \neq 0;

PC = Address (NZERO)

If REG = 0;

PC = Address (ZERO)

PIC18CXX8

IORLW **Inclusive OR literal with WREG**

Syntax: [*label*] IORLW *k*
 Operands: $0 \leq k \leq 255$
 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 W |

Example: IORLW 0x35

Before Instruction
 WREG = 0x9A
 N = ?
 Z = ?

After Instruction
 WREG = 0xBF
 N = 1
 Z = 0

IORWF **Inclusive OR WREG with f**

Syntax: [*label*] IORWF *f* [,d [,a]]
 Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$
 Operation: (WREG) .OR. (*f*) → dest
 Status Affected: N,Z
 Encoding:

| | | | |
|------|------|------|------|
| 0001 | 00da | ffff | ffff |
|------|------|------|------|

 Description: Inclusive OR W 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 = 0x13
 WREG = 0x91
 N = ?
 Z = ?

After Instruction
 RESULT = 0x13
 WREG = 0x93
 N = 1
 Z = 0

LFSR **Load FSR**

Syntax: [*label*] LFSR f,k

Operands: $0 \leq f \leq 2$
 $0 \leq k \leq 4095$

Operation: $k \rightarrow \text{FSRf}$

Status Affected: None

Encoding:

| | | | |
|------|------|----------|-------------|
| 1110 | 1110 | 00ff | $k_{11}kkk$ |
| 1111 | 0000 | k_7kkk | $kkkk$ |

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, 0x3AB

After Instruction

FSR2H = 0x03
 FSR2L = 0xAB

MOVF **Move f**

Syntax: [*label*] MOVF f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $f \rightarrow \text{dest}$

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 W |

Example: MOVF REG, W

Before Instruction

REG = 0x22
 WREG = 0xFF
 N = ?
 Z = ?

After Instruction

REG = 0x22
 WREG = 0x22
 N = 0
 Z = 0

PIC18CXX8

MOVFF Move f to f

Syntax: `[label] MOVFF fs,fd`

Operands: $0 \leq f_s \leq 4095$
 $0 \leq f_d \leq 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

| | | | | |
|--------------------|------|------|------|--------------------|
| Encoding: | | | | |
| 1st word (source) | 1100 | ffff | ffff | fffff _s |
| 2nd word (destin.) | 1111 | ffff | ffff | fffff _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 = 0x33
 REG2 = 0x11

After Instruction

REG1 = 0x33,
 REG2 = 0x33

MOVLB Move literal to low nibble in BSR

Syntax: `[label] MOVLB k`

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow \text{BSR}$

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 0x05`

Before Instruction

BSR register = 0x02

After Instruction

BSR register = 0x05

MOVLW Move literal to WREG

Syntax: [*label*] MOVLW *k*
 Operands: $0 \leq k \leq 255$
 Operation: $k \rightarrow \text{WREG}$
 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 W |

Example: MOVLW 0x5A

After Instruction
 WREG = 0x5A

MOVWF Move WREG to f

Syntax: [*label*] MOVWF *f* [,a]
 Operands: $0 \leq f \leq 255$
 $a \in [0,1]$
 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 = 0x4F
 REG = 0xFF

After Instruction
 WREG = 0x4F
 REG = 0x4F

PIC18CXX8

MULLW Multiply Literal with WREG

Syntax: [*label*] MULLW *k*

Operands: $0 \leq k \leq 255$

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 0xC4

Before Instruction

WREG = 0xE2

PRODH = ?

PRODL = ?

After Instruction

WREG = 0xE2

PRODH = 0xAD

PRODL = 0x08

MULWF Multiply WREG with f

Syntax: [*label*] MULWF *f* [,a]

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

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 = 0xC4

REG = 0xB5

PRODH = ?

PRODL = ?

After Instruction

WREG = 0xC4

REG = 0xB5

PRODH = 0x8A

PRODL = 0x94

NEGF **Negate f**

Syntax: [*label*] **NEGF** f [*a*]

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(\bar{f}) + 1 \rightarrow f$

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' |

NOP **No Operation**

Syntax: [*label*] **NOP**

Operands: None

Operation: No operation

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 0000 | 0000 | 0000 | 0000 |
| 1111 | xxxx | xxxx | xxxx |

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|--------------|--------------|--------------|
| Decode | No operation | No operation | No operation |

Example:

None.

Example: **NEGF** **REG**

Before Instruction

REG = 0011 1010 [0x3A]
N = ?
OV = ?
C = ?
DC = ?
Z = ?

After Instruction

REG = 1100 0110 [0xC6]
N = 1
OV = 0
C = 0
DC = 0
Z = 0

PIC18CXX8

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 | NEW |
| GOTO | |

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: $-1024 \leq n \leq 1023$
 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 MCLR Reset.

Status Affected: All

Encoding:

| | | | |
|------|------|------|------|
| 0000 | 0000 | 1111 | 1111 |
|------|------|------|------|

Description: This instruction provides a way to execute a 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

PIC18CXX8

RETFIE Return from Interrupt

Syntax: [*label*] RETFIE [*s*]

Operands: $s \in [0,1]$

Operation: (TOS) → PC,
 1 → GIE/GIEH or PEIE/GIEL,
 if $s = 1$
 (WS) → W,
 (STATUS) → STATUS,
 (BSRS) → BSR,
 PCLATU, PCLATH are unchanged.

Status Affected: None

Encoding:

| | | | |
|------|------|------|--------------|
| 0000 | 0000 | 0001 | 000 <i>s</i> |
|------|------|------|--------------|

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, STATUS and BSR 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 = BSR
 STATUS = STATUS
 GIE/GIEH, PEIE/GIEL = 1

RETLW Return Literal to WREG

Syntax: [*label*] RETLW *k*

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow W$,
 (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 W |
| 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 = 0x07

After Instruction

WREG = value of kn

RETURN Return from Subroutine

Syntax: [*label*] RETURN [*s*]

Operands: $s \in [0,1]$

Operation: (TOS) → PC,
if $s = 1$
(WS) → W,
(STATUS) → 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, STATUS and BSR 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
WREG = 0x04
STATUS = 0x00
BSR = 0x00

After Instruction
WREG = 0x04
STATUS = 0x00
BSR = 0x00
PC = TOS
    
```

RLCF Rotate Left f through Carry

Syntax: [*label*] RLCF f [, *d* [, *a*]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

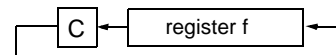
Operation: (f<n>) → dest<n+1>,
(f<7>) → C,
(C) → dest<0>

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
    
```

PIC18CXX8

RLNCF Rotate Left f (no carry)

Syntax: [*label*] RLNCF f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

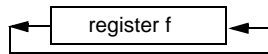
Operation: $(f\langle n \rangle) \rightarrow \text{dest}\langle n+1 \rangle$,
 $(f\langle 7 \rangle) \rightarrow \text{dest}\langle 0 \rangle$

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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

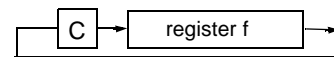
Operation: $(f\langle n \rangle) \rightarrow \text{dest}\langle n-1 \rangle$,
 $(f\langle 0 \rangle) \rightarrow C$,
 $(C) \rightarrow \text{dest}\langle 7 \rangle$

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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

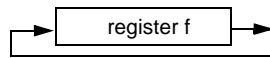
Operation: $(f\langle n \rangle) \rightarrow \text{dest}\langle n-1 \rangle,$
 $(f\langle 0 \rangle) \rightarrow \text{dest}\langle 7 \rangle$

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: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: FFh \rightarrow 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 = 0x5A

After Instruction

REG = 0xFF

PIC18CXX8

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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

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 = 0xFF
 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 ≤ k ≤ 255

Operation: k – (WREG) → 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 W |

Example 1: SUBLW 0x02

Before Instruction
 WREG = 1
 C = ?

After Instruction
 WREG = 1
 C = 1 ; result is positive
 Z = 0
 N = 0

Example 2: SUBLW 0x02

Before Instruction
 WREG = 2
 C = ?

After Instruction
 WREG = 0
 C = 1 ; result is zero
 Z = 1
 N = 0

Example 3: SUBLW 0x02

Before Instruction
 WREG = 3
 C = ?

After Instruction
 WREG = 0xFF ; (2's complement)
 C = 0 ; result is negative
 Z = 0
 N = 1

PIC18CXX8

SUBWF Subtract WREG from f

Syntax: [*label*] SUBWF f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - (WREG) \rightarrow \text{dest}$

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 Subtract WREG from f (cont'd)

