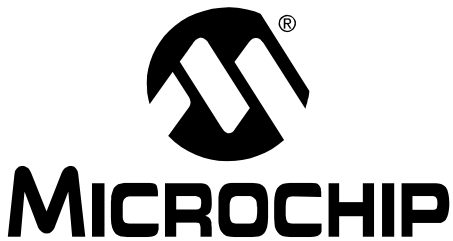
chanhvinh.com



# PIC18FXX8
# Data Sheet

## 28/40-Pin High-Performance, Enhanced Flash Microcontrollers with CAN Module

© 2006 Microchip Technology Inc.

chanhvinh.com

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

**Trademarks**

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Linear Active Thermistor, Mindi, MiWi, MPASM, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2006, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*


© 2006 Microchip Technology Inc.

chanhvinh.com

# PIC18FXX8

## 28/40-Pin High-Performance, Enhanced Flash Microcontrollers with CAN

### High-Performance RISC CPU:

- Linear program memory addressing up to 2 Mbytes
- Linear data memory addressing to 4 Kbytes
- Up to 10 MIPS operation
- DC – 40 MHz clock input
- 4 MHz-10 MHz oscillator/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
- Three 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
- Capture/Compare/PWM (CCP) modules; CCP pins can be configured as:
  - Capture input: 16-bit, max resolution 6.25 ns
  - Compare: 16-bit, max resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
    Max. PWM freq. @:8-bit resolution = 156 kHz
    10-bit resolution = 39 kHz
- Enhanced CCP module which has all the features of the standard CCP module, but also has the following features for advanced motor control:
  - 1, 2 or 4 PWM outputs
  - Selectable PWM polarity
  - Programmable PWM dead time
- Master Synchronous Serial Port (MSSP) with two modes of operation:
  - 3-wire SPI™ (Supports all 4 SPI modes)
  - $I^2C$™ Master and Slave mode
- Addressable USART module:
  - Supports interrupt-on-address bit

### Advanced Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter module (A/D) with:
  - Conversion available during Sleep
  - Up to 8 channels available
- Analog Comparator module:
  - 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:

- Complies with ISO CAN Conformance Test
- 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 (PLL) of primary oscillator
  - Secondary Oscillator (32 kHz) clock input
- In-Circuit Serial Programming™ (ICSP™) via two pins

### Flash Technology:

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

© 2006 Microchip Technology Inc.

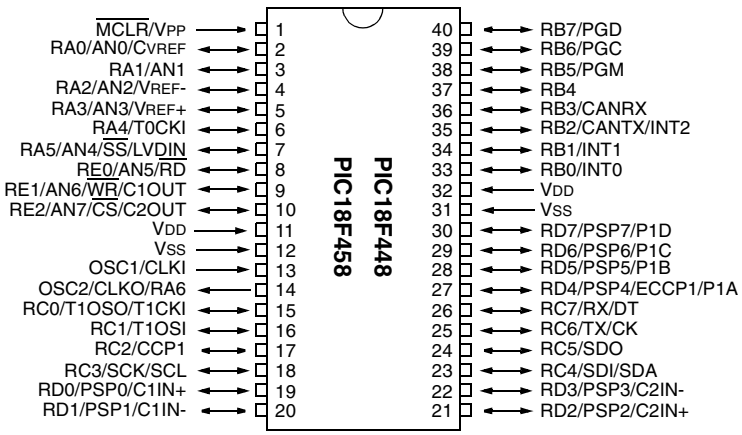
chanhvinh.com

# PIC18FXX8

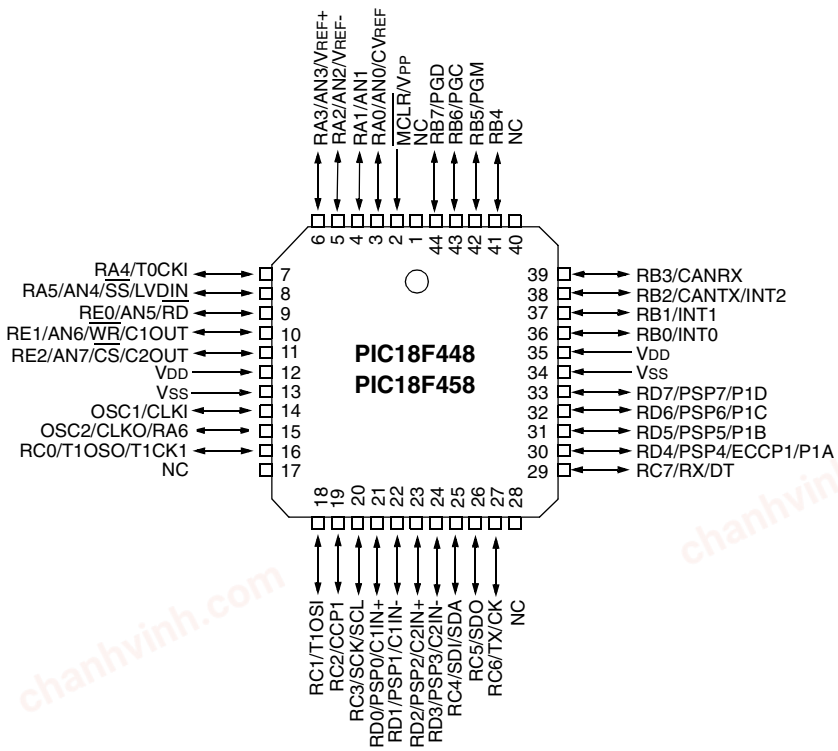
| Device | Program Memory | | Data Memory | | I/O | 10-bit A/D (ch) | Comparators | CCP/ ECCP (PWM) | MSSP | | USART | Timers 8/16-bit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | | SPI™ | Master $I^2C$™ | | |
| PIC18F248 | 16K | 8192 | 768 | 256 | 22 | 5 | — | 1/0 | Y | Y | Y | 1/3 |
| PIC18F258 | 32K | 16384 | 1536 | 256 | 22 | 5 | — | 1/0 | Y | Y | Y | 1/3 |
| PIC18F448 | 16K | 8192 | 768 | 256 | 33 | 8 | 2 | 1/1 | Y | Y | Y | 1/3 |
| PIC18F458 | 32K | 16384 | 1536 | 256 | 33 | 8 | 2 | 1/1 | Y | Y | Y | 1/3 |

## Pin Diagrams

### PDIP




### PLCC





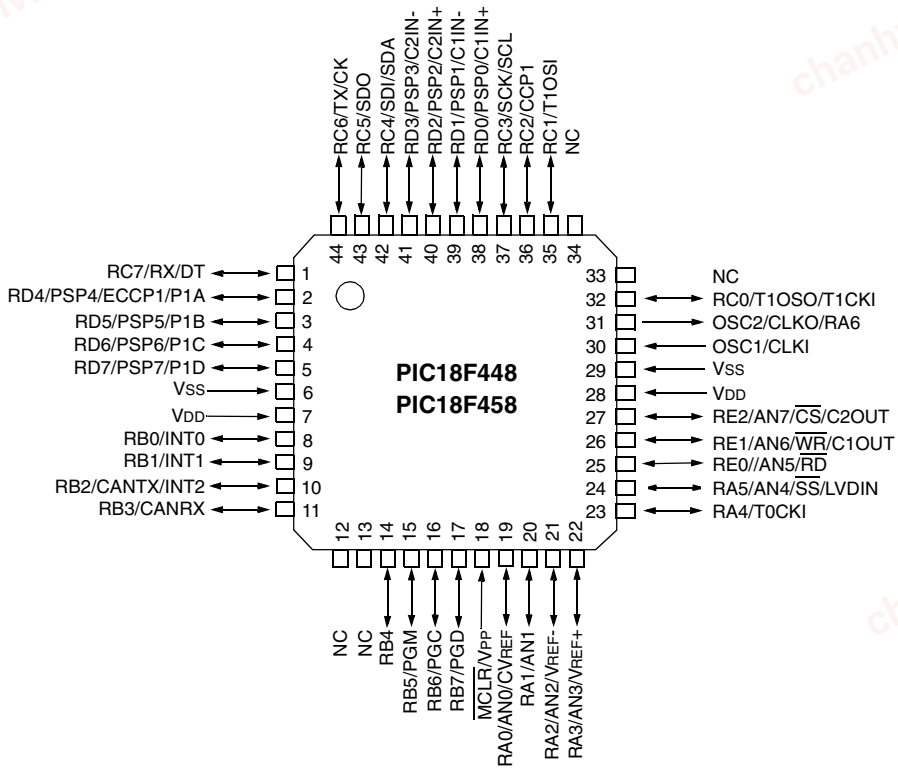
© 2006 Microchip Technology Inc.

chanhvinh.com

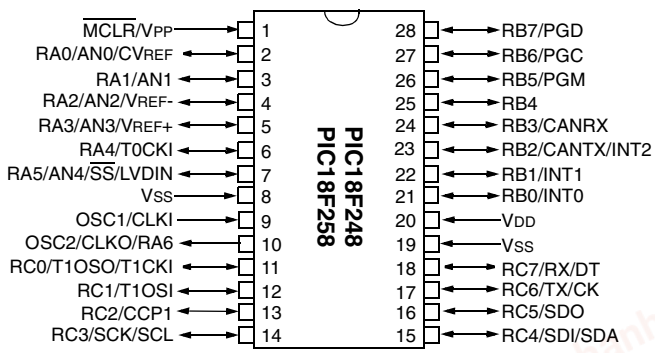


## Pin Diagrams (Continued)

### TQFP




### SPDIP, SOIC



chanhvinh.com

# PIC18FXX8

## Table of Contents


1.0 Device Overview ..... 7
2.0 Oscillator Configurations ..... 17
3.0 Reset ..... 25
4.0 Memory Organization ..... 37
5.0 Data EEPROM Memory ..... 59
6.0 Flash Program Memory ..... 65
7.0 8 x 8 Hardware Multiplier ..... 75
8.0 Interrupts ..... 77
9.0 I/O Ports ..... 93
10.0 Parallel Slave Port ..... 107
11.0 Timer0 Module ..... 109
12.0 Timer1 Module ..... 113
13.0 Timer2 Module ..... 117
14.0 Timer3 Module ..... 119
15.0 Capture/Compare/PWM (CCP) Modules ..... 123
16.0 Enhanced Capture/Compare/PWM (ECCP) Module ..... 131
17.0 Master Synchronous Serial Port (MSSP) Module ..... 143
18.0 Addressable Universal Synchronous Asynchronous Receiver Transmitter (USART) ..... 183
19.0 CAN Module ..... 199
20.0 Compatible 10-Bit Analog-to-Digital Converter (A/D) Module ..... 241
21.0 Comparator Module ..... 249
22.0 Comparator Voltage Reference Module ..... 255
23.0 Low-Voltage Detect ..... 259
24.0 Special Features of the CPU ..... 265
25.0 Instruction Set Summary ..... 281
26.0 Development Support ..... 323
27.0 Electrical Characteristics ..... 329
28.0 DC and AC Characteristics Graphs and Tables ..... 361
29.0 Packaging Information ..... 377
Appendix A: Data Sheet Revision History ..... 385
Appendix B: Device Differences ..... 385
Appendix C: Device Migrations ..... 386
Appendix D: Migrating From Other PICmicro® Devices ..... 386
Index ..... 387
On-Line Support ..... 397
Systems Information and Upgrade Hot Line ..... 397
Reader Response ..... 398
PIC18FXX8 Product Identification System ..... 399



© 2006 Microchip Technology Inc.

chanhvinh.com



## TO OUR VALUED CUSTOMERS

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

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

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

### Customer Notification System

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

chanhvinh.com

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F248
- PIC18F258
- PIC18F448
- PIC18F458

These devices are available in 28-pin, 40-pin and 44-pin packages. They are differentiated from each other in four ways:

1. PIC18FX58 devices have twice the Flash program memory and data RAM of PIC18FX48 devices (32 Kbytes and 1536 bytes vs. 16 Kbytes and 768 bytes, respectively).
2. PIC18F2X8 devices implement 5 A/D channels, as opposed to 8 for PIC18F4X8 devices.
3. PIC18F2X8 devices implement 3 I/O ports, while PIC18F4X8 devices implement 5.
4. Only PIC18F4X8 devices implement the Enhanced CCP module, analog comparators and the Parallel Slave Port.

All other features for devices in the PIC18FXX8 family, including the serial communications modules, are identical. These are summarized in Table 1-1.

Block diagrams of the PIC18F2X8 and PIC18F4X8 devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2.

**TABLE 1-1: PIC18FXX8 DEVICE FEATURES**

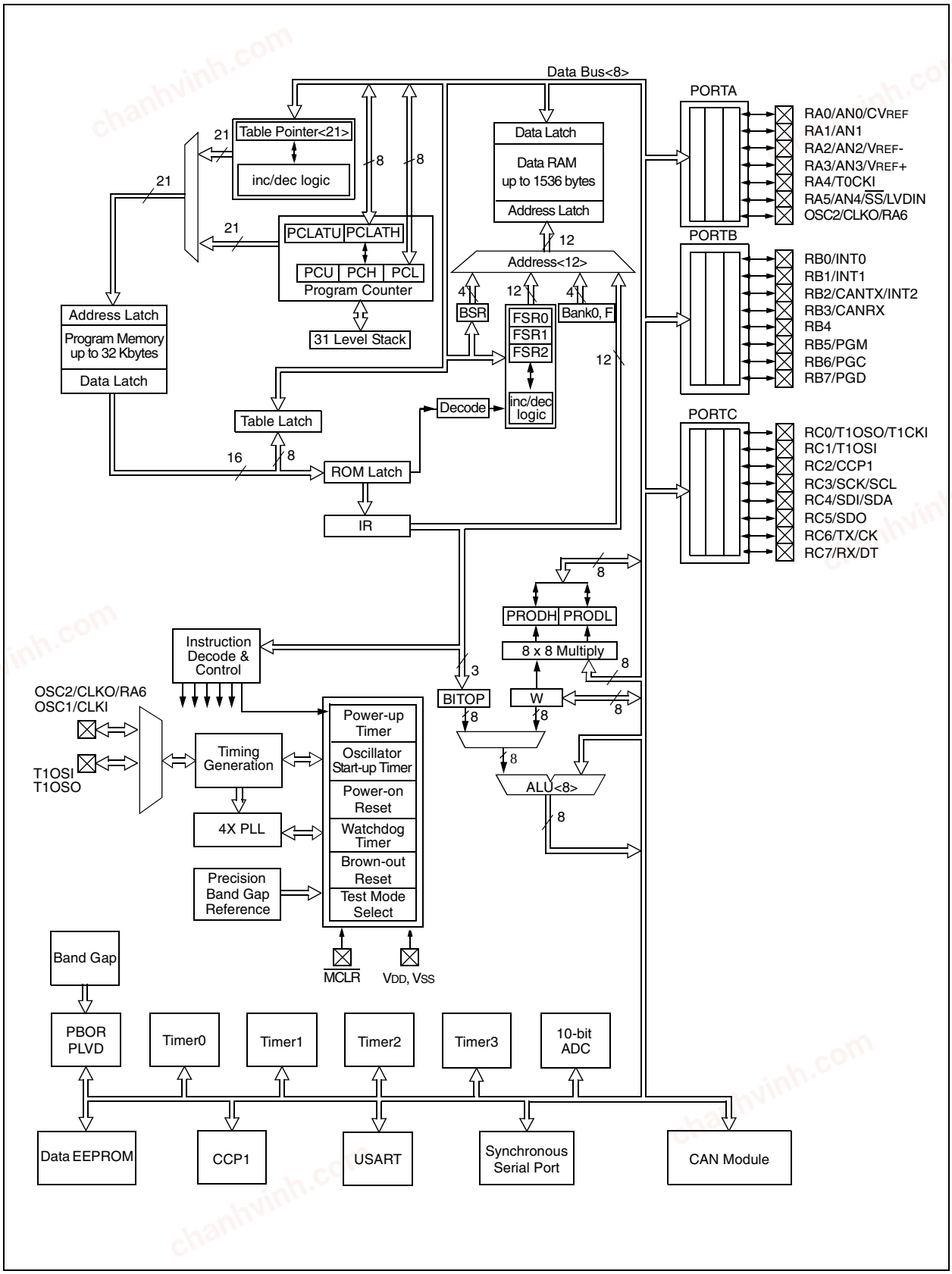
| Features | | PIC18F248 | PIC18F258 | PIC18F448 | PIC18F458 |
|---|---|---|---|---|---|
| Operating Frequency | | DC – 40 MHz | DC – 40 MHz | DC – 40 MHz | DC – 40 MHz |
| Internal Program Memory | Bytes | 16K | 32K | 16K | 32K |
| | # of Single-Word Instructions | 8192 | 16384 | 8192 | 16384 |
| Data Memory (Bytes) | | 768 | 1536 | 768 | 1536 |
| Data EEPROM Memory (Bytes) | | 256 | 256 | 256 | 256 |
| Interrupt Sources | | 17 | 17 | 21 | 21 |
| I/O Ports | | Ports A, B, C | Ports A, B, C | Ports A, B, C, D, E | Ports A, B, C, D, E |
| Timers | | 4 | 4 | 4 | 4 |
| Capture/Compare/PWM Modules | | 1 | 1 | 1 | 1 |
| Enhanced Capture/Compare/ PWM Modules | | — | — | 1 | 1 |
| Serial Communications | | MSSP, CAN, Addressable USART | MSSP, CAN, Addressable USART | MSSP, CAN, Addressable USART | MSSP, CAN, Addressable USART |
| Parallel Communications (PSP) | | No | No | Yes | Yes |
| 10-bit Analog-to-Digital Converter | | 5 input channels | 5 input channels | 8 input channels | 8 input channels |
| Analog Comparators | | No | No | 2 | 2 |
| Analog Comparators VREF Output | | N/A | N/A | Yes | Yes |
| Resets (and Delays) | | POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST) | POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST) | 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 | Yes | Yes |
| Programmable Brown-out Reset | | Yes | Yes | Yes | Yes |
| CAN Module | | Yes | Yes | Yes | Yes |
| In-Circuit Serial Programming™ (ICSP™) | | Yes | Yes | Yes | Yes |
| Instruction Set | | 75 Instructions | 75 Instructions | 75 Instructions | 75 Instructions |
| Packages | | 28-pin SPDIP 28-pin SOIC | 28-pin SPDIP 28-pin SOIC | 40-pin PDIP 44-pin PLCC 44-pin TQFP | 40-pin PDIP 44-pin PLCC 44-pin TQFP |

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 1-1: PIC18F248/258 BLOCK DIAGRAM**



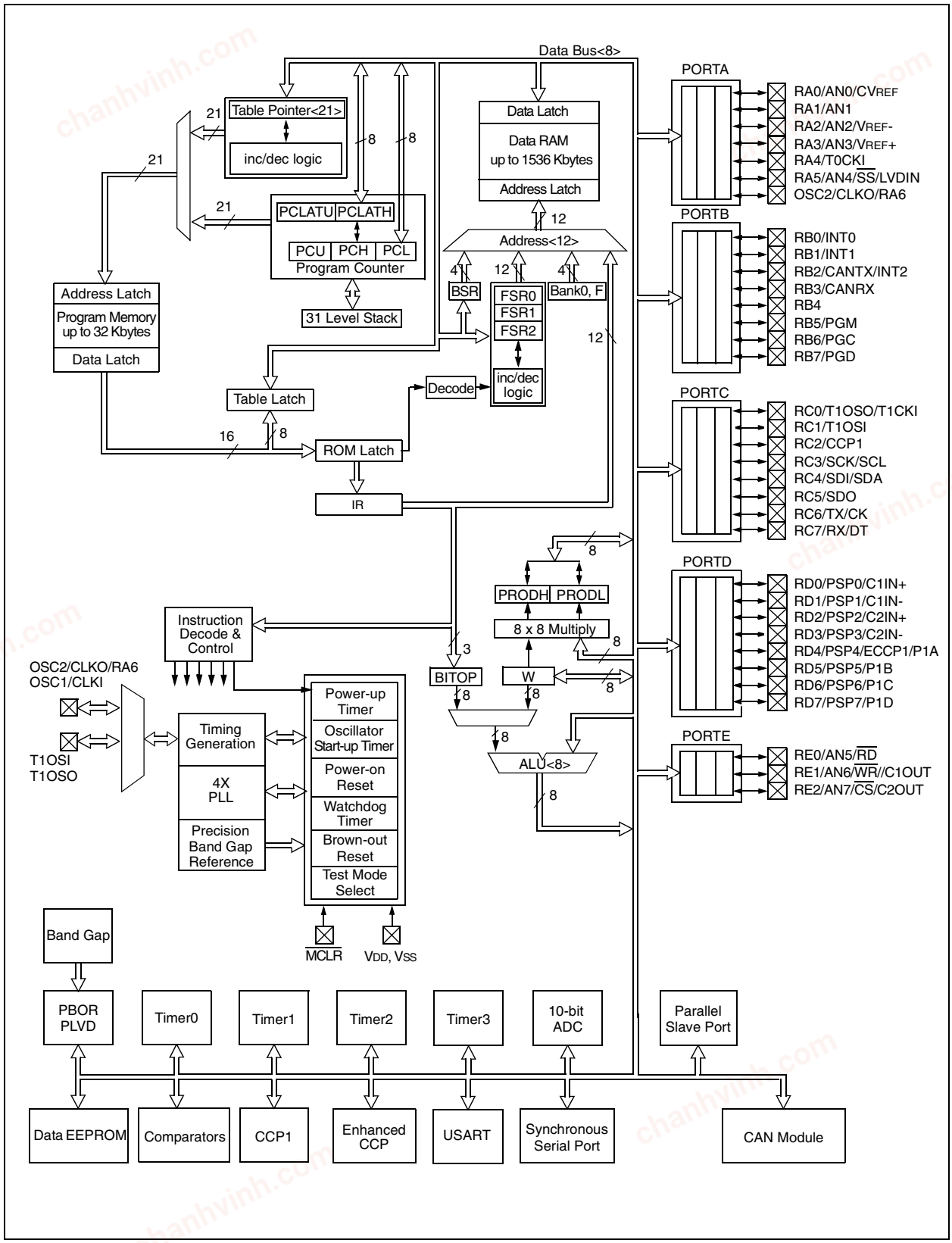


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 1-2: PIC18F448/458 BLOCK DIAGRAM**



chanhvinh.com



**TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS**

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|
| | PIC18F248/258 | PIC18F448/458 | | | | | |
| | SPDIP, SOIC | PDIP | TQFP | PLCC | | | |
| $\overline{MCLR}$/VPP | 1 | 1 | 18 | 2 | | | Master Clear (input) or programming voltage (output). |
| $\overline{MCLR}$ | | | | | I | ST | Master Clear (Reset) input. This pin is an active low Reset to the device. |
| VPP | | | | | P | — | Programming voltage input. |
| NC | — | — | 12, 13, 33, 34 | 1, 17, 28, 40 | — | — | These pins should be left unconnected. |
| OSC1/CLKI | 9 | 13 | 30 | 14 | | | Oscillator crystal or external clock input. |
| OSC1 | | | | | I | CMOS/ST | Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise, CMOS. |
| CLKI | | | | | I | CMOS | External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins). |
| OSC2/CLKO/RA6 | 10 | 14 | 31 | 15 | | | Oscillator crystal or clock output. |
| OSC2 | | | | | O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | | | | O | — | In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. |
| RA6 | | | | | I/O | TTL | General purpose I/O pin. |

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



© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|
| | PIC18F248/258 | PIC18F448/458 | | | | | |
| | SPDIP, SOIC | PDIP | TQFP | PLCC | | | |
| | | | | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0/CVREF | 2 | 2 | 19 | 3 | | | |
| RA0 | | | | | I/O | TTL | Digital I/O. |
| AN0 | | | | | I | Analog | Analog input 0. |
| CVREF | | | | | O | Analog | Comparator voltage reference output. |
| RA1/AN1 | 3 | 3 | 20 | 4 | | | |
| RA1 | | | | | I/O | TTL | Digital I/O. |
| AN1 | | | | | I | Analog | Analog input 1. |
| RA2/AN2/VREF- | 4 | 4 | 21 | 5 | | | |
| RA2 | | | | | I/O | TTL | Digital I/O. |
| AN2 | | | | | I | Analog | Analog input 2. |
| VREF- | | | | | I | Analog | A/D reference voltage (Low) input. |
| RA3/AN3/VREF+ | 5 | 5 | 22 | 6 | | | |
| RA3 | | | | | I/O | TTL | Digital I/O. |
| AN3 | | | | | I | Analog | Analog input 3. |
| VREF+ | | | | | I | Analog | A/D reference voltage (High) input. |
| RA4/T0CKI | 6 | 6 | 23 | 7 | | | |
| RA4 | | | | | I/O | TTL/OD | Digital I/O – open-drain when configured as output. |
| T0CKI | | | | | I | ST | Timer0 external clock input. |
| RA5/AN4/$\overline{SS}$/LVDIN | 7 | 7 | 24 | 8 | | | |
| RA5 | | | | | I/O | TTL | Digital I/O. |
| AN4 | | | | | I | Analog | Analog input 4. |
| $\overline{SS}$ | | | | | I | ST | SPI™ slave select input. |
| LVDIN | | | | | I | Analog | Low-Voltage Detect input. |
| 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)

chanhvinh.com



**TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|
| | PIC18F248/258 | PIC18F448/458 | | | | | |
| | SPDIP, SOIC | PDIP | TQFP | PLCC | | | |
| | | | | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/INT0 | 21 | 33 | 8 | 36 | | | |
| RB0 | | | | | I/O | TTL | Digital I/O. |
| INT0 | | | | | I | ST | External interrupt 0. |
| RB1/INT1 | 22 | 34 | 9 | 37 | | | |
| RB1 | | | | | I/O | TTL | Digital I/O. |
| INT1 | | | | | I | ST | External interrupt 1. |
| RB2/CANTX/INT2 | 23 | 35 | 10 | 38 | | | |
| RB2 | | | | | I/O | TTL | Digital I/O. |
| CANTX | | | | | O | TTL | Transmit signal for CAN bus. |
| INT2 | | | | | I | ST | External interrupt 2. |
| RB3/CANRX | 24 | 36 | 11 | 39 | | | |
| RB3 | | | | | I/O | TTL | Digital I/O. |
| CANRX | | | | | I | TTL | Receive signal for CAN bus. |
| RB4 | 25 | 37 | 14 | 41 | I/O | TTL | Digital I/O. Interrupt-on-change pin. |
| RB5/PGM | 26 | 38 | 15 | 42 | | | |
| RB5 | | | | | I/O | TTL | Digital I/O. Interrupt-on-change pin. |
| PGM | | | | | I | ST | Low-voltage ICSP™ programming enable. |
| RB6/PGC | 27 | 39 | 16 | 43 | | | |
| RB6 | | | | | I/O | TTL | Digital I/O. In-Circuit Debugger pin. Interrupt-on-change pin. |
| PGC | | | | | I | ST | ICSP programming clock. |
| RB7/PGD | 28 | 40 | 17 | 44 | | | |
| RB7 | | | | | I/O | TTL | Digital I/O. In-Circuit Debugger pin. Interrupt-on-change pin. |
| PGD | | | | | I/O | ST | ICSP programming data. |

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



© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|
| | PIC18F248/258 | PIC18F448/458 | | | | | |
| | SPDIP, SOIC | PDIP | TQFP | PLCC | | | |
| | | | | | | | PORTC is a bidirectional I/O port. |
| RC0/T1OSO/T1CKI | 11 | 15 | 32 | 16 | | | |
| RC0 | | | | | I/O | ST | Digital I/O. |
| T1OSO | | | | | O | — | Timer1 oscillator output. |
| T1CKI | | | | | I | ST | Timer1/Timer3 external clock input. |
| RC1/T1OSI | 12 | 16 | 35 | 18 | | | |
| RC1 | | | | | I/O | ST | Digital I/O. |
| T1OSI | | | | | I | CMOS | Timer1 oscillator input. |
| RC2/CCP1 | 13 | 17 | 36 | 19 | | | |
| RC2 | | | | | I/O | ST | Digital I/O. |
| CCP1 | | | | | I/O | ST | Capture 1 input/Compare 1 output/PWM1 output. |
| RC3/SCK/SCL | 14 | 18 | 37 | 20 | | | |
| RC3 | | | | | I/O | ST | Digital I/O. |
| SCK | | | | | I/O | ST | Synchronous serial clock input/output for SPI™ mode. |
| SCL | | | | | I/O | ST | Synchronous serial clock input/output for $I^2C$™ mode. |
| RC4/SDI/SDA | 15 | 23 | 42 | 25 | | | |
| RC4 | | | | | I/O | ST | Digital I/O. |
| SDI | | | | | I | ST | SPI data in. |
| SDA | | | | | I/O | ST | $I^2C$ data I/O. |
| RC5/SDO | 16 | 24 | 43 | 26 | | | |
| RC5 | | | | | I/O | ST | Digital I/O. |
| SDO | | | | | O | — | SPI data out. |
| RC6/TX/CK | 17 | 25 | 44 | 27 | | | |
| RC6 | | | | | I/O | ST | Digital I/O. |
| TX | | | | | O | — | USART asynchronous transmit. |
| CK | | | | | I/O | ST | USART synchronous clock (see RX/DT). |
| RC7/RX/DT | 18 | 26 | 1 | 29 | | | |
| RC7 | | | | | I/O | ST | Digital I/O. |
| RX | | | | | I | ST | USART asynchronous receive. |
| DT | | | | | I/O | ST | USART synchronous data (see TX/CK). |

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

© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|
| | PIC18F248/258 | PIC18F448/458 | | | | | |
| | SPDIP, SOIC | PDIP | TQFP | PLCC | | | |
| | | | | | | | PORTD is a bidirectional I/O port. These pins have TTL input buffers when external memory is enabled. |
| RD0/PSP0/C1IN+ | — | 19 | 38 | 21 | | | |
| RD0 | | | | | I/O | ST | Digital I/O. |
| PSP0 | | | | | I/O | TTL | Parallel Slave Port data. |
| C1IN+ | | | | | I | Analog | Comparator 1 input. |
| RD1/PSP1/C1IN- | — | 20 | 39 | 22 | | | |
| RD1 | | | | | I/O | ST | Digital I/O. |
| PSP1 | | | | | I/O | TTL | Parallel Slave Port data. |
| C1IN- | | | | | I | Analog | Comparator 1 input. |
| RD2/PSP2/C2IN+ | — | 21 | 40 | 23 | | | |
| RD2 | | | | | I/O | ST | Digital I/O. |
| PSP2 | | | | | I/O | TTL | Parallel Slave Port data. |
| C2IN+ | | | | | I | Analog | Comparator 2 input. |
| RD3/PSP3/C2IN- | — | 22 | 41 | 24 | | | |
| RD3 | | | | | I/O | ST | Digital I/O. |
| PSP3 | | | | | I/O | TTL | Parallel Slave Port data. |
| C2IN- | | | | | I | Analog | Comparator 2 input. |
| RD4/PSP4/ECCP1/ P1A | — | 27 | 2 | 30 | | | |
| RD4 | | | | | I/O | ST | Digital I/O. |
| PSP4 | | | | | I/O | TTL | Parallel Slave Port data. |
| ECCP1 | | | | | I/O | ST | ECCP1 capture/compare. |
| P1A | | | | | O | — | ECCP1 PWM output A. |
| RD5/PSP5/P1B | — | 28 | 3 | 31 | | | |
| RD5 | | | | | I/O | ST | Digital I/O. |
| PSP5 | | | | | I/O | TTL | Parallel Slave Port data. |
| P1B | | | | | O | — | ECCP1 PWM output B. |
| RD6/PSP6/P1C | — | 29 | 4 | 32 | | | |
| RD6 | | | | | I/O | ST | Digital I/O. |
| PSP6 | | | | | I/O | TTL | Parallel Slave Port data. |
| P1C | | | | | O | — | ECCP1 PWM output C. |
| RD7/PSP7/P1D | — | 30 | 5 | 33 | | | |
| RD7 | | | | | I/O | ST | Digital I/O. |
| PSP7 | | | | | I/O | TTL | Parallel Slave Port data. |
| P1D | | | | | O | — | ECCP1 PWM output D. |

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



© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|
| | PIC18F248/258 | PIC18F448/458 | | | | | |
| | SPDIP, SOIC | PDIP | TQFP | PLCC | | | |
| | | | | | | | PORTE is a bidirectional I/O port. |
| RE0/AN5/$\overline{RD}$ | — | 8 | 25 | 9 | | | |
| RE0 | | | | | I/O | ST | Digital I/O. |
| AN5 | | | | | I | Analog | Analog input 5. |
| $\overline{RD}$ | | | | | I | TTL | Read control for Parallel Slave Port (see $\overline{WR}$ and $\overline{CS}$ pins). |
| RE1/AN6/$\overline{WR}$/C1OUT | — | 9 | 26 | 10 | | | |
| RE1 | | | | | I/O | ST | Digital I/O. |
| AN6 | | | | | I | Analog | Analog input 6. |
| $\overline{WR}$ | | | | | I | TTL | Write control for Parallel Slave Port (see $\overline{CS}$ and $\overline{RD}$ pins). |
| C1OUT | | | | | O | Analog | Comparator 1 output. |
| RE2/AN7/$\overline{CS}$/C2OUT | — | 10 | 27 | 11 | | | |
| RE2 | | | | | I/O | ST | Digital I/O. |
| AN7 | | | | | I | Analog | Analog input 7. |
| $\overline{CS}$ | | | | | I | TTL | Chip select control for Parallel Slave Port (see $\overline{RD}$ and $\overline{WR}$ pins). |
| C2OUT | | | | | O | Analog | Comparator 2 output. |
| VSS | 19, 8 | 12, 31 | 6, 29 | 13, 34 | — | — | Ground reference for logic and I/O pins. |
| VDD | 20 | 11, 32 | 7, 28 | 12, 35 | — | — | Positive supply for logic and I/O pins. |

**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)

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 2.0 OSCILLATOR CONFIGURATIONS

### 2.1 Oscillator Types

The PIC18FXX8 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

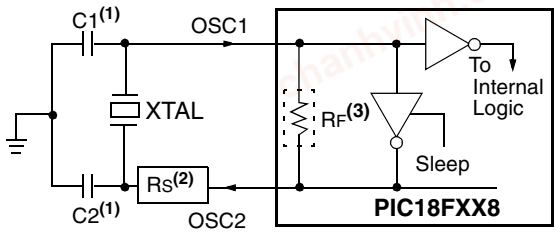
### 2.2 Crystal Oscillator/Ceramic Resonators

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 PIC18FXX8 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)**




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

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

**3:** RF varies with the crystal chosen.

**TABLE 2-1: CERAMIC RESONATORS**

| Ranges Tested: | | | |
|---|---|---|---|
| **Mode** | **Freq** | **OSC1** | **OSC2** |
| XT | 455 kHz<br>2.0 MHz<br>4.0 MHz | 68-100 pF<br>15-68 pF<br>15-68 pF | 68-100 pF<br>15-68 pF<br>15-68 pF |
| HS | 8.0 MHz<br>16.0 MHz | 10-68 pF<br>10-22 pF | 10-68 pF<br>10-22 pF |
| **These values are for design guidance only.** See notes following Table 2-2. | | | |
| **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. | | | |

chanhvinh.com



**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 | 15-33 pF | 15-33 pF |
| **These values are for design guidance only.** See notes on this page. | | | |
| **Crystals Used** | | | |
| 32.0 kHz | Epson C-001R32.768K-A | ±20 PPM | |
| 200 kHz | STD XTL 200.000KHz | ±20 PPM | |
| 1.0 MHz | ECS ECS-10-13-1 | ±50 PPM | |
| 4.0 MHz | ECS ECS-40-20-1 | ±50 PPM | |
| 8.0 MHz | EPSON CA-301 8.000M-C | ±30 PPM | |
| 20.0 MHz | EPSON CA-301 20.000M-C | ±30 PPM | |

**Note 1:** Recommended values of C1 and C2 are identical to the ranges tested (Table 2-1).

**2:** Higher capacitance increases the stability of the oscillator but also increases the start-up time.

**3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

**4:** Rs may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specification.

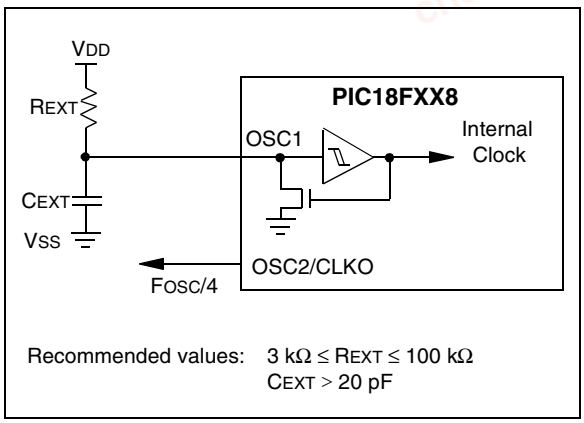
## 2.3 RC Oscillator

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

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

**Note:** If the oscillator frequency divided by 4 signal is not required in the application, it is recommended to use RCIO mode to save current.

**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).


© 2006 Microchip Technology Inc.

chanhvinh.com

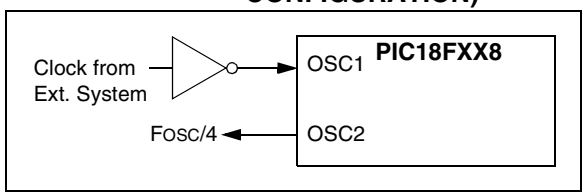


## 2.4 External Clock Input

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.

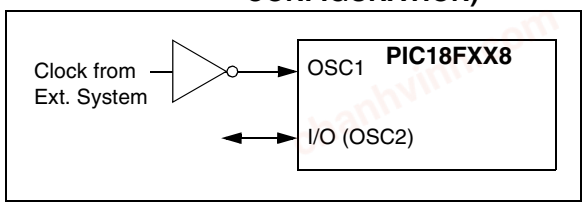
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC Oscillator mode.

**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. Figure 2-4 shows the pin connections for the ECIO Oscillator mode.

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




## 2.5 HS4 (PLL)

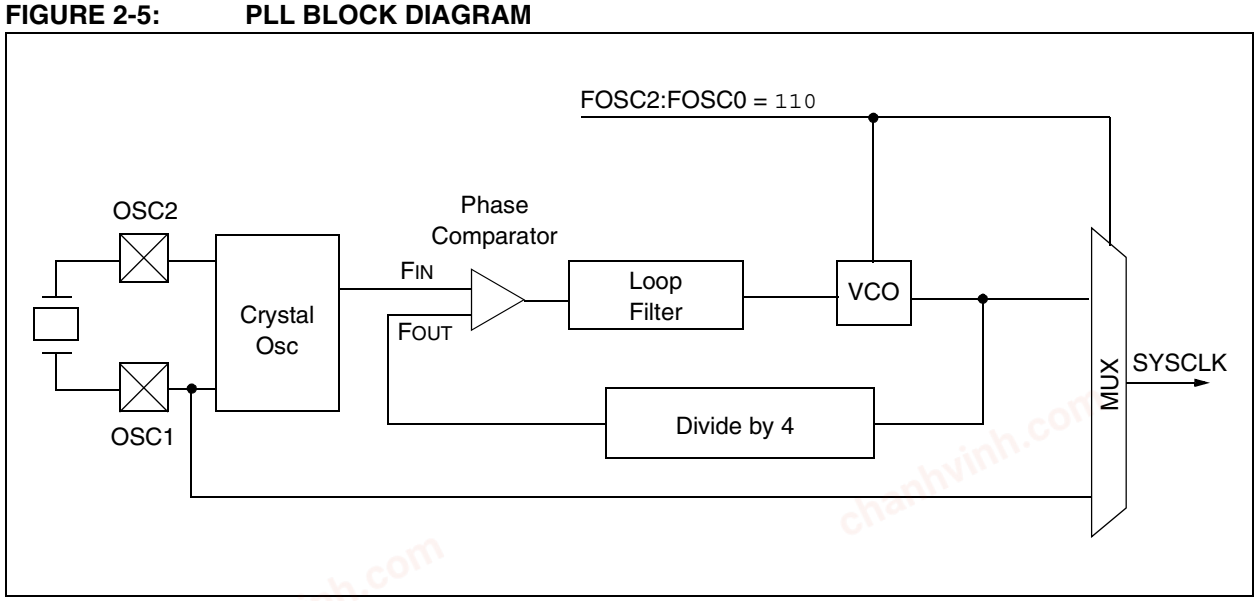
A Phase Locked Loop circuit is provided as a programmable option for users that want to multiply the frequency of the incoming crystal oscillator signal by 4. For an input clock frequency of 10 MHz, the internal clock frequency will be multiplied to 40 MHz. This is useful for customers who are concerned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



chanhvinh.com



## 2.6 Oscillator Switching Feature

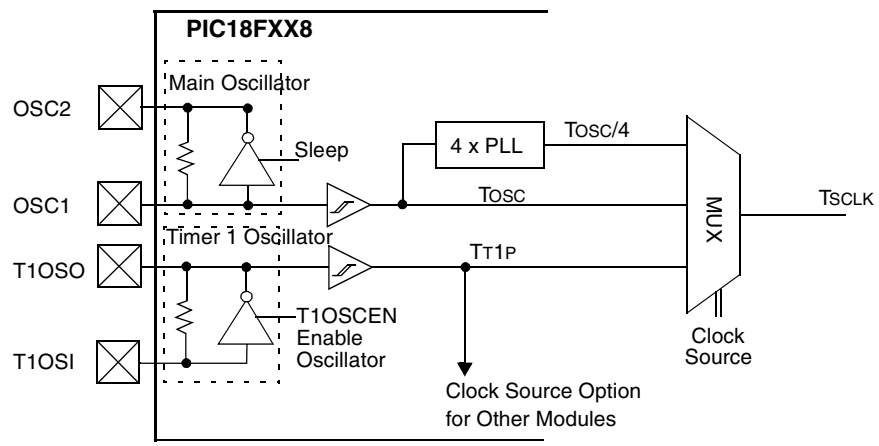
The PIC18FXX8 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low-frequency clock source. For the PIC18FXX8 devices, this alternate clock source is the Timer1 oscillator. If a low-frequency crystal (32 kHz, for example) has been attached 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 12.2 "Timer1 Oscillator"** for further details of the Timer1 oscillator and **Section 24.1 "Configuration Bits"** 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 comes 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 continues to be the system clock source. |

**FIGURE 2-6: DEVICE CLOCK SOURCES**




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

**REGISTER 2-1: OSCCON: OSCILLATOR CONTROL 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



### 2.6.2 OSCILLATOR TRANSITIONS

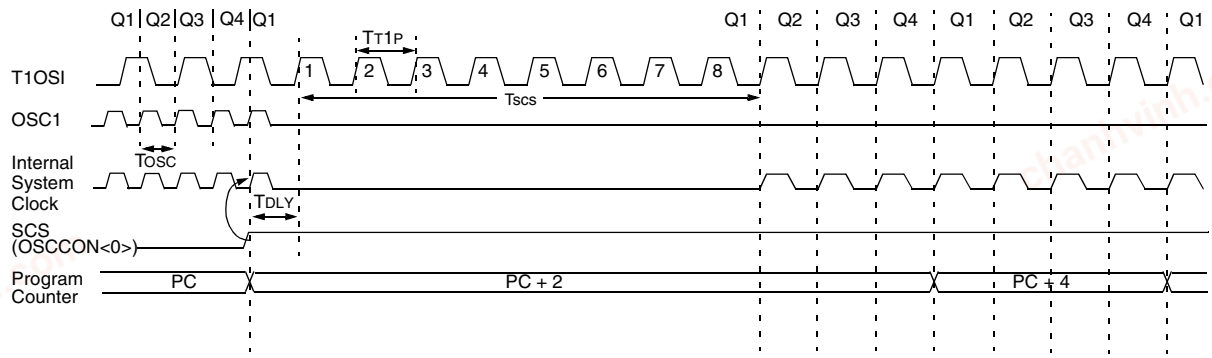
The PIC18FXX8 devices contain circuitry to prevent "glitches" when switching between oscillator sources. Essentially, the circuitry waits for eight rising edges of the clock source that the processor is switching to. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

Figure 2-7 shows a timing diagram indicating the transition from the main oscillator to the Timer1 oscillator. 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.

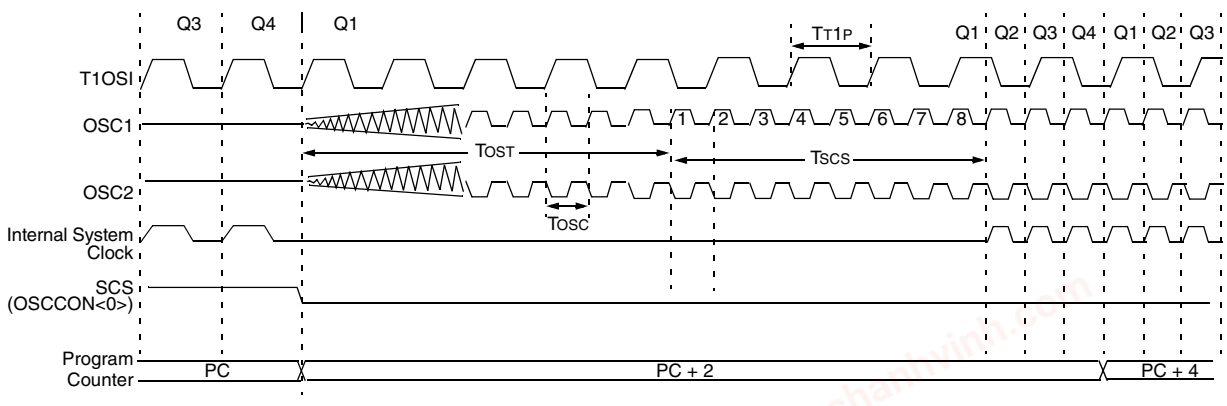
If the main oscillator is configured for an external crystal (HS, XT, LP), the transition will take place after an oscillator start-up time (TOST) has occurred. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS, XT and LP modes is shown in Figure 2-8.

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




**Note 1:** Delay on internal system clock is eight oscillator cycles for synchronization.

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




**Note 1:** TOST = 1024 TOSC (drawing not to scale).

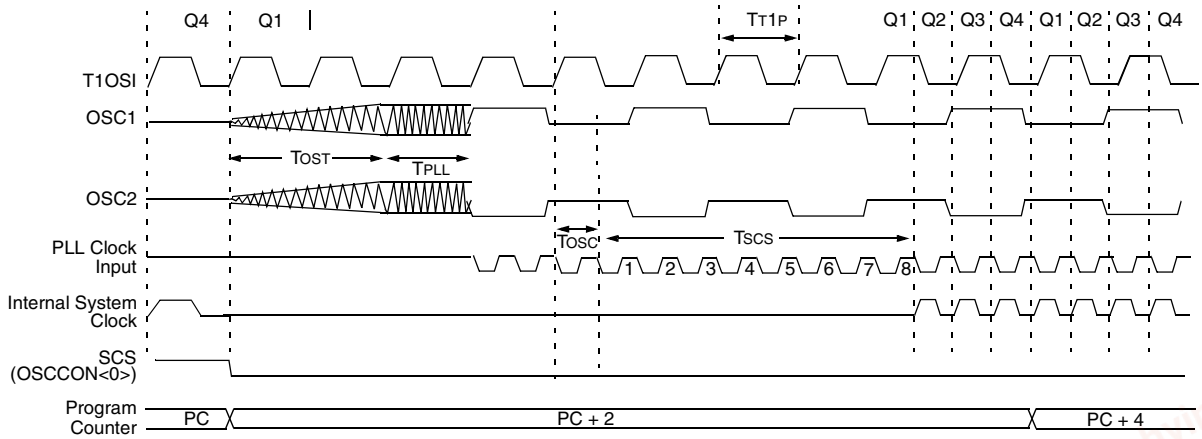
chanhvinh.com



If the main oscillator is configured for HS4 (PLL) mode, an oscillator start-up time (TOST) plus an additional PLL time-out (TPLL) will occur. The PLL time-out is typically 2 ms and allows the PLL to lock to the main oscillator frequency. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS4 mode is shown in Figure 2-9.

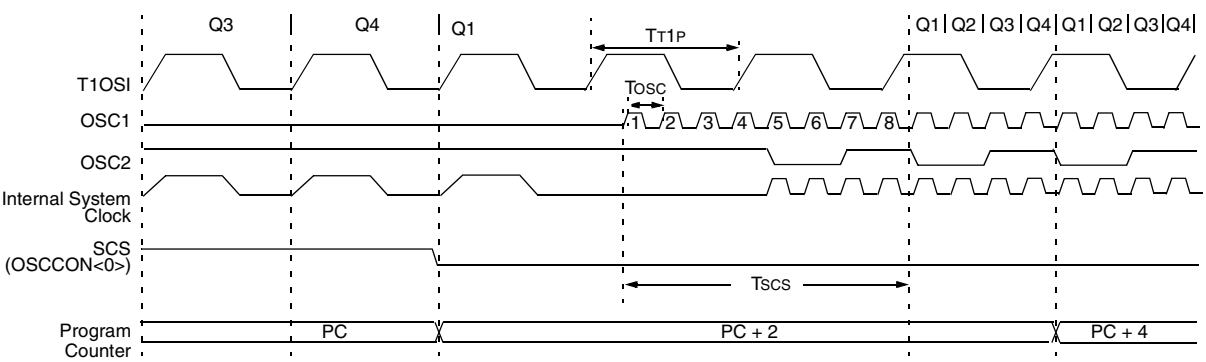
If the main oscillator is configured in the RC, RCIO, EC or ECIO modes, there is no oscillator start-up time-out. Operation will resume after eight cycles of the main oscillator have been counted. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for RC, RCIO,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

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




**Note 1:** TOST = 1024 TOSC (drawing not to scale).

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




**Note 1:** RC Oscillator mode assumed.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 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 #D033) 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 fixed 2 ms (nominal) 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 PORTA, bit 6 |
| ECIO | Floating | Configured as PORTA, 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 $\overline{\text{MCLR}}$ Reset.

© 2006 Microchip Technology Inc.

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 3.0 RESET

The PIC18FXX8 differentiates between various kinds of RESET:

a) Power-on Reset (POR)
b) $\overline{\text{MCLR}}$ Reset during normal operation
c) $\overline{\text{MCLR}}$ Reset during Sleep
d) Watchdog Timer (WDT) Reset during normal operation
e) Programmable Brown-out Reset (PBOR)
f) RESET Instruction
g) Stack Full Reset
h) 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.

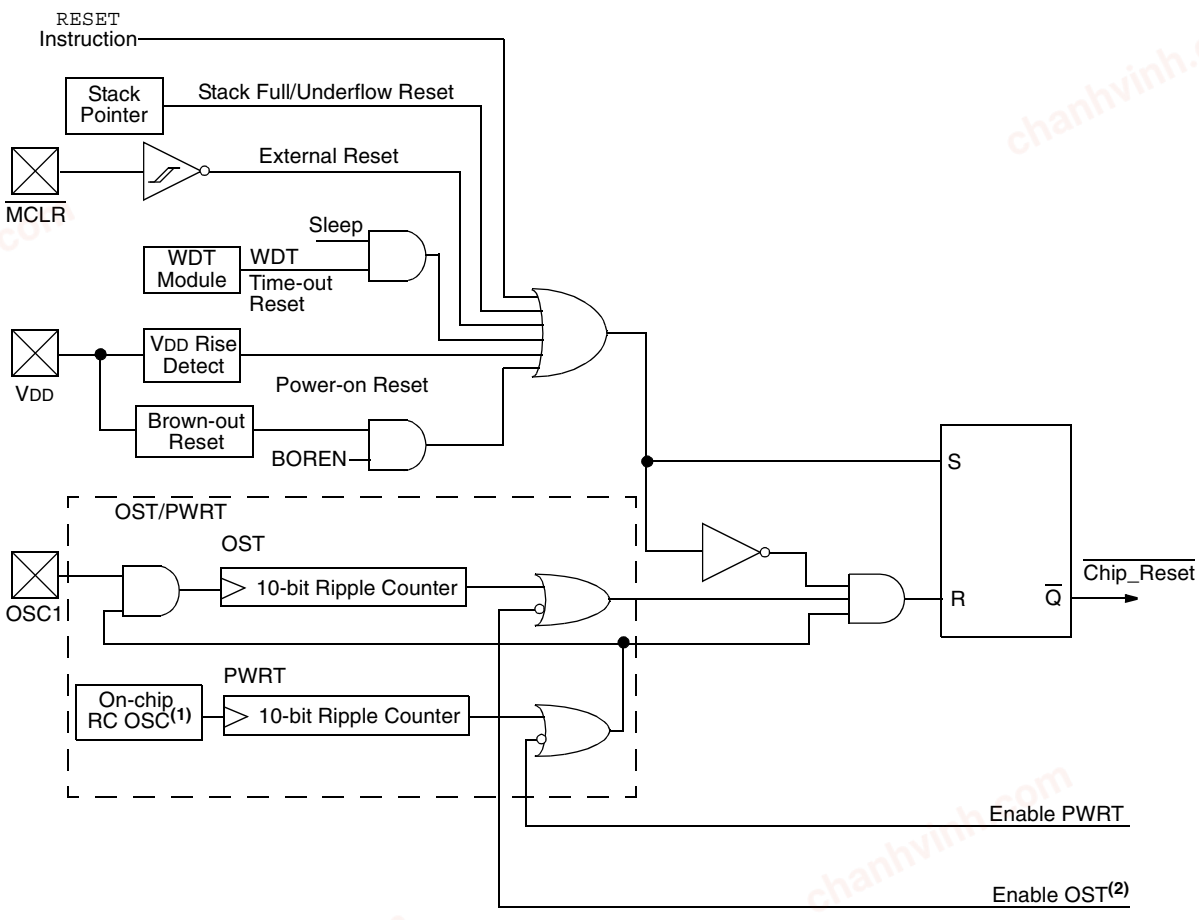
Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register, $\overline{\text{RI}}$, $\overline{\text{TO}}$, $\overline{\text{PD}}$, $\overline{\text{POR}}$ and $\overline{\text{BOR}}$ are set or cleared differently 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**




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

chanhvinh.com



## 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 eliminates external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified (refer to parameter D004). For a slow rise time, see Figure 3-2.

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

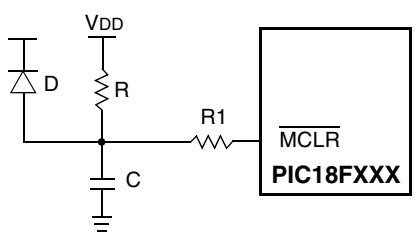
## 3.2 MCLR

PIC18FXX8 devices have a noise filter in the MCLR Reset path. The filter will detect and ignore small pulses.

It should be noted that a WDT Reset does not drive MCLR pin low.

The behavior of the ESD protection on the MCLR pin differs from previous devices of this family. Voltages applied to the pin that exceed its specification can result in both Resets and current draws outside of device specification during the Reset event. For this reason, Microchip recommends that the MCLR pin no longer be tied directly to VDD. The use of an RC network, as shown in Figure 3-2, is suggested.

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




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

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

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

## 3.3 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.

## 3.4 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides a 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over (parameter #32). This additional delay 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.

## 3.5 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).

## 3.6 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.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 3.7 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 $\overline{MCLR}$ is kept low long enough, the time-outs will expire. Bringing $\overline{MCLR}$ high will begin execution immediately (Figure 3-5). This is useful for testing purposes or to synchronize more than one PIC18FXX8 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 |
|---|---|---|---|---|
| | $\overline{PWRTEN}$ = 0 | $\overline{PWRTEN}$ = 1 | | |
| HS with PLL enabled(1) | 72 ms + 1024 TOSC + 2 ms | 1024 TOSC + 2 ms | 72 ms + 1024 TOSC + 2 ms | 1024 TOSC + 2 ms |
| HS, XT, LP | 72 ms + 1024 TOSC | 1024 TOSC | 72 ms + 1024 TOSC | 1024 TOSC |
| 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 | U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

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

| Condition | Program Counter | RCON Register | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | STKFUL | STKUNF |
|---|---|---|---|---|---|---|---|---|---|
| Power-on Reset | 0000h | 0--1 110q | 1 | 1 | 1 | 0 | 0 | u | u |
| $\overline{MCLR}$ Reset during normal operation | 0000h | 0--0 011q | u | u | u | u | u | u | u |
| Software Reset during normal operation | 0000h | 0--0 011q | 0 | u | u | u | u | u | u |
| Stack Full Reset during normal operation | 0000h | 0--0 011q | u | u | u | 1 | 1 | u | 1 |
| Stack Underflow Reset during normal operation | 0000h | 0--0 011q | u | u | u | 1 | 1 | 1 | u |
| $\overline{MCLR}$ Reset during Sleep | 0000h | 0--0 011q | u | 1 | 0 | u | u | u | u |
| WDT Reset | 0000h | 0--0 011q | u | 0 | 1 | u | u | u | u |
| WDT Wake-up | PC + 2 | 0--1 101q | u | 0 | 0 | u | u | u | u |
| Brown-out Reset | 0000h | 0--1 110q | 1 | 1 | 1 | u | 0 | u | u |
| Interrupt wake-up from Sleep | PC + 2(1) | 0--1 101q | u | 1 | 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 (000008h or 000018h).

chanhvinh.com



**FIGURE 3-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD)**

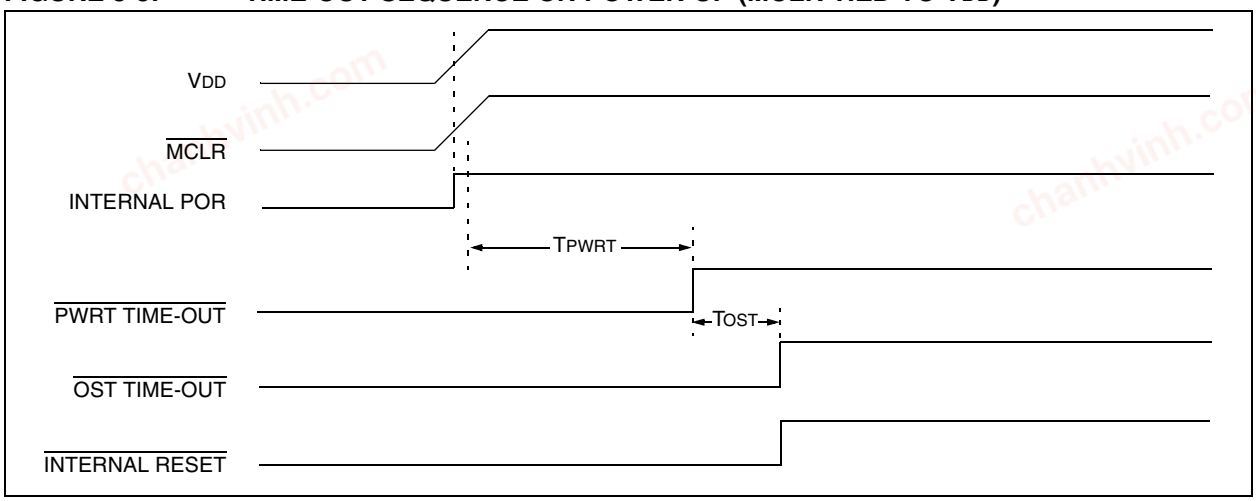



**FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1**

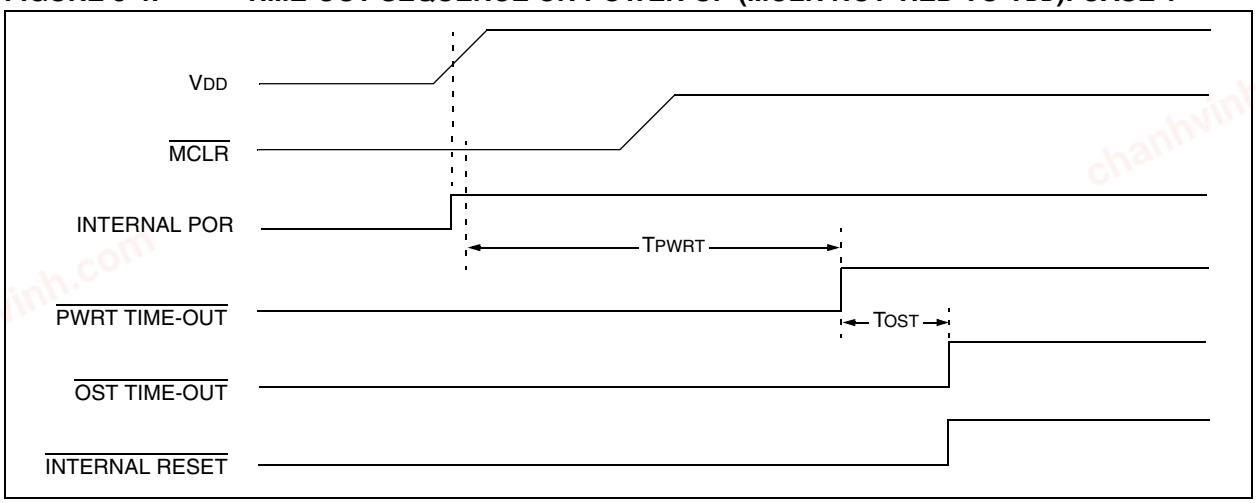
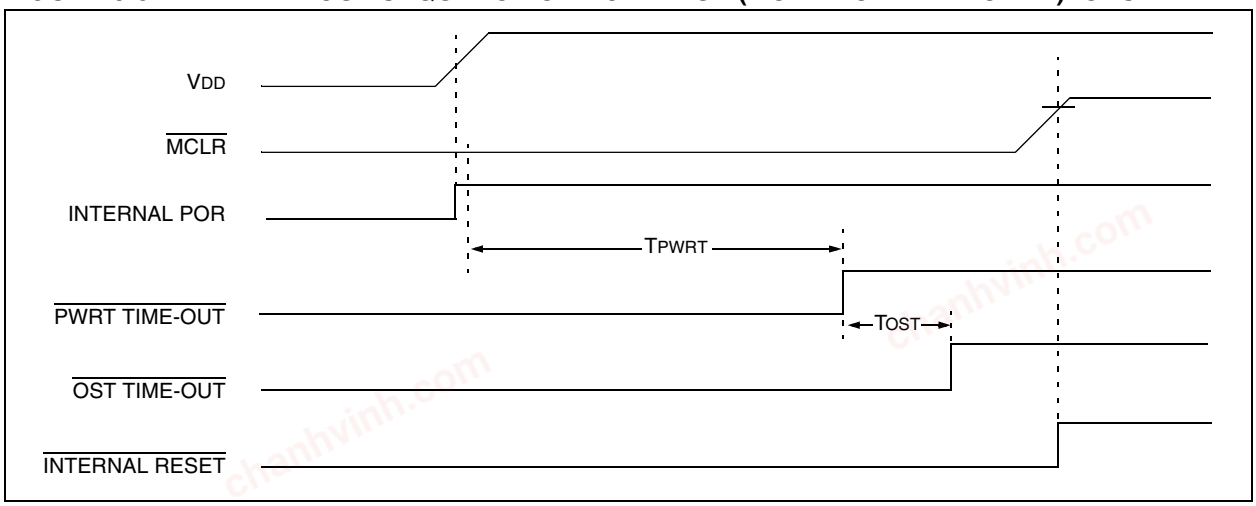



**FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2**





© 2006 Microchip Technology Inc.

chanhvinh.com



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

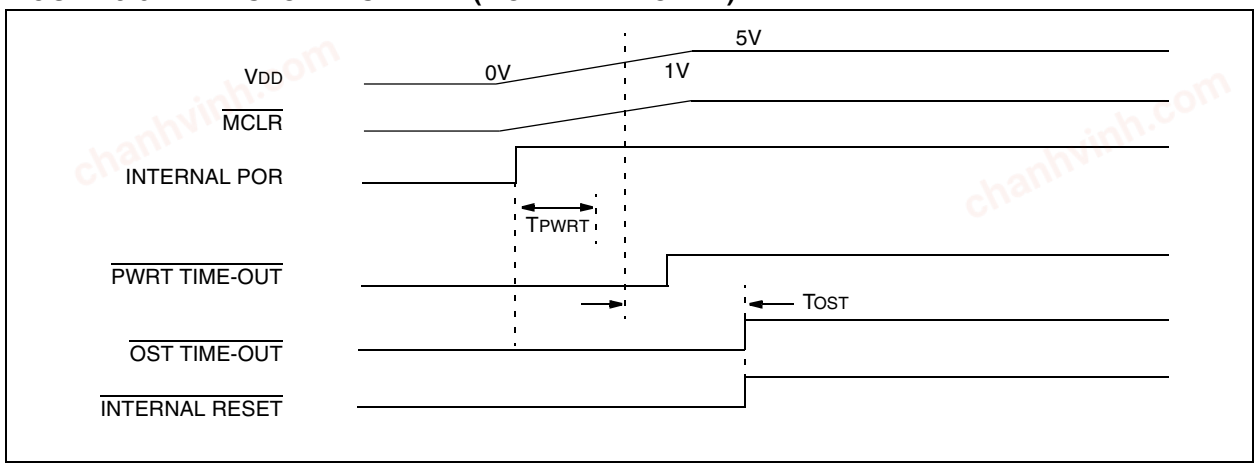
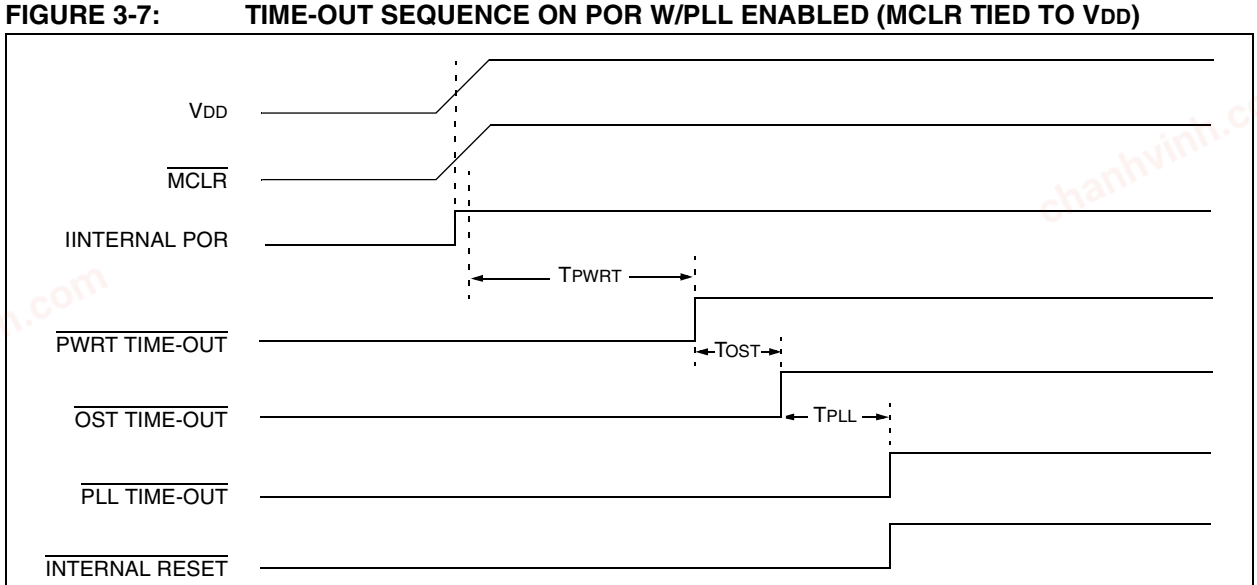



**FIGURE 3-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED ($\overline{MCLR}$ TIED TO VDD)**




**Note:** TOST = 1024 clock cycles.
TPLL ≈ 2 ms max. First three stages of the PWRT timer.

chanhvinh.com



**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 |
|---|---|---|---|---|---|
| TOSU | PIC18F2X8 | PIC18F4X8 | ---0 0000 | ---0 0000 | ---0 uuuu(3) |
| TOSH | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| TOSL | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| STKPTR | PIC18F2X8 | PIC18F4X8 | 00-0 0000 | uu-0 0000 | uu-u uuuu(3) |
| PCLATU | PIC18F2X8 | PIC18F4X8 | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | PC + 2(2) |
| TBLPTRU | PIC18F2X8 | PIC18F4X8 | --00 0000 | --00 0000 | --uu uuuu |
| TBLPTRH | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRL | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TABLAT | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PRODH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INTCON | PIC18F2X8 | PIC18F4X8 | 0000 000x | 0000 000u | uuuu uuuu(1) |
| INTCON2 | PIC18F2X8 | PIC18F4X8 | 111- -1-1 | 111- -1-1 | uuu- -u-u(1) |
| INTCON3 | PIC18F2X8 | PIC18F4X8 | 11-0 0-00 | 11-0 0-00 | uu-u u-uu(1) |
| INDF0 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| POSTINC0 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| POSTDEC0 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| PREINC0 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| PLUSW0 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| FSR0H | PIC18F2X8 | PIC18F4X8 | ---- xxxx | ---- uuuu | ---- uuuu |
| FSR0L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF1 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| POSTINC1 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| POSTDEC1 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| PREINC1 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| PLUSW1 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**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:** Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).


© 2006 Microchip Technology Inc.

chanhvinh.com



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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| FSR1H | PIC18F2X8 | PIC18F4X8 | ---- xxxx | ---- uuuu | ---- uuuu |
| FSR1L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| BSR | PIC18F2X8 | PIC18F4X8 | ---- 0000 | ---- 0000 | ---- uuuu |
| INDF2 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| POSTINC2 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| POSTDEC2 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| PREINC2 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| PLUSW2 | PIC18F2X8 | PIC18F4X8 | N/A | N/A | N/A |
| FSR2H | PIC18F2X8 | PIC18F4X8 | ---- xxxx | ---- uuuu | ---- uuuu |
| FSR2L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| STATUS | PIC18F2X8 | PIC18F4X8 | ---x xxxx | ---u uuuu | ---u uuuu |
| TMR0H | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR0L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T0CON | PIC18F2X8 | PIC18F4X8 | 1111 1111 | 1111 1111 | uuuu uuuu |
| OSCCON | PIC18F2X8 | PIC18F4X8 | ---- ---0 | ---- ---0 | ---- ---u |
| LVDCON | PIC18F2X8 | PIC18F4X8 | --00 0101 | --00 0101 | --uu uuuu |
| WDTCON | PIC18F2X8 | PIC18F4X8 | ---- ---0 | ---- ---0 | ---- ---u |
| RCON(4) | PIC18F2X8 | PIC18F4X8 | 0--1 110q | 0--0 011q | 0--1 101q |
| TMR1H | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR1L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T1CON | PIC18F2X8 | PIC18F4X8 | 0-00 0000 | u-uu uuuu | u-uu uuuu |
| TMR2 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR2 | PIC18F2X8 | PIC18F4X8 | 1111 1111 | 1111 1111 | 1111 1111 |
| T2CON | PIC18F2X8 | PIC18F4X8 | -000 0000 | -000 0000 | -uuu uuuu |
| SSPBUF | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SSPADD | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPSTAT | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON1 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON2 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ADRESH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADCON0 | PIC18F2X8 | PIC18F4X8 | 0000 00-0 | 0000 00-0 | uuuu uu-u |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**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:** Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

© 2006 Microchip Technology Inc.

chanhvinh.com



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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| ADCON1 | PIC18F2X8 | PIC18F4X8 | 00-- 0000 | 00-- 0000 | uu-- uuuu |
| CCPR1H | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR1L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP1CON | PIC18F2X8 | PIC18F4X8 | --00 0000 | --00 0000 | --uu uuuu |
| ECCPR1H | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ECCPR1L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ECCP1CON | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | 0000 0000 |
| ECCP1DEL | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | 0000 0000 |
| ECCPAS | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | 0000 0000 |
| CVRCON | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CMCON | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR3H | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T3CON | PIC18F2X8 | PIC18F4X8 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| SPBRG | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXSTA | PIC18F2X8 | PIC18F4X8 | 0000 -010 | 0000 -010 | uuuu -uuu |
| RCSTA | PIC18F2X8 | PIC18F4X8 | 0000 000x | 0000 000u | uuuu uuuu |
| EEADR | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| EEDATA | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| EECON2 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| EECON1 | PIC18F2X8 | PIC18F4X8 | xx-0 x000 | uu-0 u000 | uu-0 u000 |
| IPR3 | PIC18F2X8 | PIC18F4X8 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR3 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE3 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IPR2 | PIC18F2X8 | PIC18F4X8 | -1-1 1111 | -1-1 1111 | -u-u uuuu |
| PIR2 | PIC18F2X8 | PIC18F4X8 | -0-0 0000 | -0-0 0000 | -u-u uuuu(1) |
| PIE2 | PIC18F2X8 | PIC18F4X8 | -0-0 0000 | -0-0 0000 | -u-u uuuu |
| IPR1 | PIC18F2X8 | PIC18F4X8 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR1 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu(1) |
| PIE1 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**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:** Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).



© 2006 Microchip Technology Inc.

chanhvinh.com



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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| TRISE | PIC18F2X8 | PIC18F4X8 | 0000 -111 | 0000 -111 | uuuu -uuu |
| TRISD | PIC18F2X8 | PIC18F4X8 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISC | PIC18F2X8 | PIC18F4X8 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISB | PIC18F2X8 | PIC18F4X8 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISA(5) | PIC18F2X8 | PIC18F4X8 | -111 1111(5) | -111 1111(5) | -uuu uuuu(5) |
| LATE | PIC18F2X8 | PIC18F4X8 | ---- -xxx | ---- -uuu | ---- -uuu |
| LATD | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATC | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATB | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATA(5) | PIC18F2X8 | PIC18F4X8 | -xxx xxxx(5) | -uuu uuuu(5) | -uuu uuuu(5) |
| PORTE | PIC18F2X8 | PIC18F4X8 | ---- -xxx | ---- -000 | ---- -uuu |
| PORTD | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTC | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTB | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA(5) | PIC18F2X8 | PIC18F4X8 | -x0x 0000 (5) | -u0u 0000(5) | -uuu uuuu(5) |
| TXERRCNT | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXERRCNT | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| COMSTAT | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CIOCON | PIC18F2X8 | PIC18F4X8 | --00 ---- | --00 ---- | --uu ---- |
| BRGCON3 | PIC18F2X8 | PIC18F4X8 | -0-- -000 | -0-- -000 | -u-- -uuu |
| BRGCON2 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| BRGCON1 | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CANCON | PIC18F2X8 | PIC18F4X8 | xxxx xxx- | uuuu uuu- | uuuu uuu- |
| CANSTAT(6) | PIC18F2X8 | PIC18F4X8 | xxx- xxx- | uuu- uuu- | uuu- uuu- |
| RXB0D7 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D6 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D5 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D4 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D3 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D2 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D1 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D0 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**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:** Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

© 2006 Microchip Technology Inc.

chanhvinh.com



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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| RXB0DLC | PIC18F2X8 | PIC18F4X8 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| RXB0EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0SIDL | PIC18F2X8 | PIC18F4X8 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| RXB0SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0CON | PIC18F2X8 | PIC18F4X8 | 000- 0000 | 000- 0000 | uuu- uuuu |
| RXB1D7 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D6 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D5 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D4 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D3 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D2 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D1 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D0 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1DLC | PIC18F2X8 | PIC18F4X8 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| RXB1EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1SIDL | PIC18F2X8 | PIC18F4X8 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| RXB1SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1CON | PIC18F2X8 | PIC18F4X8 | 000- 0000 | 000- 0000 | uuu- uuuu |
| TXB0D7 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D6 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D5 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D4 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D3 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D2 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D1 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D0 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0DLC | PIC18F2X8 | PIC18F4X8 | -x-- xxxx | -u-- uuuu | -u-- uuuu |
| TXB0EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**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:** Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).


© 2006 Microchip Technology Inc.

chanhvinh.com



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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| TXB0SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0CON | PIC18F2X8 | PIC18F4X8 | -000 0-00 | -000 0-00 | -uuu u-uu |
| TXB1D7 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D6 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D5 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D4 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D3 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D2 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D1 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D0 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1DLC | PIC18F2X8 | PIC18F4X8 | -x-- xxxx | -u-- uuuu | -u-- uuuu |
| TXB1EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| TXB1SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1CON | PIC18F2X8 | PIC18F4X8 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXB2D7 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D6 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D5 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D4 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D3 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D2 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D1 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2D0 | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2DLC | PIC18F2X8 | PIC18F4X8 | -x-- xxxx | -u-- uuuu | -u-- uuuu |
| TXB2EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| TXB2SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2CON | PIC18F2X8 | PIC18F4X8 | -000 0-00 | -000 0-00 | -uuu u-uu |
| RXM1EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM1EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**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:** Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

© 2006 Microchip Technology Inc.

chanhvinh.com



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

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| RXM1SIDL | PIC18F2X8 | PIC18F4X8 | xxx- --xx | uuu- --uu | uuu- --uu |
| RXM1SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0SIDL | PIC18F2X8 | PIC18F4X8 | xxx- --xx | uuu- --uu | uuu- --uu |
| RXM0SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF5SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF4SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF3SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF2SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF1SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0EIDL | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0EIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0SIDL | PIC18F2X8 | PIC18F4X8 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF0SIDH | PIC18F2X8 | PIC18F4X8 | xxxx xxxx | uuuu uuuu | uuuu uuuu |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**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:** Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).