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 = 0xFF ;(2's complement)
WREG = 2
C = 0 ; result is negative
Z = 0
N = 1
```

| | | | | | | | | | |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| SUBWFB | Subtract WREG from f with Borrow | | | | | | | | |
| Syntax: | [<i>label</i>] SUBWFB f [,d [,a]] | | | | | | | | |
| Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | | | | | | |
| Operation: | (f) – (WREG) – (\bar{C}) → dest | | | | | | | | |
| Status Affected: | N,OV, C, DC, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px 10px;">0101</td> <td style="padding: 2px 10px;">10da</td> <td style="padding: 2px 10px;">ffff</td> <td style="padding: 2px 10px;">ffff</td> </tr> </table> | 0101 | 10da | ffff | ffff | | | | |
| 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: | | | | | | | | | |
| | <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px 10px;">Q1</td> <td style="padding: 2px 10px;">Q2</td> <td style="padding: 2px 10px;">Q3</td> <td style="padding: 2px 10px;">Q4</td> </tr> <tr> <td style="padding: 2px 10px;">Decode</td> <td style="padding: 2px 10px;">Read register 'f'</td> <td style="padding: 2px 10px;">Process Data</td> <td style="padding: 2px 10px;">Write to destination</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

| | |
|--------------------|--------------------------------------------------|
| SUBWFB | Subtract WREG from f with Borrow (cont'd) |
| <u>Example 1:</u> | SUBWFB REG |
| Before Instruction | |
| REG | = 0x19 (0001 1001) |
| WREG | = 0x0D (0000 1101) |
| C | = 1 |
| After Instruction | |
| REG | = 0x0C (0000 1011) |
| WREG | = 0x0D (0000 1101) |
| C | = 1 |
| Z | = 0 |
| N | = 0 ; result is positive |
| <u>Example 2:</u> | SUBWFB REG, W |
| Before Instruction | |
| REG | = 0x1B (0001 1011) |
| WREG | = 0x1A (0001 1010) |
| C | = 0 |
| After Instruction | |
| REG | = 0x1B (0001 1011) |
| WREG | = 0x00 |
| C | = 1 |
| Z | = 1 ; result is zero |
| N | = 0 |
| <u>Example 3:</u> | SUBWFB REG |
| Before Instruction | |
| REG | = 0x03 (0000 0011) |
| WREG | = 0x0E (0000 1101) |
| C | = 1 |
| After Instruction | |
| REG | = 0xF5 (1111 0100) [2's comp] |
| WREG | = 0x0E (0000 1101) |
| C | = 0 |
| Z | = 0 |
| N | = 1 ; result is negative |

PIC18CXX8

SWAPF Swap nibbles in f

Syntax: [*label*] SWAPF f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

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 = 0x53

After Instruction

REG = 0x35

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 | No operation (Read Program Memory) | No operation | No operation | No operation (Write TABLAT) |

TBLRD **Table Read (cont'd)**

Example 1: TBLRD *+ ;

 Before Instruction

 TABLAT = 0x55

 TBLPTR = 0x00A356

 MEMORY(0x00A356) = 0x34

 After Instruction

 TABLAT = 0x34

 TBLPTR = 0x00A357

Example 2: TBLRD +* ;

 Before Instruction

 TABLAT = 0xAA

 TBLPTR = 0x01A357

 MEMORY(0x01A357) = 0x12

 MEMORY(0x01A358) = 0x34

 After Instruction

 TABLAT = 0x34

 TBLPTR = 0x01A358

PIC18CXX8

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 Table Write (Continued)

Example 1: TBLWT *+;

Before Instruction
 TABLAT = 0x55
 TBLPTR = 0x00A356
 MEMORY(0x00A356) = 0xFF

After Instructions (table write completion)
 TABLAT = 0x55
 TBLPTR = 0x00A357
 MEMORY(0x00A356) = 0x55

Example 2: TBLWT +*;

Before Instruction
 TABLAT = 0x34
 TBLPTR = 0x01389A
 MEMORY(0x01389A) = 0xFF
 MEMORY(0x01389B) = 0xFF

After Instruction (table write completion)
 TABLAT = 0x34
 TBLPTR = 0x01389B
 MEMORY(0x01389A) = 0xFF
 MEMORY(0x01389B) = 0x34

TSTFSZ Test f, skip if 0

Syntax: [*label*] TSTFSZ f [,a]

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

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 = 0x00,
 PC = Address (ZERO)
 If CNT \neq 0x00,
 PC = Address (NZERO)

XORLW Exclusive OR literal with WREG

Syntax: [*label*] XORLW k

Operands: $0 \leq k \leq 255$

Operation: (WREG) .XOR. k \rightarrow 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 0xAF

Before Instruction

WREG = 0xB5
 N = ?
 Z = ?

After Instruction

WREG = 0x1A
 N = 0
 Z = 0

PIC18CXX8

XORWF Exclusive OR WREG with f

Syntax: [*label*] XORWF f [,d [,a]]

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: (WREG) .XOR. (f) \rightarrow 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 = 0xAF
WREG = 0xB5
N = ?
Z = ?

After Instruction

REG = 0x1A
WREG = 0xB5
N = 0
Z = 0

24.0 DEVELOPMENT SUPPORT

The PICmicro® 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™ Linker/MPLIB™ Librarian
- Simulators
 - MPLAB SIM Software Simulator
- Emulators
 - MPLAB ICE 2000 In-Circuit Emulator
 - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
 - MPLAB ICD for PIC16F877
- 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

24.1 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 which contains:

- Multiple functionality
 - editor
 - simulator
 - programmer (sold separately)
 - emulator (sold separately)
- A full featured editor
- A project manager
- Customizable tool bar 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 PICmicro tools (automatically updates all project information)
- Debug using:
 - source files
 - absolute listing file
 - object code

The ability to use MPLAB IDE with Microchip's MPLAB SIM simulator, allows a consistent platform and the ability to easily switch from the cost effective simulator to the full featured emulator with minimal retraining.

24.2 MPASM Assembler

The MPASM assembler is a full featured universal macro assembler for all PICmicro MCU's. It can produce absolute code directly in the form of HEX files for device programmers, or it can generate relocatable objects for the MPLINK object linker.

The MPASM assembler has a command line interface and a Windows shell and can be used as a stand-alone application on a Windows 3.x, or greater, system. The MPASM assembler generates relocatable object files, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file, which contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- MPASM assembler and MPLINK object linker are integrated into MPLAB IDE projects.
- MPASM assembler allows user defined macros to be created for streamlined assembly.
- MPASM assembler allows conditional assembly for multi-purpose source files.
- MPASM assembler directives allow complete control over the assembly process.

24.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI 'C' compilers and integrated development environments 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.

24.4 MPLINK Linker/MPLIB Librarian

The MPLINK object linker is a relocatable linker for the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can link relocatable objects from assembly or C source files, along with 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:

- MPLINK object linker works with MPASM assembler and MPLAB C17 and MPLAB C18 C compilers.
- MPLINK object linker allows all memory areas to be defined as sections to provide link-time flexibility.

The MPLIB object librarian features include:

- MPLIB object librarian makes linking easier because single libraries can be included instead of many smaller files.
- MPLIB object librarian helps keep code maintainable by grouping related modules together.
- MPLIB object librarian commands allow libraries to be created and modules to be added, listed, replaced, deleted or extracted.

24.5 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC host environment by simulating the PICmicro 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.

24.6 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 PICmicro microcontrollers (MCUs). Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, “make” and download and source debugging from a single environment.

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 PICmicro 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 3.x/95/98 environment were chosen to best make these features available to you, the end user.

The MPLAB ICE in-circuit emulator is available in two versions: MPLAB ICE 1000 and MPLAB ICE 2000. The MPLAB ICE 1000 is a basic, low cost emulator system with simple trace capabilities. The MPLAB ICE 2000 is a full featured emulator system with enhanced trace, trigger and data monitoring features. Both systems use the same processor modules and will operate across the full operating speed range of the PICmicro MCU.

24.7 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.

24.8 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 PIC16F877 and can be used to develop this and other PICmicro 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 programming and 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. The MPLAB ICD is also a programmer for the FLASH PIC16F87X family.

24.9 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 VDD and VPP supplies, which allow it to verify programmed memory at VDD min and VDD 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 PICmicro devices. It can also set code-protect bits in this mode.

24.10 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 one of the COM (RS-232) ports. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

The PICSTART Plus development programmer supports all PICmicro 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.

24.11 PICDEM 1 Low Cost PICmicro 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.

24.12 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 I²C™ bus and separate headers for connection to an LCD module and a keypad.

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 an 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 an 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 3.1 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.

PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756, 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 down-loading 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.

KEELOQ Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchip's HCS Secure Data Products. The HCS evaluation kit includes an LCD display to show changing codes, a decoder to decode transmissions and a programming interface to program test transmitters.

TABLE 24-1: DEVELOPMENT TOOLS FROM MICROCHIP

| Development Tools | PIC12CXXX | PIC14000 | PIC16C5X | PIC16C6X | PIC16CXX | PIC16F62X | PIC16C7X | PIC16C7XX | PIC16C8X | PIC16F8XX | PIC16C9XX | PIC17C4X | PIC17C7XX | PIC18CXX2 | 24CXX/ 25CXX/ 93CXX | HC5XX | MCRFXXX | MCP2510 |
|---------------------------------------------------|-----------|----------|----------|----------|----------|-----------|----------|-----------|----------|-----------|-----------|----------|-----------|-----------|---------------------------|-------|---------|---------|
| MPLAB® Integrated Development Environment | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| MPLAB® C17 C Compiler | | | | | | | | | | | | | | | | | | |
| MPLAB® C18 C Compiler | | | | | | | | | | | | | | | | | | |
| MPASM™ Assembler/ MPLINK™ Object Linker | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| MPLAB® ICE In-Circuit Emulator | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| ICEPIC™ In-Circuit Emulator | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | | | |
| MPLAB® ICD In-Circuit Debugger | | | | ✓* | | | ✓* | | | ✓ | | | | | | | | |
| PICSTART® Plus Entry Level Development Programmer | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| PRO MATE® II Universal Device Programmer | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 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.

PIC18CXX8

NOTES:

25.0 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 VSS (except VDD, $\overline{\text{MCLR}}$, and RA4) | -0.3V to (VDD + 0.3V) |
| Voltage on VDD with respect to VSS | -0.3V to +7.5V |
| Voltage on $\overline{\text{MCLR}}$ with respect to VSS (Note 2) | 0V to +13.25V |
| Voltage on RA4 with respect to Vss | 0V to +8.5V |
| Total power dissipation (Note 1) | 1.0W |
| Maximum current out of VSS pin | 300 mA |
| Maximum current into VDD pin | 250 mA |
| Input clamp current, I _{IK} (V _I < 0 or V _I > VDD) | ±20 mA |
| Output clamp current, I _{OK} (V _O < 0 or V _O > VDD) | ±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:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OI} \times I_{OL})$$

- 2:** Voltage spikes below VSS at the $\overline{\text{MCLR}}$ /VPP 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 $\overline{\text{MCLR}}$ /VPP pin, rather than pulling this pin directly to VSS.

† 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.

PIC18CXX8

FIGURE 25-1: PIC18CXX8 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

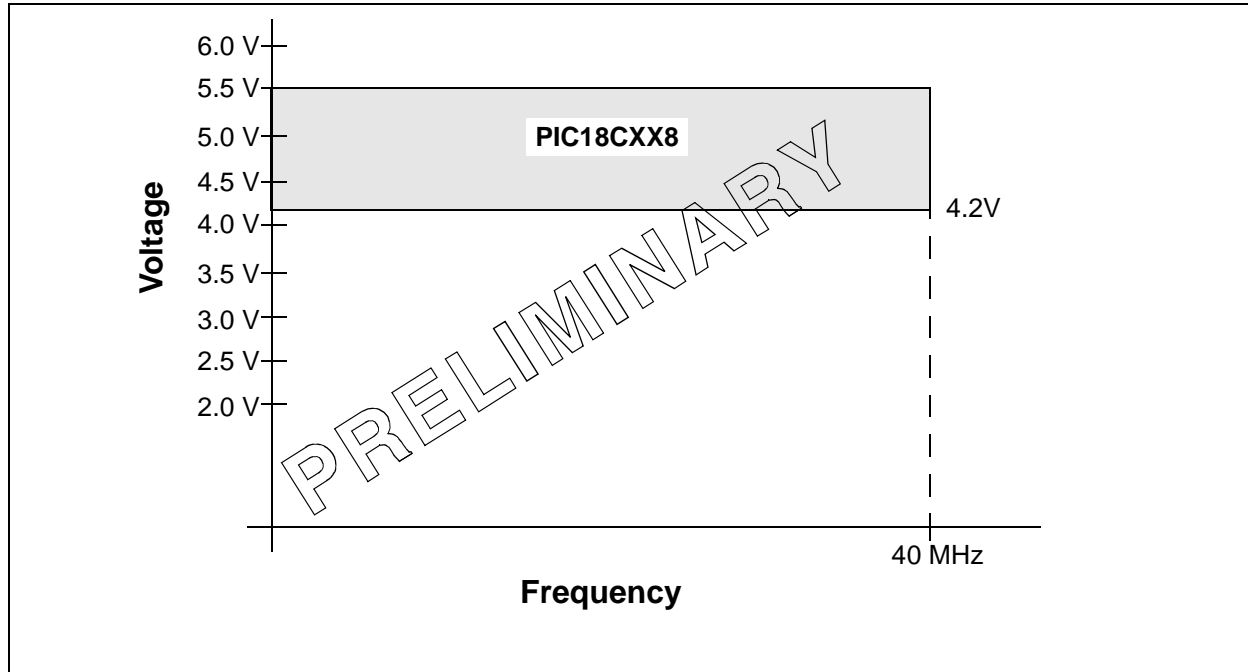
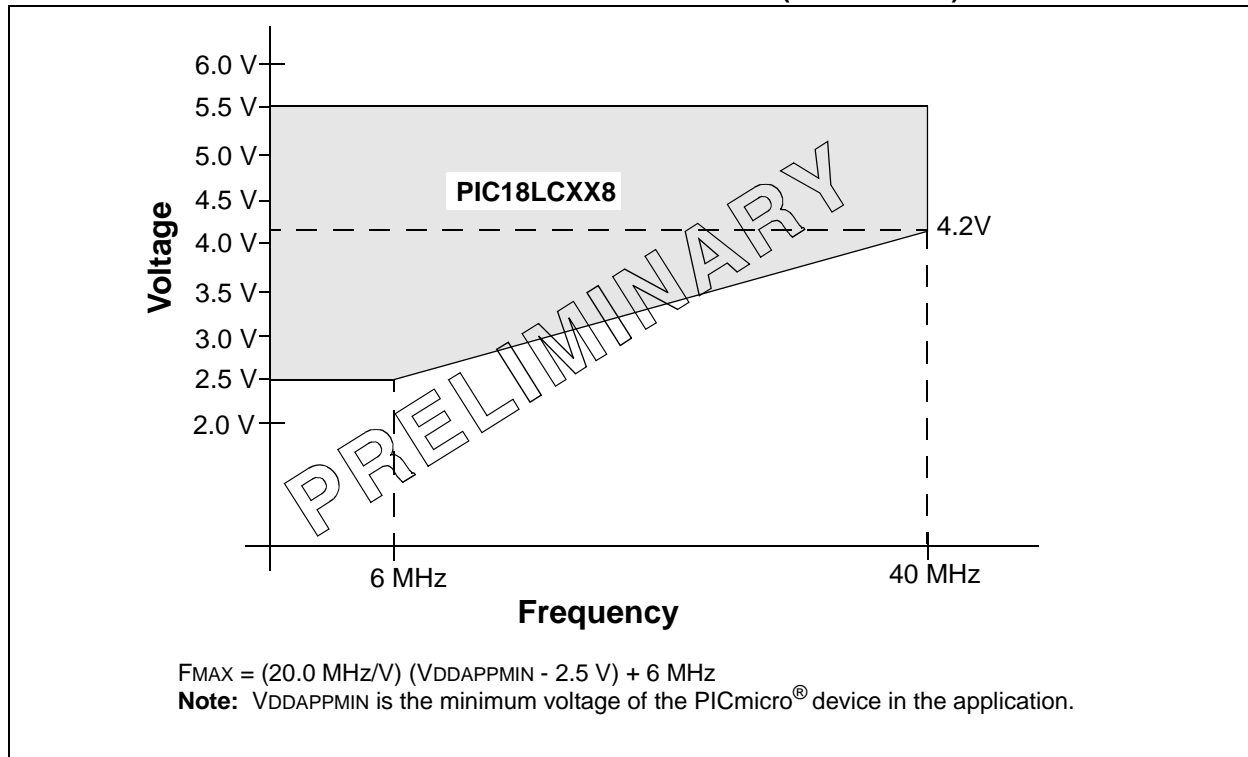


FIGURE 25-2: PIC18LCXX8 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



25.1 DC Characteristics

| PIC18LCXX8 (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial | | | | | | |
|-------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|------|-----|-------|-------------------------------------------|------------------------------------------|
| PIC18CXX8 (Industrial, Extended) | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended | | | | | | |
| Param No. | Symbol | Characteristic/ Device | Min | Typ | Max | Units | Conditions | |
| D001 | VDD | Supply Voltage | | | | | | |
| | | PIC18LCXX8 | 2.5 | — | 5.5 | V | HS, XT, RC and LP osc mode | |
| D001 | | PIC18CXX8 | 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 | |
| D005 | VBOR | Brown-out Reset Voltage | | | | | | |
| | | PIC18LCXX8 | BORV1:BORV0 = 11 | 2.5 | — | 2.66 | V | |
| | | | BORV1:BORV0 = 10 | 2.7 | — | 2.86 | V | |
| | | | BORV1:BORV0 = 01 | 4.2 | — | 4.46 | V | |
| | | | BORV1:BORV0 = 00 | 4.5 | — | 4.78 | V | |
| D005 | | PIC18CXX8 | BORV1:BORV0 = 1x | N.A. | — | N.A. | V | Not in operating voltage range of device |
| | | | BORV1:BORV0 = 01 | 4.2 | — | 4.46 | V | |
| | | | BORV1:BORV0 = 00 | 4.5 | — | 4.78 | V | |

Legend: Rows 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 configuration, current through REXT is not included. The current through the resistor can be estimated by the formula $I_r = V_{DD}/2R_{EXT}$ (mA) with REXT in kOhm.

PIC18CXX8

25.1 DC Characteristics (cont'd)

| PIC18LCXX8 (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial | | | | | |
|-------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----|-----|-----------------------------------------------------|-------------------------------------------------------------|
| PIC18CXX8 (Industrial, Extended) | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended | | | | | |
| Param No. | Symbol | Characteristic/ Device | Min | Typ | Max | Units | Conditions |
| D010 | IDD | Supply Current^(2,4) | | | | | |
| | | PIC18LCXX8 | — | — | 4 | mA | XT, RC, RCIO osc configurations FOSC = 4 MHz, VDD = 2.5V |
| D010 | | PIC18CXX8 | — | — | TBD | mA | XT, RC, RCIO osc configurations FOSC = 4 MHz, VDD = 4.2V |
| D010A | | PIC18LCXX8 | — | — | 48 | μA | LP osc configuration FOSC = 32 kHz, VDD = 2.5V |
| D010A | | PIC18CXX8 | — | — | TBD | μA | LP osc configuration FOSC = 32 kHz, VDD = 4.2V |
| D010C | | PIC18LCXX8 | — | — | 45 | mA | EC, ECIO osc configurations, Fosc = 40 MHz, VDD = 5.5V |
| D010C | | PIC18CXX8 | — | — | 45 | mA | EC, ECIO osc configurations, Fosc = 40 MHz, VDD = 5.5V |
| D013 | | PIC18LCXX8 | — | — | TBD | mA | HS osc configurations Fosc = 6 MHz, VDD = 2.5V |
| | | | — | — | 50 | mA | Fosc = 25 MHz, VDD = 5.5V |
| | | | — | — | 50 | mA | HS + PLL osc configuration Fosc = 10 MHz, VDD = 5.5V |
| D013 | | PIC18CXX8 | — | — | 50 | mA | HS osc configurations Fosc = 25 MHz, VDD = 5.5V |
| | | | — | — | 50 | mA | HS + PLL osc configuration Fosc = 10 MHz, VDD = 5.5V |
| D014 | | PIC18LCXX8 | — | — | 48 | μA | Timer1 osc configuration FOSC = 32 kHz, VDD = 2.5V |
| | | | — | — | TBD | μA | FOSC = 32 kHz, VDD = 2.5V, 25°C |
| D014 | PIC18CXX8 | — | — | TBD | μA | OSCB osc configuration FOSC = 32 kHz, VDD = 4.2V | |
| | | — | — | TBD | μA | FOSC = 32 kHz, VDD = 4.2V, 25°C | |

Legend: Rows 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 and VSS, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, ...).

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula $I_r = V_{DD}/2R_{EXT}$ (mA) with REXT in kOhm.

25.1 DC Characteristics (cont'd)

| PIC18LCXX8 (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial | | | | | |
|-------------------------------------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------------------------------|-----|---------------|----------------------------------------------------------------------|
| PIC18CXX8 (Industrial, Extended) | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended | | | | | |
| Param No. | Symbol | Characteristic/ Device | Min | Typ | Max | Units | Conditions |
| D020 | IPD | Power-down Current⁽³⁾ | | | | | |
| | | PIC18LCXX8 | — | <2.5 | 5 | μA | $V_{DD} = 2.5\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| | | | — | — | 36 | μA | $V_{DD} = 5.5\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| — | — | TBD | μA | $V_{DD} = 2.5\text{V}, 25^{\circ}\text{C}$ | | | |
| D020 | | PIC18CXX8 | — | <1 | TBD | μA | $V_{DD} = 4.2\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| D020A | | | — | — | 36 | μA | $V_{DD} = 5.5\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| D021B | | | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, 25^{\circ}\text{C}$ |
| | | | — | TBD | TBD | μA | $V_{DD} = 4.2\text{V}, -40^{\circ}\text{C to } +125^{\circ}\text{C}$ |
| | | | — | — | 42 | μA | $V_{DD} = 5.5\text{V}, -40^{\circ}\text{C to } +125^{\circ}\text{C}$ |
| D022 | ΔI_{WDT} | Module Differential Current | | | | | |
| D022 | | PIC18LCXX8 Watchdog Timer | — | — | 12 | μA | $V_{DD} = 2.5\text{V}$ |
| | | | — | — | 25 | μA | $V_{DD} = 5.5\text{V}$ |
| | | | — | — | TBD | μA | $V_{DD} = 2.5\text{V}, 25^{\circ}\text{C}$ |
| D022 | | PIC18CXX8 Watchdog Timer | — | — | 25 | μA | $V_{DD} = 5.5\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| | | | — | — | TBD | μA | $V_{DD} = 5.5\text{V}, -40^{\circ}\text{C to } +125^{\circ}\text{C}$ |
| | | | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, 25^{\circ}\text{C}$ |
| D022A | ΔI_{BOR} | PIC18LCXX8 Brown-out Reset | — | — | 50 | μA | $V_{DD} = 5.5\text{V}$ |
| D022A | | | — | — | TBD | μA | $V_{DD} = 2.5\text{V}, 25^{\circ}\text{C}$ |
| D022A | | PIC18CXX8 Brown-out Reset | — | — | 50 | μA | $V_{DD} = 5.5\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| D022B | | | — | — | TBD | μA | $V_{DD} = 5.5\text{V}, -40^{\circ}\text{C to } +125^{\circ}\text{C}$ |
| D022B | | | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, 25^{\circ}\text{C}$ |
| D022B | ΔI_{LVD} | PIC18LCXX8 Low Voltage Detect | — | — | 50 | μA | $V_{DD} = 2.5\text{V}$ |
| D022B | | | — | — | TBD | μA | $V_{DD} = 2.5\text{V}, 25^{\circ}\text{C}$ |
| D022B | | PIC18CXX8 Low Voltage Detect | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| D025 | | | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, -40^{\circ}\text{C to } +125^{\circ}\text{C}$ |
| D025 | | | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, 25^{\circ}\text{C}$ |
| D025 | ΔI_{OSCB} | PIC18LCXX8 Timer1 Oscillator | — | — | 3 | μA | $V_{DD} = 2.5\text{V}$ |
| D025 | | | — | — | TBD | μA | $V_{DD} = 2.5\text{V}, 25^{\circ}\text{C}$ |
| D025 | | PIC18CXX8 Timer1 Oscillator | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, -40^{\circ}\text{C to } +85^{\circ}\text{C}$ |
| | | | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, -40^{\circ}\text{C to } +125^{\circ}\text{C}$ |
| | | | — | — | TBD | μA | $V_{DD} = 4.2\text{V}, 25^{\circ}\text{C}$ |

Legend: Rows are shaded for improved readability.

Note 1: This is the limit to which V_{DD} 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 I_{DD} measurements in active operation mode are:

$OSC1$ = external square wave, from rail to rail; all I/O pins tri-stated, pulled to V_{DD}

$MCLR$ = V_{DD} ; 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 V_{DD} and V_{SS} , and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, ...).
- 4:** For RC osc configuration, current through R_{EXT} is not included. The current through the resistor can be estimated by the formula $I_r = V_{DD}/2R_{EXT}$ (mA) with R_{EXT} in kOhm.

PIC18CXX8

25.2 DC Characteristics: PIC18CXX8 (Industrial, Extended) and PIC18LCXX8 (Industrial)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended | | | |
|--------------------|-----------|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|---------------|------------------------------------------------------------|
| Param No. | Symbol | Characteristic/ Device | Min | Max | Units | Conditions |
| | V_{IL} | Input Low Voltage | | | | |
| D030 | V_{IL} | I/O ports: with TTL buffer | V_{SS} | $0.15V_{DD}$ | V | $V_{DD} < 4.5\text{V}$ |
| D030A | | | — | 0.8 | V | $4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$ |
| D031 | | with Schmitt Trigger buffer RC3 and RC4 | V_{SS} | $0.2V_{DD}$ | V | |
| D032 | | | V_{SS} | $0.3V_{DD}$ | V | |
| D032A | | MCLR | V_{SS} | $0.2V_{DD}$ | V | |
| D032A | | OSC1 (in XT, HS and LP modes) and T1OSI | V_{SS} | $0.3V_{DD}$ | V | |
| D033 | | OSC1 (in RC mode) ⁽¹⁾ | V_{SS} | $0.2V_{DD}$ | V | |
| | V_{IH} | Input High Voltage | | | | |
| D040 | V_{IH} | I/O ports: with TTL buffer | $0.25V_{DD} + 0.8\text{V}$ | V_{DD} | V | $V_{DD} < 4.5\text{V}$ |
| D040A | | | 2.0 | V_{DD} | V | $4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$ |
| D041 | | with Schmitt Trigger buffer RC3 and RC4 | $0.8V_{DD}$ | V_{DD} | V | |
| D042 | | | $0.7V_{DD}$ | V_{DD} | V | |
| D042 | | MCLR | $0.8V_{DD}$ | V_{DD} | V | |
| D042A | | OSC1 (in XT, HS and LP modes) and T1OSI | $0.7V_{DD}$ | V_{DD} | V | |
| D043 | | OSC1 (RC mode) ⁽¹⁾ | $0.9V_{DD}$ | V_{DD} | V | |
| | V_{HYS} | Hysteresis of Schmitt Trigger Inputs | | | | |
| D050 | | | TBD | TBD | V | |
| | I_{IL} | Input Leakage Current^(2,3) | | | | |
| D060 | I_{IL} | I/O ports | — | ± 1 | μA | $V_{SS} \leq V_{PIN} \leq V_{DD}$, Pin at hi-impedance |
| D061 | | MCLR | — | ± 5 | μA | $V_{SS} \leq V_{PIN} \leq V_{DD}$ |
| D063 | | OSC1 | — | ± 5 | μA | $V_{SS} \leq V_{PIN} \leq V_{DD}$ |
| | IPU | Weak Pull-up Current | | | | |
| D070 | IPURB | PORTB weak pull-up current | 50 | 400 | μA | $V_{DD} = 5\text{V}$, $V_{PIN} = V_{SS}$ |

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PICmicro device be driven with an external clock while in RC mode.

2: The leakage current on the 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.

25.2 DC Characteristics: PIC18CXX8 (Industrial, Extended) and PIC18LCXX8 (Industrial) (cont'd)

| 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 |
| Output Low Voltage | | | | | | |
| D080 | VOL | 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 |
| Output High Voltage⁽³⁾ | | | | | | |
| D090 | VOH | 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 |
| Open-drain High Voltage | | | | | | |
| D150 | VOD | | — | 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 ² C mode |