© 2006 Microchip Technology Inc.

chanhvinh.com



## 4.0 MEMORY ORGANIZATION

There are three memory blocks in Enhanced MCU devices. These memory blocks are:

- Enhanced Flash Program Memory
- Data Memory
- EEPROM Data Memory

Data and program memory use separate busses, which allows concurrent access of these blocks. Additional detailed information on data EEPROM and Flash program memory is provided in **Section 5.0 "Data EEPROM Memory"** and **Section 6.0 "Flash Program Memory"**, respectively.

### 4.1 Program Memory Organization

The PIC18F258/458 devices have a 21-bit program counter that is capable of addressing a 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: PROGRAM MEMORY MAP AND STACK FOR PIC18F248/448**

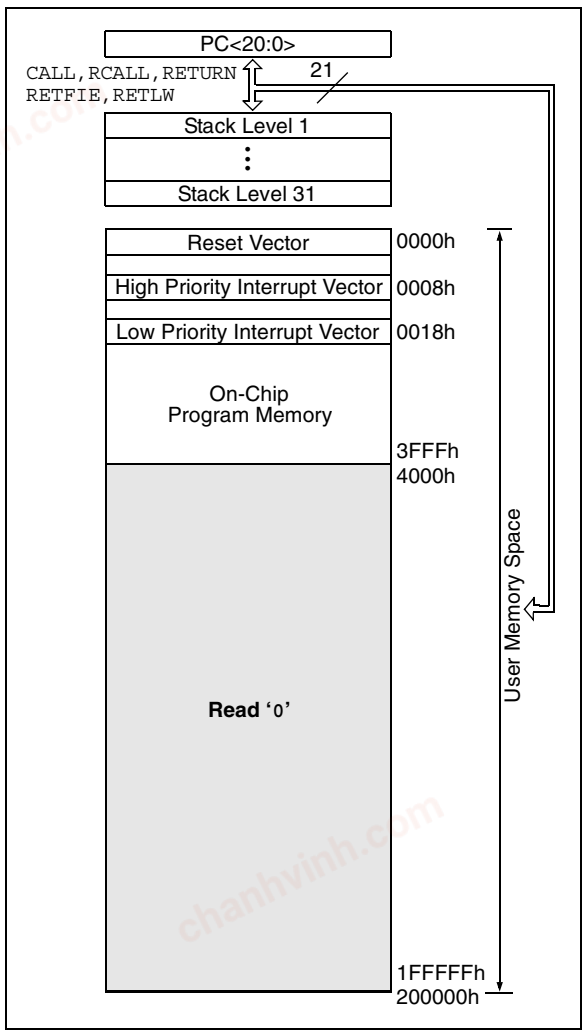
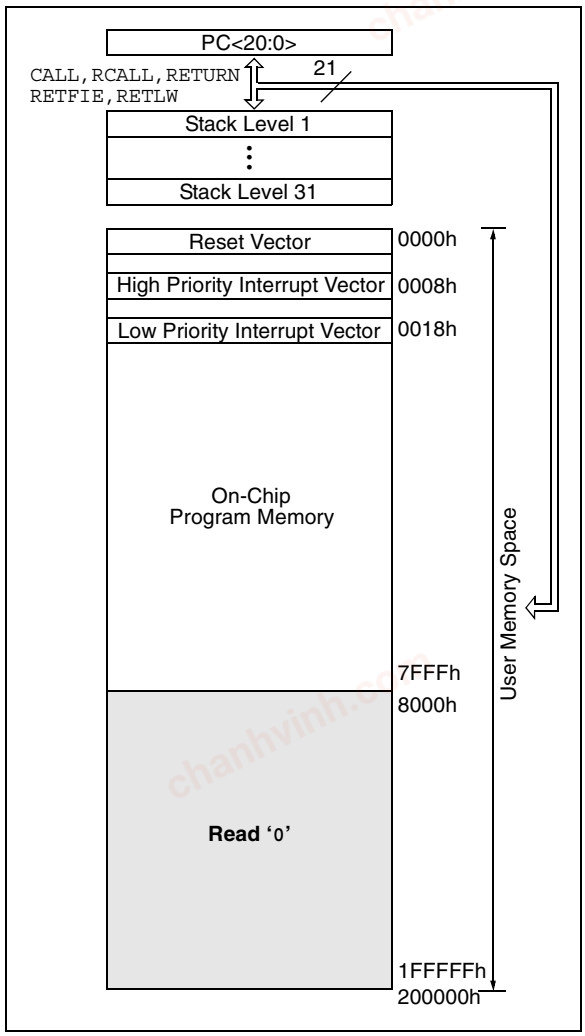



Figure 4-1 shows the diagram for program memory map and stack for the PIC18F248 and PIC18F448. Figure 4-2 shows the diagram for the program memory map and stack for the PIC18F258 and PIC18F458.

#### 4.1.1 INTERNAL PROGRAM MEMORY OPERATION

The PIC18F258 and the PIC18F458 have 32 Kbytes of internal Enhanced Flash program memory. This means that the PIC18F258 and the PIC18F458 can store up to 16K of single-word instructions. The PIC18F248 and PIC18F448 have 16 Kbytes of Enhanced Flash program memory. This translates into 8192 single-word instructions, which can be stored in the program memory. 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 4-2: PROGRAM MEMORY MAP AND STACK FOR PIC18F258/458**



chanhvinh.com



## 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 register, 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 are 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 21.0 "Comparator Module"** 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. |
|---|---|


© 2006 Microchip Technology Inc.

chanhvinh.com



**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

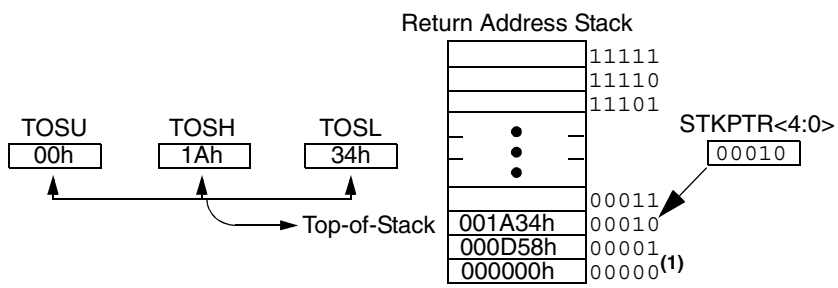
bit 5 **Unimplemented**: Read as '0'

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

**Note:** Bit 7 and bit 6 need to be cleared following a stack underflow or a stack overflow.

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

**FIGURE 4-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**




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

chanhvinh.com



### 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 condition 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 “Computed GOTO”**).


© 2006 Microchip Technology Inc.

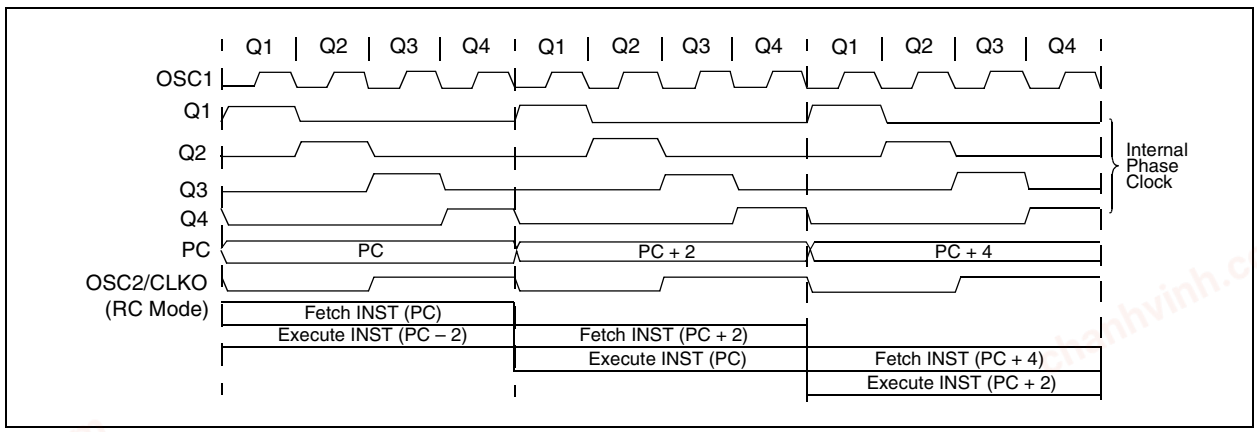
chanhvinh.com



## 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 4-4.

**FIGURE 4-4: CLOCK/INSTRUCTION CYCLE**




## 4.6 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute take 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

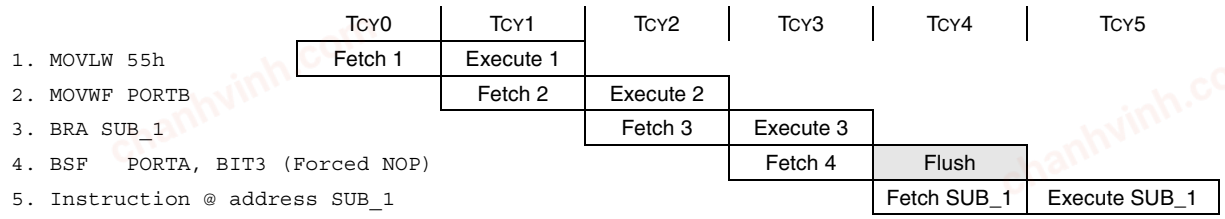
The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = `0`). Figure 4-3 shows an example of how instruction words are stored 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 “PCL, PCLATH and PCLATU”**).

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 Example 4-3 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 25.0 “Instruction Set Summary”** provides further details of the instruction set.

chanhvinh.com



**EXAMPLE 4-2: INSTRUCTION PIPELINE FLOW**




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

**EXAMPLE 4-3: INSTRUCTIONS IN PROGRAM MEMORY**

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


© 2006 Microchip Technology Inc.

chanhvinh.com



### 4.7.1 TWO-WORD INSTRUCTIONS

The PIC18FXX8 devices have 4 two-word instructions: MOVFF, CALL, GOTO and LFSR. The 4 Most Significant bits of the second word are set to '1's and indicate a special NOP instruction. The lower 12 bits of the second word contain the 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 4-4. Refer to **Section 25.0 "Instruction Set Summary"** for further details of the instruction set.

## 4.8 Look-up Tables

Look-up 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 look-up 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.

**Note 1:** The LSb of PCL is fixed to a value of '0'. Hence, computed GOTO to an odd address is not possible.

**2:** The ADDWF PCL instruction does not update PCLATH/PCLATU. A read operation on PCL must be performed to update PCLATH and PCLATU.

### 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.

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

A description of the table read/table write operation is shown in **Section 6.1 "Table Reads and Table Writes"**.

**EXAMPLE 4-4: 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 |

chanhvinh.com



## 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 4-6 shows the data memory organization for the PIC18FXX8 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 (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. The SFRs start at the last location of Bank 15 (FFFh) and grow downwards. GPRs 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 (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. **Section 4.10 "Access Bank"** 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 "Indirect Addressing, INDF and FSR Registers"**.

Enhanced MCU devices may have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

Data RAM is available for use as GPR registers by all instructions. Bank 15 (F00h to FFFh) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

### 4.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-1.

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

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

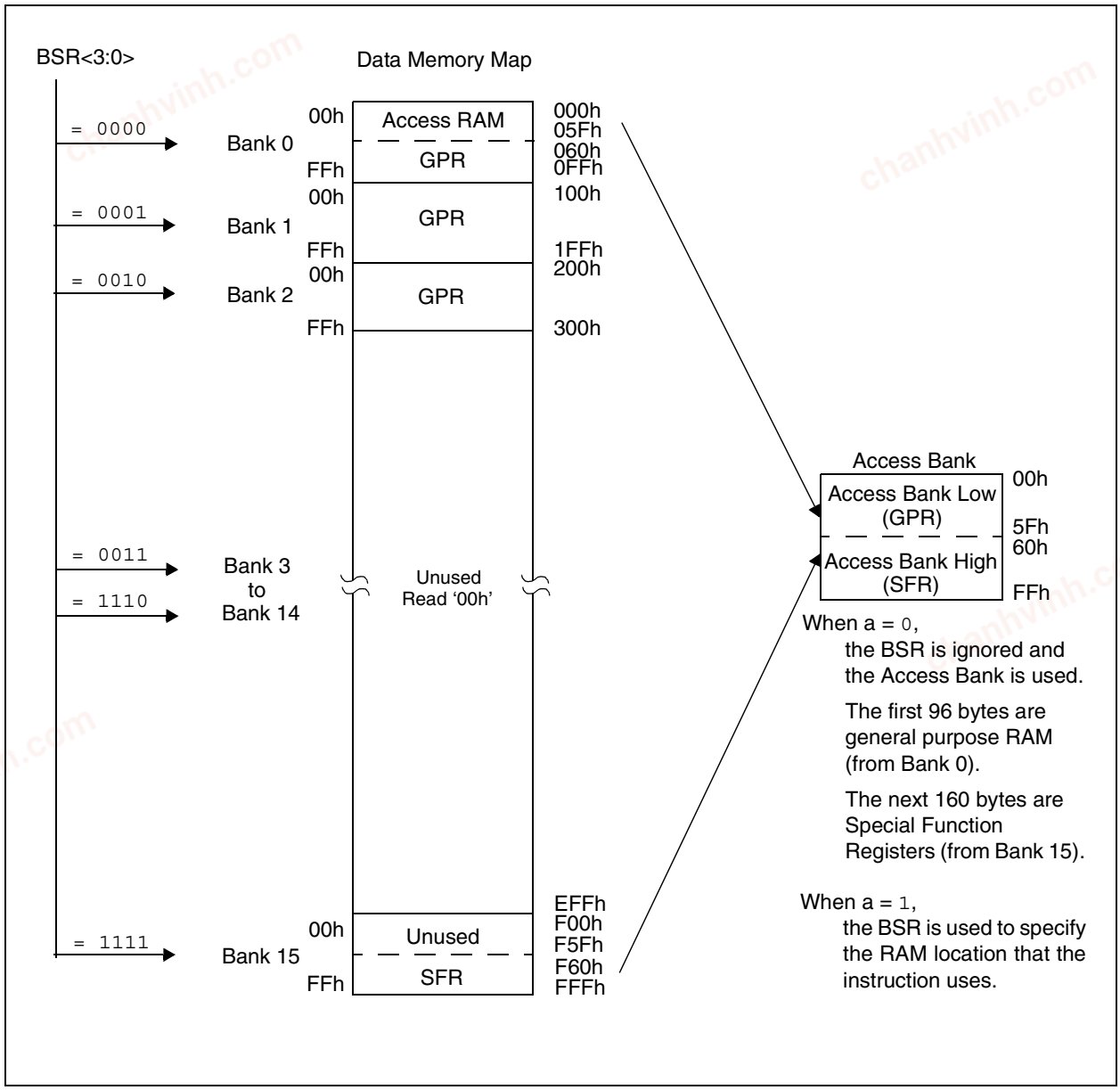
The unused SFR locations will be unimplemented and read as '0's. See Table 4-1 for addresses for the SFRs.


© 2006 Microchip Technology Inc.

chanhvinh.com



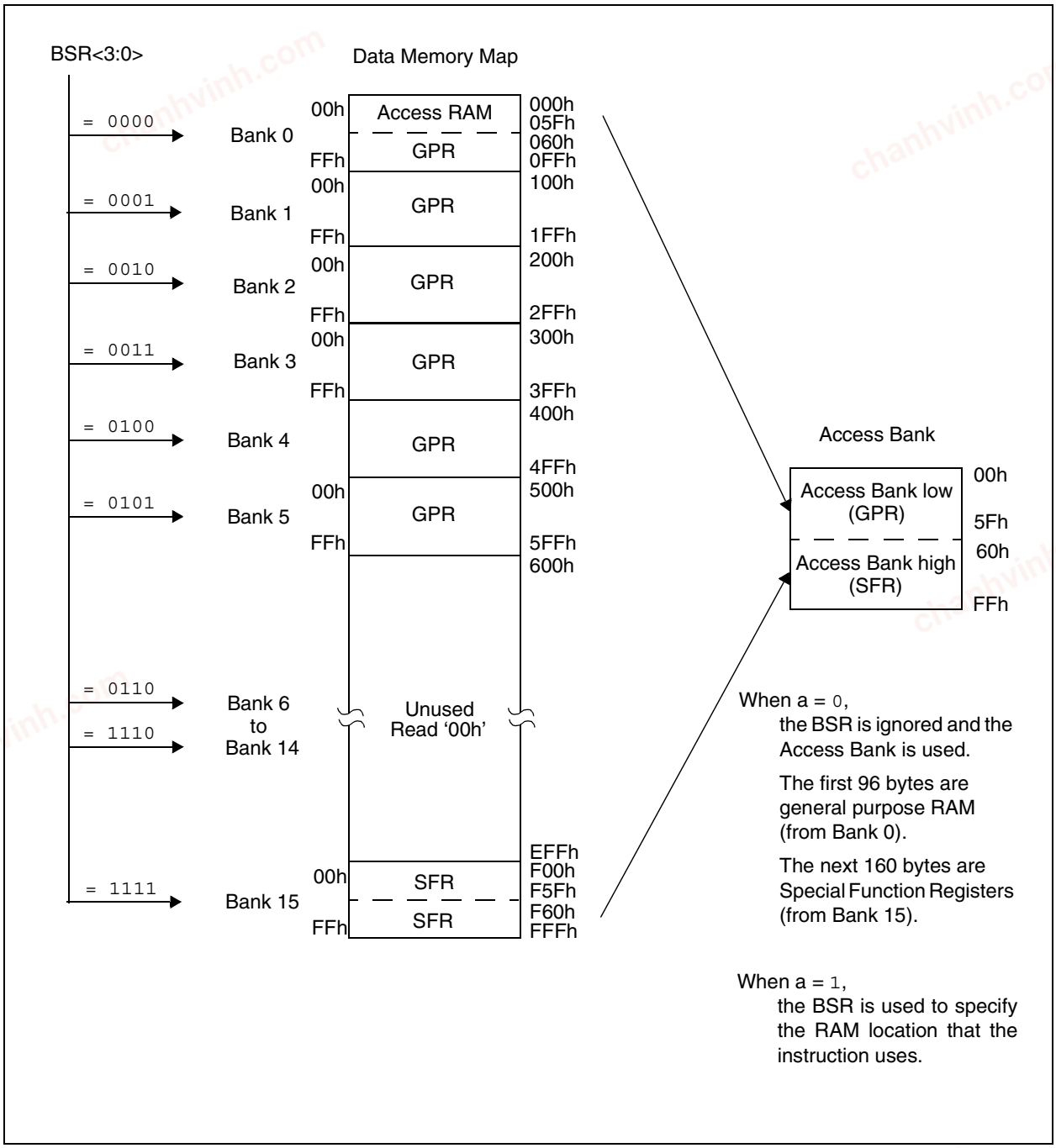
**FIGURE 4-5: DATA MEMORY MAP FOR PIC18F248/448**



chanhvinh.com



**FIGURE 4-6: DATA MEMORY MAP FOR PIC18F258/458**





© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 4-1: 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 | ECCPR1H(5) | F9Ch | — |
| FFBh | PCLATU | FDBh | PLUSW2(2) | FBBh | ECCPR1L(5) | F9Bh | — |
| FFAh | PCLATH | FDAh | FSR2H | FBAh | ECCP1CON(5) | F9Ah | — |
| FF9h | PCL | FD9h | FSR2L | FB9h | — | F99h | — |
| FF8h | TBLPTRU | FD8h | STATUS | FB8h | — | F98h | — |
| FF7h | TBLPTRH | FD7h | TMR0H | FB7h | ECCP1DEL(5) | F97h | — |
| FF6h | TBLPTRL | FD6h | TMR0L | FB6h | ECCPAS(5) | F96h | TRISE(5) |
| FF5h | TABLAT | FD5h | T0CON | FB5h | CVRCON(5) | F95h | TRISD(5) |
| FF4h | PRODH | FD4h | — | FB4h | CMCON(5) | 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 | — |
| FF0h | INTCON3 | FD0h | RCON | FB0h | — | F90h | — |
| FEFh | INDF0(2) | FCFh | TMR1H | FAFh | SPBRG | F8Fh | — |
| FEEh | POSTINC0(2) | FCEh | TMR1L | FAEh | RCREG | F8Eh | — |
| FEDh | POSTDEC0(2) | FCDh | T1CON | FADh | TXREG | F8Dh | LATE(5) |
| FECh | PREINC0(2) | FCCh | TMR2 | FACh | TXSTA | F8Ch | LATD(5) |
| FEBh | PLUSW0(2) | FCBh | PR2 | FABh | RCSTA | F8Bh | LATC |
| FEAh | FSR0H | FCAh | T2CON | FAAh | — | F8Ah | LATB |
| FE9h | FSR0L | FC9h | SSPBUF | FA9h | EEADR | F89h | LATA |
| FE8h | WREG | FC8h | SSPADD | FA8h | EEDATA | F88h | — |
| FE7h | INDF1(2) | FC7h | SSPSTAT | FA7h | EECON2 | F87h | — |
| FE6h | POSTINC1(2) | FC6h | SSPCON1 | FA6h | EECON1 | F86h | — |
| FE5h | POSTDEC1(2) | FC5h | SSPCON2 | FA5h | IPR3 | F85h | — |
| FE4h | PREINC1(2) | FC4h | ADRESH | FA4h | PIR3 | F84h | PORTE(5) |
| FE3h | PLUSW1(2) | FC3h | ADRESL | FA3h | PIE3 | F83h | PORTD(5) |
| FE2h | FSR1H | FC2h | ADCON0 | FA2h | IPR2 | F82h | PORTC |
| FE1h | FSR1L | FC1h | ADCON1 | FA1h | PIR2 | F81h | PORTB |
| FE0h | BSR | FC0h | — | FA0h | PIE2 | F80h | PORTA |

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

**2:** This is not a physical register.

**3:** Contents of register are dependent on WIN2:WIN0 bits in the 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:** These registers are not implemented on the PIC18F248 and PIC18F258.

© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 4-1: SPECIAL FUNCTION REGISTER MAP (CONTINUED)**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| F7Fh | — | F5Fh | — | F3Fh | — | F1Fh | RXM1EIDL |
| F7Eh | — | F5Eh | CANSTATRO1(4) | F3Eh | CANSTATRO3(4) | 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 | CANSTATRO2(4) | F2Eh | CANSTATRO4(4) | F0Eh | RXF3EIDH |
| 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 | RXF2EIDL |
| F6Ah | RXB0D4(3) | F4Ah | TXB0D4 | F2Ah | TXB2D4 | F0Ah | RXF2EIDH |
| 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 | RXF1EIDL |
| F66h | RXB0D0(3) | F46h | TXB0D0 | F26h | TXB2D0 | F06h | RXF1EIDH |
| 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 are dependent on WIN2:WIN0 bits in the 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:** These registers are not implemented on the PIC18F248 and PIC18F258.



© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 4-2: REGISTER FILE SUMMARY**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| TOSU | — | — | — | Top-of-Stack Upper Byte (TOS<20:16>) | | | | | ---0 0000 | 30, 38 |
| TOSH | Top-of-Stack High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | 30, 38 |
| TOSL | Top-of-Stack Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | 30, 38 |
| STKPTR | STKFUL | STKUNF | — | Return Stack Pointer | | | | | 00-0 0000 | 30, 39 |
| PCLATU | — | — | bit 21(2) | Holding Register for PC<20:16> | | | | | ---0 0000 | 30, 40 |
| PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | 30, 40 |
| PCL | PC Low Byte (PC<7:0>) | | | | | | | | 0000 0000 | 30, 40 |
| TBLPTRU | — | — | bit 21(2) | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 | 30, 68 |
| TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 30, 68 |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 30, 68 |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 30, 68 |
| PRODH | Product Register High Byte | | | | | | | | xxxx xxxx | 30, 75 |
| PRODL | Product Register Low Byte | | | | | | | | xxxx xxxx | 30, 75 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 30, 79 |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | — | — | TMR0IP | — | RBIP | 111- -1-1 | 30, 80 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | 30, 81 |
| INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | N/A | 30, 55 |
| POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | N/A | 30, 55 |
| POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | N/A | 30, 55 |
| PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | N/A | 30, 55 |
| PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 offset by W (not a physical register) | | | | | | | | N/A | 30, 55 |
| FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High | | | | ---- xxxx | 30, 55 |
| FSR0L | Indirect Data Memory Address Pointer 0 Low Byte | | | | | | | | xxxx xxxx | 30, 55 |
| WREG | Working Register | | | | | | | | xxxx xxxx | 30, 55 |
| INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | N/A | 30, 55 |
| POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | N/A | 30, 55 |
| POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | N/A | 30, 55 |
| PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | N/A | 30, 55 |
| PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 offset by W (not a physical register) | | | | | | | | N/A | 30, 55 |
| FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High | | | | ---- xxxx | 31, 55 |
| FSR1L | Indirect Data Memory Address Pointer 1 Low Byte | | | | | | | | xxxx xxxx | 31, 55 |
| BSR | — | — | — | — | Bank Select Register | | | | ---- 0000 | 31, 54 |
| INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | N/A | 31, 55 |
| POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | N/A | 31, 55 |
| POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | N/A | 31, 55 |
| PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | N/A | 31, 55 |
| PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 offset by W (not a physical register) | | | | | | | | N/A | 31, 55 |
| FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High | | | | ---- xxxx | 31, 55 |
| FSR2L | Indirect Data Memory Address Pointer 2 Low Byte | | | | | | | | xxxx xxxx | 31, 55 |
| STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | 31, 57 |
| TMR0H | Timer0 Register High Byte | | | | | | | | 0000 0000 | 31, 111 |
| TMR0L | Timer0 Register Low Byte | | | | | | | | xxxx xxxx | 31, 111 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 31, 109 |
| OSCCON | — | — | — | — | — | — | — | SCS | ---- ---0 | 31, 20 |
| LVDCON | — | — | IRVST | LVDEN | LVDL3 | LVDL2 | LVDL1 | LVDL0 | --00 0101 | 31, 261 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | ---- ---0 | 31, 272 |
| RCON | IPEN | — | — | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 0--1 110q | 31, 58, 91 |

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

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

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

**3:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| TMR1H | Timer1 Register High Byte | | | | | | | | xxxx xxxx | 31, 116 |
| TMR1L | Timer1 Register Low Byte | | | | | | | | xxxx xxxx | 31, 116 |
| T1CON | RD16 | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0-00 0000 | 31, 113 |
| TMR2 | Timer2 Register | | | | | | | | 0000 0000 | 31, 118 |
| PR2 | Timer2 Period Register | | | | | | | | 1111 1111 | 31, 118 |
| T2CON | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | 31, 117 |
| SSPBUF | SSP Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | 31, 146 |
| SSPADD | SSP Address Register in $I^2C$™ Slave mode. SSP Baud Rate Reload Register in $I^2C$ Master mode. | | | | | | | | 0000 0000 | 31, 152 |
| SSPSTAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 0000 0000 | 31, 144, 153 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 31, 145, 145 |
| SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 31, 155 |
| ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | 31, 243 |
| ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | 31, 243 |
| ADCON0 | ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | — | ADON | 0000 00-0 | 31, 241 |
| ADCON1 | ADFM | ADCS2 | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 00-- 0000 | 32, 242 |
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 32, 124 |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 32, 124 |
| CCP1CON | — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | --00 0000 | 32, 123 |
| ECCPR1H(1) | Enhanced Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 32, 133 |
| ECCPR1L(1) | Enhanced Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 32, 133 |
| ECCP1CON(1) | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 0000 0000 | 32, 131 |
| ECCP1DEL(1) | EPDC7 | EPDC6 | EPDC5 | EPDC4 | EPDC3 | EPDC2 | EPDC1 | EPDC0 | 0000 0000 | 32, 140 |
| ECCPAS(1) | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 | 0000 0000 | 32, 142 |
| CVRCON(1) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 0000 0000 | 32, 255 |
| CMCON(1) | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 0000 0000 | 32, 249 |
| TMR3H | Timer3 Register High Byte | | | | | | | | xxxx xxxx | 32, 121 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | xxxx xxxx | 32, 121 |
| T3CON | RD16 | T3ECCP1 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0000 0000 | 32, 119 |
| SPBRG | USART Baud Rate Generator | | | | | | | | 0000 0000 | 32, 185 |
| RCREG | USART Receive Register | | | | | | | | 0000 0000 | 32, 191 |
| TXREG | USART Transmit Register | | | | | | | | 0000 0000 | 32, 189 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 32, 183 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 32, 184 |
| EEADR | EEPROM Address Register | | | | | | | | xxxx xxxx | 32, 59 |
| EEDATA | EEPROM Data Register | | | | | | | | xxxx xxxx | 32, 59 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | xxxx xxxx | 32, 59 |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | $\overline{\text{WR}}$ | $\overline{\text{RD}}$ | xx-0 x000 | 32, 60, 67 |
| IPR3 | IRXIP | WAKIP | ERRIP | TXB2IP | TXB1IP | TXB0IP | RXB1IP | RXB0IP | 1111 1111 | 32, 90 |
| PIR3 | IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF | TXB0IF | RXB1IF | RXB0IF | 0000 0000 | 32, 84 |
| PIE3 | IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE | TXB0IE | RXB1IE | RXB0IE | 0000 0000 | 32, 87 |
| IPR2 | — | CMIP | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP(1) | -1-1 1111 | 32, 89 |
| PIR2 | — | CMIF | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF(1) | -0-0 0000 | 32, 83 |
| PIE2 | — | CMIE | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE(1) | -0-0 0000 | 32, 86 |

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

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

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

**3:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.



© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 32, 88 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 32, 82 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 32, 85 |
| TRISE(1) | IBF | OBF | IBOV | PSPMODE | — | Data Direction bits for PORTE(1) | | | 0000 -111 | 33, 105 |
| TRISD(1) | Data Direction Control Register for PORTD(1) | | | | | | | | 1111 1111 | 33, 102 |
| TRISC | Data Direction Control Register for PORTC | | | | | | | | 1111 1111 | 33, 100 |
| TRISB | Data Direction Control Register for PORTB | | | | | | | | 1111 1111 | 33, 96 |
| TRISA(3) | — | Data Direction Control Register for PORTA | | | | | | | -111 1111 | 33, 93 |
| LATE(1) | — | — | — | — | — | Read PORTE Data Latch, Write PORTE Data Latch(1) | | | ---- -xxx | 33, 104 |
| LATD(1) | Read PORTD Data Latch, Write PORTD Data Latch(1) | | | | | | | | xxxx xxxx | 33, 102 |
| LATC | Read PORTC Data Latch, Write PORTC Data Latch | | | | | | | | xxxx xxxx | 33, 100 |
| LATB | Read PORTB Data Latch, Write PORTB Data Latch | | | | | | | | xxxx xxxx | 33, 96 |
| LATA(3) | — | Read PORTA Data Latch, Write PORTA Data Latch | | | | | | | -xxx xxxx | 33, 93 |
| PORTE(1) | — | — | — | — | — | Read PORTE pins, Write PORTE Data Latch(1) | | | ---- -xxx | 33, 104 |
| PORTD(1) | Read PORTD pins, Write PORTD Data Latch(1) | | | | | | | | xxxx xxxx | 33, 102 |
| PORTC | Read PORTC pins, Write PORTC Data Latch | | | | | | | | xxxx xxxx | 33, 100 |
| PORTB | Read PORTB pins, Write PORTB Data Latch | | | | | | | | xxxx xxxx | 33, 96 |
| PORTA(3) | — | Read PORTA pins, Write PORTA Data Latch | | | | | | | -x0x 0000 | 33, 93 |
| TXERRCNT | TEC7 | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0 | 0000 0000 | 33, 209 |
| RXERRCNT | REC7 | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0 | 0000 0000 | 33, 214 |
| COMSTAT | RXB0OVFL | RXB1OVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN | 0000 0000 | 33, 205 |
| CIOCON | — | — | ENDRHI | CANCAP | — | — | — | — | --00 ---- | 33, 221 |
| BRGCON3 | — | WAKFIL | — | — | — | SEG2PH2 | SEG2PH1 | SEG2PH0 | -0-- -000 | 33, 220 |
| BRGCON2 | SEG2PHTS | SAM | SEG1PH2 | SEG1PH1 | SEG1PH0 | PRSEG2 | PRSEG1 | PRSEG0 | 0000 0000 | 33, 219 |
| BRGCON1 | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | 0000 0000 | 33, 218 |
| CANCON | REQOP2 | REQOP1 | REQOP0 | ABAT | WIN2 | WIN1 | WIN0 | — | xxxx xxx- | 33, 201 |
| CANSTAT | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | 33, 202 |
| RXB0D7 | RXB0D77 | RXB0D76 | RXB0D75 | RXB0D74 | RXB0D73 | RXB0D72 | RXB0D71 | RXB0D70 | xxxx xxxx | 33, 214 |
| RXB0D6 | RXB0D67 | RXB0D66 | RXB0D65 | RXB0D64 | RXB0D63 | RXB0D62 | RXB0D61 | RXB0D60 | xxxx xxxx | 33, 214 |
| RXB0D5 | RXB0D57 | RXB0D56 | RXB0D55 | RXB0D54 | RXB0D53 | RXB0D52 | RXB0D51 | RXB0D50 | xxxx xxxx | 33, 214 |
| RXB0D4 | RXB0D47 | RXB0D46 | RXB0D45 | RXB0D44 | RXB0D43 | RXB0D42 | RXB0D41 | RXB0D40 | xxxx xxxx | 33, 214 |
| RXB0D3 | RXB0D37 | RXB0D36 | RXB0D35 | RXB0D34 | RXB0D33 | RXB0D32 | RXB0D31 | RXB0D30 | xxxx xxxx | 33, 214 |
| RXB0D2 | RXB0D27 | RXB0D26 | RXB0D25 | RXB0D24 | RXB0D23 | RXB0D22 | RXB0D21 | RXB0D20 | xxxx xxxx | 33, 214 |
| RXB0D1 | RXB0D17 | RXB0D16 | RXB0D15 | RXB0D14 | RXB0D13 | RXB0D12 | RXB0D11 | RXB0D10 | xxxx xxxx | 33, 214 |
| RXB0D0 | RXB0D07 | RXB0D06 | RXB0D05 | RXB0D04 | RXB0D03 | RXB0D02 | RXB0D01 | RXB0D00 | xxxx xxxx | 33, 214 |
| RXB0DLC | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 34, 213 |
| RXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 34, 213 |
| RXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 34, 212 |
| RXB0SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 34, 212 |
| RXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 34, 212 |
| RXB0CON | RXFUL | RXM1 | RXM0 | — | RXRTRRO | RXB0DBEN | JTOFF | FILHIT0 | 000- 0000 | 34, 210 |

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

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

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

**3:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| CANSTATRO1 | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | 33, 202 |
| RXB1D7 | RXB1D77 | RXB1D76 | RXB1D75 | RXB1D74 | RXB1D73 | RXB1D72 | RXB1D71 | RXB1D70 | xxxx xxxx | 34, 214 |
| RXB1D6 | RXB1D67 | RXB1D66 | RXB1D65 | RXB1D64 | RXB1D63 | RXB1D62 | RXB1D61 | RXB1D60 | xxxx xxxx | 34, 214 |
| RXB1D5 | RXB1D57 | RXB1D56 | RXB1D55 | RXB1D54 | RXB1D53 | RXB1D52 | RXB1D51 | RXB1D50 | xxxx xxxx | 34, 214 |
| RXB1D4 | RXB1D47 | RXB1D46 | RXB1D45 | RXB1D44 | RXB1D43 | RXB1D42 | RXB1D41 | RXB1D40 | xxxx xxxx | 34, 214 |
| RXB1D3 | RXB1D37 | RXB1D36 | RXB1D35 | RXB1D34 | RXB1D33 | RXB1D32 | RXB1D31 | RXB1D30 | xxxx xxxx | 34, 214 |
| RXB1D2 | RXB1D27 | RXB1D26 | RXB1D25 | RXB1D24 | RXB1D23 | RXB1D22 | RXB1D21 | RXB1D20 | xxxx xxxx | 34, 214 |
| RXB1D1 | RXB1D17 | RXB1D16 | RXB1D15 | RXB1D14 | RXB1D13 | RXB1D12 | RXB1D11 | RXB1D10 | xxxx xxxx | 34, 214 |
| RXB1D0 | RXB1D07 | RXB1D06 | RXB1D05 | RXB1D04 | RXB1D03 | RXB1D02 | RXB1D01 | RXB1D00 | xxxx xxxx | 34, 214 |
| RXB1DLC | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 34, 213 |
| RXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 34, 213 |
| RXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 34, 212 |
| RXB1SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 34, 212 |
| RXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 34, 212 |
| RXB1CON | RXFUL | RXM1 | RXM0 | — | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 | 000- 0000 | 34, 211 |
| CANSTATRO2 | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | 33, 202 |
| TXB0D7 | TXB0D77 | TXB0D76 | TXB0D75 | TXB0D74 | TXB0D73 | TXB0D72 | TXB0D71 | TXB0D70 | xxxx xxxx | 34, 208 |
| TXB0D6 | TXB0D67 | TXB0D66 | TXB0D65 | TXB0D64 | TXB0D63 | TXB0D62 | TXB0D61 | TXB0D60 | xxxx xxxx | 34, 208 |
| TXB0D5 | TXB0D57 | TXB0D56 | TXB0D55 | TXB0D54 | TXB0D53 | TXB0D52 | TXB0D51 | TXB0D50 | xxxx xxxx | 34, 208 |
| TXB0D4 | TXB0D47 | TXB0D46 | TXB0D45 | TXB0D44 | TXB0D43 | TXB0D42 | TXB0D41 | TXB0D40 | xxxx xxxx | 34, 208 |
| TXB0D3 | TXB0D37 | TXB0D36 | TXB0D35 | TXB0D34 | TXB0D33 | TXB0D32 | TXB0D31 | TXB0D30 | xxxx xxxx | 34, 208 |
| TXB0D2 | TXB0D27 | TXB0D26 | TXB0D25 | TXB0D24 | TXB0D23 | TXB0D22 | TXB0D21 | TXB0D20 | xxxx xxxx | 34, 208 |
| TXB0D1 | TXB0D17 | TXB0D16 | TXB0D15 | TXB0D14 | TXB0D13 | TXB0D12 | TXB0D11 | TXB0D10 | xxxx xxxx | 34, 208 |
| TXB0D0 | TXB0D07 | TXB0D06 | TXB0D05 | TXB0D04 | TXB0D03 | TXB0D02 | TXB0D01 | TXB0D00 | xxxx xxxx | 34, 208 |
| TXB0DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 34, 209 |
| TXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 34, 208 |
| TXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 34, 207 |
| TXB0SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 34, 207 |
| TXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 35, 207 |
| TXB0CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | -000 0-00 | 35, 206 |
| CANSTATRO3 | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | 33, 202 |
| TXB1D7 | TXB1D77 | TXB1D76 | TXB1D75 | TXB1D74 | TXB1D73 | TXB1D72 | TXB1D71 | TXB1D70 | xxxx xxxx | 35, 208 |
| TXB1D6 | TXB1D67 | TXB1D66 | TXB1D65 | TXB1D64 | TXB1D63 | TXB1D62 | TXB1D61 | TXB1D60 | xxxx xxxx | 35, 208 |
| TXB1D5 | TXB1D57 | TXB1D56 | TXB1D55 | TXB1D54 | TXB1D53 | TXB1D52 | TXB1D51 | TXB1D50 | xxxx xxxx | 35, 208 |
| TXB1D4 | TXB1D47 | TXB1D46 | TXB1D45 | TXB1D44 | TXB1D43 | TXB1D42 | TXB1D41 | TXB1D40 | xxxx xxxx | 35, 208 |
| TXB1D3 | TXB1D37 | TXB1D36 | TXB1D35 | TXB1D34 | TXB1D33 | TXB1D32 | TXB1D31 | TXB1D30 | xxxx xxxx | 35, 208 |
| TXB1D2 | TXB1D27 | TXB1D26 | TXB1D25 | TXB1D24 | TXB1D23 | TXB1D22 | TXB1D21 | TXB1D20 | xxxx xxxx | 35, 208 |
| TXB1D1 | TXB1D17 | TXB1D16 | TXB1D15 | TXB1D14 | TXB1D13 | TXB1D12 | TXB1D11 | TXB1D10 | xxxx xxxx | 35, 208 |
| TXB1D0 | TXB1D07 | TXB1D06 | TXB1D05 | TXB1D04 | TXB1D03 | TXB1D02 | TXB1D01 | TXB1D00 | xxxx xxxx | 35, 208 |
| TXB1DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 35, 209 |
| TXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 35, 208 |
| TXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 35, 207 |
| TXB1SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 35, 207 |
| TXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 35, 207 |
| TXB1CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0000 | 35, 206 |

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

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

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

**3:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.



© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| CANSTATRO4 | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx- xxx- | 33, 202 |
| TXB2D7 | TXB2D77 | TXB2D76 | TXB2D75 | TXB2D74 | TXB2D73 | TXB2D72 | TXB2D71 | TXB2D70 | xxxx xxxx | 35, 208 |
| TXB2D6 | TXB2D67 | TXB2D66 | TXB2D65 | TXB2D64 | TXB2D63 | TXB2D62 | TXB2D61 | TXB2D60 | xxxx xxxx | 35, 208 |
| TXB2D5 | TXB2D57 | TXB2D56 | TXB2D55 | TXB2D54 | TXB2D53 | TXB2D52 | TXB2D51 | TXB2D50 | xxxx xxxx | 35, 208 |
| TXB2D4 | TXB2D47 | TXB2D46 | TXB2D45 | TXB2D44 | TXB2D43 | TXB2D42 | TXB2D41 | TXB2D40 | xxxx xxxx | 35, 208 |
| TXB2D3 | TXB2D37 | TXB2D36 | TXB2D35 | TXB2D34 | TXB2D33 | TXB2D32 | TXB2D31 | TXB2D30 | xxxx xxxx | 35, 208 |
| TXB2D2 | TXB2D27 | TXB2D26 | TXB2D25 | TXB2D24 | TXB2D23 | TXB2D22 | TXB2D21 | TXB2D20 | xxxx xxxx | 35, 208 |
| TXB2D1 | TXB2D17 | TXB2D16 | TXB2D15 | TXB2D14 | TXB2D13 | TXB2D12 | TXB2D11 | TXB2D10 | xxxx xxxx | 35, 208 |
| TXB2D0 | TXB2D07 | TXB2D06 | TXB2D05 | TXB2D04 | TXB2D03 | TXB2D02 | TXB2D01 | TXB2D00 | xxxx xxxx | 35, 208 |
| TXB2DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 35, 209 |
| TXB2EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 35, 208 |
| TXB2EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 35, 207 |
| TXB2SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 35, 207 |
| TXB2SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 35, 207 |
| TXB2CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | -000 0-00 | 35, 206 |
| RXM1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 35, 217 |
| RXM1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 35, 217 |
| RXM1SIDL | SID2 | SID1 | SID0 | — | — | — | EID17 | EID16 | xxx- --xx | 36, 217 |
| RXM1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 216 |
| RXM0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 36, 217 |
| RXM0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 36, 217 |
| RXM0SIDL | SID2 | SID1 | SID0 | — | — | — | EID17 | EID16 | xxx- --xx | 36, 217 |
| RXM0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 216 |
| RXF5EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 36, 216 |
| RXF5EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 36, 216 |
| RXF5SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 36, 215 |
| RXF5SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 215 |
| RXF4EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 36, 216 |
| RXF4EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 36, 216 |
| RXF4SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 36, 215 |
| RXF4SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 215 |
| RXF3EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 36, 216 |
| RXF3EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 36, 216 |
| RXF3SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 36, 215 |
| RXF3SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 215 |
| RXF2EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 36, 216 |
| RXF2EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 36, 216 |
| RXF2SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 36, 215 |
| RXF2SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 215 |
| RXF1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 36, 216 |
| RXF1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 36, 216 |
| RXF1SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 36, 215 |
| RXF1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 215 |
| RXF0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 36, 216 |
| RXF0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 36, 216 |
| RXF0SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 36, 215 |
| RXF0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 36, 215 |

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

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

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

**3:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

chanhvinh.com



## 4.10 Access Bank

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

This data memory region can be used for:

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

The Access Bank is comprised of the upper 160 bytes in Bank 15 (SFRs) 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 4-6 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.

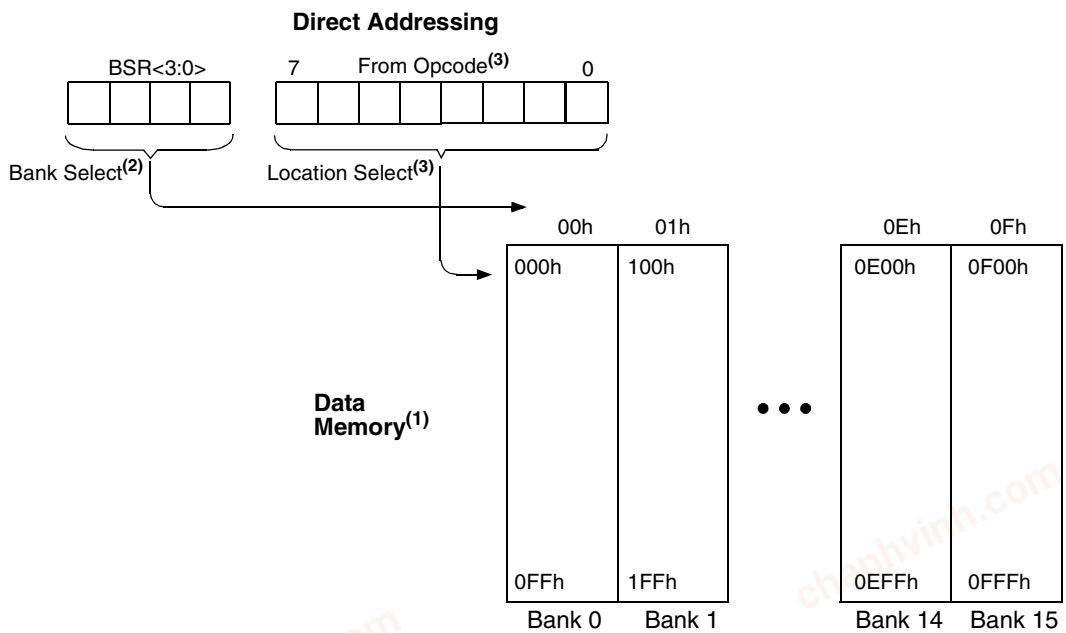
If the currently selected bank is not implemented, any read will return all '0's and all writes are ignored. The Status register bits will be set/cleared as appropriate for the instruction performed.

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 "Indirect Addressing, INDF and FSR Registers"** provides a description of indirect addressing, which allows linear addressing of the entire RAM space.

**FIGURE 4-7: DIRECT ADDRESSING**




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

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

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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 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 4-8 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 4-8.

The INDFn ($0 \le n \le 2$) register is not a physical register. Addressing INDFn actually addresses the register whose address is contained in the FSRn register (FSRn is a pointer). This is indirect addressing.

Example 4-5 shows a simple use of indirect addressing to clear the RAM in Bank 1 (locations 100h-1FFh) in a minimum number of instructions.

**EXAMPLE 4-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING**

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

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

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

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

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

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

### 4.12.1 INDIRECT ADDRESSING OPERATION

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

- When data access is done to one of the five INDFn locations, the address selected will configure the FSRn register to:
  - Do nothing to FSRn after an indirect access (no change) – INDFn
  - Auto-decrement FSRn after an indirect access (post-decrement) – POSTDECn
  - Auto-increment FSRn after an indirect access (post-increment) – POSTINCn
  - Auto-increment FSRn before an indirect access (pre-increment) – PREINCn
  - Use the value in the WREG register as an offset to FSRn. Do not modify the value of the WREG or the FSRn register after an indirect access (no change) – PLUSWn

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

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

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

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

If an FSR register contains a value that indicates one of the INDFn, 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).

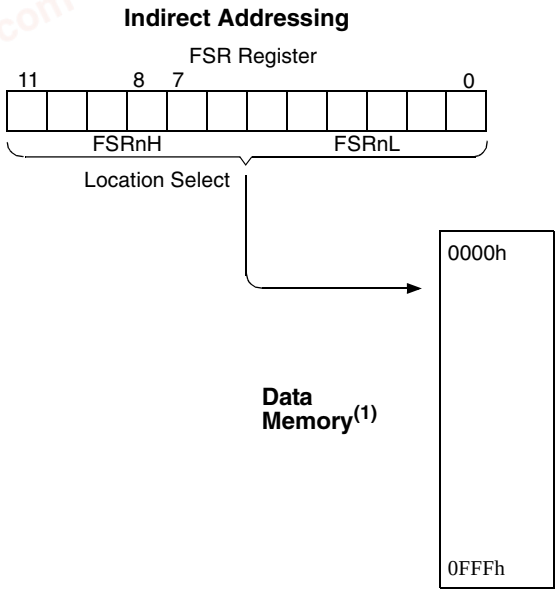
If an indirect addressing operation is done where the target address is an FSRnH or FSRnL register, the write operation will dominate over the pre- or post-increment/decrement functions.

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 4-8: INDIRECT ADDRESSING**




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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 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 25-2.

**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 2's complement of the second operand. For rotate (`RRCF`, `RRNCF`, `RLCF` and `RLNCF`) instructions, this bit is loaded with either 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 2'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 |

chanhvinh.com



## 4.14 RCON Register

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

**Note 1:** If the BOREN configuration bit is set, $\overline{BOR}$ is '1' on Power-on Reset. If the BOREN configuration bit is clear, $\overline{BOR}$ is unknown on Power-on Reset.
The $\overline{BOR}$ 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). $\overline{BOR}$ 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 $\overline{POR}$ bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

**REGISTER 4-3: RCON: RESET CONTROL REGISTER**

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

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

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

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

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

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

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

bit 0 **$\overline{BOR}$:** 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



## 5.0 DATA EEPROM MEMORY

The data EEPROM is readable and writable during normal operation over the entire VDD range. The data memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers (SFR).

There are four SFRs used to read and write the program and data EEPROM memory. These registers are:

- EECON1
- EECON2
- EEDATA
- EEADR

The EEPROM data memory allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. The PIC18FXX8 devices have 256 bytes of data EEPROM with an address range from 00h to FFh.

The EEPROM data memory is rated for high erase/write cycles. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature, as well as from chip-to-chip. Please refer to the specifications for exact limits.

### 5.1 EEADR Register

The address register can address up to a maximum of 256 bytes of data EEPROM.

### 5.2 EECON1 and EECON2 Registers

EECON1 is the control register for EEPROM memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the EEPROM write sequence.

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

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{MCLR}$ Reset, or a WDT Time-out Reset, during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR) due to the Reset condition forcing the contents of the registers to zero.

| **Note:** | Interrupt flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in software. |
|---|---|

chanhvinh.com



**REGISTER 5-1: EECON1: EEPROM CONTROL REGISTER 1**

| R/W-x | R/W-x | U-0 | R/W-0 | R/W-x | R/W-0 | R/S-0 | R/S-0 |
|---|---|---|---|---|---|---|---|
| EEPGD | CFGS | — | FREE | WRERR | WREN | $\overline{WR}$ | $\overline{RD}$ |
| bit 7 | | | | | | | bit 0 |

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
1 = Access program Flash memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EE or Configuration Select bit
1 = Access Configuration registers
0 = Access program Flash or data EEPROM memory

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

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next $\overline{WR}$ command (reset by hardware)
0 = Perform write only

bit 3 **WRERR:** Write Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{MCLR}$ or any WDT Reset during self-timed programming in normal operation)
0 = The write operation completed

**Note:** When a WRERR occurs, the EEPGD or FREE bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM or Flash memory

bit 1 **$\overline{WR}$:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle (The operation is self-timed and the bit is cleared by hardware once write is complete. The $\overline{WR}$ bit can only be set (not cleared) in software.)
0 = Write cycle is complete

bit 0 **$\overline{RD}$:** Read Control bit
1 = Initiates an EEPROM read (Read takes one cycle. $\overline{RD}$ is cleared in hardware. The $\overline{RD}$ bit can only be set (not cleared) in software. $\overline{RD}$ bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 5.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD and CFGS control bits (EECON1<7:6>) and then set control bit $\overline{RD}$ (EECON1<0>). The data is available in the very next instruction cycle of the EEDATA register; therefore, it can be read by the next instruction. EEDATA will hold this value until another read operation or until it is written to by the user (during a write operation).

**EXAMPLE 5-1: DATA EEPROM READ**

```
MOVLW   DATA_EE_ADDR    ;
MOVWF   EEADR           ;Data Memory Address
                        ;to read
BCF     EECON1, EEPGD   ;Point to DATA memory
BCS     EECON1, CFGS    ;
BSF     EECON1, RD      ;EEPROM Read
MOVF    EEDATA, W       ;W = EEDATA
```

## 5.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADR register and the data written to the EEDATA register. Then, the sequence in Example 5-2 must be followed to initiate the write cycle.

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write 0AAh to EECON2, then set $\overline{WR}$ bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, clearing the WREN bit will not affect the current write cycle. The $\overline{WR}$ bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both $\overline{WR}$ and WREN cannot be set with the same instruction.

At the completion of the write cycle, the $\overline{WR}$ bit is cleared in hardware and the EEPROM Write Complete Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt or roll this bit. EEIF must be cleared by software.

**EXAMPLE 5-2: DATA EEPROM WRITE**

```
                MOVLW   DATA_EE_ADDR    ;
                MOVWF   EEADR           ; Data Memory Address to read
                MOVLW   DATA_EE_DATA    ;
                MOVWF   EEDATA          ; Data Memory Value to write
                BCF     EECON1, EEPGD   ; Point to DATA memory
                BCF     EECON1, CFGS    ; Access program FLASH or Data EEPROM memory
                BSF     EECON1, WREN    ; Enable writes

                BCF     INTCON, GIE     ; Disable interrupts
  Required      MOVLW   55h             ;
  Sequence      MOVWF   EECON2          ; Write 55h
                MOVLW   0AAh            ;
                MOVWF   EECON2          ; Write AAh
                BSF     EECON1, WR      ; Set WR bit to begin write
                BSF     INTCON, GIE     ; Enable interrupts

                .                       ; user code execution
                .
                .
                BCF     EECON1, WREN    ; Disable writes on write complete (EEIF set)
```

© 2006 Microchip Technology Inc.

chanhvinh.com



## 5.5 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

Generally, a write failure will be a bit which was written as a '1', but reads back as a '0' (due to leakage off the cell).

## 5.6 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, the WREN bit is cleared. Also, the Power-up Timer (72 ms duration) prevents EEPROM write.

The write initiate sequence and the WREN bit together reduce the probability of an accidental write during brown-out, power glitch or software malfunction.

## 5.7 Operation During Code-Protect

Data EEPROM memory has its own code-protect mechanism. External read and write operations are disabled if either of these mechanisms are enabled.

The microcontroller itself can both read and write to the internal data EEPROM, regardless of the state of the code-protect configuration bit. Refer to **Section 24.0 "Special Features of the CPU"** for additional information.

## 5.8 Using the Data EEPROM

The data EEPROM is a high-endurance, byte address-able array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). Frequently changing values will typically be updated more often than specification D124 or D124A. If this is not the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory. A simple data EEPROM refresh routine is shown in Example 5-3.

| **Note:** | If data EEPROM is only used to store constants and/or data that changes rarely, an array refresh is likely not required. See specification D124 or D124A. |
|---|---|

**EXAMPLE 5-3: DATA EEPROM REFRESH ROUTINE**

```
        CLRF    EEADR           ; Start at address 0
        BCF     EECON1, CFGS    ; Set for memory
        BCF     EECON1, EEPGD   ; Set for Data EEPROM
        BCF     INTCON, GIE     ; Disable interrupts
        BSF     EECON1, WREN    ; Enable writes
Loop                            ; Loop to refresh array
        BSF     EECON1, RD      ; Read current address
        MOVLW   55h             ;
        MOVWF   EECON2          ; Write 55h
        MOVLW   0AAh            ;
        MOVWF   EECON2          ; Write AAh
        BSF     EECON1, WR      ; Set WR bit to begin write
        BTFSC   EECON1, WR      ; Wait for write to complete
        BRA     $-2

        INCFSZ  EEADR, F        ; Increment address
        BRA     Loop            ; Not zero, do it again

        BCF     EECON1, WREN    ; Disable writes
        BSF     INTCON, GIE     ; Enable interrupts
```


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 5-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY**

| 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 |
| EEADR | EEPROM Address Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| EEDATA | EEPROM Data Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | — | — |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | $\overline{WR}$ | $\overline{RD}$ | xx-0 x000 | uu-0 u000 |
| IPR2 | — | CMIP | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP(1) | -1-1 1111 | -1-1 1111 |
| PIR2 | — | CMIF | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF(1) | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE(1) | -0-0 0000 | -0-0 0000 |

**Legend:** x = unknown, u = unchanged, r = reserved, - = unimplemented, read as '0'.
Shaded cells are not used during Flash/EEPROM access.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

chanhvinh.com

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 6.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 8 bytes at a time. Program memory is erased in blocks of 64 bytes at a time. A bulk erase operation may not be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

### 6.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

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

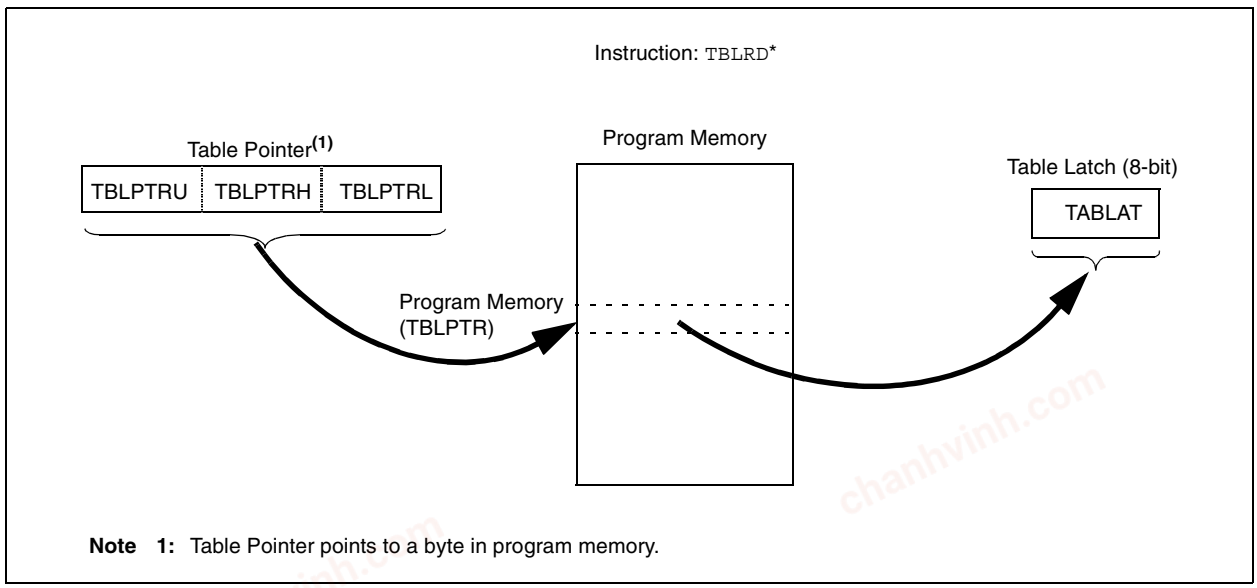
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

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

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 6.5 "Writing to Flash Program Memory"**. Figure 6-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word aligned.

**FIGURE 6-1: TABLE READ OPERATION**

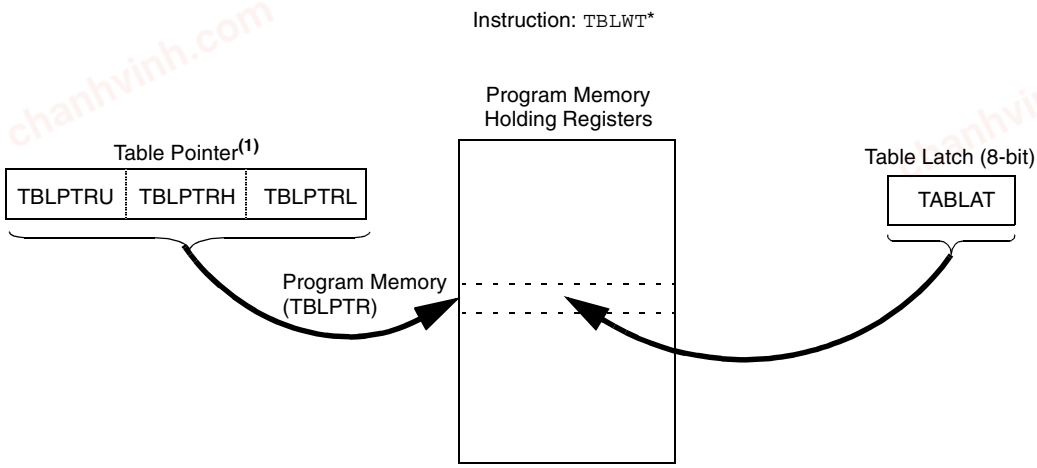



**Note 1:** Table Pointer points to a byte in program memory.

chanhvinh.com



**FIGURE 6-2: TABLE WRITE OPERATION**




**Note 1:** Table Pointer actually points to one of eight holding registers, the address of which is determined by TBLPTRL<2:0>. The process for physically writing data to the program memory array is discussed in **Section 6.5 "Writing to Flash Program Memory"**.

## 6.2 Control Registers

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

- EECON1 register
- EECON2 register
- TABLAT register
- TBLPTR registers

### 6.2.1 EECON1 AND EECON2 REGISTERS

EECON1 is the control register for memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the memory write and erase sequences.

Control bit EEPGD determines if the access will be a program or data EEPROM memory access. When clear, any subsequent operations will operate on the data EEPROM memory. When set, any subsequent operations will operate on the program memory.

Control bit CFGS determines if the access will be to the Configuration/Calibration registers or to program memory/data EEPROM memory. When set, subsequent operations will operate on Configuration registers regardless of EEPGD (see **Section 24.0 "Special Features of the CPU"**). When clear, memory selection access is determined by EEPGD.

The FREE bit, when set, will allow a program memory erase operation. When the FREE bit is set, the erase operation is initiated on the next $\overline{WR}$ command. When FREE is clear, only writes are enabled.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{MCLR}$ Reset or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR) due to Reset values of zero.

Control bits, $\overline{RD}$ and $\overline{WR}$, initiate read and write operations, respectively. These bits cannot be cleared, only set, in software. They are cleared in hardware at the completion of the read or write operation. The inability to clear the $\overline{WR}$ bit in software prevents the accidental or premature termination of a write operation. The $\overline{RD}$ bit cannot be set when accessing program memory (EEPGD = 1).

| **Note:** | Interrupt flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in software. |
|---|---|


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 6-1: EECON1: EEPROM CONTROL REGISTER 1**

| R/W-x | R/W-x | U-0 | R/W-0 | R/W-x | R/W-0 | R/S-0 | R/S-0 |
|---|---|---|---|---|---|---|---|
| EEPGD | CFGS | — | FREE | WRERR | WREN | $\overline{WR}$ | $\overline{RD}$ |
| bit 7 | | | | | | | bit 0 |

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
1 = Access program Flash memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EE or Configuration Select bit
1 = Access Configuration registers
0 = Access program Flash or data EEPROM memory

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

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next $\overline{WR}$ command (cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Write Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{MCLR}$ or any WDT Reset during self-timed programming in normal operation)
0 = The write operation completed

**Note:** When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM or Flash memory

bit 1 **$\overline{WR}$:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle (The operation is self-timed and the bit is cleared by hardware once write is complete. The $\overline{WR}$ bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **$\overline{RD}$:** Read Control bit
1 = Initiates an EEPROM read (Read takes one cycle. $\overline{RD}$ is cleared in hardware. The $\overline{RD}$ bit can only be set (not cleared) in software. $\overline{RD}$ bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

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

© 2006 Microchip Technology Inc.

chanhvinh.com



### 6.2.2 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 RAM.

### 6.2.3 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, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers 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 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 6-1. These operations on the TBLPTR only affect the low-order 21 bits.

### 6.2.4 TABLE POINTER BOUNDARIES

TBLPTR is used in reads, writes and erases of the Flash program memory.

When a `TBLRD` is executed, all 22 bits of the Table Pointer determine which byte is read from program memory into TABLAT.

When a `TBLWT` is executed, the three LSbs of the Table Pointer (TBLPTR<2:0>) determine which of the eight program memory holding registers is written to. When the timed write to program memory (long write) begins, the 19 MSbs of the Table Pointer, TBLPTR (TBLPTR<21:3>), will determine which program memory block of 8 bytes is written to. For more detail, see **Section 6.5 "Writing to Flash Program Memory"**.

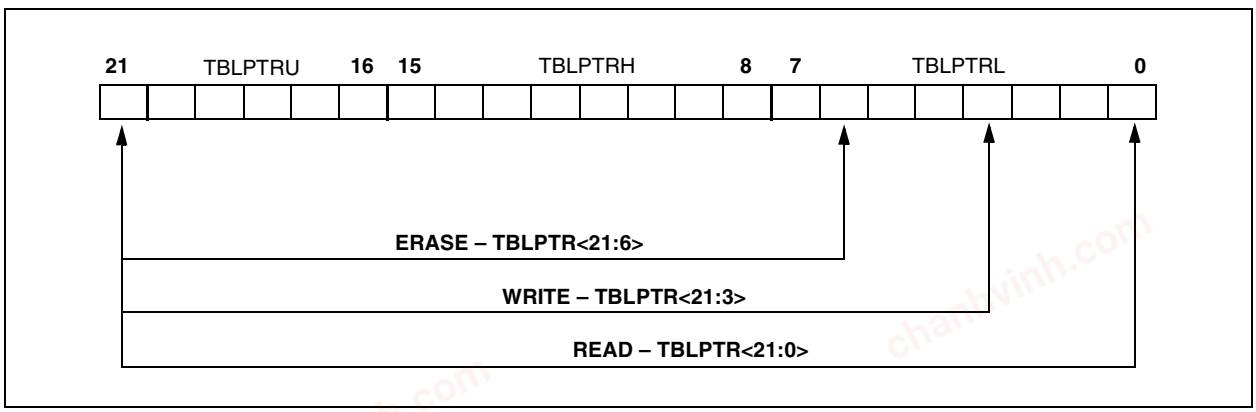
When an erase of program memory is executed, the 16 MSbs of the Table Pointer (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 6-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

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

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

**FIGURE 6-3: TABLE POINTER BOUNDARIES BASED ON OPERATION**





© 2006 Microchip Technology Inc.

chanhvinh.com



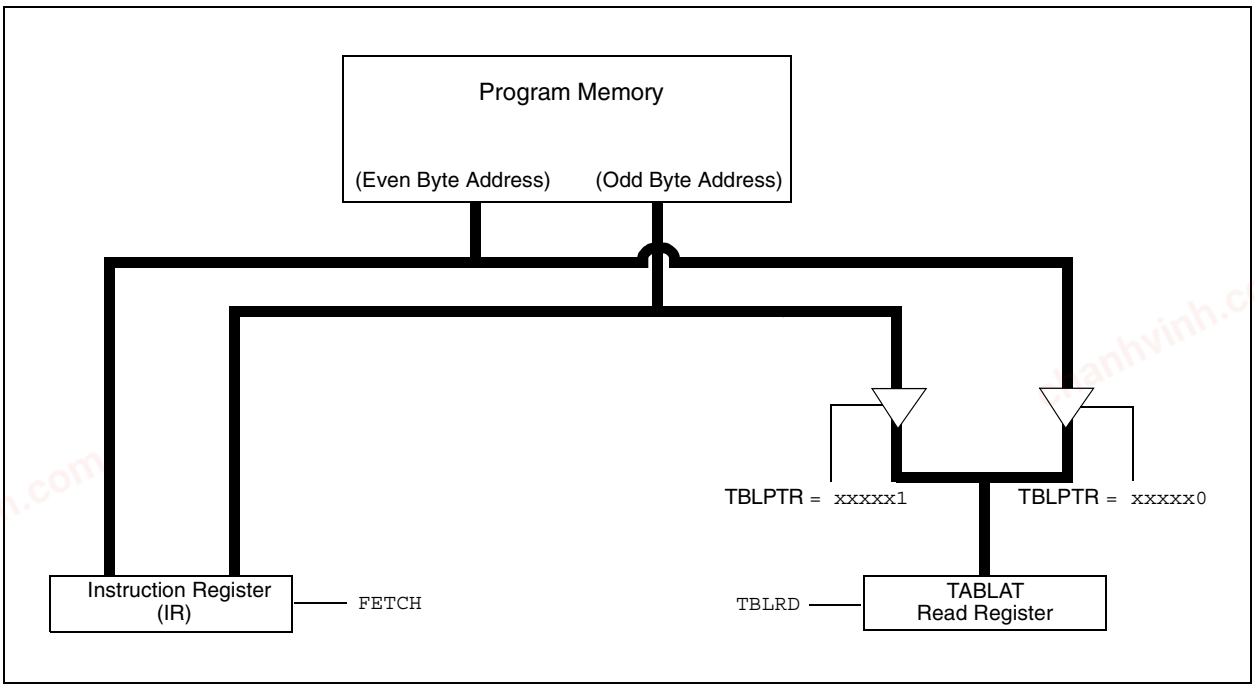
## 6.3 Reading the Flash Program Memory

The TBLRD instruction is used to retrieve data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

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.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 6-4 shows the interface between the internal program memory and the TABLAT.

**FIGURE 6-4: READS FROM FLASH PROGRAM MEMORY**




**EXAMPLE 6-1: READING A FLASH PROGRAM MEMORY WORD**

```
            MOVLW   CODE_ADDR_UPPER          ; Load TBLPTR with the base
            MOVWF   TBLPTRU                  ; address of the word
            MOVLW   CODE_ADDR_HIGH
            MOVWF   TBLPTRH
            MOVLW   CODE_ADDR_LOW
            MOVWF   TBLPTRL
READ_WORD
            TBLRD*+                          ; read into TABLAT and increment
            MOVF    TABLAT, W                ; get data
            MOVWF   WORD_LSB
            TBLRD*+                          ; read into TABLAT and increment
            MOVF    TABLAT, W                ; get data
            MOVWF   WORD_MSB
```

chanhvinh.com



## 6.4 Erasing Flash Program Memory

The minimum erase block is 32 words or 64 bytes. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the micro-controller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. TBLPTR<5:0> are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

For protection, the write initiate sequence for EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

### 6.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load Table Pointer with address of row being erased.
2. Set the EECON1 register for the erase operation:
   - set the EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set the WREN bit to enable writes;
   - set the FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the $\overline{WR}$ bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Re-enable interrupts.

**EXAMPLE 6-2: ERASING A FLASH PROGRAM MEMORY ROW**

```
                MOVLW   upper (CODE_ADDR)      ; load TBLPTR with the base
                MOVWF   TBLPTRU                ; address of the memory block
                MOVLW   high (CODE_ADDR)
                MOVWF   TBLPTRH
                MOVLW   low (CODE_ADDR)
                MOVWF   TBLPTRL
    ERASE_ROW
                BSF     EECON1, EEPGD          ; point to FLASH program memory
                BCF     EECON1, CFGS           ; access FLASH program memory
                BSF     EECON1, WREN           ; enable write to memory
                BSF     EECON1, FREE           ; enable Row Erase operation
                BCF     INTCON, GIE            ; disable interrupts
                MOVLW   55h
                MOVWF   EECON2                 ; write 55H
Required        MOVLW   0AAh
Sequence        MOVWF   EECON2                 ; write 0AAH
                BSF     EECON1, WR             ; start erase (CPU stall)
                NOP                            ; NOP needed for proper code execution
                BSF     INTCON, GIE            ; re-enable interrupts
```


© 2006 Microchip Technology Inc.

chanhvinh.com



## 6.5 Writing to Flash Program Memory

The minimum programming block is 4 words or 8 bytes. Word or byte programming is not supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. There are 8 holding registers used by the table writes for programming.

Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction has to be executed 8 times for each programming operation. All of the table write operations will essentially be short writes, because only the holding registers are written. At the end of updating 8 registers, the EECON1 register must be written to, to start the programming operation with a long write.

The long write is necessary for programming the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

The EEPROM on-chip timer controls the write time. The write/erase voltages are generated by an on-chip charge pump rated to operate over the voltage range of the device for byte or word operations.

### 6.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

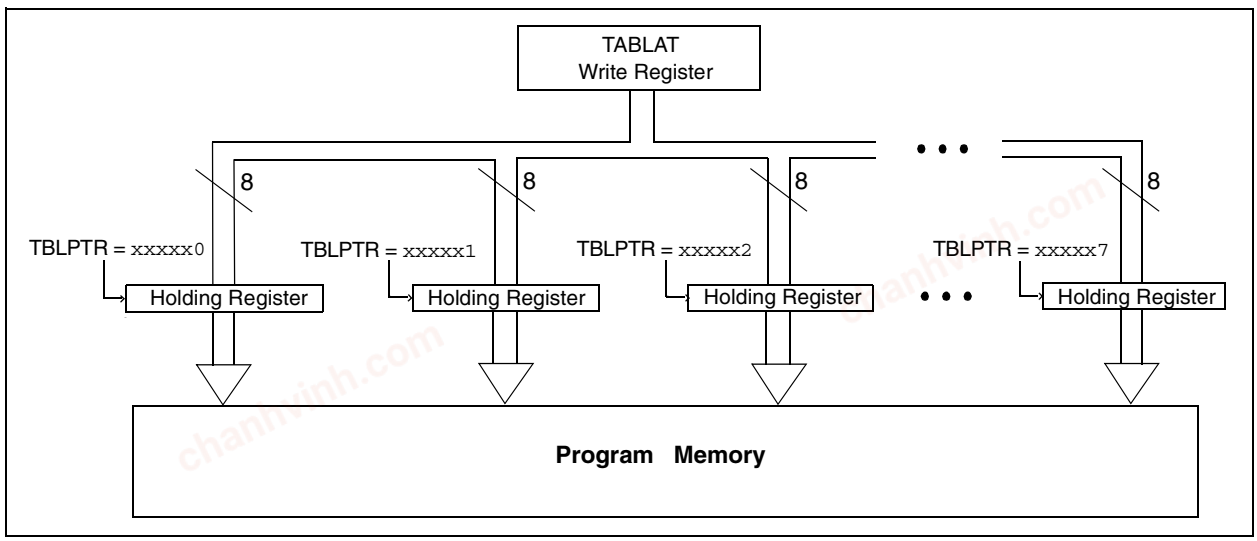
The sequence of events for programming an internal program memory location should be:

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer with address being erased.
4. Do the row erase procedure.
5. Load Table Pointer with address of first byte being written.
6. Write the first 8 bytes into the holding registers using the TBLWT instruction, auto-increment may be used.
7. Set the EECON1 register for the write operation:
   - set the EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set the WREN to enable byte writes.
8. Disable interrupts.
9. Write 55h to EECON2.
10. Write AAh to EECON2.
11. Set the $\overline{WR}$ bit. This will begin the write cycle.
12. The CPU will stall for duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Repeat steps 6-14 seven times to write 64 bytes.
15. Verify the memory (table read).

This procedure will require about 18 ms to update one row of 64 bytes of memory. An example of the required code is given in Example 6-3.

| **Note:** | Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the 8 bytes in the holding registers. |
|---|---|

**FIGURE 6-5: TABLE WRITES TO FLASH PROGRAM MEMORY**



chanhvinh.com



**EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY**

```
                MOVLW   D'64                    ; number of bytes in erase block
                MOVWF   COUNTER
                MOVLW   high (BUFFER_ADDR)      ; point to buffer
                MOVWF   FSR0H
                MOVLW   low (BUFFER_ADDR)
                MOVWF   FSR0L
                MOVLW   upper (CODE_ADDR)       ; Load TBLPTR with the base
                MOVWF   TBLPTRU                 ; address of the memory block
                MOVLW   high (CODE_ADDR)
                MOVWF   TBLPTRH
                MOVLW   low (CODE_ADDR)
                MOVWF   TBLPTRL
READ_BLOCK
                TBLRD*+                         ; read into TABLAT, and inc
                MOVF    TABLAT, W               ; get data
                MOVWF   POSTINC0                ; store data
                DECFSZ  COUNTER                 ; done?
                BRA     READ_BLOCK              ; repeat
MODIFY_WORD
                MOVLW   DATA_ADDR_HIGH          ; point to buffer
                MOVWF   FSR0H
                MOVLW   DATA_ADDR_LOW
                MOVWF   FSR0L
                MOVLW   NEW_DATA_LOW            ; update buffer word
                MOVWF   POSTINC0
                MOVLW   NEW_DATA_HIGH
                MOVWF   INDF0
ERASE_BLOCK
                MOVLW   upper (CODE_ADDR)       ; load TBLPTR with the base
                MOVWF   TBLPTRU                 ; address of the memory block
                MOVLW   high (CODE_ADDR)
                MOVWF   TBLPTRH
                MOVLW   low (CODE_ADDR)
                MOVWF   TBLPTRL
                BSF     EECON1, EEPGD           ; point to FLASH program memory
                BCF     EECON1, CFGS            ; access FLASH program memory
                BSF     EECON1, WREN            ; enable write to memory
                BSF     EECON1, FREE            ; enable Row Erase operation
                BCF     INTCON, GIE             ; disable interrupts
                MOVLW   55h
Required        MOVWF   EECON2                  ; write 55H
Sequence        MOVLW   0AAh
                MOVWF   EECON2                  ; write AAH
                BSF     EECON1, WR              ; start erase (CPU stall)
                NOP
                BSF     INTCON, GIE             ; re-enable interrupts
                TBLRD*-                         ; dummy read decrement
WRITE_BUFFER_BACK
                MOVLW   8                       ; number of write buffer groups of 8 bytes
                MOVWF   COUNTER_HI
                MOVLW   high (BUFFER_ADDR)      ; point to buffer
                MOVWF   FSR0H
                MOVLW   low (BUFFER_ADDR)
                MOVWF   FSR0L
PROGRAM_LOOP
                MOVLW   8                       ; number of bytes in holding register
                MOVWF   COUNTER
WRITE_WORD_TO_HREGS
                MOVFW   POSTINC0, W             ; get low byte of buffer data
                MOVWF   TABLAT                  ; present data to table latch
                TBLWT+*                         ; write data, perform a short write
                                                ; to internal TBLWT holding register.
                DECFSZ  COUNTER                 ; loop until buffers are full
                BRA     WRITE_WORD_TO_HREGS
```


© 2006 Microchip Technology Inc.

chanhvinh.com



**EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)**

```
WRITE_WORD_TO_HREGS
                MOVFW   POSTINC0, W             ; get low byte of buffer data
                MOVWF   TABLAT                  ; present data to table latch
                TBLWT+*                         ; write data, perform a short write
                                                ; to internal TBLWT holding register.
                DECFSZ  COUNTER                 ; loop until buffers are full
                BRA     WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
                BSF     EECON1, EEPGD           ; point to FLASH program memory
                BCF     EECON1, CFGS            ; access FLASH program memory
                BSF     EECON1, WREN            ; enable write to memory
                BCF     INTCON, GIE             ; disable interrupts
                MOVLW   55h                     ; write 55h
Required        MOVWF   EECON2
Sequence        MOVLW   0AAh                    ; write 0AAh
                MOVWF   EECON2                  ; start program (CPU stall)
                BSF     EECON1, WR
                NOP
                BSF     INTCON, GIE             ; re-enable interrupts
                DECFSZ  COUNTER_HI              ; loop until done
                BRA     PROGRAM_LOOP
                BCF     EECON1, WREN            ; disable write to memory
```

### 6.5.2 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

### 6.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed.The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation. In these situations, users can check the WRERR bit and rewrite the location.

### 6.5.4 PROTECTION AGAINST SPURIOUS WRITES

To reduce the probability against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 24.0 "Special Features of the CPU"** for more detail.

## 6.6 Flash Program Operation During Code Protection

See **Section 24.0 "Special Features of the CPU"** for details on code protection of Flash program memory.

chanhvinh.com



**TABLE 6-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|
| TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 | --00 0000 |
| TBPLTRH | 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 |
| INTCON | GIE/GIEH | PEIE/ GIEL | TMR0IE | INTE | RBIE | TMR0IF | INTF | RBIF | 0000 000x | 0000 000u |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | — | — |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | $\overline{WR}$ | $\overline{RD}$ | xx-0 x000 | uu-0 u000 |
| IPR2 | — | CMIP | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP(1) | -1-1 1111 | -1-1 1111 |
| PIR2 | — | CMIF | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF(1) | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE(1) | -0-0 0000 | -0-0 0000 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, read as '0'.
Shaded cells are not used during Flash/EEPROM access.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.


© 2006 Microchip Technology Inc.

chanhvinh.com



# 7.0 8 x 8 HARDWARE MULTIPLIER

## 7.1 Introduction

An 8 x 8 hardware multiplier is included in the ALU of the PIC18FXX8 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 in the 16-bit product register pair (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA 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 7-1 shows a performance comparison between Enhanced devices using the single-cycle hardware multiply and performing the same function without the hardware multiply.

## 7.2 Operation

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

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

**EXAMPLE 7-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE**

```
MOVF    ARG1, W     ;
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ;   PRODH:PRODL
```

**EXAMPLE 7-2: 8 x 8 SIGNED MULTIPLY ROUTINE**

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

**TABLE 7-1: 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 µs | 27.6 µs | 69 µs |
| | Hardware multiply | 1 | 1 | 100 ns | 400 ns | 1 µs |
| 8 x 8 signed | Without hardware multiply | 33 | 91 | 9.1 µs | 36.4 µs | 91 µs |
| | Hardware multiply | 6 | 6 | 600 ns | 2.4 µs | 6 µs |
| 16 x 16 unsigned | Without hardware multiply | 21 | 242 | 24.2 µs | 96.8 µs | 242 µs |
| | Hardware multiply | 24 | 24 | 2.4 µs | 9.6 µs | 24 µs |
| 16 x 16 signed | Without hardware multiply | 52 | 254 | 25.4 µs | 102.6 µs | 254 µs |
| | Hardware multiply | 36 | 36 | 3.6 µs | 14.4 µs | 36 µs |

© 2006 Microchip Technology Inc.

chanhvinh.com



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

**EQUATION 7-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM**

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

**EXAMPLE 7-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE**

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

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

**EQUATION 7-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM**

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

**EXAMPLE 7-4: 16 x 16 SIGNED MULTIPLY ROUTINE**

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



© 2006 Microchip Technology Inc.

chanhvinh.com



## 8.0 INTERRUPTS

The PIC18FXX8 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® IDE, be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

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

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

The interrupt priority feature is enabled by setting the IPEN bit (RCON register). When interrupt priority is enabled, there are two bits that enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts. 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.

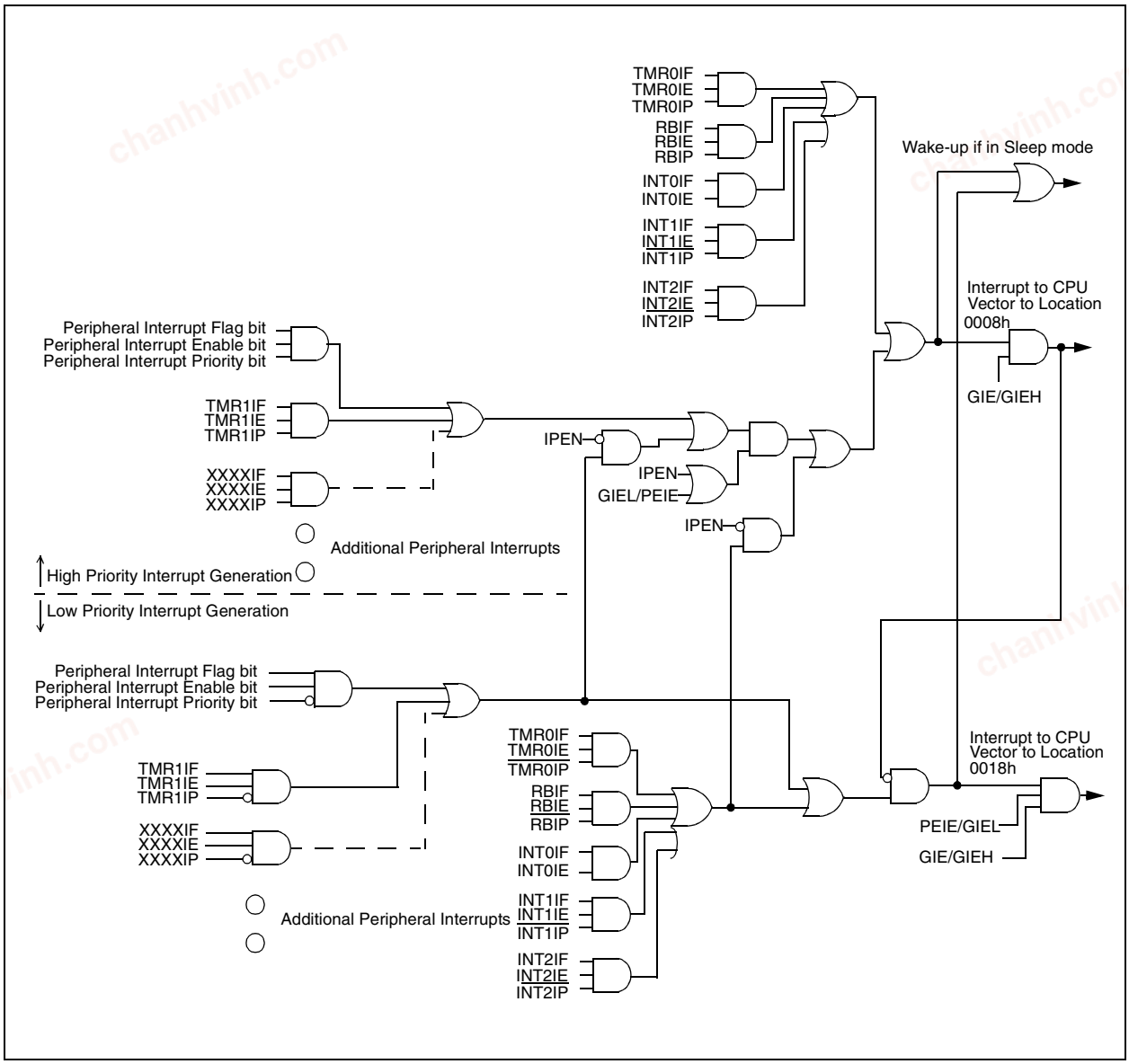
| **Note:** | Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior. |
|---|---|

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 8-1: INTERRUPT LOGIC**





© 2006 Microchip Technology Inc.

chanhvinh.com



## 8.1 INTCON Registers

The INTCON registers are readable and writable registers which contain various enable, priority and flag bits. Because of the number of interrupts to be controlled, PIC18FXX8 devices have three INTCON registers. They are detailed in Register 8-1 through Register 8-3.

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

**REGISTER 8-1: INTCON: INTERRUPT CONTROL 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 (RCON<7>) = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts

When IPEN (RCON<7>) = 1:
1 = Enables all high priority interrupts
0 = Disables all priority interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit

When IPEN (RCON<7>) = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

When IPEN (RCON<7>) = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low 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

**Note:** A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be 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 |

chanhvinh.com



**REGISTER 8-2: INTCON2: INTERRUPT CONTROL REGISTER 2**

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

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

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

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

bit 4-3 **Unimplemented:** Read as '0'

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

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

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

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

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


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 8-3: INTCON3: INTERRUPT CONTROL REGISTER 3**

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

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

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

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

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

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

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

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

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

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

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

chanhvinh.com

chanhvinh.com



## 8.2 PIR Registers

The Peripheral Interrupt Request (PIR) registers contain the individual flag bits for the peripheral interrupts (Register 8-4 through Register 8-6). 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 Interrupt 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.

**REGISTER 8-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1**

| R/W-0 | R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF |
| bit 7 | | | | | | | bit 0 |

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit(1)
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
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

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

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

**Note 1:** This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 8-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2**

| U-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | CMIF(1) | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF(1) |
| bit 7 | | | | | | | bit 0 |

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

bit 6 **CMIF:** Comparator Interrupt Flag bit(1)
1 = Comparator input has changed
0 = Comparator input has not changed

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

bit 4 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = Write operation is complete (must be cleared in software)
0 = Write operation is not complete

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 **ECCP1IF:** ECCP1 Interrupt Flag bit(1)
Capture mode:
1 = A TMR1 (TMR3) register capture occurred (must be cleared in software)
0 = No TMR1 (TMR3) register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

**Note 1:** This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads 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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 8-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3**

| 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:** 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



## 8.3 PIE Registers

The Peripheral Interrupt Enable (PIE) registers contain the individual enable bits for the peripheral interrupts (Register 8-7 through Register 8-9). 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.

**REGISTER 8-7: PIE1: PERIPHERAL INTERRUPT ENABLE 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 |
|---|---|---|---|---|---|---|---|
| PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |
| bit 7 | | | | | | | bit 0 |

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
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

**Note 1:** This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads 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 |

chanhvinh.com



**REGISTER 8-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| U-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | CMIE(1) | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE(1) |
| bit 7 | | | | | | | bit 0 |

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

bit 6 **CMIE:** Comparator Interrupt Enable bit(1)
1 = Enables the comparator interrupt
0 = Disables the comparator interrupt

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

bit 4 **EEIE**: EEPROM Write Interrupt Enable bit
1 = Enabled
0 = Disabled

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 **ECCP1IE**: ECCP1 Interrupt Enable bit(1)
1 = Enables the ECCP1 interrupt
0 = Disables the ECCP1 interrupt

**Note 1:** This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 8-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE | TXB0IE | RXB1IE | RXB0IE |
| bit 7 | | | | | | | bit 0 |

bit 7 **IRXIE:** Invalid CAN Message Received Interrupt Enable bit
1 = Enables the invalid CAN message received interrupt
0 = Disables the invalid CAN message received interrupt

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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



## 8.4 IPR Registers

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

**REGISTER 8-10: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 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 |
|---|---|---|---|---|---|---|---|
| PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP |
| bit 7 | | | | | | | bit 0 |

bit 7 **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit(1)
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

**Note 1:** This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 8-11: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2**

| U-0 | R/W-1 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| — | CMIP(1) | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP(1) |
| bit 7 | | | | | | | bit 0 |

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

bit 6 **CMIP:** Comparator Interrupt Priority bit(1)
1 = High priority
0 = Low priority

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

bit 4 **EEIP**: EEPROM Write Interrupt Priority bit
1 = High priority
0 = Low priority

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 **ECCP1IP**: ECCP1 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

**Note 1:** This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads 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 |

chanhvinh.com



**REGISTER 8-12: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3**

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

bit 7 **IRXIP:** Invalid Message Received Interrupt Priority bit
1 = High priority
0 = Low priority

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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



## 8.5 RCON Register

The Reset Control (RCON) register contains the IPEN bit which is used to enable prioritized interrupts. The functions of the other bits in this register are discussed in more detail in **Section 4.14 "RCON Register"**.

**REGISTER 8-13: RCON: RESET CONTROL REGISTER**

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

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6-5 **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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



## 8.6 INT Interrupts

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

Interrupt priority for INT1 and INT2 is determined by the value contained in the interrupt priority bits INT1IP (INTCON3<6>) and INT2IP (INTCON3<7>). There is no priority bit associated with INT0; it is always a high priority interrupt source.

## 8.7 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 11.0 "Timer0 Module"** for further details.

## 8.8 PORTB Interrupt-on-Change

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

## 8.9 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 "Fast Register Stack"**), the user may need to save the WREG, Status and BSR registers in software. Depending on the user's application, other registers may also need to be saved. Example 8-1 saves and restores the WREG, Status and BSR registers during an Interrupt Service Routine.

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

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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 9.0 I/O PORTS

Depending on the device selected, there are up to five general purpose I/O ports available on PIC18FXX8 devices. 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:

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

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

### 9.1 PORTA, TRISA and LATA Registers

PORTA is a 7-bit wide, bidirectional 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 high-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 read and write 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 Register 1). On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

| | |
|---|---|
| **Note:** | On a Power-on Reset, RA5 and RA3:RA0 are configured as analog inputs and read as '0'. RA6 and RA4 are configured as digital inputs. |

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.

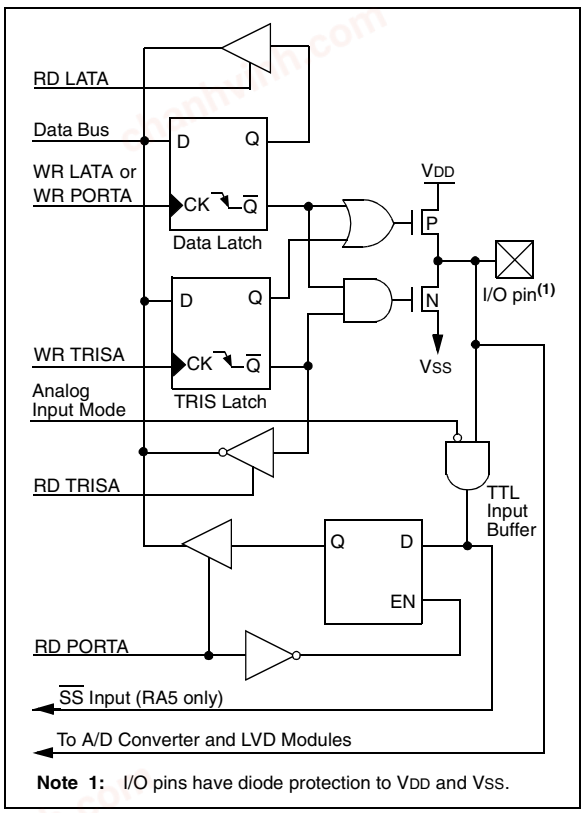
**EXAMPLE 9-1: INITIALIZING PORTA**

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

chanhvinh.com

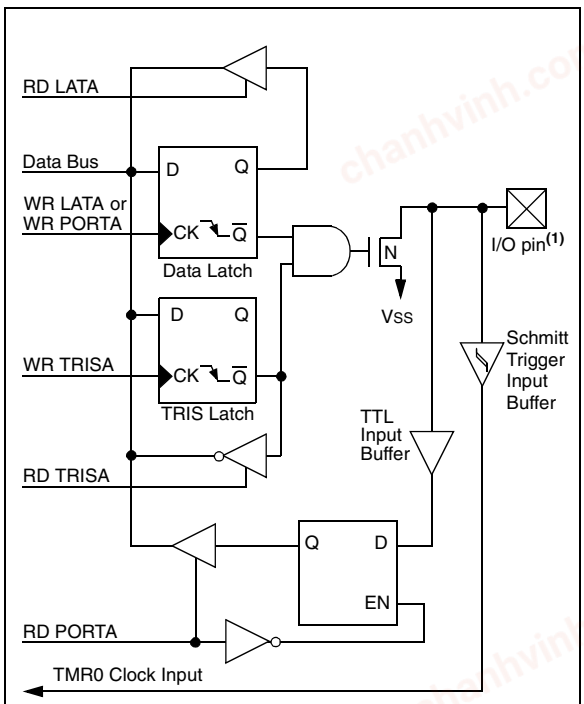


**FIGURE 9-1: RA3:RA0 AND RA5 PINS BLOCK DIAGRAM**



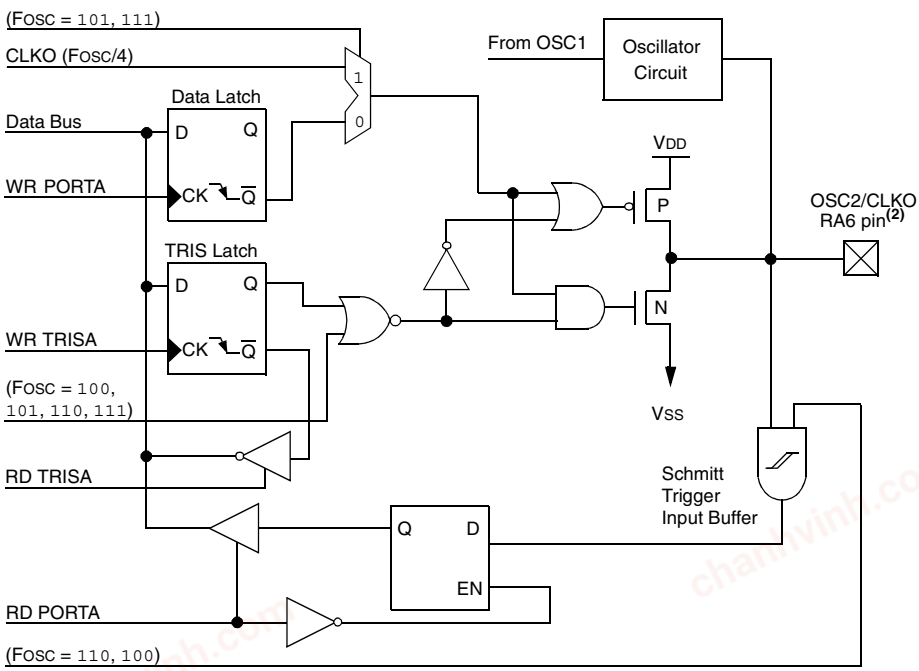

**Note 1:** I/O pins have diode protection to VDD and VSS.

**FIGURE 9-2: RA4/T0CKI PIN BLOCK DIAGRAM**




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

**FIGURE 9-3: OSC2/CLKO/RA6 PIN BLOCK DIAGRAM**




**Note 1:** CLKO is 1/4 of FOSC.
**2:** I/O pin has diode protection to VDD and VSS.


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 9-1: PORTA FUNCTIONS**

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RA0/AN0/CVREF | bit 0 | TTL | Input/output, analog input or analog comparator voltage reference output. |
| RA1/AN1 | bit 1 | TTL | Input/output or analog input. |
| RA2/AN2/VREF- | bit 2 | TTL | Input/output, analog input or VREF-. |
| RA3/AN3/VREF+ | bit 3 | TTL | Input/output, analog input or VREF+. |
| RA4/T0CKI | bit 4 | ST/OD | Input/output, external clock input for Timer0, output is open-drain type. |
| RA5/AN4/$\overline{SS}$/LVDIN | bit 5 | TTL | Input/output, analog input, slave select input for synchronous serial port or Low-Voltage Detect input. |
| OSC2/CLKO/RA6 | bit 6 | TTL | Oscillator clock output or input/output. |

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

**TABLE 9-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTA | — | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | -00x 0000 | -uuu uuuu |
| LATA | — | Latch A Data Output Register | | | | | | | -xxx xxxx | -uuu uuuu |
| TRISA | — | PORTA Data Direction Register | | | | | | | -111 1111 | -111 1111 |
| ADCON1 | ADFM | ADCS2 | — | — | 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.

chanhvinh.com



## 9.2 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional 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 high-impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

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

**EXAMPLE 9-2: INITIALIZING PORTB**

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

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

Four of the PORTB 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 ORed together to generate the RB Port Change Interrupt with Flag bit RBIF (INTCON register).

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

a) Any read or write of PORTB (except with the MOVFF instruction). This will end the mismatch condition.

b) Clear flag bit RBIF.

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

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

**Note 1:** While in Low-Voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O pin and should not be held low during normal operation to protect against inadvertent ICSP mode entry.

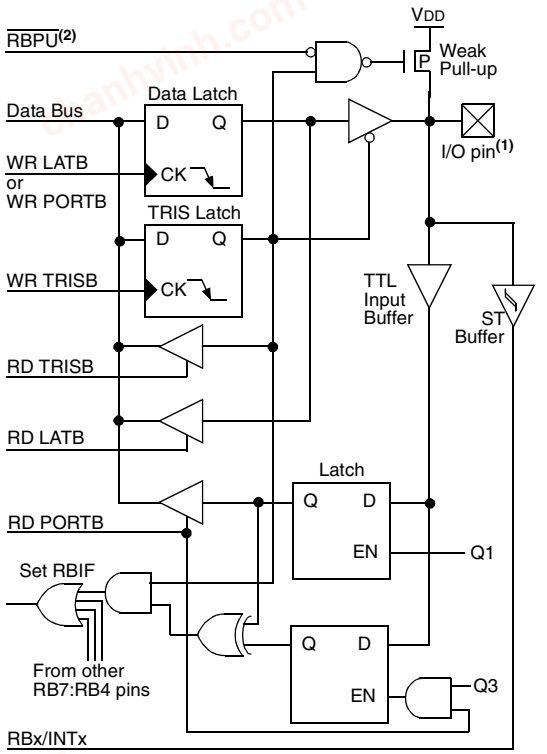
**2:** When using Low-Voltage ICSP Programming (LVP), the pull-up on RB5 becomes disabled. If TRISB bit 5 is cleared, thereby setting RB5 as an output, LATB bit 5 must also be cleared for proper operation.


© 2006 Microchip Technology Inc.

chanhvinh.com

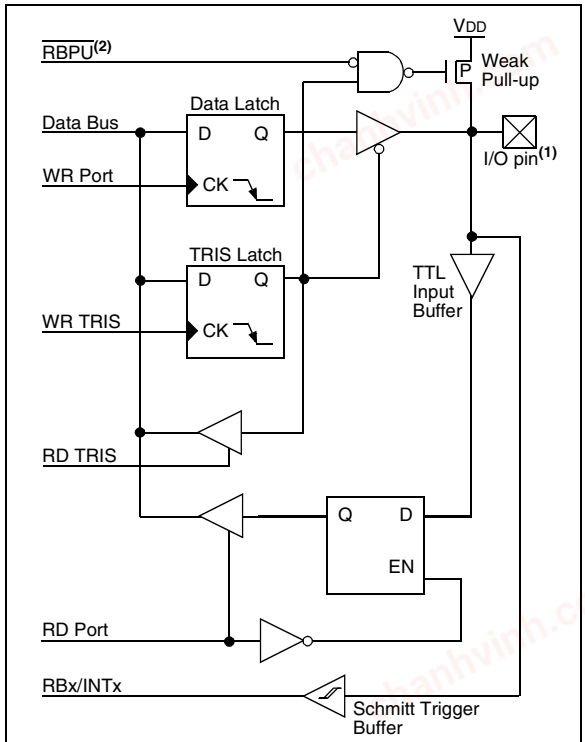


**FIGURE 9-4: RB7:RB4 PINS BLOCK DIAGRAM**




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

**FIGURE 9-5: RB1:RB0 PINS BLOCK DIAGRAM**

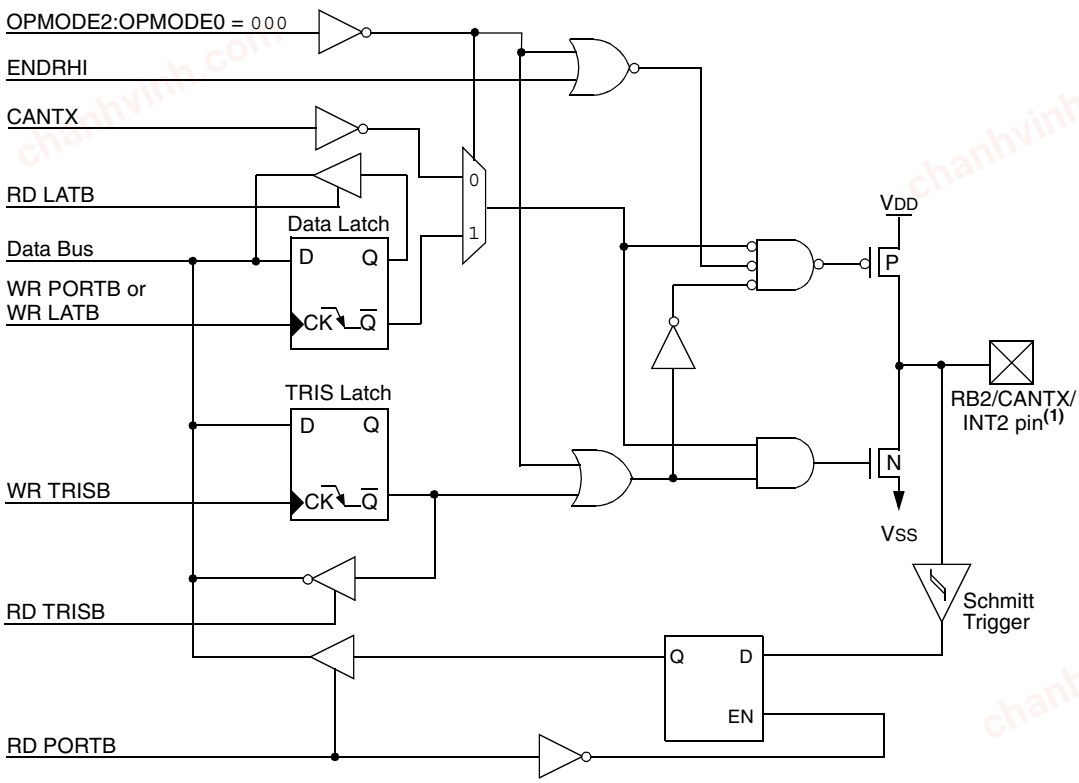



**Note 1:** I/O pins have diode protection to VDD and VSS.
**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2 register).

chanhvinh.com

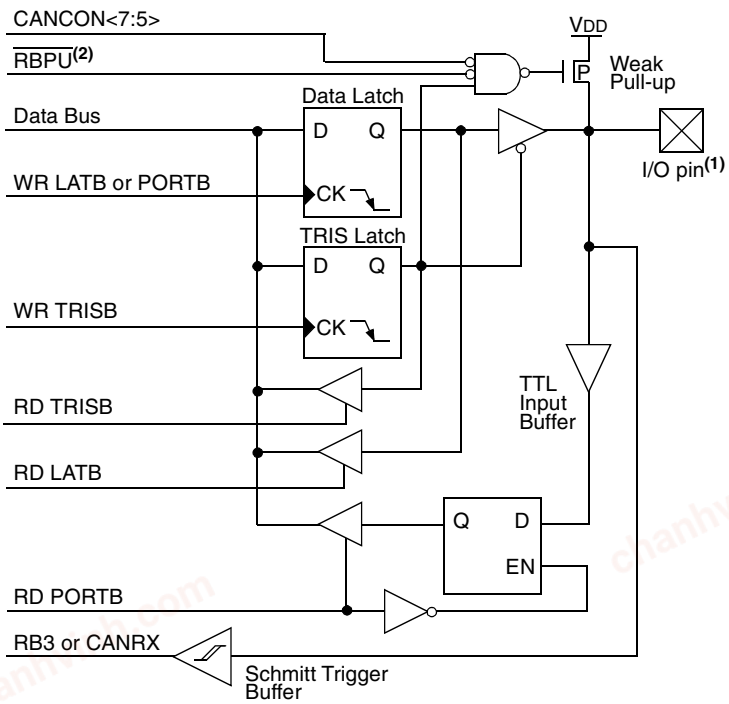


**FIGURE 9-6: RB2/CANTX/INT2 PIN BLOCK DIAGRAM**




**Note 1:** I/O pin has diode protection to VDD and VSS.

**FIGURE 9-7: RB3/CANRX PIN BLOCK DIAGRAM**




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

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).



© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 9-3: PORTB FUNCTIONS**

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RB0/INT0 | bit 0 | TTL/ST(1) | Input/output pin or external interrupt 0 input.<br>Internal software programmable weak pull-up. |
| RB1/INT1 | bit 1 | TTL/ST(1) | Input/output pin or external interrupt 1 input.<br>Internal software programmable weak pull-up. |
| RB2/CANTX/ INT2 | bit 2 | TTL/ST(1) | Input/output pin, CAN bus transmit pin or external interrupt 2 input.<br>Internal software programmable weak pull-up. |
| RB3/CANRX | bit 3 | TTL | Input/output pin or CAN bus receive pin.<br>Internal software programmable weak pull-up. |
| RB4 | bit 4 | TTL | Input/output pin (with interrupt-on-change).<br>Internal software programmable weak pull-up. |
| RB5/PGM | bit 5 | TTL | Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Low-voltage serial programming enable. |
| RB6/PGC | bit 6 | TTL/ST(2) | Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock. |
| RB7/PGD | bit 7 | 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 9-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxx xxxx | uuuu uuuu |
| LATB | LATB Data Output Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | — | — | TMR0IP | — | RBIP | 111- -1-1 | 111- -1-1 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | 11-1 0-00 |

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

© 2006 Microchip Technology Inc.

chanhvinh.com



## 9.3 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bidirectional 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 high-impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

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

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

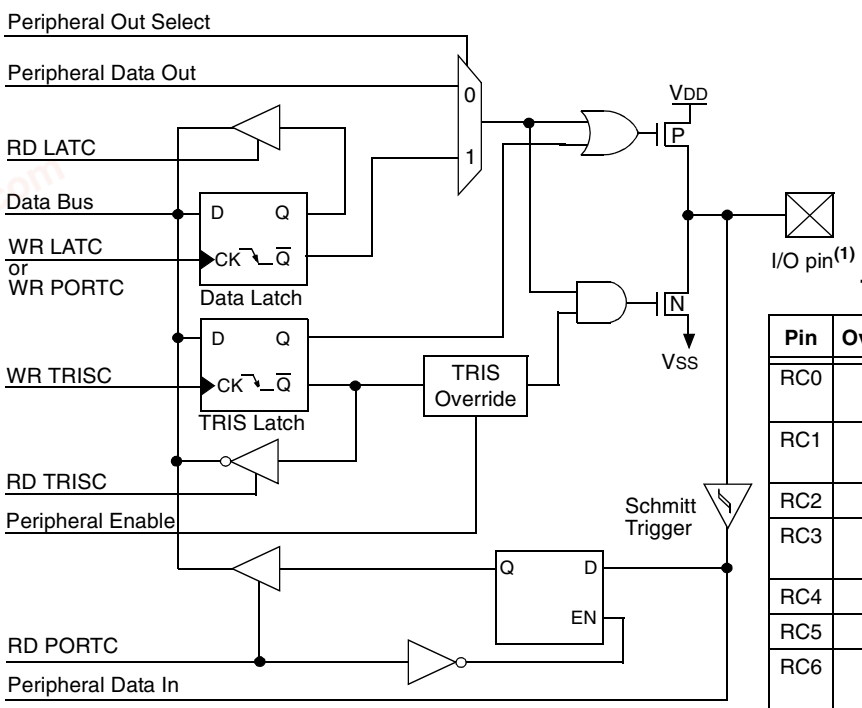
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the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

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

**EXAMPLE 9-3: INITIALIZING PORTC**

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

**FIGURE 9-8: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE)**




**TRIS OVERRIDE**

| Pin | Override | Peripheral |
|---|---|---|
| RC0 | Yes | Timer1 Oscillator for Timer1/Timer3 |
| RC1 | Yes | Timer1 Oscillator for Timer1/Timer3 |
| RC2 | No | — |
| RC3 | Yes | SPI™/I²C™ Master Clock |
| RC4 | Yes | I²C Data Out |
| RC5 | Yes | SPI Data Out |
| RC6 | Yes | USART Async Xmit, Sync Clock |
| RC7 | Yes | USART Sync Data Out |

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


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 9-5: PORTC FUNCTIONS**

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RC0/T1OSO/T1CKI | bit 0 | ST | Input/output port pin, Timer1 oscillator output or Timer1/Timer3 clock input. |
| RC1/T1OSI | bit 1 | ST | Input/output port pin or Timer1 oscillator input. |
| RC2/CCP1 | bit 2 | ST | Input/output port pin or Capture 1 input/Compare 1 output/ PWM1 output. |
| RC3/SCK/SCL | bit 3 | ST | Input/output port pin or synchronous serial clock for SPI™/I$^2$C™. |
| RC4/SDI/SDA | bit 4 | ST | Input/output port pin or SPI data in (SPI mode) or data I/O (I$^2$C mode). |
| RC5/SDO | bit 5 | ST | Input/output port pin or synchronous serial port data output. |
| RC6/TX/CK | bit 6 | ST | Input/output port pin, addressable USART asynchronous transmit or addressable USART synchronous clock. |
| RC7/RX/DT | bit 7 | ST | Input/output port pin, addressable USART asynchronous receive or addressable USART synchronous data. |

**Legend:** ST = Schmitt Trigger input

**TABLE 9-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | xxxx xxxx | uuuu uuuu |
| LATC | LATC Data Output Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TRISC | PORTC Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |

**Legend:** x = unknown, u = unchanged

© 2006 Microchip Technology Inc.

chanhvinh.com



## 9.4 PORTD, TRISD and LATD Registers

**Note:** This port is only available on the PIC18F448 and PIC18F458.

PORTD is an 8-bit wide, bidirectional port. The corresponding Data Direction register for the port is TRISD. Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., put the corresponding output driver in a high-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 read and write the latched output value for PORTD.

PORTD uses Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

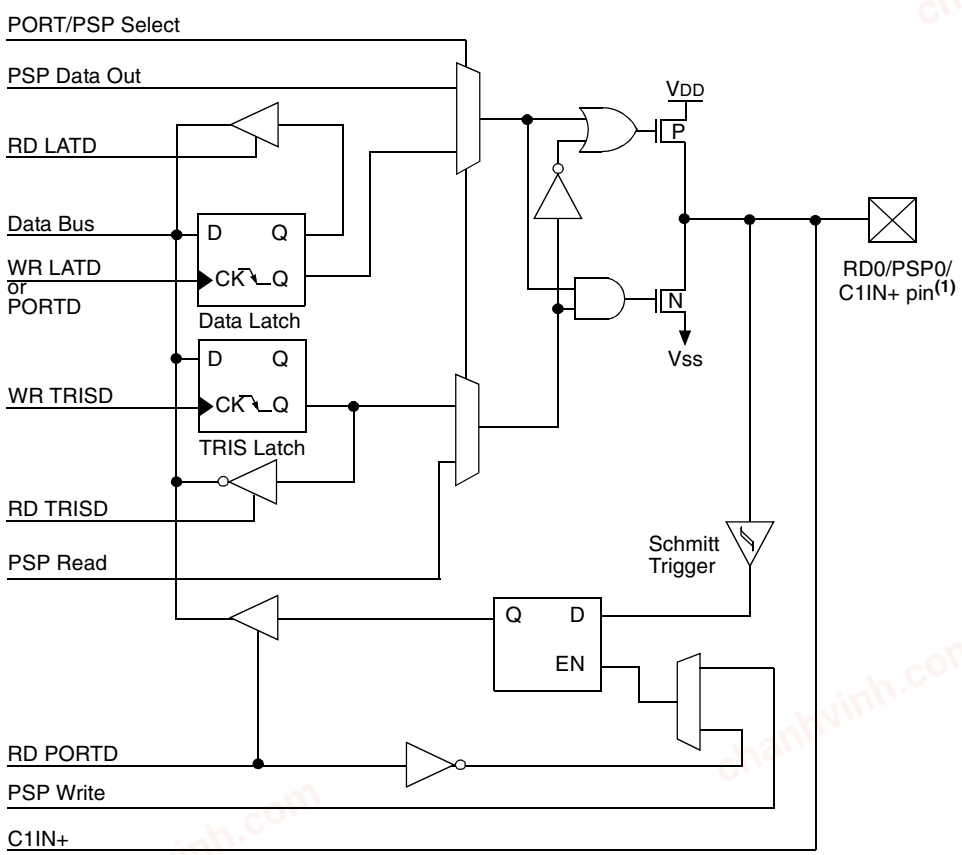
PORTD can be configured as an 8-bit wide, microprocessor port (Parallel Slave Port or PSP) by setting the control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. See **Section 10.0 "Parallel Slave Port"** for additional information.

PORTD is also multiplexed with the analog comparator module and the ECCP module.

**EXAMPLE 9-4: INITIALIZING PORTD**

```
CLRF    PORTD       ; Initialize PORTD by
                    ; clearing output
                    ; data latches
CLRF    LATD        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   07h         ; comparator off
MOVWF   CMCON
MOVLW   0CFh        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISD       ; Set RD3:RD0 as inputs
                    ; RD5:RD4 as outputs
                    ; RD7:RD6 as inputs
```

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




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


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 9-7: PORTD FUNCTIONS**

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RD0/PSP0/C1IN+ | bit 0 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 0 or C1IN+ comparator input. |
| RD1/PSP1/C1IN- | bit 1 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 1 or C1IN- comparator input. |
| RD2/PSP2/C2IN+ | bit 2 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 2 or C2IN+ comparator input. |
| RD3/PSP3/C2IN- | bit 3 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 3 or C2IN- comparator input. |
| RD4/PSP4/ECCP1/P1A | bit 4 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 4 or ECCP1/P1A pin. |
| RD5/PSP5/P1B | bit 5 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 5 or P1B pin. |
| RD6/PSP6/P1C | bit 6 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 6 or P1C pin. |
| RD7/PSP7/P1D | bit 7 | ST/TTL(1) | Input/output port pin, Parallel Slave Port bit 7 or P1D pin. |

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

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

**TABLE 9-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx xxxx | uuuu uuuu |
| LATD | LATD Data Output Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TRISD | PORTD Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| TRISE | IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 | 0000 -111 | 0000 -111 |

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

chanhvinh.com



## 9.5 PORTE, TRISE and LATE Registers

**Note:** This port is only available on the PIC18F448 and PIC18F458.

PORTE is a 3-bit wide, bidirectional port. PORTE has three pins (RE0/AN5/RD, RE1/AN6/WR/C1OUT and RE2/AN7/CS/C2OUT) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

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

The corresponding Data Direction register for the port is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a high-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).

The TRISE register also controls the operation of the Parallel Slave Port through the control bits in the upper half of the register. These are shown in Register 9-1.

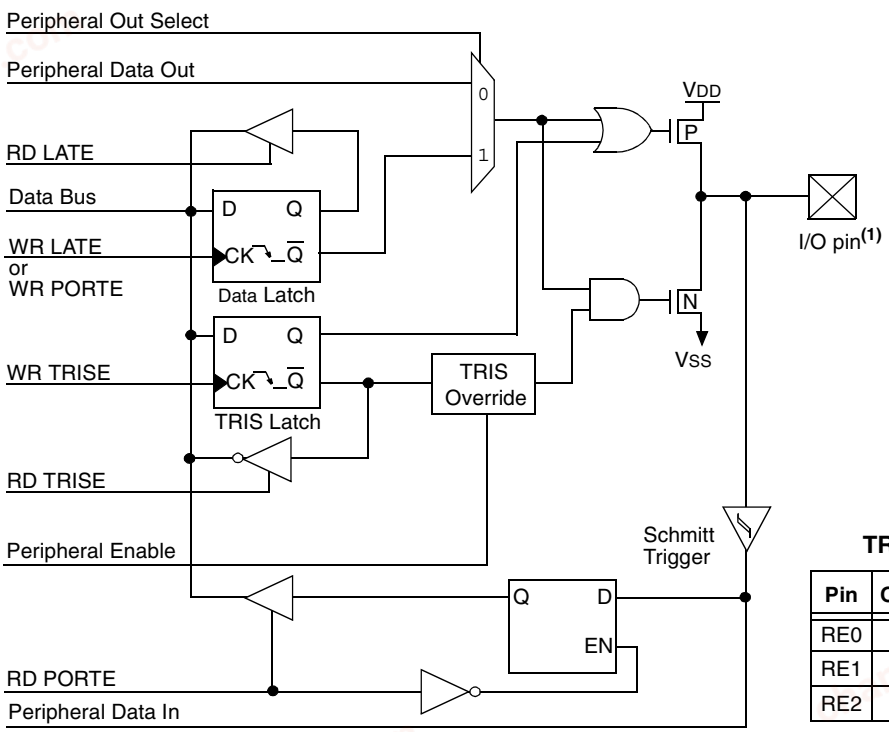
When the Parallel Slave Port is active, the PORTE pins function as its control inputs. For additional details, refer to **Section 10.0 "Parallel Slave Port"**.

PORTE pins are also multiplexed with inputs for the A/D converter and outputs for the analog comparators. When selected as an analog input, these pins will read as '0's. Direction bits TRISE<2:0> control the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

**EXAMPLE 9-5: INITIALIZING PORTE**

```
CLRF    PORTE       ; Initialize PORTE by
                    ; clearing output
                    ; data latches
CLRF    LATE        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   03h         ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISE       ; Set RE1:RE0 as inputs
                    ; RE2 as an output
                    ; (RE4=0 - PSPMODE Off)
```

**FIGURE 9-10: PORTE BLOCK DIAGRAM**




**TRIS OVERRIDE**

| Pin | Override | Peripheral |
|---|---|---|
| RE0 | Yes | PSP |
| RE1 | Yes | PSP |
| RE2 | Yes | PSP |

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


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 9-1: TRISE REGISTER**

| R-0 | R-0 | R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 |
| 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 **Unimplemented:** Read as '0'

bit 2 **TRISE2**: RE2 Direction Control bit
1 = Input
0 = Output

bit 1 **TRISE1**: RE1 Direction Control bit
1 = Input
0 = Output

bit 0 **TRISE0**: RE0 Direction Control bit
1 = Input
0 = Output

| **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 |

chanhvinh.com



**TABLE 9-9: PORTE FUNCTIONS**

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RE0/AN5/$\overline{RD}$ | bit 0 | ST/TTL(1) | Input/output port pin, analog input or read control input in Parallel Slave Port mode. |
| RE1/AN6/$\overline{WR}$/C1OUT | bit 1 | ST/TTL(1) | Input/output port pin, analog input, write control input in Parallel Slave Port mode or Comparator 1 output. |
| RE2/AN7/$\overline{CS}$/C2OUT | bit 2 | ST/TTL(1) | Input/output port pin, analog input, chip select control input in Parallel Slave Port mode or Comparator 2 output. |

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

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

**TABLE 9-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| TRISE | IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 | 0000 -111 | 0000 -111 |
| PORTE | — | — | — | — | — | Read PORTE pin/ Write PORTE Data Latch | | | ---- -xxx | ---- -uuu |
| LATE | — | — | — | — | — | Read PORTE Data Latch/ Write PORTE Data Latch | | | ---- -xxx | ---- -uuu |
| ADCON1 | ADFM | ADCS2 | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 00-- 0000 | 00-- 0000 |

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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 10.0 PARALLEL SLAVE PORT

**Note:** The Parallel Slave Port is only available on PIC18F4X8 devices.

In addition to its function as a general I/O port, PORTD can also operate as an 8-bit wide Parallel Slave Port (PSP) or microprocessor port. PSP operation is controlled by the 4 upper bits of the TRISE register (Register 9-1). Setting control bit PSPMODE (TRISE<4>) enables PSP operation. In Slave mode, the port is asynchronously readable and writable by the external world.

The PSP 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 the control bit PSPMODE enables the PORTE I/O pins to become control inputs for the microprocessor port. When set, port pin RE0 is the $\overline{RD}$ input, RE1 is the $\overline{WR}$ input and RE2 is 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 timing for the control signals in Write and Read modes is shown in Figure 10-2 and Figure 10-3, respectively.

**FIGURE 10-1: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)**

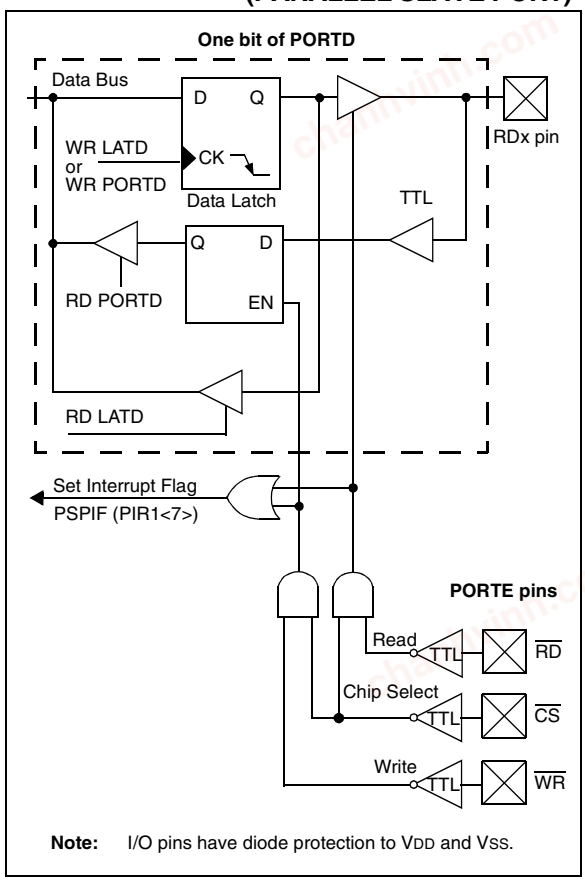
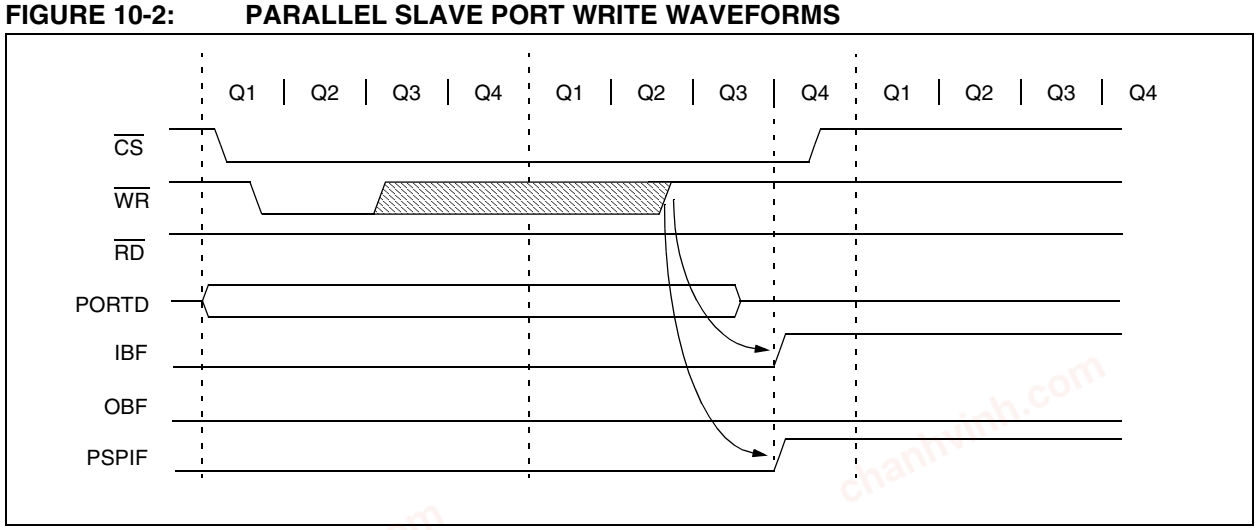



**FIGURE 10-2: PARALLEL SLAVE PORT WRITE WAVEFORMS**



chanhvinh.com



**FIGURE 10-3: PARALLEL SLAVE PORT READ WAVEFORMS**

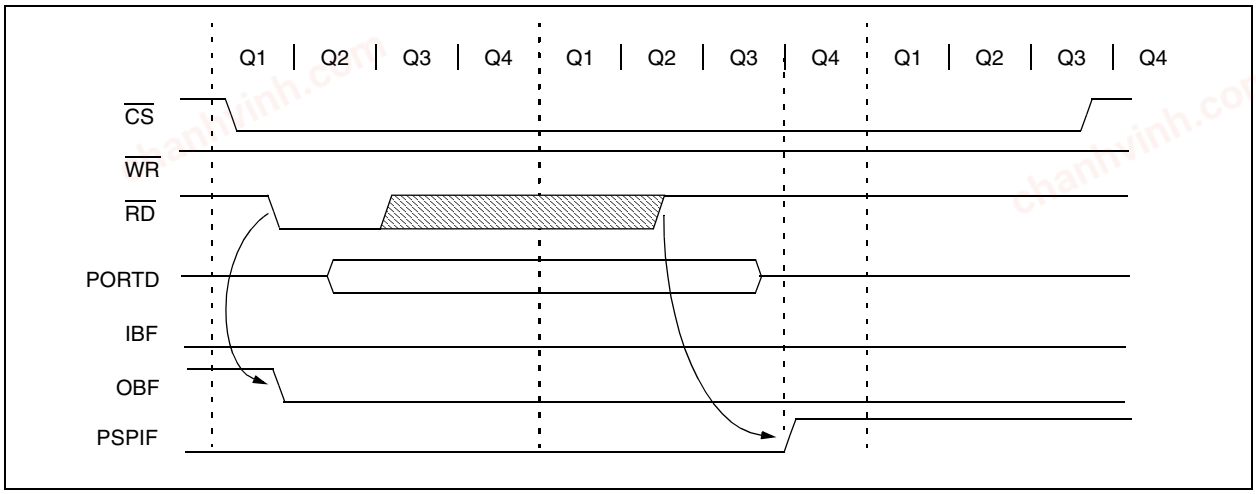



**TABLE 10-1: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT**

| 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 | — | — | — | — | — | RE2 | RE1 | RE0 | ---- -xxx | ---- -000 |
| LATE | LATE Data Output bits | | | | | | | | ---- -xxx | ---- -uuu |
| TRISE | IBF | OBF | IBOV | PSPMODE | — | PORTE Data Direction bits | | | 0000 -111 | 0000 -111 |
| 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 | 1111 1111 | 1111 1111 |

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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 11.0 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 11-1 shows the Timer0 Control register (T0CON).

Figure 11-1 shows a simplified block diagram of the Timer0 module in 8-bit mode and Figure 11-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

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

| **Note:** | Timer0 is enabled on POR. |
|---|---|

**REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER**

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

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

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

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

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

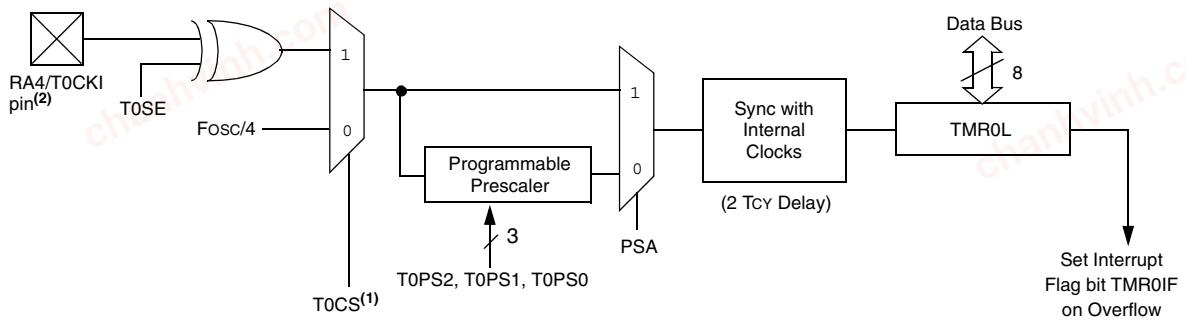
| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

© 2006 Microchip Technology Inc.

chanhvinh.com

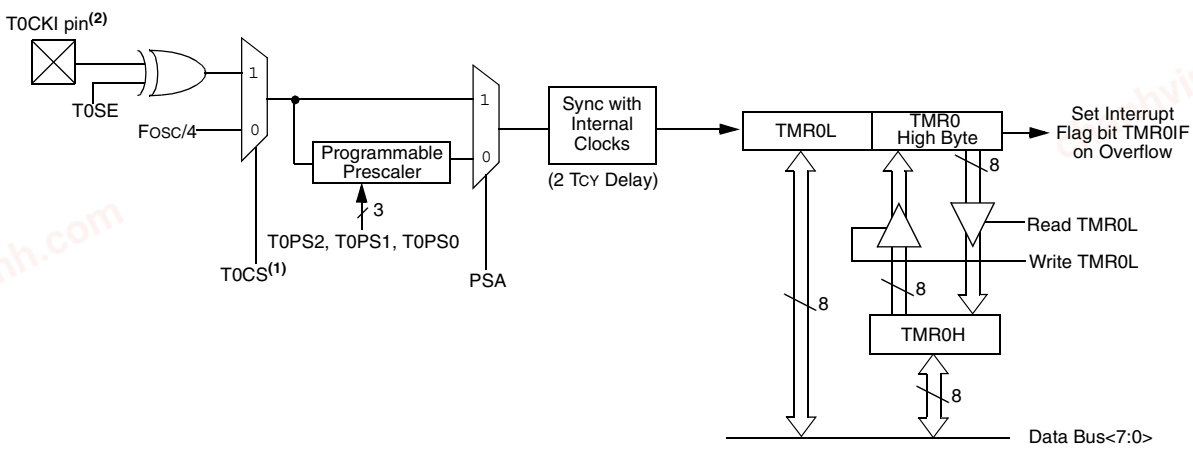


**FIGURE 11-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE**




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

**FIGURE 11-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE**




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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 11.1 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.

## 11.2 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. |
|---|---|

### 11.2.1 SWITCHING PRESCALER ASSIGNMENT

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

## 11.3 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.

## 11.4 16-Bit Mode Timer Reads and Writes

Timer0 can be set in 16-bit mode by clearing the T08BIT in T0CON. Registers TMR0H and TMR0L are used to access the 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 11-1). The high byte of the Timer0 timer/counter is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid, due to a rollover between successive reads of the high and low byte.

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

**TABLE 11-1: 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 Low Byte Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR0H | Timer0 Module 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) | | | | | | | -111 1111 | -111 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 is enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, it is disabled and reads as ‘0’.

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 12.0 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 12-1 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 12-1 is a simplified block diagram of the Timer1 module.

| **Note:** | Timer1 is disabled on POR. |
|---|---|

**REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER**

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

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

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

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

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

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

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (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 |

chanhvinh.com



## 12.1 Timer1 Operation

Timer1 can operate in one of these modes:

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

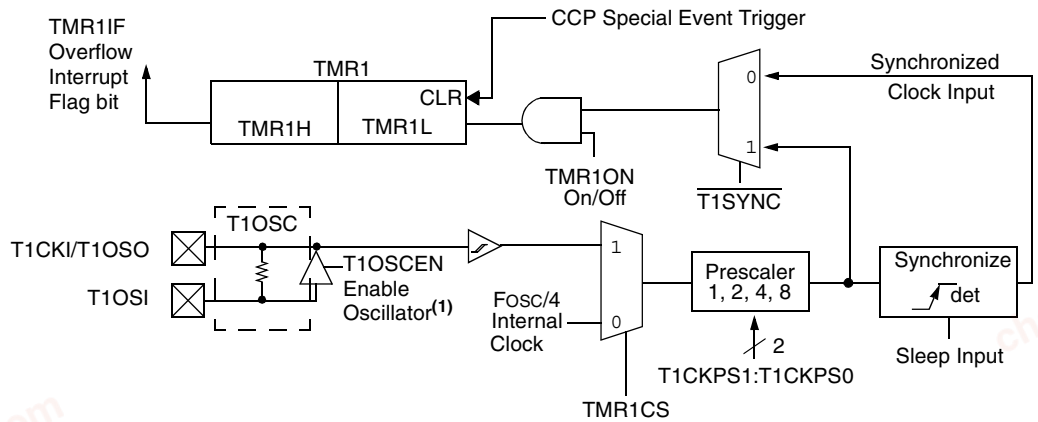
The operating mode is determined by the clock select bit, TMR1CS (T1CON register).

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.

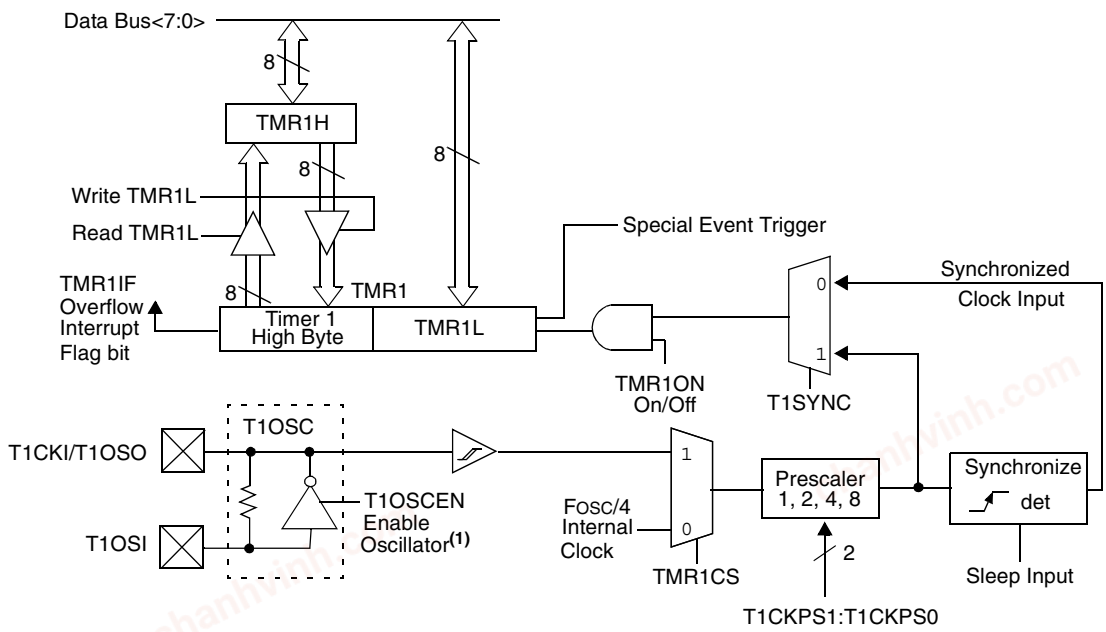
Timer1 also has an internal "Reset input". This Reset can be generated by the CCP module (**Section 15.1 "CCP1 Module"**).

**FIGURE 12-1: TIMER1 BLOCK DIAGRAM**




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

**FIGURE 12-2: TIMER1 BLOCK DIAGRAM: 16-BIT READ/WRITE MODE**




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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 12.2 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 50 kHz. It will continue to run during Sleep. It is primarily intended for a 32 kHz crystal. Table 12-1 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 12-1: CAPACITOR SELECTION FOR THE ALTERNATE OSCILLATOR**

| Osc Type | Freq | C1 | C2 |
|---|---|---|---|
| LP | 32 kHz | TBD(1) | TBD(1) |
| **Crystal to be Tested:** | | | |
| 32.768 kHz | Epson C-001R32.768K-A | | ±20 PPM |

**Note 1:** Microchip suggests 33 pF as a starting point in validating the oscillator circuit.

**2:** Higher capacitance increases the stability of the oscillator, but also increases the start-up time.

**3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

**4:** Capacitor values are for design guidance only.

## 12.3 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).

## 12.4 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 register pair effectively becomes the period register for Timer1.

## 12.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). 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 register. 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.

© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 12-2: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0-00 0000 | u-uu uuuu |

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

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.



© 2006 Microchip Technology Inc.

chanhvinh.com



## 13.0 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 13-1 shows the Timer2 Control register. Timer2 can be shut-off by clearing control bit TMR2ON (T2CON register) to minimize power consumption. Figure 13-1 is a simplified block diagram of the Timer2 module. The prescaler and postscaler selection of Timer2 are controlled by this register.

### 13.1 Timer2 Operation

Timer2 can be used as the PWM time base for the PWM mode of the CCP module. The TMR2 register is readable and writable and is cleared on any device Reset. The input clock (Fosc/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON register). The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, PIR registers).

The prescaler and postscaler counters are cleared when any of the following occurs:

- A write to the TMR2 register
- A write to the T2CON register
- Any device Reset (Power-on Reset, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

| **Note:** | Timer2 is disabled on POR. |
|---|---|

**REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| bit 7 | | | | | | | bit 0 |

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

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

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

© 2006 Microchip Technology Inc.

chanhvinh.com



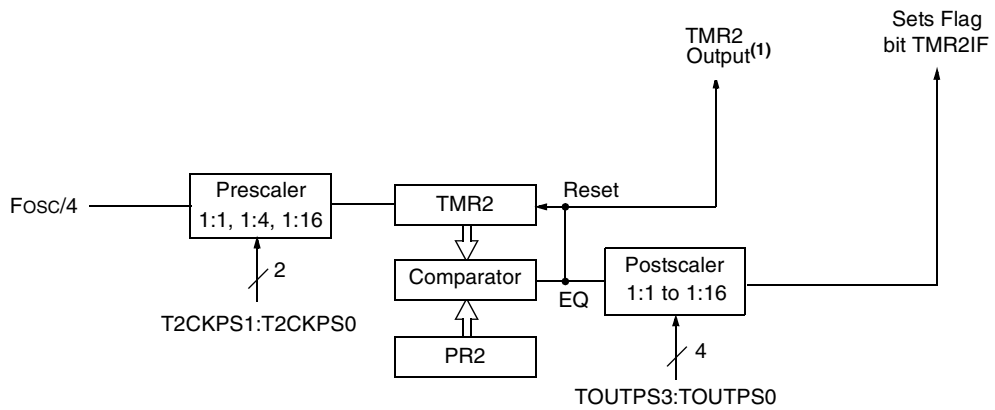
## 13.2 Timer2 Interrupt

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.

## 13.3 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 13-1: TIMER2 BLOCK DIAGRAM**




**Note 1:** TMR2 register output can be software selected by the SSP module as a baud clock.

**TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| TMR2 | Timer2 Module 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.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 14.0 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 CCP1/ECCP1 module trigger

Figure 14-1 is a simplified block diagram of the Timer3 module.

Register 14-1 shows the Timer3 Control register. This register controls the operating mode of the Timer3 module and sets the CCP1 and ECCP1 clock source.

Register 12-1 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 14-1: T3CON:TIMER3 CONTROL 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 | T3ECCP1 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON |
| bit 7 | | | | | | | bit 0 |

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
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 **T3ECCP1:T3CCP1:** Timer3 and Timer1 to CCP1/ECCP1 Enable bits
1x = Timer3 is the clock source for compare/capture CCP1 and ECCP1 modules
01 = Timer3 is the clock source for compare/capture of ECCP1,
Timer1 is the clock source for compare/capture of CCP1
00 = Timer1 is the clock source for compare/capture CCP1 and ECCP1 modules

bit 5-4 **T3CKPS1:T3CKPS0**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T1CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

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

chanhvinh.com



## 14.1 Timer3 Operation

Timer3 can operate in one of these modes:

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

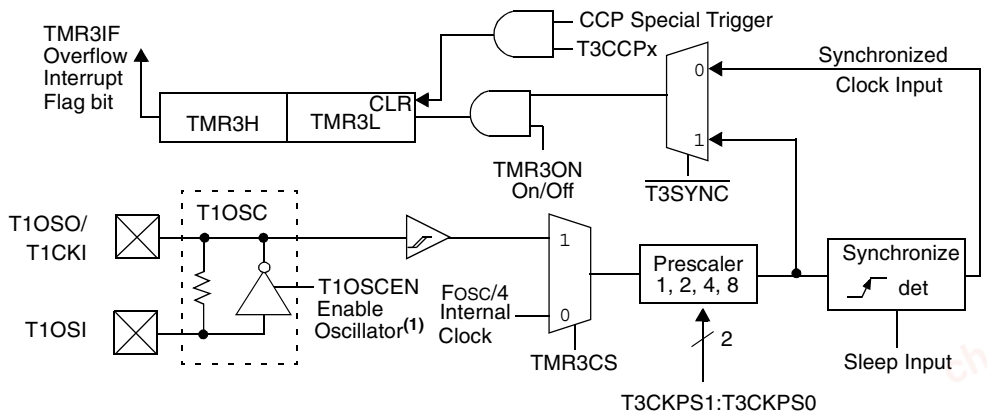
The operating mode is determined by the clock select bit, TMR3CS (T3CON register).

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.

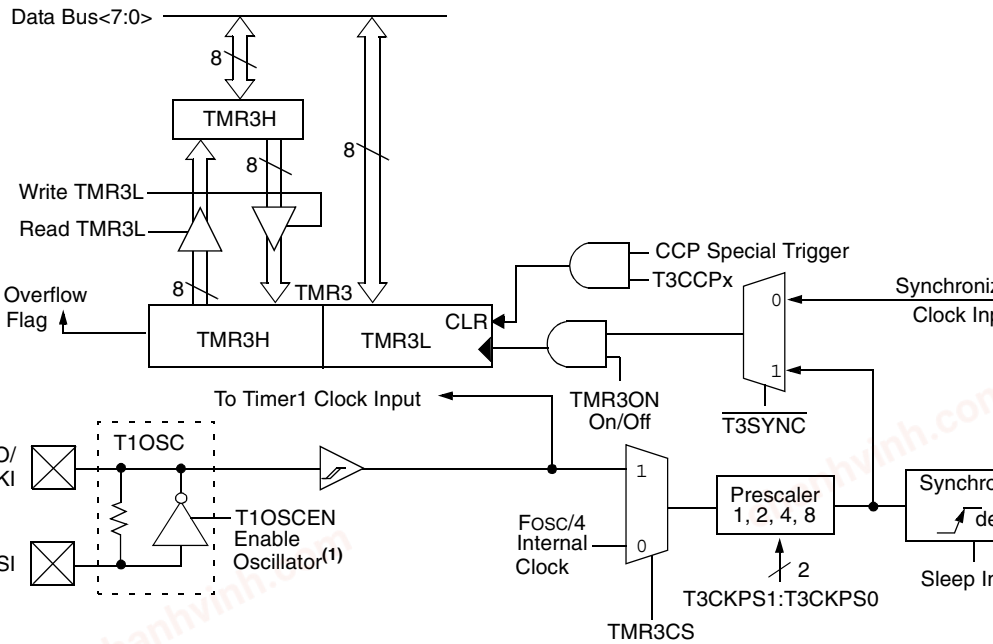
Timer3 also has an internal "Reset input". This Reset can be generated by the CCP module (**Section 15.1 "CCP1 Module"**).

**FIGURE 14-1: TIMER3 BLOCK DIAGRAM**




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

**FIGURE 14-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE**




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


© 2006 Microchip Technology Inc.

chanhvinh.com



## 14.2 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 50 kHz. Refer to **Section 12.0 "Timer1 Module"** for Timer1 oscillator details.

## 14.3 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to 0FFFFh 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).

## 14.4 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 register pair becomes the period register for Timer3. Refer to **Section 15.0 "Capture/Compare/PWM (CCP) Modules"** for CCP details.

**TABLE 14-1: 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 | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE | -0-0 0000 | -0-0 0000 |
| IPR2 | — | CMIP | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP | -1-1 1111 | -1-1 1111 |
| TMR3L | Holding Register for the Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0-00 0000 | u-uu uuuu |
| T3CON | RD16 | T3ECCP1 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0000 0000 | uuuu uuuu |

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

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 15.0 CAPTURE/COMPARE/PWM (CCP) MODULES

The 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.

The operation of the CCP module is identical to that of the ECCP module (discussed in detail in **Section 16.0 "Enhanced Capture/Compare/PWM (ECCP) Module"**) with two exceptions. The CCP module has a Capture special event trigger that can be used as a message received time-stamp for the CAN module (refer to **Section 19.0 "CAN Module"** for CAN operation) which the ECCP module does not. The ECCP module, on the other hand, has Enhanced PWM functionality and auto-shutdown capability. Aside from these, the operation of the module described in this section is the same as the ECCP.

The control register for the CCP module is shown in Register 15-1. Table 15-2 (following page) details the interactions of the CCP and ECCP modules.

**REGISTER 15-1: CCP1CON: CCP1 CONTROL REGISTER**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
| bit 7 | | | | | | | bit 0 |

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

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle bit 1 and bit 0

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs (bit 1 and bit 0) 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, CCP pin is unaffected (CCPIF bit is set)
1011 = Compare mode, trigger special event (CCP1IF bit is set; CCP resets TMR1 or TMR3 and starts an A/D conversion if the A/D module is enabled)
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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



## 15.1 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.

Table 15-1 shows the timer resources of the CCP module modes.

**TABLE 15-1: CCP1 MODE – TIMER RESOURCE**

| CCP1 Mode | Timer Resource |
|---|---|
| Capture | Timer1 or Timer3 |
| Compare | Timer1 or Timer3 |
| PWM | Timer2 |

## 15.2 Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 or TMR3 register 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.

### 15.2.1 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. |
|---|---|

### 15.2.2 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 15-2: INTERACTION OF CCP1 AND ECCP1 MODULES**

| CCP1 Mode | ECCP1 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. |


© 2006 Microchip Technology Inc.

chanhvinh.com



### 15.2.3 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.

### 15.2.4 CCP1 PRESCALER

There are four prescaler settings specified by bits CCP1M3:CCP1M0. Whenever the CCP1 module is turned off, or the CCP1 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 15-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

### 15.2.5 CAN MESSAGE TIME-STAMP

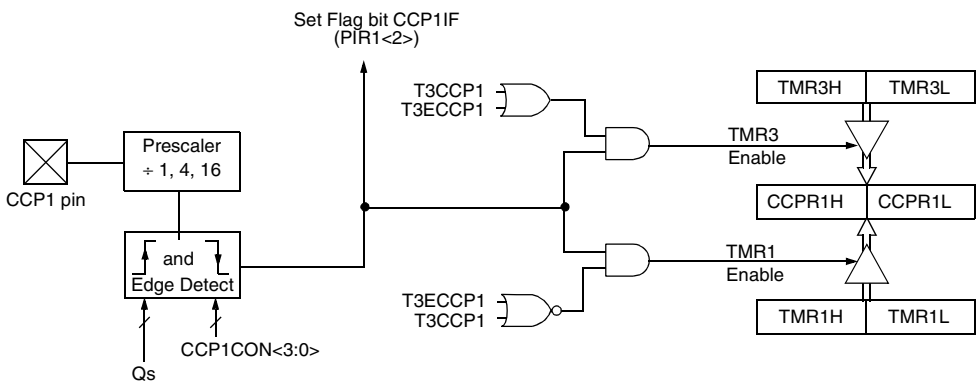
The CAN capture event occurs when a message is received in either of the receive buffers. The CAN module provides a rising edge to the CCP1 module to cause a capture event. This feature is provided to time-stamp the received CAN messages.

This feature is enabled by setting the CANCAP bit of the CAN I/O control register (CIOCON<4>). The message receive signal from the CAN module then takes the place of the events on RC2/CCP1.

**EXAMPLE 15-1: 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 15-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**




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

chanhvinh.com



## 15.3 Compare Mode

In Compare mode, the 16-bit CCPR1 and ECCPR1 register value is constantly compared against either the TMR1 register pair value or the TMR3 register pair value. When a match occurs, the CCP1 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. At the same time, interrupt flag bit CCP1IF is set.

### 15.3.1 CCP1 PIN CONFIGURATION

The user must configure the CCP1 pin as an output by clearing the appropriate TRISC bit.

| | |
|---|---|
| **Note:** | Clearing the CCP1CON register will force the CCP1 compare output latch to the default low level. This is not the data latch. |

### 15.3.2 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.

### 15.3.3 SOFTWARE INTERRUPT MODE

When generate software interrupt is chosen, the CCP1 pin is not affected. Only a CCP interrupt is generated (if enabled).