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PICmicro device be driven with an external clock while in RC mode.

2: The leakage current on the 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.

PIC18CXX8

FIGURE 25-3: LOW VOLTAGE DETECT CHARACTERISTICS

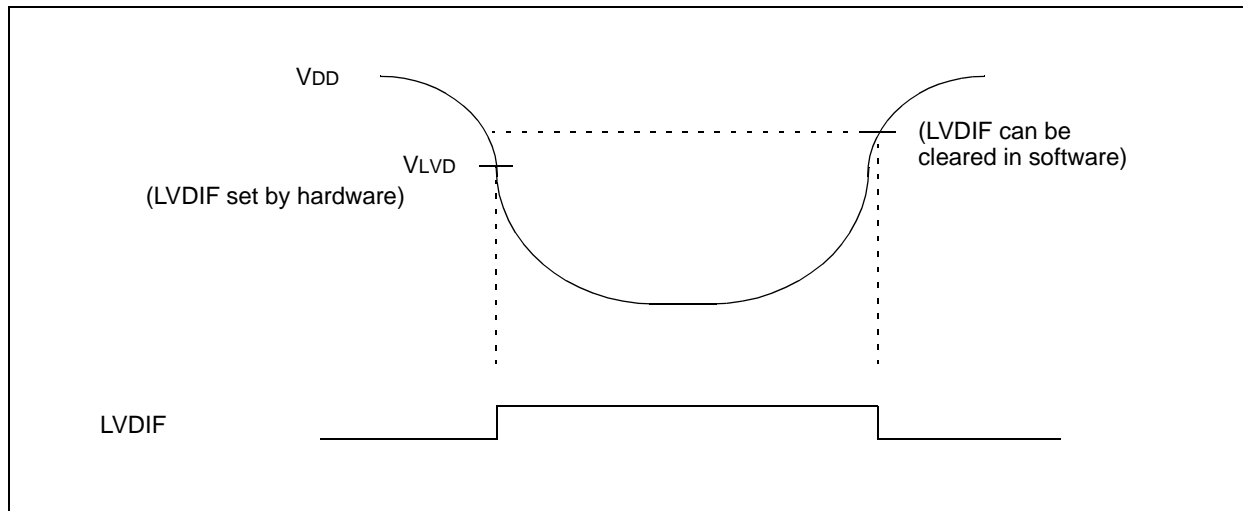


TABLE 25-1: LOW VOLTAGE DETECT CHARACTERISTICS

| | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for Industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended | | | | | |
|-----------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|-----|------|-------|------------|
| Param No. | Symbol | Characteristic/ | | Min | Max | Units | Conditions |
| D420 | VLVD | LVD Voltage | LVDL<3:0> = 0100 | 2.5 | 2.66 | V | |
| | | | LVDL<3:0> = 0101 | 2.7 | 2.86 | V | |
| | | | LVDL<3:0> = 0110 | 2.8 | 2.98 | V | |
| | | | LVDL<3:0> = 0111 | 3.0 | 3.2 | V | |
| | | | LVDL<3:0> = 1000 | 3.3 | 3.52 | V | |
| | | | LVDL<3:0> = 1001 | 3.5 | 3.72 | V | |
| | | | LVDL<3:0> = 1010 | 3.6 | 3.84 | V | |
| | | | LVDL<3:0> = 1011 | 3.8 | 4.04 | V | |
| | | | LVDL<3:0> = 1100 | 4.0 | 4.26 | V | |
| | | | LVDL<3:0> = 1101 | 4.2 | 4.46 | V | |
| | | | LVDL<3:0> = 1110 | 4.5 | 4.78 | V | |

TABLE 25-2: EPROM PROGRAMMING REQUIREMENTS

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +40^{\circ}\text{C}$ | | | |
|-----------------------------------------------------------|--------|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|-------|---------------|-------------------------------------------------------|
| Param. No. | Sym | Characteristic | Min | Max | Units | Conditions |
| Internal Program Memory Programming Specs (Note 1) | | | | | | |
| D110 | VPP | Voltage on MCLR/VPP pin | 12.75 | 13.25 | V | (Note 2) |
| D111 | VDDP | Supply voltage during programming | 4.75 | 5.25 | V | |
| D112 | IPP | Current into MCLR/VPP pin | — | 50 | mA | |
| D113 | IDDP | Supply current during programming | — | 30 | mA | |
| D114 | TPROG | Programming pulse width | 100 | 1000 | μs | Terminated via internal/external interrupt or a RESET |
| D115 | TERASE | EPROM erase time | | | | |
| | | Device operation $\leq 3\text{V}$ | 4 | — | hrs | |
| | | Device operation $\geq 3\text{V}$ | TBD | — | hrs | |

Note 1: These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC18CXX8 Programming Specifications (Literature number DS39028).

2: The MCLR/VPP pin may be kept in this range at times other than programming, but is not recommended.

PIC18CXX8

25.3 AC (Timing) Characteristics

25.3.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

- | | | |
|-------------|-----------|----------------------------------------|
| 1. TppS2ppS | 3. Tcc:ST | (I ² C specifications only) |
| 2. TppS | 4. Ts | (I ² C 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 |
| I²C only | | | |
| AA | output access | High | High |
| BUF | Bus free | Low | Low |

Tcc:ST (I²C specifications only)

| | | | |
|-----------|-----------------|-----|----------------|
| CC | | | |
| HD | Hold | SU | Setup |
| ST | | | |
| DAT | DATA input hold | STO | STOP condition |
| STA | START condition | | |

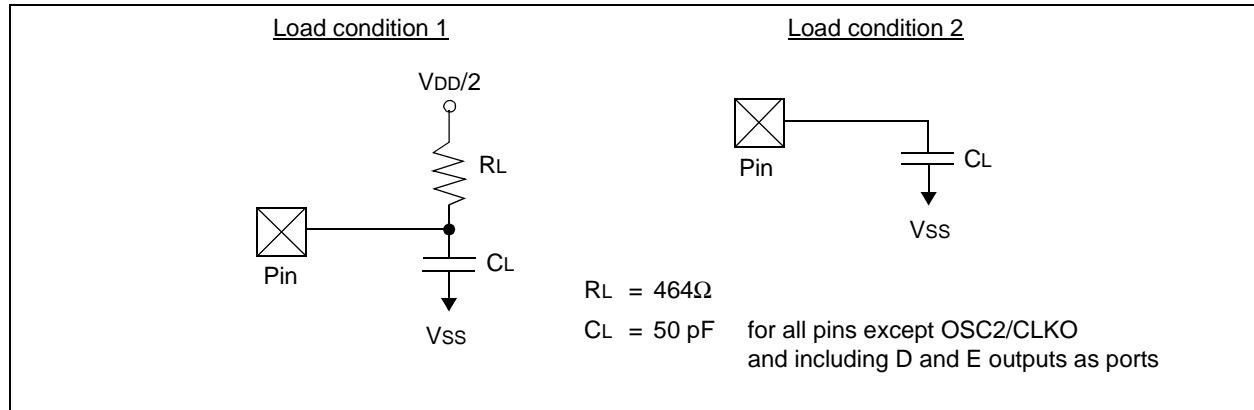
25.3.2 TIMING CONDITIONS

The temperature and voltages specified in Table 25-3 apply to all timing specifications, unless otherwise noted. Figure 25-4 specifies the load conditions for the timing specifications.

TABLE 25-3: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC

| | |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AC CHARACTERISTICS | Standard Operating Conditions (unless otherwise stated) |
| | Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended Operating voltage V_{DD} range as described in DC spec Section 25.1. LC parts operate for industrial temperatures only. |

FIGURE 25-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



PIC18CXX8

25.3.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 25-5: EXTERNAL CLOCK TIMING

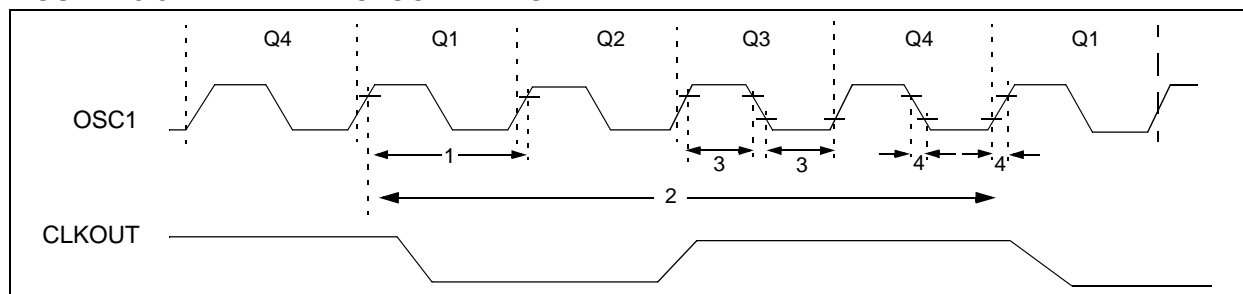


TABLE 25-4: EXTERNAL CLOCK TIMING REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|------------|-------------------------------------|-----------------------------------------|--------|-----|--------------|---------------|
| 1A | Fosc | External CLKIN Frequency ⁽¹⁾ | DC | 40 | MHz | XT osc |
| | | | DC | 40 | MHz | HS osc |
| | | | 4 | 10 | MHz | HS + PLL osc |
| | | | DC | 40 | kHz | LP osc |
| | | | DC | 40 | MHz | EC |
| | Oscillator Frequency ⁽¹⁾ | DC | 4 | MHz | RC osc | |
| | | 0.1 | 4 | MHz | XT osc | |
| | | 4 | 25 | MHz | HS osc | |
| | | 4 | 10 | MHz | HS + PLL osc | |
| | | 5 | 200 | kHz | LP osc mode | |
| 1 | Tosc | External CLKIN Period ⁽¹⁾ | 250 | — | ns | XT and RC osc |
| | | | 40 | — | ns | HS osc |
| | | | 100 | — | ns | HS + PLL osc |
| | | | 5 | — | μs | LP osc |
| | | | 5 | — | ns | EC |
| | Oscillator Period ⁽¹⁾ | 250 | — | ns | RC osc | |
| | | 250 | 10,000 | ns | XT osc | |
| | | 100 | 10,000 | ns | HS osc | |
| | | 40 | 100 | ns | HS + PLL osc | |
| | | 5 | — | μs | LP osc | |
| 2 | Tcy | Instruction Cycle Time ⁽¹⁾ | 100 | — | ns | Tcy = 4/Fosc |
| 3 | TosL, TosH | External Clock in (OSC1) | 30 | — | ns | XT osc |
| | | High or Low Time | 2.5 | — | ns | LP osc |
| | | | 10 | — | μs | HS osc |
| 4 | TosR, TosF | External Clock in (OSC1) | — | 20 | ns | XT osc |
| | | Rise or Fall Time | — | 50 | ns | LP osc |
| | | | — | 7.5 | ns | HS osc |

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 25-5: PLL CLOCK TIMING SPECIFICATION (V_{DD} = 4.2V - 5.5V)

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|-----------|------------------|-------------------------------------|-----|-----|-------|------------|
| 7 | T _{PLL} | PLL Start-up Time (Lock Time) | — | 2 | ms | |
| | ΔCLK | CLKOUT Stability (Jitter) using PLL | -2 | +2 | % | |

PRELIMINARY

PIC18CXX8

FIGURE 25-6: CLKOUT AND I/O TIMING

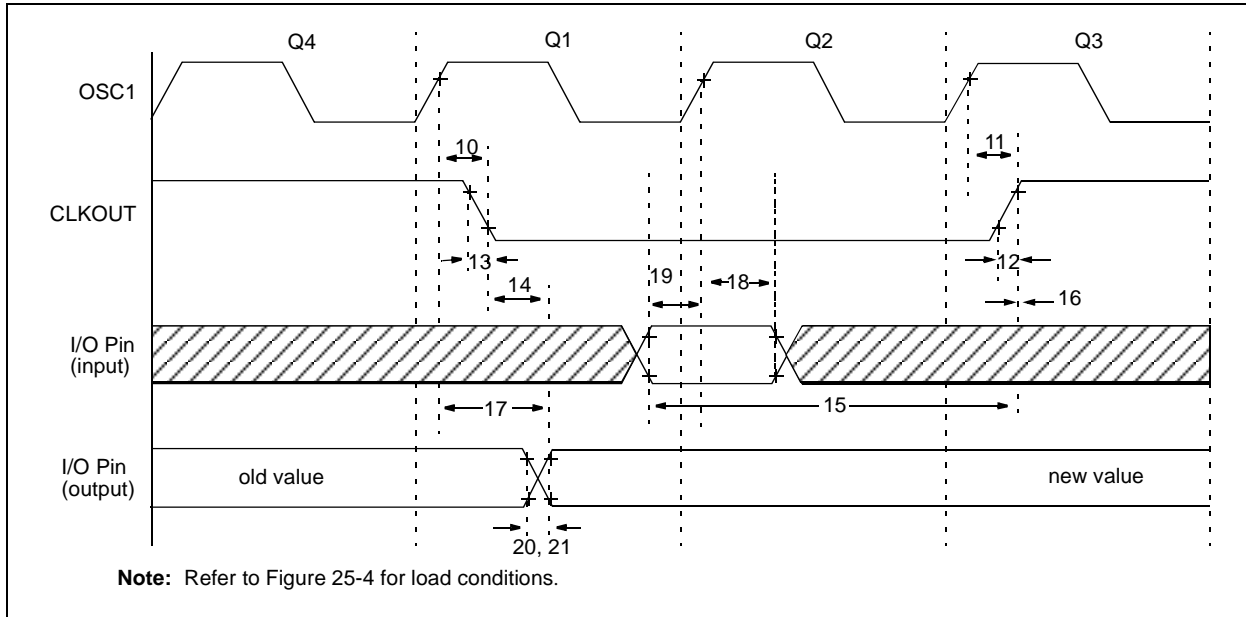


TABLE 25-6: CLKOUT AND I/O TIMING REQUIREMENTS

| 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 | — | — | 0.5T _{CY} + 20 | ns | (1) | |
| 15 | TioV2ckH | Port in valid before CLKOUT ↑ | 0.25T _{CY} + 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 | 100 | — | — | ns | | |
| 18A | | (I/O in hold time) | 200 | — | — | ns | | |
| 19 | TioV2osH | Port input valid to OSC1↑ (I/O in setup time) | 0 | — | — | ns | | |
| 20 | TioR | Port output rise time | PIC18CXX8 | — | 10 | 25 | ns | |
| 20A | | | PIC18LCXX8 | — | — | 60 | ns | |
| 21 | TioF | Port output fall time | PIC18CXX8 | — | 10 | 25 | ns | |
| 21A | | | PIC18LCXX8 | — | — | 60 | ns | |
| 22†† | TINP | INT pin high or low time | T _{CY} | — | — | ns | | |
| 23†† | TRBP | RB7:RB4 change INT high or low time | T _{CY} | — | — | ns | | |
| 24†† | TRCP | RC7:RC4 change INT high or low time | 20 | — | — | ns | | |

†† These parameters are asynchronous events, not related to any internal clock edges.

Note 1: Measurements are taken in RC mode where CLKOUT pin output is 4 x T_{OSC}.

FIGURE 25-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

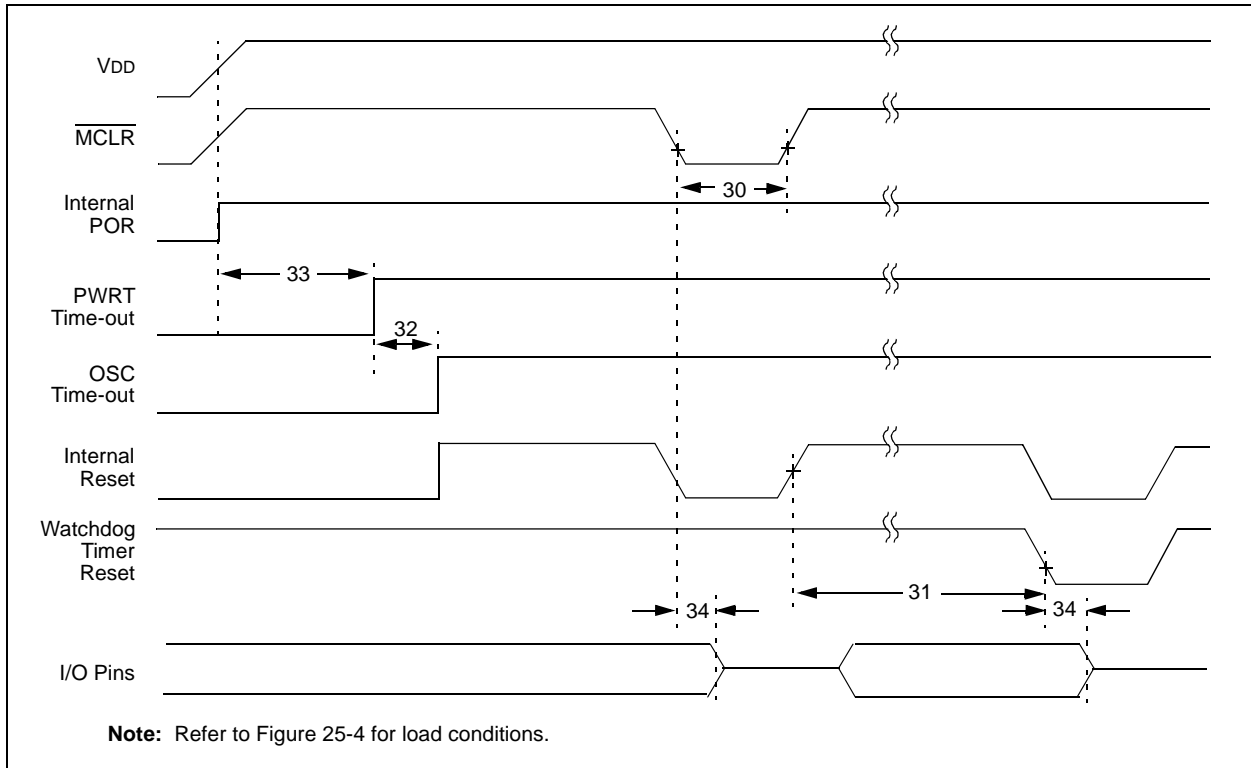


FIGURE 25-8: BROWN-OUT RESET TIMING

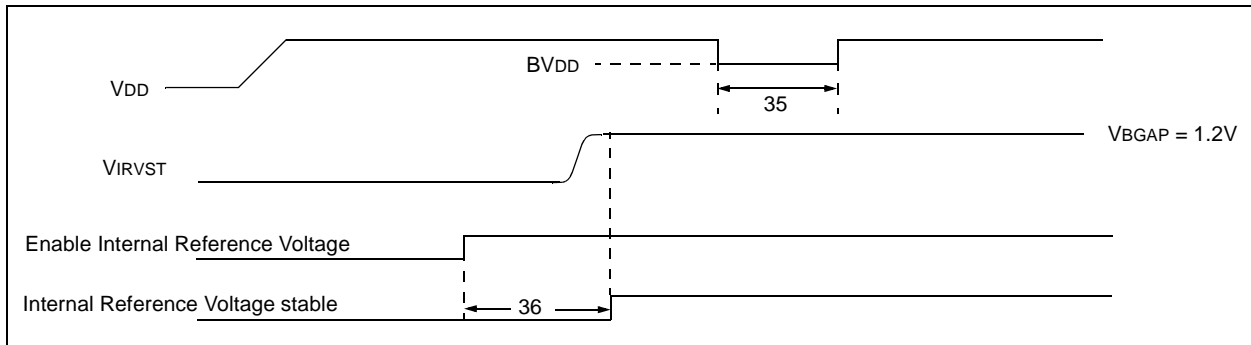


TABLE 25-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|------------|--------|--------------------------------------------------------|----------|-----|----------|-------|-----------------------|
| 30 | TmCL | 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 | — | 1024Tosc | — | Tosc = OSC1 period |
| 33 | TPWRT | Power up Timer Period | 28 | 72 | 132 | ms | |
| 34 | TIOZ | I/O Hi-impedance from MCLR Low or Watchdog Timer Reset | — | 2 | — | μs | |
| 35 | TBOR | Brown-out Reset Pulse Width | 200 | — | — | μs | VDD ≤ BVDD (See D005) |
| 36 | TIVRST | Time for Internal Reference Voltage to become stable | — | 20 | 50 | μs | |

PIC18CXX8

FIGURE 25-9: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

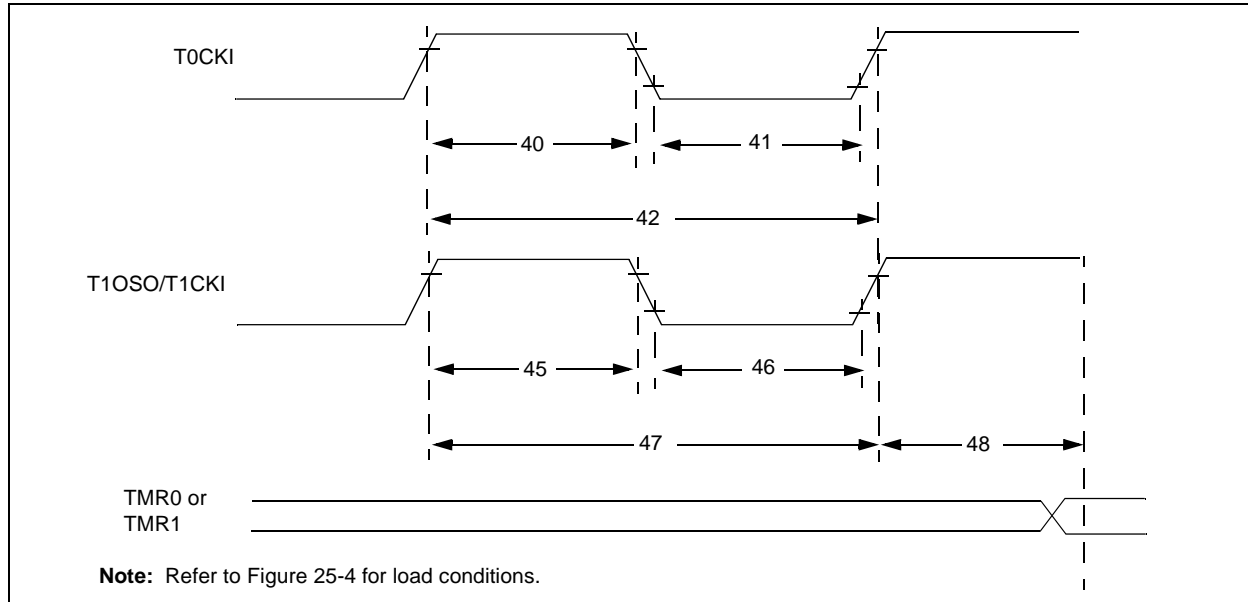


TABLE 25-8: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions | |
|------------|-----------|---------------------------------------------------------|-----------------------------|----------------------------------------------------------|------------|-------|---------------------------------|----|
| 40 | Tt0H | T0CKI High Pulse Width | No Prescaler | $0.5T_{CY} + 20$ | — | ns | | |
| | | | With Prescaler | 10 | — | ns | | |
| 41 | Tt0L | T0CKI Low Pulse Width | No Prescaler | $0.5T_{CY} + 20$ | — | ns | | |
| | | | With Prescaler | 10 | — | ns | | |
| 42 | Tt0P | T0CKI Period | No Prescaler | $T_{CY} + 10$ | — | ns | | |
| | | | With Prescaler | Greater of: $20 \text{ ns or } \frac{T_{CY} + 40}{N}$ | — | ns | | |
| 45 | Tt1H | T1CKI High Time | Synchronous, no prescaler | $0.5T_{CY} + 20$ | — | ns | | |
| | | | Synchronous, with prescaler | PIC18CXX8 | 10 | — | | ns |
| | | | | PIC18LCXX8 | 25 | — | | ns |
| | | | Asynchronous | PIC18CXX8 | 30 | — | | ns |
| PIC18LCXX8 | 50 | — | | ns | | | | |
| 46 | Tt1L | T1CKI Low Time | Synchronous, no prescaler | $0.5T_{CY} + 5$ | — | ns | | |
| | | | Synchronous, with prescaler | PIC18CXX8 | 10 | — | | ns |
| | | | | PIC18LCXX8 | 25 | — | | ns |
| | | | Asynchronous | PIC18CXX8 | 30 | — | | ns |
| PIC18LCXX8 | TBD | TBD | | ns | | | | |
| 47 | Tt1P | T1CKI Input Period | Synchronous | Greater of: $20 \text{ ns or } \frac{T_{CY} + 40}{N}$ | — | ns | N = prescale value (1, 2, 4, 8) | |
| | | | Asynchronous | 60 | — | ns | | |
| | Ft1 | T1CKI oscillator input frequency range | | DC | 50 | kHz | | |
| 48 | Tcke2tmr1 | Delay from external T1CKI clock edge to timer increment | | $2T_{osc}$ | $7T_{osc}$ | — | | |

FIGURE 25-10: CAPTURE/COMPARE/PWM TIMINGS (CCP1 AND CCP2)

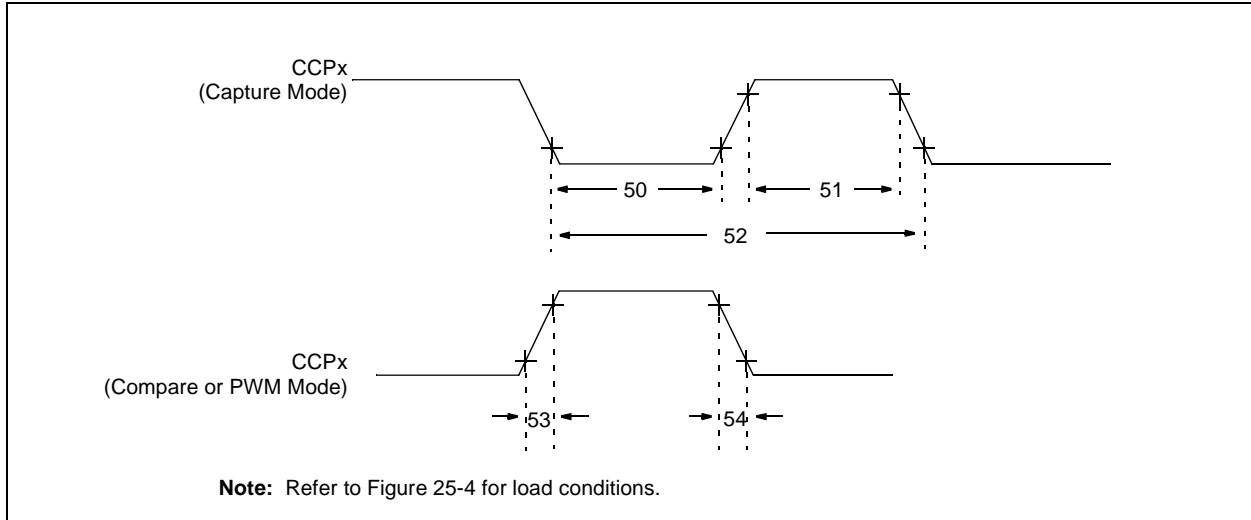


TABLE 25-9: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP1 AND CCP2)

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|------------|--------|-----------------------|----------------|--------------------------|----------|--------|--------------------------------|
| 50 | TccL | CCPx input low time | No Prescaler | $0.5T_{CY} + 20$ | — | ns | |
| | | | With Prescaler | PIC18CXX8 PIC18LCXX8 | 10 20 | — — | |
| 51 | TccH | CCPx input high time | No Prescaler | $0.5T_{CY} + 20$ | — | ns | |
| | | | With Prescaler | PIC18CXX8 PIC18LCXX8 | 10 20 | — — | |
| 52 | TccP | CCPx input period | | $\frac{3T_{CY} + 40}{N}$ | — | ns | N = prescale value (1,4 or 16) |
| 53 | TccR | CCPx output fall time | PIC18CXX8 | — | 25 | ns | |
| | | | PIC18LCXX8 | — | 45 | ns | |
| 54 | TccF | CCPx output fall time | PIC18CXX8 | — | 25 | ns | |
| | | | PIC18LCXX8 | — | 45 | ns | |

PIC18CXX8

FIGURE 25-11: PARALLEL SLAVE PORT TIMING (PIC18C658 AND PIC18C858)

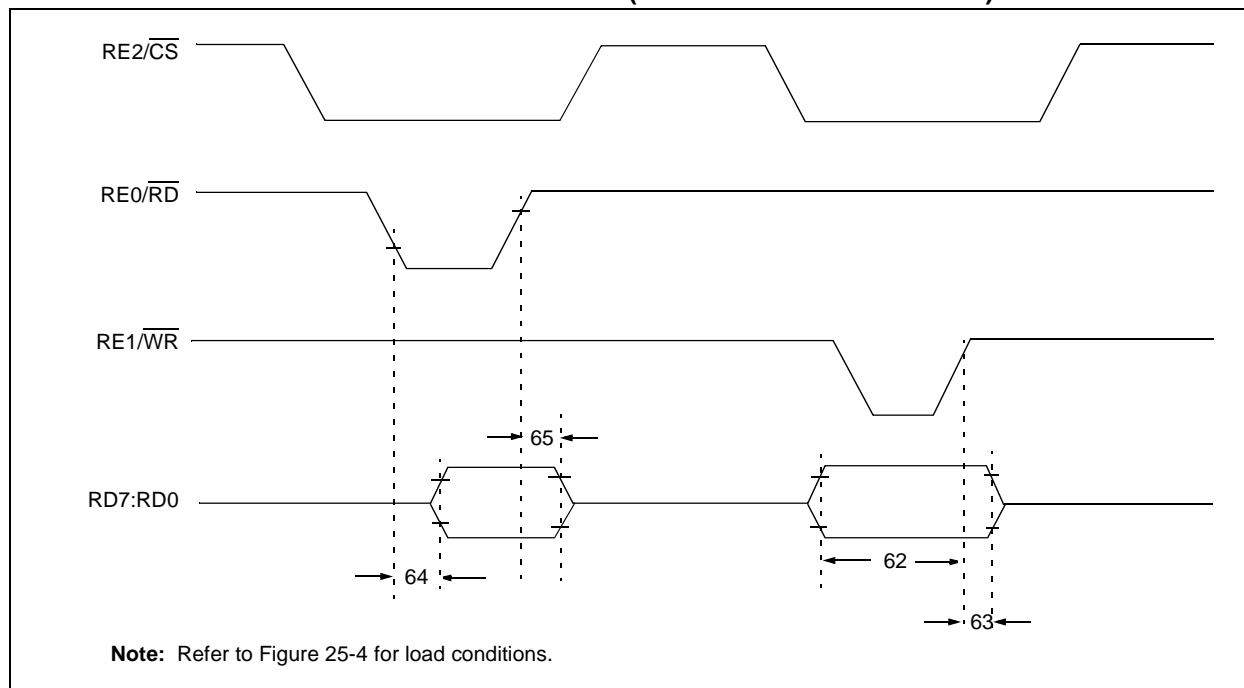


TABLE 25-10: PARALLEL SLAVE PORT REQUIREMENTS (PIC18C658 AND PIC18C858)

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions | |
|------------|----------|--------------------------------------------------------|------------|------|-------|---------------------|--|
| 62 | TdtV2wrH | Data-in valid before WR↑ or CS↑ (setup time) | 20 | — | ns | Extended Temp range | |
| | | | 25 | — | ns | | |
| 63 | TwrH2dtl | WR↑ or CS↑ to data-in invalid (hold time) | PIC18CXX8 | 20 | — | ns | |
| | | | PIC18LCXX8 | 35 | — | ns | |
| 64 | TrdL2dtV | RD↓ and CS↓ to data-out valid | — | 80 | ns | Extended Temp range | |
| | | | — | 90 | ns | | |
| 65 | TrdH2dtl | RD↑ or CS↓ to data-out invalid | 10 | 30 | ns | | |
| 66 | TibfINH | Inhibit the IBF flag bit being cleared from WR↑ or CS↑ | — | 3TCY | ns | | |

FIGURE 25-12: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