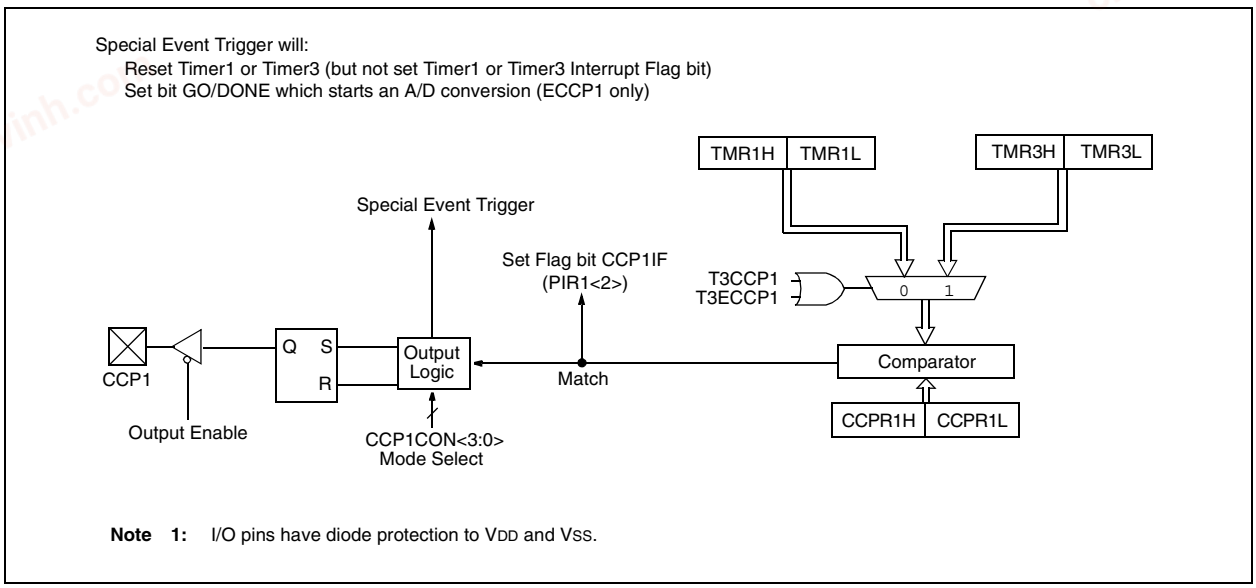
### 15.3.4 SPECIAL EVENT TRIGGER

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 either the TMR1 or TMR3 register pair. Additionally, the ECCP1 special event trigger will start an A/D conversion if the A/D module is enabled.

| | |
|---|---|
| **Note:** | The special event trigger from the ECCP1 module will not set the Timer1 or Timer3 interrupt flag bits. |

**FIGURE 15-2: COMPARE MODE OPERATION BLOCK DIAGRAM**




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


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 15-3: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| TRISD | PORTD Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0-00 0000 | u-uu uuuu |
| CCPR1L | Capture/Compare/PWM Register 1 (LSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| CCPR1H | Capture/Compare/PWM Register 1 (MSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| CCP1CON | — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | --00 0000 | --00 0000 |
| PIR2 | — | CMIF | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE | -0-0 0000 | -0-0 0000 |
| IPR2 | — | CMIP | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP | -1-1 1111 | -1-1 1111 |
| 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 | T3ECCP1 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0000 0000 | uuuu uuuu |

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

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

chanhvinh.com



## 15.4 PWM Mode

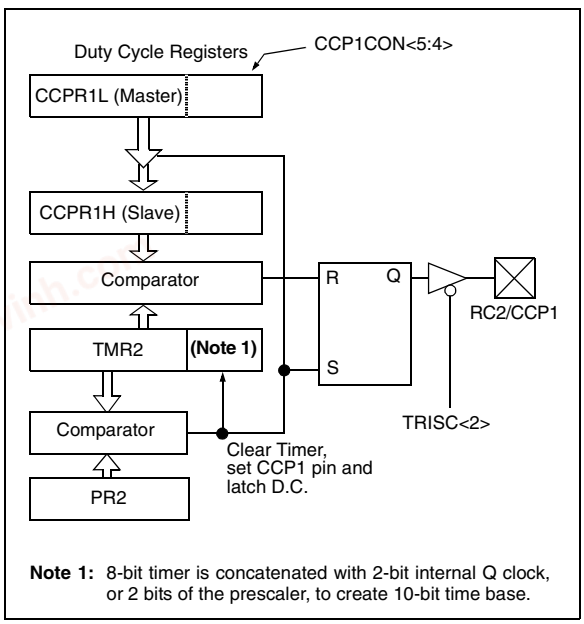
In Pulse-Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

| **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 15-3 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 15.4.3 "Setup for PWM Operation"**.

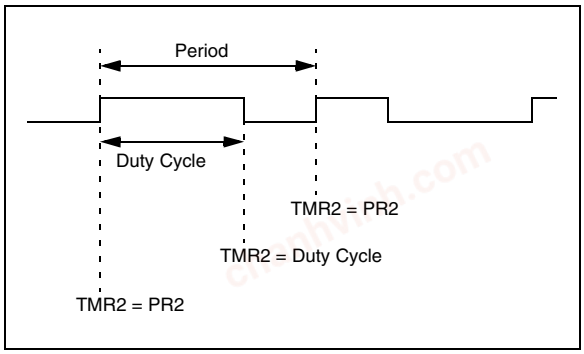
**FIGURE 15-3: SIMPLIFIED PWM BLOCK DIAGRAM**




**Note 1:** 8-bit timer is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create 10-bit time base.

A PWM output (Figure 15-4) 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 15-4: PWM OUTPUT**




### 15.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula.

**EQUATION 15-1:**

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

| **Note:** | The Timer2 postscaler (see **Section 13.0 "Timer2 Module"**) 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. |
|---|---|

### 15.4.2 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.

**EQUATION 15-2:**

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.


© 2006 Microchip Technology Inc.

chanhvinh.com



The maximum PWM resolution (bits) for a given PWM frequency is given by the following equation.

**EQUATION 15-3:**

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{FOSC}{FPWM}\right)}{\log(2)} \text{bits}$$

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

### 15.4.3 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 15-4: 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 | 0FFh | 0FFh | 0FFh | 3Fh | 1Fh | 17h |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 5.5 |

**TABLE 15-5: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| TRISD | PORTD Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| TMR2 | Timer2 Module Register | | | | | | | | 0000 0000 | 0000 0000 |
| PR2 | Timer2 Module 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 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

chanhvinh.com



**NOTES:**

chanhvinh.com


© 2006 Microchip Technology Inc.

chanhvinh.com



## 16.0 ENHANCED CAPTURE/ COMPARE/PWM (ECCP) MODULE

| **Note:** | The ECCP (Enhanced Capture/Compare/ PWM) module is only available on PIC18F448 and PIC18F458 devices. |
|---|---|

This module contains a 16-bit register which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

The operation of the ECCP module differs from the CCP (discussed in detail in **Section 15.0 "Capture/ Compare/PWM (CCP) Modules"**) with the addition of an Enhanced PWM module which allows for up to 4 output channels and user selectable polarity. These features are discussed in detail in **Section 16.5 "Enhanced PWM Mode"**. The module can also be programmed for automatic shutdown in response to various analog or digital events.

The control register for ECCP1 is shown in Register 16-1.

**REGISTER 16-1: ECCP1CON: ECCP1 CONTROL 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 |
|---|---|---|---|---|---|---|---|
| EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 |
| bit 7 | | | | | | | bit 0 |

bit 7-6 **EPWM1M<1:0>:** PWM Output Configuration bits

If ECCP1M<3:2> = 00, 01, 10:

xx = P1A assigned as Capture/Compare input; P1B, P1C, P1D assigned as port pins

If ECCP1M<3:2> = 11:

00 = Single output; P1A modulated; P1B, P1C, P1D assigned as port pins
01 = Full-bridge output forward; P1D modulated; P1A active; P1B, P1C inactive
10 = Half-bridge output; P1A, P1B modulated with deadband control; P1C, P1D assigned as port pins
11 = Full-bridge output reverse; P1B modulated; P1C active; P1A, P1D inactive

bit 5-4 **EDC1B<1:0>:** PWM Duty Cycle Least Significant bits

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in ECCPR1L.

bit 3-0 **ECCP1M<3:0>:** ECCP1 Mode Select bits

0000 = Capture/Compare/PWM off (resets ECCP module)
0001 = Unused (reserved)
0010 = Compare mode, toggle output on match (ECCP1IF bit is set)
0011 = Unused (reserved)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (ECCP1IF bit is set)
1001 = Compare mode, clear output on match (ECCP1IF bit is set)
1010 = Compare mode, ECCP1 pin is unaffected (ECCP1IF bit is set)
1011 = Compare mode, trigger special event (ECCP1IF bit is set; ECCP resets TMR1or TMR3 and starts an A/D conversion if the A/D module is enabled)
1100 = PWM mode; P1A, P1C active-high; P1B, P1D active-high
1101 = PWM mode; P1A, P1C active-high; P1B, P1D active-low
1110 = PWM mode; P1A, P1C active-low; P1B, P1D active-high
1111 = PWM mode; P1A, P1C active-low; P1B, P1D active-low

| **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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



## 16.1 ECCP1 Module

Enhanced Capture/Compare/PWM Register 1 (ECCPR1) is comprised of two 8-bit registers: ECCPR1L (low byte) and ECCPR1H (high byte). The ECCP1CON register controls the operation of ECCP1; the additional registers, ECCPAS and ECCP1DEL, control Enhanced PWM specific features. All registers are readable and writable.

Table 16-1 shows the timer resources for the ECCP module modes. Table 16-2 describes the interactions of the ECCP module with the standard CCP module.

In PWM mode, the ECCP module can have up to four available outputs, depending on which operating mode is selected. These outputs are multiplexed with PORTD and the Parallel Slave Port. Both the operating mode and the output pin assignments are configured by setting PWM output configuration bits, EPWM1M1:EPWM1M0 (ECCP1CON<7:6>). The specific pin assignments for the various output modes are shown in Table 16-3.

**TABLE 16-1: ECCP1 MODE – TIMER RESOURCE**

| ECCP1 Mode | Timer Resource |
|---|---|
| Capture | Timer1 or Timer3 |
| Compare | Timer1 or Timer3 |
| PWM | Timer2 |

**TABLE 16-2: INTERACTION OF CCP1 AND ECCP1 MODULES**

| ECCP1 Mode | CCP1 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 |

**TABLE 16-3: PIN ASSIGNMENTS FOR VARIOUS ECCP MODES**

| ECCP Mode(1) | ECCP1CON Configuration | RD4 | RD5 | RD6 | RD7 |
|---|---|---|---|---|---|
| Conventional CCP Compatible | 00xx11xx | ECCP1 | RD<5>, PSP<5> | RD<6>, PSP<6> | RD<7>, PSP<7> |
| Dual Output PWM(2) | 10xx11xx | P1A | P1B | RD<6>, PSP<6> | RD<7>, PSP<7> |
| Quad Output PWM(2) | x1xx11xx | P1A | P1B | P1C | P1D |

**Legend:** x = Don't care. Shaded cells indicate pin assignments not used by ECCP in a given mode.

**Note 1:** In all cases, the appropriate TRISD bits must be cleared to make the corresponding pin an output.

**2:** In these modes, the PSP I/O control for PORTD is overridden by P1B, P1C and P1D.



© 2006 Microchip Technology Inc.

chanhvinh.com



## 16.2 Capture Mode

The Capture mode of the ECCP module is virtually identical in operation to that of the standard CCP module as discussed in **Section 15.1 "CCP1 Module"**. The differences are in the registers and port pins involved:

- The 16-bit Capture register is ECCPR1 (ECCPR1H and ECCPR1L);
- The capture event is selected by control bits ECCP1M3:ECCP1M0 (ECCP1CON<3:0>);
- The interrupt bits are ECCP1IE (PIE2<0>) and ECCP1IF (PIR2<0>); and
- The capture input pin is RD4 and its corresponding direction control bit is TRISD<4>.

Other operational details, including timer selection, output pin configuration and software interrupts, are exactly the same as the standard CCP module.

### 16.2.1 CAN MESSAGE TIME-STAMP

The special capture event for the reception of CAN messages (**Section 15.2.5 "CAN Message Time-Stamp"**) is not available with the ECCP module.

## 16.3 Compare Mode

The Compare mode of the ECCP module is virtually identical in operation to that of the standard CCP module as discussed in **Section 15.2 "Capture Mode"**. The differences are in the registers and port pins as described in **Section 16.2 "Capture Mode"**. All other details are exactly the same.

### 16.3.1 SPECIAL EVENT TRIGGER

Except as noted below, the special event trigger output of ECCP1 functions identically to that of the standard CCP module. It may be used to start an A/D conversion if the A/D module is enabled.

| **Note:** | The special event trigger from the ECCP1 module will not set the Timer1 or Timer3 interrupt flag bits. |
|---|---|

**TABLE 16-4: REGISTERS ASSOCIATED WITH ENHANCED 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 |
| PIR2 | — | CMIF | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE | -0-0 0000 | -0-0 0000 |
| IPR2 | — | CMIP | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP | -1-1 1111 | -1-1 1111 |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| T1CON | RD16 | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0-00 0000 | u-uu uuuu |
| 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 | T3ECCP1 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0000 0000 | uuuu uuuu |
| TRISD | PORTD Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| ECCPR1L | Capture/Compare/PWM Register1 (LSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| ECCPR1H | Capture/Compare/PWM Register1 (MSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| ECCP1CON | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 0000 0000 | 0000 0000 |

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

© 2006 Microchip Technology Inc.

chanhvinh.com



## 16.4 Standard PWM Mode

When configured in Single Output mode, the ECCP module functions identically to the standard CCP module in PWM mode as described in **Section 15.4 "PWM Mode"**. The differences in registers and ports are as described in **Section 16.2 "Capture Mode"**. In addition, the two Least Significant bits of the 10-bit PWM duty cycle value are represented by ECCP1CON<5:4>.

| | |
|---|---|
| **Note:** | When setting up single output PWM operations, users are free to use either of the processes described in **Section 15.4.3 "Setup for PWM Operation"** or **Section 16.5.8 "Setup for PWM Operation"**. The latter is more generic, but will work for either single or multi-output PWM. |

## 16.5 Enhanced PWM Mode

The Enhanced PWM mode provides additional PWM output options for a broader range of control applications. The module is an upwardly compatible version of the standard CCP module and is modified to provide up to four outputs, designated P1A through P1D. Users are also able to select the polarity of the signal (either active-high or active-low). The module's output mode and polarity are configured by setting the EPWM1M1:EPWM1M0 and ECCP1M3:ECCP1M0 bits of the ECCP1CON register (ECCP1CON<7:6> and ECCP1CON<3:0>, respectively).

Figure 16-1 shows a simplified block diagram of PWM operation. All control registers are double-buffered and are loaded at the beginning of a new PWM cycle (the period boundary when the assigned timer resets) in order to prevent glitches on any of the outputs. The exception is the PWM Delay register, ECCP1DEL, which is loaded at either the duty cycle boundary or the boundary period (whichever comes first). Because of the buffering, the module waits until the assigned timer resets instead of starting immediately. This means that Enhanced PWM waveforms do not exactly match the standard PWM waveforms, but are instead offset by one full instruction cycle (4 TOSC).

As before, the user must manually configure the appropriate TRISD bits for output.

### 16.5.1 PWM OUTPUT CONFIGURATIONS

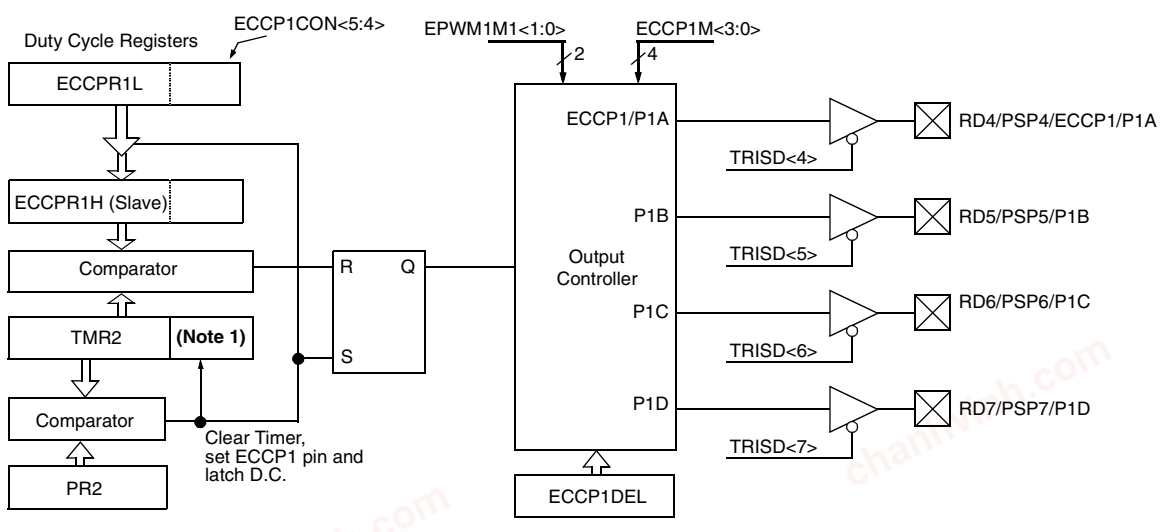
The EPWM1M<1:0> bits in the ECCP1CON register allow one of four configurations:

- Single Output
- Half-Bridge Output
- Full-Bridge Output, Forward mode
- Full-Bridge Output, Reverse mode

The Single Output mode is the standard PWM mode discussed in **Section 15.4 "PWM Mode"**. The Half-Bridge and Full-Bridge Output modes are covered in detail in the sections that follow.

The general relationship of the outputs in all configurations is summarized in Figure 16-2.

**FIGURE 16-1: SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODULE**



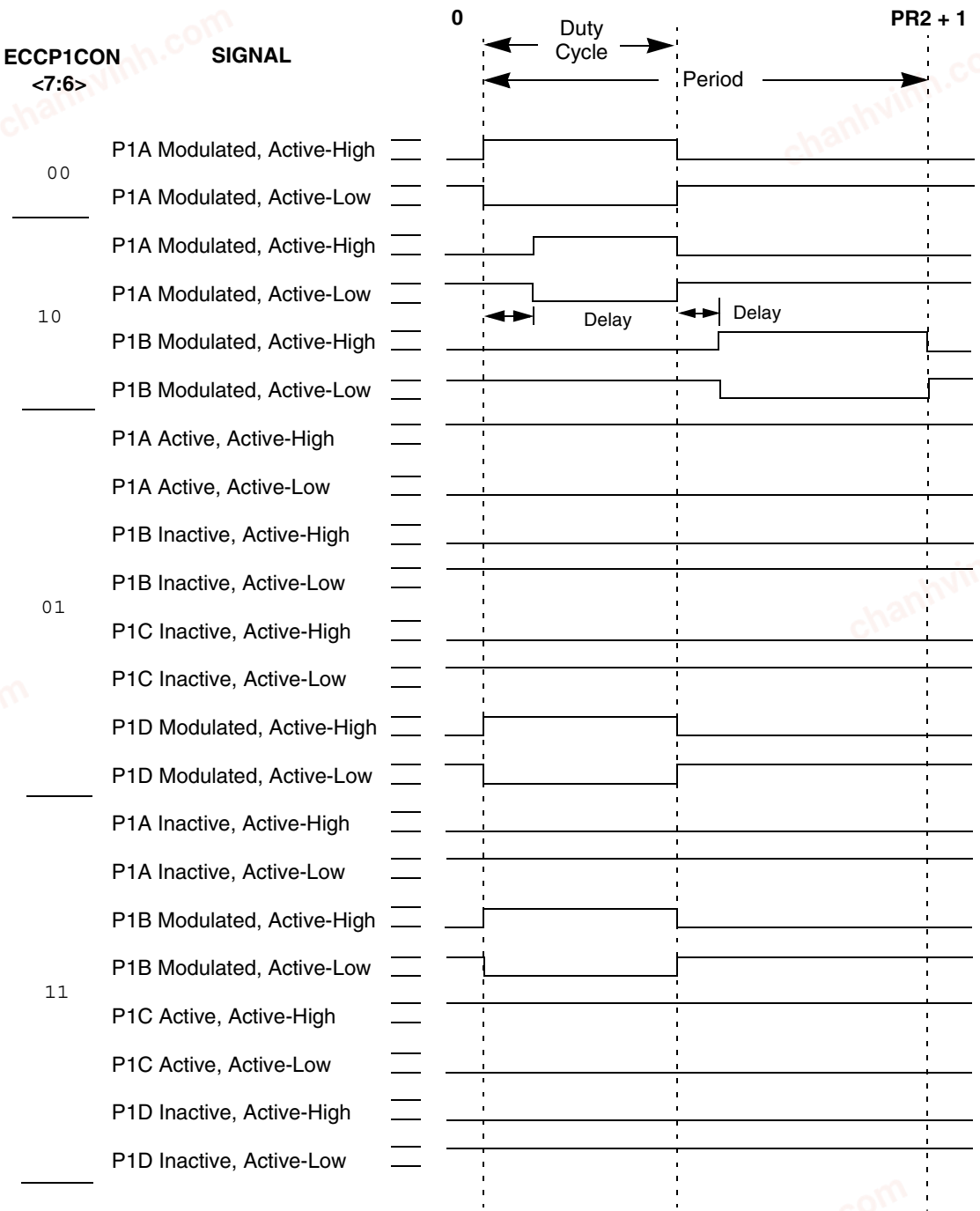

**Note:** The 8-bit TMR2 register is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler, to create the 10-bit time base.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 16-2: PWM OUTPUT RELATIONSHIPS**




**Relationships:**

- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Duty Cycle = TOSC * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * ECCP1DEL

© 2006 Microchip Technology Inc.

chanhvinh.com



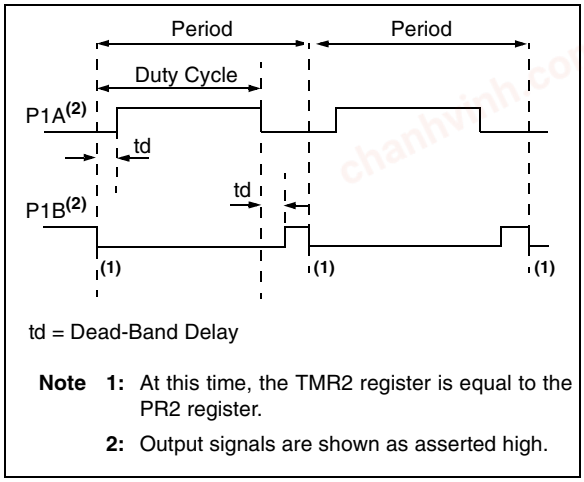
### 16.5.2 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The RD4/PSP4/ECCP1/P1A pin has the PWM output signal, while the RD5/PSP5/P1B pin has the complementary PWM output signal (Figure 16-3). This mode can be used for half-bridge applications, as shown in Figure 16-4, or for full-bridge applications where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in bridge power devices. The value of register ECCP1DEL dictates the number of clock cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 16.5.4 "Programmable Dead-Band Delay"** for more details of the dead-band delay operations.

Since the P1A and P1B outputs are multiplexed with the PORTD<4> and PORTD<5> data latches, the TRISD<4> and TRISD<5> bits must be cleared to configure P1A and P1B as outputs.

**FIGURE 16-3: HALF-BRIDGE PWM OUTPUT**



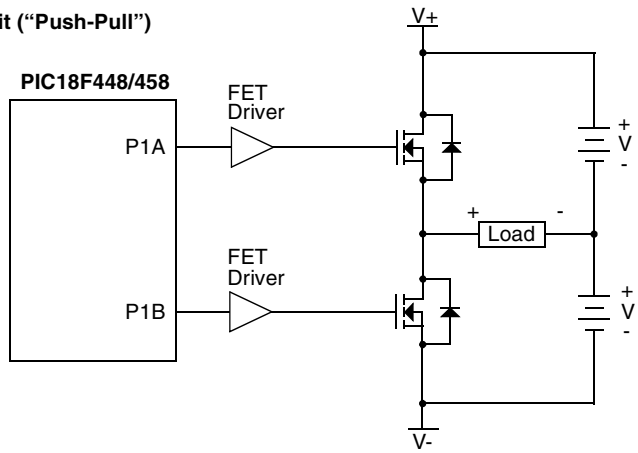

td = Dead-Band Delay

**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

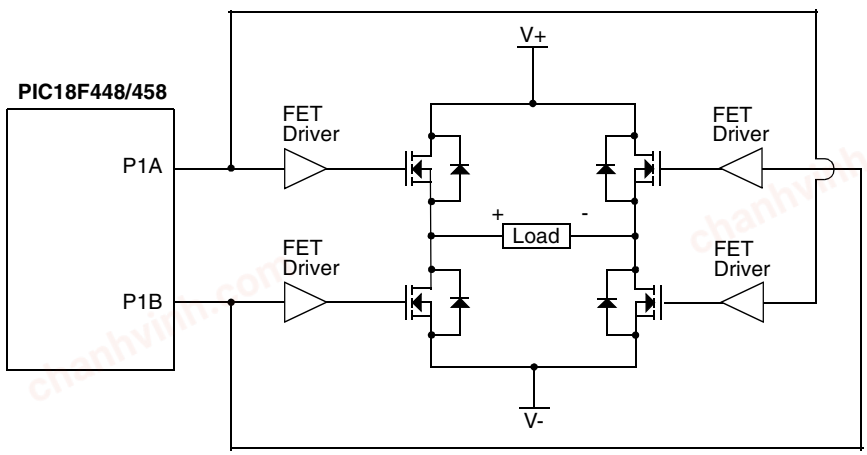
**2:** Output signals are shown as asserted high.

**FIGURE 16-4: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS**

**Standard Half-Bridge Circuit ("Push-Pull")**




**Half-Bridge Output Driving a Full-Bridge Circuit**





© 2006 Microchip Technology Inc.

chanhvinh.com

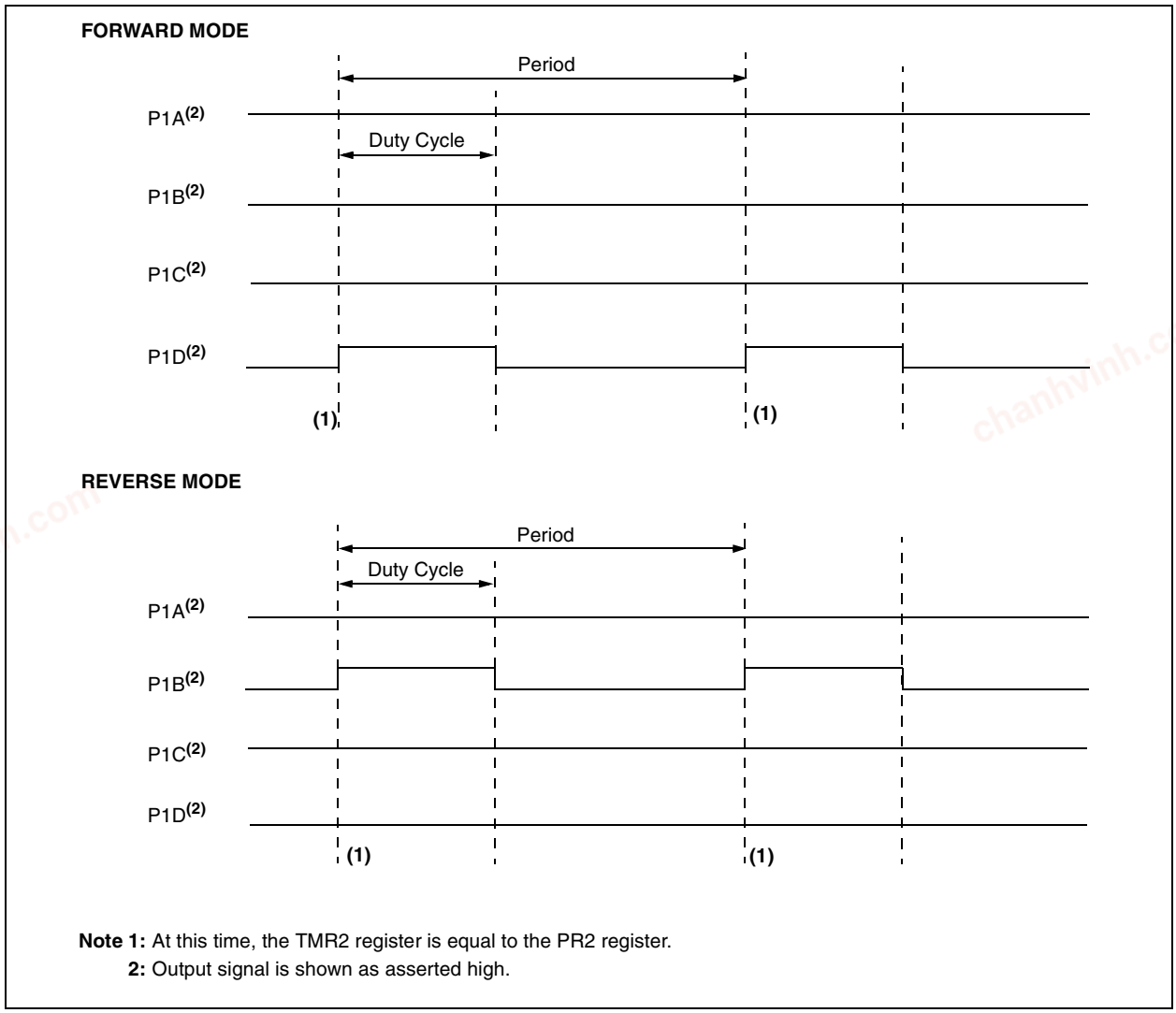


### 16.5.3 FULL-BRIDGE MODE

In Full-Bridge Output mode, four pins are used as outputs; however, only two outputs are active at a time. In the Forward mode, pin RD4/PSP4/ECCP1/P1A is continuously active and pin RD7/PSP7/P1D is modulated. In the Reverse mode, RD6/PSP6/P1C pin is continuously active and RD5/PSP5/P1B pin is modulated. These are illustrated in Figure 16-5.

P1A, P1B, P1C and P1D outputs are multiplexed with the PORTD<4:7> data latches. The TRISD<4:7> bits must be cleared to make the P1A, P1B, P1C and P1D pins output.

**FIGURE 16-5: FULL-BRIDGE PWM OUTPUT**

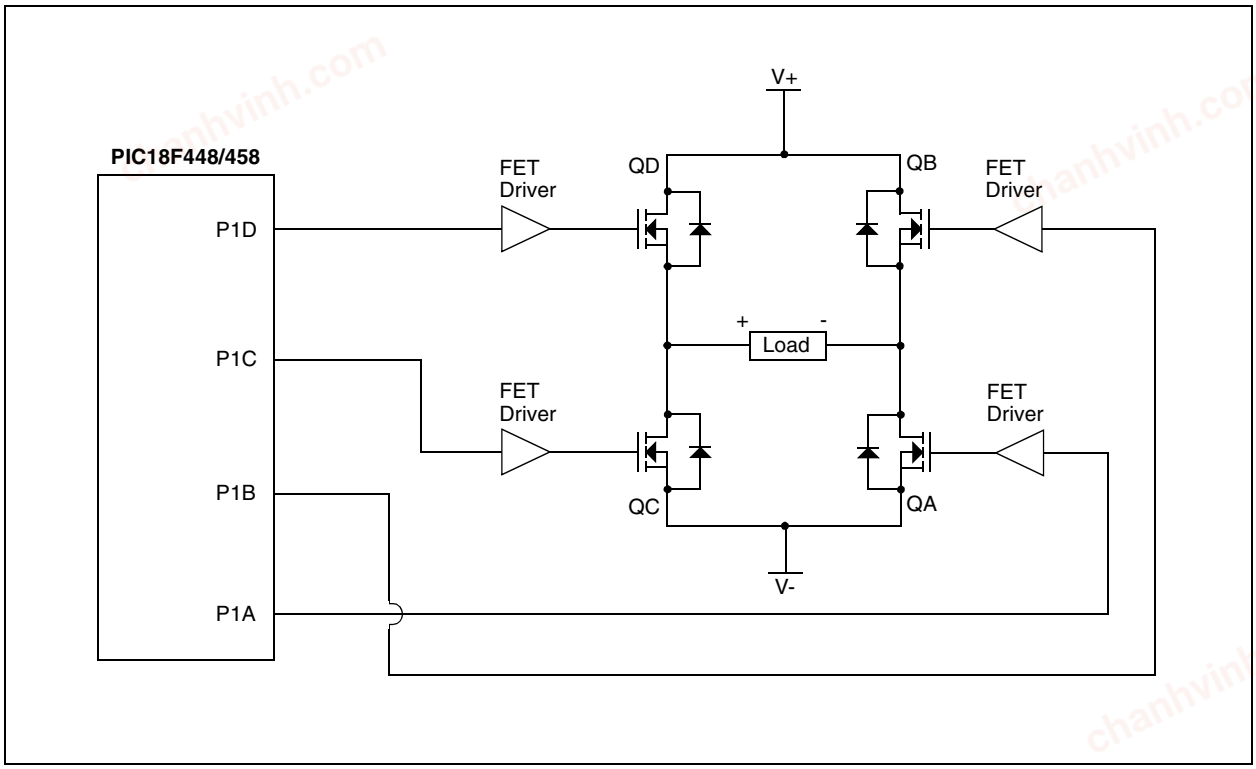



**Note 1:** At this time, the TMR2 register is equal to the PR2 register.
**2:** Output signal is shown as asserted high.

chanhvinh.com



**FIGURE 16-6: EXAMPLE OF FULL-BRIDGE APPLICATION**




### 16.5.3.1 Direction Change in Full-Bridge Mode

In the Full-Bridge Output mode, the EPWM1M1 bit in the ECCP1CON register allows the user to control the forward/reverse direction. When the application firmware changes this direction control bit, the ECCP1 module will assume the new direction on the next PWM cycle. The current PWM cycle still continues, however, the non-modulated outputs, P1A and P1C signals, will transition to the new direction TOSC, 4 TOSC or 16 TOSC earlier (for T2CKRS<1:0> = 00, 01 or 1x, respectively) before the end of the period. During this transition cycle, the modulated outputs, P1B and P1D, will go to the inactive state (Figure 16-7).

Note that in the Full-Bridge Output mode, the ECCP module does not provide any dead-band delay. In general, since only one output is modulated at all times, dead-band delay is not required. However, there is a situation where a dead-band delay might be required. This situation occurs when all of the following conditions are true:

1. The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than turn-on time.

Figure 16-8 shows an example where the PWM direction changes from forward to reverse at a near 100% duty cycle. At time t1, the outputs P1A and P1D become inactive, while output P1C becomes active. In this example, since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current flows through power devices QB and QD (see Figure 16-6) for the duration of 't'. The same phenomenon will occur to power devices QA and QC for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, one of the following requirements must be met:

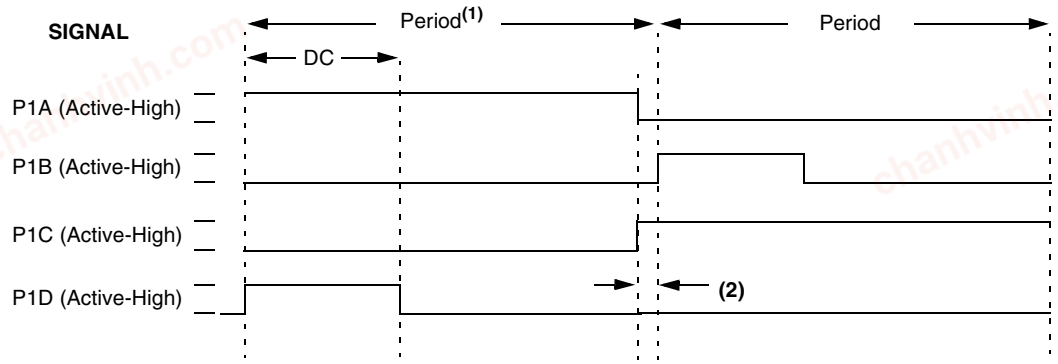
1. Avoid changing PWM output direction at or near 100% duty cycle.
2. Use switch drivers that compensate the slow turn off of the power devices. The total turn-off time ($t_{off}$) of the power device and the driver must be less than the turn-on time ($t_{on}$).


© 2006 Microchip Technology Inc.

chanhvinh.com



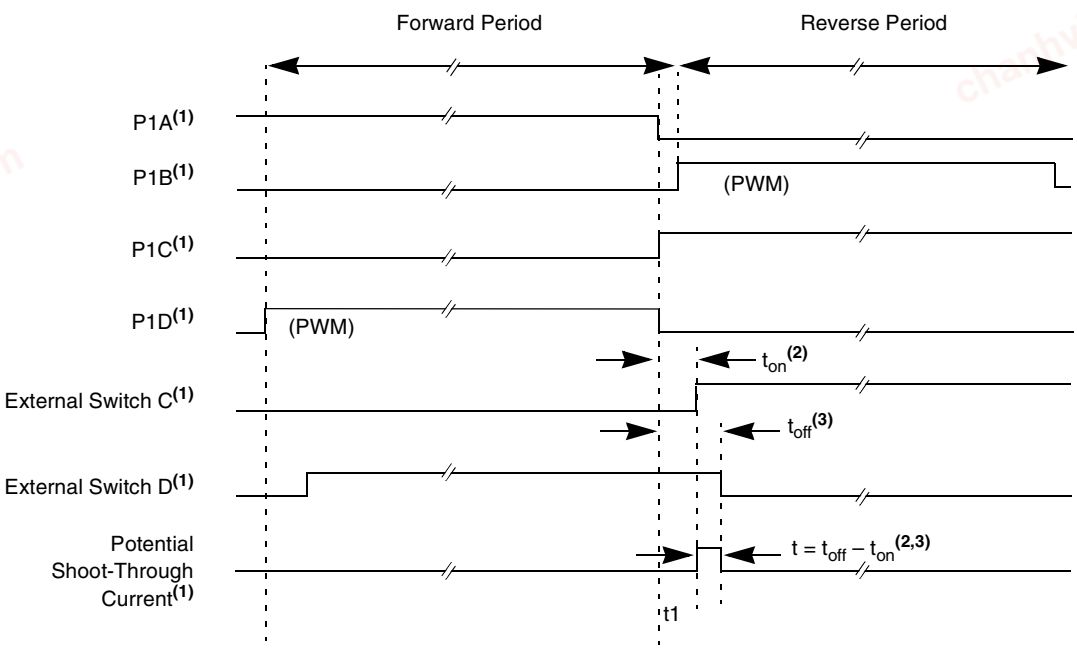
**FIGURE 16-7: PWM DIRECTION CHANGE**




**Note 1:** The direction bit in the ECCP1 Control Register (ECCP1CON.EPWM1M1) is written any time during the PWM cycle.

**2:** The P1A and P1C signals switch at intervals of TOSC, 4 TOSC or 16 TOSC, depending on the Timer2 prescaler value earlier when changing direction. The modulated P1B and P1D signals are inactive at this time.

**FIGURE 16-8: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE**




**Note 1:** All signals are shown as active-high.

**2:** $t_{on}$ is the turn-on delay of power switch and driver.

**3:** $t_{off}$ is the turn-off delay of power switch and driver.

chanhvinh.com



### 16.5.4 PROGRAMMABLE DEAD-BAND DELAY

In half-bridge or full-bridge applications, where all power switches are modulated at the PWM frequency at all times, the power switches normally require longer time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on and the other turned off), both switches will be on for a short period of time until one switch completely turns off. During this time, a very high current (*shoot-through current*) flows through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on the power switch is normally delayed to allow the other switch to completely turn off.

In the Half-Bridge Output mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. See Figure 16-3 for illustration. The ECCP1DEL register (Register 16-2) sets the amount of delay.

### 16.5.5 SYSTEM IMPLEMENTATION

When the ECCP module is used in the PWM mode, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins. When the microcontroller powers up, all of the I/O pins are in the high-impedance state. The external pull-up and pull-down resistors must keep the power switch devices in the off state until the microcontroller drives the I/O pins with the proper signal levels, or activates the PWM output(s).

### 16.5.6 START-UP CONSIDERATIONS

Prior to enabling the PWM outputs, the P1A, P1B, P1C and P1D latches may not be in the proper states. Enabling the TRISD bits for output at the same time with the ECCP1 module may cause damage to the power switch devices. The ECCP1 module must be enabled in the proper output mode with the TRISD bits enabled as inputs. Once the ECCP1 completes a full PWM cycle, the P1A, P1B, P1C and P1D output latches are properly initialized. At this time, the TRISD bits can be enabled for outputs to start driving the power switch devices. The completion of a full PWM cycle is indicated by the TMR2IF bit going from a '0' to a '1'.

### 16.5.7 OUTPUT POLARITY CONFIGURATION

The ECCP1M<1:0> bits in the ECCP1CON register allow user to choose the logic conventions (asserted high/low) for each of the outputs.

The PWM output polarities must be selected before the PWM outputs are enabled. Charging the polarity configuration while the PWM outputs are active is not recommended since it may result in unpredictable operation.

**REGISTER 16-2: ECCP1DEL: PWM DELAY 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 |
|---|---|---|---|---|---|---|---|
| EPDC7 | EPDC6 | EPDC5 | EPDC4 | EPDC3 | EPDC2 | EPDC1 | EPDC0 |
| bit 7 | | | | | | | bit 0 |

bit 7-0 **EPDC<7:0>:** PWM Delay Count for Half-Bridge Output Mode bits
Number of FOSC/4 (TOSC * 4) cycles between the P1A transition and the P1B transition.

| **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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



### 16.5.8 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the ECCP1 module for PWM operation:

1. Configure the PWM module:
   a) Disable the ECCP1/P1A, P1B, P1C and/or P1D outputs by setting the respective TRISD bits.
   b) Set the PWM period by loading the PR2 register.
   c) Set the PWM duty cycle by loading the ECCPR1L register and ECCP1CON<5:4> bits.
   d) Configure the ECCP1 module for the desired PWM operation by loading the ECCP1CON register with the appropriate value. With the ECCP1M<3:0> bits, select the active-high/low levels for each PWM output. With the EPWM1M<1:0> bits, select one of the available output modes.
   e) For Half-Bridge Output mode, set the dead-band delay by loading the ECCP1DEL register with the appropriate value.
2. Configure and start TMR2:
   a) Clear the TMR2 interrupt flag bit by clearing the TMR2IF bit in the PIR1 register.
   b) Set the TMR2 prescale value by loading the T2CKPS bits (T2CON<1:0>).
   c) Enable Timer2 by setting the TMR2ON bit (T2CON<2>) register.
3. Enable PWM outputs after a new cycle has started:
   a) Wait until TMR2 overflows (TMR2IF bit becomes a '1'). The new PWM cycle begins here.
   b) Enable the ECCP1/P1A, P1B, P1C and/or P1D pin outputs by clearing the respective TRISD bits.

**TABLE 16-5: REGISTERS ASSOCIATED WITH ENHANCED 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 |
| RCON | IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 0--1 110q | 0--0 011q |
| IPR2 | — | CMIP | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP | -1-1 1111 | -1-1 1111 |
| PIR2 | — | CMIF | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE | -0-0 0000 | -0-0 0000 |
| TMR2 | Timer2 Module Register | | | | | | | | 0000 0000 | 0000 0000 |
| PR2 | Timer2 Module Period Register | | | | | | | | 1111 1111 | 1111 1111 |
| T2CON | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | -000 0000 |
| TRISD | PORTD Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| ECCPR1H | Enhanced Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| ECCPR1L | Enhanced Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| ECCP1CON | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 0000 0000 | 0000 0000 |
| ECCPAS | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 | 0000 0000 | 0000 0000 |
| ECCP1DEL | EPDC7 | EPDC6 | EPDC5 | EPDC4 | EPDC3 | EPDC2 | EPDC1 | EPDC0 | 0000 0000 | uuuu uuuu |

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

© 2006 Microchip Technology Inc.

chanhvinh.com



## 16.6 Enhanced CCP Auto-Shutdown

When the ECCP is programmed for any of the PWM modes, the output pins associated with its function may be configured for auto-shutdown.

Auto-shutdown allows the internal output of either of the two comparator modules, or the external interrupt 0, to asynchronously disable the ECCP output pins. Thus, an external analog or digital event can discontinue an ECCP sequence. The comparator output(s) to be used is selected by setting the proper mode bits in the ECCPAS register. To use external interrupt INT0 as a shutdown event, INT0IE must be set. To use either of the comparator module outputs as a shutdown event, corresponding comparators must be enabled. When a shutdown occurs, the selected output values (PSSACn, PSSBDn) are written to the ECCP port pins.

The internal shutdown signal is gated with the outputs and will immediately and asynchronously disable the outputs. If the internal shutdown is still in effect at the time a new cycle begins, that entire cycle is suppressed, thus eliminating narrow, glitchy pulses.

The ECCPASE bit is set by hardware upon a comparator event and can only be cleared in software. The ECCP outputs can be re-enabled only by clearing the ECCPASE bit.

The Auto-Shutdown mode can be manually entered by writing a '1' to the ECCPASE bit.

**REGISTER 16-3: ECCPAS: ENHANCED CAPTURE/COMPARE/PWM AUTO-SHUTDOWN CONTROL 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 |
|---|---|---|---|---|---|---|---|
| ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **ECCPASE:** ECCP Auto-Shutdown Event Status bit
0 = ECCP outputs enabled, no shutdown event
1 = A shutdown event has occurred, must be reset in software to re-enable ECCP

bit 6-4 **ECCPAS<2:0>:** ECCP Auto-Shutdown bits
000 = No auto-shutdown enabled, comparators have no effect on ECCP
001 = Comparator 1 output will cause shutdown
010 = Comparator 2 output will cause shutdown
011 = Either Comparator 1 or 2 can cause shutdown
100 = INT0
101 = INT0 or Comparator 1 output
110 = INT0 or Comparator 2 output
111 = INT0 or Comparator 1 or Comparator 2 output

bit 3-2 **PSSACn:** Pins A and C Shutdown State Control bits
00 = Drive Pins A and C to '0'
01 = Drive Pins A and C to '1'
1x = Pins A and C tri-state

bit 1-0 **PSSBDn:** Pins B and D Shutdown State Control bits
00 = Drive Pins B and D to '0'
01 = Drive Pins B and D to '1'
1x = Pins B and D tri-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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



## 17.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

### 17.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit ($I^2C$)
  - Full Master mode
  - Slave mode (with general address call)

The $I^2C$ interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

### 17.2 Control Registers

The MSSP module has three associated registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual configuration bits differ significantly, depending on whether the MSSP module is operated in SPI or $I^2C$ mode.

Additional details are provided under the individual sections.

### 17.3 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:

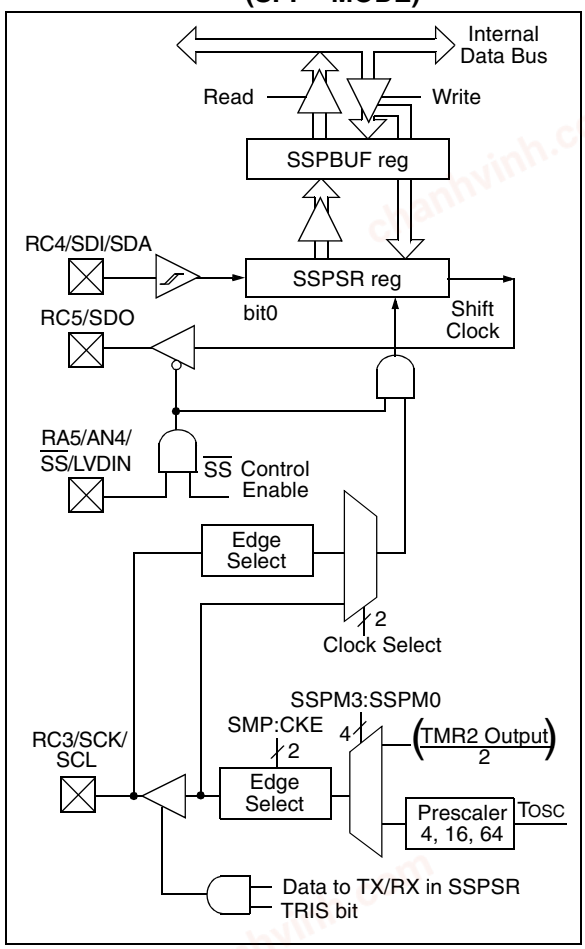
- Serial Data Out (SDO) – RC5/SDO
- Serial Data In (SDI) – RC4/SDI/SDA
- Serial Clock (SCK) – RC3/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SS}$) – RA5/AN4/$\overline{SS}$/LVDIN

Figure 17-1 shows the block diagram of the MSSP module when operating in SPI mode.

**FIGURE 17-1: MSSP BLOCK DIAGRAM (SPI™ MODE)**




© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

**REGISTER 17-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| bit 7 | | | | | | | bit 0 |

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Edge Select bit
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state
**Note:** Polarity of clock state is set by the CKP bit (SSPCON1<4>).

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit
Used in $I^2C$ mode only.

bit 4 **P:** Stop bit
Used in $I^2C$ mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit
Used in $I^2C$ mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ Information bit
Used in $I^2C$ mode only.

bit 1 **UA:** Update Address bit
Used in $I^2C$ mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 17-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)**

| 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 (Transmit mode only)
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
SPI Slave 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.The user must read the SSPBUF even if only transmitting data to avoid setting overflow (must be cleared in software).
0 = No overflow

**Note:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins
0 = Disables serial port and configures these pins as I/O port pins

**Note:** When enabled, these pins must be properly configured as input or output.

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0011 = SPI Master mode, clock = TMR2 output/2
0010 = SPI Master mode, clock = FOSC/64
0001 = SPI Master mode, clock = FOSC/16
0000 = SPI Master mode, clock = FOSC/4

**Note:** Bit combinations not specifically listed here are either reserved or implemented in $I^2C$ mode only.

| **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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.3.2 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)

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<0>) and the interrupt flag bit SSPIF are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the Write Collision detect bit, WCOL (SSPCON1<7>), 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. Buffer Full bit, BF (SSPSTAT<0>), 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 17-1 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) indicates the various status conditions.

**EXAMPLE 17-1: LOADING THE SSPBUF (SSPSR) REGISTER**

```
LOOP BTFSS   SSPSTAT, BF     ;Has data been received(transmit complete)?
     BRA     LOOP            ;No
     MOVF    SSPBUF, W       ;WREG reg = contents of SSPBUF
     MOVWF   RXDATA          ;Save in user RAM, if data is meaningful
     MOVF    TXDATA, W       ;W reg = contents of TXDATA
     MOVWF   SSPBUF          ;New data to xmit
```


© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.3.3 ENABLING SPI I/O

To enable the serial port, SSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize 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 as follows:

- 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 TRISA<5> bit set

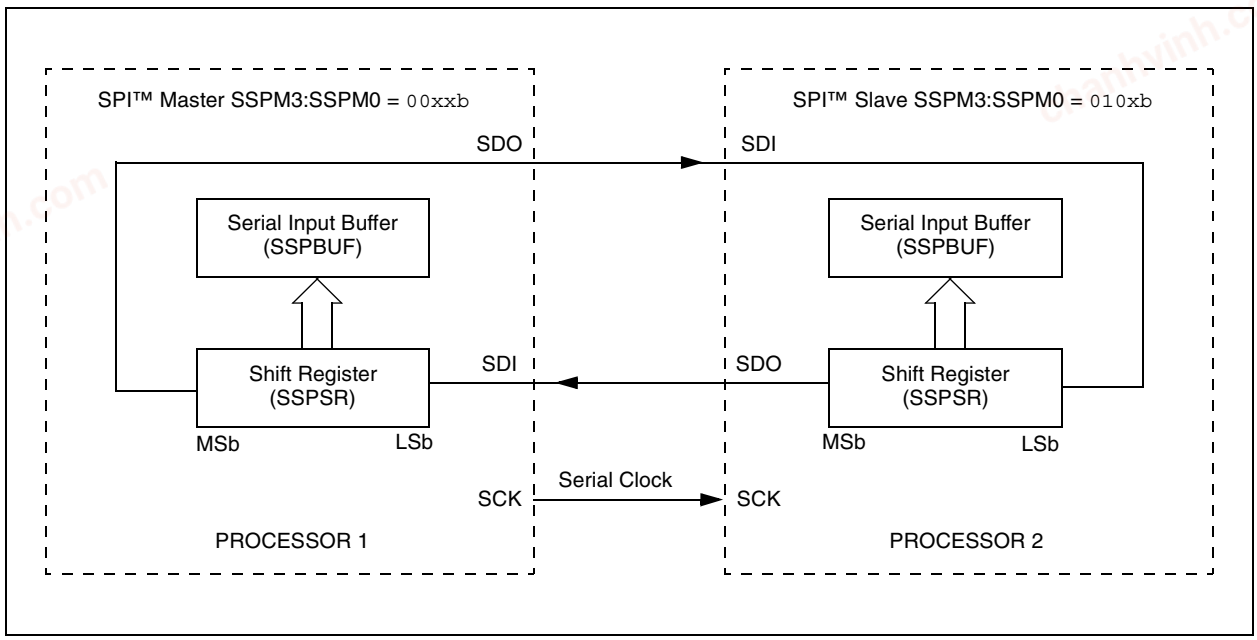
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

### 17.3.4 TYPICAL CONNECTION

Figure 17-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

**FIGURE 17-2: SPI™ MASTER/SLAVE CONNECTION**



chanhvinh.com



### 17.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 17-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

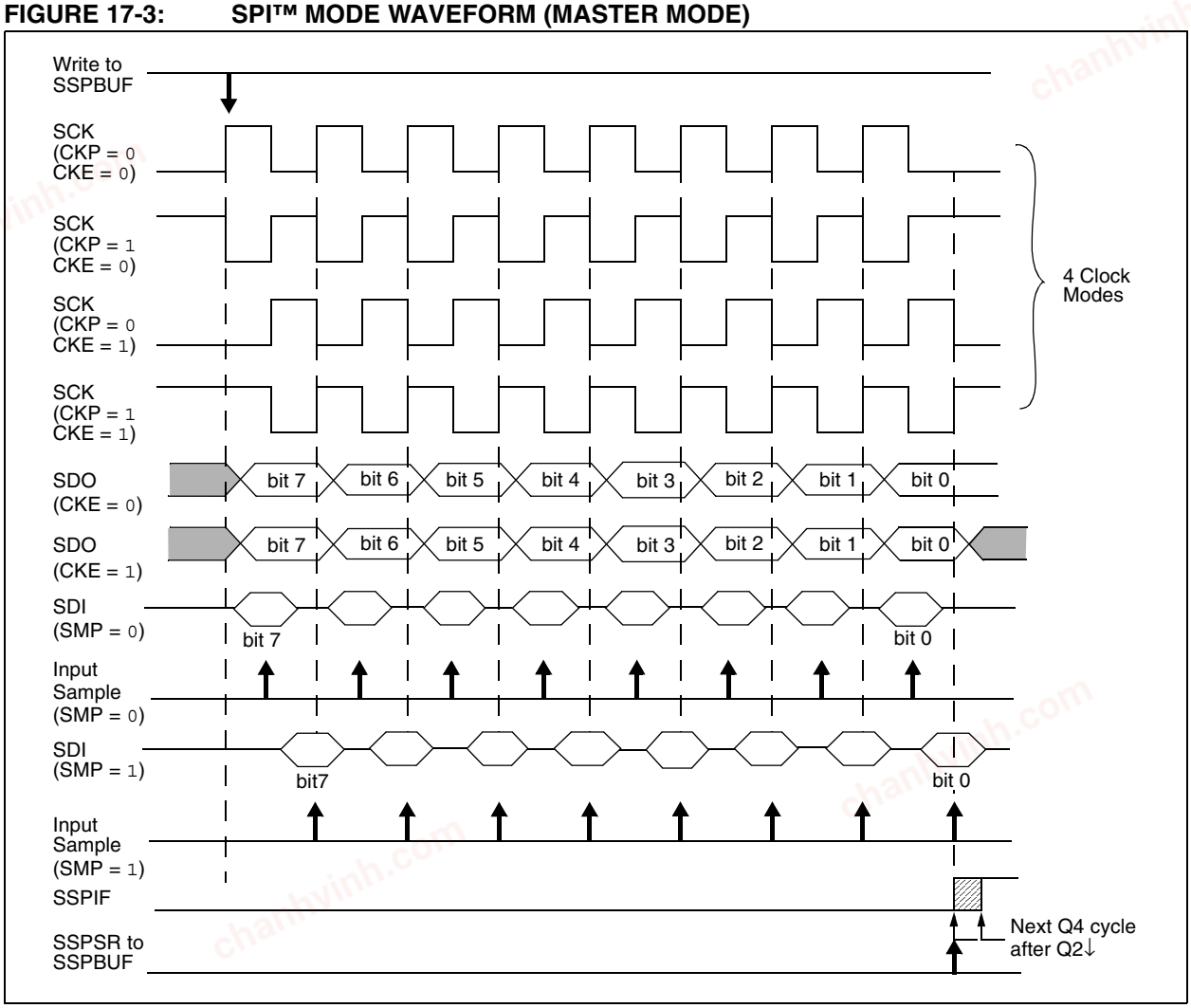
The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 17-3, Figure 17-5 and Figure 17-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

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

Figure 17-3 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 17-3: SPI™ MODE WAVEFORM (MASTER MODE)**





© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.3.6 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. Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit (SSPCON1<4>).

### 17.3.7 SLAVE SELECT SYNCHRONIZATION

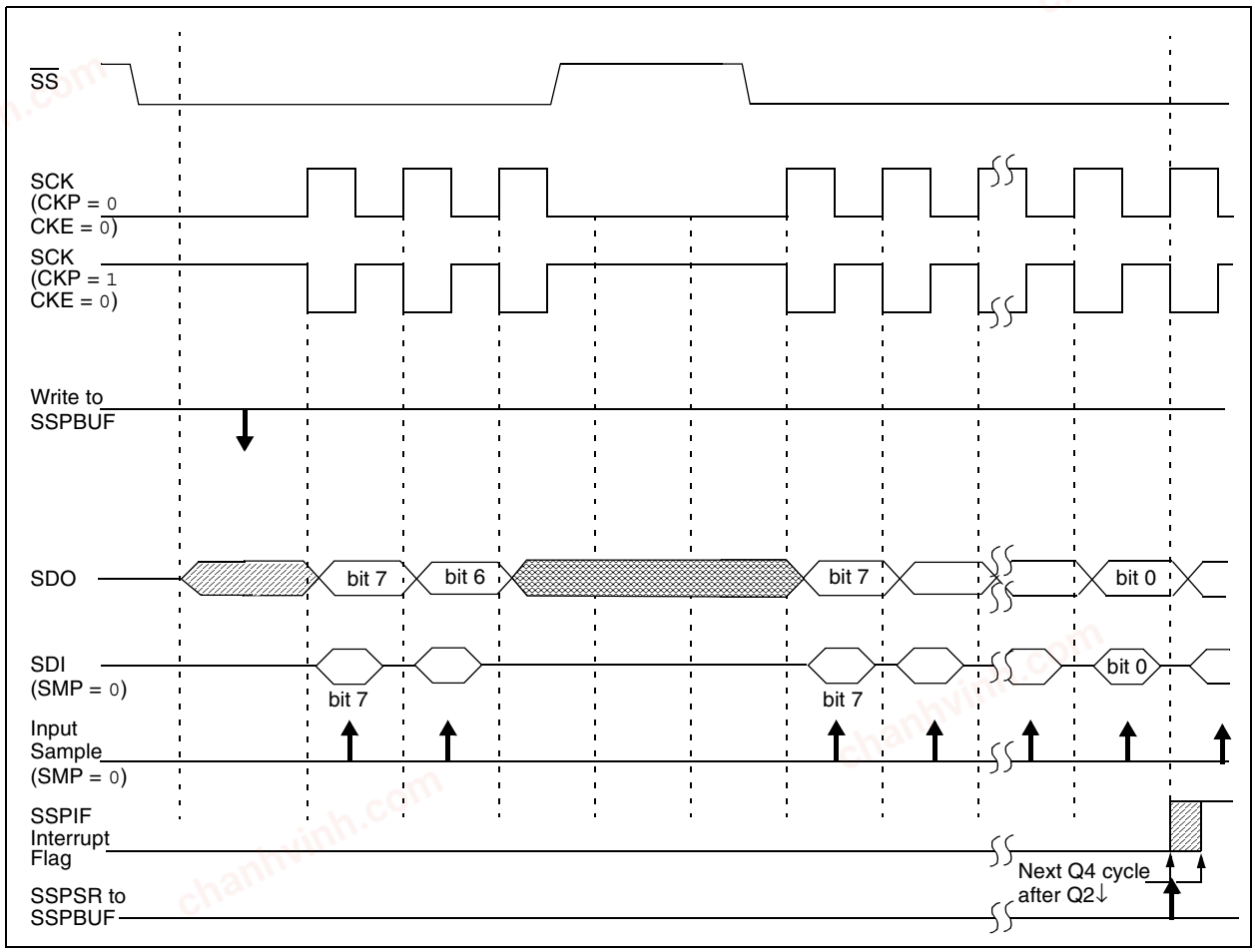
The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the $\overline{SS}$ pin to function as an input. The data latch must be high. When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven. When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

| | |
|---|---|
| **Note 1:** | When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to VDD. |
| **2:** | If the SPI is used in Slave mode with CKE set, then the $\overline{SS}$ pin control must be enabled. |

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

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

**FIGURE 17-4: SLAVE SYNCHRONIZATION WAVEFORM**



chanhvinh.com



**FIGURE 17-5: SPI™ MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

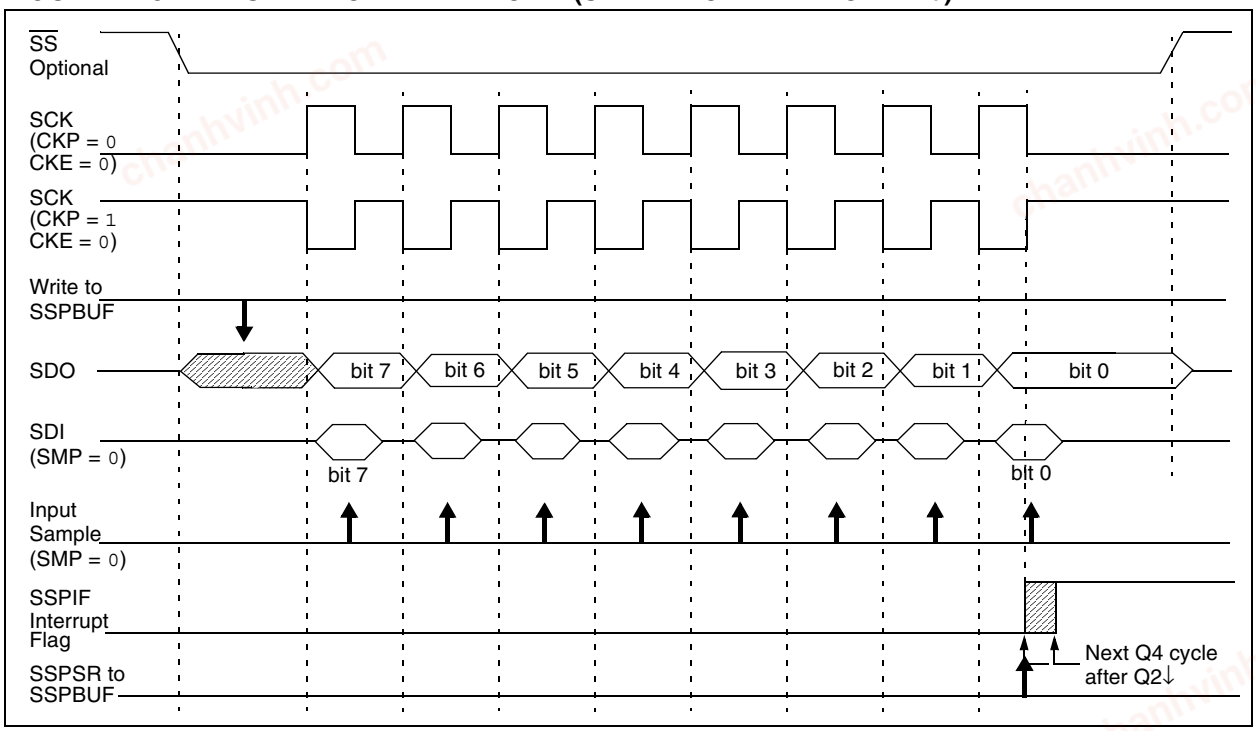
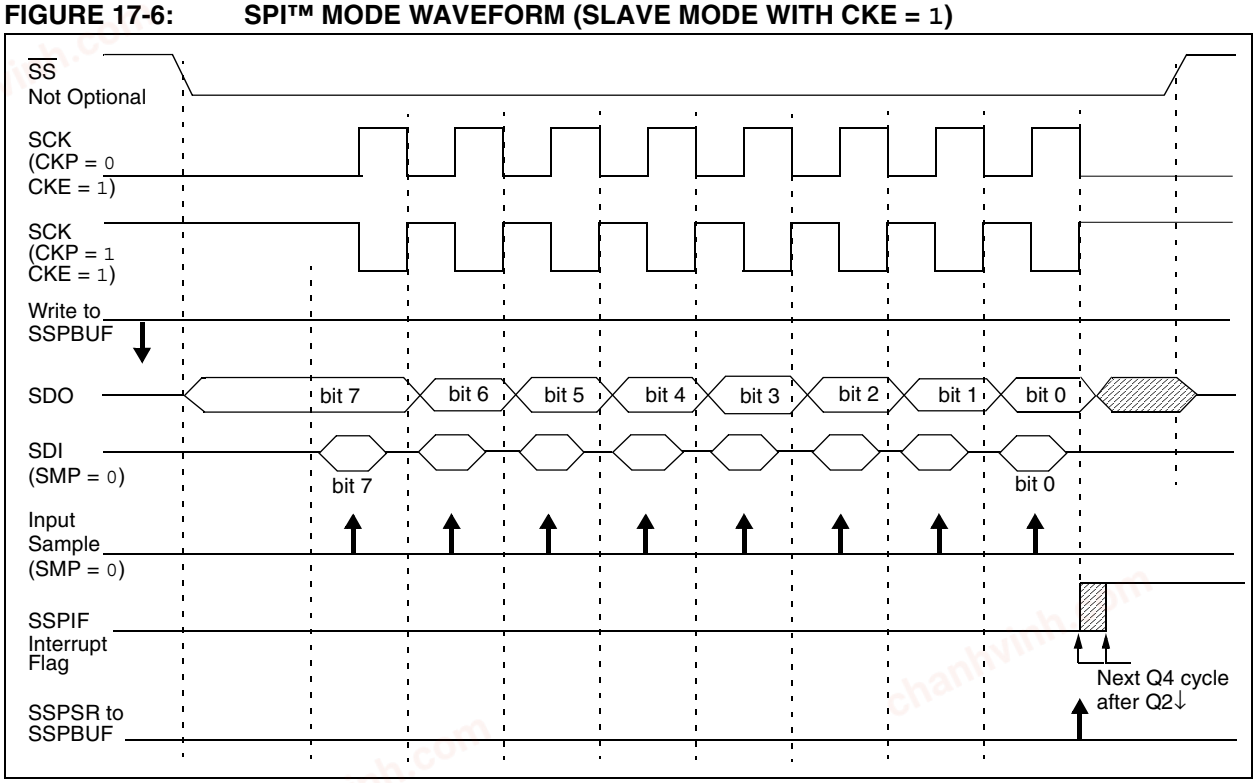



**FIGURE 17-6: SPI™ MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**





© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.3.8 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 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device from Sleep.

### 17.3.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 17.3.10 BUS MODE COMPATIBILITY

Table 17-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

**TABLE 17-1: 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 an SMP bit which controls when the data is sampled.

**TABLE 17-2: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| TRISC | PORTC Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| TRISA | — | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | -111 1111 | -111 1111 |
| SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 0000 0000 |
| SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the MSSP in SPI™ mode.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

chanhvinh.com



## 17.4 I²C Mode

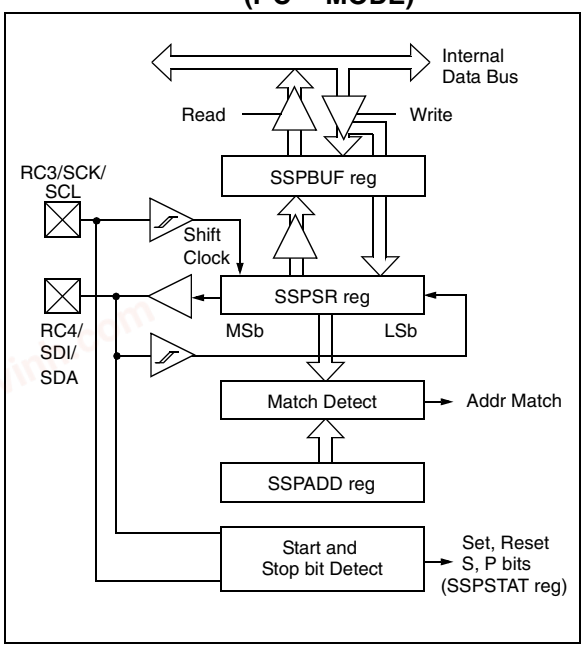
The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial clock (SCL) – RC3/SCK/SCL
- Serial data (SDA) – RC4/SDI/SDA

The user must configure these pins as inputs or outputs through the TRISC<4:3> bits.

**FIGURE 17-7: MSSP BLOCK DIAGRAM (I²C™ MODE)**




### 17.4.1 REGISTERS

The MSSP module has six registers for I²C operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)

SSPCON1, SSPCON2 and SSPSTAT are the control and status registers in I²C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

SSPADD register holds the slave device address when the SSP is configured in I²C Slave mode. When the SSP is configured in Master mode, the lower seven bits of SSPADD act as the Baud Rate Generator reload value.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 17-3: SSPSTAT: MSSP STATUS REGISTER (I²C MODE)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| bit 7 | | | | | | | bit 0 |

bit 7 **SMP:** Slew Rate Control bit
In 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:** SMBus Select bit
In Master or Slave mode:
1 = Enable SMBus specific inputs
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit
In Master mode:
Reserved.
In Slave mode:
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
1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

**Note:** This bit is cleared on Reset and when SSPEN is cleared.

bit 3 **S:** Start bit
1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

**Note:** This bit is cleared on Reset and when SSPEN is cleared.

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ Information bit (I²C mode only)
In Slave mode:
1 = Read
0 = Write

**Note:** This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{ACK}$ bit.

In Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress

**Note:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode.

bit 1 **UA:** Update Address bit (10-bit Slave 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
In Transmit mode:
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
In Receive mode:
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 |

chanhvinh.com



**REGISTER 17-4: SSPCON1: MSSP CONTROL REGISTER 1 ($I^2C$ MODE)**

| 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

In Master Transmit mode:

1 = A write to the SSPBUF register was attempted while the $I^2C$ conditions were not valid for a transmission to be started (must be cleared in software)

0 = No collision

In Slave Transmit mode:

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

In Receive mode (Master or Slave modes):

This is a “don’t care” bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)

0 = No overflow

In Transmit mode:

This is a “don’t care” bit in Transmit mode.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

**Note:** When enabled, the SDA and SCL pins must be properly configured as input or output.

bit 4 **CKP:** SCK Release Control bit

In Slave mode:

1 = Release clock

0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:

Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

1111 = $I^2C$ Slave mode, 10-bit address with Start and Stop bit interrupts enabled

1110 = $I^2C$ Slave mode, 7-bit address with Start and Stop bit interrupts enabled

1011 = $I^2C$ Firmware Controlled Master mode (Slave Idle)

1000 = $I^2C$ Master mode, clock = FOSC/(4 * (SSPADD + 1))

0111 = $I^2C$ Slave mode, 10-bit address

0110 = $I^2C$ Slave mode, 7-bit address

**Note:** Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

| **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 |



© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 17-5: SSPCON2: MSSP CONTROL REGISTER 2 ($I^2C$ MODE)**

| 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 (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 (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)
1 = Not Acknowledge
0 = Acknowledge

**Note:** Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)
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 (Master Mode only)
1 = Enables Receive mode for $I^2C$
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit (Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enable/Stretch Enable bit
In Master mode:
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is enabled for slave transmit only (Legacy 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 |

**Note:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the $I^2C$ module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

chanhvinh.com



### 17.4.2 OPERATION

The MSSP module functions are enabled by setting MSSP Enable bit, SSPEN (SSPCON1<5>).

The SSPCON1 register allows control of the $I^2C$ operation. Four mode selection bits (SSPCON1<3:0>) allow one of the following $I^2C$ modes to be selected:

- $I^2C$ Master mode, clock = OSC/4 (SSPADD +1)
- $I^2C$ Slave mode (7-bit address)
- $I^2C$ Slave mode (10-bit address)
- $I^2C$ Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Firmware Controlled Master mode, slave is Idle

Selection of any $I^2C$ 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. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

### 17.4.3 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).

The $I^2C$ Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits.

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{ACK}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{ACK}$ pulse:

- The Buffer Full bit, BF (SSPSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPCON1<6>), was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF (PIR1<3>) 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 $I^2C$ specification, as well as the requirement of the MSSP module, are shown in timing parameter #100 and parameter #101.

#### 17.4.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8 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:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit BF is set.
3. An $\overline{ACK}$ pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF (PIR1<3>), is set (interrupt is generated if enabled) on the falling edge of the ninth SCL pulse.

In 10-bit Address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{W}$ (SSPSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '`11110 A9 A8 0`', where '`A9`' and '`A8`' are the two MSbs of the address. The sequence of events for 10-bit address is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits SSPIF, BF and bit UA (SSPSTAT<1>) 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.


© 2006 Microchip Technology Inc.

chanhvinh.com



#### 17.4.3.2 Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{ACK}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set or bit SSPOV (SSPCON1<6>) is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR1<3>) must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON2<0> = 1), RC3/SCK/SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit CKP (SSPCON1<4>). See **Section 17.4.4 "Clock Stretching"** for more detail.

#### 17.4.3.3 Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low regardless of SEN (see **Section 17.4.4 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then, pin RC3/SCK/SCL should be enabled by setting bit CKP (SSPCON1<4>). 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 17-9).

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. In this case, 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 next transmit data must be loaded into the SSPBUF register. Again, pin RC3/SCK/SCL must be enabled by setting bit CKP.

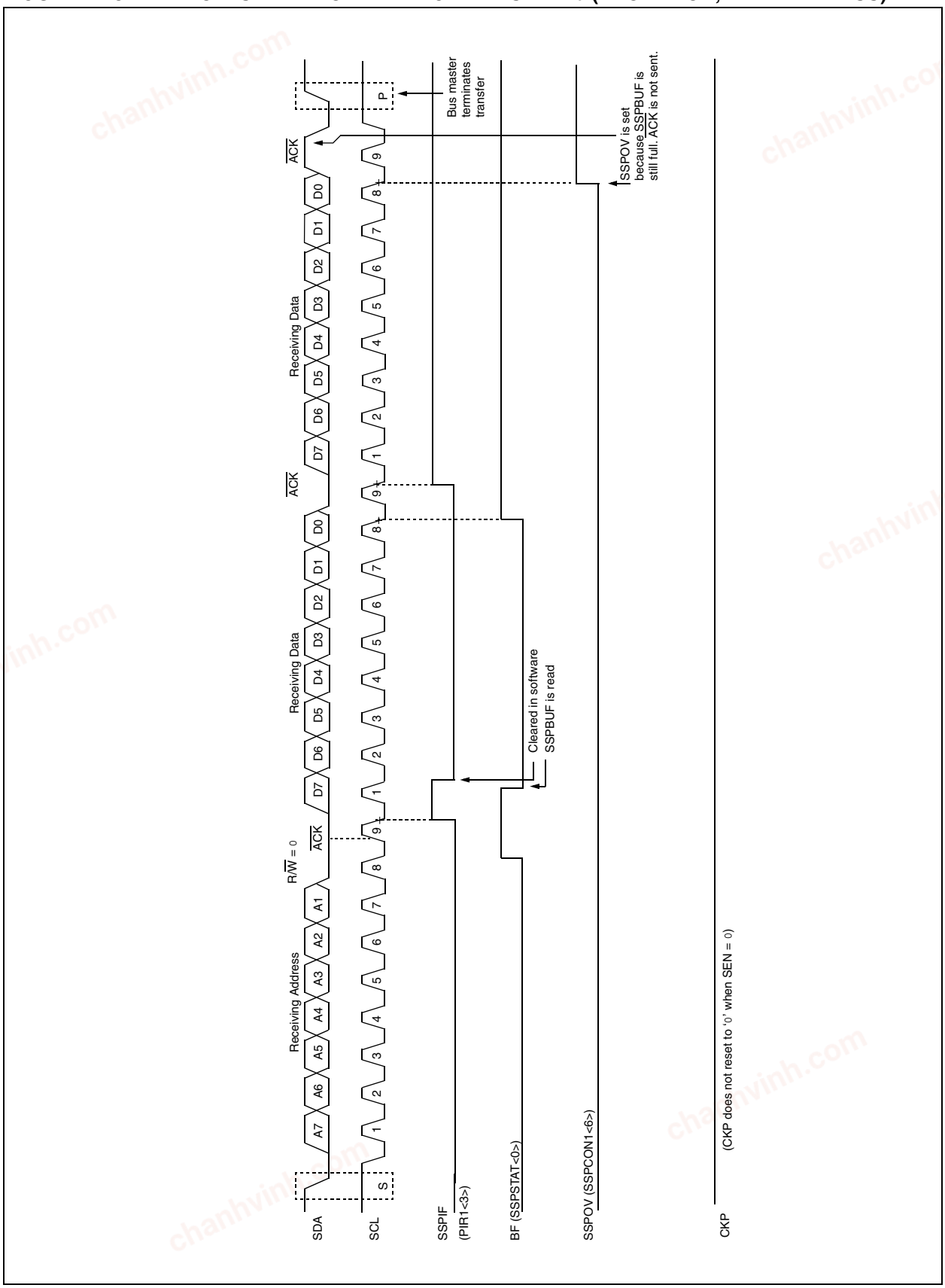
An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 17-8: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)**



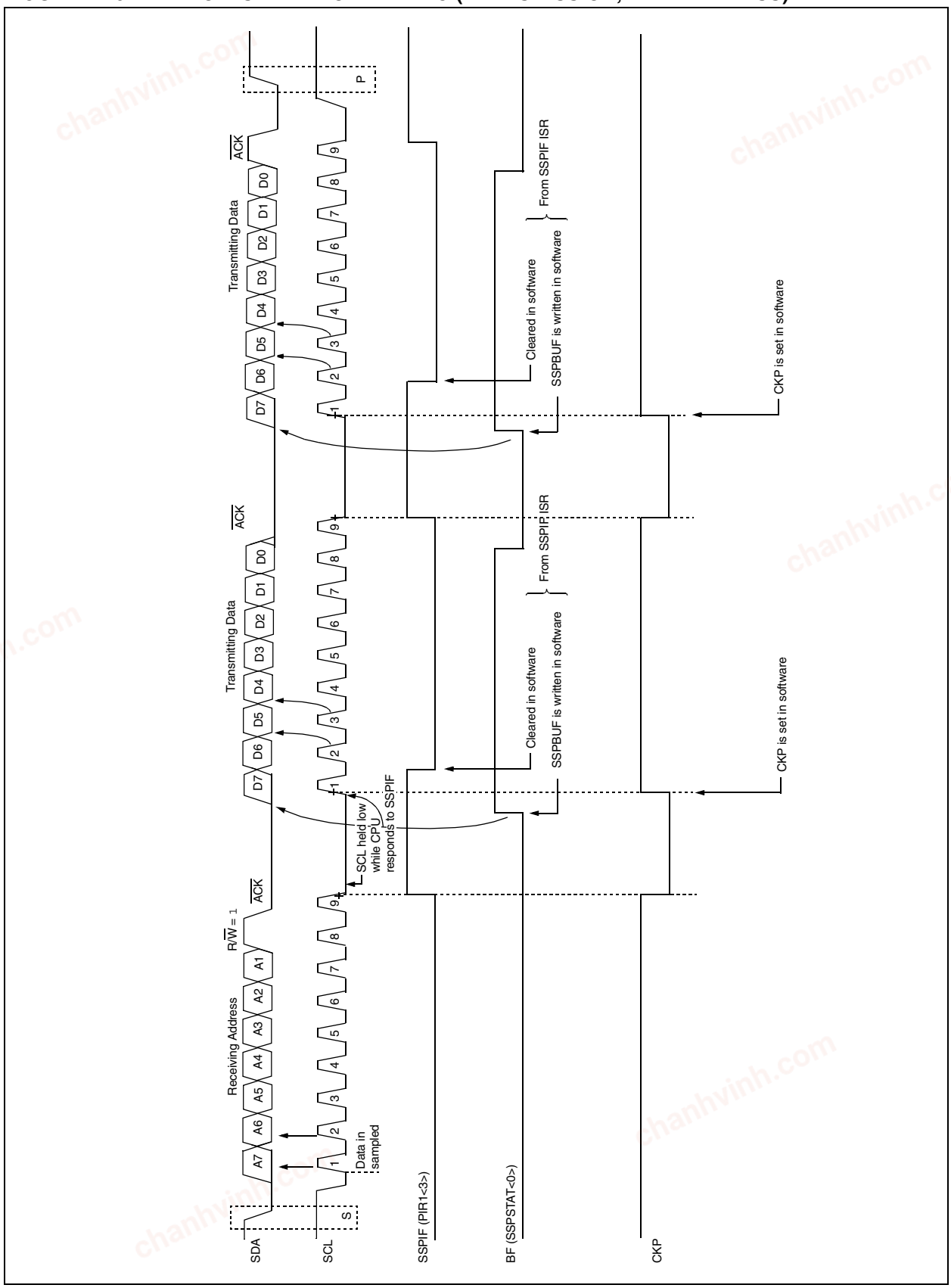


© 2006 Microchip Technology Inc.

chanhvinh.com



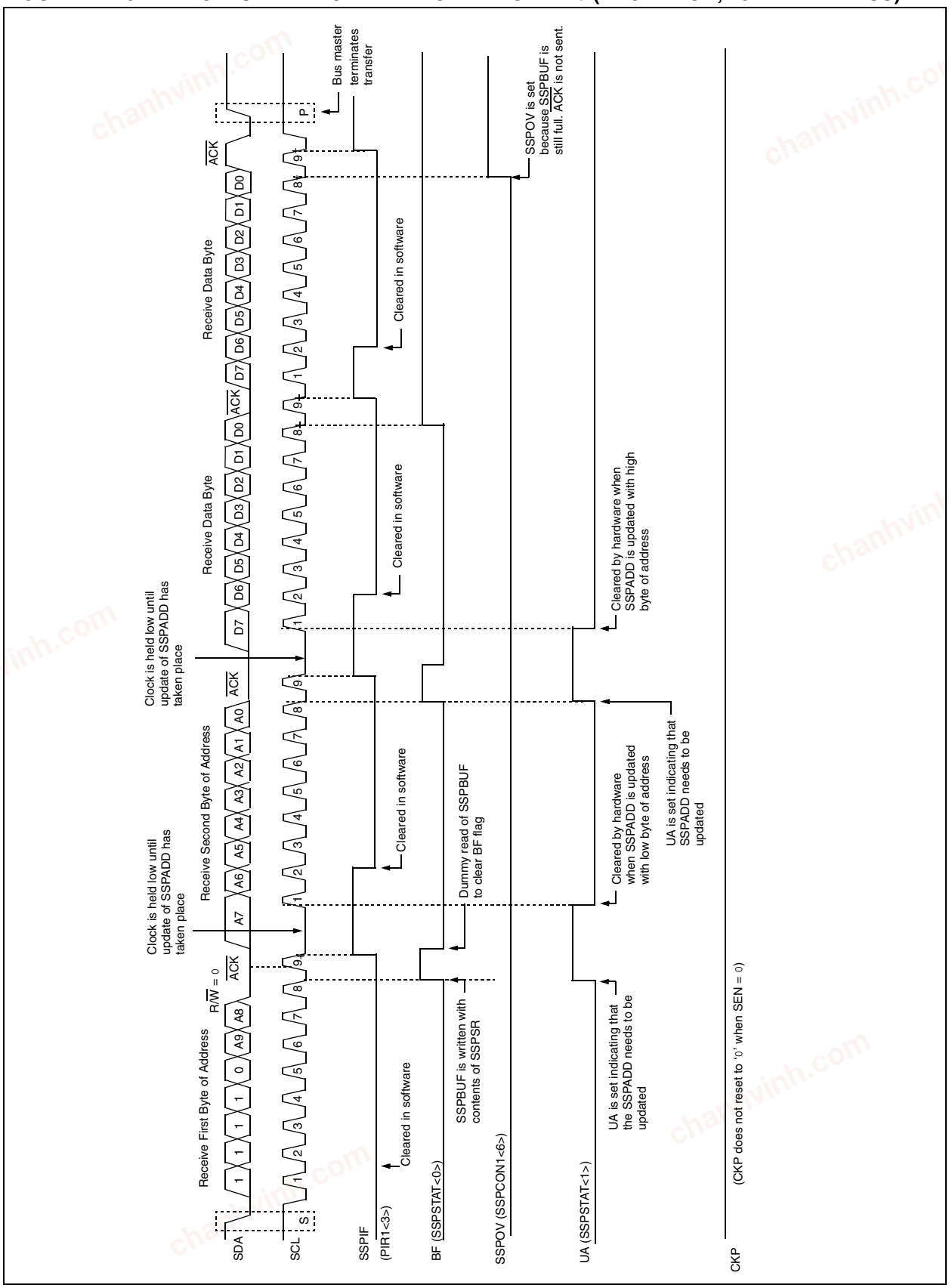
**FIGURE 17-9: I²C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**



chanhvinh.com



**FIGURE 17-10: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)**



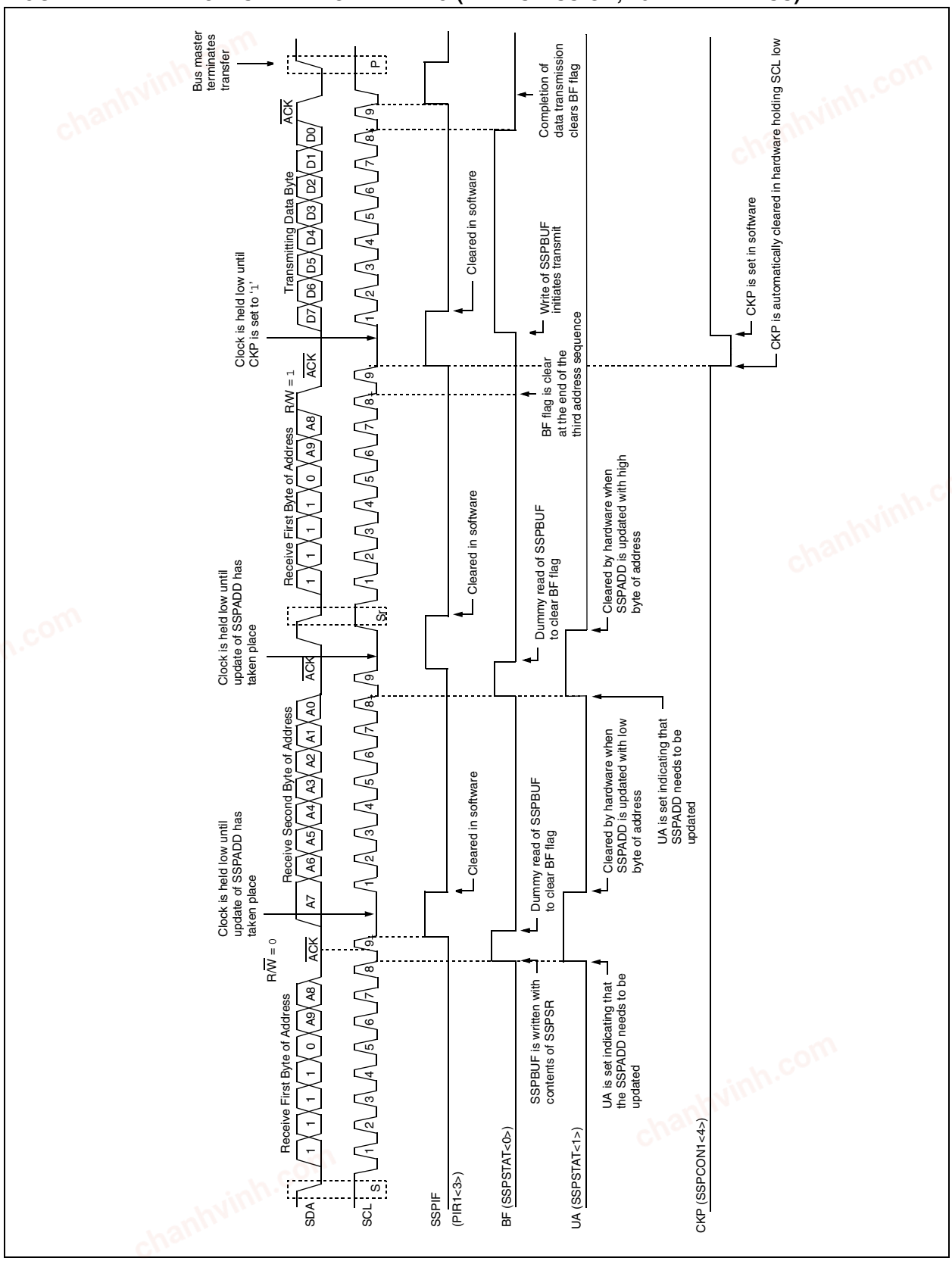



© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 17-11: I²C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**



chanhvinh.com



### 17.4.4 CLOCK STRETCHING

Both 7 and 10-bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

#### 17.4.4.1 Clock Stretching for 7-bit Slave Receive Mode (SEN = 1)

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{ACK}$ sequence, if the BF bit is set, the CKP bit in the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring.

| | |
|---|---|
| **Note 1:** | If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur. |
| **2:** | The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition. |

#### 17.4.4.2 Clock Stretching for 10-bit Slave Receive Mode (SEN = 1)

In 10-bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{W}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

| | |
|---|---|
| **Note:** | If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence. |

#### 17.4.4.3 Clock Stretching for 7-bit Slave Transmit Mode

7-bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 17-9).

| | |
|---|---|
| **Note 1:** | If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur. |
| **2:** | The CKP bit can be set in software regardless of the state of the BF bit. |

#### 17.4.4.4 Clock Stretching for 10-bit Slave Transmit Mode

In 10-bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high-order bits of the 10-bit address and the R/$\overline{W}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-bit Slave Transmit mode (see Figure 17-11).


© 2006 Microchip Technology Inc.

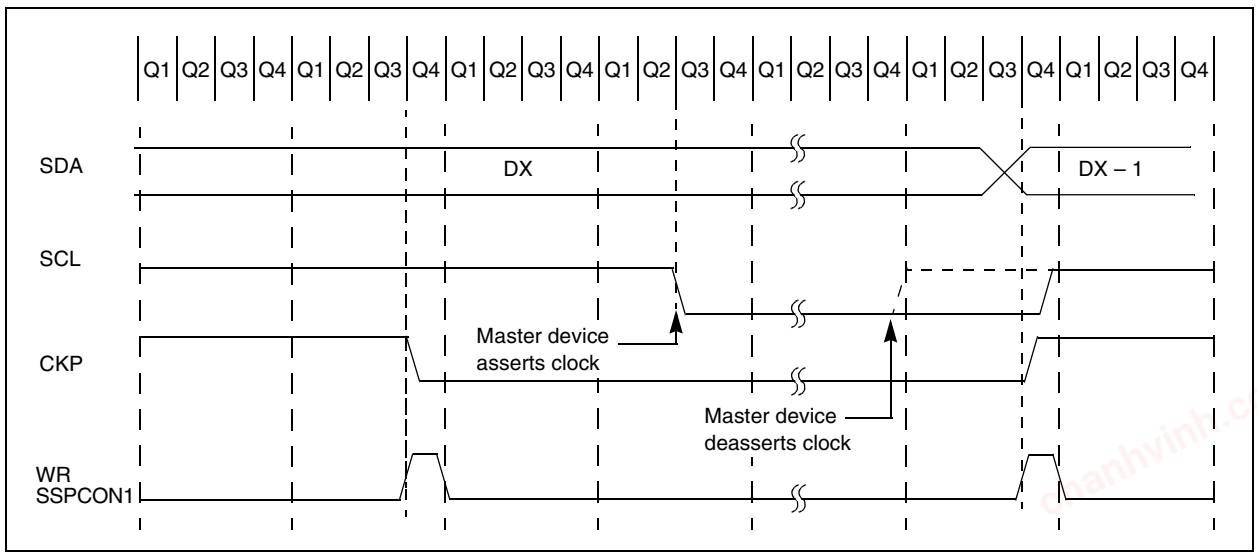
chanhvinh.com



#### 17.4.4.5 Clock Synchronization and the CKP bit

If a user clears the CKP bit, the SCL output is forced to '0'. Setting the CKP bit will not assert the SCL output low until the SCL output is already sampled low. If the user attempts to drive SCL low, the CKP bit will not assert the SCL line until an external $I^2C$ master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the $I^2C$ bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 17-12).

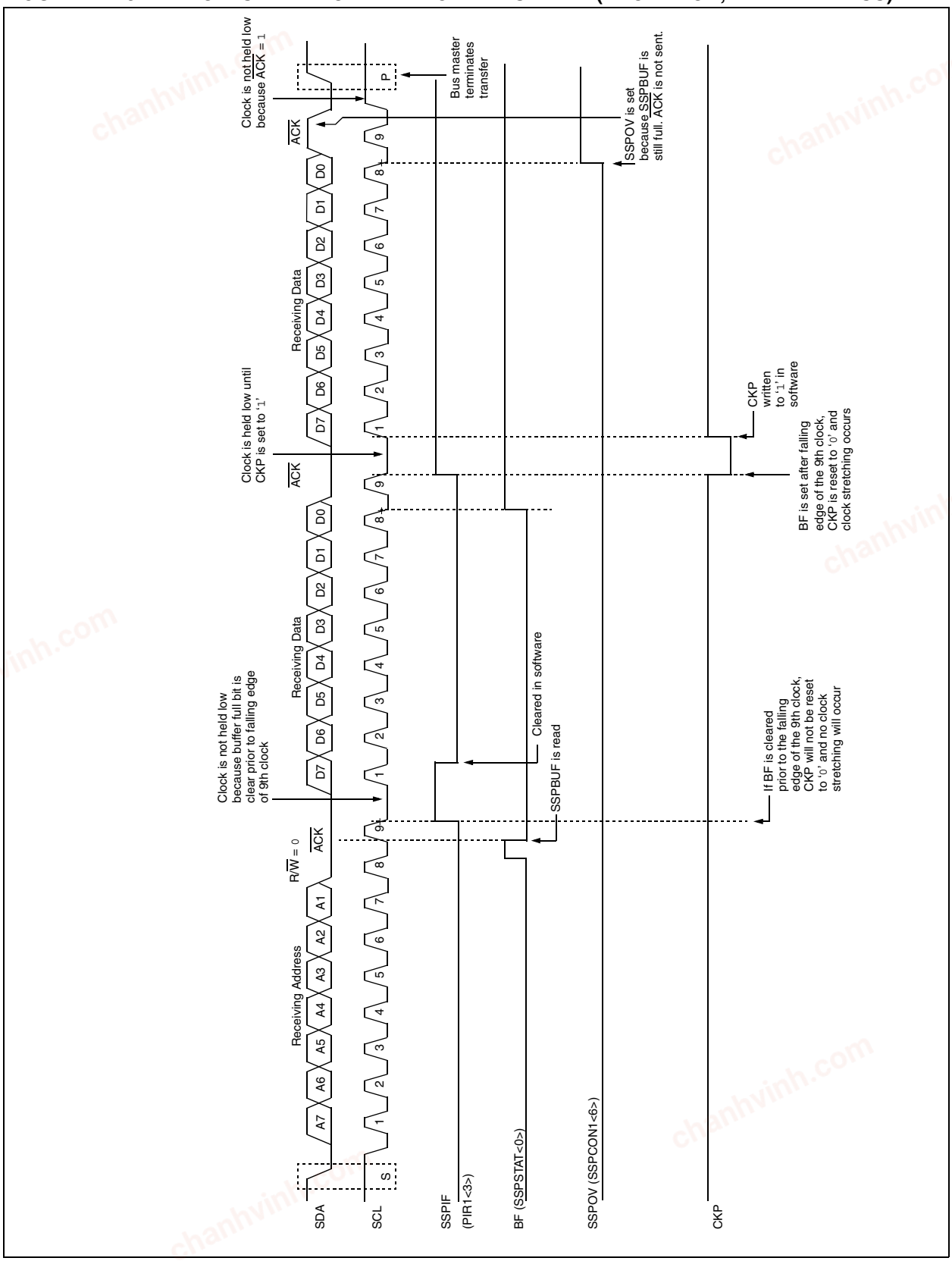
**FIGURE 17-12: CLOCK SYNCHRONIZATION TIMING**



chanhvinh.com



**FIGURE 17-13: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)**



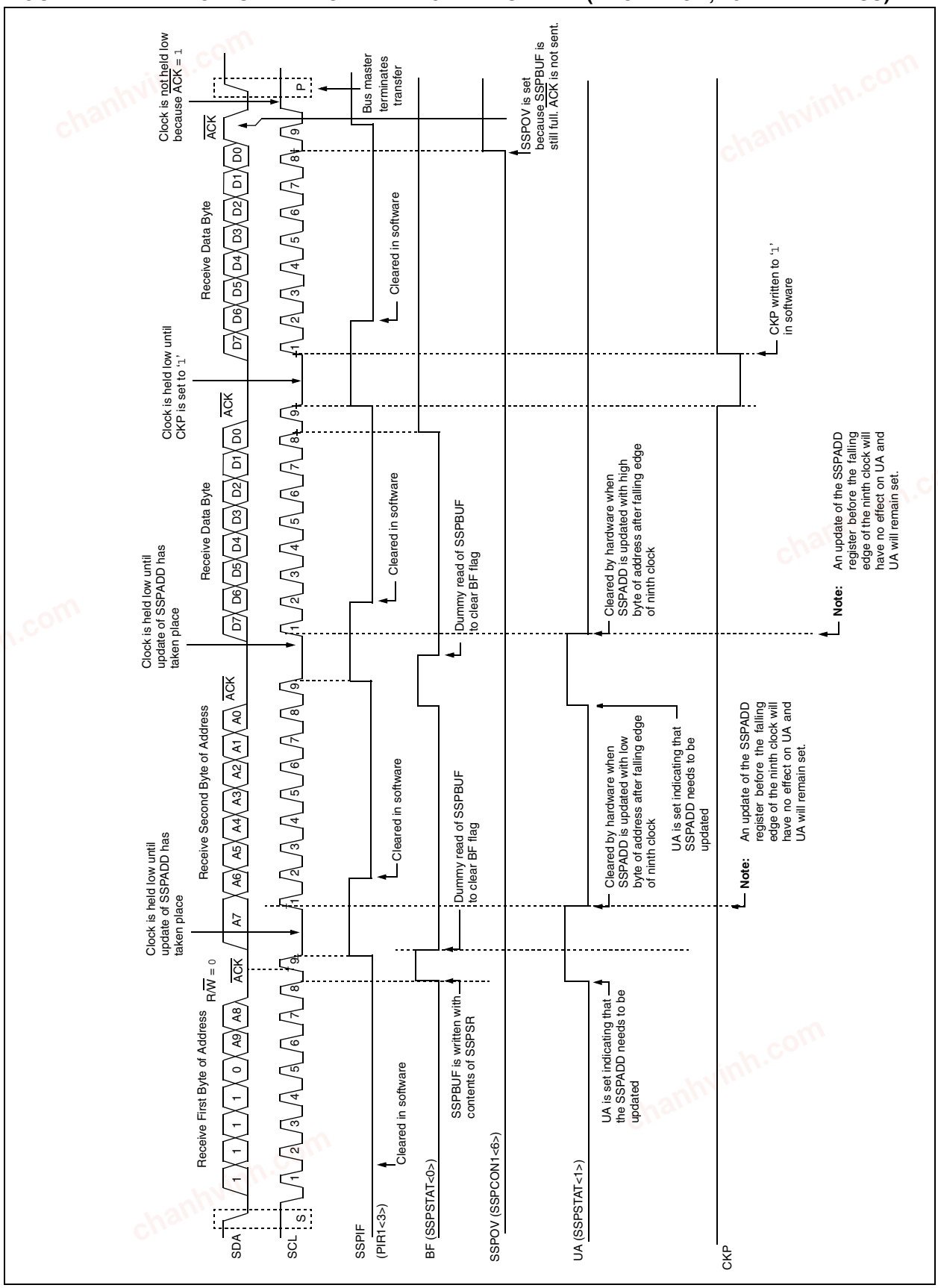


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 17-14: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)**



chanhvinh.com



### 17.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the $I^2C$ bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an Acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the $I^2C$ protocol. It consists of all '0's with R/$\overline{W}$ = 0.

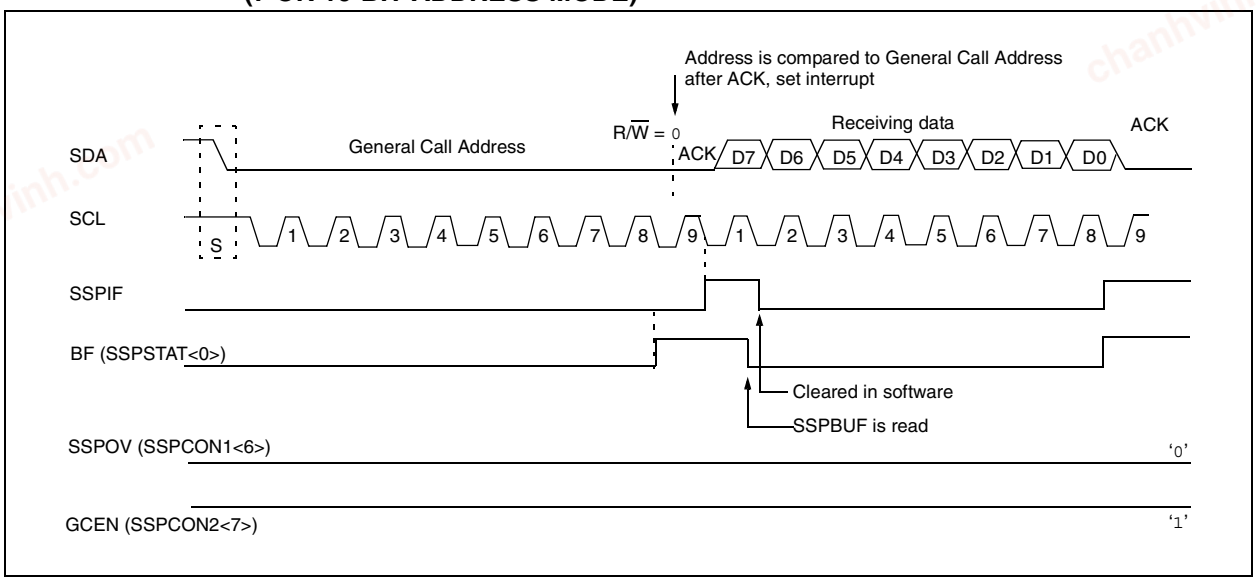
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> set). Following a Start bit detect, 8 bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eighth bit) and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device specific or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when the GCEN bit is set, 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 17-15).

**FIGURE 17-15: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESS MODE)**





© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the $I^2C$ bus may be taken when the P bit is set or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all $I^2C$ bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

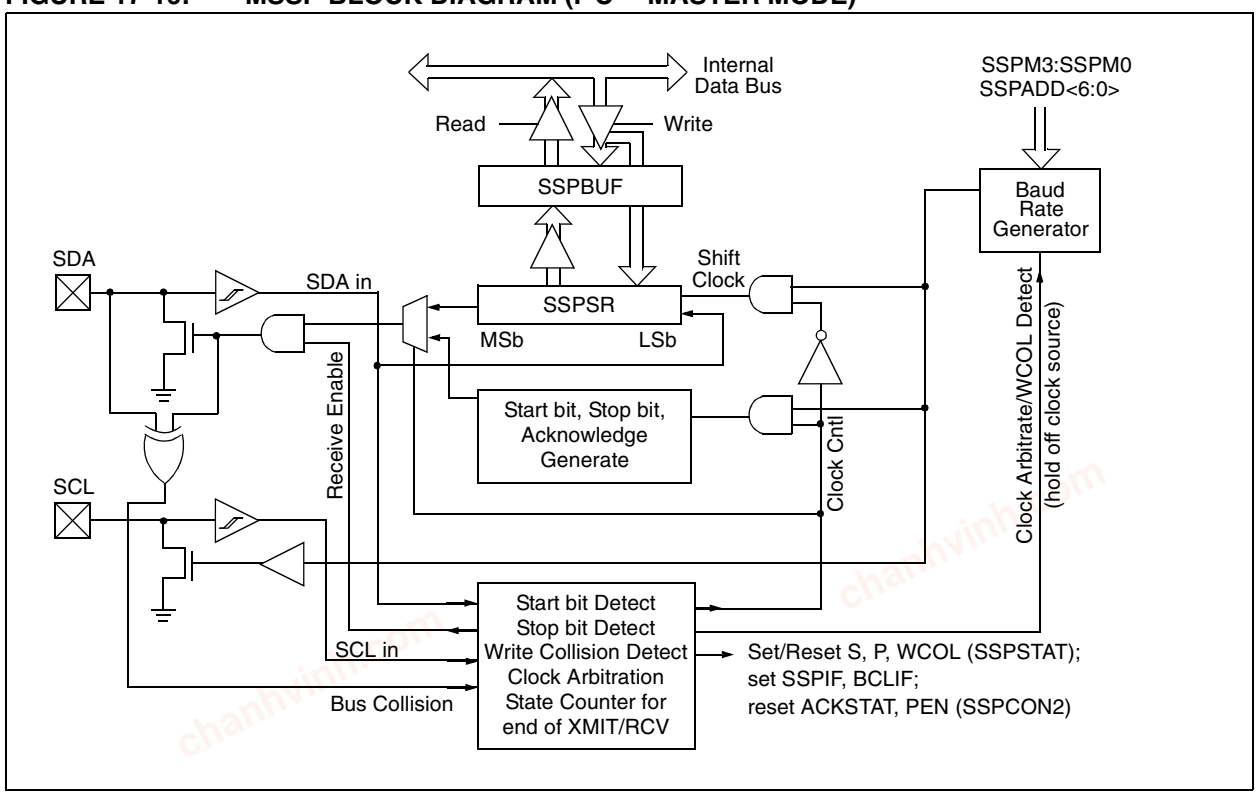
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the $I^2C$ port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

**Note:** The MSSP module, when configured in $I^2C$ Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

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

**FIGURE 17-16: MSSP BLOCK DIAGRAM ($I^2C$™ MASTER MODE)**



chanhvinh.com



#### 17.4.6.1 I²C Master Mode Operation

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

In Master Transmitter mode, serial data is output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate receive bit. Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8 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 used to set the SCL clock frequency for either 100 kHz, 400 kHz or 1 MHz I²C operation. See **Section 17.4.7 "Baud Rate Generator"** for more details.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start Enable bit, SEN (SSPCON2<0>).
2. SSPIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPBUF with the slave address to transmit.
4. Address is shifted out the SDA pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a Stop condition by setting the Stop Enable bit PEN (SSPCON2<2>).
12. Interrupt is generated once the Stop condition is complete.


© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.4.7 BAUD RATE GENERATOR

In $I^2C$ Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPADD register (Figure 17-17). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In $I^2C$ Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by $\overline{ACK}$), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 17-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

**FIGURE 17-17: BAUD RATE GENERATOR BLOCK DIAGRAM**

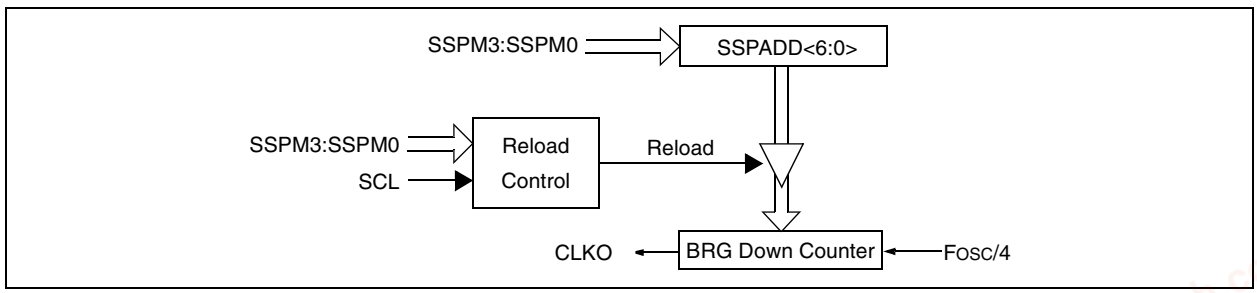



**TABLE 17-3: $I^2C$™ CLOCK RATE w/BRG**

| FOSC | FCY | FCY * 2 | BRG Value | FSCL (2 Rollovers of BRG) |
|---|---|---|---|---|
| 40 MHz | 10 MHz | 20 MHz | 18h | 400 kHz(1) |
| 40 MHz | 10 MHz | 20 MHz | 1Fh | 312.5 kHz |
| 40 MHz | 10 MHz | 20 MHz | 63h | 100 kHz |
| 16 MHz | 4 MHz | 8 MHz | 09h | 400 kHz(1) |
| 16 MHz | 4 MHz | 8 MHz | 0Ch | 308 kHz |
| 16 MHz | 4 MHz | 8 MHz | 27h | 100 kHz |
| 4 MHz | 1 MHz | 2 MHz | 02h | 333 kHz(1) |
| 4 MHz | 1 MHz | 2 MHz | 09h | 100kHz |
| 4 MHz | 1 MHz | 2 MHz | 00h | 1 MHz(1) |

**Note 1:** The $I^2C$™ interface does not conform to the 400 kHz $I^2C$ specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

© 2006 Microchip Technology Inc.

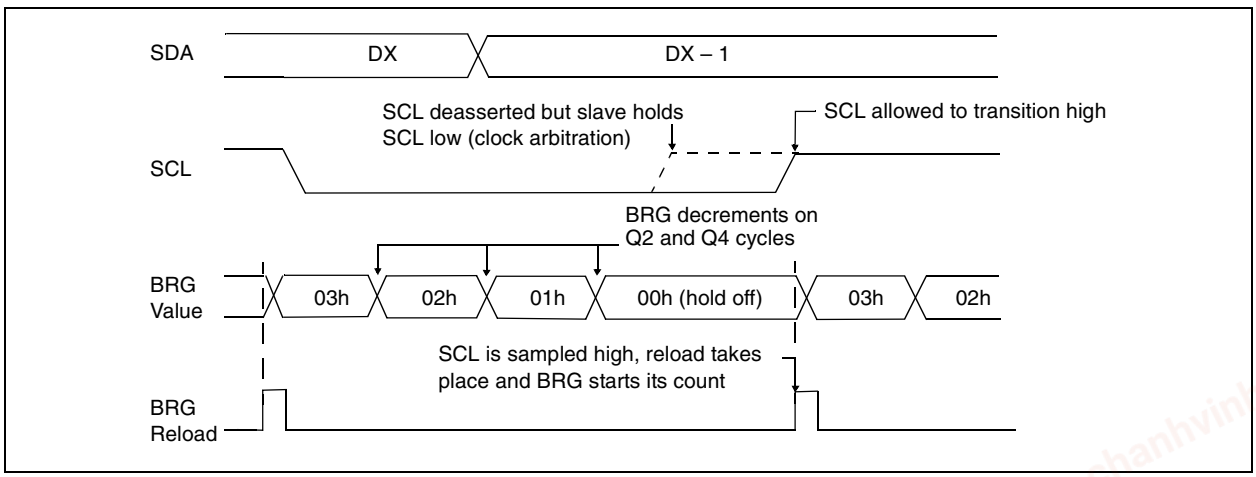
chanhvinh.com



#### 17.4.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts 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 17-18).

**FIGURE 17-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**





© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.4.8 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start condition enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded 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<3>) 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<0>) will be automatically cleared by hardware, the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

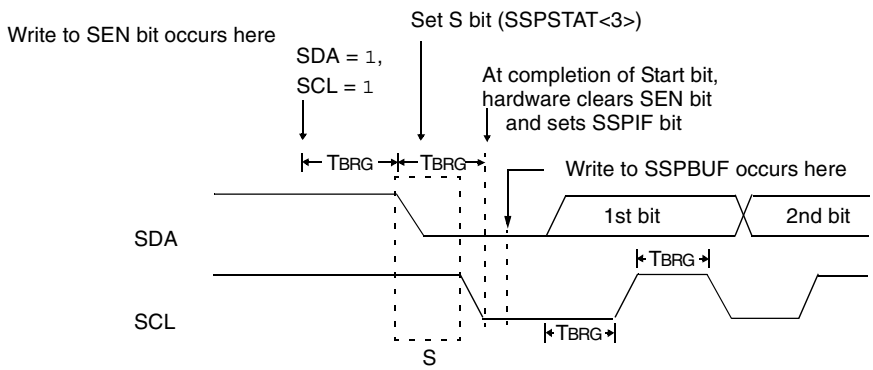
| | |
|---|---|
| **Note:** | If, at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs; the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state. |

#### 17.4.8.1 WCOL Status Flag

If the user writes the SSPBUF when a Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

| | |
|---|---|
| **Note:** | Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the Start condition is complete. |

**FIGURE 17-19: FIRST START BIT TIMING**



chanhvinh.com



### 17.4.9 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) 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 deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded 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<1>) 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<3>) 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'.

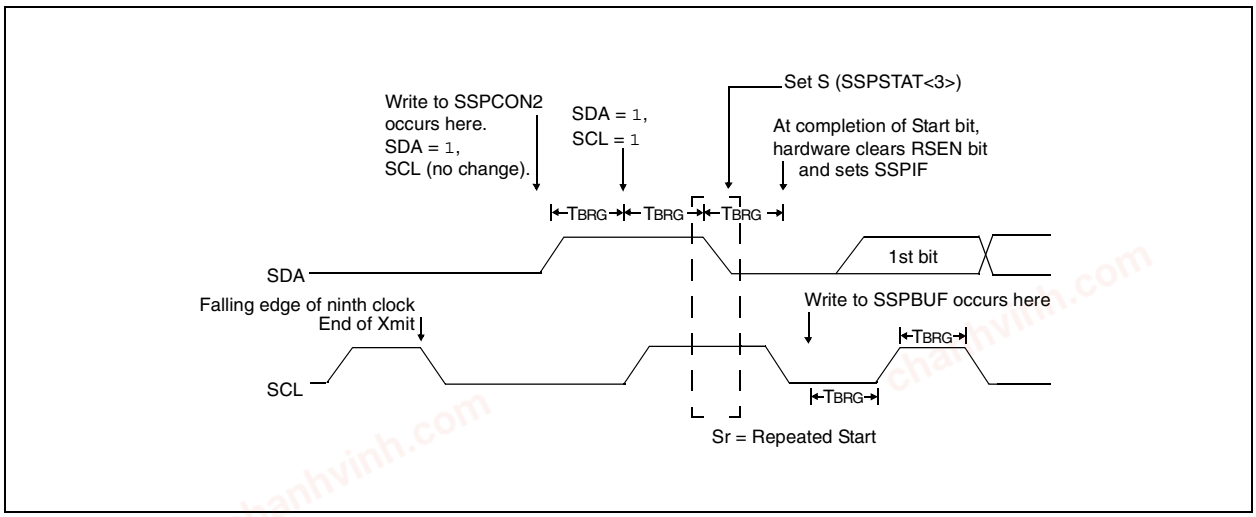
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

#### 17.4.9.1 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 17-20: REPEATED START CONDITION WAVEFORM**





© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.4.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit BF and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification parameter #106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification parameter #107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time, if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ is written into the 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 17-21).

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/$\overline{W}$ bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF bit is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

#### 17.4.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT<0>) is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

#### 17.4.10.2 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.

#### 17.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call) or when the slave has properly received its data.

### 17.4.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPCON2<3>).

| **Note:** | The RCEN bit should be set after the $\overline{ACK}$ sequence is complete or the RCEN bit will be disregarded. |
|---|---|

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>).

#### 17.4.11.1 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.

#### 17.4.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

#### 17.4.11.3 WCOL Status Flag

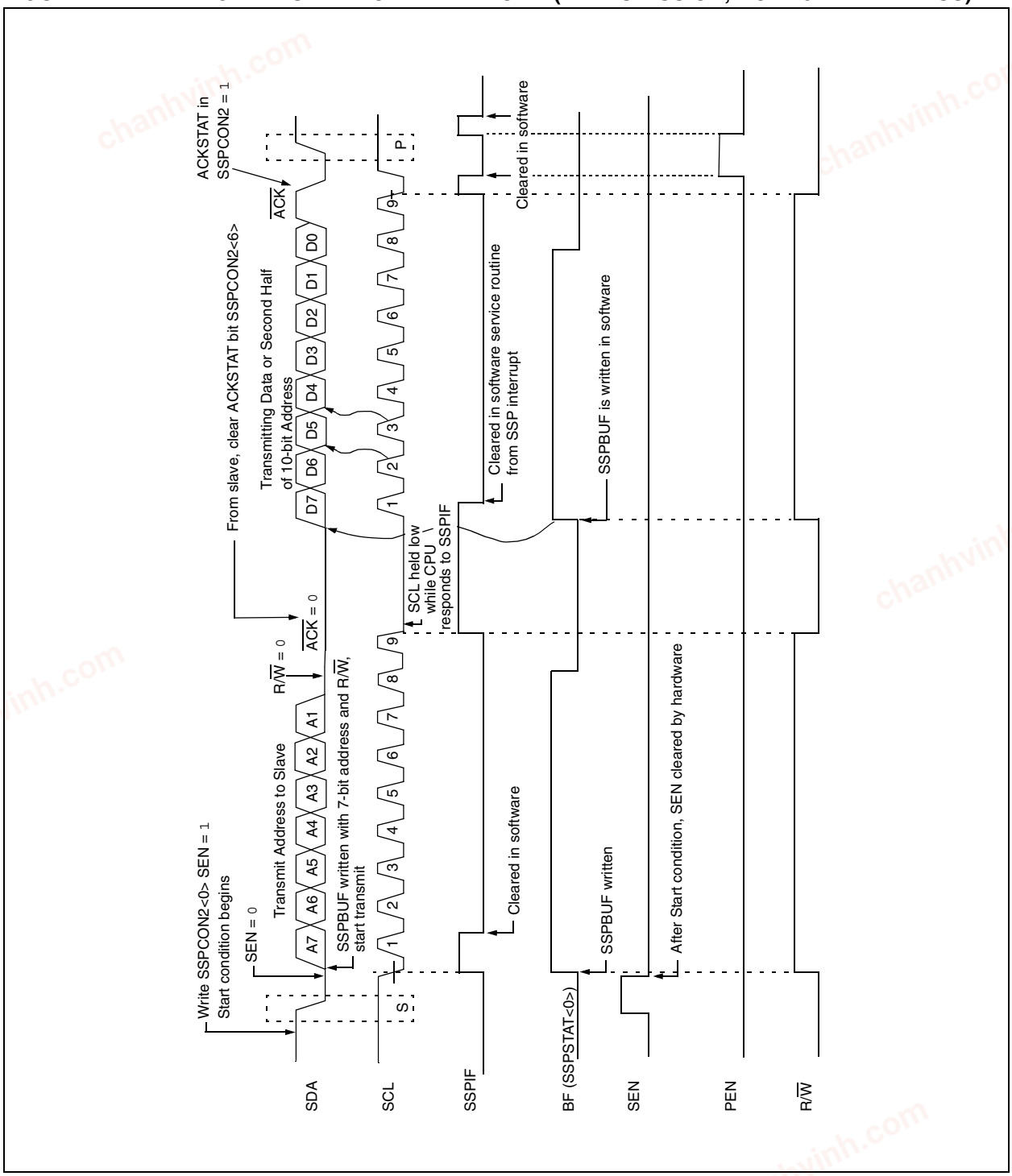
If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 17-21: I²C™ MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**



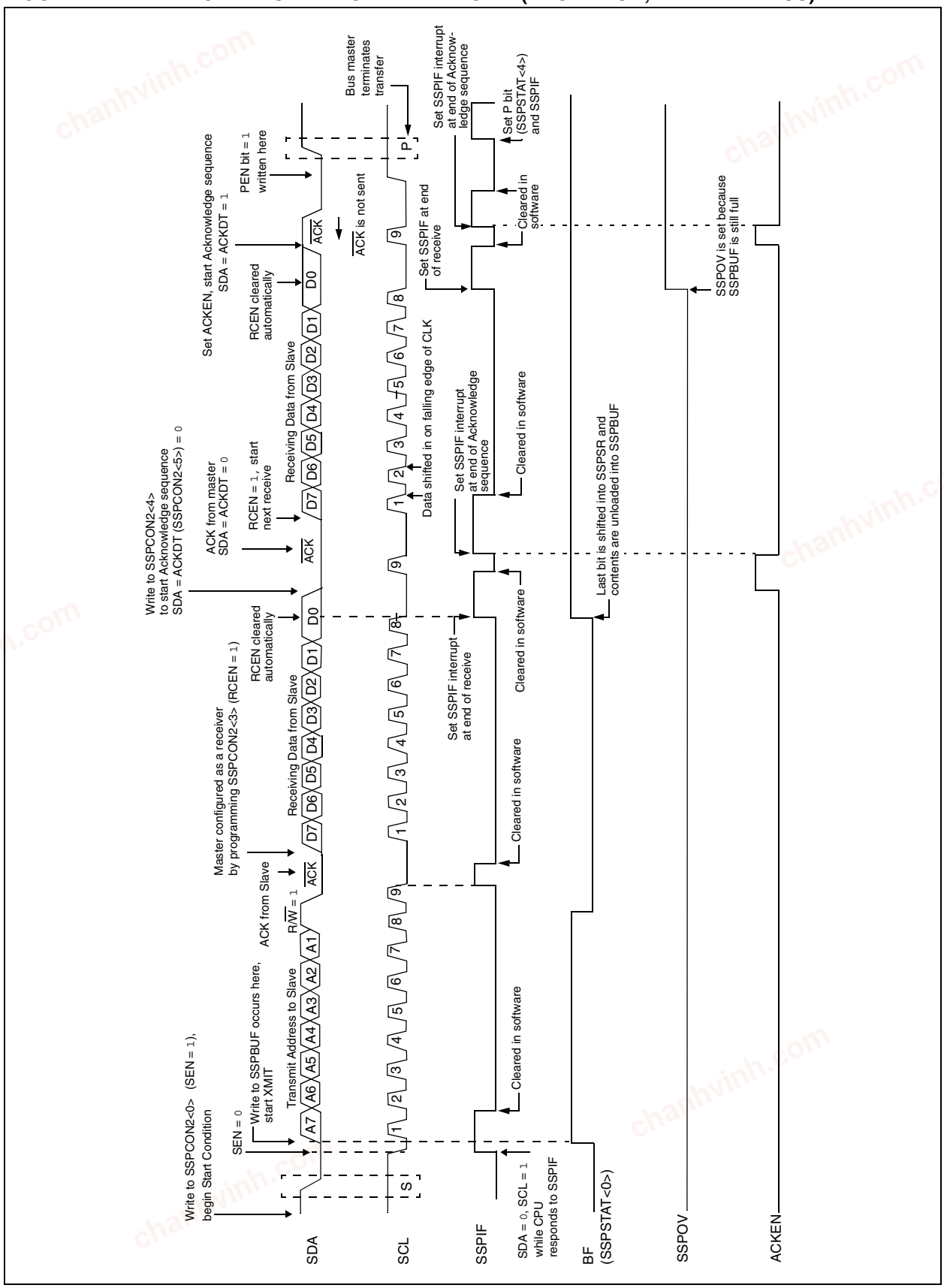



© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 17-22: I²C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)**



chanhvinh.com



### 17.4.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>). When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode (Figure 17-23).

#### 17.4.12.1 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).

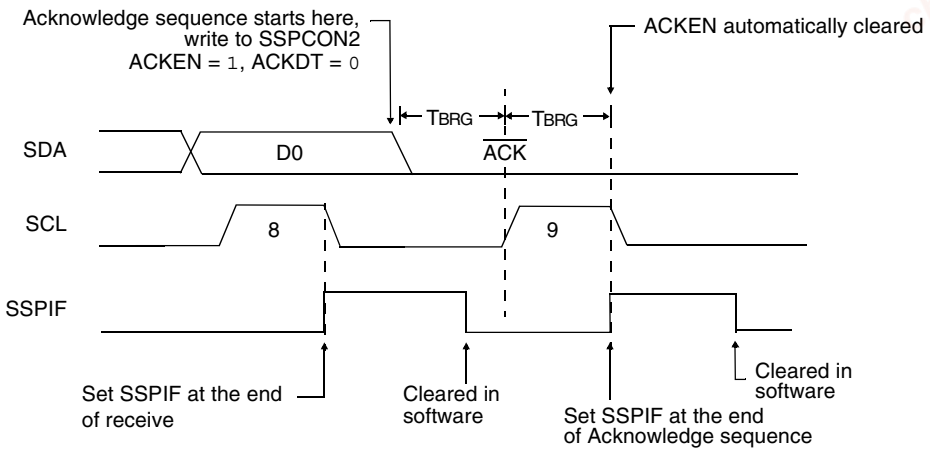
### 17.4.13 STOP CONDITION TIMING

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<2>). At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to 0. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 17-24).

#### 17.4.13.1 WCOL Status Flag

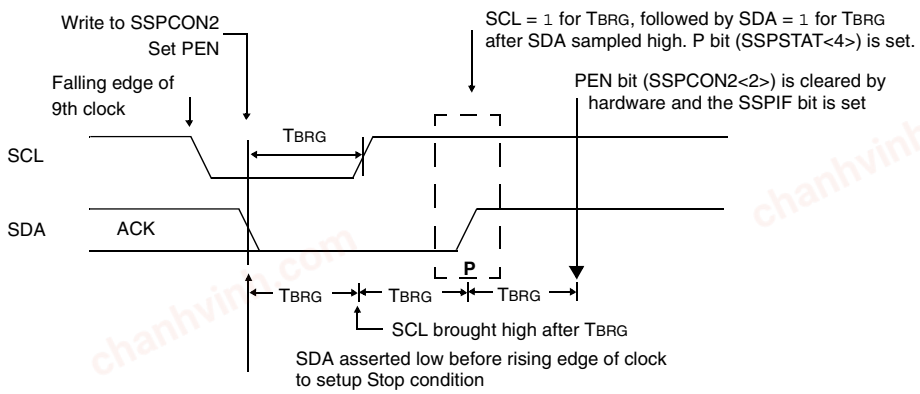
If the user writes the SSPBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 17-23: ACKNOWLEDGE SEQUENCE WAVEFORM**




**Note:** TBRG = one Baud Rate Generator period.

**FIGURE 17-24: STOP CONDITION RECEIVE OR TRANSMIT MODE**




**Note:** TBRG = one Baud Rate Generator period.


© 2006 Microchip Technology Inc.

chanhvinh.com



### 17.4.14 SLEEP OPERATION

While in Sleep mode, the $I^2C$ module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

### 17.4.15 EFFECT OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 17.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the $I^2C$ bus may be taken when the P bit (SSPSTAT<4>) 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.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

### 17.4.17 MULTI -MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = 0, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag BCLIF and reset the $I^2C$ port to its Idle state (Figure 17-25).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the $I^2C$ bus is free, the user can resume communication by asserting a Start condition.

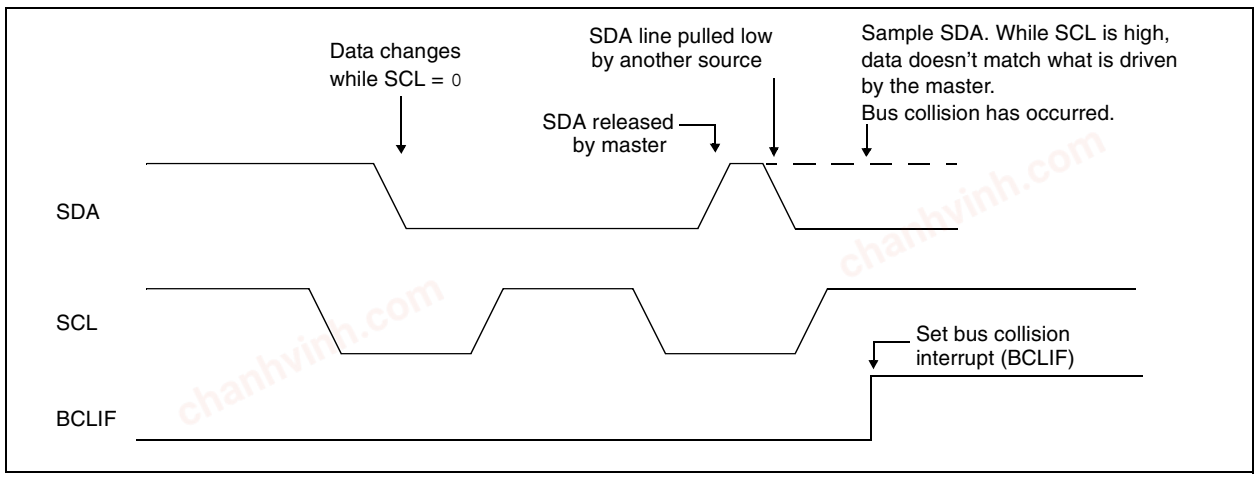
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the $I^2C$ bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the $I^2C$ bus can be taken when the P bit is set in the SSPSTAT register or the bus is Idle and the S and P bits are cleared.

**FIGURE 17-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



chanhvinh.com



#### 17.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 17-26).

b) SCL is sampled low before SDA is asserted low (Figure 17-27).

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

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

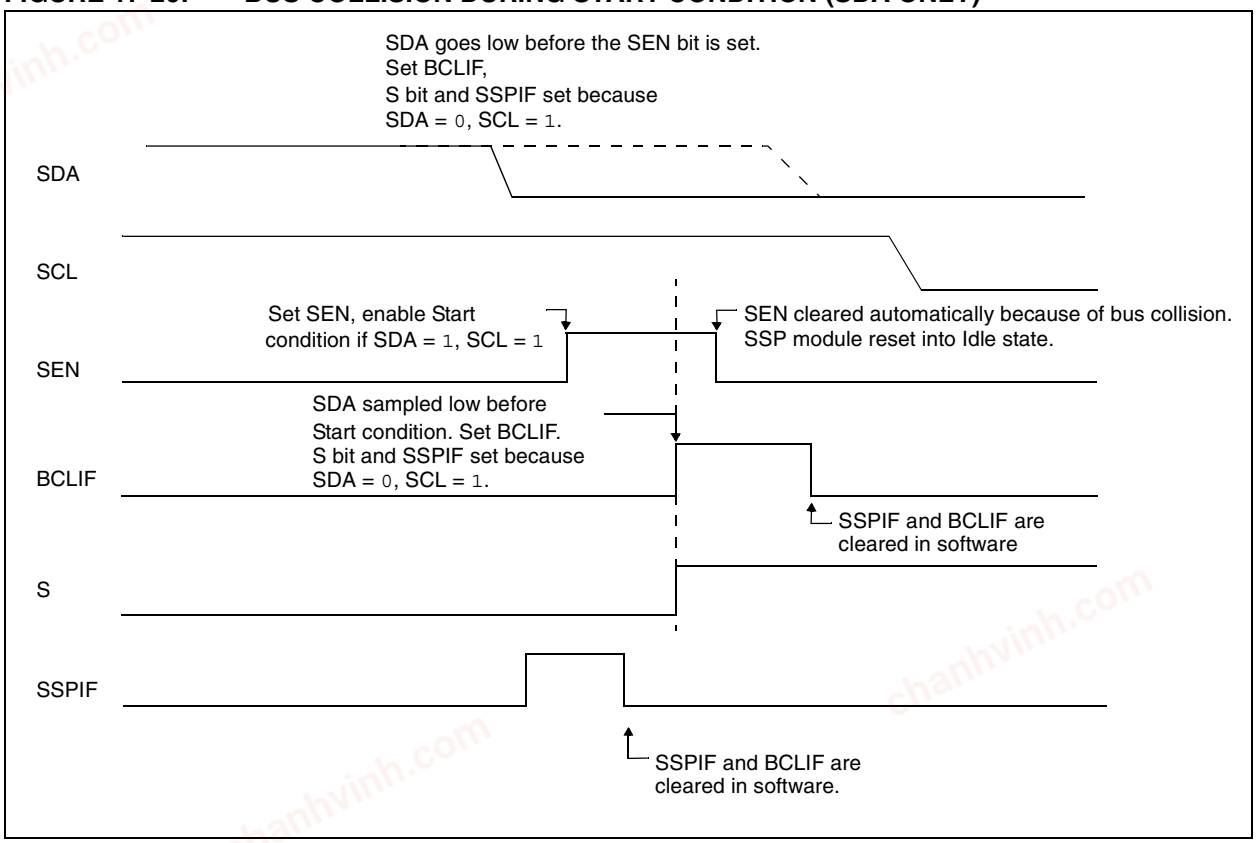
- the Start condition is aborted,
- the BCLIF flag is set and
- the MSSP module is reset to its Idle state (Figure 17-26).

The Start condition begins with the SDA and SCL pins deasserted. 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 17-28). 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 pins are 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 17-26: BUS COLLISION DURING START CONDITION (SDA ONLY)**





© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 17-27: BUS COLLISION DURING START CONDITION (SCL = 0)**

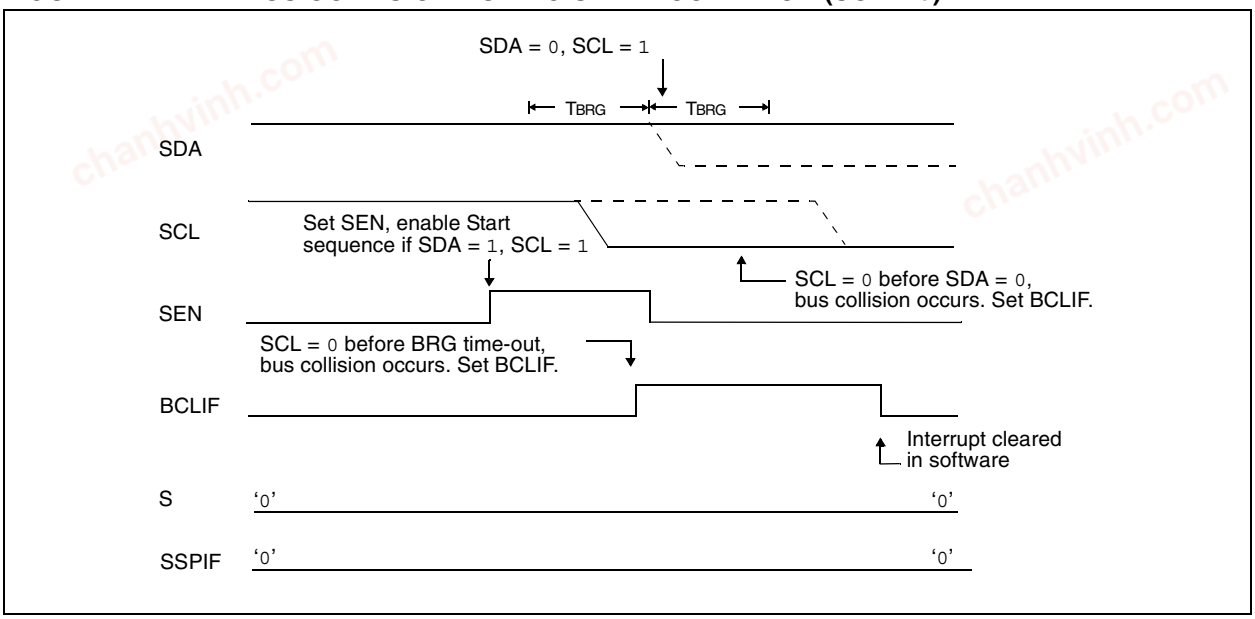
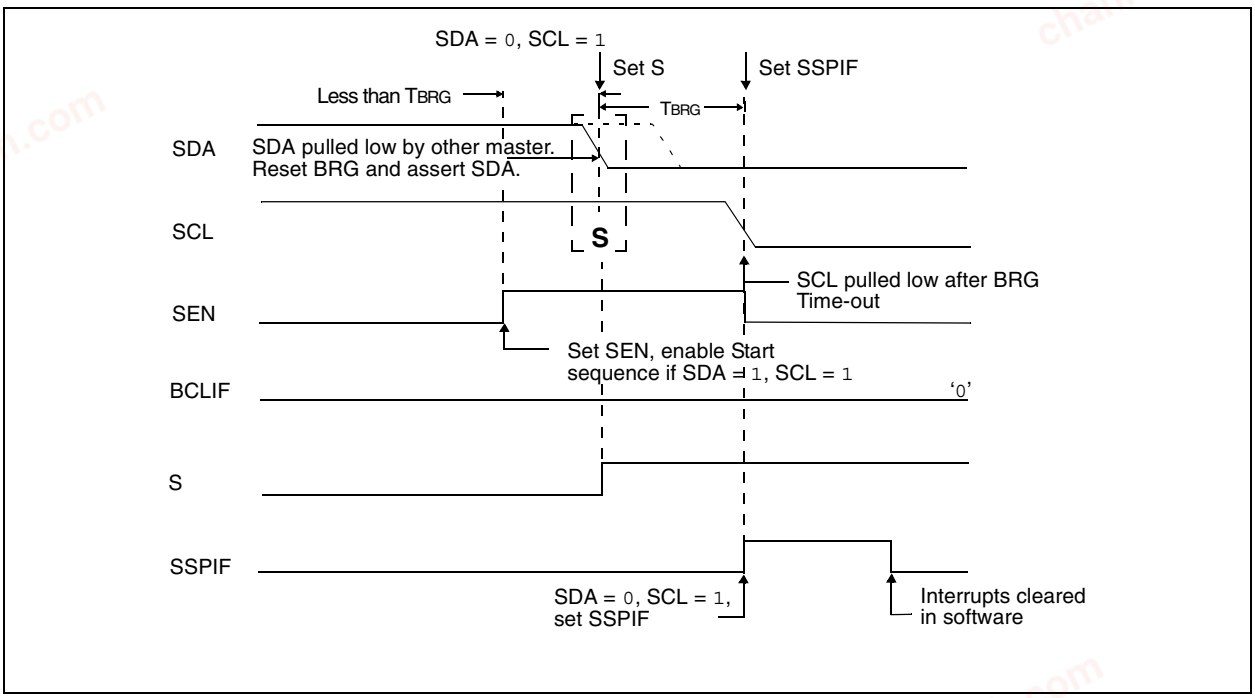



**FIGURE 17-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**



chanhvinh.com



### 17.4.17.2 Bus Collision During a Repeated Start Condition

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

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

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts 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 deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 17-29). 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 17-30).

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 17-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

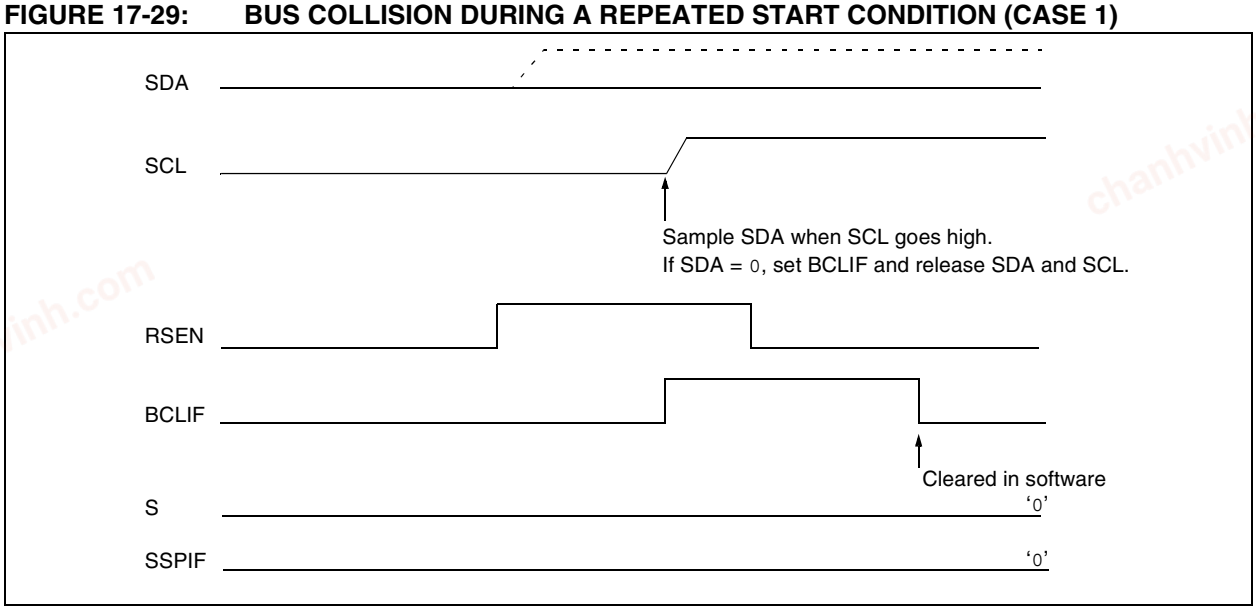
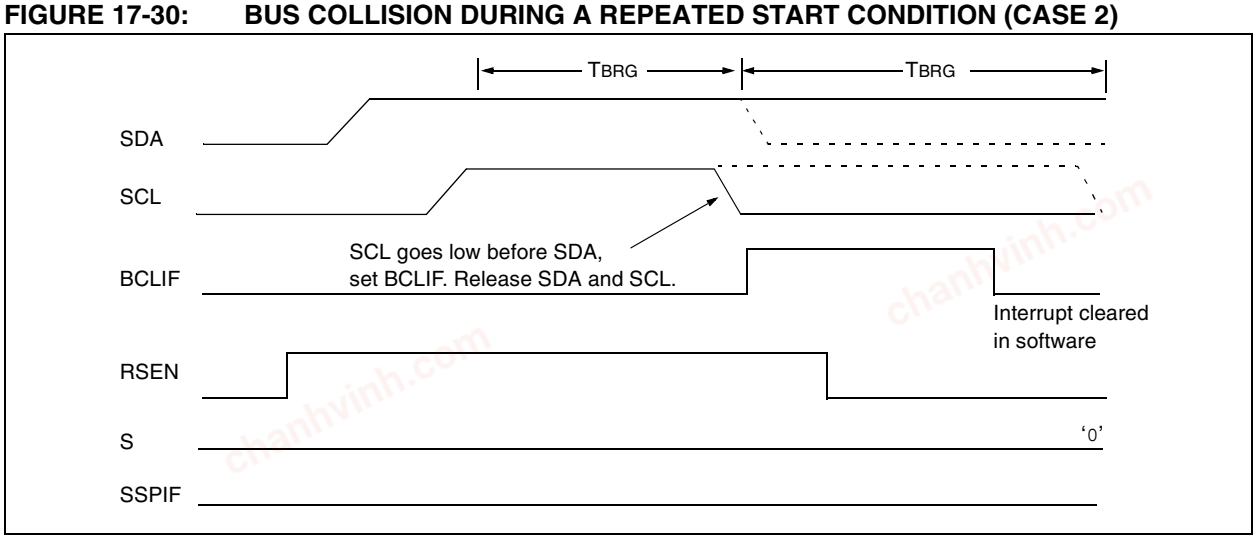



**FIGURE 17-30: BUS COLLISION DURING A REPEATED START CONDITION (CASE 2)**





© 2006 Microchip Technology Inc.

chanhvinh.com



#### 17.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

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

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD<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 17-31). 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 17-32).

**FIGURE 17-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

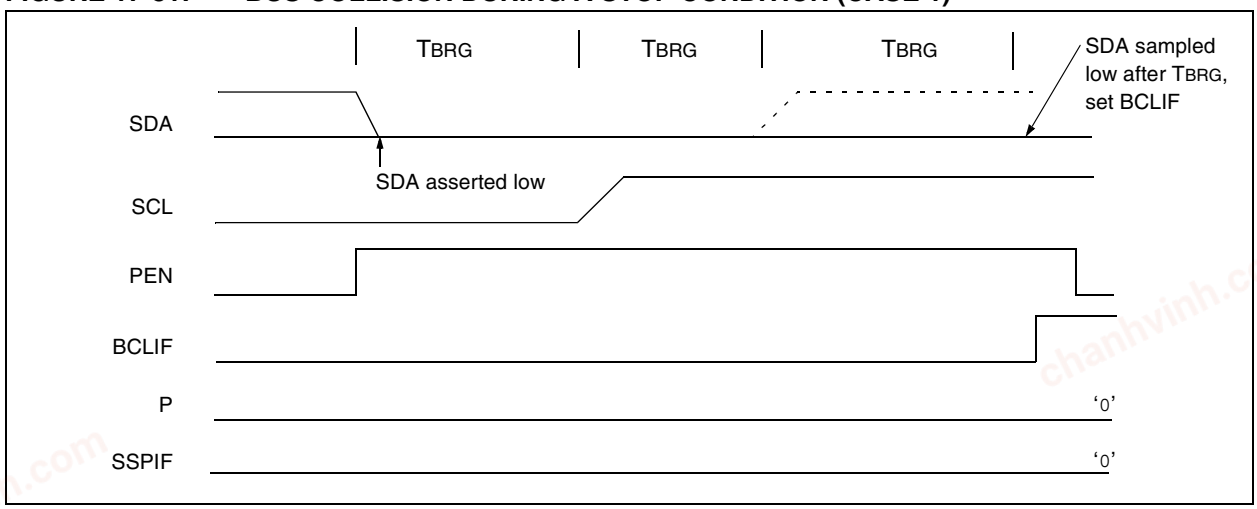
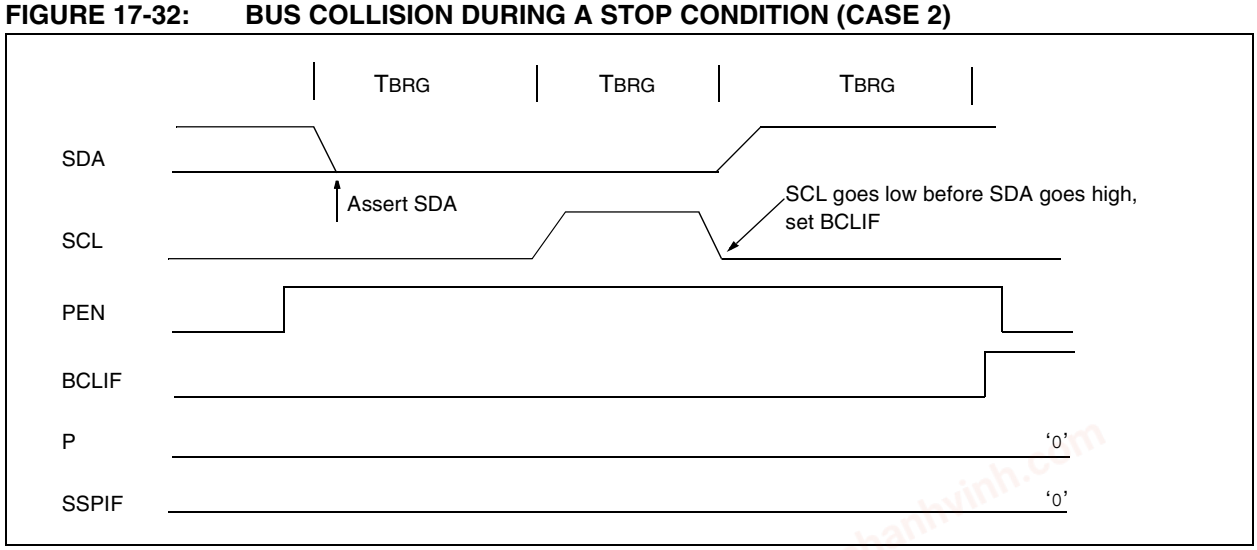



**FIGURE 17-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)**



chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 18.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the three serial I/O modules incorporated into PIC18FXX8 devices. (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 18-1 shows the Transmit Status and Control register (TXSTA) and Register 18-2 shows the Receive Status and Control register (RCSTA).

**REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R-1 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D |
| bit 7 | | | | | | | bit 0 |

bit 7 **CSRC:** Clock Source Select bit

Asynchronous mode:
Don't care.

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

**Note:** SREN/CREN overrides TXEN in Sync mode.

bit 4 **SYNC:** USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

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

bit 2 **BRGH**: High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode.

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data
Can be address/data bit or a parity bit.

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

chanhvinh.com



**REGISTER 18-2: RCSTA: RECEIVE STATUS AND CONTROL 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, enables 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



## 18.1 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 18-1 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 18-1. From this, the error in baud rate can be determined.

Example 18-1 shows the calculation of the baud rate error for the following conditions:

FOSC = 16 MHz
Desired Baud Rate = 9600
BRGH = 0
SYNC = 0

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the FOSC/(16(X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

### 18.1.1 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 18-1: CALCULATING BAUD RATE ERROR**

Desired Baud Rate = FOSC/(64 (X + 1))

Solving for X:

X = ((FOSC/Desired Baud Rate)/64) – 1
X = ((16000000/9600)/64) – 1
X = [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615

$$\text{Error} = \frac{\text{(Calculated Baud Rate – Desired Baud Rate)}}{\text{Desired Baud Rate}}$$
= (9615 – 9600)/9600
= 0.16%

**TABLE 18-1: BAUD RATE FORMULA**

| SYNC | BRGH = 0 (Low Speed) | BRGH = 1 (High Speed) |
|---|---|---|
| 0 | (Asynchronous) Baud Rate = FOSC/(64 (X + 1)) | Baud Rate = FOSC/(16 (X + 1)) |
| 1 | (Synchronous) Baud Rate = FOSC/(4 (X + 1)) | NA |

**Legend:** X = value in SPBRG (0 to 255)

**TABLE 18-2: 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 000u |
| SPBRG | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

chanhvinh.com



**TABLE 18-3: BAUD RATES FOR SYNCHRONOUS MODE**

| BAUD RATE (Kbps) | Fosc = 40 MHz | | SPBRG value (decimal) | 33 MHz | | SPBRG value (decimal) | 25 MHz | | SPBRG value (decimal) | 20 MHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 | | SPBRG value (decimal) | 10 MHz | | SPBRG value (decimal) | 7.15909 MHz | | SPBRG value (decimal) | 5.0688 MHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 | | SPBRG value (decimal) | 3.579545 MHz | | SPBRG value (decimal) | 1 MHz | | SPBRG value (decimal) | 32.768 kHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 18-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)**

| BAUD RATE (Kbps) | Fosc = 40 MHz | | SPBRG value (decimal) | 33 MHz | | SPBRG value (decimal) | 25 MHz | | SPBRG value (decimal) | 20 MHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 | | SPBRG value (decimal) | 10 MHz | | SPBRG value (decimal) | 7.15909 MHz | | SPBRG value (decimal) | 5.0688 MHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 | | SPBRG value (decimal) | 3.579545 MHz | | SPBRG value (decimal) | 1 MHz | | SPBRG value (decimal) | 32.768 kHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 18-5: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)**

| BAUD RATE (Kbps) | Fosc = 40 MHz | | SPBRG value (decimal) | 33 MHz | | SPBRG value (decimal) | 25 MHz | | SPBRG value (decimal) | 20 MHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 | | SPBRG value (decimal) | 10 MHz | | SPBRG value (decimal) | 7.15909 MHz | | SPBRG value (decimal) | 5.0688 MHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 | | SPBRG value (decimal) | 3.579545 MHz | | SPBRG value (decimal) | 1 MHz | | SPBRG value (decimal) | 32.768 kHz | | SPBRG value (decimal) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | | KBAUD | % ERROR | |
| 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



## 18.2 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.

### 18.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 18-1. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The TSR register obtains its data from the Read/Write Transmit Buffer register (TXREG). The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR1 register) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1 register). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit, TXIF, indicated the status of the TXREG register, another bit, TRMT (TXSTA register), shows the status of the TSR register. Status bit TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

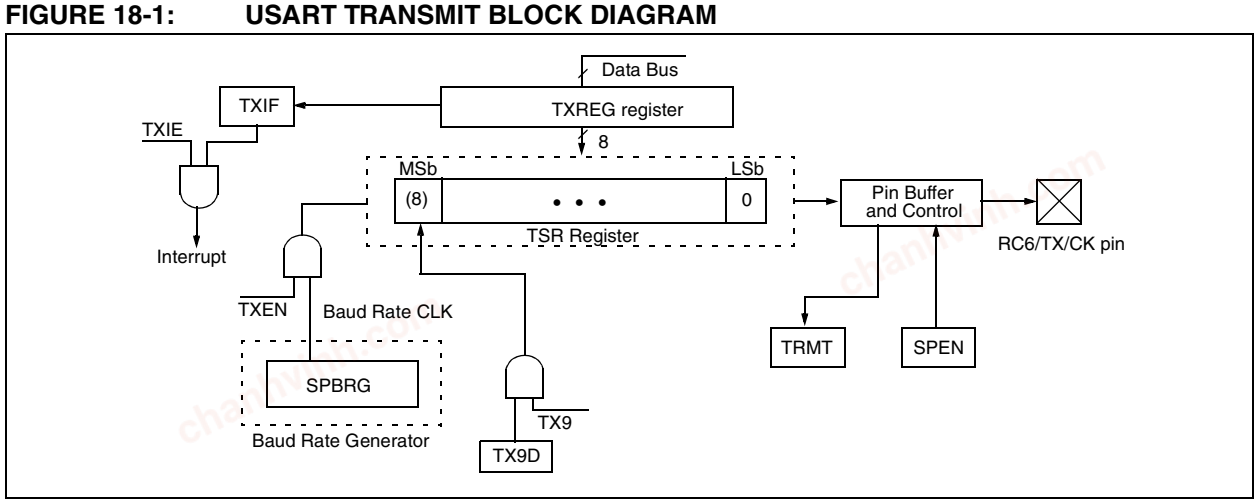
| | |
|---|---|
| **Note 1:** | The TSR register is not mapped in data memory, so it is not available to the user. |
| **2:** | Flag bit TXIF is set when enable bit TXEN is set. |

Steps to follow when setting up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (**Section 18.1 "USART Baud Rate Generator (BRG)"**).
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).

| | |
|---|---|
| **Note:** | TXIF is not cleared immediately upon loading data into the transmit buffer TXREG. The flag bit becomes valid in the second instruction cycle following the load instruction. |

**FIGURE 18-1: USART TRANSMIT BLOCK DIAGRAM**



chanhvinh.com



**FIGURE 18-2: ASYNCHRONOUS TRANSMISSION**

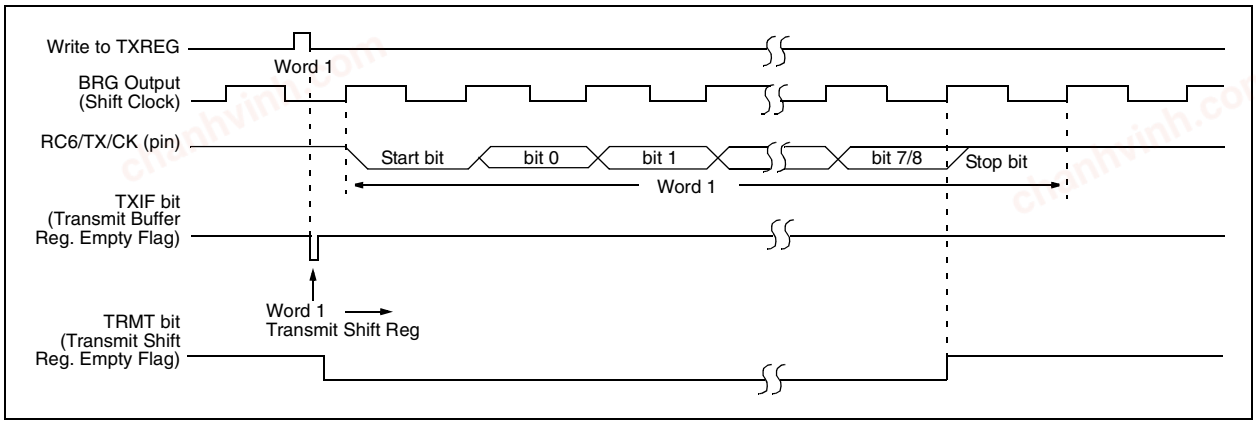
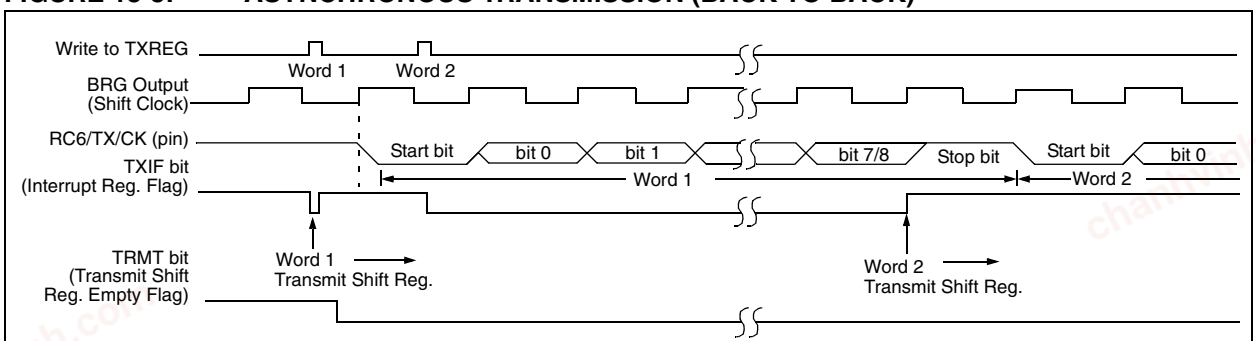



**FIGURE 18-3: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)**




**Note:** This timing diagram shows two consecutive transmissions.

**TABLE 18-6: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000u |
| TXREG | USART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 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.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.