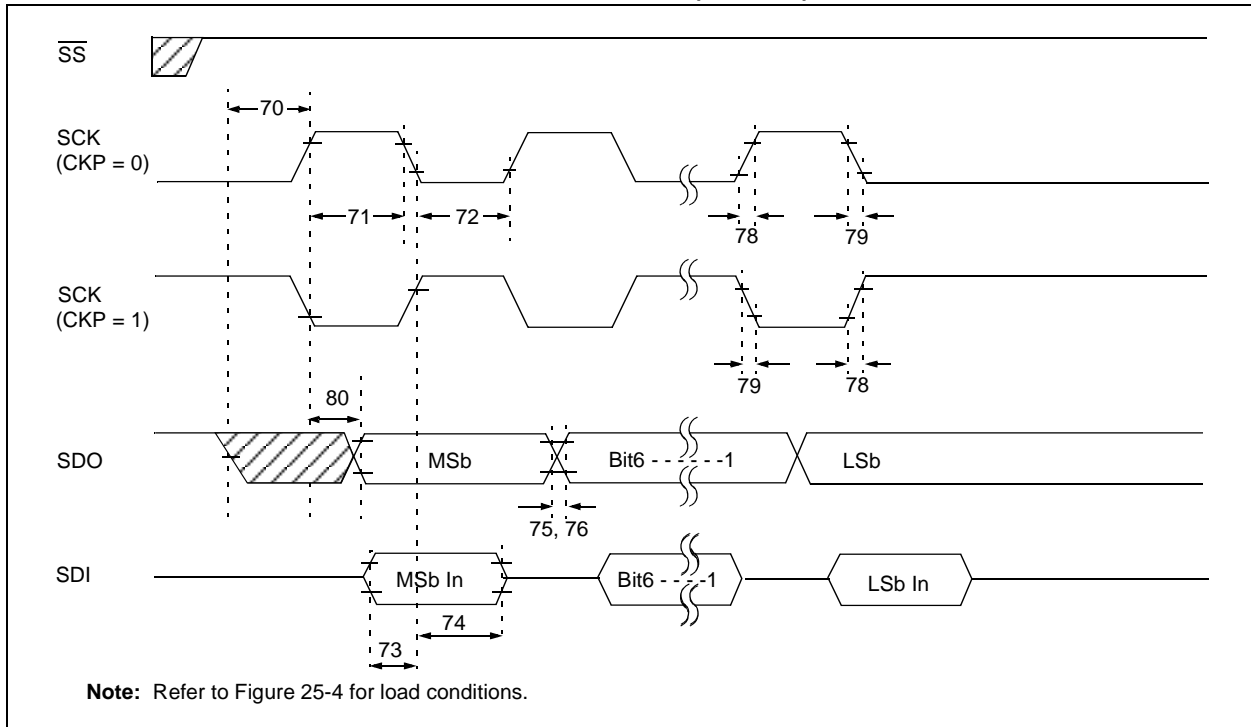


TABLE 25-11: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|------------|-----------------------|---------------------------------------------------------|-------------------------|--------------------------|-------|------------|
| 70 | TssL2scH, TssL2scL | SS↓ to SCK↓ or SCK↑ input | T _{CY} | — | ns | |
| 71 | TscH | SCK input high time (Slave mode) | Continuous | 1.25T _{CY} + 30 | — | ns |
| 71A | | | Single Byte | 40 | — | ns |
| 72 | TscL | SCK input low time (Slave mode) | Continuous | 1.25T _{CY} + 30 | — | ns |
| 72A | | | Single Byte | 40 | — | ns |
| 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 | 1.5T _{CY} + 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 | PIC18CXX8 | — | 25 | ns |
| | | | PIC18LCXX8 | — | 45 | ns |
| 76 | TdoF | SDO data output fall time | — | 25 | ns | |
| 78 | TscR | SCK output rise time (Master mode) | PIC18CXX8 | — | 25 | ns |
| | | | PIC18LCXX8 | — | 45 | ns |
| 79 | TscF | SCK output fall time (Master mode) | — | 25 | ns | |
| 80 | Tsch2doV, TscL2doV | SDO data output valid after SCK edge | PIC18CXX8 | — | 50 | ns |
| | | | PIC18LCXX8 | — | 100 | ns |

Note 1: Requires the use of parameter # 73A.

2: Only if parameter #s 71A and 72A are used.

PIC18CXX8

FIGURE 25-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)

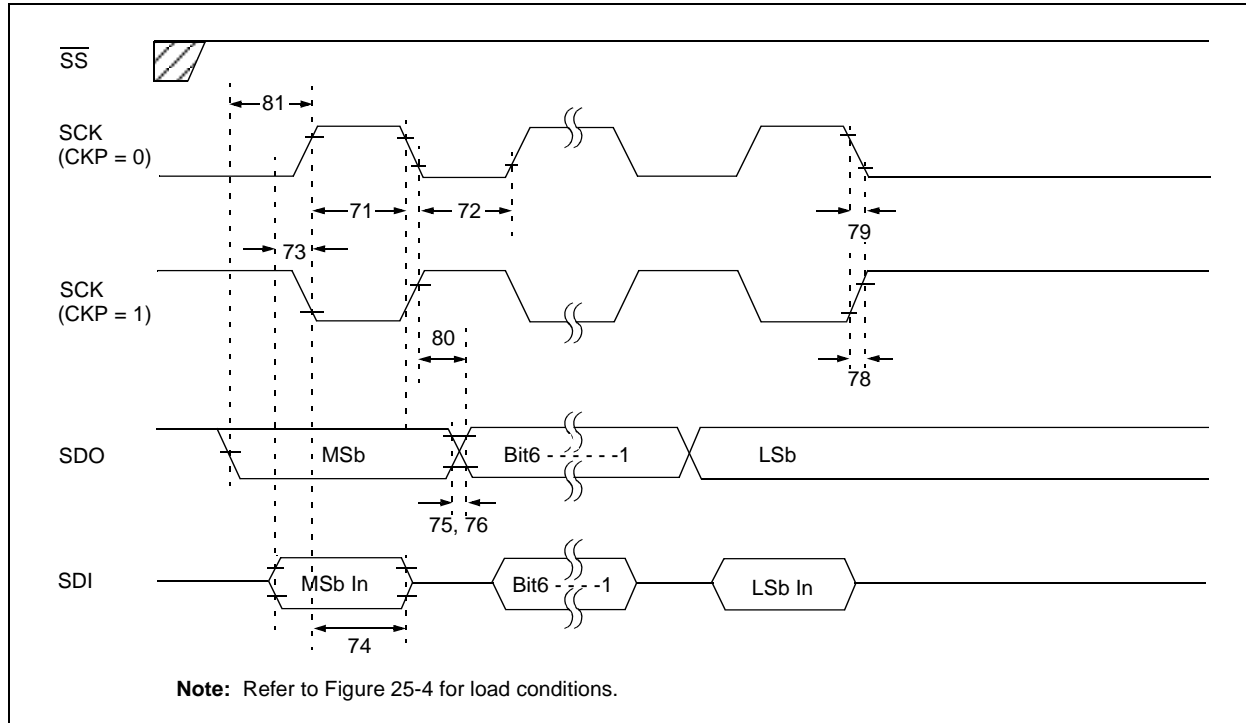


TABLE 25-12: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|------------|--------------------|---------------------------------------------------------|------------------|-------------------|-------|-------------|
| 71 | TscH | SCK input high time (Slave mode) | Continuous | $1.25T_{CY} + 30$ | — | ns |
| 71A | | | Single Byte | 40 | — | ns (Note 1) |
| 72 | TscL | SCK input low time (Slave mode) | Continuous | $1.25T_{CY} + 30$ | — | 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 | $1.5T_{CY} + 40$ | — | ns | (Note 2) |
| 74 | Tsch2diL, TscL2diL | Hold time of SDI data input to SCK edge | 100 | — | ns | |
| 75 | TdoR | SDO data output rise time | PIC18CXX8 — | 25 | ns | |
| 76 | TdoF | SDO data output fall time | — | 25 | ns | |
| 78 | TscR | SCK output rise time (Master mode) | PIC18CXX8 | — | 25 | ns |
| | | | PIC18LCXX8 | — | 45 | ns |
| 79 | TscF | SCK output fall time (Master mode) | — | 25 | ns | |
| 80 | Tsch2doV, TscL2doV | SDO data output valid after SCK edge | PIC18CXX8 | — | 50 | ns |
| | | | PIC18LCXX8 | — | 100 | ns |
| 81 | TdoV2sch, TdoV2scL | SDO data output setup to SCK edge | T_{CY} | — | ns | |

Note 1: Requires the use of parameter # 73A.

Note 2: Only if parameter #s 71A and 72A are used.

FIGURE 25-14: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)

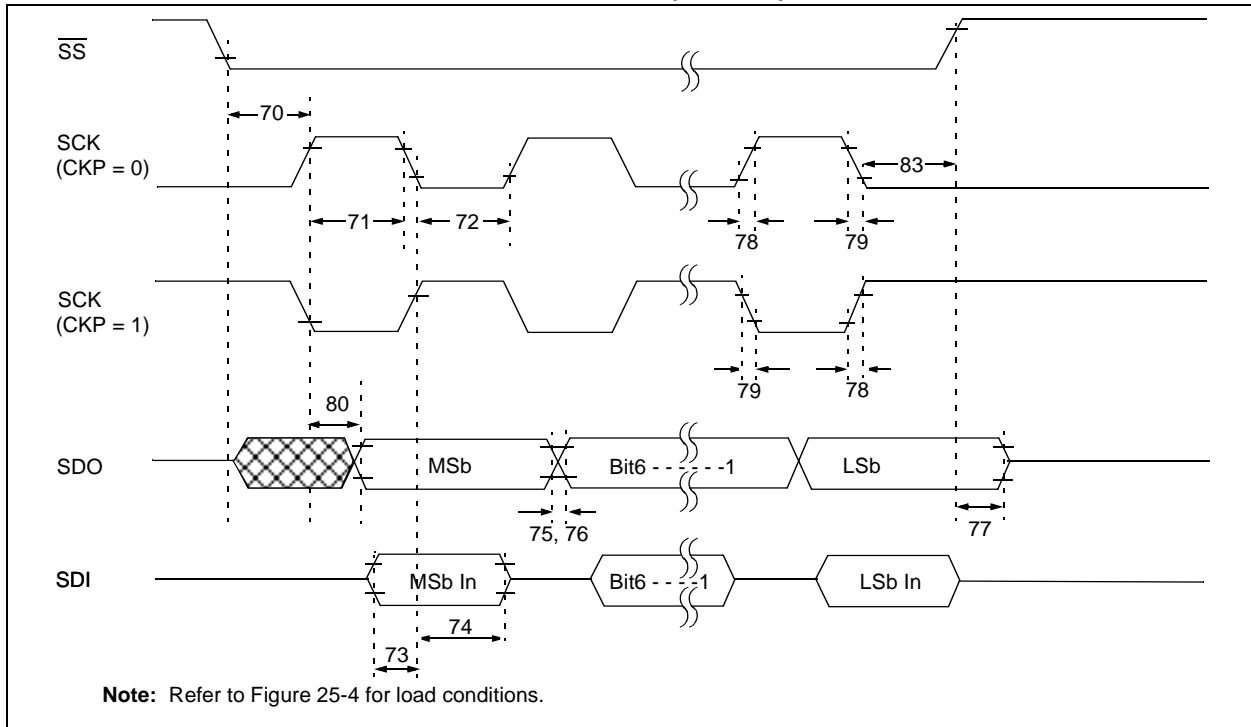


TABLE 25-13: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING (CKE = 0))

| Parm. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|-----------|--------------------|------------------------------------------------------------------------|-------------------------|--------------------------|-------|------------|
| 70 | TssL2sch, TssL2scl | $\overline{SS} \downarrow$ to SCK \downarrow or SCK \uparrow input | T _{CY} | — | ns | |
| 71 | Tsch | SCK input high time (Slave mode) | Continuous | 1.25T _{CY} + 30 | — | ns |
| 71A | | | Single Byte | 40 | — | ns |
| 72 | TscL | SCK input low time (Slave mode) | Continuous | 1.25T _{CY} + 30 | — | ns |
| 72A | | | Single Byte | 40 | — | ns |
| 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 | 1.5T _{CY} + 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 | PIC18CXX8 | — | 25 | ns |
| 76 | | | PIC18LCXX8 | — | 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) | PIC18CXX8 | — | 25 | ns |
| 79 | | | PIC18LCXX8 | — | 45 | ns |
| 79 | TscF | SCK output fall time (Master mode) | — | 25 | ns | |
| 80 | Tsch2doV, TscL2doV | SDO data output valid after SCK edge | PIC18CXX8 | — | 50 | ns |
| 83 | | | PIC18LCXX8 | — | 100 | ns |
| 83 | Tsch2ssH, TscL2ssH | $\overline{SS} \uparrow$ after SCK edge | 1.5T _{CY} + 40 | — | ns | |

Note 1: Requires the use of parameter # 73A.

Note 2: Only if parameter #s 71A and 72A are used.

PIC18CXX8

FIGURE 25-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)

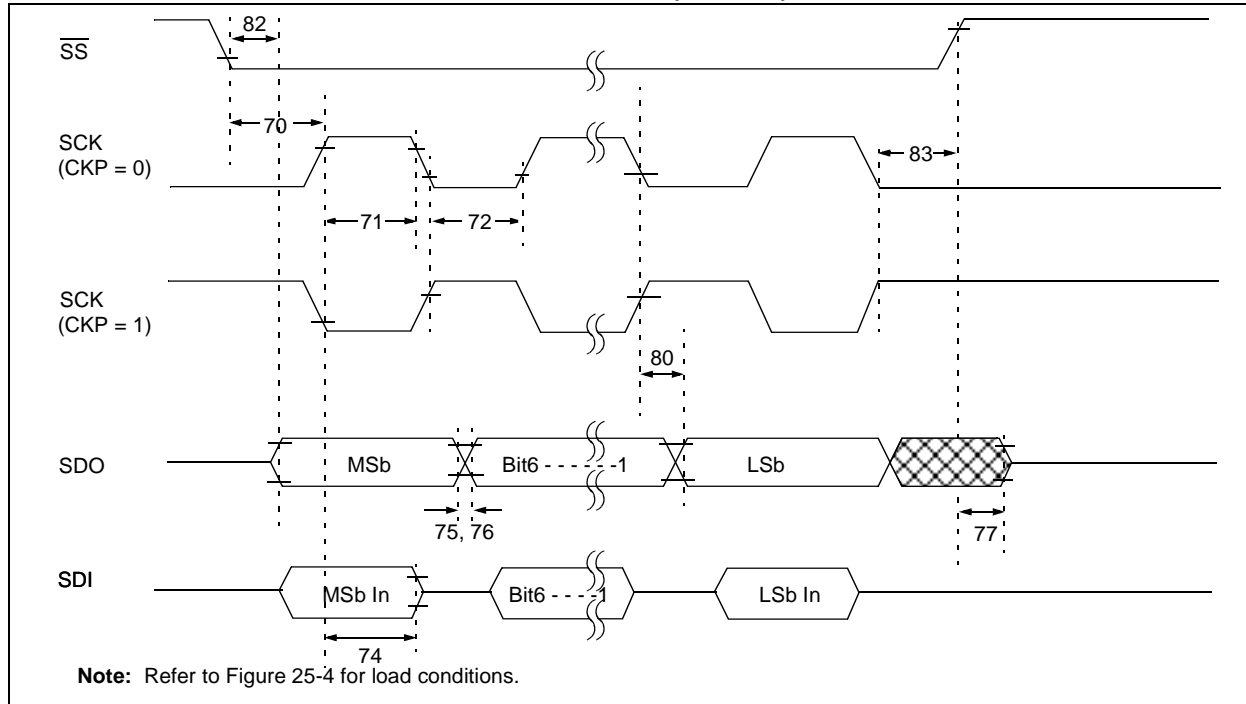


TABLE 25-14: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|------------|--------------------|-----------------------------------------------------------------------|--------------------------|-----|-------|------------|
| 70 | TssL2sch, TssL2scl | $\overline{SS}\downarrow$ to SCK \downarrow or SCK \uparrow input | T _{CY} | — | ns | |
| 71 | Tsch | SCK input high time (Slave mode) | 1.25T _{CY} + 30 | — | ns | |
| 71A | | Single Byte | 40 | — | ns | (Note 1) |
| 72 | Tscl | SCK input low time (Slave mode) | 1.25T _{CY} + 30 | — | ns | |
| 72A | | Single Byte | 40 | — | ns | (Note 1) |
| 73A | Tb2b | Last clock edge of Byte1 to the 1st clock edge of Byte2 | 1.5T _{CY} + 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 | PIC18CXX8 — | 25 | ns | |
| | | | PIC18LCXX8 — | 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) | PIC18CXX8 — | 25 | ns | |
| | | | PIC18LCXX8 — | 45 | ns | |
| 79 | TscF | SCK output fall time (Master mode) | — | 25 | ns | |
| 80 | Tsch2doV, TscL2doV | SDO data output valid after SCK edge | PIC18CXX8 — | 50 | ns | |
| | | | PIC18LCXX8 — | 100 | ns | |
| 82 | TssL2doV | SDO data output valid after $\overline{SS}\downarrow$ edge | PIC18CXX8 — | 50 | ns | |
| | | | PIC18LCXX8 — | 100 | ns | |
| 83 | Tsch2ssH, TscL2ssH | $\overline{SS}\uparrow$ after SCK edge | 1.5T _{CY} + 40 | — | ns | |

Note 1: Requires the use of parameter # 73A.

2: Only if parameter #s 71A and 72A are used.

FIGURE 25-16: I²C BUS START/STOP BITS TIMING

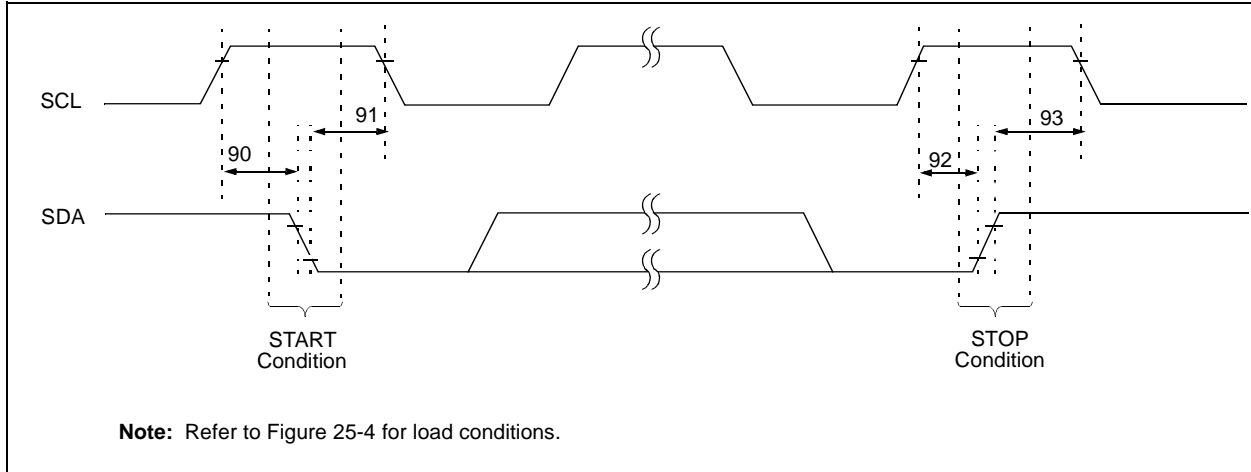


TABLE 25-15: I²C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

| Parm. 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 25-17: I²C BUS DATA TIMING

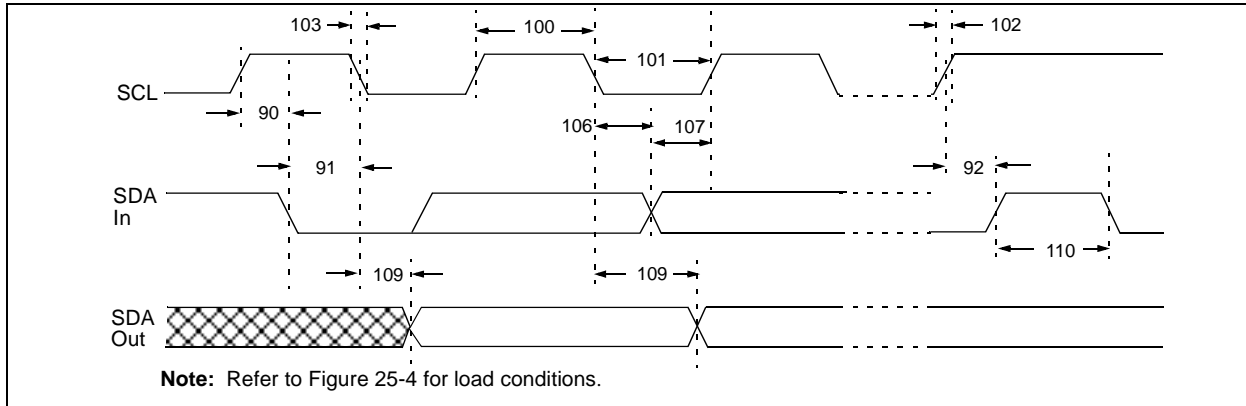


TABLE 25-16: I²C BUS DATA REQUIREMENTS (SLAVE MODE)

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions | | |
|------------|---------------------|----------------------------|------------------------------|--------------------|--------|------------|---------------------------------------------------------------|------------------------------------------------------|
| 100 | T _{HIGH} | Clock high time | 100 kHz mode | 4.0 | — | μs | PIC18CXX8 must operate at a minimum of 1.5 MHz | |
| | | | 400 kHz mode | 0.6 | — | μs | | PIC18CXX8 must operate at a minimum of 10 MHz |
| | | | SSP Module | 1.5T _{CY} | — | — | | |
| 101 | T _{LOW} | Clock low time | 100 kHz mode | 4.7 | — | μs | PIC18CXX8 must operate at a minimum of 1.5 MHz | |
| | | | 400 kHz mode | 1.3 | — | μs | | PIC18CXX8 must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5T _{CY} | — | ns | | |
| 102 | T _R | 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 | | |
| 103 | T _F | 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 | | |
| 90 | T _{SU:STA} | START condition setup time | 100 kHz mode 400 kHz mode | 4.7 0.6 | — — | μs μs | Only relevant for repeated START condition | |
| 91 | T _{HD:STA} | START condition hold time | 100 kHz mode 400 kHz mode | 4.0 0.6 | — — | μs μs | | After this period the first clock pulse is generated |
| 106 | T _{HD:DAT} | Data input hold time | 100 kHz mode | 0 | — | ns | | |
| | | | 400 kHz mode | 0 | 0.9 | μs | | |
| 107 | T _{SU:DAT} | Data input setup time | 100 kHz mode | 250 | — | ns | (Note 2) | |
| | | | 400 kHz mode | 100 | — | ns | | |
| 92 | T _{SU:STO} | STOP condition setup time | 100 kHz mode | 4.7 | — | μs | | |
| | | | 400 kHz mode | 0.6 | — | μs | | |
| 109 | T _{AA} | Output valid from clock | 100 kHz mode | — | 3500 | ns | (Note 1) | |
| | | | 400 kHz mode | — | — | ns | | |
| 110 | T _{BUF} | 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 | C _b | 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 t_{SU: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, T_R max. + t_{SU:DAT} = 1000 + 250 = 1250 ns (according to the standard mode I²C bus specification).

FIGURE 25-18: MASTER SSP I²C BUS START/STOP BITS TIMING WAVEFORMS

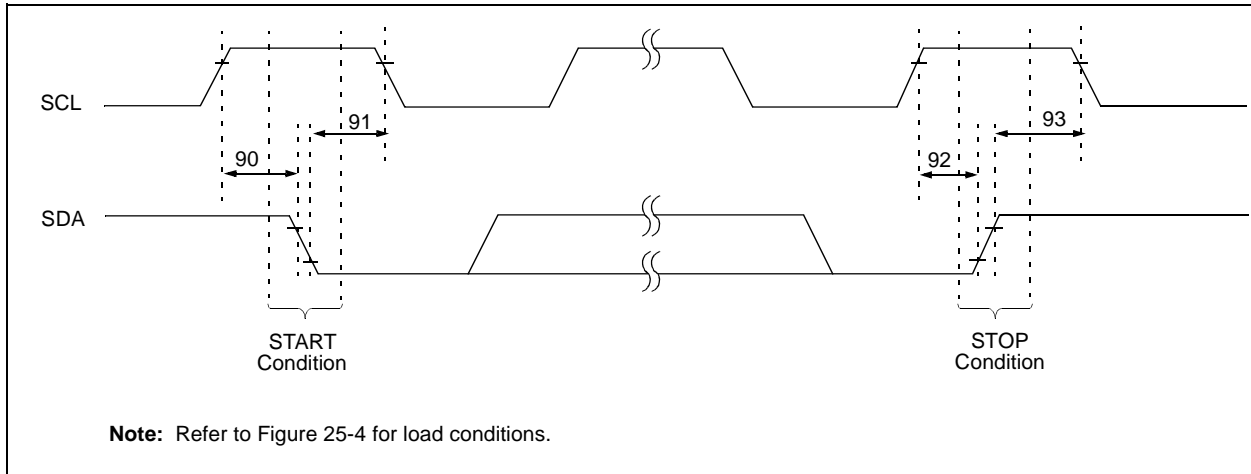


TABLE 25-17: MASTER SSP I²C BUS START/STOP BITS REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions | |
|------------|---------|----------------------------|---------------------------|-----------------------|-------|------------|-------------------------------------------------------|
| 90 | TSU:STA | START condition Setup time | 100 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | ns | Only relevant for Repeated START condition |
| | | | 400 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | | |
| | | | 1 MHz mode ⁽¹⁾ | $2(T_{OSC})(BRG + 1)$ | — | | |
| 91 | THD:STA | START condition Hold time | 100 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | | |
| | | | 1 MHz mode ⁽¹⁾ | $2(T_{OSC})(BRG + 1)$ | — | | |
| 92 | TSU:STO | STOP condition Setup time | 100 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | ns | |
| | | | 400 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | | |
| | | | 1 MHz mode ⁽¹⁾ | $2(T_{OSC})(BRG + 1)$ | — | | |
| 93 | THD:STO | STOP condition Hold time | 100 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | ns | |
| | | | 400 kHz mode | $2(T_{OSC})(BRG + 1)$ | — | | |
| | | | 1 MHz mode ⁽¹⁾ | $2(T_{OSC})(BRG + 1)$ | — | | |

Note 1: Maximum pin capacitance = 10 pF for all I²C pins.

PIC18CXX8

FIGURE 25-19: MASTER SSP I²C BUS DATA TIMING

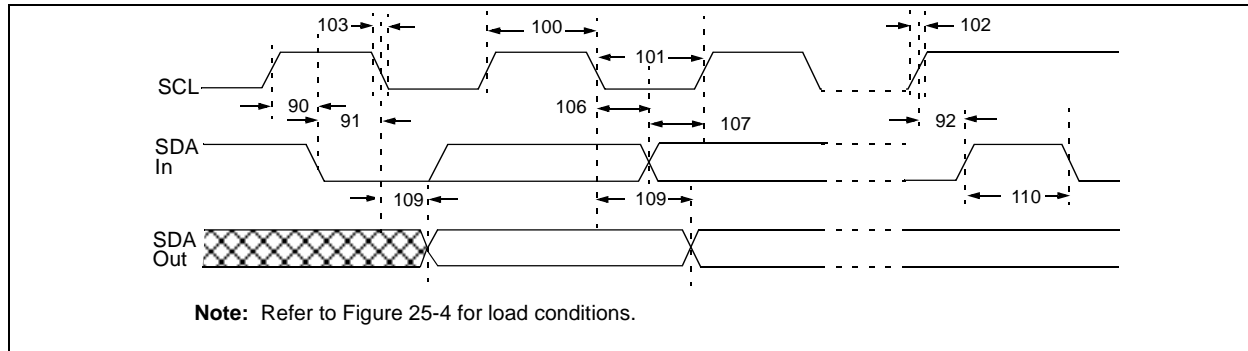


TABLE 25-18: MASTER SSP I²C BUS DATA REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|------------|---------------------|----------------------------|---------------------------|-------------------------------|-------|------------|
| 100 | T _{HIGH} | Clock high time | 100 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 400 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 1 MHz mode ⁽¹⁾ | 2(T _{osc})(BRG + 1) | — | ms |
| 101 | T _{LOW} | Clock low time | 100 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 400 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 1 MHz mode ⁽¹⁾ | 2(T _{osc})(BRG + 1) | — | ms |
| 102 | T _R | SDA and SCL rise time | 100 kHz mode | — | 1000 | ns |
| | | | 400 kHz mode | 20 + 0.1C _b | 300 | ns |
| | | | 1 MHz mode ⁽¹⁾ | — | 300 | ns |
| 103 | T _F | SDA and SCL fall time | 100 kHz mode | — | 300 | ns |
| | | | 400 kHz mode | 20 + 0.1C _b | 300 | ns |
| | | | 1 MHz mode ⁽¹⁾ | — | 100 | ns |
| 90 | T _{SU:STA} | START condition setup time | 100 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 400 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 1 MHz mode ⁽¹⁾ | 2(T _{osc})(BRG + 1) | — | ms |
| 91 | T _{HD:STA} | START condition hold time | 100 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 400 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 1 MHz mode ⁽¹⁾ | 2(T _{osc})(BRG + 1) | — | ms |
| 106 | T _{HD:DAT} | Data input hold time | 100 kHz mode | 0 | — | ns |
| | | | 400 kHz mode | 0 | 0.9 | ms |
| | | | 1 MHz mode ⁽¹⁾ | TBD | — | ns |
| 107 | T _{SU:DAT} | Data input setup time | 100 kHz mode | 250 | — | ns |
| | | | 400 kHz mode | 100 | — | ns |
| | | | 1 MHz mode ⁽¹⁾ | TBD | — | ns |
| 92 | T _{SU:STO} | STOP condition setup time | 100 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 400 kHz mode | 2(T _{osc})(BRG + 1) | — | ms |
| | | | 1 MHz mode ⁽¹⁾ | 2(T _{osc})(BRG + 1) | — | ms |