© 2006 Microchip Technology Inc.

chanhvinh.com



### 18.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 18-4. 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:

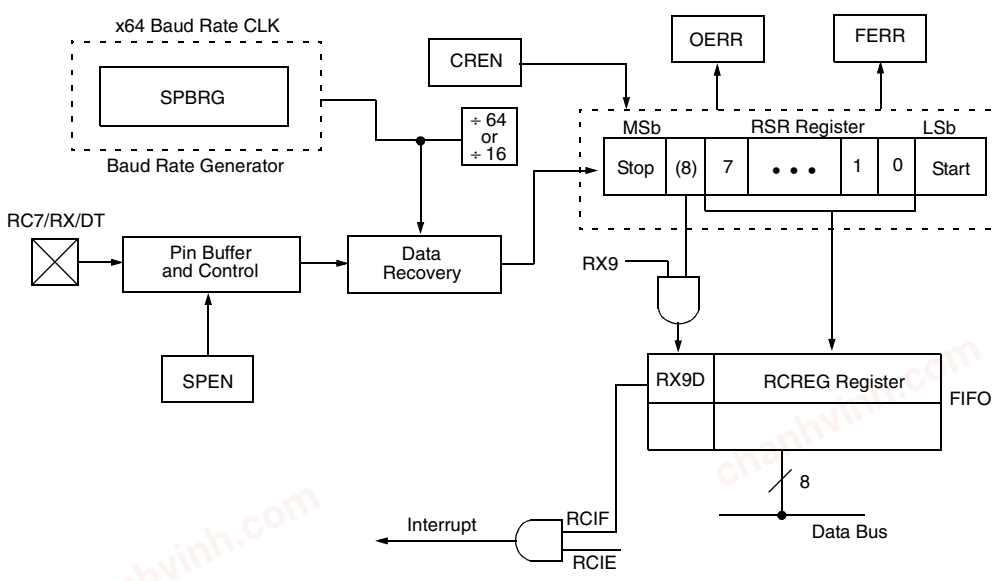
1. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (**Section 18.1 "USART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCIE.
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.

### 18.2.3 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 18-4: USART RECEIVE BLOCK DIAGRAM**

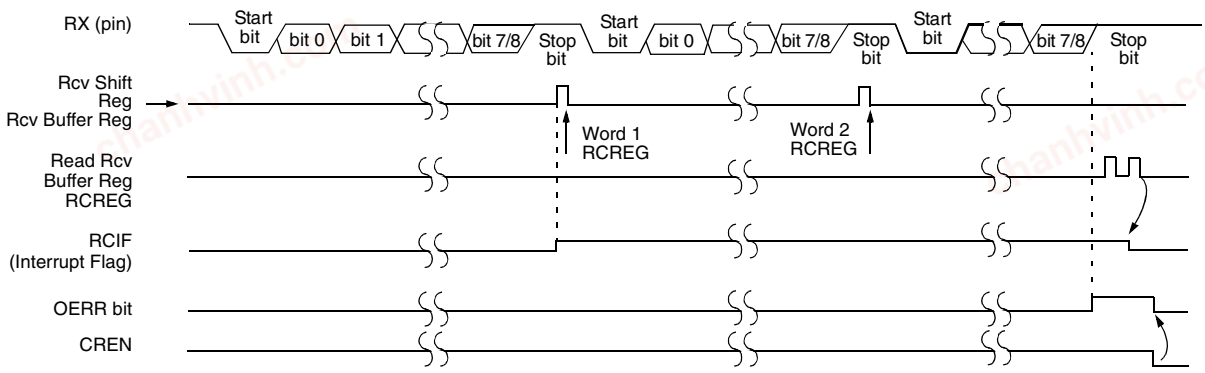



**Note:** I/O pins have diode protection to VDD and VSS.

chanhvinh.com



**FIGURE 18-5: ASYNCHRONOUS RECEPTION**




**Note:** This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

**TABLE 18-7: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000u |
| RCREG | USART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 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.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 18.3 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).

### 18.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 18-1. 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 (PIR1 register) is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1 register). 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 18.1 "USART Baud Rate Generator (BRG)"**).
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.

| **Note:** | TXIF is not cleared immediately upon loading data into the transmit buffer TXREG. The flag bit becomes valid in the second instruction cycle following the load instruction. |
|---|---|

**TABLE 18-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000u |
| TXREG | USART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| SPBRG | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

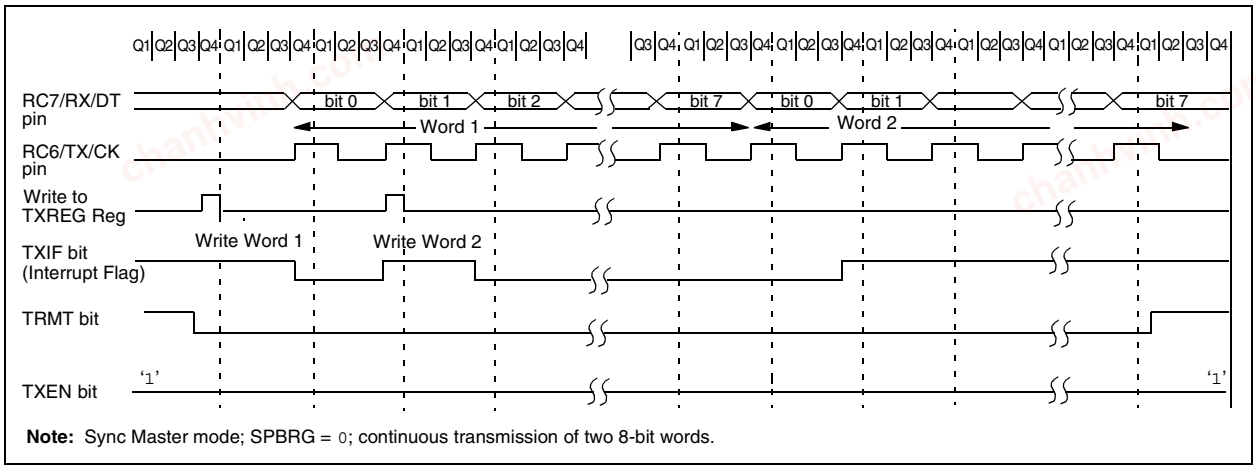
**Legend:** x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

chanhvinh.com

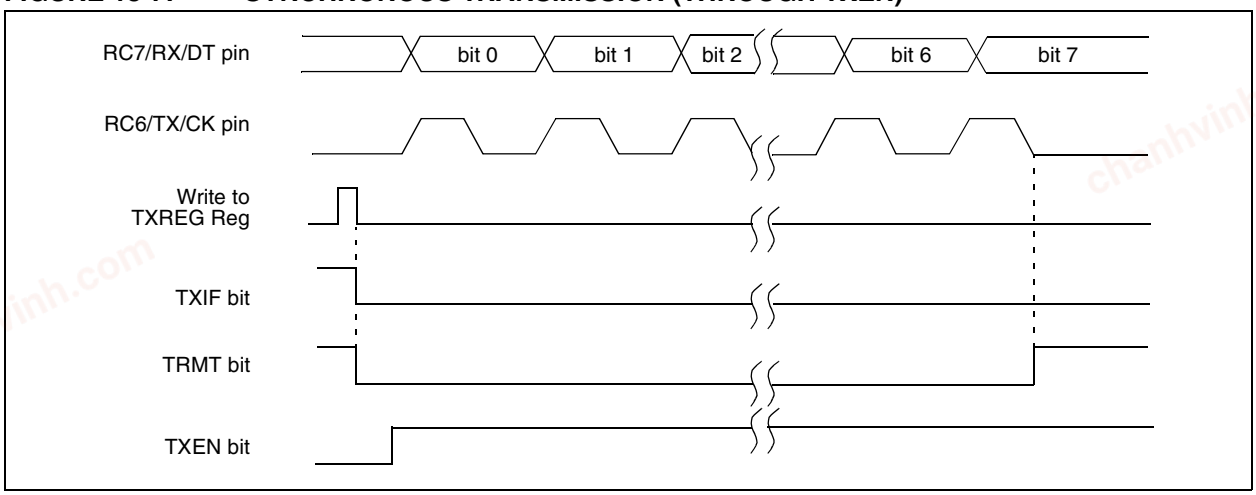


**FIGURE 18-6: SYNCHRONOUS TRANSMISSION**




**Note:** Sync Master mode; SPBRG = 0; continuous transmission of two 8-bit words.

**FIGURE 18-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**





© 2006 Microchip Technology Inc.

chanhvinh.com



### 18.3.2 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:

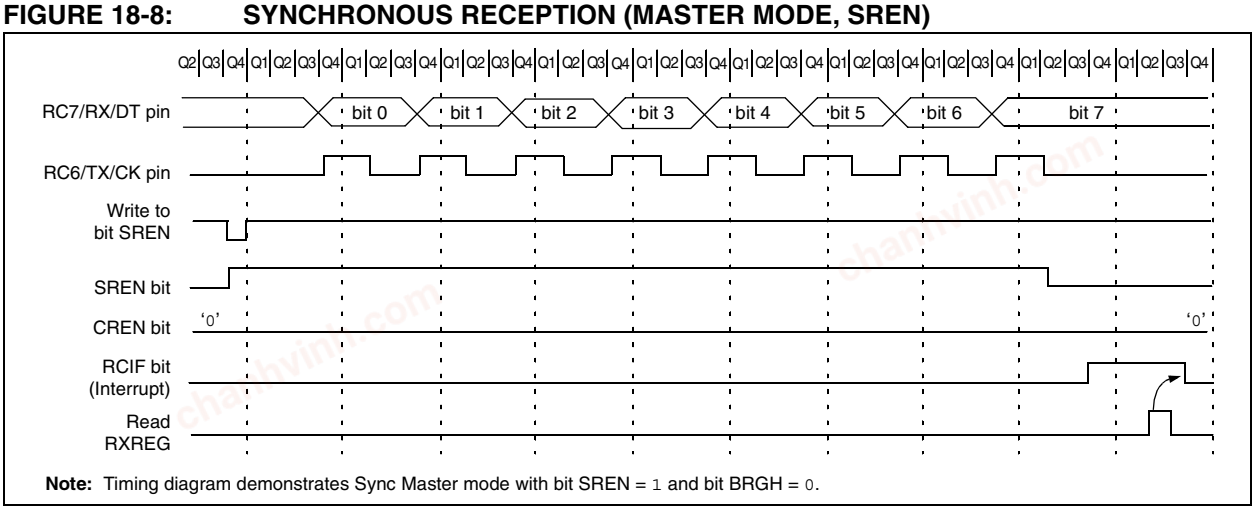
1. Initialize the SPBRG register for the appropriate baud rate (**Section 18.1 "USART Baud Rate Generator (BRG)"**).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, set enable bit RCIE.
5. If 9-bit 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 18-9: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000u |
| RCREG | USART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 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.
**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

**FIGURE 18-8: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**




**Note:** Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.

chanhvinh.com



## 18.4 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).

### 18.4.1 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.

### 18.4.2 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.


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 18-10: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000u |
| TXREG | USART Transmit Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 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.
**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

**TABLE 18-11: 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000u |
| RCREG | USART Receive Register | | | | | | | | 0000 0000 | 0000 0000 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| 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.
**Note 1:** These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

chanhvinh.com

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



# 19.0 CAN MODULE

## 19.1 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, CAN 2.0B 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:

- Complies with ISO CAN Conformance Test
- Implementation of the CAN protocol CAN 1.2, CAN 2.0A and CAN 2.0B
- 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

### 19.1.1 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

CAN module uses RB3/CANRX and RB2/CANTX/INT2 pins to interface with CAN bus. In order to configure CANRX and CANTX as CAN interface:

- bit TRISB<3> must be set;
- bit TRISB<2> must be cleared.

### 19.1.2 TRANSMIT/RECEIVE BUFFERS

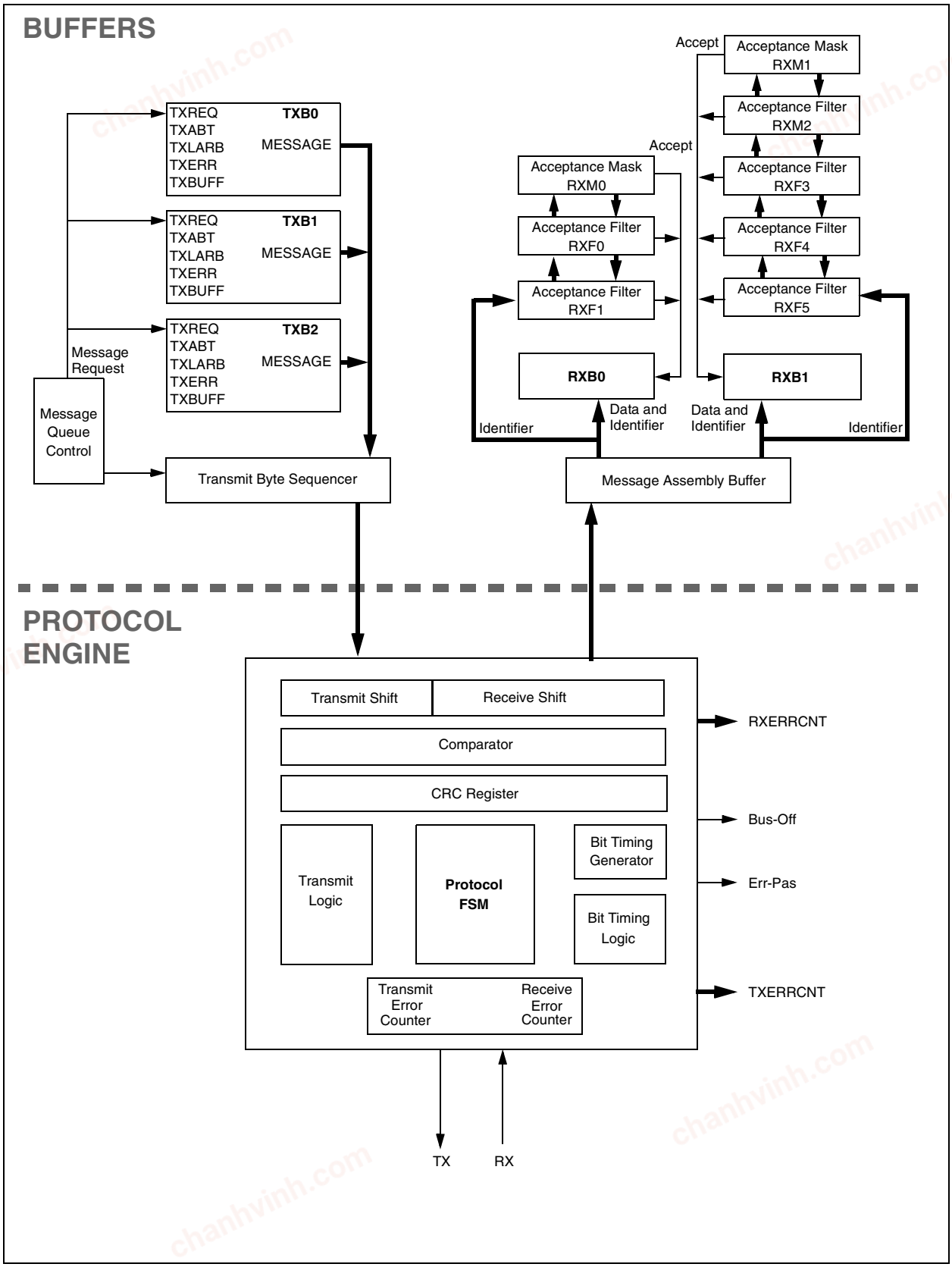
The PIC18FXX8 has three transmit and two receive buffers, two acceptance masks (one for each receive buffer) and a total of six acceptance filters. Figure 19-1 is a block diagram of these buffers and their connection to the protocol engine.

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 19-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM**





© 2006 Microchip Technology Inc.

chanhvinh.com



## 19.2 CAN Module Registers

| Note: | Not all CAN registers are available in the Access Bank. |
|---|---|

There are many control and data registers associated with the CAN module. For convenience, their descriptions have been grouped into the following sections:

- Control and Status Registers
- Transmit Buffer Registers (Data and Control)
- Receive Buffer Registers (Data and Control)
- Baud Rate Control Registers
- I/O Control Register
- Interrupt Status and Control Registers

### 19.2.1 CAN CONTROL AND STATUS REGISTERS

The registers described in this section control the overall operation of the CAN module and show its operational status.

**REGISTER 19-1: 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 7 | | | | | | | bit 0 |

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 19-1 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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-2: 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 19-1 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**EXAMPLE 19-1: 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
```

© 2006 Microchip Technology Inc.

chanhvinh.com



**EXAMPLE 19-1: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS (CONTINUED)**

```
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    CANCON, W                   ; Clear CANCON.WIN bits before copying
                                        ; new ones.
    ANDLW   b'11110001'                 ; Use previously saved CANCON value to
                                        ; make sure same value.
    MOVWF   CANCON                      ; 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   CANCON                      ; Copy ICODE bits to WIN bits.
                                        ; Copy the result to actual CANCON

    ; Access current buffer…
    ; User code

    ; Restore CANCON.WIN bits
    MOVF    CANCON, W                   ; Preserve current non WIN bits
    ANDLW   b'11110001'
    IORWF   TempCANCON, W               ; Restore original WIN bits

    MOVWF   CANCON

    ; Do not need to restore CANSTAT - it is read-only register.

    ; Return from interrupt or check for another module interrupt source
```


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-3: 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 = 127 ≥ Transmit Error Counter > 95
0 = Transmit Error Counter ≤ 95

bit 1 **RXWARN:** Receiver Warning bit
1 = 127 ≥ 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 | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

© 2006 Microchip Technology Inc.

chanhvinh.com



### 19.2.2 CAN TRANSMIT BUFFER REGISTERS

This section describes the CAN Transmit Buffer registers and their associated control registers.

**REGISTER 19-4: TXBnCON: TRANSMIT BUFFER n CONTROL REGISTERS**

| 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, TXLARB 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 the Transmit Buffer will be transferred. They do not alter the 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-5: TXBnSIDH: TRANSMIT BUFFER n STANDARD IDENTIFIER, HIGH BYTE REGISTERS**

| 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) or 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 19-6: TXBnSIDL: TRANSMIT BUFFER n STANDARD IDENTIFIER, LOW BYTE REGISTERS**

| 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 or 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 19-7: TXBnEIDH: TRANSMIT BUFFER n EXTENDED IDENTIFIER, HIGH BYTE REGISTERS**

| 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 |

chanhvinh.com



**REGISTER 19-8: TXBnEIDL: TRANSMIT BUFFER n EXTENDED IDENTIFIER, LOW BYTE REGISTERS**

| 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 19-9: TXBnDm: TRANSMIT BUFFER n DATA FIELD BYTE m REGISTERS**

| 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 7-0 **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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-10: TXBnDLC: TRANSMIT BUFFER n DATA LENGTH CODE REGISTERS**

| 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 19-11: 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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



### 19.2.3 CAN RECEIVE BUFFER REGISTERS

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

**REGISTER 19-12: 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-0 |
|---|---|---|---|---|---|---|---|
| RXFUL(1) | RXM1(1) | RXM0(1) | — | RXRTRRO | RXB0DBEN | JTOFF | FILHIT0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **RXFUL:** Receive Full Status bit(1)
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 must be cleared by software after the buffer is read.

bit 6-5 **RXM1:RXM0:** Receive Buffer Mode bits(1)
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 RXB0DBEN)
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)

**Note 1:** Bits RXFUL, RXM1 and RXM0 of RXB0CON are not mirrored in RXB1CON.

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


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-13: 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(1) | RXM1(1) | RXM0(1) | — | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **RXFUL:** Receive Full Status bit(1)

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(1)

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

**Note 1:** Bits RXFUL, RXM1 and RXM0 of RXB1CON are not mirrored in RXB0CON.

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

chanhvinh.com



**REGISTER 19-14: RXBnSIDH: RECEIVE BUFFER n STANDARD IDENTIFIER, HIGH BYTE REGISTERS**

| 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) or 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 |

**REGISTER 19-15: RXBnSIDL: RECEIVE BUFFER n STANDARD IDENTIFIER, LOW BYTE REGISTERS**

| 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 or Extended Identifier bits EID20:EID18 if EXID = 1

bit 4 **SRR:** Substitute Remote Request bit
This bit is always '0' when EXID = 1 or equal to the value of RXRTRRO (RXnBCON<3>) when EXID = 0.

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 19-16: RXBnEIDH: RECEIVE BUFFER n EXTENDED IDENTIFIER, HIGH BYTE REGISTERS**

| 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-17: RXBnEIDL: RECEIVE BUFFER n EXTENDED IDENTIFIER, LOW BYTE REGISTERS**

| 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 19-18: RXBnDLC: RECEIVE BUFFER n DATA LENGTH CODE REGISTERS**

| U-0 | 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 |

chanhvinh.com



**REGISTER 19-19: RXBnDm: RECEIVE BUFFER n DATA FIELD BYTE m REGISTERS**

| 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 19-20: 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



### 19.2.3.1 Message Acceptance Filters and Masks

This subsection describes the message acceptance filters and masks for the CAN receive buffers.

**REGISTER 19-21: RXFnSIDH: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER, HIGH BYTE REGISTERS**

| 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 or 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 19-22: RXFnSIDL: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER, LOW BYTE REGISTERS**

| 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 or Extended Identifier Filter bits EID20:EID18 if EXIDEN = 1

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 |

chanhvinh.com



**REGISTER 19-23: RXFnEIDH: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER, HIGH BYTE REGISTERS**

| 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 |

**REGISTER 19-24: RXFnEIDL: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER, LOW BYTE REGISTERS**

| 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 19-25: RXMnSIDH: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK, HIGH BYTE REGISTERS**

| 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-26: RXMnSIDL: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK, LOW BYTE REGISTERS**

| 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 19-27: RXMnEIDH: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK, HIGH BYTE REGISTERS**

| 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 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 19-28: RXMnEIDL: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK, LOW BYTE REGISTERS**

| 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 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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



### 19.2.4 CAN BAUD RATE REGISTERS

This subsection describes the CAN Baud Rate registers.

**REGISTER 19-29: 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 TQ
10 = Synchronization Jump Width Time = 3 x TQ
01 = Synchronization Jump Width Time = 2 x TQ
00 = Synchronization Jump Width Time = 1 x TQ

bit 5-0 **BRP5:BRP0:** Baud Rate Prescaler bits

111111 = TQ = (2 x 64)/FOSC
111110 = TQ = (2 x 63)/FOSC
:
:
000001 = TQ = (2 x 2)/FOSC
000000 = TQ = (2 x 1)/FOSC

| **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 accessible in Configuration mode only.


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-30: 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 TQ
110 = Phase Segment 1 Time = 7 x TQ
101 = Phase Segment 1 Time = 6 x TQ
100 = Phase Segment 1 Time = 5 x TQ
011 = Phase Segment 1 Time = 4 x TQ
010 = Phase Segment 1 Time = 3 x TQ
001 = Phase Segment 1 Time = 2 x TQ
000 = Phase Segment 1 Time = 1 x TQ

bit 2-0 **PRSEG2:PRSEG0:** Propagation Time Select bits
111 = Propagation Time = 8 x TQ
110 = Propagation Time = 7 x TQ
101 = Propagation Time = 6 x TQ
100 = Propagation Time = 5 x TQ
011 = Propagation Time = 4 x TQ
010 = Propagation Time = 3 x TQ
001 = Propagation Time = 2 x TQ
000 = Propagation Time = 1 x TQ

| **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 accessible in Configuration mode only.

chanhvinh.com



**REGISTER 19-31: 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(1) | SEG2PH1(1) | SEG2PH0(1) |
| 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(1)
111 = Phase Segment 2 Time = 8 x TQ
110 = Phase Segment 2 Time = 7 x TQ
101 = Phase Segment 2 Time = 6 x TQ
100 = Phase Segment 2 Time = 5 x TQ
011 = Phase Segment 2 Time = 4 x TQ
010 = Phase Segment 2 Time = 3 x TQ
001 = Phase Segment 2 Time = 2 x TQ
000 = Phase Segment 2 Time = 1 x TQ

**Note 1:** Ignored if SEG2PHTS bit (BRGCON2<7>) 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



### 19.2.5 CAN MODULE I/O CONTROL REGISTER

This register controls the operation of the CAN module's I/O pins in relation to the rest of the microcontroller.

**REGISTER 19-32: CIOCON: CAN I/O CONTROL REGISTER**

| U-0 | U-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | ENDRHI | CANCAP | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

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

bit 5 **ENDRHI:** Enable Drive High bit
1 = CANTX pin will drive VDD when recessive
0 = CANTX pin will tri-state when recessive

bit 4 **CANCAP:** CAN Message Receive Capture Enable bit
1 = Enable CAN capture, CAN message receive signal replaces input on RC2/CCP1
0 = Disable CAN capture, RC2/CCP1 input to CCP1 module

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 |

chanhvinh.com



### 19.2.6 CAN INTERRUPT REGISTERS

The registers in this section are the same as described in **Section 8.0 "Interrupts"**. They are duplicated here for convenience.

**REGISTER 19-33: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3**

| 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 reloaded
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 reloaded
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 reloaded
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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-34: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

| 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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 19-35: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 19-1: CAN CONTROLLER REGISTER MAP**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| F7Fh | — | F5Fh | — | F3Fh | — | F1Fh | RXM1EIDL |
| F7Eh | — | F5Eh | CANSTATRO1(2) | F3Eh | CANSTATRO3(2) | 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 | CANSTATRO2(2) | F2Eh | CANSTATRO4(2) | 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 | RXF0EIDH |
| F61h | RXB0SIDH | F41h | TXB0SIDH | F21h | TXB2SIDH | F01h | RXF0SIDL |
| F60h | RXB0CON | F40h | TXB0CON | F20h | TXB2CON | F00h | RXF0SIDH |

**Note 1:** Shaded registers are available in Access Bank low area while the rest are available in Bank 15.

**2:** 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.

© 2006 Microchip Technology Inc.

chanhvinh.com



## 19.3 CAN Modes of Operation

The PIC18FXX8 has six main modes of operation:

- Configuration mode
- Disable mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode

All modes, except Error Recognition, are requested by setting the REQOP bits (CANCON<7:5>); Error Recognition is requested through the RXM bits of the Receive Buffer register(s). 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.

### 19.3.1 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 the REQOP2 bit. Only when the OPMODE2 status bit has a high level can the initialization be performed. Afterwards, the Configuration registers, the Acceptance Mask registers and the Acceptance Filter registers can be written. The module is activated by setting the REQOP control bits 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 online. 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.

### 19.3.2 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. If the processor receives an interrupt while it is sleeping, more than one message may get lost. User firmware must anticipate this condition and request retransmission. If the processor is running while it receives an interrupt, only the first message may get lost.

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

### 19.3.3 NORMAL MODE

This is the standard operating mode of the PIC18FXX8. 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 PIC18FXX8 will transmit messages over the CAN bus.

### 19.3.4 LISTEN ONLY MODE

Listen Only mode provides a means for the PIC18FXX8 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.


© 2006 Microchip Technology Inc.

chanhvinh.com



### 19.3.5 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.

### 19.3.6 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive all message. The Error Recognition mode is activated by setting the RXM<1:0> bits in the RXBnCON registers to '11'. In this mode, all messages, valid or invalid, are received and copied to the receive buffer.

## 19.4 CAN Message Transmission

### 19.4.1 TRANSMIT BUFFERS

The PIC18FXX8 implements three transmit buffers (Figure 19-2). Each of these buffers occupies 14 bytes of SRAM and are mapped into the device memory map.

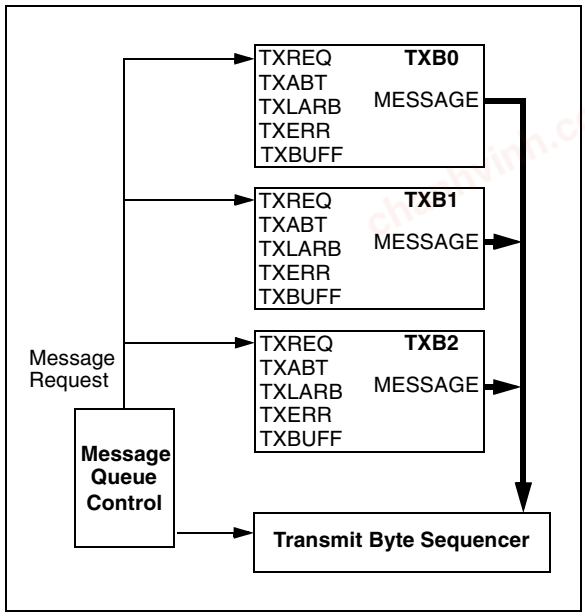
For the MCU to have write access to the message buffer, the TXREQ bit must be clear, indicating that the message buffer is clear of any pending message to be transmitted. At a minimum, the TXBnSIDH, TXBnSIDL and TXBnDLC registers must be loaded. If data bytes are present in the message, the TXBnDm registers 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 19.4.2 "Transmit Priority"**).

### 19.4.2 TRANSMIT PRIORITY

Transmit priority is a prioritization within the PIC18FXX8 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.

**FIGURE 19-2: TRANSMIT BUFFER BLOCK DIAGRAM**



chanhvinh.com



### 19.4.3 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.

### 19.4.4 ABORTING TRANSMISSION

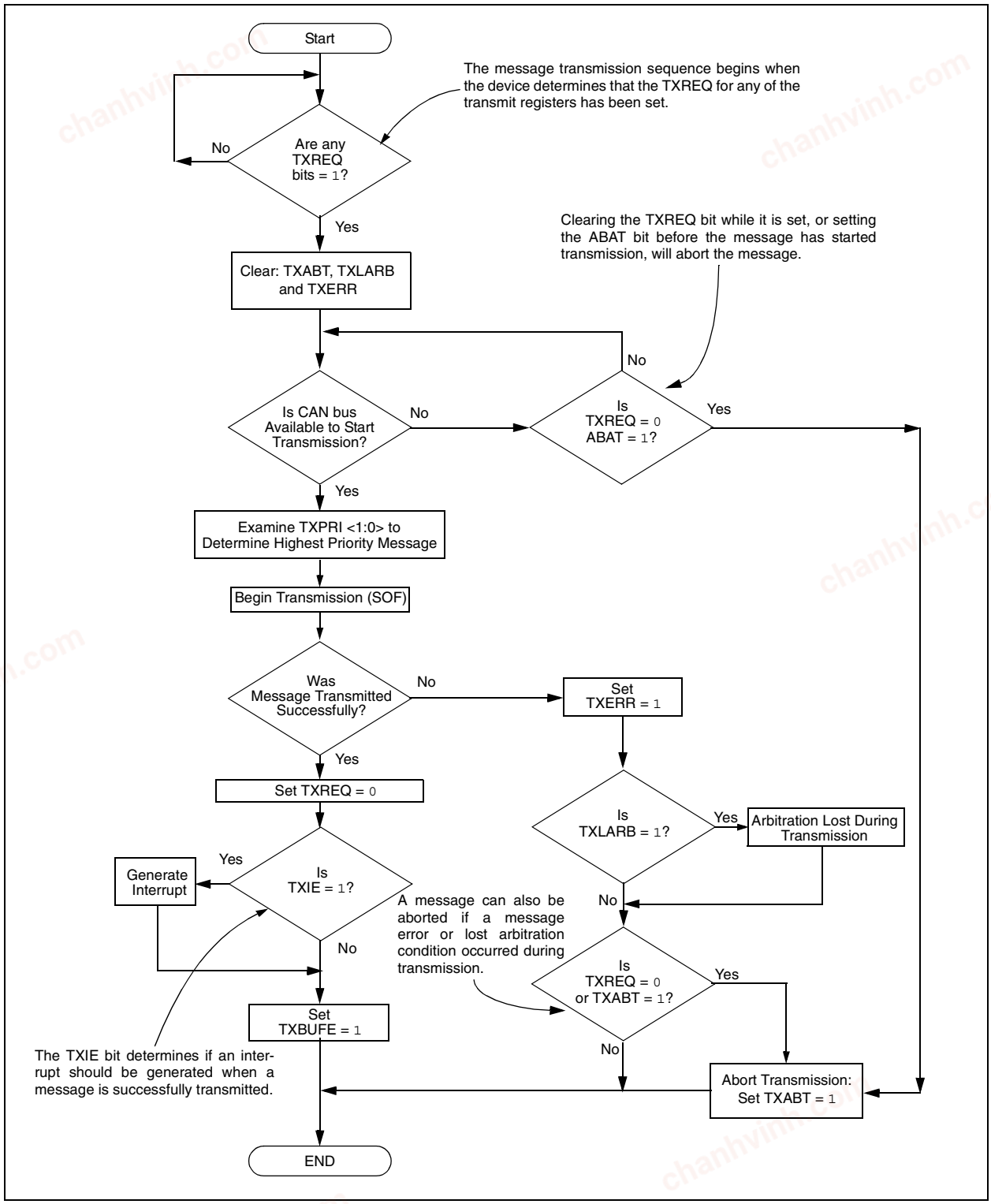
The MCU can request to abort a message by clearing the TXREQ bit associated with the corresponding message buffer (TXBnCON<3>). Setting the ABAT bit (CANCON<4>) will request an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets the ABT bits for the corresponding buffer (TXBnCON<6>). 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 ABT 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 retransmitted and the ABT bit will be set, indicating that the message was successfully aborted.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 19-3: INTERNAL TRANSMIT MESSAGE FLOWCHART**



chanhvinh.com



## 19.5 Message Reception

### 19.5.1 RECEIVE MESSAGE BUFFERING

The PIC18FXX8 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 19-4).

### 19.5.2 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 are 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 PIC18FXX8 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.

### 19.5.3 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 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 19.6 "Message Acceptance Filters and Masks"**).

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.

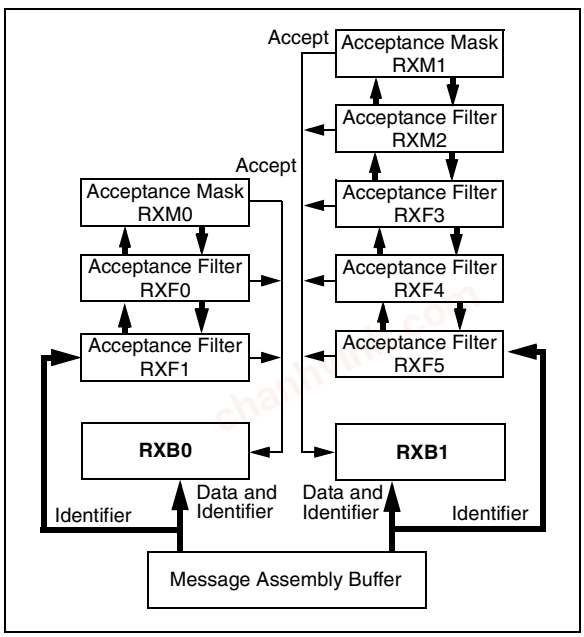
The RXM bits set special Receive modes. Normally, these bits are set to '00' to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the 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.

### 19.5.4 TIME-STAMPING

The CAN module can be programmed to generate a time-stamp for every message that is received. When enabled, the module generates a capture signal for CCP1 which in turns captures the value of either Timer1 or Timer3. This value can be used as the message time-stamp.

To use the time-stamp capability, the CANCAP bit (CIOCAN<4>) must be set. This replaces the capture input for CCP1 with the signal generated from the CAN module. In addition, CCP1CON<3:0> must be set to '0011' to enable the CCP special event trigger for CAN events.

**FIGURE 19-4: RECEIVE BUFFER BLOCK DIAGRAM**



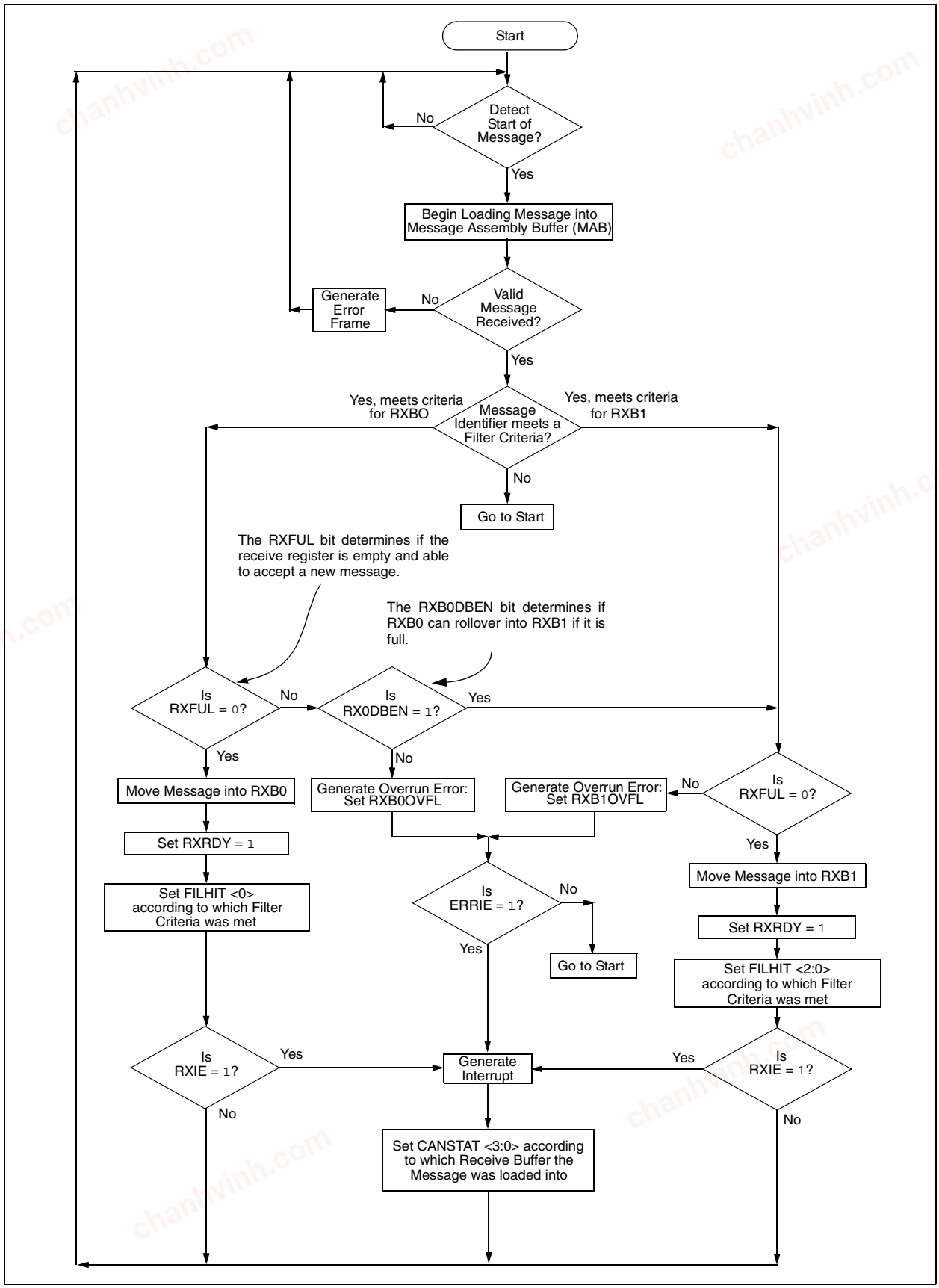


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 19-5: INTERNAL MESSAGE RECEPTION FLOWCHART**



chanhvinh.com



## 19.6 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 19-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if a 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 19-2: 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 buffer block diagram (Figure 19-4), 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 rollover 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 rollover into RXB1.

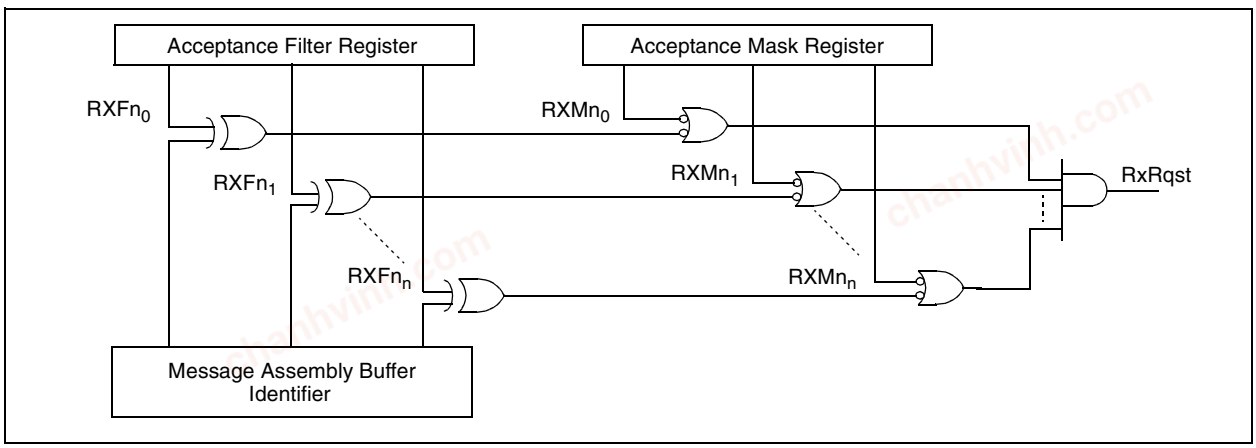
- 111 = Acceptance Filter 1 (RXF1)
- 110 = Acceptance Filter 0 (RXF0)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0

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 rollover 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 PIC18FXX8 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 19-6: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION**





© 2006 Microchip Technology Inc.

chanhvinh.com



## 19.7 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 PIC18FXX8 is implemented using a DPLL that is configured to synchronize to the incoming data and provides 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* (TQ).

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 1 Mb/s.

The *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 (Figure 19-7) include:

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

The time segments (and thus, the nominal bit time) are, in turn, made up of integer units of time called time quanta or TQ (see Figure 19-7). By definition, the nominal bit time is programmable from a minimum of 8 TQ to a maximum of 25 TQ. Also, by definition, the minimum nominal bit time is 1 µs corresponding to a maximum 1 Mb/s rate. The actual duration is given by the relationship:

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

The time quantum is a fixed unit derived from the oscillator period. It is also defined by the programmable baud rate prescaler, with integer values from 1 to 64, in addition to a fixed divide-by-two for clock generation. Mathematically, this is

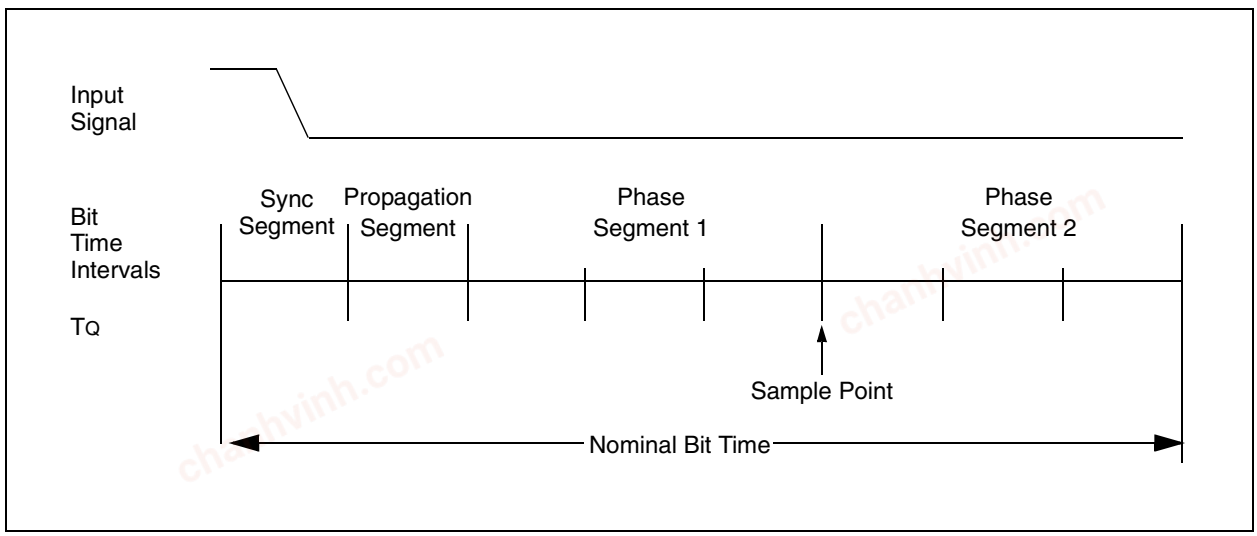
$$T_Q\ (\mu s) = (2 * (BRP + 1))/F_{OSC}\ (MHz)$$

or

$$T_Q\ (\mu s) = (2 * (BRP + 1)) * T_{OSC}\ (\mu s)$$

where FOSC is the clock frequency, TOSC is the corresponding oscillator period and BRP is an integer (0 through 63) represented by the binary values of BRGCON1<5:0>.

**FIGURE 19-7: BIT TIME PARTITIONING**



chanhvinh.com



### 19.7.1 TIME QUANTA

As already mentioned, the time quanta is a fixed unit derived from the oscillator period and baud rate prescaler. Its relationship to TBIT and the nominal bit rate is shown in Example 19-2.

**EXAMPLE 19-2: CALCULATING TQ, NOMINAL BIT RATE AND NOMINAL BIT TIME**

TQ (µs) = (2 * (BRP + 1))/FOSC (MHz)

TBIT (µs) = TQ (µs) * number of TQ per bit interval

Nominal Bit Rate (bits/s) = 1/TBIT

**CASE 1:**

*For FOSC = 16 MHz, BRP<5:0> = 00h and Nominal Bit Time = 8 TQ:*

TQ = (2 * 1)/16 = 0.125 µs (125 ns)

TBIT = 8 * 0.125 = 1 µs ($10^{-6}$s)

Nominal Bit Rate = $1/10^{-6} = 10^{6}$ bits/s (1 Mb/s)

**CASE 2:**

*For FOSC = 20 MHz, BRP<5:0> = 01h and Nominal Bit Time = 8 TQ:*

TQ = (2 * 2)/20 = 0.2 µs (200 ns)

TBIT = 8 * 0.2 = 1.6 µs (1.6 * $10^{-6}$s)

Nominal Bit Rate = 1/1.6 * $10^{-6}$s = 625,000 bits/s (625 Kb/s)

**CASE 3:**

*For FOSC = 25 MHz, BRP<5:0> = 3Fh and Nominal Bit Time = 25 TQ:*

TQ = (2 * 64)/25 = 5.12 µs

TBIT = 25 * 5.12 = 128 µs (1.28 * $10^{-4}$s)

Nominal Bit Rate = 1/1.28 * $10^{-4}$ = 7813 bits/s (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 an 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 ensured to operate correctly.

### 19.7.2 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.

### 19.7.3 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.

### 19.7.4 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).

### 19.7.5 SAMPLE POINT

The sample point is the point of time at which the bus level is read and the 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.

### 19.7.6 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 PIC18FXX8 defines this time to be 2 TQ. Thus, Phase Segment 2 must be at least 2 TQ long.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 19.8 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.

### 19.8.1 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.

### 19.8.2 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 19-8) or subtracted from Phase Segment 2 (see Figure 19-9). The SJW is programmable between 1 TQ and 4 TQ.

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 TQ. The phase error is defined in magnitude of TQ as follows:

- e = 0 if the edge lies within Sync_Seg.
- 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.

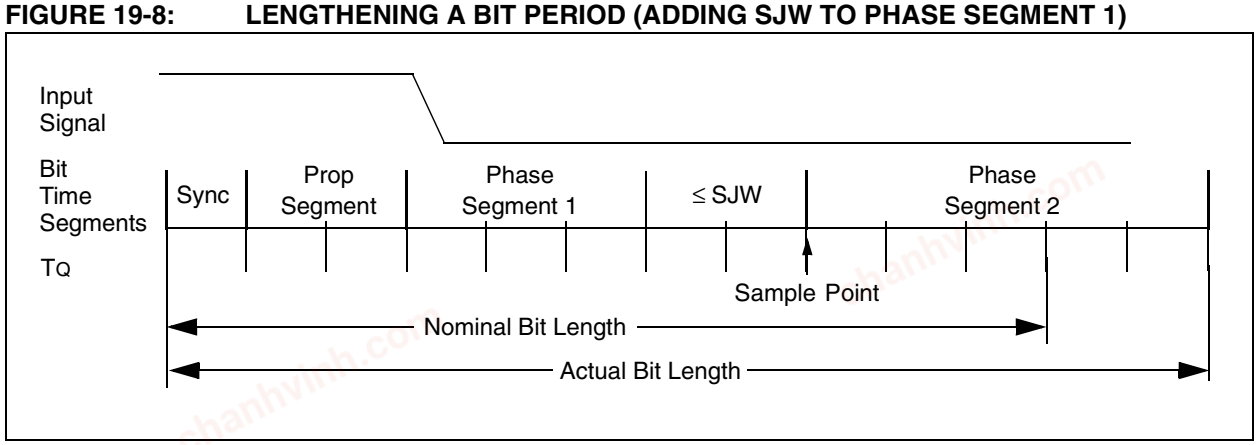
If the magnitude of the phase error is larger than the synchronization jump width and if the phase error is positive, then Phase Segment 1 is lengthened by an amount equal to the synchronization jump width.

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.

### 19.8.3 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.

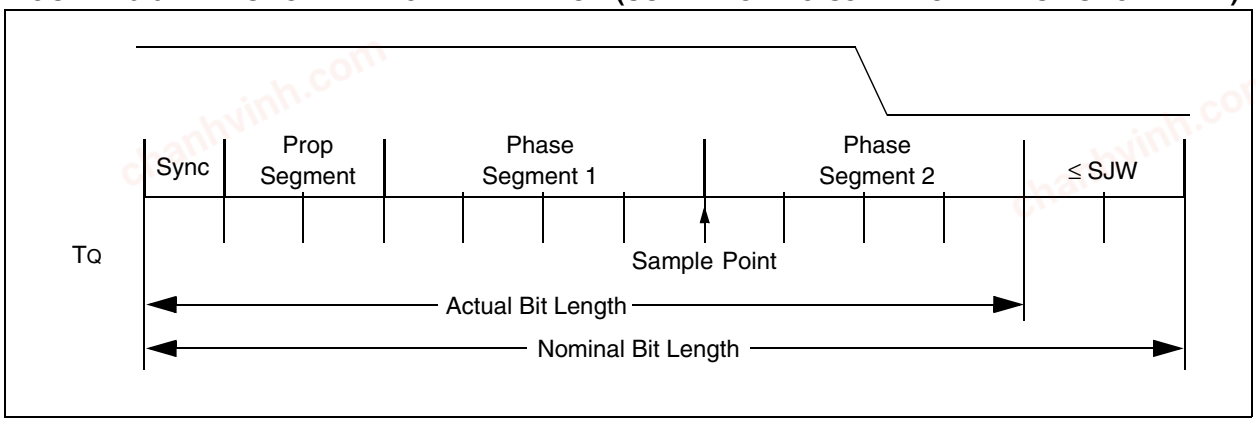
**FIGURE 19-8: LENGTHENING A BIT PERIOD (ADDING SJW TO PHASE SEGMENT 1)**



chanhvinh.com



**FIGURE 19-9: SHORTENING A BIT PERIOD (SUBTRACTING SJW FROM PHASE SEGMENT 2)**




## 19.9 Programming Time Segments

Some requirements for programming of the time segments:

- Prop Seg + Phase Seg 1 ≥ Phase Seg 2
- Phase Seg 2 ≥ Sync Jump Width

For example, assume that a 125 kHz CAN baud rate is desired using 20 MHz for FOSC. With a TOSC of 50 ns, a baud rate prescaler value of 04h gives a TQ of 500 ns. To obtain a nominal bit rate of 125 kHz, the nominal bit time must be 8 µs or 16 TQ.

Using 1 TQ for the Sync Segment, 2 TQ for the Propagation Segment and 7 TQ for Phase Segment 1 would place the sample point at 10 TQ after the transition. This leaves 6 TQ for Phase Segment 2.

By the rules above, the Sync Jump Width could be the maximum of 4 TQ. 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. Typically, an SJW of 1 is enough.

## 19.10 Oscillator Tolerance

As a rule of thumb, the bit timing requirements allow ceramic resonators to be used in applications with transmission rates of up to 125 Kbit/sec. 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.

## 19.11 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 PIC18FXX8 is in Configuration mode.

### 19.11.1 BRGCON1

The BRP bits control the baud rate prescaler. The SJW<1:0> bits select the synchronization jump width in terms of multiples of TQ.

### 19.11.2 BRGCON2

The PRSEG bits set the length of the Propagation Segment in terms of TQ. The SEG1PH bits set the length of Phase Segment 1 in TQ. 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 TQ/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 TQ for the PIC18FXX8).

### 19.11.3 BRGCON3

The PHSEG2<2:0> bits set the length (in TQ) 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.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 19.12 Error Detection

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

### 19.12.1 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.

### 19.12.2 ACKNOWLEDGE ERROR

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which was 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.

### 19.12.3 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.

### 19.12.4 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.

### 19.12.5 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.

### 19.12.6 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 activate 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.

### 19.12.7 ERROR MODES AND ERROR COUNTERS

The PIC18FXX8 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 PIC18FXX8 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 19-10). 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.

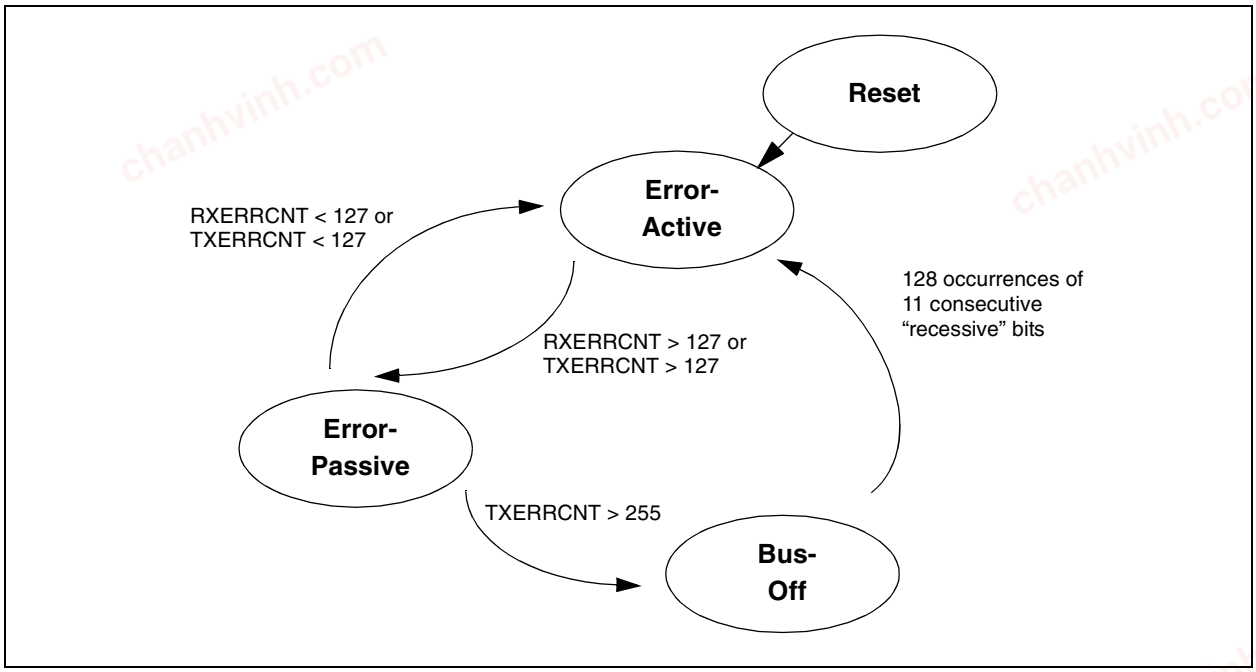
Additionally, there is an Error State Warning flag bit, EWARN, which is set if at least one of the error counters equals or exceeds the error warning limit of 96. EWARN is reset if both error counters are less than the error warning limit.

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 19-10: ERROR MODES STATE DIAGRAM**




## 19.13 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, the 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

### 19.13.1 INTERRUPT CODE BITS

The source of a pending interrupt is indicated in the ICODE (Interrupt Code) bits of the CANSTAT register (ICODE<2:0>). Interrupts are internally prioritized such that the higher priority interrupts are assigned lower ICODE values. 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 19-3, following page). Note that only those interrupt sources that have their associated CANINTE enable bit set will be reflected in the ICODE bits.

### 19.13.2 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'.

### 19.13.3 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'.


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 19-3: VALUES FOR ICODE<2:0>**

| ICOD <2:0> | Interrupt | Boolean Expression |
|---|---|---|
| 000 | None | $\overline{ERR}$•$\overline{WAK}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•$\overline{RX0}$•$\overline{RX1}$ |
| 001 | Error | ERR |
| 010 | TXB2 | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•TX2 |
| 011 | TXB1 | $\overline{ERR}$•$\overline{TX0}$•TX1 |
| 100 | TXB0 | $\overline{ERR}$•TX0 |
| 101 | RXB1 | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•$\overline{RX0}$•RX1 |
| 110 | RXB0 | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•RX0 |
| 111 | Wake on Interrupt | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•$\overline{RX0}$•$\overline{RX1}$•WAK |

Key:
ERR = ERRIF * ERRIE RX0 = RXB0IF * RXB0IE
TX0 = TXB0IF * TXB0IE RX1 = RXB1IF * RXB1IE
TX1 = TXB1IF * TXB1IE WAK = WAKIF * WAKIE
TX2 = TXB2IF * TXB2IE

### 19.13.4 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.

### 19.13.5 BUS ACTIVITY WAKE-UP INTERRUPT

When the PIC18FXX8 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 PIC18FXX8 to exit Sleep mode. The interrupt is reset by the MCU, clearing the WAKIF bit.

### 19.13.6 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.

#### 19.13.6.1 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.

#### 19.13.6.2 Receiver Warning

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

#### 19.13.6.3 Transmitter Warning

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

#### 19.13.6.4 Receiver Bus Passive

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

#### 19.13.6.5 Transmitter Bus Passive

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

#### 19.13.6.6 Bus-Off

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

### 19.13.7 INTERRUPT ACKNOWLEDGE

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

chanhvinh.com

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 20.0 COMPATIBLE 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has five inputs for the PIC18F2X8 devices and eight for the PIC18F4X8 devices. This module has the ADCON0 and ADCON1 register definitions that are compatible with the PICmicro® mid-range A/D module.

The A/D allows conversion of an analog input signal to a corresponding 10-bit digital number.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)

The ADCON0 register, shown in Register 20-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 20-2, configures the functions of the port pins.

**REGISTER 20-1: ADCON0: A/D CONTROL REGISTER 0**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | — | ADON |
| bit 7 | | | | | | | bit 0 |

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

| ADCON1 <ADCS2> | ADCON0 <ADCS1:ADCS0> | Clock Conversion |
|---|---|---|
| 0 | **00** | FOSC/2 |
| 0 | **01** | FOSC/8 |
| 0 | **10** | FOSC/32 |
| 0 | **11** | FRC (clock derived from the internal A/D RC oscillator) |
| 1 | **00** | FOSC/4 |
| 1 | **01** | FOSC/16 |
| 1 | **10** | FOSC/64 |
| 1 | **11** | FRC (clock derived from the internal A/D RC oscillator) |

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits
000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)(1)
110 = Channel 6 (AN6)(1)
111 = Channel 7 (AN7)(1)

**Note 1:** These channels are unimplemented on PIC18F2X8 (28-pin) devices. Do not select any unimplemented channel.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

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

bit 0 **ADON:** A/D On bit
1 = A/D converter module is powered up
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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 20-2: ADCON1: A/D CONTROL REGISTER 1**

| R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADFM | ADCS2 | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in **bold**)

| ADCON1 <ADCS2> | ADCON0 <ADCS1:ADCS0> | Clock Conversion |
|---|---|---|
| **0** | 00 | FOSC/2 |
| **0** | 01 | FOSC/8 |
| **0** | 10 | FOSC/32 |
| **0** | 11 | FRC (clock derived from the internal A/D RC oscillator) |
| **1** | 00 | FOSC/4 |
| **1** | 01 | FOSC/16 |
| **1** | 10 | FOSC/64 |
| **1** | 11 | FRC (clock derived from the internal A/D RC oscillator) |

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

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

| PCFG | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 | VREF+ | VREF- | C/R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | A | A | A | A | A | A | A | A | VDD | VSS | 8/0 |
| 0001 | A | A | A | A | VREF+ | A | A | A | AN3 | VSS | 7/1 |
| 0010 | D | D | D | A | A | A | A | A | VDD | VSS | 5/0 |
| 0011 | D | D | D | A | VREF+ | A | A | A | AN3 | VSS | 4/1 |
| 0100 | D | D | D | D | A | D | A | A | VDD | VSS | 3/0 |
| 0101 | D | D | D | D | VREF+ | D | A | A | AN3 | VSS | 2/1 |
| 011x | D | D | D | D | D | D | D | D | — | — | 0/0 |
| 1000 | A | A | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 6/2 |
| 1001 | D | D | A | A | A | A | A | A | VDD | VSS | 6/0 |
| 1010 | D | D | A | A | VREF+ | A | A | A | AN3 | VSS | 5/1 |
| 1011 | D | D | A | A | VREF+ | VREF- | A | A | AN3 | AN2 | 4/2 |
| 1100 | D | D | D | A | VREF+ | VREF- | A | A | AN3 | AN2 | 3/2 |
| 1101 | D | D | D | D | VREF+ | VREF- | A | A | AN3 | AN2 | 2/2 |
| 1110 | D | D | D | D | D | D | D | A | VDD | VSS | 1/0 |
| 1111 | D | D | D | D | VREF+ | VREF- | D | A | AN3 | AN2 | 1/2 |

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

**Note:** Shaded cells indicate channels available only on PIC18F4X8 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:** On any device Reset, the port pins that are multiplexed with analog functions (ANx) are forced to be analog inputs.


© 2006 Microchip Technology Inc.

chanhvinh.com



The analog reference voltage is software selectable to either the device's positive and negative supply voltage ($V_{DD}$ and $V_{SS}$) or the voltage level on the RA3/AN3/$V_{REF+}$ pin and RA2/AN2/$V_{REF-}$ pin.

The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

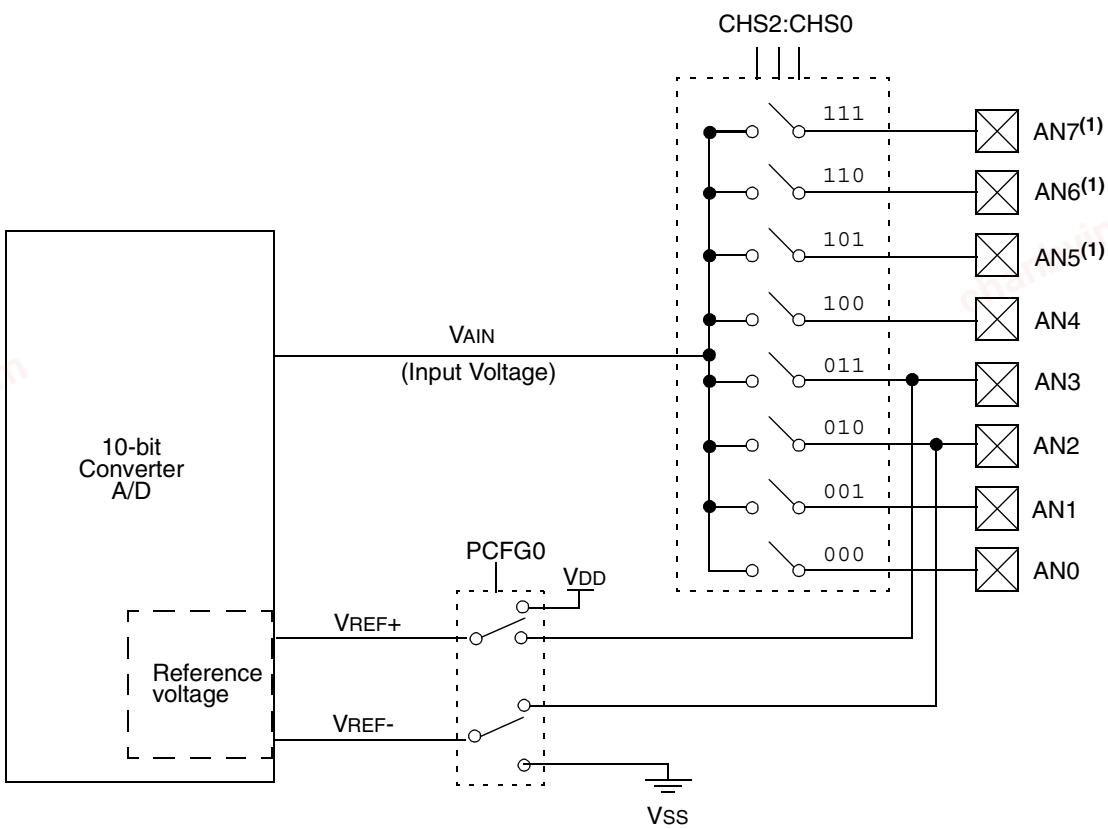
The output of the sample and hold is the input into the converter which generates the result via successive approximation.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion is aborted.

Each port pin associated with the A/D converter can be configured as an analog input (RA3 can also be a voltage reference) or as a digital I/O.

The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH/ADRESL registers, the GO/$\overline{DONE}$ bit (ADCON0<2>) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 20-1.

**FIGURE 20-1: A/D BLOCK DIAGRAM**




**Note 1:** Channels AN5 through AN7 are not available on PIC18F2X8 devices.

**2:** All I/O pins have diode protection to $V_{DD}$ and $V_{SS}$.

chanhvinh.com



The value that is 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 20.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started. The following steps should be followed for doing 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 (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

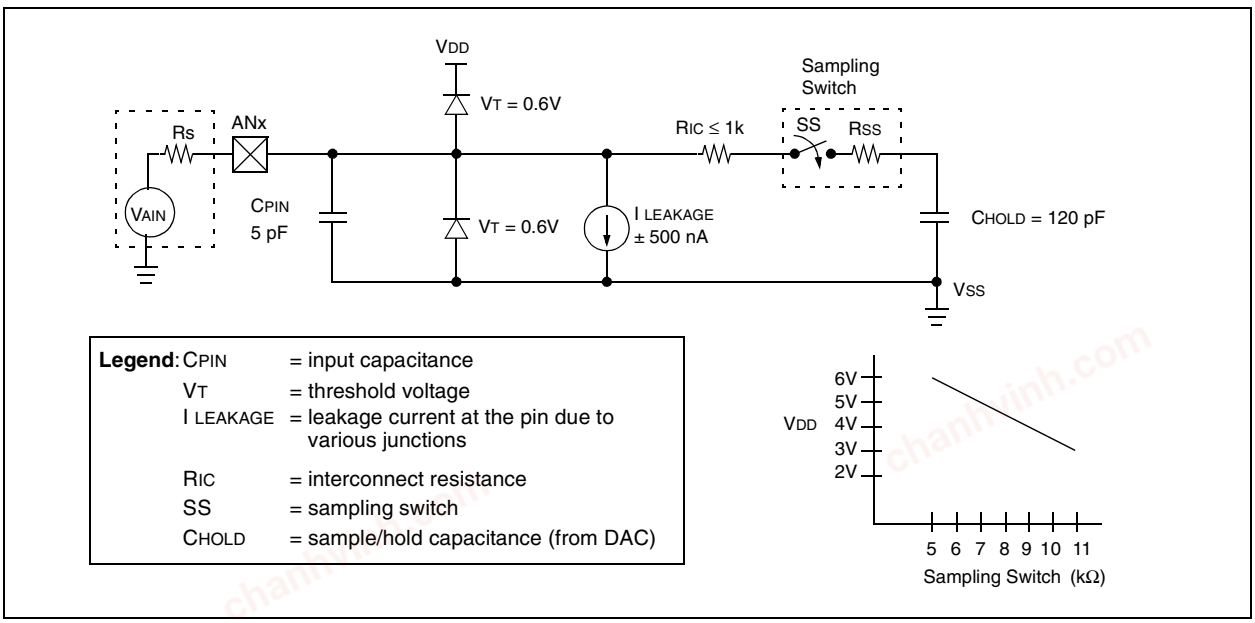
   OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH/ADRESL); clear bit ADIF if required.
7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required before next acquisition starts.

## 20.1 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 20-2. 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 2.5 kΩ.** 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. |
|---|---|

**FIGURE 20-2: ANALOG INPUT MODEL**





© 2006 Microchip Technology Inc.

chanhvinh.com



To calculate the minimum acquisition time, Equation 20-1 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 20-1 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 20-1: ACQUISITION TIME**

| | | |
|---|---|---|
| TACQ | = | Amplifier Settling Time + Holding Capacitor Charging Time + Temperature Coefficient |
| | = | TAMP + TC + TCOFF |

**EQUATION 20-2: A/D MINIMUM CHARGING TIME**

| | | |
|---|---|---|
| VHOLD | = | $(V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$ |
| or | | |
| TC | = | -(120 pF)(1 kΩ + RSS + RS) ln(1/2047) |

**EXAMPLE 20-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME**

| | | |
|---|---|---|
| TACQ | = | TAMP + TC + TCOFF |
| Temperature coefficient is only required for temperatures > 25°C. | | |
| TACQ | = | 2 µs + TC + [(Temp – 25°C)(0.05 µs/°C)] |
| TC | = | -CHOLD (RIC + RSS + RS) ln(1/2047)<br>-120 pF (1 kΩ + 7 kΩ + 2.5 kΩ) ln(0.0004885)<br>-120 pF (10.5 kΩ) ln(0.0004885)<br>-1.26 µs (-7.6241)<br>9.61 µs |
| TACQ | = | 2 µs + 9.61 µs + [(50°C – 25°C)(0.05 µs/°C)]<br>11.61 µs + 1.25 µs<br>12.86 µs |

**Note:** When using external voltage references with the A/D converter, the source impedance of the external voltage references must be less than 20Ω to obtain the specified A/D resolution. Higher reference source impedances will increase both offset and gain errors. Resistive voltage dividers will not provide a sufficiently low source impedance.

To maintain the best possible performance in A/D conversions, external VREF inputs should be buffered with an operational amplifier or other low output impedance circuit.

chanhvinh.com



## 20.2 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. The seven possible options for TAD are:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal RC oscillator.

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6 µs.

Table 20-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

## 20.3 Configuring Analog Port Pins

The ADCON1, TRISA and TRISE registers control the operation of the A/D port pins. The port pins that are desired 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 CHS2: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 that is defined as a digital input (including the AN4:AN0 pins) may cause the input buffer to consume current that is out of the device's specification. |

**TABLE 20-1: TAD vs. DEVICE OPERATING FREQUENCIES**

| AD Clock Source (TAD) | | Device Frequency | | | |
|---|---|---|---|---|---|
| **Operation** | **ADCS2:ADCS0** | **20 MHz** | **5 MHz** | **1.25 MHz** | **333.33 kHz** |
| 2 TOSC | 000 | 100 ns(2) | 400 ns(2) | 1.6 µs | 6 µs |
| 4 TOSC | 100 | 200 ns(2) | 800 ns(2) | 3.2 µs | 12 µs |
| 8 TOSC | 001 | 400 ns(2) | 1.6 µs | 6.4 µs | 24 µs(3) |
| 16 TOSC | 101 | 800 ns(2) | 3.2 µs | 12.8 µs | 48 µs(3) |
| 32 TOSC | 010 | 1.6 µs | 6.4 µs | 25.6 µs(3) | 96 µs(3) |
| 64 TOSC | 110 | 3.2 µs | 12.8 µs | 51.2 µs(3) | 192 µs(3) |
| RC | 011 | 2-6 µs(1) | 2-6 µs(1) | 2-6 µs(1) | 2-6 µs(1) |

**Legend:** Shaded cells are outside of recommended range.
**Note 1:** The RC source has a typical TAD time of 4 µs.
**2:** These values violate the minimum required TAD time.
**3:** For faster conversion times, the selection of another clock source is recommended.

**TABLE 20-2: TAD vs. DEVICE OPERATING FREQUENCIES (FOR EXTENDED, LF DEVICES)**

| AD Clock Source (TAD) | | Device Frequency | | | |
|---|---|---|---|---|---|
| **Operation** | **ADCS2:ADCS0** | **4 MHz** | **2 MHz** | **1.25 MHz** | **333.33 kHz** |
| 2 TOSC | 000 | 500 ns(2) | 1.0 µs(2) | 1.6 µs(2) | 6 µs |
| 4 TOSC | 100 | 1.0 µs(2) | 2.0 µs(2) | 3.2 µs(2) | 12 µs |
| 8 TOSC | 001 | 2.0 µs(2) | 4.0 µs | 6.4 µs | 24 µs(3) |
| 16 TOSC | 101 | 4.0 µs(2) | 8.0 µs | 12.8 µs | 48 µs(3) |
| 32 TOSC | 010 | 8.0 µs | 16.0 µs | 25.6 µs(3) | 96 µs(3) |
| 64 TOSC | 110 | 16.0 µs | 32.0 µs | 51.2 µs(3) | 192 µs(3) |
| RC | 011 | 3-9 µs(1) | 3-9 µs(1) | 3-9 µs(1) | 3-9 µs(1) |

**Legend:** Shaded cells are outside of recommended range.
**Note 1:** The RC source has a typical TAD time of 6 µs.
**2:** These values violate the minimum required TAD time.
**3:** For faster conversion times, the selection of another clock source is recommended.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 20.4 A/D Conversions

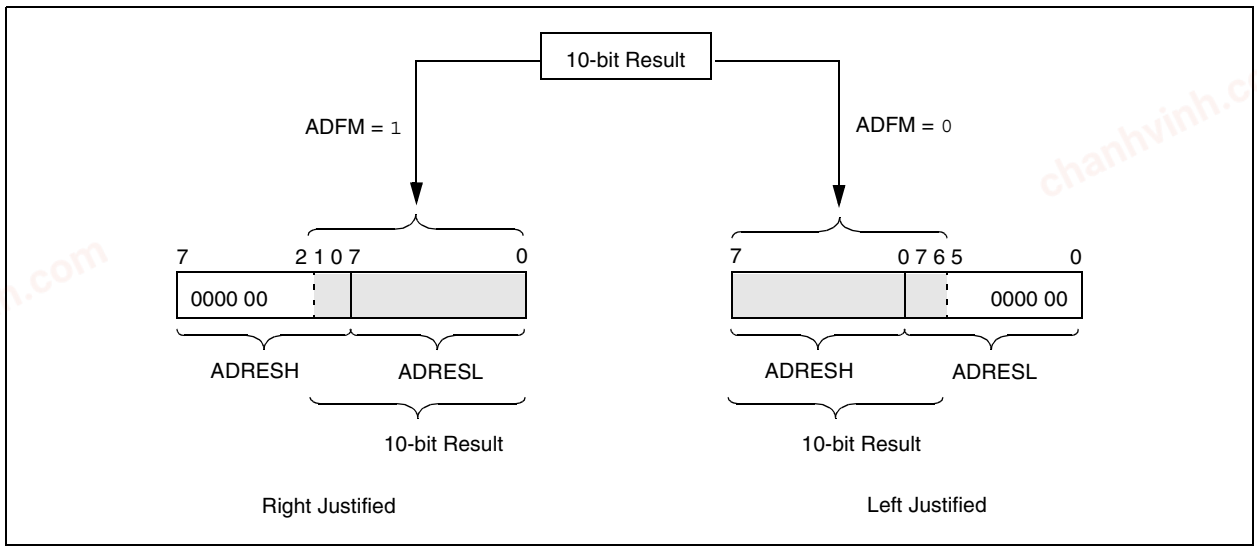
Figure 20-4 shows the operation of the A/D converter after the GO bit has been set. Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D Result register pair will not be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL 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 2 TAD wait is required before the next acquisition is started. After this 2 TAD wait, acquisition on the selected channel is automatically started.

| **Note:** | The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D. |
|---|---|

### 20.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16 bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification. Figure 20-3 shows the operation of the A/D result justification. The extra bits are loaded with '0's. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

**FIGURE 20-3: A/D RESULT JUSTIFICATION**



chanhvinh.com



## 20.5 Use of the ECCP Trigger

An A/D conversion can be started by the "special event trigger" of the ECCP module. This requires that the ECCP1M3:ECCP1M0 bits (ECCP1CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ 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/$\overline{\text{DONE}}$ 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 20-4: A/D CONVERSION TAD CYCLES**

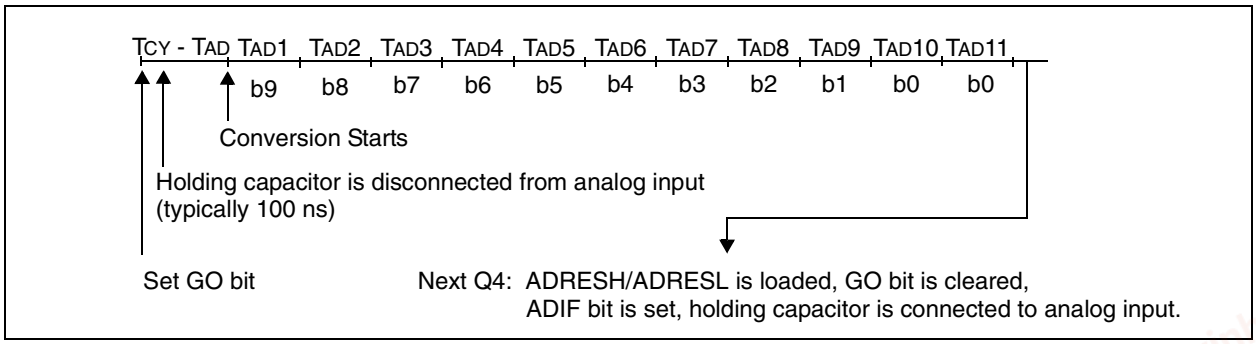



**TABLE 20-3: SUMMARY OF A/D REGISTERS**

| 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(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 1111 1111 |
| PIR2 | — | CMIF(1) | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF(1) | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE(1) | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE(1) | -0-0 0000 | -0-0 0000 |
| IPR2 | — | CMIP(1) | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP(1) | -1-1 1111 | -1-1 1111 |
| ADRESH | A/D Result Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| ADRESL | A/D Result Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| ADCON0 | ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | — | ADON | 0000 00-0 | 0000 00-0 |
| ADCON1 | ADFM | ADCS2 | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 00-- 0000 | 00-- 0000 |
| PORTA | — | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | -x0x 0000 | -u0u 0000 |
| TRISA | — | PORTA Data Direction Register | | | | | | | -111 1111 | -111 1111 |
| PORTE | — | — | — | — | — | RE2 | RE1 | RE0 | ---- -xxx | ---- -000 |
| LATE | — | — | — | — | — | LATE2 | LATE1 | LATE0 | ---- -xxx | ---- -uuu |
| TRISE | IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 | 0000 -111 | 0000 -111 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** These bits are reserved on PIC18F2X8 devices; always maintain these bits clear.



© 2006 Microchip Technology Inc.

chanhvinh.com



## 21.0 COMPARATOR MODULE

| | |
|---|---|
| **Note:** | The analog comparators are only available on the PIC18F448 and PIC18F458. |

The comparator module contains two analog comparators. The inputs to the comparators are multiplexed with the RD0 through RD3 pins. The on-chip voltage reference (**Section 22.0 "Comparator Voltage Reference Module"**) can also be an input to the comparators.

The CMCON register, shown in Register 21-1, controls the comparator input and output multiplexers. A block diagram of the comparator is shown in Figure 21-1.

**REGISTER 21-1: CMCON: COMPARATOR CONTROL 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 bit

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 bit

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 bit
1 = C2 output inverted
0 = C2 output not inverted

bit 4 **C1INV**: Comparator 1 Output Inversion bit
1 = C1 output inverted
0 = C1 output not inverted

bit 3 **CIS**: Comparator Input Switch bit

When CM2:CM0 = 110:
1 = C1 VIN- connects to RD0/PSP0
C2 VIN- connects to RD2/PSP2
0 = C1 VIN- connects to RD1/PSP1
C2 VIN- connects to RD3/PSP3

bit 2-0 **CM2:CM0**: Comparator Mode bits
Figure 21-1 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 |

chanhvinh.com

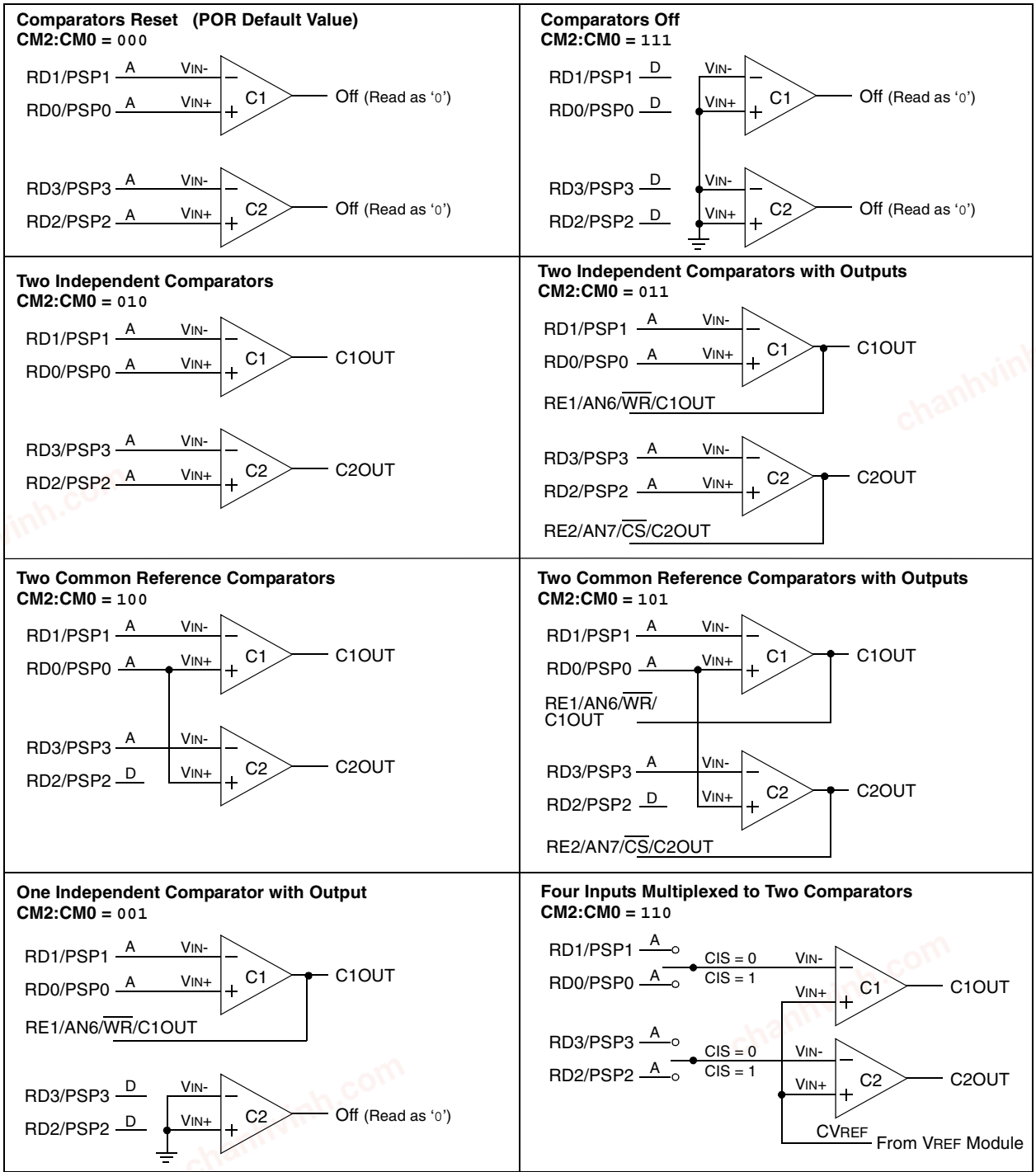


## 21.1 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select these modes. Figure 21-1 shows the eight possible modes. The TRISD 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 **Section 27.0 "Electrical Characteristics"**.

| **Note:** | Comparator interrupts should be disabled during a Comparator mode change; otherwise, a false interrupt may occur. |
|---|---|

**FIGURE 21-1: COMPARATOR I/O OPERATING MODES**




A = Analog Input, port reads zeros always
D = Digital Input
CIS (CMCON<3>) is the Comparator Input Switch


© 2006 Microchip Technology Inc.

chanhvinh.com



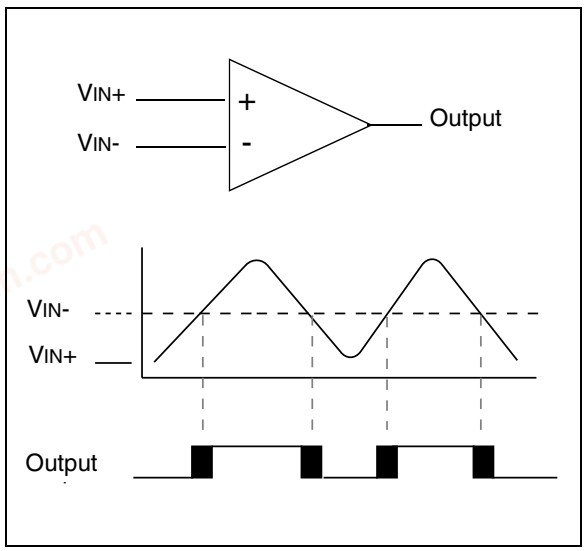
## 21.2 Comparator Operation

A single comparator is shown in Figure 21-2 along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 21-2 represent the uncertainty due to input offsets and response time.

## 21.3 Comparator Reference

An external or internal reference signal may be used depending on the comparator operating mode. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly (Figure 21-2).

**FIGURE 21-2: SINGLE COMPARATOR**




### 21.3.1 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 VSS and VDD and can be applied to either pin of the comparator(s).

### 21.3.2 INTERNAL REFERENCE SIGNAL

The comparator module also allows the selection of an internally generated voltage reference for the comparators. **Section 22.0 "Comparator Voltage Reference Module"** contains a detailed description of the module that provides this signal. The internal reference signal is used when comparators are in mode CM<2:0> = 110 (Figure 21-1). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

## 21.4 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 27.0 "Electrical Characteristics"**).

## 21.5 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 RE1 and RE2 I/O pins. When enabled, multiplexors in the output path of the RE1 and RE2 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 21-3 shows the comparator output block diagram.

The TRISE bits will still function as an output enable/disable for the RE1 and RE2 pins while in this mode.

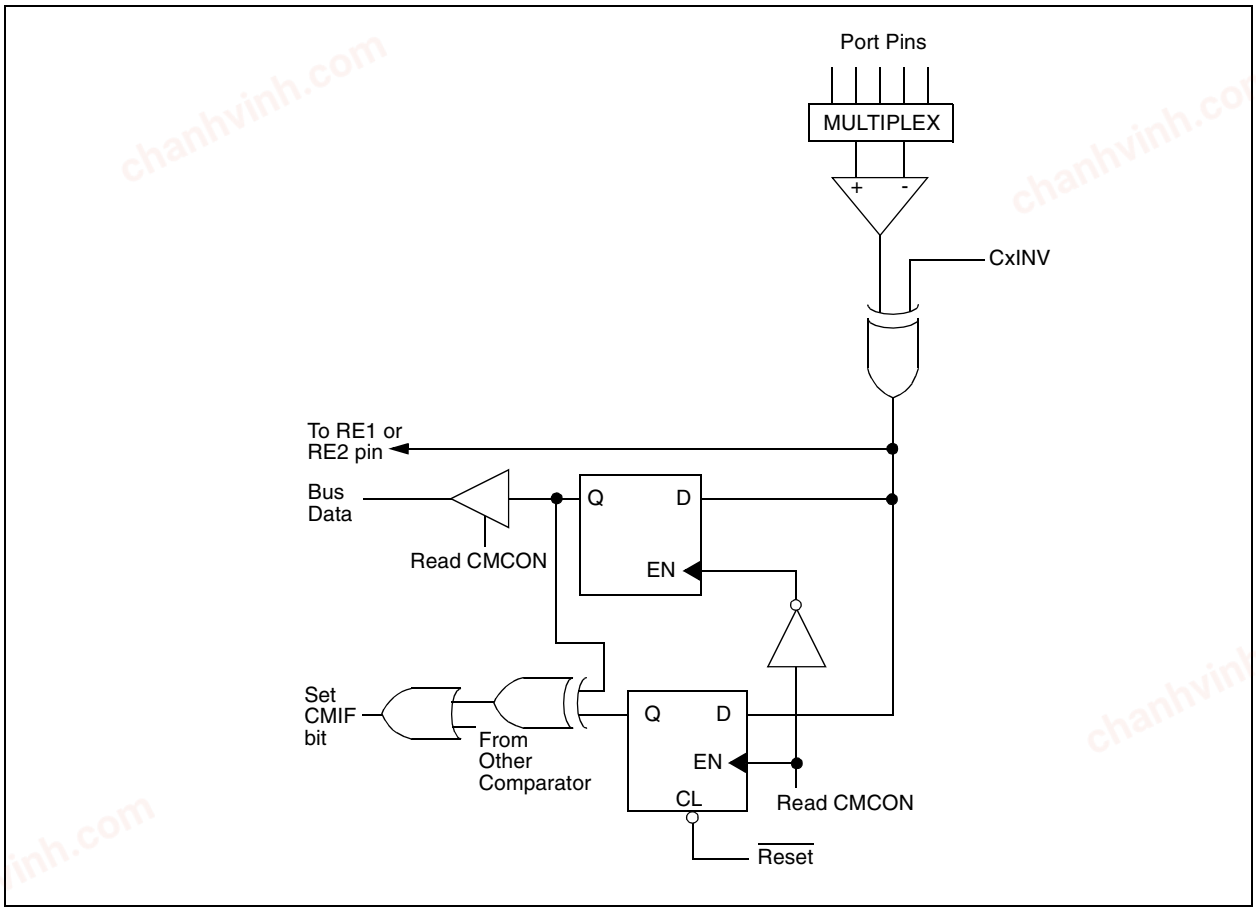
The polarity of the comparator outputs can be changed using the C2INV and C1INV bits (CMCON<4:5>).

| | |
|---|---|
| **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. |

chanhvinh.com



**FIGURE 21-3: COMPARATOR OUTPUT BLOCK DIAGRAM**




## 21.6 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 (PIR2 register) 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 (PIE2 register) 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 (PIR2 register) 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.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 21.7 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.

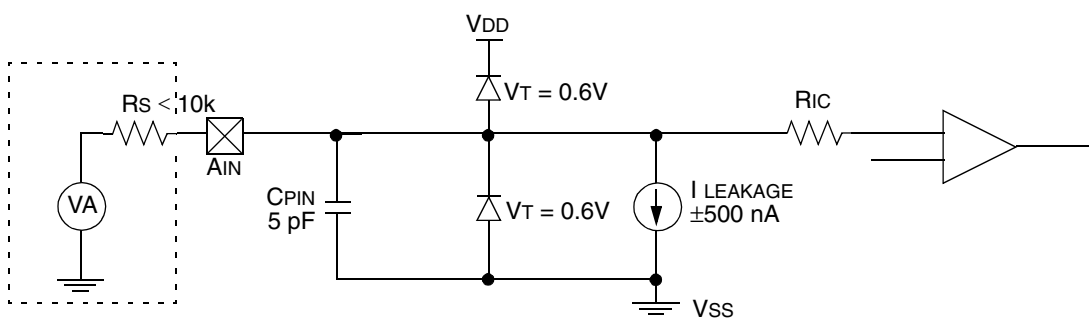
## 21.8 Effects of a Reset

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.

## 21.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 21-4. 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 0.6V 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 21-4: ANALOG INPUT MODEL**




**Legend:**

| | | |
|---|---|---|
| CPIN | = | Input Capacitance |
| VT | = | Threshold Voltage |
| I LEAKAGE | = | Leakage Current at the pin due to various junctions |
| RIC | = | Interconnect Resistance |
| RS | = | Source Impedance |
| VA | = | Analog Voltage |

chanhvinh.com



**TABLE 21-1: 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 |
| CVRCON | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 0000 0000 | 0000 0000 |
| INTCON | GIE/ GIEH | PEIE/ GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 0000 000u |
| PIR2 | — | CMIF(1) | — | EEIF | BCLIF | LVDIF | TMR3IF | ECCP1IF(1) | -0-0 0000 | -0-0 0000 |
| PIE2 | — | CMIE(1) | — | EEIE | BCLIE | LVDIE | TMR3IE | ECCP1IE(1) | -0-0 0000 | -0-0 0000 |
| IPR2 | — | CMIP(1) | — | EEIP | BCLIP | LVDIP | TMR3IP | ECCP1IP(1) | -1-1 1111 | -1-1 1111 |
| PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx xxxx | uuuu uuuu |
| LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxx xxxx | uuuu uuuu |
| TRISD | PORTD Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| PORTE | — | — | — | — | — | RE2 | RE1 | RE0 | ---- -xxx | ---- -000 |
| LATE | — | — | — | — | — | LATE2 | LATE1 | LATE0 | ---- -xxx | ---- -uuu |
| TRISE | IBF(1) | OBF(1) | IBOV(1) | PSPMODE(1) | — | TRISE2 | TRISE1 | TRISE0 | 0000 -111 | 0000 -111 |

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

**Note 1:** These bits are reserved on PIC18F2X8 devices; always maintain these bits clear.



© 2006 Microchip Technology Inc.

chanhvinh.com



## 22.0 COMPARATOR VOLTAGE REFERENCE MODULE

**Note:** The comparator voltage reference is only available on the PIC18F448 and PIC18F458.

This module 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 22-1. The block diagram is shown in Figure 22-1.

The comparator and reference supply voltage can come from either VDD and 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.

### 22.1 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.

**EQUATION 22-1:**

If CVRR = 1:
$CV_{REF} = (CVR<3:0>/24) \times CV_{RSRC}$
where:
CVRSS = 1, $CV_{RSRC} = (V_{REF+}) - (V_{REF-})$
CVRSS = 0, $CV_{RSRC} = AV_{DD} - AV_{SS}$

**EQUATION 22-2:**

If CVRR = 0:
$CV_{REF} = (CV_{RSRC} \times 1/4) + (CVR<3:0>/32) \times CV_{RSRC}$
where:
CVRSS = 1, $CV_{RSRC} = (V_{REF+}) - (V_{REF-})$
CVRSS = 0, $CV_{RSRC} = AV_{DD} - AV_{SS}$

The settling time of the Comparator Voltage Reference must be considered when changing the RA0/AN0/CVREF output (see Table 27-4 in **Section 27.2 "DC Characteristics"**).

**REGISTER 22-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL 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 |
|---|---|---|---|---|---|---|---|
| CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **CVREN**: Comparator Voltage Reference Enable bit
1 = CVREF circuit powered on
0 = CVREF circuit powered down

bit 6 **CVROE**: Comparator VREF Output Enable bit
1 = CVREF voltage level is also output on the RA0/AN0/CVREF pin
0 = CVREF voltage is disconnected from the RA0/AN0/CVREF pin

bit 5 **CVRR**: Comparator VREF Range Selection bit
1 = 0.00 CVRSRC to 0.625 CVRSRC with CVRSRC/24 step size
0 = 0.25 CVRSRC to 0.719 CVRSRC with CVRSRC/32 step size

bit 4 **CVRSS**: Comparator VREF Source Selection bit
1 = Comparator reference source, $CV_{RSRC} = (V_{REF+}) - (V_{REF-})$
0 = Comparator reference source, $CV_{RSRC} = V_{DD} - V_{SS}$

bit 3-0 **CVR<3:0>:** Comparator VREF Value Selection 0 ≤ CVR3:CVR0 ≤ 15 bits
When CVRR = 1:
$CV_{REF} = (CVR3:CVR0/24) \bullet (CV_{RSRC})$
When CVRR = 0:
$CV_{REF} = 1/4 \bullet (CV_{RSRC}) + (CVR3:CVR0/32) \bullet (CV_{RSRC})$

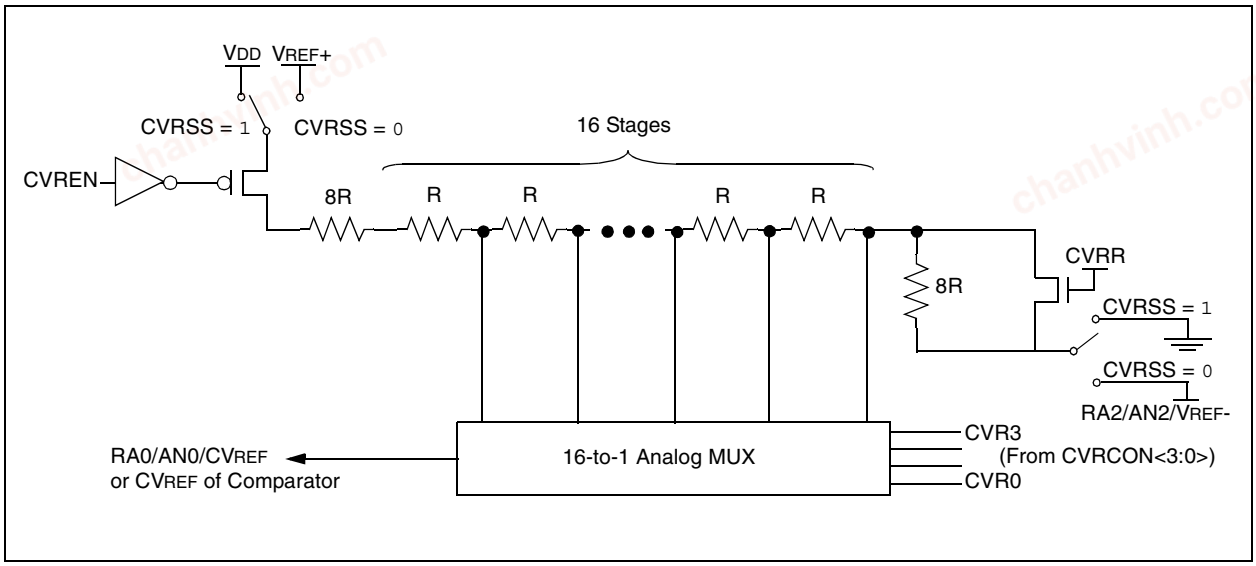
| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 22-1: VOLTAGE REFERENCE BLOCK DIAGRAM**




## 22.2 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 22-1) 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 absolute accuracy of the voltage reference can be found in **Section 27.0 "Electrical Characteristics"**.

## 22.3 Operation During Sleep

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

## 22.4 Effects of a Reset

A device Reset disables the voltage reference by clearing bit CVREN (CVRCON register). This Reset also disconnects the reference from the RA2 pin by clearing bit CVROE (CVRCON register) and selects the high-voltage range by clearing bit CVRR (CVRCON register). The CVRSS value select bits, CVRCON<3:0>, are also cleared.

## 22.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RA0/AN0 pin if the TRISA<0> bit is set and the CVROE bit (CVRCON<6>) is set. Enabling the voltage reference output onto the RA0/AN0 pin, with an input signal present, will increase current consumption. Connecting RA0/AN0 as a digital output, with CVRSS enabled, will also increase current consumption.

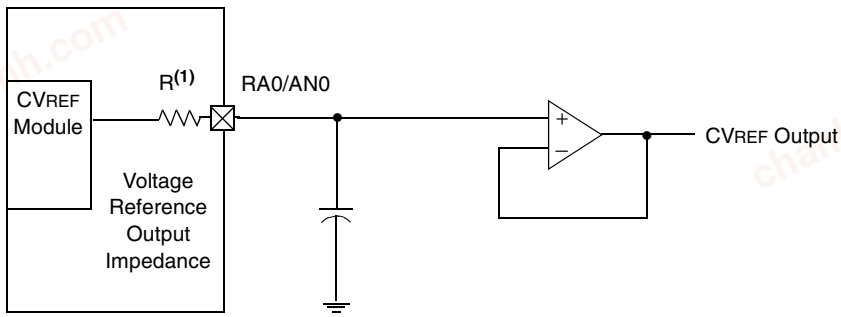
The RA0/AN0 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 22-2 shows an example buffering technique.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 22-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE**




**Note 1:** R is dependent upon the voltage reference configuration CVRCON<3:0> and CVRCON<5>.

**TABLE 22-1: 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 |
|---|---|---|---|---|---|---|---|---|---|---|
| CVRCON | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 0000 0000 | 0000 0000 |
| CMCON | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 0000 0000 | 0000 0000 |
| TRISA | — | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | -111 1111 | -111 1111 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, read as '0'.
Shaded cells are not used with the comparator voltage reference.

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 23.0 LOW-VOLTAGE DETECT

In many applications, the ability to determine if the device voltage ($V_{DD}$) is below a specified voltage level is a desirable feature. A window of operation for the application can be created, where the application software can do “housekeeping tasks” before the device voltage exits the valid operating range. This can be done using the Low-Voltage Detect module.

This module is a software programmable circuitry, where a device voltage trip point can be specified. 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.

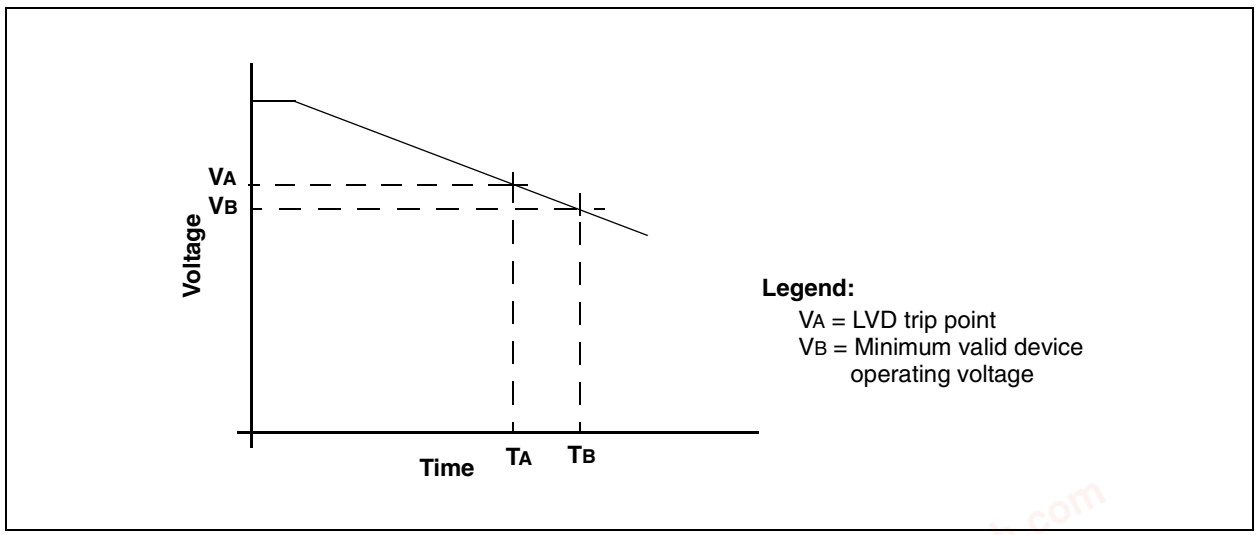
The Low-Voltage Detect circuitry is completely under software control. This allows the circuitry to be “turned off” by the software which minimizes the current consumption for the device.

Figure 23-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage $V_A$, the LVD logic generates an interrupt. This occurs at time $T_A$. The application software then has the time, until the device voltage is no longer in valid operating range, to shutdown the system. Voltage point $V_B$ is the minimum valid operating voltage specification. This occurs at time $T_B$. The difference $T_B - T_A$ is the total time for shutdown.

The block diagram for the LVD module is shown in Figure 23-2. 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 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 is equal to the internal reference voltage generated by the 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 23-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

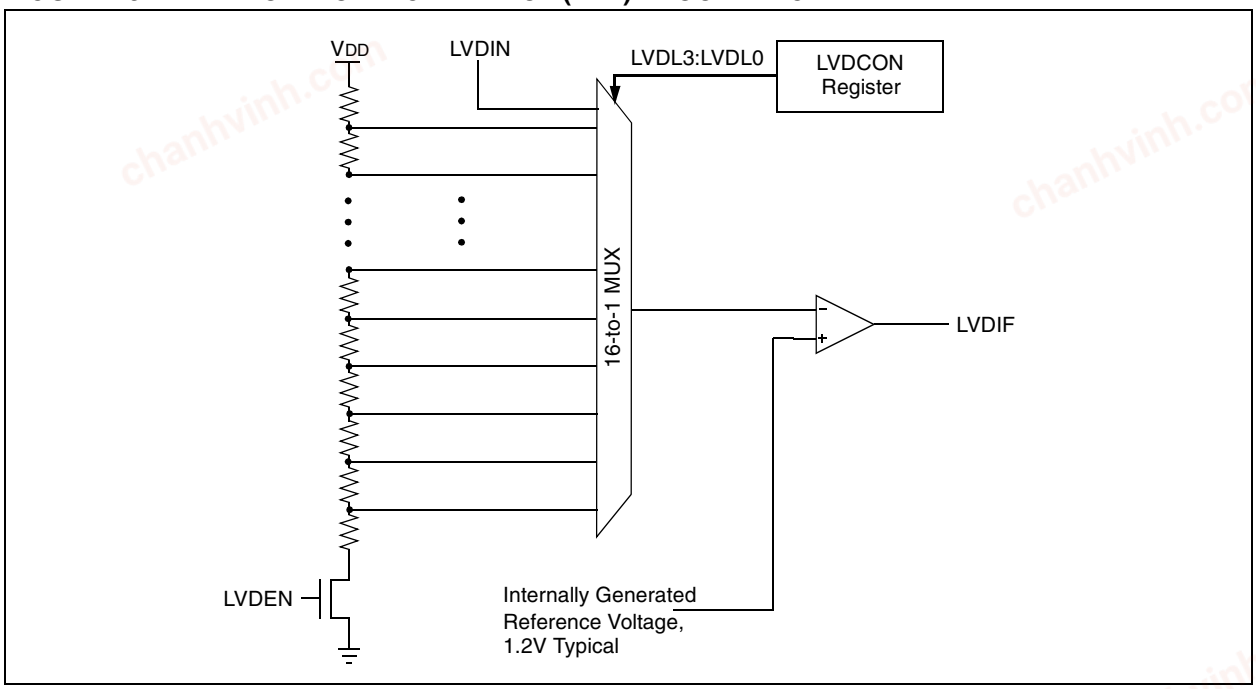
**FIGURE 23-1: TYPICAL LOW-VOLTAGE DETECT APPLICATION**



chanhvinh.com



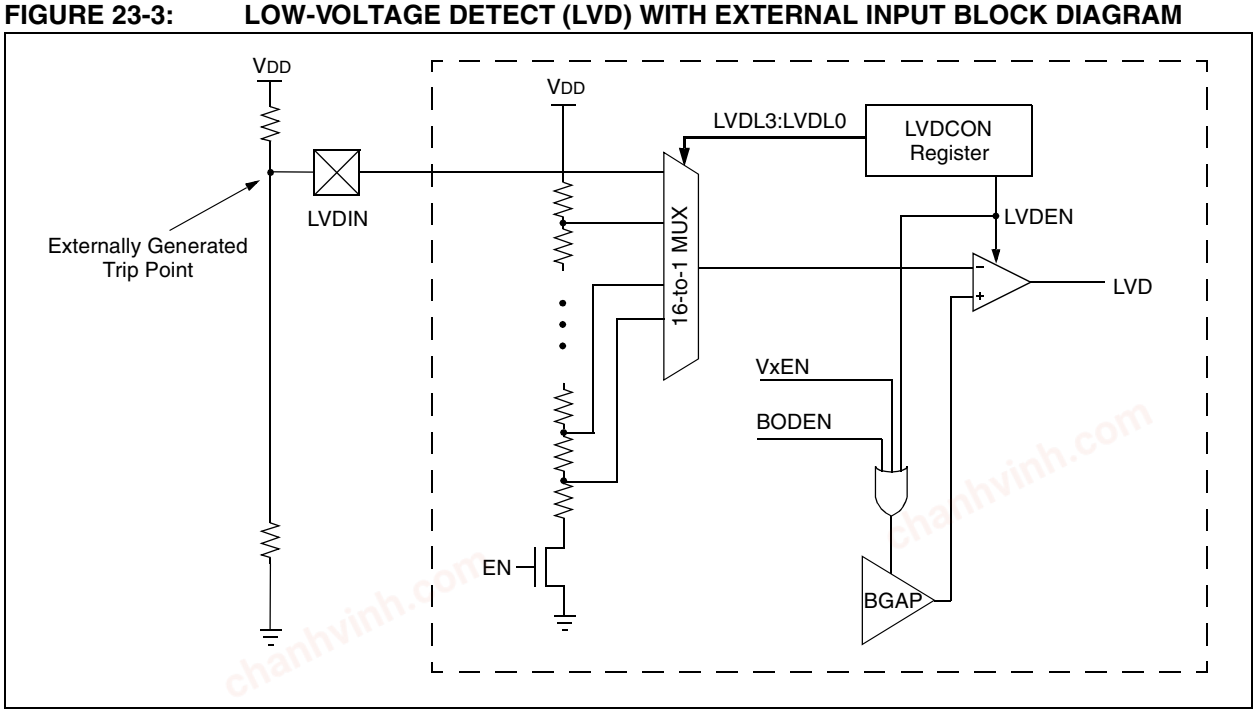
**FIGURE 23-2: LOW-VOLTAGE DETECT (LVD) BLOCK DIAGRAM**




The LVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits LVDL3:LVDL0 are set to '1111'. In this state, the comparator input is multiplexed from the external input pin LVDIN to one input of the comparator (Figure 23-3).

The other input is connected to the internally generated voltage reference (parameter #D423 in **Section 27.2 "DC Characteristics"**). This gives users flexibility, because it allows them to configure the Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

**FIGURE 23-3: LOW-VOLTAGE DETECT (LVD) WITH EXTERNAL INPUT BLOCK DIAGRAM**





© 2006 Microchip Technology Inc.

chanhvinh.com



## 23.1 Control Register

The Low-Voltage Detect Control register controls the operation of the Low Voltage Detect circuitry.

**REGISTER 23-1: LVDCON: LOW-VOLTAGE DETECT CONTROL 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 = 4.45V min.-4.83V max.
1101 = 4.16V min.-4.5V max.
1100 = 3.96V min.-4.2V max.
1011 = 3.76V min.-4.08V max.
1010 = 3.57V min.-3.87V max.
1001 = 3.47V min.-3.75V max.
1000 = 3.27V min.-3.55V max.
0111 = 2.98V min.-3.22V max.
0110 = 2.77V min.-3.01V max.
0101 = 2.67V min.-2.89V max.
0100 = 2.48V min.-2.68V max.
0011 = 2.37V min.-2.57V max.
0010 = 2.18V min.-2.36V max.
0001 = 1.98V min.-2.14V max.
0000 = Reserved

**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 |

© 2006 Microchip Technology Inc.

chanhvinh.com



## 23.2 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 the current requirements, 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.

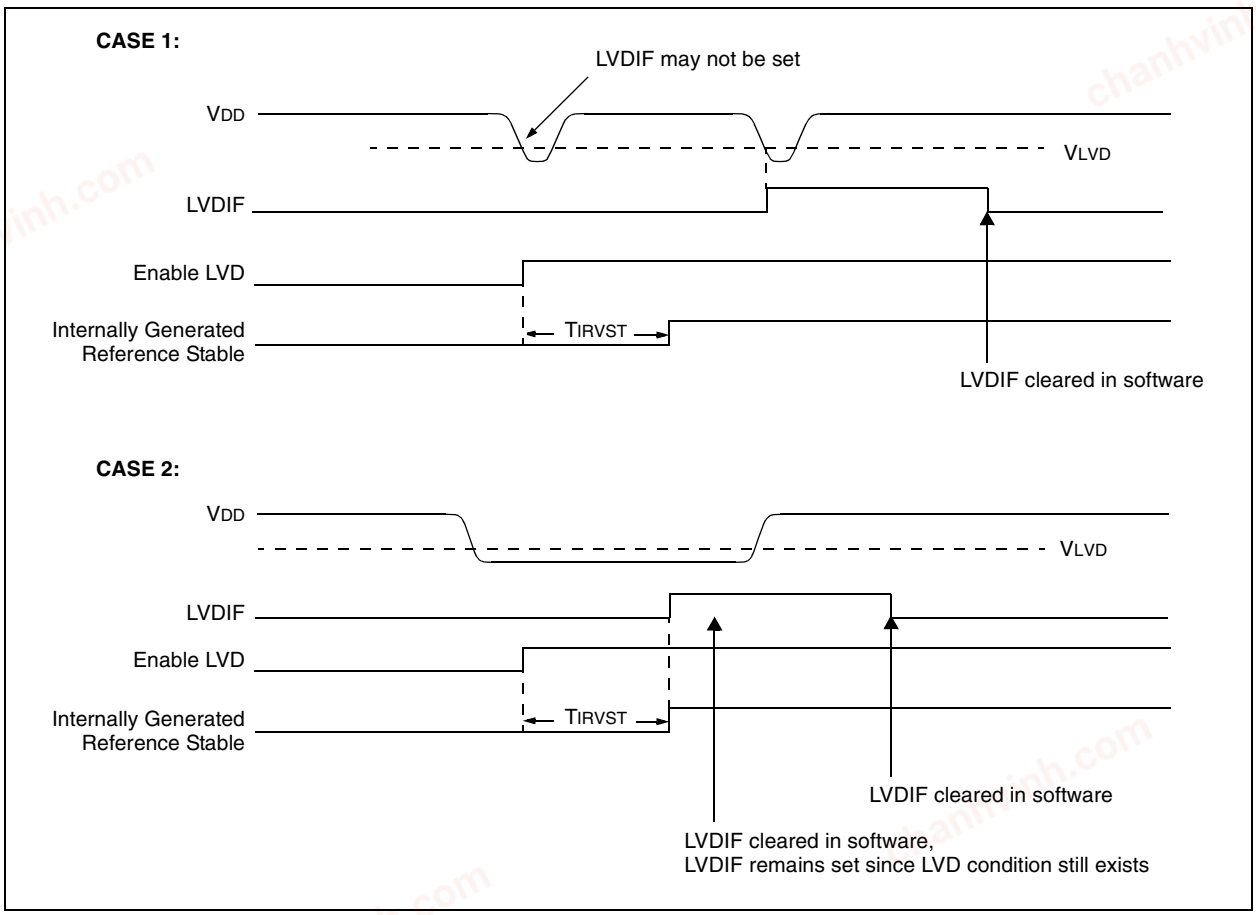
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

The following steps are needed to set up 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 23-4 shows typical waveforms that the LVD module may be used to detect.

**FIGURE 23-4: LOW-VOLTAGE DETECT WAVEFORMS**





© 2006 Microchip Technology Inc.

chanhvinh.com



#### 23.2.1 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 a 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 23-4.

#### 23.2.2 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.

### 23.3 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.

### 23.4 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the LVD module to be turned off.

chanhvinh.com

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 24.0 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. These are:

- Oscillator Selection
- Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Sleep
- Code Protection
- ID Locations
- In-Circuit Serial Programming

All PIC18FXX8 devices have a Watchdog Timer which is permanently enabled via the configuration bits or 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 is used to select various options.

### 24.1 Configuration Bits

The configuration bits can be programmed (read as '0') or left unprogrammed (read as '1'), to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh) which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The EECON1 register $\overline{WR}$ bit starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction, with the TBLPTR pointed to the Configuration register, sets up the address and the data for the Configuration register write. Setting the $\overline{WR}$ bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a '1' or a '0' into the cell.

**TABLE 24-1: CONFIGURATION BITS AND DEVICE IDS**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Default/ Unprogrammed Value |
|---|---|---|---|---|---|---|---|---|---|---|
| 300001h | CONFIG1H | — | — | $\overline{OSCSEN}$ | — | — | FOSC2 | FOSC1 | FOSC0 | --1- -111 |
| 300002h | CONFIG2L | — | — | — | — | BORV1 | BORV0 | BOREN | $\overline{PWRTEN}$ | ---- 1111 |
| 300003h | CONFIG2H | — | — | — | — | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN | ---- 1111 |
| 300006h | CONFIG4L | $\overline{DEBUG}$ | — | — | — | — | LVP | — | STVREN | 1--- -1-1 |
| 300008h | CONFIG5L | — | — | — | — | CP3 | CP2 | CP1 | CP0 | ---- 1111 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — | 11-- ---- |
| 30000Ah | CONFIG6L | — | — | — | — | WRT3 | WRT2 | WRT1 | WRT0 | ---- 1111 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — | 111- ---- |
| 30000Ch | CONFIG7L | — | — | — | — | EBTR3 | EBTR2 | EBTR1 | EBTR0 | ---- 1111 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — | -1-- ---- |
| 3FFFFEh | DEVID1 | DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | (1) |
| 3FFFFFh | DEVID2 | DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | 0000 1000 |

**Legend:** x = unknown, u = unchanged, - = unimplemented, q = value depends on condition. Shaded cells are unimplemented, read as '0'.
**Note 1:** See Register 24-11 for DEVID1 values.

© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 24-1: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)**

| U-0 | U-0 | R/P-1 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | $\overline{\text{OSCSEN}}$ | — | — | FOSC2 | FOSC1 | FOSC0 |
| bit 7 | | | | | | | bit 0 |

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

bit 5 **$\overline{\text{OSCSEN}}$**: 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 = HS 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 24-2: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)**

| U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | BORV1 | BORV0 | BOREN | $\overline{\text{PWRTEN}}$ |
| 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 2.0V
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 = Brown-out Reset enabled
0 = Brown-out Reset disabled

bit 0 **$\overline{\text{PWRTEN}}$:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT 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 |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 24-3: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)**

| U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN |
| bit 7 | | | | | | | bit 0 |

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

bit 3-1 **WDTPS2:WDTPS0:** Watchdog Timer Postscale Select bits

111 = 1:128
110 = 1:64
101 = 1:32
100 = 1:16
011 = 1:8
010 = 1:4
001 = 1:2
000 = 1:1

**Note:** The Watchdog Timer postscale select bits configuration used in the PIC18FXXX devices has changed from the configuration used in the PIC18CXXX devices.

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 |

**REGISTER 24-4: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)**

| R/P-1 | U-0 | U-0 | U-0 | U-0 | R/P-1 | U-0 | R/P-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{DEBUG}}$ | — | — | — | — | LVP | — | STVREN |
| bit 7 | | | | | | | bit 0 |

bit 7 **$\overline{\text{DEBUG}}$:** Background Debugger Enable bit

1 = Background Debugger disabled. RB6 and RB7 configured as general purpose I/O pins.
0 = Background Debugger enabled. RB6 and RB7 are dedicated to In-Circuit Debug.

bit 6-3 **Unimplemented:** Read as '0'

bit 2 **LVP:** Low-Voltage ICSP Enable bit

1 = Low-Voltage ICSP enabled
0 = Low-Voltage ICSP disabled

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

bit 0 **STVREN:** Stack Full/Underflow Reset Enable bit

1 = Stack Full/Underflow will cause Reset
0 = Stack Full/Underflow will not cause Reset

| **Legend:** | | |
|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

chanhvinh.com



**REGISTER 24-5: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)**

| U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | CP3(1) | CP2(1) | CP1 | CP0 |
| bit 7 | | | | | | | bit 0 |

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

bit 3 **CP3:** Code Protection bit(1)
1 = Block 3 (006000-007FFFh) not code-protected
0 = Block 3 (006000-007FFFh) code-protected

bit 2 **CP2:** Code Protection bit(1)
1 = Block 2 (004000-005FFFh) not code-protected
0 = Block 2 (004000-005FFFh) code-protected

bit 1 **CP1:** Code Protection bit
1 = Block 1 (002000-003FFFh) not code-protected
0 = Block 1 (002000-003FFFh) code-protected

bit 0 **CP0:** Code Protection bit
1 = Block 0 (000200-001FFFh) not code-protected
0 = Block 0 (000200-001FFFh) code-protected

**Note 1:** Unimplemented in PIC18FX48 devices; maintain this bit set.

| **Legend:** | | |
|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

**REGISTER 24-6: CONFIG5H: CONFIGURATION REGISTER 5 HIGH (BYTE ADDRESS 300009h)**

| R/C-1 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| CPD | CPB | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

bit 7 **CPD:** Data EEPROM Code Protection bit
1 = Data EEPROM not code-protected
0 = Data EEPROM code-protected

bit 6 **CPB:** Boot Block Code Protection bit
1 = Boot Block (000000-0001FFh) not code-protected
0 = Boot Block (000000-0001FFh) code-protected

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

| **Legend:** | | |
|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 24-7: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)**

| U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | WRT3(1) | WRT2(1) | WRT1 | WRT0 |
| bit 7 | | | | | | | bit 0 |

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

bit 3 **WRT3:** Write Protection bit(1)
1 = Block 3 (006000-007FFFh) not write-protected
0 = Block 3 (006000-007FFFh) write-protected

bit 2 **WRT2:** Write Protection bit(1)
1 = Block 2 (004000-005FFFh) not write-protected
0 = Block 2 (004000-005FFFh) write-protected

bit 1 **WRT1:** Write Protection bit
1 = Block 1 (002000-003FFFh) not write-protected
0 = Block 1 (002000-003FFFh) write-protected

bit 0 **WRT0:** Write Protection bit
1 = Block 0 (000200-001FFFh) not write-protected
0 = Block 0 (000200-001FFFh) write-protected

**Note 1:** Unimplemented in PIC18FX48 devices; maintain this bit set.

| **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 24-8: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)**

| R/P-1 | R/P-1 | R-1 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| WRTD | WRTB | WRTC | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

bit 7 **WRTD:** Data EEPROM Write Protection bit
1 = Data EEPROM not write-protected
0 = Data EEPROM write-protected

bit 6 **WRTB:** Boot Block Write Protection bit
1 = Boot Block (000000-0001FFh) not write-protected
0 = Boot Block (000000-0001FFh) write-protected

bit 5 **WRTC:** Configuration Register Write Protection bit
1 = Configuration registers (300000-3000FFh) not write-protected
0 = Configuration registers (300000-3000FFh) write-protected

**Note:** This bit is read-only and cannot be changed in user mode.

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

| **Legend:** | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

chanhvinh.com



**REGISTER 24-9: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)**

| U-0 | U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | EBTR3(1) | EBTR2(1) | EBTR1 | EBTR0 |
| bit 7 | | | | | | | bit 0 |

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

bit 3 **EBTR3:** Table Read Protection bit(1)
1 = Block 3 (006000-007FFFh) not protected from table reads executed in other blocks
0 = Block 3 (006000-007FFFh) protected from table reads executed in other blocks

bit 2 **EBTR2:** Table Read Protection bit(1)
1 = Block 2 (004000-005FFFh) not protected from table reads executed in other blocks
0 = Block 2 (004000-005FFFh) protected from table reads executed in other blocks

bit 1 **EBTR1:** Table Read Protection bit
1 = Block 1 (002000-003FFFh) not protected from table reads executed in other blocks
0 = Block 1 (002000-003FFFh) protected from table reads executed in other blocks

bit 0 **EBTR0:** Table Read Protection bit
1 = Block 0 (000200-001FFFh) not protected from table reads executed in other blocks
0 = Block 0 (000200-001FFFh) protected from table reads executed in other blocks

**Note 1:** Unimplemented in PIC18FX48 devices; maintain this bit set.

| **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 24-10: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)**

| U-0 | R/P-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | EBTRB | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

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

bit 6 **EBTRB:** Boot Block Table Read Protection bit
1 = Boot Block (000000-0001FFh) not protected from table reads executed in other blocks
0 = Boot Block (000000-0001FFh) protected from table reads executed in other blocks

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

| **Legend:** | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |


© 2006 Microchip Technology Inc.

chanhvinh.com



**REGISTER 24-11: DEVID1: DEVICE ID REGISTER 1 FOR PIC18FXX8 DEVICES (BYTE ADDRESS 3FFFFEh)**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| 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 DEV<10:3> bits in the Device ID Register 2 to identify the part number.

000 = PIC18F248
001 = PIC18F448
010 = PIC18F258
011 = PIC18F458

bit 4-0 **REV4:REV0:** Revision ID bits

These bits are used to indicate the device revision.

| **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 24-12: DEVID2: DEVICE ID REGISTER 2 FOR PIC18FXX8 DEVICES (BYTE ADDRESS 3FFFFFh)**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| 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 DEV<2:0> bits in the Device ID Register 1 to identify the part number.

00001000 = PIC18FXX8

| **Legend:** | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

chanhvinh.com



## 24.2 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 WDTEN configuration bit is cleared, the SWDTEN bit enables/disables the operation of the WDT.

The WDT time-out period values may be found in **Section 27.0 "Electrical Characteristics"** 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 postscaler is assigned to the WDT, the postscaler count will be cleared but the postscaler assignment is not changed. |
|---|---|

### 24.2.1 CONTROL REGISTER

Register 24-13 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 24-13: WDTCON: WATCHDOG TIMER CONTROL 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 WDTEN configuration bit in the Configuration register = 0

| **Legend:** | |
|---|---|
| R = Readable bit | W = Writable bit |
| U = Unimplemented bit, read as '0' | -n = Value at POR |


© 2006 Microchip Technology Inc.

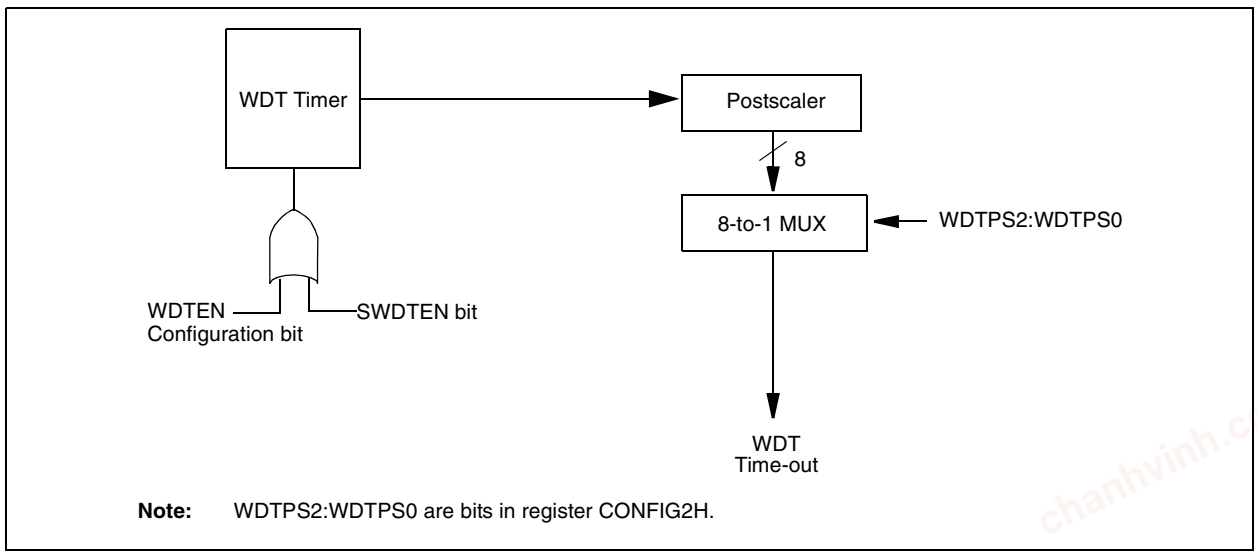
chanhvinh.com



### 24.2.2 WDT POSTSCALER

The WDT has a postscaler that can extend the WDT Reset period. The postscaler is selected at the time of device programming by the value written to the CONFIG2H Configuration register.

**FIGURE 24-1: WATCHDOG TIMER BLOCK DIAGRAM**




**Note:** WDTPS2:WDTPS0 are bits in register CONFIG2H.

**TABLE 24-2: SUMMARY OF WATCHDOG TIMER REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| CONFIG2H | — | — | — | — | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN |
| RCON | IPEN | — | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| WDTCON | — | — | — | — | — | — | — | SWDTEN |

**Legend:** Shaded cells are not used by the Watchdog Timer.

chanhvinh.com



## 24.3 Power-Down Mode (Sleep)

Power-down mode is entered by executing a SLEEP instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the $\overline{PD}$ bit (RCON<2>) is cleared, the $\overline{TO}$ bit (RCON<3>) is set and the oscillator driver is turned off. The I/O ports maintain the status they had before the SLEEP instruction was executed (driving high, low or high-impedance).

For lowest current consumption in this mode, place all I/O pins at either VDD or VSS, ensure no external circuitry is drawing current from the I/O pin, power-down the A/D and disable external clocks. Pull all I/O pins that are high-impedance inputs, high or low externally, to avoid switching currents caused by floating inputs. The T0CKI input should also be at VDD or VSS for lowest current consumption. The contribution from on-chip pull-ups on PORTB should be considered.

The $\overline{MCLR}$ pin must be at a logic high level (VIHMC).

### 24.3.1 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/$I^2C$).
8. USART RX or TX (Synchronous Slave mode).
9. A/D conversion (when A/D clock source is RC).
10. EEPROM write operation complete.
11. LVD interrupt.

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 prefetched. 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.

### 24.3.2 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 the $\overline{PD}$ bit 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.

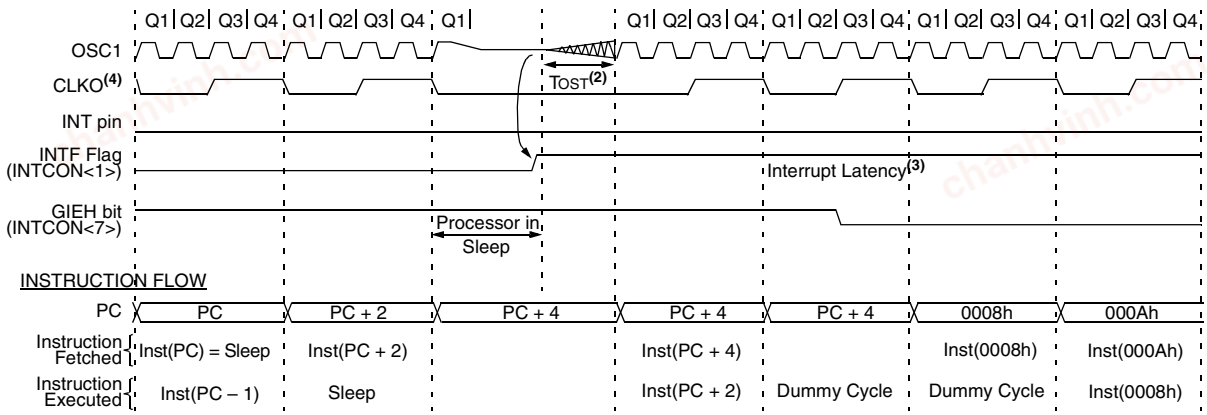
To ensure that the WDT is cleared, a CLRWDT instruction should be executed before a SLEEP instruction.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 24-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT(1,2)**




**Note 1:** XT, HS or LP Oscillator mode assumed.
**2:** GIE = 1 assumed. In this case, after wake-up, the processor jumps to the interrupt routine. If GIE = 0, execution will continue in-line.
**3:** TOST = 1024 TOSC (drawing not to scale). This delay will not occur for RC and EC Oscillator modes.
**4:** CLKO is not available in these oscillator modes but shown here for timing reference.

chanhvinh.com

chanhvinh.com



## 24.4 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PICmicro devices.

The user program memory is divided into five blocks. One of these is a boot block of 512 bytes. The remainder of the memory is divided into four blocks on binary boundaries.

Each of the five blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 24-3 shows the program memory organization for 16 and 32-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 24-3.

**FIGURE 24-3: CODE-PROTECTED PROGRAM MEMORY FOR PIC18FXX8**

| MEMORY SIZE/DEVICE | | | |
|---|---|---|---|
| 16 Kbytes (PIC18FX48) | 32 Kbytes (PIC18FX58) | Address Range | Block Code Protection Controlled By: |
| Boot Block | Boot Block | 000000h<br>0001FFh | CPB, WRTB, EBTRB |
| Block 0 | Block 0 | 000200h<br>001FFFh | CP0, WRT0, EBTR0 |
| Block 1 | Block 1 | 002000h<br>003FFFh | CP1, WRT1, EBTR1 |
| Unimplemented Read '0's | Block 2 | 004000h<br>005FFFh | CP2, WRT2, EBTR2 |
| Unimplemented Read '0's | Block 3 | 006000h<br>007FFFh | CP3, WRT3, EBTR3 |
| Unimplemented Read '0's | Unimplemented Read '0's | 008000h<br>1FFFFFh | (Unimplemented Memory Space) |

**TABLE 24-3: SUMMARY OF CODE PROTECTION REGISTERS**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 300008h | CONFIG5L | — | — | — | — | CP3 | CP2 | CP1 | CP0 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — |
| 30000Ah | CONFIG6L | — | — | — | — | WRT3 | WRT2 | WRT1 | WRT0 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — |
| 30000Ch | CONFIG7L | — | — | — | — | EBTR3 | EBTR2 | EBTR1 | EBTR0 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — |

**Legend:** Shaded cells are unimplemented.


© 2006 Microchip Technology Inc.

chanhvinh.com



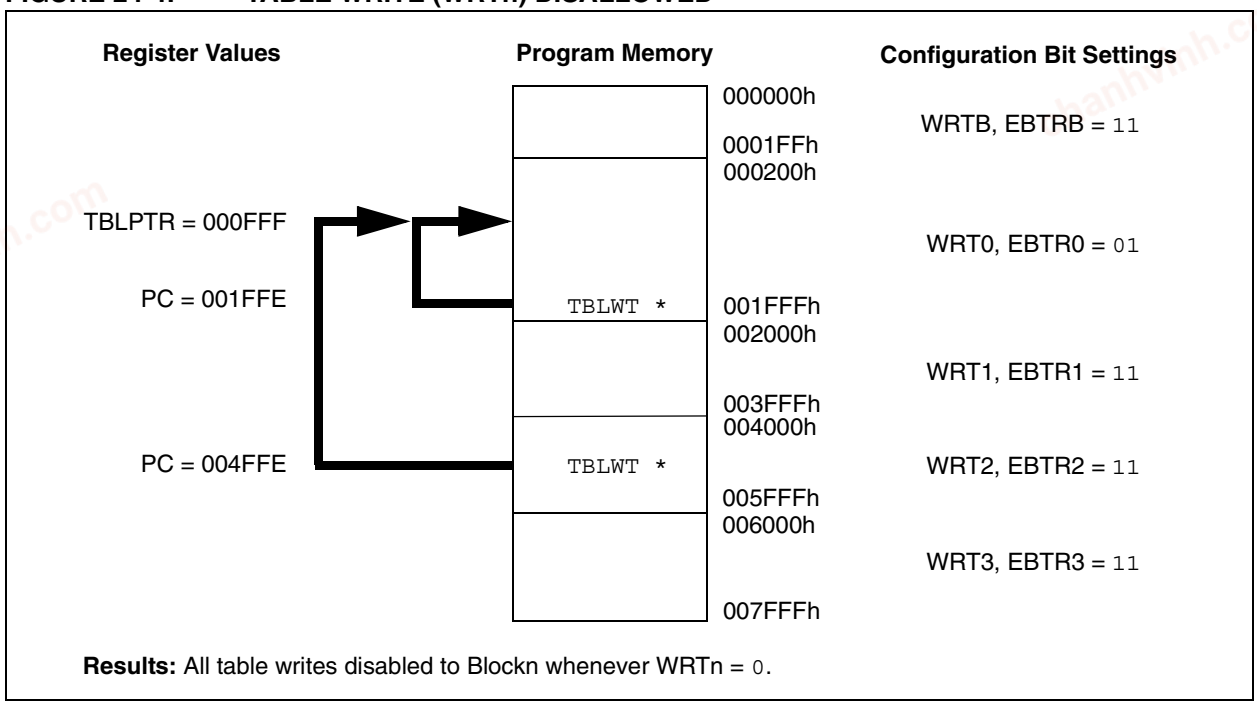
### 24.4.1 PROGRAM MEMORY CODE PROTECTION

The user memory may be read to or written from any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In user mode, the CPn bits have no direct effect. CPn bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTn configuration bit is '0'. The EBTRn bits control table reads. For a block of user memory with the EBTRn bit set to '0', a table read instruction that executes from within that block is allowed to read. A table read instruction that executes from a location outside of that block is not allowed to read and will result in reading '0's. Figures 24-4 through 24-6 illustrate table write and table read protection.

| **Note:** | Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP or an external programmer. |
|---|---|

**FIGURE 24-4: TABLE WRITE (WRTn) DISALLOWED**

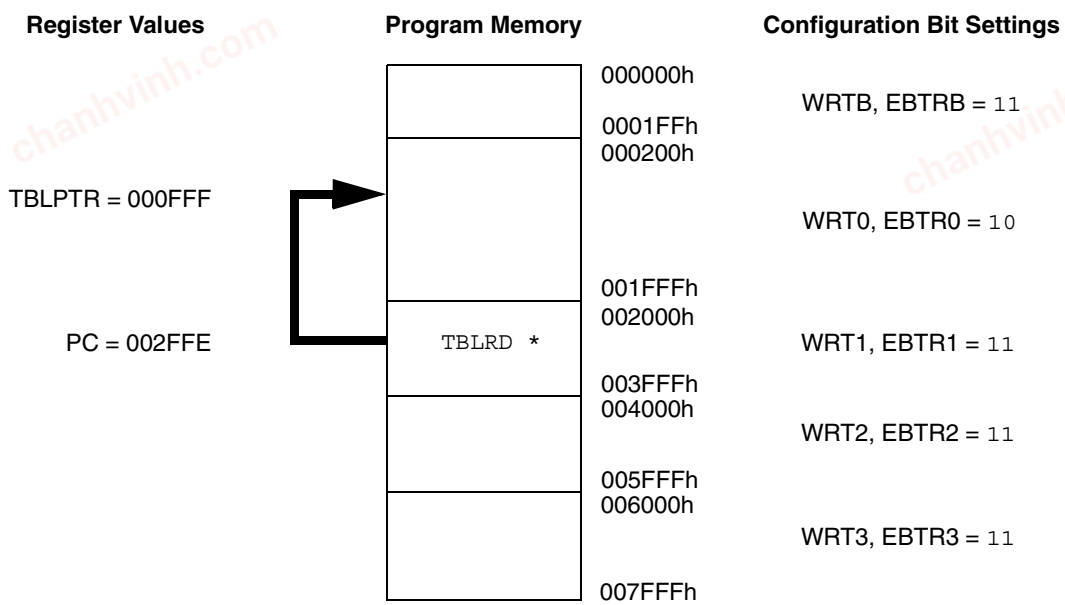



**Results:** All table writes disabled to Blockn whenever WRTn = 0.

chanhvinh.com

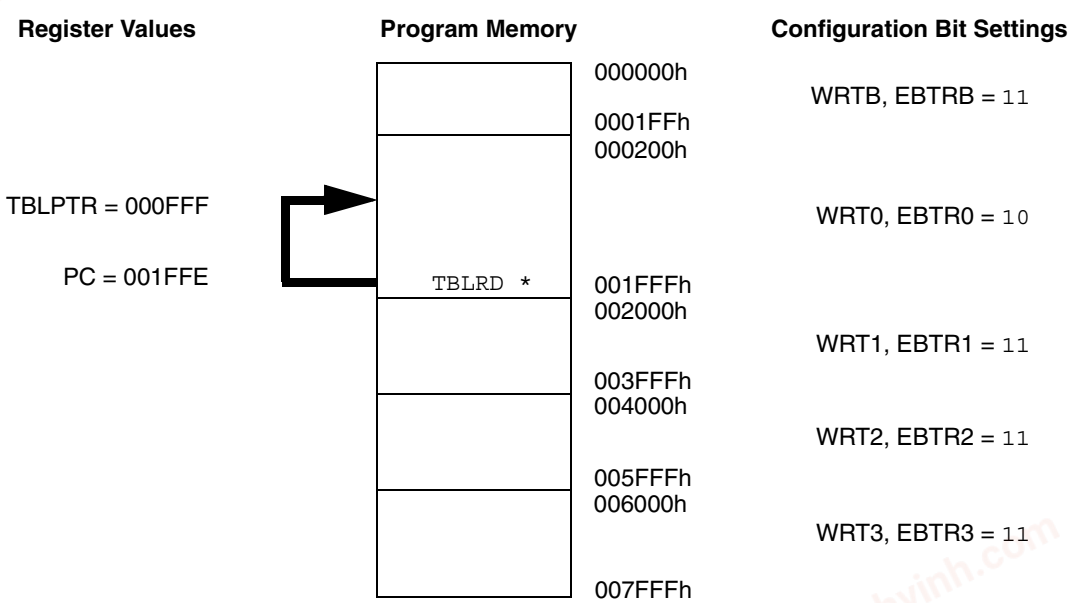


**FIGURE 24-5: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED**




**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRn = 0.
TABLAT register returns a value of '0'.

**FIGURE 24-6: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED**




**Results:** Table reads permitted within Blockn even when EBTRBn = 0.
TABLAT register returns the value of the data at the location TBLPTR.


© 2006 Microchip Technology Inc.

chanhvinh.com



### 24.4.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits external writes to data EEPROM. The CPU can continue to read and write data EEPROM regardless of the protection bit settings.

### 24.4.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In user mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

## 24.5 ID Locations

Eight memory locations (200000h-200007h) 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 and TBLWT instructions or during program/verify. The ID locations can be read when the device is code-protected.

## 24.6 In-Circuit Serial Programming

PIC18FXXX 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.

## 24.7 In-Circuit Debugger

When the DEBUG bit in Configuration register, CONFIG4L, is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some of the resources are not available for general use. Resources used include 2 I/O pins, stack locations, program memory and data memory. For more information on the resources required, see the User's Guide for the In-Circuit Debugger you are using.

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to $\overline{MCLR}/V_{PP}$, $V_{DD}$, GND, RB7 and RB6. This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies. The Microchip In-Circuit Debugger (ICD) used with the PIC18FXXX microcontrollers is the MPLAB® ICD 2.

## 24.8 Low-Voltage ICSP Programming

The LVP bit in Configuration register, CONFIG4L, enables Low-Voltage ICSP Programming. This mode allows the microcontroller to be programmed via ICSP using a $V_{DD}$ source in the operating voltage range. This only means that $V_{PP}$ does not have to be brought to $V_{IHH}$ but can instead be left at the normal operating voltage. In this mode, the RB5/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. During programming, $V_{DD}$ is applied to the $\overline{MCLR}/V_{PP}$ pin. To enter Programming mode, $V_{DD}$ must be applied to the RB5/PGM pin, provided the LVP bit is set. The LVP bit defaults to a ('1') from the factory.

| | |
|---|---|
| **Note 1:** | The High-Voltage Programming mode is always available, regardless of the state of the LVP bit, by applying $V_{IHH}$ to the $\overline{MCLR}$ pin. |
| **2:** | While in Low-Voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O pin. |
| **3:** | When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 5 in the TRISB register must be cleared to disable the pull-up on RB5 and ensure the proper operation of the device. |

If Low-Voltage Programming mode is not used, the LVP bit can be programmed to a '0' and RB5/PGM becomes a digital I/O pin. However, the LVP bit may only be programmed when programming is entered with $V_{IHH}$ on $\overline{MCLR}/V_{PP}$. The LVP bit can only be charged when using high voltage on $\overline{MCLR}$.

It should be noted that once the LVP bit is programmed to '0', only the High-Voltage Programming mode is available and only High-Voltage Programming mode can be used to program the device.

When using Low-Voltage ICSP Programming, the part must be supplied 4.5V to 5.5V if a bulk erase will be executed. This includes reprogramming of the code-protect bits from an ON state to an OFF state. For all other cases of Low-Voltage ICSP Programming, the part may be programmed at the normal operating voltage. This means unique user IDs or user code can be reprogrammed or added.

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 25.0 INSTRUCTION SET SUMMARY

The PIC18 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 PIC18 instruction set summary in Table 25-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 25-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The destination of the result (specified by ‘d’)
3. The accessed memory (specified by ‘a’)

The file register designator ‘f’ specifies which file register is to be used by the instruction.

The destination designator ‘d’ specifies where the result of the operation is to be placed. If ‘d’ is zero, the result is placed in the WREG register. If ‘d’ is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The bit in the file register (specified by ‘b’)
3. The accessed memory (specified by ‘a’)

The bit field designator ‘b’ selects the number of the bit affected by the operation, while the file register designator ‘f’ represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The **control** instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the `CALL` or `RETURN` instructions (specified by ‘s’)
- The mode of the table read and table write instructions (specified by ‘m’)
- No operand required (specified by ‘—’)

All instructions are a single word, except for three 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 MSbs are ‘1’s. If this second word is executed as an instruction (by itself), it will execute as a `NOP`.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a `NOP`.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 µs. Two-word branch instructions (if true) would take 3 µs.

Figure 25-1 shows the general formats that the instructions can have.

All examples use the format ‘nnh’ to represent a hexadecimal number, where ‘h’ signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 25-2, lists the instructions recognized by the Microchip MPASM™ Assembler.

**Section 25.2 “Instruction Set”** provides a description of each instruction.

### 25.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified and the result is stored according to either the instruction or the destination designator ‘d’. A read operation is performed on a register even if the instruction writes to that register.

For example, a “`CLRF PORTB`” instruction will read PORTB, clear all the data bits, then write the result back to PORTB. This example would have the unintended result that the condition that sets the RBIF flag would be cleared.

chanhvinh.com



**TABLE 25-1: OPCODE FIELD DESCRIPTIONS**

| Field | Description |
|---|---|
| a | RAM access bit:<br>a = 0: RAM location in Access RAM (BSR register is ignored)<br>a = 1: RAM bank is specified by BSR register |
| 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:<br>d = 0: store result in WREG<br>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). |
| fs | 12-bit register file address (0x000 to 0xFFF). This is the source address. |
| fd | 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<br><br>*<br>*+<br>*-<br>+* | The mode of the TBLPTR register for the table read and table write instructions.<br>Only used with table read and table write instructions:<br>No change to register (such as TBLPTR with table reads and writes)<br>Post-Increment register (such as TBLPTR with table reads and writes)<br>Post-Decrement register (such as TBLPTR with table reads and writes)<br>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. |
| PRODL | Product of Multiply Low Byte. |
| s | Fast Call/Return mode select bit:<br>s = 0: do not update into/from shadow registers<br>s = 1: certain registers loaded into/from shadow registers (Fast mode) |
| u | Unused or unchanged. |
| WREG | Working register (accumulator). |
| x | Don't care (0 or 1).<br>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). |
| TABLAT | 8-bit Table Latch. |
| TOS | Top-of-Stack. |
| PC | Program Counter |
| PCL | Program Counter Low Byte. |
| PCH | Program Counter High Byte. |
| PCLATH | Program Counter High Byte Latch. |
| PCLATU | Program Counter Upper Byte Latch. |
| GIE | Global Interrupt Enable bit. |
| WDT | Watchdog Timer. |
| $\overline{TO}$ | Time-out bit. |
| $\overline{PD}$ | Power-Down bit. |
| C, DC, Z, OV, N | ALU status bits: Carry, Digit Carry, Zero, Overflow, Negative. |
| [ ] | Optional. |
| ( ) | Contents. |
| → | Assigned to. |
| < > | Register bit field. |
| ∈ | In the set of. |
| italics | User defined term (font is courier). |


© 2006 Microchip Technology Inc.

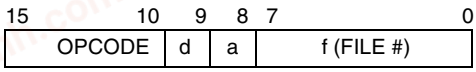
chanhvinh.com



**FIGURE 25-1: GENERAL FORMAT FOR INSTRUCTIONS**

**Byte-oriented** file register operations

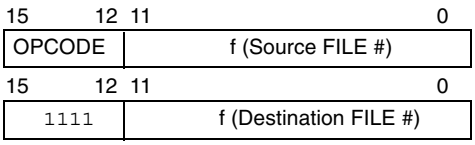
**Example Instruction**




ADDWF MYREG, W, B

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

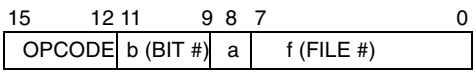
**Byte to Byte** move operations (2-word)




MOVFF MYREG1, MYREG2

f = 12-bit file register address

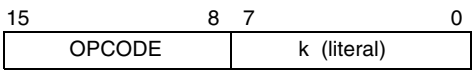
**Bit-oriented** file register operations




BSF MYREG, bit, B

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

**Literal** operations

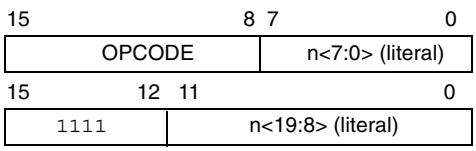



MOVLW 0x7F

k = 8-bit immediate value

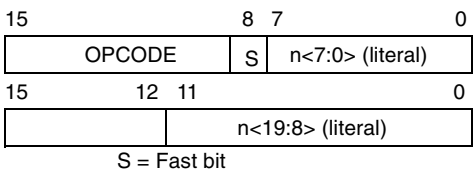
**Control** operations

**CALL, GOTO and Branch** operations



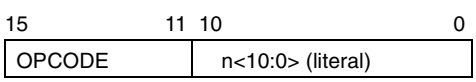

GOTO Label

n = 20-bit immediate value

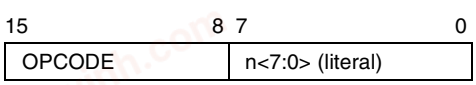



CALL MYFUNC

S = Fast bit




BRA MYFUNC




BC MYFUNC

chanhvinh.com



**TABLE 25-2: PIC18FXXX INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb | LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|---|
| **BYTE-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | |
| ADDWF | f, d, a | Add WREG and f | 1 | 0010 01da | ffff ffff | C, DC, Z, OV, N | 1, 2 |
| ADDWFC | f, d, a | Add WREG and Carry bit to f | 1 | 0010 00da | ffff ffff | C, DC, Z, OV, N | 1, 2 |
| ANDWF | f, d, a | AND WREG with f | 1 | 0001 01da | ffff ffff | Z, N | 1,2 |
| CLRF | f, a | Clear f | 1 | 0110 101a | ffff ffff | Z | 2 |
| COMF | f, d, a | Complement f | 1 | 0001 11da | ffff ffff | Z, N | 1, 2 |
| CPFSEQ | f, a | Compare f with WREG, skip = | 1 (2 or 3) | 0110 001a | ffff ffff | None | 4 |
| CPFSGT | f, a | Compare f with WREG, skip > | 1 (2 or 3) | 0110 010a | ffff ffff | None | 4 |
| CPFSLT | f, a | Compare f with WREG, skip < | 1 (2 or 3) | 0110 000a | ffff ffff | None | 1, 2 |
| DECF | f, d, a | Decrement f | 1 | 0000 01da | ffff ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| DECFSZ | f, d, a | Decrement f, Skip if 0 | 1 (2 or 3) | 0010 11da | ffff ffff | None | 1, 2, 3, 4 |
| DCFSNZ | f, d, a | Decrement f, Skip if Not 0 | 1 (2 or 3) | 0100 11da | ffff ffff | None | 1, 2 |
| INCF | f, d, a | Increment f | 1 | 0010 10da | ffff ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| INCFSZ | f, d, a | Increment f, Skip if 0 | 1 (2 or 3) | 0011 11da | ffff ffff | None | 4 |
| INFSNZ | f, d, a | Increment f, Skip if Not 0 | 1 (2 or 3) | 0100 10da | ffff ffff | None | 1, 2 |
| IORWF | f, d, a | Inclusive OR WREG with f | 1 | 0001 00da | ffff ffff | Z, N | 1, 2 |
| MOVF | f, d, a | Move f | 1 | 0101 00da | ffff ffff | Z, N | 1 |
| MOVFF | $f_s$, $f_d$ | Move $f_s$ (source) to 1st word<br>$f_d$ (destination) 2nd word | 2 | 1100 ffff<br>1111 ffff | ffff ffff<br>ffff ffff | None | |
| MOVWF | f, a | Move WREG to f | 1 | 0110 111a | ffff ffff | None | |
| MULWF | f, a | Multiply WREG with f | 1 | 0000 001a | ffff ffff | None | |
| NEGF | f, a | Negate f | 1 | 0110 110a | ffff ffff | C, DC, Z, OV, N | 1, 2 |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 01da | ffff ffff | C, Z, N | |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 01da | ffff ffff | Z, N | 1, 2 |
| RRCF | f, d, a | Rotate Right f through Carry | 1 | 0011 00da | ffff ffff | C, Z, N | |
| RRNCF | f, d, a | Rotate Right f (No Carry) | 1 | 0100 00da | ffff ffff | Z, N | |
| SETF | f, a | Set f | 1 | 0110 100a | ffff ffff | None | |
| SUBFWB | f, d, a | Subtract f from WREG with borrow | 1 | 0101 01da | ffff ffff | C, DC, Z, OV, N | 1, 2 |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 11da | ffff ffff | C, DC, Z, OV, N | |
| SUBWFB | f, d, a | Subtract WREG from f with borrow | 1 | 0101 10da | ffff ffff | C, DC, Z, OV, N | 1, 2 |
| SWAPF | f, d, a | Swap nibbles in f | 1 | 0011 10da | ffff ffff | None | 4 |
| TSTFSZ | f, a | Test f, skip if 0 | 1 (2 or 3) | 0110 011a | ffff ffff | None | 1, 2 |
| XORWF | f, d, a | Exclusive OR WREG with f | 1 | 0001 10da | ffff ffff | Z, N | |
| **BIT-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | |
| BCF | f, b, a | Bit Clear f | 1 | 1001 bbba | ffff ffff | None | 1, 2 |
| BSF | f, b, a | Bit Set f | 1 | 1000 bbba | ffff ffff | None | 1, 2 |
| BTFSC | f, b, a | Bit Test f, Skip if Clear | 1 (2 or 3) | 1011 bbba | ffff ffff | None | 3, 4 |
| BTFSS | f, b, a | Bit Test f, Skip if Set | 1 (2 or 3) | 1010 bbba | ffff ffff | None | 3, 4 |
| BTG | f, d, a | Bit Toggle f | 1 | 0111 bbba | ffff ffff | None | 1, 2 |

**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.


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 25-2: PIC18FXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|
| **CONTROL OPERATIONS** | | | | | | |
| BC | n | Branch if Carry | 1 (2) | 1110 0010 nnnn nnnn | None | |
| BN | n | Branch if Negative | 1 (2) | 1110 0110 nnnn nnnn | None | |
| BNC | n | Branch if Not Carry | 1 (2) | 1110 0011 nnnn nnnn | None | |
| BNN | n | Branch if Not Negative | 1 (2) | 1110 0111 nnnn nnnn | None | |
| BNOV | n | Branch if Not Overflow | 1 (2) | 1110 0101 nnnn nnnn | None | |
| BNZ | n | Branch if Not Zero | 2 | 1110 0001 nnnn nnnn | None | |
| BOV | n | Branch if Overflow | 1 (2) | 1110 0100 nnnn nnnn | None | |
| BRA | n | Branch Unconditionally | 1 (2) | 1101 0nnn nnnn nnnn | None | |
| BZ | n | Branch if Zero | 1 (2) | 1110 0000 nnnn nnnn | None | |
| CALL | n, s | Call subroutine 1st 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 address 1st 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 | 1 | 1111 xxxx xxxx xxxx | None | 4 |
| 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.

© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 25-2: PIC18FXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|
| **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 | Move literal (12-bit) 2nd word to FSRx 1st word | 2 | 1110 1110 00ff kkkk<br>1111 0000 kkkk 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 `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.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 25.2 Instruction Set

**ADDLW** **ADD Literal to W**

Syntax: [ *label* ] ADDLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) + k → W

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1111 | kkkk | kkkk |

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

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
W = 0x10
After Instruction
W = 0x25

**ADDWF** **ADD W to f**

Syntax: [ *label* ] ADDWF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (W) + (f) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 01da | ffff | ffff |

Description: Add W to register 'f'. 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. If 'a' is '1', the BSR is used.

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
W = 0x17
REG = 0xC2
After Instruction
W = 0xD9
REG = 0xC2

© 2006 Microchip Technology Inc.

chanhvinh.com



**ADDWFC** **ADD W and Carry bit to f**

Syntax: [ *label* ] ADDWFC f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) + (f) + (C) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 00da | ffff | ffff |

Description: Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR will not be overridden.

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
Carry bit = 1
REG = 0x02
W = 0x4D
After Instruction
Carry bit = 0
REG = 0x02
W = 0x50

**ANDLW** **AND Literal with W**

Syntax: [ *label* ] ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. k → W

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1011 | kkkk | kkkk |

Description: The contents of W are ANDed with the 8-bit literal 'k'. The result is placed in W.

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
W = 0xA3
After Instruction
W = 0x03


© 2006 Microchip Technology Inc.

chanhvinh.com



**ANDWF** **AND W with f**

Syntax: [ *label* ] ANDWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 01da | ffff | ffff |

Description: The contents of W are ANDed with register 'f'. 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. If 'a' is '1', the BSR will not be overridden (default).

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
W = 0x17
REG = 0xC2
After Instruction
W = 0x02
REG = 0xC2

**BC** **Branch if Carry**

Syntax: [ *label* ] BC n

Operands: $-128 \le n \le 127$

Operation: if Carry bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0010 | nnnn | nnnn |

Description: If the Carry bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BC JUMP`

Before Instruction
PC = address (HERE)
After Instruction
If Carry = 1;
PC = address (JUMP)
If Carry = 0;
PC = address (HERE + 2)

© 2006 Microchip Technology Inc.

chanhvinh.com



**BCF** **Bit Clear f**

Syntax: [ *label* ] BCF f,b[,a]

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 0 → 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, then 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: `BCF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

**BN** **Branch if Negative**

Syntax: [ *label* ] BN n

Operands: -128 ≤ n ≤ 127

Operation: if Negative bit is '1'
(PC) + 2 + 2n → 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)


© 2006 Microchip Technology Inc.

chanhvinh.com



**BNC** **Branch if Not Carry**

Syntax: [ *label* ] BNC n

Operands: -128 ≤ n ≤ 127

Operation: if Carry bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0011 | nnnn | nnnn |

Description: If the Carry bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BNC Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If Carry = 0;
PC = address `(Jump)`
If Carry = 1;
PC = address `(HERE + 2)`

**BNN** **Branch if Not Negative**

Syntax: [ *label* ] BNN n

Operands: -128 ≤ n ≤ 127

Operation: if Negative bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0111 | nnnn | nnnn |

Description: If the Negative bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BNN Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If Negative = 0;
PC = address `(Jump)`
If Negative = 1;
PC = address `(HERE + 2)`

© 2006 Microchip Technology Inc.

chanhvinh.com



**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)`


© 2006 Microchip Technology Inc.

chanhvinh.com



**BRA** **Unconditional Branch**

Syntax: [ *label* ] BRA n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1101 | 0nnn | nnnn | nnnn |

Description: Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: HERE BRA Jump

Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

**BSF** **Bit Set f**

Syntax: [ *label* ] BSF f,b[,a]

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 1 → f<b>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | bbba | ffff | ffff |

Description: Bit 'b' in register 'f' is set. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then 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: BSF FLAG_REG, 7

Before Instruction
FLAG_REG = 0x0A
After Instruction
FLAG_REG = 0x8A

© 2006 Microchip Technology Inc.

chanhvinh.com



**BTFSC** **Bit Test File, Skip if Clear**

Syntax: [ *label* ] BTFSC f,b[,a]

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

| 1011 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '0', then the next instruction is skipped.
If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSC   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

**BTFSS** **Bit Test File, Skip if Set**

Syntax: [ *label* ] BTFSS f,b[,a]

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

| 1010 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '1', then the next instruction is skipped.
If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSS   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)


© 2006 Microchip Technology Inc.

chanhvinh.com



**BTG** **Bit Toggle f**

Syntax: [ *label* ] BTG f,b[,a]

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0111 | bbba | ffff | ffff |

Description: Bit 'b' in data memory location 'f' is inverted. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then 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: `BTG PORTC, 4`

Before Instruction:
PORTC = 0111 0101 [0x75]
After Instruction:
PORTC = 0110 0101 [0x65]

**BOV** **Branch if Overflow**

Syntax: [ *label* ] BOV n

Operands: $-128 \le n \le 127$

Operation: if Overflow bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0100 | nnnn | nnnn |

Description: If the Overflow bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BOV   JUMP
```
Before Instruction
PC = address (HERE)
After Instruction
If Overflow = 1;
PC = address (JUMP)
If Overflow = 0;
PC = address (HERE + 2)