| 109 | T _A | Output valid from clock | 100 kHz mode | — | 3500 | ns |
| | | | 400 kHz mode | — | 1000 | ns |
| | | | 1 MHz mode ⁽¹⁾ | — | — | ns |
| 110 | T _{BUF} | Bus free time | 100 kHz mode | 4.7 | — | ms |
| | | | 400 kHz mode | 1.3 | — | ms |
| | | | 1 MHz mode ⁽¹⁾ | TBD | — | ms |
| D102 | C _b | 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 25-20: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

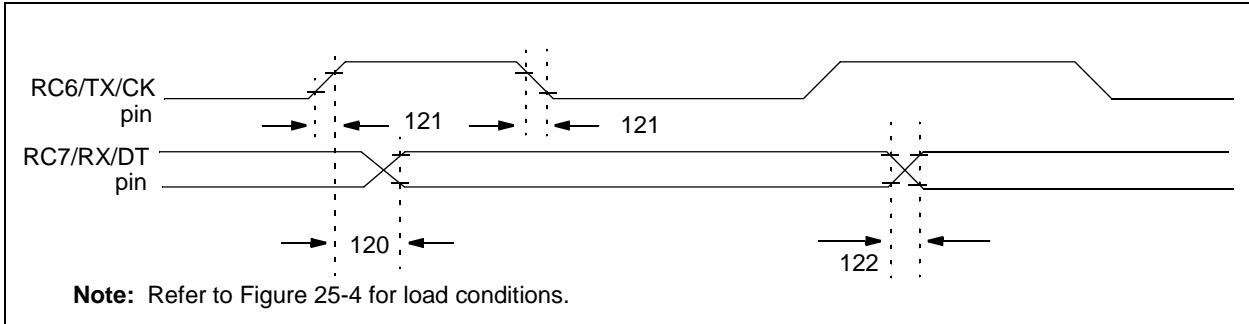


TABLE 25-19: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|------------|----------|------------------------------------------------------------|------------|-----|-------|------------|
| 120 | TckH2dtV | SYNC XMIT (Master & Slave) Clock high to data-out valid | PIC18CXX8 | — | 40 | ns |
| | | | PIC18LCXX8 | — | 100 | ns |
| 121 | Tckrf | Clock out rise time and fall time (Master mode) | PIC18CXX8 | — | 20 | ns |
| | | | PIC18LCXX8 | — | 50 | ns |
| 122 | Tdtrf | Data-out rise time and fall time | PIC18CXX8 | — | 20 | ns |
| | | | PIC18LCXX8 | — | 50 | ns |

PIC18CXX8

FIGURE 25-21: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

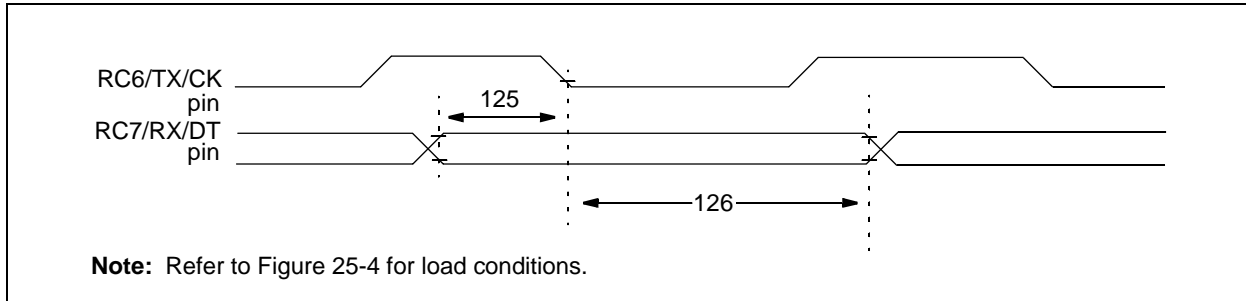


TABLE 25-20: USART SYNCHRONOUS RECEIVE REQUIREMENTS

| 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 25-21: A/D CONVERTER CHARACTERISTICS: PIC18CXX8 (INDUSTRIAL, EXTENDED)
PIC18LCXX8 (INDUSTRIAL)**

| Param No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|-----------|--------|------------------------------------------------|---------------------------|-----|------------------|------------|---------------------------------------------------------------------------------------------------------------------------------|
| A01 | NR | Resolution | — | — | 10 TBD | bit bit | $V_{REF} = V_{DD} \geq 3.0V$ $V_{REF} = V_{DD} < 3.0V$ |
| A03 | EIL | Integral linearity error | — | — | $< \pm 1$ TBD | LSb LSb | $V_{REF} = V_{DD} \geq 3.0V$ $V_{REF} = V_{DD} < 3.0V$ |
| A04 | EDL | Differential linearity error | — | — | $< \pm 1$ TBD | LSb LSb | $V_{REF} = V_{DD} \geq 3.0V$ $V_{REF} = V_{DD} < 3.0V$ |
| A05 | EFS | Full scale error | — | — | $< \pm 1$ TBD | LSb LSb | $V_{REF} = V_{DD} \geq 3.0V$ $V_{REF} = V_{DD} < 3.0V$ |
| A06 | EOFF | Offset error | — | — | $< \pm 1$ TBD | LSb LSb | $V_{REF} = V_{DD} \geq 3.0V$ $V_{REF} = V_{DD} < 3.0V$ |
| A10 | — | Monotonicity | guaranteed ⁽³⁾ | | | — | $V_{SS} \leq V_{AIN} \leq V_{REF}$ |
| A20 | VREF | Reference voltage | 0V | — | — | V | For 10-bit resolution |
| A20A | | (VREFH - VREFL) | 3V | — | — | V | |
| A21 | VREFH | Reference voltage High | AVSS | — | AVDD + 0.3V | V | |
| A22 | VREFL | Reference voltage Low | AVSS - 0.3V | — | AVDD | V | |
| A25 | VAIN | Analog input voltage | AVSS - 0.3V | — | VREF + 0.3V | V | |
| A30 | ZAIN | Recommended impedance of analog voltage source | — | — | 10.0 | k Ω | |
| A40 | IAD | A/D conversion current (VDD) | PIC18CXX8 — | 180 | — | μA | Average current consumption when A/D is on ⁽¹⁾ . |
| | | PIC18LCXXX | — | 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 18.0. During A/D conversion cycle. |
| | | | — | — | 10 | μA | |

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: $V_{SS} \leq V_{AIN} \leq V_{REF}$

3: The A/D conversion result either increases or remains constant as the analog input increases.

PIC18CXX8

FIGURE 25-22: A/D CONVERSION TIMING

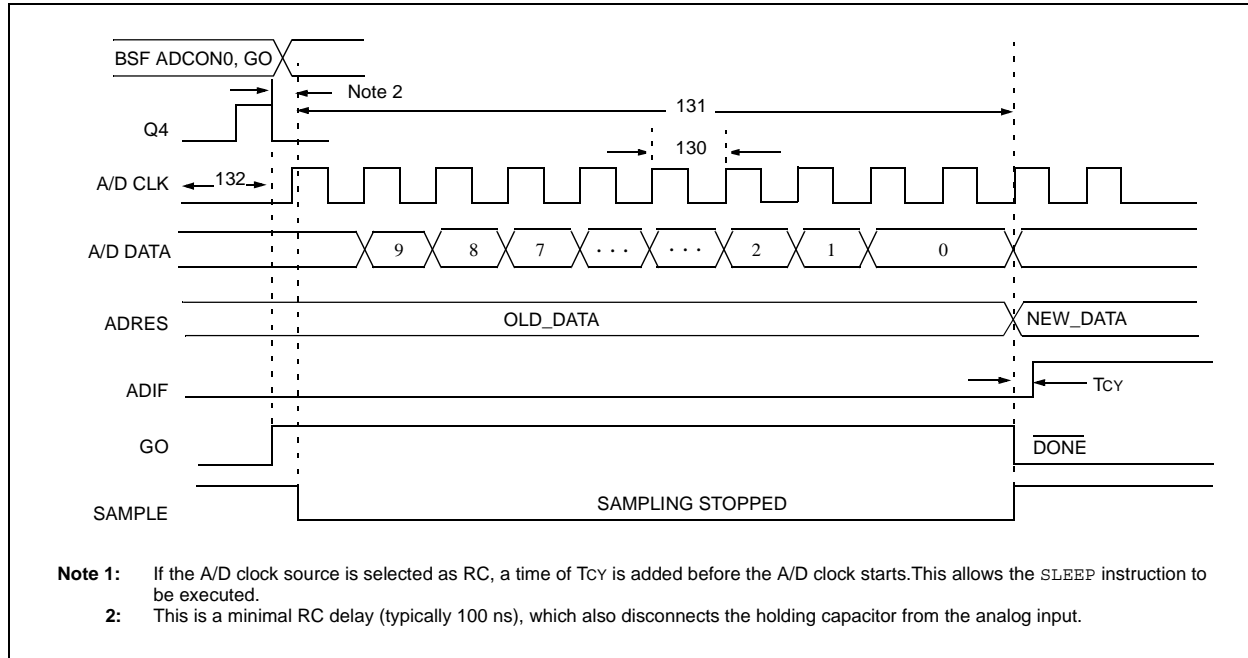


TABLE 25-22: A/D CONVERSION REQUIREMENTS

| Param No. | Sym- bol | Characteristic | Min | Max | Units | Conditions | |
|-----------|----------|-----------------------------------------------------------------|------------|----------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| 130 | TAD | A/D clock period | PIC18CXX8 | 1.6 | 20 ⁽⁵⁾ | μs | TOSC based, VREF ≥ 3.0V |
| | | | PIC18LCXX8 | 3.0 | 20 ⁽⁵⁾ | μs | TOSC based, VREF full range |
| | | | PIC18CXX8 | 2.0 | 6.0 | μs | A/D RC mode |
| | | | PIC18LCXX8 | 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 | — | μs | -40°C ≤ Temp ≤ 125°C |
| | | | | 10 | — | μs | 0°C ≤ Temp ≤ 125°C |
| 135 | TSWC | Switching time from convert → sample | — | (Note 4) | | | |
| 136 | TAMP | Amplifier settling time (Note 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 T_{cy} cycle.

2: See Section 18.0 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 (R_s) on the input channels is 50 Ω.

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.

26.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

Graphs and Tables are not available at this time.

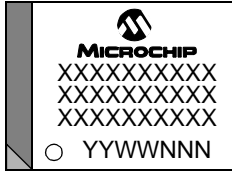
PIC18CXX8

NOTES:

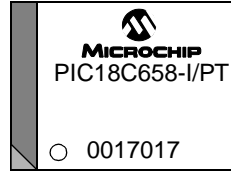
27.0 PACKAGING INFORMATION

27.1 Package Marking Information

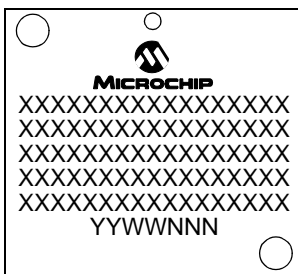
64-Lead TQFP



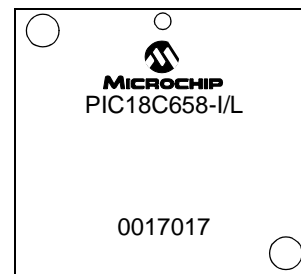
Example



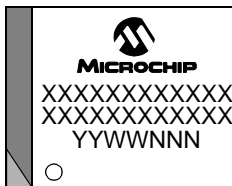
68-Lead PLCC



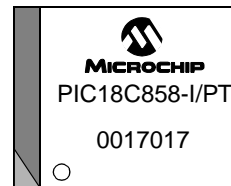
Example



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 |

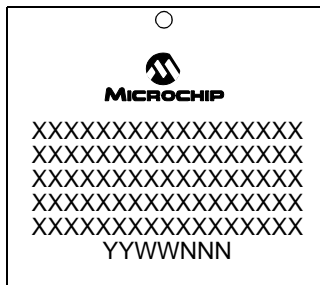
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

* Standard OTP marking consists of Microchip part number, year code, week code and traceability 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.

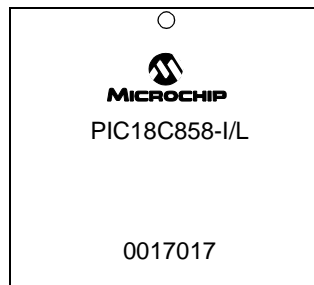
PIC18CXX8

Package Marking Information (Cont'd)

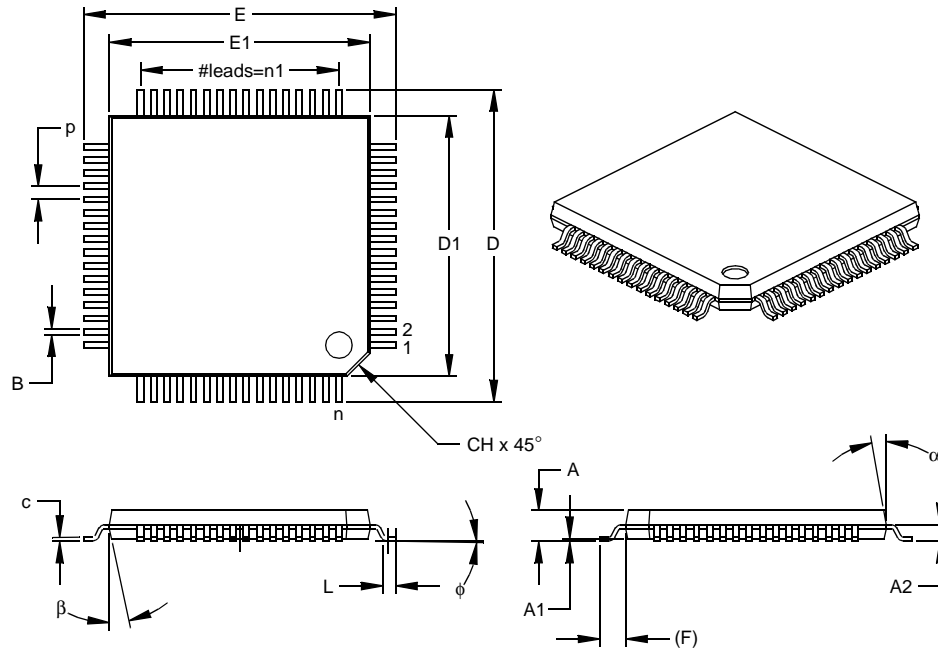
84-Lead PLCC



Example



64-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)



| Dimension Limits | Units | INCHES | | | MILLIMETERS* | | |
|--------------------------|-------|--------|------|------|--------------|-------|-------|
| | | 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 | phi | 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 | alpha | 5 | 10 | 15 | 5 | 10 | 15 |
| Mold Draft Angle Bottom | beta | 5 | 10 | 15 | 5 | 10 | 15 |

* Controlling Parameter
 § Significant Characteristic

Notes:

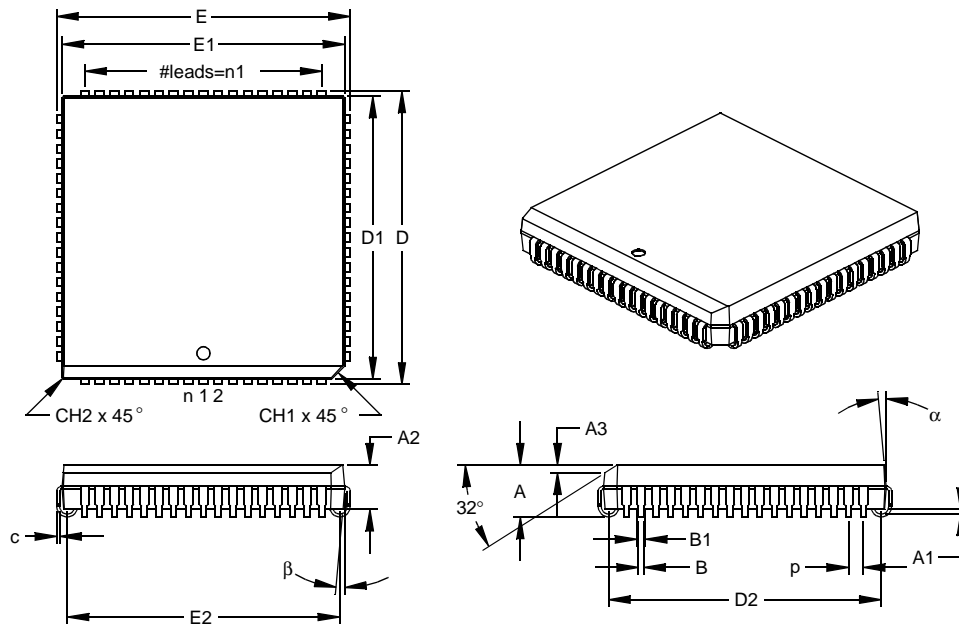
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC Equivalent: MS-026

Drawing No. C04-085

PIC18CXX8

68-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)



| 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:

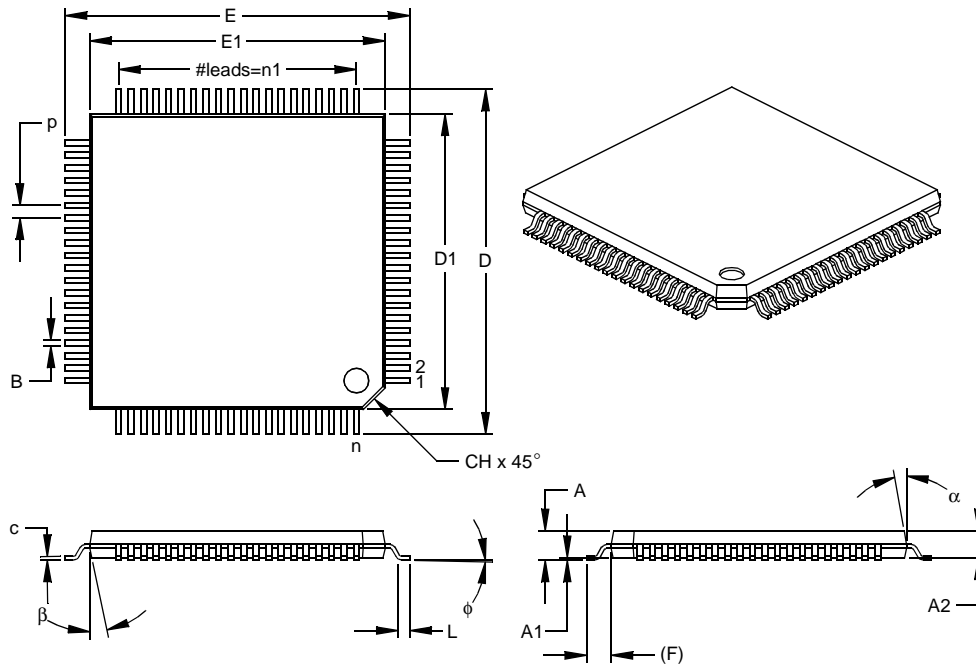
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed

.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)



| Dimension Limits | Units | INCHES | | | MILLIMETERS* | | |
|--------------------------|-------|--------|------|------|--------------|-------|-------|
| | | 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:

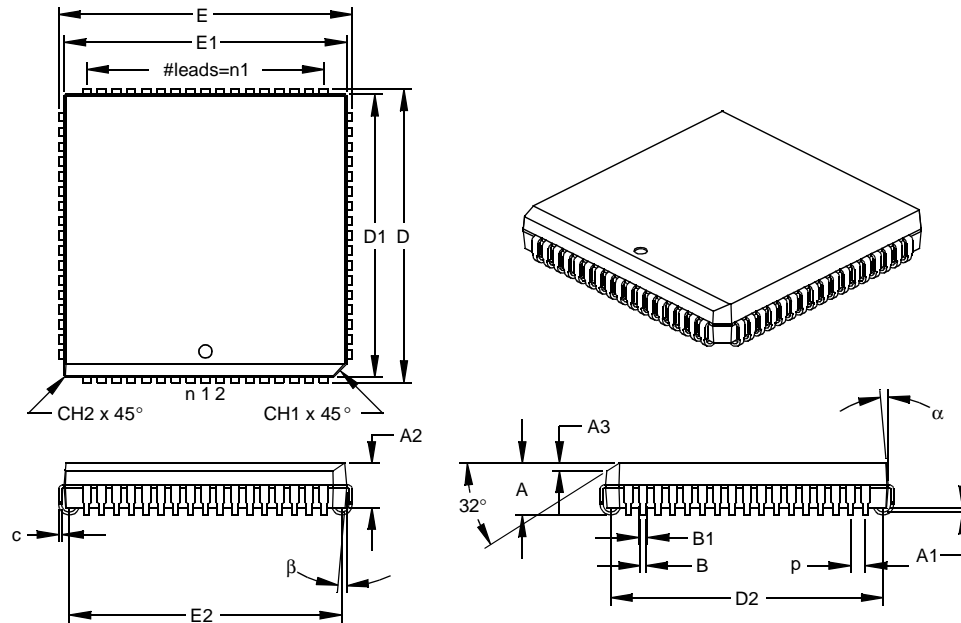
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC Equivalent: MS-026

Drawing No. C04-092

PIC18CXX8

84-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)



| Dimension Limits | | INCHES* | | | MILLIMETERS | | |
|--------------------------|-----|---------|------|------|-------------|-------|-------|
| | | 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.

APPENDIX B: DEVICE DIFFERENCES

The differences between the PIC18CXX8 devices listed in this data sheet are shown in Table B-1.

TABLE B-1: DEVICE DIFFERENCES

| Feature | | PIC18C658 | PIC18C858 |
|----------------------------|----------|-----------|-----------|
| Program Memory (Bytes) | | 32K | 32K |
| Data Memory (Bytes) | | 1.5K | 1.5K |
| A/D Channels | | 12 | 16 |
| Parallel Slave Port (PSP) | | Yes | Yes |
| External Memory Capability | | No | No |
| Package Types | TQFP | 64-pin | 80-pin |
| | PLCC | 68-pin | 84-pin |
| | JCERPACK | 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 PICMICRO DEVICES

This discusses some of the issues in migrating from other PICmicro 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: version 5.11

MPLAB-SIM: version 7.10

MPLAB-ICE 2000:

PIC18CXX8 Processor Module:

Part Number - PCM 18XB0

PIC18CXX8 Device Adapter:

Socket Part Number

64-pin TQFP DVD18P2640

68-pin PLCC DVD18XL680

80-pin TQFP DVD18PQ800

84-pin PLCC DVD18XL840

MPLAB-ICD: Not Available

PROMATE II: version 5.20

PICSTART Plus: version 2.20

MPASM: version 2.50

MPLAB-C18: version 1.00

CAN-TOOL: Not available at time of printing.

| |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>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.</p> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

PIC18CXX8

NOTES:

INDEX

A

| | |
|---------------------------------------------|----------|
| A/D | 227 |
| A/D Converter Flag (ADIF Bit) | 230 |
| A/D Converter Interrupt, Configuring | 231 |
| ADCON0 Register | 227, 229 |
| ADCON1 Register | 227, 228 |
| ADCON2 Register | 227 |
| ADRES Register | 227, 230 |
| Analog Port Pins, Configuring | 233 |
| Block Diagram | 230 |
| Block Diagram, Analog Input Model | 231 |
| Configuring the Module | 231 |
| Conversion Clock (TAD) | 233 |
| Conversion Status (GO/DONE Bit) | 230 |
| Conversions | 234 |
| Converter Characteristics | 339 |
| converter characteristics | 318 |
| Effects of a RESET | 250 |
| Equations | 232 |
| Operation During SLEEP | 250 |
| Sampling Requirements | 232 |
| Sampling Time | 232 |
| Special Event Trigger (CCP) | 130, 234 |
| Timing Diagram | 340 |
| Absolute Maximum Ratings | 311 |
| Acknowledge Error | 223 |
| ADCON0 Register | 227, 229 |
| GO/DONE Bit | 230 |
| ADCON1 Register | 227, 228 |
| ADCON2 Register | 227 |
| ADDLW | 267 |
| ADDWF | 267 |
| ADDWFC | 268 |
| ADRES Register | 227, 230 |
| AKS | 156 |
| Analog-to-Digital Converter. <i>See</i> A/D | |
| ANDLW | 268 |
| ANDWF | 269 |
| Assembler | |
| MPASM Assembler | 305 |

B

| | |
|------------------------------------------|---------|
| Baud Rate Generator | 153 |
| BCF | 270 |
| BF | 156 |
| Bit Error | 223 |
| Bit Timing | 218 |
| Bit Timing Configuration Registers | 222 |
| Block Diagrams | |
| Baud Rate Generator | 153 |
| Comparator I/O Operating Modes | 238 |
| PORTK | 108 |
| SSP (SPI Mode) | 141 |
| Timer3 | 124 |
| BOR. <i>See</i> Brown-out Reset | |
| BRG | 153 |
| Brown-out Reset (BOR) | 30, 251 |
| Timing Diagram | 325 |

| | |
|------------------------------------------------|----------------------------------------|
| BSF | 269, 270, 271, 272, 273, 275, 276, 291 |
| BTFSC | 274 |
| BTFSS | 274 |
| BTG | 275 |
| Bus Activity Wake-up Interrupt | 225 |
| Bus Collision During a RESTART Condition | 165 |
| Bus Collision During a START Condition | 163 |
| Bus Collision During a STOP Condition | 166 |
| Bus Off | 226 |

C

| | |
|-----------------------------------------------------|--------------------|
| CALL | 276 |
| CAN Buffers and Protocol Engine Block Diagram | 184 |
| Capture (CCP Module) | 128 |
| Block Diagram | 129 |
| CCP Pin Configuration | 128 |
| CCPR1H:CCPR1L Registers | 128 |
| Changing Between Capture Prescalers | 129 |
| Software Interrupt | 129 |
| Timer1 Mode Selection | 128 |
| Capture/Compare/PWM (CCP) | 127 |
| Capture Mode. <i>See</i> Capture | |
| CCP1 | 128 |
| CCPR1H Register | 128 |
| CCPR1L Register | 128 |
| CCP2 | 128 |
| CCPR2H Register | 128 |
| CCPR2L Register | 128 |
| Compare Mode. <i>See</i> Compare | |
| Interaction of Two CCP Modules | 128 |
| PWM Mode. <i>See</i> PWM | |
| Timer Resources | 128 |
| Timing Diagram | 327 |
| Clocking Scheme | 45 |
| CLRF | 277, 295 |
| CLRWDT | 277 |
| Code Examples | |
| Loading the SSPBUF Register | 142 |
| Code Protection | 251, 259 |
| COMF | 278 |
| Comparator Interrupts | 241 |
| Comparator Operation | 239 |
| Comparator Reference | 239 |
| Compare (CCP Module) | 130 |
| Block Diagram | 130 |
| CCP Pin Configuration | 130 |
| CCPR1H:CCPR1L Registers | 130 |
| Software Interrupt | 130 |
| Special Event Trigger | 119, 125, 130, 234 |
| Timer1 Mode Selection | 130 |
| Configuration Bits | 251 |
| Configuration Mode | 210 |
| Configuring the Voltage Reference | 243 |
| CPFSEQ | 278 |
| CPFSGT | 279 |
| CPFSLT | 279 |
| CRC Error | 223 |
| CVRCON Register | 243 |

PIC18CXX8

D

| | |
|----------------------------------|-------------------------|
| Data Memory | 48 |
| General Purpose Registers | 48 |
| Special Function Registers | 48 |
| DAW | 280 |
| DC Characteristics | 313, 314, 315, 316, 317 |
| DECF | 280 |
| DECFSNZ | 281 |
| DECFSZ | 281 |
| Device Differences | 349 |
| Device Functionality | 184 |
| Direct Addressing | 62 |

E

| | |
|--------------------------------------|-----|
| Electrical Characteristics | 311 |
| Errata | 7 |
| Error Detection | 223 |
| Error Interrupt | 226 |
| Error Modes | 224 |
| Error Modes and Error Counters | 223 |
| Error States | 223 |

F

| | |
|-------------------------------|-----|
| Filter/Mask Truth Table | 216 |
| Firmware Instructions | 261 |
| Form Error | 223 |

G

| | |
|-------------------------------------|-----|
| General Call Address Sequence | 150 |
| General Call Address Support | 150 |
| GOTO | 282 |

H

| | |
|----------------------------|-----|
| Hard Synchronization | 220 |
|----------------------------|-----|

I

| | |
|------------------------------------------------------|---------------|
| I/O Ports | 89 |
| I ² C (SSP Module) | 147 |
| ACK Pulse | 147, 148, 149 |
| Addressing | 148 |
| Block Diagram | 147 |
| Read/Write Bit Information (R/W Bit) | 148, 149 |
| Reception | 149 |
| Serial Clock (RC3/SCK/SCL) | 149 |
| Slave Mode | 147 |
| Timing Diagram, Data | 334 |
| Timing Diagram, Start/Stop Bits | 333 |
| Transmission | 149 |
| I ² C Master Mode Reception | 156 |
| I ² C Master Mode Restart Condition | 155 |
| I ² C Module | |
| Acknowledge Sequence timing | 159 |
| Baud Rate Generator | 153 |
| BRG Block Diagram | 153 |
| BRG Reset due to SDA Collision | 164 |
| BRG Timing | 153 |
| Bus Collision | |
| Acknowledge | 162 |
| Restart Condition | 165 |
| Restart Condition Timing (Case1) | 165 |
| Restart Condition Timing (Case2) | 165 |
| START Condition | 163 |
| Start Condition Timing | 163, 164 |
| STOP Condition | 166 |
| STOP Condition Timing (Case1) | 166 |
| STOP Condition Timing (Case2) | 166 |
| Transmit Timing | 162 |

| | |
|--------------------------------------------------|----------------------------------------|
| Bus Collision timing | 162 |
| Clock Arbitration | 161 |
| Clock Arbitration Timing (Master Transmit) | 161 |
| General Call Address Support | 150 |
| Master Mode 7-bit Reception timing | 158 |
| Master Mode Operation | 152 |
| Master Mode Start Condition | 154 |
| Master Mode Transmission | 156 |
| Master Mode Transmit Sequence | 152 |
| Multi-Master Mode | 162 |
| Repeat START Condition timing | 155 |
| STOP Condition Receive or Transmit timing | 160 |
| STOP Condition timing | 159 |
| Waveforms for 7-bit Reception | 149 |
| Waveforms for 7-bit Transmission | 149 |
| ID Locations | 251, 259 |
| INCF | 282 |
| INCFSNZ | 283 |
| INCFSZ | 283 |
| In-Circuit Serial Programming (ICSP) | 251, 259 |
| Indirect Addressing | 62 |
| FSR Register | 61 |
| Information Processing Time | 219 |
| Initiating Message Transmission | 211 |
| Instruction Cycle | 45 |
| Instruction Flow/Pipelining | 46 |
| Instruction Format | 263 |
| Instruction Set | 261 |
| ADDLW | 267 |
| ADDWF | 267 |
| ADDWFC | 268 |
| ANDLW | 268 |
| ANDWF | 269 |
| BCF | 270 |
| BSF | 269, 270, 271, 272, 273, 275, 276, 291 |
| BTFSC | 274 |
| BTFSS | 274 |
| BTG | 275 |
| CALL | 276 |
| CLRF | 277, 295 |
| CLRWDT | 277 |
| COMF | 278 |
| CPFSEQ | 278 |
| CPFSGT | 279 |
| CPFSLT | 279 |
| DAW | 280 |
| DECF | 280 |
| DECFSNZ | 281 |
| DECFSZ | 281 |
| GOTO | 282 |
| INCF | 282 |
| INCFSNZ | 283 |
| INCFSZ | 283 |
| IORLW | 284 |
| IORWF | 284 |
| MOVFP | 286 |
| MOVLB | 285 |
| MOVLR | 285, 286 |
| MOVLW | 287 |
| MOVWF | 287 |
| MULLW | 288 |
| MULWF | 288 |
| NEGW | 289 |
| NOP | 289 |
| RETfie | 291, 292 |
| RETLW | 292 |

| | | | |
|-----------------------------------------------------|---------------|---------------------------------------------------------|---------------|
| RETURN | 293 | MPLAB Integrated Development Environment Software | 305 |
| RLCF | 293 | MULLW | 288 |
| RLNCF | 294 | Multi-Master Mode | 162 |
| RRCF | 294 | Multiply Examples | |
| RRNCF | 295 | 16 x 16 Routine | 72 |
| SLEEP | 296 | 16 x 16 Signed Routine | 73 |
| SUBLW | 297 | 8 x 8 Routine | 72 |
| SUBWF | 297, 298 | 8 x 8 Signed Routine | 72 |
| SUBWFB | 299 | MULWF | 288 |
| SWAPF | 300 | N | |
| TABLRD | 301 | NEGW | 289 |
| TABLWT | 302 | NOP | 289 |
| TSTFSZ | 303 | Normal Mode | 210 |
| XORLW | 303 | O | |
| XORWF | 304 | OPTION_REG Register | 64 |
| Summary Table | 264 | PS2:PS0 Bits | 115 |
| INT Interrupt (RB0/INT). See Interrupt Sources | | PSA Bit | 115 |
| INTCON Register | | T0CS Bit | 115 |
| RBIF Bit | 91 | T0SE Bit | 115 |
| Inter-Integrated Circuit. See I ² C | | OSCCON | 25 |
| Interrupt Acknowledge | 226 | OSCCON Register | 25 |
| Interrupt Sources | 75, 251 | Oscillator Configuration | 21, 251 |
| A/D Conversion Complete | 231 | HS | 21 |
| Capture Complete (CCP) | 129 | HS + PLL | 21 |
| Compare Complete (CCP) | 130 | LP | 21 |
| Interrupt-on-Change (RB7:RB4) | 91 | RC | 21, 23 |
| RB0/INT Pin, External | 88 | RCIO | 21 |
| SSP Receive/Transmit Complete | 135 | XT | 21 |
| TMR0 Overflow | 116 | Oscillator Tolerance | 222 |
| TMR1 Overflow | 117, 119 | Oscillator, Timer1 | 117, 119, 123 |
| TMR2 to PR2 Match | 122 | Oscillator, Timer3 | 125 |
| TMR2 to PR2 Match (PWM) | 121, 132 | Oscillator, WDT | 255 |
| TMR3 Overflow | 123, 125 | Overview | 183 |
| USART Receive/Transmit Complete | 167 | P | |
| Interrupts | 225 | Packaging | 343 |
| Interrupts, Enable Bits | | Parallel Slave Port (PSP) | 95, 109 |
| CCP1 Enable (CCP1IE Bit) | 129 | Block Diagram | 109 |
| Interrupts, Flag Bits | | RE0/RD | 109 |
| A/D Converter Flag (ADIF Bit) | 230 | RE1/WR | 109 |
| CCP1 Flag (CCP1IF Bit) | 128, 129, 130 | RE2/CS | 109 |
| Interrupt on Change (RB7:RB4) Flag (RBIF Bit) | 91 | Read Waveforms | 111 |
| IORLW | 284 | Select (PSPMODE Bit) | 95, 109 |
| IORWF | 284 | Timing Diagram | 328 |
| K | | Write Waveforms | 111 |
| KEELOQ Evaluation and Programming Tools | 308 | Phase Buffer Segments | 219 |
| L | | PICDEM 1 Low Cost PICmicro Demo Board | 307 |
| Lengthening a Bit Period | 221 | PICDEM 2 Low Cost PIC16CXX Demo Board | 307 |
| Listen Only Mode | 210 | PICDEM 3 Low Cost PIC16CXXX Demo Board | 308 |
| Loopback Mode | 211 | PICSTART Plus Entry Level Development System | 307 |
| M | | Pin Functions | |
| Memory Organization | | AVDD | 20 |
| Data Memory | 48 | AVSS | 20 |
| Program Memory | 41 | MCLR/VPP | 12 |
| Message Acceptance Filter | 217 | OSC1/CLKI | 12 |
| Message Acceptance Filters and Masks | 216 | OSC2/CLKO | 12 |
| Message Reception | 213 | RA0/AN0 | 13 |
| Message Reception Flowchart | 215 | RA1/AN1 | 13 |
| MOVFP | 286 | RA2/AN2/VREF- | 13 |
| MOVLB | 285 | RA3/AN3/VREF+ | 13 |
| MOVLR | 285, 286 | RA4/T0CKI | 13 |
| MOVLW | 287 | RA5/AN4/SS/LVDIN | 13 |
| MOVWF | 287 | RA6 | 13 |
| | | RB0/INT0 | 14 |

PIC18CXX8

| | | | |
|-----------------------|----|---------------------------------------------------|---------|
| RB1/INT1 | 14 | RK0 | 20 |
| RB2/INT2 | 14 | RK1 | 20 |
| RB3/INT3 | 14 | RK2 | 20 |
| RB4 | 14 | RK3 | 20 |
| RB5 | 14 | VDD | 20 |
| RB6 | 14 | Vss | 20 |
| RB7 | 14 | Pointer, FSR | 61 |
| RC0/T1OSO/T1CKI | 15 | POR. See Power-on Reset | |
| RC1/T1OSI | 15 | PORTA | |
| RC2/CCP1 | 15 | Initialization | 89 |
| RC3/SCK/SCL | 15 | PORTA Register | 89 |
| RC4/SDI/SDA | 15 | RA3:RA0 and RA5 Port Pins | 89 |
| RC5/SDO | 15 | RA4/T0CKI Pin | 90 |
| RC6/TX/CK | 15 | TRISA Register | 89 |
| RC7/RX/DT | 15 | PORTB | |
| RD0/AD0 | 16 | Initialization | 91 |
| RD0/PSP0 | 16 | PORTB Register | 91 |
| RD1/AD1 | 16 | RB0/INT Pin, External | 88 |
| RD1/PSP1 | 16 | RB3:RB0 Port Pins | 91 |
| RD2/AD2 | 16 | RB7:RB4 Interrupt on Change Flag (RBIF Bit) | 91 |
| RD2/PSP2 | 16 | RB7:RB4 Port Pins | 91 |
| RD3/AD3 | 16 | TRISB Register | 91 |
| RD3/PSP3 | 16 | PORTC | |
| RD4/AD4 | 16 | Block Diagram | 93 |
| RD4/PSP4 | 16 | Initialization | 93 |
| RD5/AD5 | 16 | PORTC Register | 93 |
| RD5/PSP5 | 16 | RC3/SCK/SCL Pin | 149 |
| RD6/AD6 | 16 | RC7/RX/DT Pin | 169 |
| RD6/PSP6 | 16 | TRISC Register | 93, 167 |
| RD7/AD7 | 16 | PORTD | |
| RD7/PSP7 | 16 | Block Diagram | 95 |
| RE0/ALE | 17 | Initialization | 95 |
| RE0/RD | 17 | Parallel Slave Port (PSP) Function | 95 |
| RE1/OE | 17 | PORTD Register | 95 |
| RE1/WR | 17 | TRISD Register | 95 |
| RE2/CS | 17 | PORTE | |
| RE2/WRL | 17 | Block Diagram | 97 |
| RE3/WRH | 17 | Initialization | 97 |
| RE4 | 17 | PORTE Register | 97 |
| RE5 | 17 | PSP Mode Select (PSPMODE Bit) | 95, 109 |
| RE6 | 17 | RE0/RD | 109 |
| RE7/CCP2 | 17 | RE1/WR | 109 |
| RF0/AN5 | 18 | RE2/CS | 109 |
| RF1/AN6 | 18 | TRISE Register | 97 |
| RF2/AN7 | 18 | PORTF | |
| RF3/AN8 | 18 | Block Diagram | 99 |
| RF4/AN9 | 18 | Block Diagram of RF7 Pin | 100 |
| RF5/AN10 | 18 | C1OUT, C2OUT | 99 |
| RF6/AN11 | 18 | Initialization | 99 |
| RF7 | 18 | PORTF Register | 99 |
| RG0/CANTX1 | 19 | RF6/RF3 and RF0 Pins Block Diagram | 100 |
| RG1/CANTX2 | 19 | TRISF | 99 |
| RG2/CANRX | 19 | PORTG | |
| RG3 | 19 | Initialization | 101 |
| RG4 | 19 | PORTG | 101 |
| RH0/A16 | 19 | RG0/CANTX0 Pin Block Diagram | 101 |
| RH1/A17 | 19 | RG1/CANTX1 Pin Block Diagram | 102 |
| RH2/A18 | 19 | RG2 Pin Block Diagram | 102 |
| RH3/A19 | 19 | RG4/RG3 Pins Block Diagram | 102 |
| RH4/AN12 | 19 | TRISG | 101 |
| RH5/AN13 | 19 | PORTH | |
| RH6/AN14 | 19 | Initialization | 104 |
| RH7/AN15 | 19 | PORTH | 104 |
| RJ0/AD8 | 20 | RH3/RH0 Pins Block Diagram | 104 |
| RJ1/AD9 | 20 | RH7/RH4 Pins Block Diagram | 104 |
| RJ2/AD10 | 20 | TRISH | 104 |
| RJ3/AD11 | 20 | | |

| | | | |
|----------------------------------------|----------|---------------------------------------------------------|---------------|
| PORTJ | | Registers | |
| Initialization | 106 | SSPSTAT | 136 |
| PORTJ | 106 | T3CON | |
| TRISJ | 106 | Diagram | 123 |
| PORTJ Block Diagram | 106 | Section | 123 |
| PORTK | | RESET | 29, 251 |
| Initialization | 108 | Timing Diagram | 325 |
| PORTK | 108 | Resynchronization | 220 |
| TRISK | 108 | RETFIE | 291, 292 |
| Postscaler, WDT | | RETLW | 292 |
| Assignment (PSA Bit) | 115 | RETURN | 293 |
| Rate Select (PS2:PS0 Bits) | 115 | Revision History | 349 |
| Switching Between Timer0 and WDT | 115 | RLCF | 293 |
| Power-down Mode. See SLEEP | | RLNCF | 294 |
| Power-on Reset (POR) | 30, 251 | RRCF | 294 |
| Oscillator Start-up Timer (OST) | 30, 251 | RRNCF | 295 |
| Power-up Timer (PWRT) | 30, 251 | | |
| Time-out Sequence | 31 | S | |
| Time-out Sequence on Power-up | 32, 33 | Sample Point | 219 |
| Timing Diagram | 325 | SCI. See USART | |
| Prescaler, Capture | 129 | SCK | 141 |
| Prescaler, Timer0 | 115 | SDI | 141 |
| Assignment (PSA Bit) | 115 | SDO | 141 |
| Rate Select (PS2:PS0 Bits) | 115 | Serial Clock, SCK | 141 |
| Switching Between Timer0 and WDT | 115 | Serial Communication Interface. See USART | |
| Prescaler, Timer1 | 118 | Serial Data In, SDI | 141 |
| Prescaler, Timer2 | 132 | Serial Data Out, SDO | 141 |
| PRO MAT™ II Universal Programmer | 307 | Serial Peripheral Interface. See SPI | |
| Program Counter | | Shortening a Bit Period | 221 |
| PCL Register | 45 | Simplified Block Diagram of On-Chip Reset Circuit | 29 |
| PCLATH Register | 45 | Slave Select Synchronization | 144 |
| Program Memory | 41 | Slave Select, \overline{SS} | 141 |
| Program Verification | 259 | SLEEP | 251, 257, 296 |
| Programmable | 251 | Software Simulator (MPLAB-SIM) | 306 |
| Programming Time Segments | 222 | Special Event Trigger. See Compare | |
| Programming, Device Instructions | 261 | Special Features of the CPU | 247, 251 |
| Propagation Segment | 219 | Special Function Registers | 48 |
| PSPCON Register | | SPI | |
| PSPMODE Bit | 95, 109 | Master Mode | 143 |
| PWM (CCP Module) | 132 | Serial Clock | 141 |
| Block Diagram | 132 | Serial Data In | 141 |
| CCPR1H:CCPR1L Registers | 132 | Serial Data Out | 141 |
| Duty Cycle | 132 | Slave Select | 141 |
| Example Frequencies/Resolutions | 133 | SPI Clock | 143 |
| Output Diagram | 132 | SPI Mode | 141 |
| Period | 132 | SPI Module | |
| Setup for PWM Operation | 133 | Slave Mode | 144 |
| TMR2 to PR2 Match | 121, 132 | Slave Select Synchronization | 144 |
| | | Slave Synch Timing | 144 |
| Q | | Slave Timing with CKE = 0 | 145 |
| Q-Clock | 132 | Slave Timing with CKE = 1 | 145 |
| | | \overline{SS} | 141 |
| R | | SSP | 135 |
| RAM. See Data Memory | | Block Diagram (SPI Mode) | 141 |
| RCSTA Register | | I ² C Mode. See I ² C | |
| SPEN Bit | 167 | SPI Mode | 141 |
| Receive Buffers | 213 | SPI Mode. See SPI | |
| Receive Buffers Diagram | 214 | SSPBUF | 143 |
| Receive Interrupt | 225 | SSPCON1 | 138 |
| Receive Message Buffering | 213 | SSPCON2 | 140 |
| Receiver Error Passive | 226 | SSPSR | 143 |
| Receiver Overrun | 226 | SSPSTAT | 136 |
| Receiver Warning | 226 | TMR2 Output for Clock Shift | 121, 122 |
| Register File | 48 | SSP Module | |
| | | SPI Master Mode | 143 |
| | | SPI Slave Mode | 144 |
| | | SSPCON1 | 138 |
| | | SSPCON2 | 140 |

PIC18CXX8

| | | | |
|-----------------------------------------------------------|---------------|------------------------------------------------------|-----|
| SSPOV | 156 | Master Mode Transmit Clock Arbitration | 161 |
| SSPSTAT | 136 | Repeat Start Condition | 155 |
| SSPSTAT Register | | Slave Synchronization | 144 |
| R/W Bit | 148, 149 | Slow Rise Time | 33 |
| Stuff Error | 223 | SPI Mode Timing (Master Mode) SPI Mode | |
| SUBLW | 297 | Master Mode Timing Diagram | 143 |
| SUBWF | 297, 298 | SPI Mode Timing (Slave Mode with CKE = 0) | 145 |
| SUBWFB | 299 | SPI Mode Timing (Slave Mode with CKE = 1) | 145 |
| SWAPF | 300 | Stop Condition Receive or Transmit | 160 |
| Synchronization | 220 | Time-out Sequence on Power-up | 32 |
| Synchronization Rules | 220 | USART Asynchronous Master Transmission | 174 |
| Synchronization Segment | 219 | USART Asynchronous Reception | 176 |
| Synchronous Serial Port. See SSP | | USART Synchronous Reception | 179 |
| T | | USART Synchronous Transmission | 178 |
| TABLRD | 301 | Wake-up from SLEEP via Interrupt | 258 |
| TABLWT | 302 | Timing Diagrams and Specifications | 322 |
| Time Quanta | 219 | A/D Conversion | 340 |
| Timer Modules | | Brown-out Reset (BOR) | 325 |
| Timer3 | | Capture/Compare/PWM (CCP) | 327 |
| Block Diagram | 124 | CLKOUT and I/O | 324 |
| Timer0 | 113 | External Clock | 322 |
| Clock Source Edge Select (T0SE Bit) | 115 | I ² C Bus Data | 334 |
| Clock Source Select (T0CS Bit) | 115 | I ² C Bus START/STOP Bits | 333 |
| Overflow Interrupt | 116 | Oscillator Start-up Timer (OST) | 325 |
| Prescaler. See Prescaler, Timer0 | | Parallel Slave Port (PSP) | 328 |
| Timing Diagram | 326 | Power-up Timer (PWRT) | 325 |
| Timer1 | 117 | Reset | 325 |
| Block Diagram | 118 | Timer0 and Timer1 | 326 |
| Oscillator | 117, 119 | USART Synchronous Receive (Master/Slave) | 338 |
| Overflow Interrupt | 117, 119 | USART Synchronous Transmission (Master/Slave) | 337 |
| Prescaler. See Prescaler, Timer1 | | Watchdog Timer (WDT) | 325 |
| Special Event Trigger (CCP) | 119, 130 | Transmit Interrupt | 225 |
| Timing Diagram | 326 | Transmit Message Aborting | 211 |
| TMR1H Register | 117 | Transmit Message Buffering | 211 |
| TMR1L Register | 117 | Transmit Message Buffers | 211 |
| TMR3L Register | 123 | Transmit Message flowchart | 212 |
| Timer2 | | Transmit Message Priority | 211 |
| Block Diagram | 122 | Transmitter Error Passive | 226 |
| Postscaler. See Postscaler, Timer2 | | Transmitter Warning | 226 |
| PR2 Register | 121, 132 | TRISE Register | 97 |
| Prescaler. See Prescaler, Timer2 | | TSTFSZ | 303 |
| SSP Clock Shift | 121, 122 | TXSTA Register | |
| TMR2 Register | 121 | BRGH Bit | 169 |
| TMR2 to PR2 Match Interrupt | 121, 122, 132 | U | |
| Timer3 | 123 | Universal Synchronous Asynchronous Receiver | |
| Oscillator | 123, 125 | Transmitter. See USART | |
| Overflow Interrupt | 123, 125 | USART | 167 |
| Special Event Trigger (CCP) | 125 | Asynchronous Mode | 173 |
| TMR3H Register | 123 | Master Transmission | 174 |
| Timing Diagrams | | Receive Block Diagram | 175 |
| Acknowledge Sequence Timing | 159 | Reception | 176 |
| Baud Rate Generator with Clock Arbitration | 153 | Transmit Block Diagram | 173 |
| BRG Reset Due to SDA Collision | 164 | Baud Rate Generator (BRG) | 169 |
| Bus Collision | | Baud Rate Error, Calculating | 169 |
| START Condition Timing | 163 | Baud Rate Formula | 169 |
| Bus Collision During a RESTART Condition | | High Baud Rate Select (BRGH Bit) | 169 |
| (Case 1) | 165 | Sampling | 169 |
| Bus Collision During a RESTART Condition | | Serial Port Enable (SPEN Bit) | 167 |
| (Case2) | 165 | Synchronous Master Mode | 177 |
| Bus Collision During a START Condition (SCL = 0) | 164 | Reception | 179 |
| Bus Collision During a STOP Condition | 166 | Timing Diagram, Synchronous Receive | 338 |
| Bus Collision for Transmit and Acknowledge | 162 | Timing Diagram, Synchronous Transmission | 337 |
| I ² C Bus Data | 336 | Transmission | 178 |
| I ² C Master Mode First Start bit timing | 154 | Synchronous Slave Mode | 180 |
| I ² C Master Mode Reception timing | 158 | | |
| I ² C Master Mode Transmission timing | 157 | | |

W

| | |
|--------------------------------------------------|---------------|
| Wake-up from SLEEP | 251, 257 |
| Timing Diagram | 258 |
| Watchdog Timer (WDT) | 251, 255 |
| Block Diagram | 256 |
| Postscaler. See Postscaler, WDT | |
| Programming Considerations | 255 |
| RC Oscillator | 255 |
| Time-out Period | 255 |
| Timing Diagram | 325 |
| Waveform for General Call Address Sequence | 150 |
| WCOL | 154, 156, 159 |
| WCOL Status Flag | 154 |
| WWW, On-Line Support | 7 |

X

| | |
|-------------|-----|
| XORLW | 303 |
| XORWF | 304 |

PIC18CXX8

NOTES:

ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web (WWW) site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape or Microsoft Explorer. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Web Site

The Microchip web site is available by using your favorite Internet browser to attach to:

www.microchip.com

The file transfer site is available by using an FTP service to connect to:

<ftp://ftp.microchip.com>

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

Systems Information and Upgrade Hot Line

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits. The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.

001024

Trademarks: The Microchip name, logo, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KEELoQ, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. Total Endurance, ICSP, In-Circuit Serial Programming, FilterLab, MXDEV, microID, *Flex*ROM, *fuzzy*LAB, MPASM, MPLINK, MPLIB, PICDEM, ICEPIC and Migratable Memory are trademarks and SQTP is a service mark of Microchip in the U.S.A.

All other trademarks mentioned herein are the property of their respective companies.

PIC18CXX8

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 Data Sheet.

To: Technical Publications Manager Total Pages Sent
RE: Reader Response
From: Name _____
Company _____
Address _____
City / State / ZIP / Country _____
Telephone: (____) _____ - _____ FAX: (____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: **PIC18CXX8** Literature Number: **DS30475A**

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 data sheet easy to follow? If not, why?

4. What additions to the data sheet do you think would enhance the structure and subject?

5. What deletions from the data sheet 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?

8. How would you improve our software, systems, and silicon products?

PIC18CXX8 PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery refer to the factory or the listed sales office

| <u>PART NO.</u> | <u>X</u> | <u>/XX</u> | <u>XXX</u> |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------|
| Device | Temperature Range | Package | Pattern |
| Device | PIC18CXX8 ⁽¹⁾ , PIC18CXX8T ⁽²⁾ , VDD range 4.2V to 5.5V PIC18LCXX5 ⁽¹⁾ , PIC18LCXX8T ⁽²⁾ , VDD range 2.5V to 5.5V | | |
| Temperature Range | I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended) | | |
| Package | CL = Windowed JCERPACK PT = TQFP (Thin Quad Flatpack) L = PLCC | | |
| Pattern | QTP, SQTP, Code or Special Requirements (blank otherwise) | | |

Examples:

- a) PIC18LC658 - I/L 301 = Industrial temp., PLCC package, Extended VDD limits, QTP pattern #301.
- b) PIC18LC858 - I/PT = Industrial temp., TQFP package, Extended VDD limits.
- c) PIC18C658 - E/L = Extended temp., PLCC package, normal VDD limits.

Note 1: C = Standard Voltage Range
LC = Wide Voltage Range

2: T = in tape and reel PLCC, and TQFP packages only.

3: CL devices are UV erasable and can be programmed to any device configuration. CL devices meet the electrical requirement of each oscillator type (including LC devices).

* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type.

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 Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System

Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

PIC18CXX8

NOTES:

NOTES:



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: 480-792-7627
Web Address: <http://www.microchip.com>

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta

500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3838 Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Dayton

Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

China - Beijing

Microchip Technology Beijing Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Shanghai

Microchip Technology Shanghai Office
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

Hong Kong

Microchip Asia Pacific
RM 2101, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India

Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471-6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

ASIA/PACIFIC (continued)

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan

Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark

Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

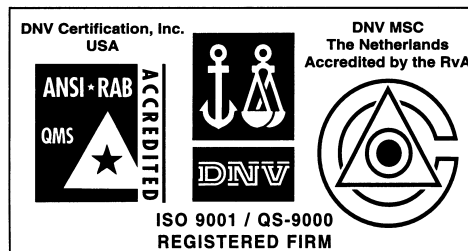
Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

10/01/00



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELoC® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

All rights reserved. © 2000 Microchip Technology Incorporated. Printed in the USA. 11/00 Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, except as maybe explicitly expressed herein, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.



Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits, General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)

Факс: 8 (812) 320-02-42

Электронная почта: org@eplast1.ru

Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.