© 2006 Microchip Technology Inc.

chanhvinh.com



**BZ** **Branch if Zero**

Syntax: [ *label* ] BZ n

Operands: -128 ≤ n ≤ 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 ≤ k ≤ 1048575
s ∈ [0,1]

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(W) → WS,
(Status) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (k<7:0>) | 1110 | 110s | $k_7$kkk | kkkk$_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}$kkk | kkkk | kkkk$_8$ |

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. `CALL` is a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | Push PC to stack | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE CALL THERE,FAST`

Before Instruction
PC = address `(HERE)`
After Instruction
PC = address `(THERE)`
TOS = address `(HERE + 4)`
WS = W
BSRS = BSR
STATUSS = Status


© 2006 Microchip Technology Inc.

chanhvinh.com



**CLRF** **Clear f**

Syntax: [ *label* ] CLRF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: 000h → f
1 → Z

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 101a | ffff | ffff |

Description: Clears the contents of the specified register. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then 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: CLRF FLAG_REG

Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00

**CLRWDT** **Clear Watchdog Timer**

Syntax: [ *label* ] CLRWDT

Operands: None

Operation: 000h → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0100 |

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | No operation |

Example: CLRWDT

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 0x00
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

© 2006 Microchip Technology Inc.

chanhvinh.com



**COMF** **Complement f**

Syntax: [ *label* ] COMF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(\overline{f}) \rightarrow dest$

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 11da | ffff | ffff |

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in 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' = 1, then 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 to destination |

Example: COMF REG, W

Before Instruction
REG = 0x13
After Instruction
REG = 0x13
W = 0xEC

**CPFSEQ** **Compare f with W, Skip if f = W**

Syntax: [ *label* ] CPFSEQ f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(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' = W, 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' = 1, then the bank will be selected as per the BSR value (default).

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
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)


© 2006 Microchip Technology Inc.

chanhvinh.com



**CPFSGT** **Compare f with W, Skip if f > W**

Syntax: [ *label* ] CPFSGT f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 010a | ffff | ffff |

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater 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, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

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)
W = ?
After Instruction
If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

**CPFSLT** **Compare f with W, Skip if f < W**

Syntax: [ *label* ] CPFSLT f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) < (W)
(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 W, 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 BSR will not be overridden (default).

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)
W = ?
After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

© 2006 Microchip Technology Inc.

chanhvinh.com



**DAW** **Decimal Adjust W Register**

Syntax: [ *label* ] DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>

If [W<7:4> > 9] or [C = 1] then
(W<7:4>) + 6 → W<7:4>;
else
(W<7:4>) → W<7:4>

Status Affected: C

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0111 |

Description: DAW adjusts the eight-bit value in W 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 W | Process Data | Write W |

Example 1: DAW

Before Instruction
W = 0xA5
C = 0
DC = 0

After Instruction
W = 0x05
C = 1
DC = 0

Example 2:

Before Instruction
W = 0xCE
C = 0
DC = 0

After Instruction
W = 0x34
C = 1
DC = 0

**DECF** **Decrement f**

Syntax: [ *label* ] DECF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 01da | ffff | ffff |

Description: Decrement register 'f'. If 'd' is '0', the result is stored in 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' = 1, then 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 to destination |

Example: DECF CNT,

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1


© 2006 Microchip Technology Inc.

chanhvinh.com



**DECFSZ** **Decrement f, Skip if 0**

Syntax: [ *label* ] DECFSZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

| 0010 | 11da | ffff | ffff |
|---|---|---|---|

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. 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' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DECFSZ  CNT
        GOTO    LOOP
CONTINUE
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT – 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE + 2)

**DCFSNZ** **Decrement f, Skip if not 0**

Syntax: [ *label* ] DCFSNZ f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

| 0100 | 11da | ffff | ffff |
|---|---|---|---|

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. 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' = 1, then the bank will be selected as per the BSR value (default).

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)

chanhvinh.com



**GOTO** **Unconditional Branch**

Syntax: [ *label* ] GOTO k

Operands: 0 ≤ k ≤ 1048575

Operation: k → 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 2-Mbyte 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 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) + 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

| 0010 | 10da | ffff | ffff |
|---|---|---|---|

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. 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' = 1, then 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 to destination |

Example: INCF CNT,

Before Instruction
CNT = 0xFF
Z = 0
C = ?
DC = ?

After Instruction
CNT = 0x00
Z = 1
C = 1
DC = 1



© 2006 Microchip Technology Inc.

chanhvinh.com



**INCFSZ** **Increment f, Skip if 0**

Syntax: [ *label* ] INCFSZ f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) + 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 11da | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. 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' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INCFSZ   CNT
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

**INFSNZ** **Increment f, Skip if not 0**

Syntax: [ *label* ] INFSNZ f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) + 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 10da | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. 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' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INFSNZ REG
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

© 2006 Microchip Technology Inc.

chanhvinh.com



**IORLW** **Inclusive OR Literal with W**

Syntax: [ *label* ] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → W

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1001 | kkkk | kkkk |

Description: The contents of W are ORed with the eight-bit literal 'k'. The result is placed in W.

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
W = 0x9A
After Instruction
W = 0xBF

**IORWF** **Inclusive OR W with f**

Syntax: [ *label* ] IORWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .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 W. 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' = 1, then 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 to destination |

Example: `IORWF RESULT, W`

Before Instruction
RESULT = 0x13
W = 0x91
After Instruction
RESULT = 0x13
W = 0x93


© 2006 Microchip Technology Inc.

chanhvinh.com



**LFSR** **Load FSR**

Syntax: [ *label* ] LFSR f,k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 1111 | 1110 0000 | 00ff $k_7$kkk | $k_{11}$kkk kkkk |

Description: The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' MSB | Process Data | Write literal 'k' MSB to FSRfH |
| Decode | Read literal 'k' LSB | Process Data | Write literal 'k' to FSRfL |

Example: `LFSR 2, 0x3AB`

After Instruction
FSR2H = 0x03
FSR2L = 0xAB

**MOVF** **Move f**

Syntax: [ *label* ] MOVF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 00da | ffff | ffff |

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. 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' = 1, then 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 W |

Example: `MOVF REG, W`

Before Instruction
REG = 0x22
W = 0xFF
After Instruction
REG = 0x22
W = 0x22

chanhvinh.com



**MOVFF** **Move f to f**

Syntax: [ *label* ] MOVFF $f_s$,$f_d$

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (source) | 1100 | ffff | ffff | $ffff_s$ |
| 2nd word (destin.) | 1111 | ffff | ffff | $ffff_d$ |

Description: The contents of source register '$f_s$' are moved to destination register '$f_d$'. Location of source '$f_s$' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination '$f_d$' can also be anywhere from 000h to FFFh.

Either source or destination can be W (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.

The MOVFF instruction should not be used to modify interrupt settings while any interrupt is enabled (see page 77).

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' (src) | Process Data | No operation |
| Decode | No operation<br>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: $0 \le k \le 255$

Operation: $k \rightarrow BSR$

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 5

Before Instruction
BSR register = 0x02
After Instruction
BSR register = 0x05


© 2006 Microchip Technology Inc.

chanhvinh.com



**MOVLW** **Move Literal to W**

Syntax: [ *label* ] MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1110 | kkkk | kkkk |

Description: The eight-bit literal 'k' is loaded into W.

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
W = 0x5A

**MOVWF** **Move W to f**

Syntax: [ *label* ] MOVWF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 111a | ffff | ffff |

Description: Move data from W 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' = 1, then 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: `MOVWF REG`

Before Instruction
W = 0x4F
REG = 0xFF
After Instruction
W = 0x4F
REG = 0x4F

© 2006 Microchip Technology Inc.

chanhvinh.com



**MULLW** **Multiply Literal with W**

Syntax: [ *label* ] MULLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) x k → PRODH:PRODL

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1101 | kkkk | kkkk |

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W 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
W = 0xE2
PRODH = ?
PRODL = ?

After Instruction
W = 0xE2
PRODH = 0xAD
PRODL = 0x08

**MULWF** **Multiply W with f**

Syntax: [ *label* ] MULWF f [,a]

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: (W) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 001a | ffff | ffff |

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte.
Both W 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'= 1, then 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 registers PRODH: PRODL |

Example: MULWF REG

Before Instruction
W = 0xC4
REG = 0xB5
PRODH = ?
PRODL = ?

After Instruction
W = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94



© 2006 Microchip Technology Inc.

chanhvinh.com



**NEGF** **Negate f**

Syntax: [ *label* ] NEGF f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

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' = 1, then the bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example:
```
NEGF    REG, 1
```
Before Instruction
REG = 0011 1010 [0x3A]
After Instruction
REG = 1100 0110 [0xC6]

**NOP** **No Operation**

Syntax: [ *label* ] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 1111 | 0000 xxxx | 0000 xxxx | 0000 xxxx |

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

Example:

None.

chanhvinh.com

chanhvinh.com



**POP** **Pop Top of Return Stack**

Syntax: [ *label* ] POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0110 |

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | POP TOS value | No operation |

Example:

```
POP
GOTO    NEW
```

Before Instruction
TOS = 0x0031A2
Stack (1 level down) = 0x014332

After Instruction
TOS = 0x014332
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 the user to implement a software stack by modifying TOS and then pushing 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 = 0x00345A
PC = 0x000124

After Instruction
PC = 0x000126
TOS = 0x000126
Stack (1 level down) = 0x00345A



© 2006 Microchip Technology Inc.

chanhvinh.com



**RCALL** **Relative Call**

Syntax: [ *label* ] RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → 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'<br>Push PC to stack | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: HERE RCALL Jump

Before Instruction
PC = Address (HERE)
After Instruction
PC = Address (Jump)
TOS = Address (HERE + 2)

**RESET** **Reset**

Syntax: [ *label* ] RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 1111 | 1111 |

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ 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

chanhvinh.com



**RETFIE** **Return from Interrupt**

Syntax: [ *label* ] RETFIE [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 000s |

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers W, 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<br>Set GIEH or GIEL |
| No operation | No operation | No operation | No operation |

Example: RETFIE 1

After Interrupt
PC = TOS
W = WS
BSR = BSRS
Status = STATUSS
GIE/GIEH, PEIE/GIEL = 1

**RETLW** **Return Literal to W**

Syntax: [ *label* ] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1100 | kkkk | kkkk |

Description: W is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Pop PC from stack, Write to W |
| No operation | No operation | No operation | No operation |

Example:

```
    CALL TABLE  ; W contains table
                ; offset value
                ; W now has
                ; table value
    :
TABLE
    ADDWF PCL   ; W = offset
    RETLW k0    ; Begin table
    RETLW k1    ;
    :
    :
    RETLW kn    ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of kn


© 2006 Microchip Technology Inc.

chanhvinh.com



**RETURN** **Return from Subroutine**

Syntax: [ *label* ] RETURN [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 001s |

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's'= 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers W, 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 Interrupt
PC = TOS

**RLCF** **Rotate Left f through Carry**

Syntax: [ *label* ] RLCF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

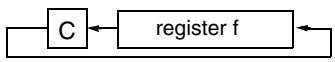
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 01da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in 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' = 1, then 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 to destination |

Example: RLCF REG, W

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 1100 1100
C = 1

© 2006 Microchip Technology Inc.

chanhvinh.com



**RLNCF** **Rotate Left f (no carry)**

Syntax: [ *label* ] RLNCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

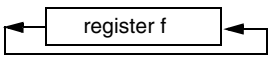
Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 01da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in 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', then 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 to destination |

Example: RLNCF REG

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

**RRCF** **Rotate Right f through Carry**

Syntax: [ *label* ] RRCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

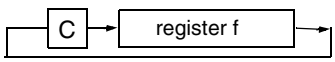
Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C, N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 00da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in W. 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', then 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 to destination |

Example: RRCF REG, W

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0


© 2006 Microchip Technology Inc.

chanhvinh.com



**RRNCF** **Rotate Right f (no carry)**

Syntax: [ *label* ] RRNCF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

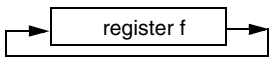
Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 00da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. 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', then 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 to destination |

Example 1:
```
RRNCF   REG, 1, 0
```
Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2:
```
RRNCF   REG, W
```
Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

**SETF** **Set f**

Syntax: [ *label* ] SETF f [,a]

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: FFh → f

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 100a | ffff | ffff |

Description: The contents of the specified register are set to FFh. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then 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:
```
SETF    REG
```
Before Instruction
REG = 0x5A
After Instruction
REG = 0xFF

chanhvinh.com



**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 W with Borrow**

Syntax: [ *label* ] SUBFWB f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (W) – (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 W (2's complement method). If 'd' is '0', the result is stored in W. 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', then 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 to destination |

Example 1: SUBFWB REG

Before Instruction
REG = 0x03
W = 0x02
C = 0x01
After Instruction
REG = 0xFF
W = 0x02
C = 0x00
Z = 0x00
N = 0x01 ; result is negative

Example 2: SUBFWB REG, 0, 0

Before Instruction
REG = 2
W = 5
C = 1
After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3: SUBFWB REG, 1, 0

Before Instruction
REG = 1
W = 2
C = 0
After Instruction
REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0


© 2006 Microchip Technology Inc.

chanhvinh.com



**SUBLW** **Subtract W from Literal**

Syntax: [ *label* ] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1000 | kkkk | kkkk |

Description: W is subtracted from the eight-bit literal 'k'. The result is placed in W.

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
W = 1
C = ?
After Instruction
W = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2:
```
SUBLW   0x02
```
Before Instruction
W = 2
C = ?
After Instruction
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3:
```
SUBLW   0x02
```
Before Instruction
W = 3
C = ?
After Instruction
W = FF ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

**SUBWF** **Subtract W from f**

Syntax: [ *label* ] SUBWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(f) - (W) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 11da | ffff | ffff |

Description: Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then 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 to destination |

Example 1:
```
SUBWF REG
```
Before Instruction
REG = 3
W = 2
C = ?
After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2:
```
SUBWF REG, W
```
Before Instruction
REG = 2
W = 2
C = ?
After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3:
```
SUBWF REG
```
Before Instruction
REG = 0x01
W = 0x02
C = ?
After Instruction
REG = 0xFFh ;(2's complement)
W = 0x02
C = 0x00 ; result is negative
Z = 0x00
N = 0x01

© 2006 Microchip Technology Inc.

chanhvinh.com



**SUBWFB** **Subtract W from f with Borrow**

Syntax: [ *label* ] SUBWFB f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – (W) – ($\overline{C}$) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 10da | ffff | ffff |

Description: Subtract W and the Carry flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then 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 to destination |

Example 1:
```
SUBWFB REG, 1, 0
```
Before Instruction
```
REG  =  0x19   (0001 1001)
W    =  0x0D   (0000 1101)
C    =  0x01
```
After Instruction
```
REG  =  0x0C   (0000 1011)
W    =  0x0D   (0000 1101)
C    =  0x01
Z    =  0x00
N    =  0x00   ; result is positive
```

Example 2:
```
SUBWFB REG, 0, 0
```
Before Instruction
```
REG  =  0x1B   (0001 1011)
W    =  0x1A   (0001 1010)
C    =  0x00
```
After Instruction
```
REG  =  0x1B   (0001 1011)
W    =  0x00
C    =  0x01
Z    =  0x01   ; result is zero
N    =  0x00
```

Example 3:
```
SUBWFB REG, 1, 0
```
Before Instruction
```
REG  =  0x03   (0000 0011)
W    =  0x0E   (0000 1101)
C    =  0x01
```
After Instruction
```
REG  =  0xF5   (1111 0100)
                ; [2's comp]
W    =  0x0E   (0000 1101)
C    =  0x00
Z    =  0x00
N    =  0x01   ; result is negative
```

**SWAPF** **Swap f**

Syntax: [ *label* ] SWAPF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<3:0>) → dest<7:4>,
(f<7:4>) → dest<3:0>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 10da | ffff | ffff |

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. 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', then 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 to destination |

Example:
```
SWAPF  REG
```
Before Instruction
```
REG  =  0x53
```
After Instruction
```
REG  =  0x35
```


© 2006 Microchip Technology Inc.

chanhvinh.com



**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<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:
- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

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

© 2006 Microchip Technology Inc. 

chanhvinh.com



**TBLWT** **Table Write**

Syntax: [ *label* ] TBLWT ( *; *+; *-; +*)

Operands: None

Operation:
if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 11nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 6.0 "Flash Program Memory"** for additional details.)
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

**TBLWT Table Write (Continued)**

Words: 1

Cycles: 2

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) |

Example 1:

```
TBLWT  *+;
```

Before Instruction
TABLAT = 0x55
TBLPTR = 0x00A356
HOLDING REGISTER (0x00A356) = 0xFF
After Instructions (table write completion)
TABLAT = 0x55
TBLPTR = 0x00A357
HOLDING REGISTER (0x00A356) = 0x55

Example 2:

```
TBLWT  +*;
```

Before Instruction
TABLAT = 0x34
TBLPTR = 0x01389A
HOLDING REGISTER (0x01389A) = 0xFF
HOLDING REGISTER (0x01389B) = 0xFF
After Instruction (table write completion)
TABLAT = 0x34
TBLPTR = 0x01389B
HOLDING REGISTER (0x01389A) = 0xFF
HOLDING REGISTER (0x01389B) = 0x34


© 2006 Microchip Technology Inc.

chanhvinh.com



**TSTFSZ** **Test f, Skip if 0**

Syntax: [ *label* ] TSTFSZ f [,a]

Operands: 0 ≤ f ≤ 255
a ∈ [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', then the bank will be selected as per the BSR value (default).

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 ≠ 0x00,
PC = Address (NZERO)

**XORLW** **Exclusive OR Literal with W**

Syntax: [ *label* ] XORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) .XOR. k → W

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1010 | kkkk | kkkk |

Description: The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example:
```
XORLW  0xAF
```
Before Instruction
W = 0xB5
After Instruction
W = 0x1A

chanhvinh.com



| XORWF | Exclusive OR W with f |
|---|---|
| Syntax: | [ *label* ] XORWF f [,d [,a]] |
| Operands: | 0 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1] |
| Operation: | (W) .XOR. (f) → dest |
| Status Affected: | N, Z |
| Encoding: | 0001 10da ffff ffff |
| Description: | Exclusive OR the contents of W with register 'f'. 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', then 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 to destination |

Example: XORWF REG

Before Instruction
REG = 0xAF
W = 0xB5
After Instruction
REG = 0x1A
W = 0xB5


© 2006 Microchip Technology Inc.

chanhvinh.com



## 26.0 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™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB C30 C Compiler
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
  - MPLAB dsPIC30 Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB ICE 4000 In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Development Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM.net™ Demonstration Board
  - PICDEM 2 Plus Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 4 Demonstration Board
  - PICDEM 17 Demonstration Board
  - PICDEM 18R Demonstration Board
  - PICDEM LIN Demonstration Board
  - PICDEM USB Demonstration Board
- Evaluation Kits
  - KEELOQ® Evaluation and Programming Tools
  - PICDEM MSC
  - microID® Developer Kits
  - CAN
  - PowerSmart® Developer Kits
  - Analog

### 26.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-bit microcontroller market. The MPLAB IDE is a Windows® based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor with color coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Extensive 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 emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files (assembly or C)
  - mixed assembly and C
  - machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increasing flexibility and power.

### 26.2 MPASM Assembler

The MPASM assembler is a full-featured, universal macro assembler for all PICmicro MCUs.

The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects
- User defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

© 2006 Microchip Technology Inc.

chanhvinh.com



## 26.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI C compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

## 26.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB object librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 26.5 MPLAB C30 C Compiler

The MPLAB C30 C compiler is a full-featured, ANSI compliant, optimizing compiler that translates standard ANSI C programs into dsPIC30F assembly language source. The compiler also supports many command line options and language extensions to take full advantage of the dsPIC30F device hardware capabilities and afford fine control of the compiler code generator.

MPLAB C30 is distributed with a complete ANSI C standard library. All library functions have been validated and conform to the ANSI C library standard. The library includes functions for string manipulation, dynamic memory allocation, data conversion, time-keeping and math functions (trigonometric, exponential and hyperbolic). The compiler provides symbolic information for high-level source debugging with the MPLAB IDE.

## 26.6 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 compiler uses the assembler to produce it's object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

## 26.7 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC hosted 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 pin. 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 MPLAB C18 C Compilers, as well as the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent, economical software development tool.

## 26.8 MPLAB SIM30 Software Simulator

The MPLAB SIM30 software simulator allows code development in a PC hosted environment by simulating the dsPIC30F 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 MPLAB SIM30 simulator fully supports symbolic debugging using the MPLAB C30 C Compiler and MPLAB ASM30 assembler. The simulator runs in either a Command Line mode for automated tasks, or from MPLAB IDE. This high-speed simulator is designed to debug, analyze and optimize time intensive DSP routines.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 26.9 MPLAB ICE 2000 High-Performance Universal In-Circuit Emulator

The MPLAB ICE 2000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers. Software control of the MPLAB ICE 2000 in-circuit emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE 2000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

## 26.10 MPLAB ICE 4000 High-Performance Universal In-Circuit Emulator

The MPLAB ICE 4000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for high-end PICmicro microcontrollers. Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICD 4000 is a premium emulator system, providing the features of MPLAB ICE 2000, but with increased emulation memory and high-speed performance for dsPIC30F and PIC18XXXX devices. Its advanced emulator features include complex triggering and timing, up to 2 Mb of emulation memory and the ability to view variables in real-time.

The MPLAB ICE 4000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

## 26.11 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PICmicro MCUs and can be used to develop for these and other PICmicro microcontrollers. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost effective in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single-stepping and watching variables, CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real-time. MPLAB ICD 2 also serves as a development programmer for selected PICmicro devices.

## 26.12 PRO MATE II Universal Device Programmer

The PRO MATE II is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features an LCD display for instructions and error messages and a modular detachable socket assembly to support various package types. In Stand-Alone mode, the PRO MATE II device programmer can read, verify and program PICmicro devices without a PC connection. It can also set code protection in this mode.

## 26.13 MPLAB PM3 Device Programmer

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

© 2006 Microchip Technology Inc.

chanhvinh.com



## 26.14 PICSTART Plus Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low-cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus development programmer supports most PICmicro devices 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.

## 26.15 PICDEM 1 PICmicro Demonstration Board

The PICDEM 1 demonstration board demonstrates the capabilities of the 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 sample microcontrollers provided with the PICDEM 1 demonstration board can be programmed with a PRO MATE II device programmer or a PICSTART Plus development programmer. The PICDEM 1 demonstration board can be connected to the MPLAB ICE in-circuit emulator for testing. A prototype area extends the circuitry for additional application components. Features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs.

## 26.16 PICDEM.net Internet/Ethernet Demonstration Board

The PICDEM.net demonstration board is an Internet/Ethernet demonstration board using the PIC18F452 microcontroller and TCP/IP firmware. The board supports any 40-pin DIP device that conforms to the standard pinout used by the PIC16F877 or PIC18C452. This kit features a user friendly TCP/IP stack, web server with HTML, a 24L256 Serial EEPROM for Xmodem download to web pages into Serial EEPROM, ICSP/MPLAB ICD 2 interface connector, an Ethernet interface, RS-232 interface and a 16 x 2 LCD display. Also included is the book and CD-ROM *"TCP/IP Lean, Web Servers for Embedded Systems,"* by Jeremy Bentham

## 26.17 PICDEM 2 Plus Demonstration Board

The PICDEM 2 Plus demonstration board supports many 18, 28 and 40-pin microcontrollers, including PIC16F87X and PIC18FXX2 devices. All the necessary hardware and software is included to run the demonstration programs. The sample microcontrollers provided with the PICDEM 2 demonstration board can be programmed with a PRO MATE II device programmer, PICSTART Plus development programmer, or MPLAB ICD 2 with a Universal Programmer Adapter. The MPLAB ICD 2 and MPLAB ICE in-circuit emulators may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area extends the circuitry for additional application components. Some of the features include an RS-232 interface, a 2 x 16 LCD display, a piezo speaker, an on-board temperature sensor, four LEDs and sample PIC18F452 and PIC16F877 Flash microcontrollers.

## 26.18 PICDEM 3 PIC16C92X Demonstration Board

The PICDEM 3 demonstration board supports the PIC16C923 and PIC16C924 in the PLCC package. All the necessary hardware and software is included to run the demonstration programs.

## 26.19 PICDEM 4 8/14/18-Pin Demonstration Board

The PICDEM 4 can be used to demonstrate the capabilities of the 8, 14 and 18-pin PIC16XXXX and PIC18XXXX MCUs, including the PIC16F818/819, PIC16F87/88, PIC16F62XA and the PIC18F1320 family of microcontrollers. PICDEM 4 is intended to showcase the many features of these low pin count parts, including LIN and Motor Control using ECCP. Special provisions are made for low-power operation with the supercapacitor circuit and jumpers allow on-board hardware to be disabled to eliminate current draw in this mode. Included on the demo board are provisions for Crystal, RC or Canned Oscillator modes, a five volt regulator for use with a nine volt wall adapter or battery, DB-9 RS-232 interface, ICD connector for programming via ICSP and development with MPLAB ICD 2, 2 x 16 liquid crystal display, PCB footprints for H-Bridge motor driver, LIN transceiver and EEPROM. Also included are: header for expansion, eight LEDs, four potentiometers, three push buttons and a prototyping area. Included with the kit is a PIC16F627A and a PIC18F1320. Tutorial firmware is included along with the User's Guide.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 26.20 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. A programmed sample is included. The PRO MATE II device programmer, or the PICSTART Plus development programmer, can be used to reprogram the device for user tailored application development. The PICDEM 17 demonstration board supports program download and execution from external on-board Flash memory. A generous prototype area is available for user hardware expansion.

## 26.21 PICDEM 18R PIC18C601/801 Demonstration Board

The PICDEM 18R demonstration board serves to assist development of the PIC18C601/801 family of Microchip microcontrollers. It provides hardware implementation of both 8-bit Multiplexed/Demultiplexed and 16-bit Memory modes. The board includes 2 Mb external Flash memory and 128 Kb SRAM memory, as well as serial EEPROM, allowing access to the wide range of memory types supported by the PIC18C601/801.

## 26.22 PICDEM LIN PIC16C43X Demonstration Board

The powerful LIN hardware and software kit includes a series of boards and three PICmicro microcontrollers. The small footprint PIC16C432 and PIC16C433 are used as slaves in the LIN communication and feature on-board LIN transceivers. A PIC16F874 Flash microcontroller serves as the master. All three microcontrollers are programmed with firmware to provide LIN bus communication.

## 26.23 PICkit™ 1 Flash Starter Kit

A complete “development system in a box”, the PICkit™ Flash Starter Kit includes a convenient multi-section board for programming, evaluation and development of 8/14-pin Flash PIC® microcontrollers. Powered via USB, the board operates under a simple Windows GUI. The PICkit 1 Starter Kit includes the User’s Guide (on CD ROM), PICkit 1 tutorial software and code for various applications. Also included are MPLAB® IDE (Integrated Development Environment) software, software and hardware “Tips 'n Tricks for 8-pin Flash PIC® Microcontrollers” Handbook and a USB interface cable. Supports all current 8/14-pin Flash PIC microcontrollers, as well as many future planned devices.

## 26.24 PICDEM USB PIC16C7X5 Demonstration Board

The PICDEM USB Demonstration Board shows off the capabilities of the PIC16C745 and PIC16C765 USB microcontrollers. This board provides the basis for future USB products.

## 26.25 Evaluation and Programming Tools

In addition to the PICDEM series of circuits, Microchip has a line of evaluation kits and demonstration software for these products.

- KEELOQ evaluation and programming tools for Microchip’s HCS Secure Data Products
- CAN developers kit for automotive network applications
- Analog design boards and filter design software
- PowerSmart battery charging evaluation/ calibration kits
- IrDA® development kit
- microID development and rfLab™ development software
- SEEVAL® designer kit for memory evaluation and endurance calculations
- PICDEM MSC demo boards for Switching mode power supply, high-power IR driver, delta sigma ADC and flow rate sensor

Check the Microchip web page and the latest Product Selector Guide for the complete list of demonstration and evaluation kits.

© 2006 Microchip Technology Inc.

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 27.0 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to $V_{SS}$ (except $V_{DD}$, $\overline{MCLR}$ and RA4) ..... -0.3V to ($V_{DD}$ + 0.3V)
Voltage on $V_{DD}$ with respect to $V_{SS}$ ..... -0.3V to +7.5V
Voltage on $\overline{MCLR}$ with respect to $V_{SS}$ **(Note 2)** ..... 0V to +13.25V
Voltage on RA4 with respect to $V_{SS}$ ..... 0V to +8.5V
Total power dissipation **(Note 1)** ..... 1.0W
Maximum current out of $V_{SS}$ pin ..... 300 mA
Maximum current into $V_{DD}$ pin ..... 250 mA
Input clamp current, $I_{IK}$ ($V_I$ < 0 or $V_I$ > $V_{DD}$) ..... ±20 mA
Output clamp current, $I_{OK}$ ($V_O$ < 0 or $V_O$ > $V_{DD}$) ..... ±20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by all ports (combined) ..... 200 mA
Maximum current sourced by all ports (combined) ..... 200 mA

**Note 1:** Power dissipation is calculated as follows:
$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

**2:** Voltage spikes below $V_{SS}$ at the $\overline{MCLR}/V_{PP}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}/V_{PP}$ pin rather than pulling this pin directly to $V_{SS}$.

| | |
|---|---|
| **Note:** | 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. |

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-1: PIC18FXX8 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)**

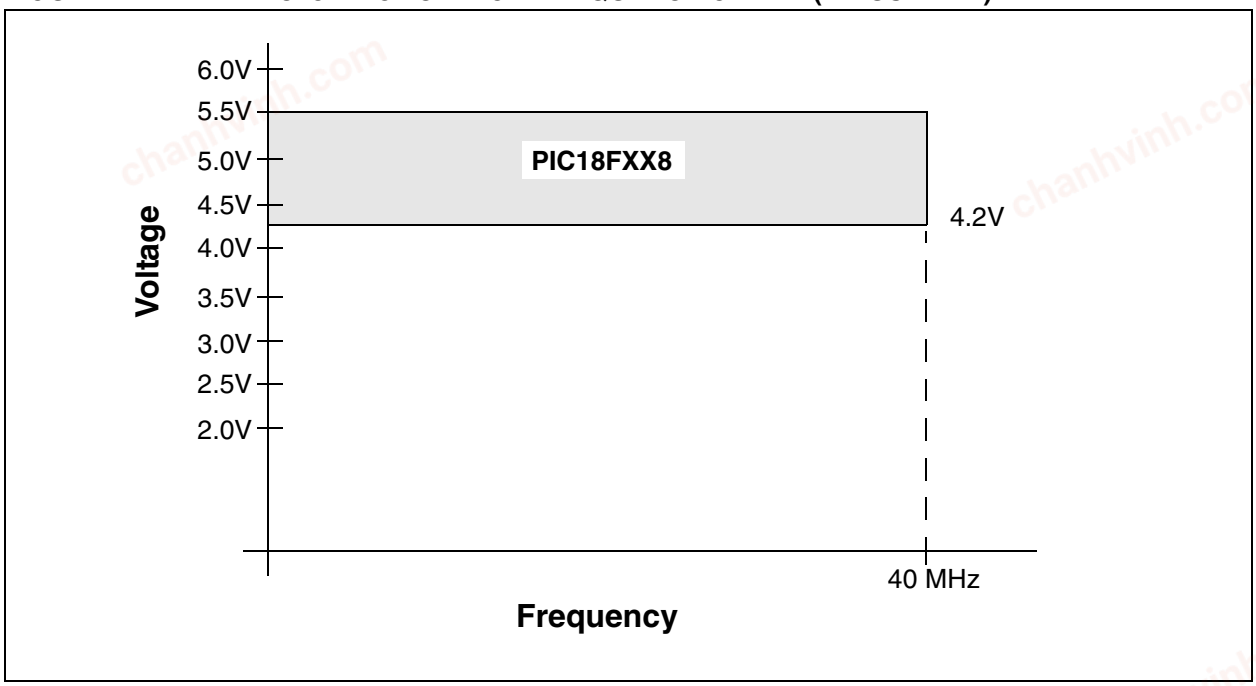
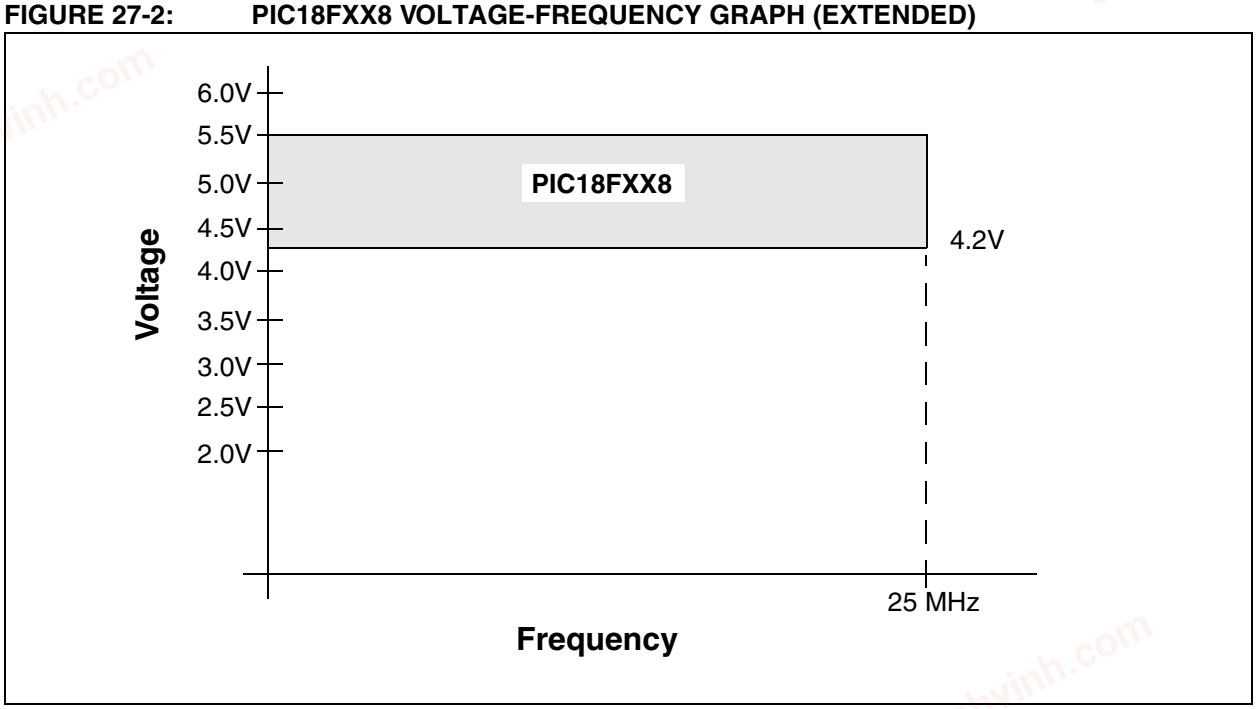



**FIGURE 27-2: PIC18FXX8 VOLTAGE-FREQUENCY GRAPH (EXTENDED)**



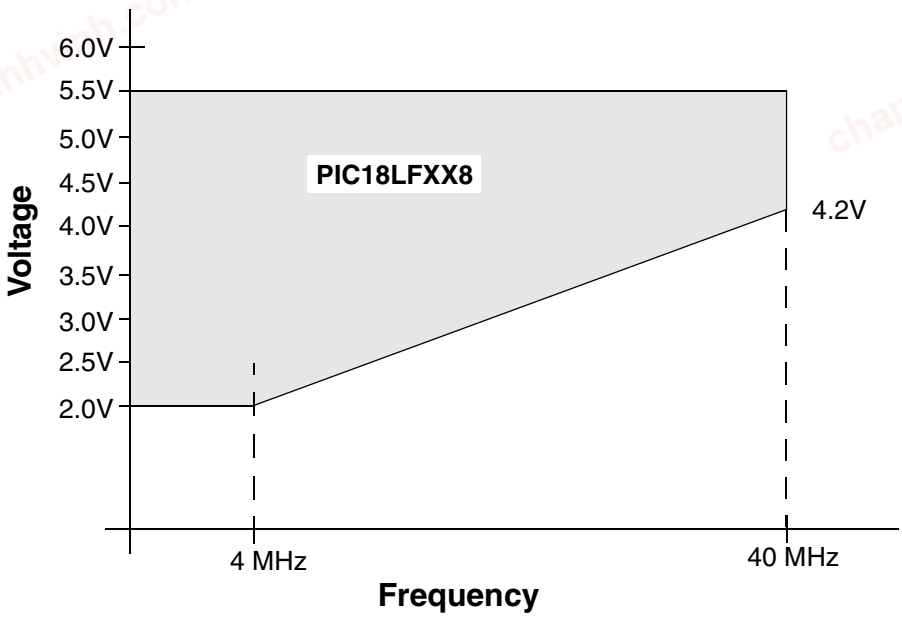


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-3: PIC18LFXX8 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)**




FMAX = (16.36 MHz/V) (VDDAPPMIN – 2.0V) + 4 MHz, if VDDAPPMIN ≤ 4.2V = 40 MHz, if VDDAPPMIN > 4.2V

**Note:** VDDAPPMIN is the minimum voltage of the PICmicro® device in the application.

chanhvinh.com



## 27.1 DC Characteristics

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LFXX8** (Industrial) | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
| **PIC18FXX8** (Industrial, Extended) | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
| **Param No.** | **Symbol** | **Characteristic/ Device** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| | VDD | **Supply Voltage** | | | | | |
| D001 | | PIC18LFXX8 | 2.0 | — | 5.5 | V | HS, XT, RC and LP Oscillator modes |
| D001 | | PIC18FXX8 | 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 |
| | VBOR | **Brown-out Reset Voltage** | | | | | |
| | | PIC18LFXX8 | | | | | |
| D005 | | BORV1:BORV0 = 11 | 1.96 | — | 2.16 | V | |
| | | BORV1:BORV0 = 10 | 2.64 | — | 2.92 | V | |
| | | BORV1:BORV0 = 01 | 4.07 | — | 4.59 | V | |
| | | BORV1:BORV0 = 00 | 4.36 | — | 4.92 | V | |
| | | PIC18FXX8 | | | | | |
| D005 | | BORV1:BORV0 = 1x | N.A. | — | N.A. | V | Not in operating voltage range of device |
| | | BORV1:BORV0 = 01 | 4.07 | — | 4.59 | V | |
| | | BORV1:BORV0 = 00 | 4.36 | — | 4.92 | 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 high-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 oscillator configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2 REXT (mA) with REXT in kOhm.

**5:** The LVD and BOR modules share a large portion of circuitry. The ΔIBOR and ΔILVD currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 27.1 DC Characteristics (Continued)

| PIC18LFXX8 (Industrial) | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18FXX8 (Industrial, Extended)** | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
| **Param No.** | **Symbol** | **Characteristic/ Device** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| | IDD | **Supply Current(2,3,4)** | | | | | |
| D010 | | PIC18LFXX8 | | | | | XT oscillator configuration |
| | | | — | .7 | 2 | mA | VDD = 2.0V, +25°C, FOSC = 4 MHz |
| | | | — | .7 | 2 | mA | VDD = 2.0V, -40°C to +85°C, FOSC = 4 MHz |
| | | | — | 1.7 | 4 | mA | VDD = 4.2V, -40°C to +85°C, FOSC = 4 MHz |
| | | | | | | | RC oscillator configuration |
| | | | — | 1 | 2.5 | mA | VDD = 2.0V, +25°C, FOSC = 4 MHz |
| | | | — | 1 | 2.5 | mA | VDD = 2.0V, -40°C to +85°C, FOSC = 4 MHz |
| | | | — | 2.5 | 5 | mA | VDD = 4.2V, -40°C to +85°C, FOSC = 4 MHz |
| | | | | | | | RCIO oscillator configuration |
| | | | — | .7 | 2.5 | mA | VDD = 2.0V, +25°C, FOSC = 4 MHz |
| | | | — | .7 | 2.5 | mA | VDD = 2.0V, -40°C to +85°C, FOSC = 4 MHz |
| | | | — | 1.8 | 4 | mA | VDD = 4.2V, -40°C to +85°C, FOSC = 4 MHz |
| D010 | | PIC18FXX8 | | | | | XT oscillator configuration |
| | | | — | 1.7 | 4 | mA | VDD = 4.2V, +25°C, FOSC = 4 MHz |
| | | | — | 1.7 | 4 | mA | VDD = 4.2V, -40°C to +85°C, FOSC = 4 MHz |
| | | | — | 1.7 | 4 | mA | VDD = 4.2V, -40°C to +125°C, FOSC = 4 MHz |
| | | | | | | | RC oscillator configuration |
| | | | — | 2.5 | 5 | mA | VDD = 4.2V, +25°C, FOSC = 4 MHz |
| | | | — | 2.5 | 5 | mA | VDD = 4.2V, -40°C to +85°C, FOSC = 4 MHz |
| | | | — | 2.5 | 6 | mA | VDD = 4.2V, -40°C to +125°C, FOSC = 4 MHz |
| | | | | | | | RCIO oscillator configuration |
| | | | — | 1.8 | 4 | mA | VDD = 4.2V, +25°C, FOSC = 4 MHz |
| | | | — | 1.8 | 5 | mA | VDD = 4.2V, -40°C to +85°C, FOSC = 4 MHz |
| | | | — | 1.8 | 5 | mA | VDD = 4.2V, -40°C to +125°C, FOSC = 4 MHz |
| D010A | | PIC18LFXX8 | | | | | LP oscillator, FOSC = 32 kHz, WDT disabled |
| | | | — | 18 | 40 | µA | VDD = 2.0V, -40°C to +85°C |
| D010A | | PIC18FXX8 | | | | | LP oscillator, FOSC = 32 kHz, WDT disabled |
| | | | — | 60 | 150 | µA | VDD = 4.2V, -40°C to +85°C |
| | | | — | 60 | 180 | µA | VDD = 4.2V, -40°C to +125°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
$\overline{MCLR}$ = 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 high-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 oscillator configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2 REXT (mA) with REXT in kOhm.

**5:** The LVD and BOR modules share a large portion of circuitry. The ΔIBOR and ΔILVD currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.

© 2006 Microchip Technology Inc.

chanhvinh.com



## 27.1 DC Characteristics (Continued)

| PIC18LFXX8 (Industrial) | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18FXX8 (Industrial, Extended)** | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
| **Param No.** | **Symbol** | **Characteristic/ Device** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| | IDD | **Supply Current(2,3,4)** | | | | | |
| D010C | | PIC18LFXX8 | — | 21 | 28 | mA | EC, ECIO oscillator configurations<br>VDD = 4.2V, -40°C to +85°C |
| D010C | | PIC18FXX8 | — | 21 | 30 | mA | EC, ECIO oscillator configurations<br>VDD = 4.2V, -40°C to +125°C,<br>FOSC = 25 MHz |
| D013 | | PIC18LFXX8 | —<br>—<br><br>— | 1.3<br>18<br><br>28 | 3<br>28<br><br>40 | mA<br>mA<br>HS + PLL osc configuration<br>mA | HS oscillator configurations<br>FOSC = 6 MHz, VDD = 2.0V<br>FOSC = 25 MHz, VDD = 5.5V<br><br>FOSC = 10 MHz, VDD = 5.5V |
| D013 | | PIC18FXX8 | —<br><br>— | 18<br><br>28 | 28<br>HS + PLL osc configuration<br>40 | mA<br><br>mA | HS oscillator configurations<br>FOSC = 25 MHz, VDD = 5.5V<br><br>FOSC = 10 MHz, VDD = 5.5V |
| D014 | | PIC18LFXX8 | — | 32 | 65 | µA | Timer1 oscillator configuration<br>FOSC = 32 kHz, VDD = 2.0V |
| D014 | | PIC18FXX8 | —<br>— | 62<br>62 | 250<br>310 | µA<br>µA | Timer1 oscillator configuration<br>FOSC = 32 kHz, VDD = 4.2V, -40°C to +85°C<br>FOSC = 32 kHz, VDD = 4.2V, -40°C to +125°C |
| | IPD | **Power-Down Current(3)** | | | | | |
| D020 | | PIC18LFXX8 | —<br>— | 0.3<br>2 | 4<br>10 | µA<br>µA | VDD = 2.0V, -40°C to +85°C<br>VDD = 4.2V, -40°C to +85°C |
| D020 | | PIC18FXX8 | — | 2 | 10 | µA | VDD = 4.2V, -40°C to +85°C |
| D021B | | | — | 6 | 40 | µA | VDD = 4.2V, -40°C to +125°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
$\overline{MCLR}$ = 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 high-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 oscillator configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2 REXT (mA) with REXT in kOhm.

**5:** The LVD and BOR modules share a large portion of circuitry. The ΔIBOR and ΔILVD currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 27.1 DC Characteristics (Continued)

| PIC18LFXX8 (Industrial) | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18FXX8 (Industrial, Extended)** | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
| **Param No.** | **Symbol** | **Characteristic/ Device** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| | ΔIWDT | **Module Differential Current** | | | | | |
| D022 | | **Watchdog Timer** | — | 0.75 | 1.5 | µA | VDD = 2.5V, +25°C |
| | | PIC18LFXX8 | — | 0.8 | 8 | µA | VDD = 2.0V, -40°C to +85°C |
| | | | — | 7 | 25 | µA | VDD = 4.2V, -40°C to +85°C |
| D022 | | **Watchdog Timer** | — | 7 | 25 | µA | VDD = 4.2V, +25°C |
| | | PIC18FXX8 | — | 7 | 25 | µA | VDD = 4.2V, -40°C to +85°C |
| | | | — | 7 | 45 | µA | VDD = 4.2V, -40°C to +125°C |
| D022A | ΔIBOR | **Brown-out Reset(5)** | — | 38 | 50 | µA | VDD = 2.0V, +25°C |
| | | PIC18LFXX8 | — | 42 | 55 | µA | VDD = 2.0V, -40°C to +85°C |
| | | | — | 49 | 65 | µA | VDD = 4.2V, -40°C to +85°C |
| D022A | | **Brown-out Reset(5)** | — | 46 | 65 | µA | VDD = 4.2V, +25°C |
| | | PIC18FXX8 | — | 49 | 65 | µA | VDD = 4.2V, -40°C to +85°C |
| | | | — | 50 | 75 | µA | VDD = 4.2V, -40°C to +125°C |
| D022B | ΔILVD | **Low-Voltage Detect(5)** | — | 36 | 50 | µA | VDD = 2.0V, +25°C |
| | | PIC18LFXX8 | — | 40 | 55 | µA | VDD = 2.0V, -40°C to +85°C |
| | | | — | 47 | 65 | µA | VDD = 4.2V, -40°C to +85°C |
| D022B | | **Low-Voltage Detect(5)** | — | 44 | 65 | µA | VDD = 4.2V, +25°C |
| | | PIC18FXX8 | — | 47 | 65 | µA | VDD = 4.2V, -40°C to +85°C |
| | | | — | 47 | 75 | µA | VDD = 4.2V, -40°C to +125°C |
| D025 | ΔITMR1 | **Timer1 Oscillator** | — | 6.2 | 40 | µA | VDD = 2.0V, +25°C |
| | | PIC18LFXX8 | — | 6.2 | 45 | µA | VDD = 2.0V, -40°C to +85°C |
| | | | — | 7.5 | 55 | µA | VDD = 4.2V, -40°C to +85°C |
| D025 | | **Timer1 Oscillator** | — | 7.5 | 55 | µA | VDD = 4.2V, +25°C |
| | | PIC18FXX8 | — | 7.5 | 55 | µA | VDD = 4.2V, -40°C to +85°C |
| | | | — | 7.5 | 65 | µA | VDD = 4.2V, -40°C to +125°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
$\overline{MCLR}$ = 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 high-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 oscillator configuration, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2 REXT (mA) with REXT in kOhm.

**5:** The LVD and BOR modules share a large portion of circuitry. The ΔIBOR and ΔILVD currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.

chanhvinh.com



## 27.2 DC Characteristics: PIC18FXX8 (Industrial, Extended) PIC18LFXX8 (Industrial)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic/ Device** | **Min** | **Max** | **Units** | **Conditions** |
| | VIL | **Input Low Voltage** | | | | |
| | | I/O ports: | | | | |
| D030 | | with TTL buffer | VSS | 0.15 VDD | V | VDD < 4.5V |
| D030A | | | — | 0.8 | V | 4.5V ≤ VDD ≤ 5.5V |
| D031 | | with Schmitt Trigger buffer | VSS | 0.2 VDD | V | |
| | | RC3 and RC4 | VSS | 0.3 VDD | V | |
| D032 | | $\overline{MCLR}$ | VSS | 0.2 VDD | V | |
| D032A | | OSC1 (in XT, HS and LP modes) and T1OSI | VSS | 0.3 VDD | V | |
| D033 | | OSC1 (in RC mode)(1) | VSS | 0.2 VDD | V | |
| | VIH | **Input High Voltage** | | | | |
| | | I/O ports: | | | | |
| D040 | | with TTL buffer | 0.25 VDD + 0.8V | VDD | V | VDD < 4.5V |
| D040A | | | 2.0 | VDD | V | 4.5V ≤ VDD ≤ 5.5V |
| D041 | | with Schmitt Trigger buffer | 0.8 VDD | VDD | V | |
| | | RC3 and RC4 | 0.7 VDD | VDD | V | |
| D042 | | $\overline{MCLR}$ | 0.8 VDD | VDD | V | |
| D042A | | OSC1 (in XT, HS and LP modes) and T1OSI | 0.7 VDD | VDD | V | |
| D043 | | OSC1 (RC mode)(1) | 0.9 VDD | VDD | V | |
| | IIL | **Input Leakage Current(2,3)** | | | | |
| D060 | | I/O ports | — | ±1 | µA | VSS ≤ VPIN ≤ VDD, Pin at high-impedance |
| D061 | | $\overline{MCLR}$ | — | ±5 | µA | Vss ≤ VPIN ≤ VDD |
| D063 | | OSC1 | — | ±5 | µA | Vss ≤ VPIN ≤ VDD |
| | IPU | **Weak Pull-up Current** | | | | |
| D070 | IPURB | PORTB weak pull-up current | 50 | 450 | µA | VDD = 5V, VPIN = VSS |

**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 $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**3:** Negative current is defined as current sourced by the pin.


© 2006 Microchip Technology Inc.

chanhvinh.com



## 27.2 DC Characteristics: PIC18FXX8 (Industrial, Extended) PIC18LFXX8 (Industrial) (Continued)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic/ Device** | **Min** | **Max** | **Units** | **Conditions** |
| | VOL | **Output Low Voltage** | | | | |
| D080 | | I/O ports | — | 0.6 | V | IOL = 8.5 mA, VDD = 4.2V, -40°C to +85°C |
| D080A | | | — | 0.6 | V | IOL = 7.0 mA, VDD = 4.2V, -40°C to +125°C |
| D083 | | OSC2/CLKO (RC mode) | — | 0.6 | V | IOL = 1.6 mA, VDD = 4.2V, -40°C to +85°C |
| D083A | | | — | 0.6 | V | IOL = 1.2 mA, VDD = 4.2V, -40°C to +125°C |
| | VOH | **Output High Voltage(3)** | | | | |
| D090 | | I/O ports | VDD – 0.7 | — | V | IOH = -3.0 mA, VDD = 4.2V, -40°C to +85°C |
| D090A | | | VDD – 0.7 | — | V | IOH = -2.5 mA, VDD = 4.2V, -40°C to +125°C |
| D092 | | OSC2/CLKO (RC mode) | VDD – 0.7 | — | V | IOH = -1.3 mA, VDD = 4.2V, -40°C to +85°C |
| D092A | | | VDD – 0.7 | — | V | IOH = -1.0 mA, VDD = 4.2V, -40°C to +125°C |
| D150 | VOD | **Open-Drain High Voltage** | — | 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 I2C™ 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 $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**3:** Negative current is defined as current sourced by the pin.

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-4: LOW-VOLTAGE DETECT CHARACTERISTICS**

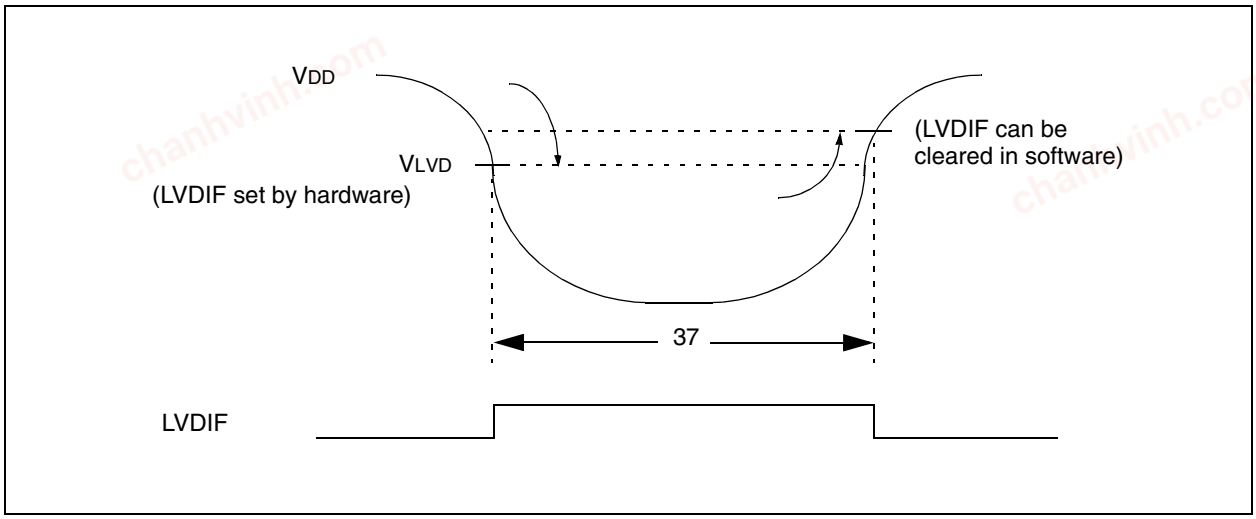



**TABLE 27-1: LOW-VOLTAGE DETECT CHARACTERISTICS**

| **Low-Voltage Detect Characteristics** | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| D420 | VLVD | LVD Voltage | LVV = 0001 | 1.96 | 2.06 | 2.16 | V | T ≥ 25°C |
| | | | LVV = 0010 | 2.16 | 2.27 | 2.38 | V | T ≥ 25°C |
| | | | LVV = 0011 | 2.35 | 2.47 | 2.59 | V | T ≥ 25°C |
| | | | LVV = 0100 | 2.43 | 2.58 | 2.69 | V | |
| | | | LVV = 0101 | 2.64 | 2.78 | 2.92 | V | |
| | | | LVV = 0110 | 2.75 | 2.89 | 3.03 | V | |
| | | | LVV = 0111 | 2.95 | 3.1 | 3.26 | V | |
| | | | LVV = 1000 | 3.24 | 3.41 | 3.58 | V | |
| | | | LVV = 1001 | 3.43 | 3.61 | 3.79 | V | |
| | | | LVV = 1010 | 3.53 | 3.72 | 3.91 | V | |
| | | | LVV = 1011 | 3.72 | 3.92 | 4.12 | V | |
| | | | LVV = 1100 | 3.92 | 4.13 | 4.34 | V | |
| | | | LVV = 1101 | 4.07 | 4.33 | 4.59 | V | |
| | | | LVV = 1110 | 4.36 | 4.64 | 4.92 | V | |


© 2006 Microchip Technology Inc.

chanhvinh.com



**TABLE 27-2: DC CHARACTERISTICS: EEPROM AND ENHANCED FLASH**

| DC Characteristics | | | Standard Operating Conditions | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| | | **Internal Program Memory Programming Specifications** | | | | | |
| D110 | VPP | Voltage on $\overline{MCLR}$/VPP pin | 9.00 | — | 13.25 | V | |
| D113 | IDDP | Supply Current during Programming | — | — | 10 | mA | |
| | | **Data EEPROM Memory** | | | | | |
| D120 | ED | Cell Endurance | 100K | 1M | — | E/W | -40°C to +85°C |
| D120A | ED | Byte Endurance | 10K | 100K | — | E/W | +85°C to +125°C |
| D121 | VDRW | VDD for Read/Write | VMIN | — | 5.5 | V | Using EECON to read/write VMIN = Minimum operating voltage |
| D122 | TDEW | Erase/Write Cycle Time | — | 4 | — | ms | |
| D123 | TRETD | Characteristic Retention | 40 | — | — | Year | Provided no other specifications are violated |
| D124 | TREF | Number of Total Erase/Write Cycles to Data EEPROM before Refresh* | 1M | 10M | — | Cycles | -40°C to +85°C |
| D124A | TREF | Number of Total Erase/Write Cycles before Refresh* | 100K | 1M | — | Cycles | +85°C to +125°C |
| | | **Program Flash Memory** | | | | | |
| D130 | EP | Cell Endurance | 10K | 100K | — | E/W | -40°C to +85°C |
| D130A | EP | Cell Endurance | 1000 | 10K | — | E/W | +85°C to +125°C |
| D131 | VPR | VDD for Read | VMIN | — | 5.5 | V | VMIN = Minimum operating voltage |
| D132 | VIE | VDD for Block Erase | 4.5 | — | 5.5 | V | Using ICSP™ port |
| D132A | VIW | VDD for Externally Timed Erase or Write | 4.5 | — | 5.5 | V | Using ICSP port |
| D132B | VPEW | VDD for Self-Timed Write | VMIN | — | 5.5 | V | VMIN = Minimum operating voltage |
| D133 | TIE | ICSP Erase Cycle Time | — | 4 | — | ms | VDD ≥ 4.5V |
| D133A | TIW | ICSP Erase or Write Cycle Time (externally timed) | 1 | — | — | ms | VDD ≥ 4.5V |
| D133A | TIW | Self-Timed Write Cycle Time | — | 2 | — | ms | |
| D134 | TRETD | Characteristic Retention | 40 | — | — | Year | Provided no other specifications are violated |

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

* See **Section 5.8 "Using the Data EEPROM"** for more information.

chanhvinh.com



**TABLE 27-3: COMPARATOR SPECIFICATIONS**

| **Operating Conditions:** $V_{DD}$ range as described in **Section 27.1 "DC Characteristics"**, -40°C < $T_A$ < +125°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D300 | $V_{IOFF}$ | Input Offset Voltage | — | ±5.0 | ±10 | mV | |
| D301 | $V_{ICM}$ | Input Common Mode Voltage | 0 | — | $V_{DD}$ – 1.5 | V | |
| D302 | CMRR | CMRR | +55* | — | — | db | |
| D300 | $T_{RESP}$ | Response Time(1) | — | 300*<br>350* | 400*<br>600* | ns<br>ns | PIC18FXX8<br>PIC18LFXX8 |
| D301 | $T_{MC2OV}$ | Comparator Mode Change to Output Valid | — | — | 10* | µs | |

* These parameters are characterized but not tested.

**Note 1:** Response time measured with one comparator input at ($V_{DD}$ – 1.5)/2 while the other input transitions from $V_{SS}$ to $V_{DD}$.

**TABLE 27-4: VOLTAGE REFERENCE SPECIFICATIONS**

| **Operating Conditions:** $V_{DD}$ range as described in **Section 27.1 "DC Characteristics"**, -40°C < $T_A$ < +125°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D310 | $V_{RES}$ | Resolution | $V_{DD}$/24 | — | $V_{DD}$/32 | LSB | |
| D311 | $V_{RAA}$ | Absolute Accuracy | — | — | 0.5 | LSB | |
| D312 | $V_{RUR}$ | Unit Resistor Value (R) | — | 2K* | — | Ω | |
| D310 | $T_{SET}$ | Settling Time(1) | — | — | 10* | µs | |

* These parameters are characterized but not tested.

**Note 1:** Settling time measured while CVRR = `1` and CVR<3:0> transitions from `0000` to `1111`.



© 2006 Microchip Technology Inc.

chanhvinh.com



## 27.3 AC (Timing) Characteristics

### 27.3.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created using one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I²C specifications only)
4. Ts (I²C 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 (High-Impedance) | V | Valid |
| L | Low | Z | High-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 | | |

chanhvinh.com



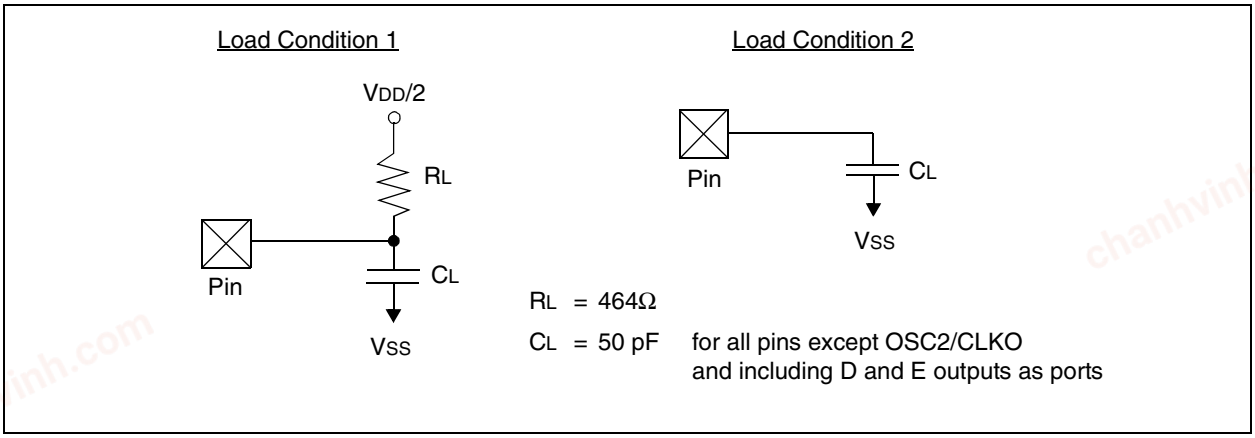
### 27.3.2 TIMING CONDITIONS

The temperature and voltages specified in Table 27-5 apply to all timing specifications unless otherwise noted. Figure 27-5 specifies the load conditions for the timing specifications.

**TABLE 27-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

| | |
|---|---|
| **AC CHARACTERISTICS** | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended<br>Operating voltage VDD range as described in DC specification, **Section 27.1 "DC Characteristics"**.<br>LF parts operate for industrial temperatures only. |

**FIGURE 27-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**





© 2006 Microchip Technology Inc.

chanhvinh.com



### 27.3.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 27-6: EXTERNAL CLOCK TIMING**

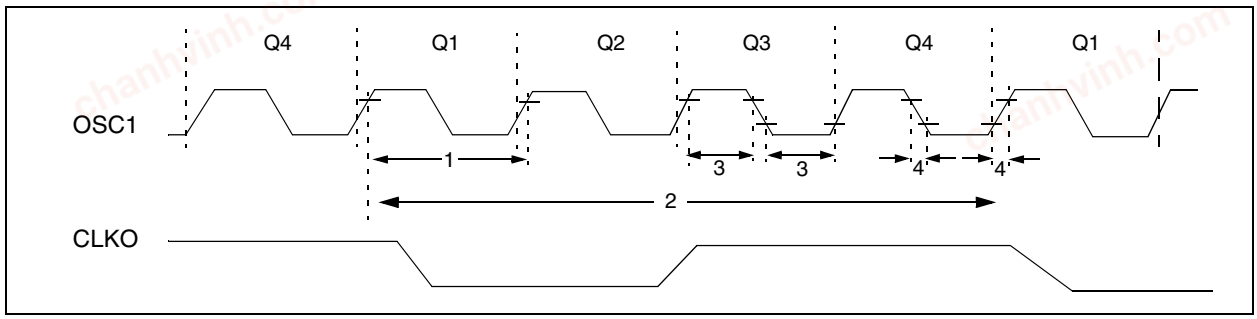



**TABLE 27-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 1A | FOSC | External CLKI Frequency(1) | DC | 40 | MHz | EC, ECIO oscillator, -40°C to +85°C |
| | | Oscillator Frequency(1) | DC | 25 | MHz | EC, ECIO oscillator, +85°C to +125°C |
| | | | DC | 4 | MHz | RC oscillator |
| | | | 0.1 | 4 | MHz | XT oscillator |
| | | | 4 | 25 | MHz | HS oscillator, -40°C to +85°C |
| | | | 4 | 25 | MHz | HS oscillator, +85°C to +125°C |
| | | | 4 | 10 | MHz | HS + PLL oscillator, -40°C to +85°C |
| | | | 4 | 6.25 | MHz | HS + PLL oscillator, +85°C to +125°C |
| | | | DC | 200 | kHz | LP oscillator |
| 1 | TOSC | External CLKI Period(1) | 25 | — | ns | EC, ECIO oscillator, -40°C to +85°C |
| | | Oscillator Period(1) | 40 | — | ns | EC, ECIO oscillator, +85°C to +125°C |
| | | | 250 | — | ns | RC oscillator |
| | | | 250 | 10,000 | ns | XT oscillator |
| | | | 40 | — | ns | HS oscillator, -40°C to +85°C |
| | | | 40 | — | ns | HS oscillator, +85°C to +125°C |
| | | | 100 | 250 | ns | HS + PLL oscillator, -40°C to +85°C |
| | | | 160 | 250 | ns | HS + PLL oscillator, +85°C to +125°C |
| | | | 5 | 200 | µs | LP oscillator |
| 2 | TCY | Instruction Cycle Time(1) | 100 | — | ns | TCY = 4/FOSC, -40°C to +85°C |
| | | | 160 | — | ns | TCY = 4/FOSC, +85°C to +125°C |
| 3 | TosL, TosH | External Clock in (OSC1) High or Low Time | 30 | — | ns | XT oscillator |
| | | | 2.5 | — | ns | LP oscillator |
| | | | 10 | — | µs | HS oscillator |
| 4 | TosR, TosF | External Clock in (OSC1) Rise or Fall Time | — | 20 | ns | XT oscillator |
| | | | — | 50 | ns | LP oscillator |
| | | | — | 7.5 | ns | HS oscillator |

**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.

© 2006 Microchip Technology Inc.

chanhvinh.com

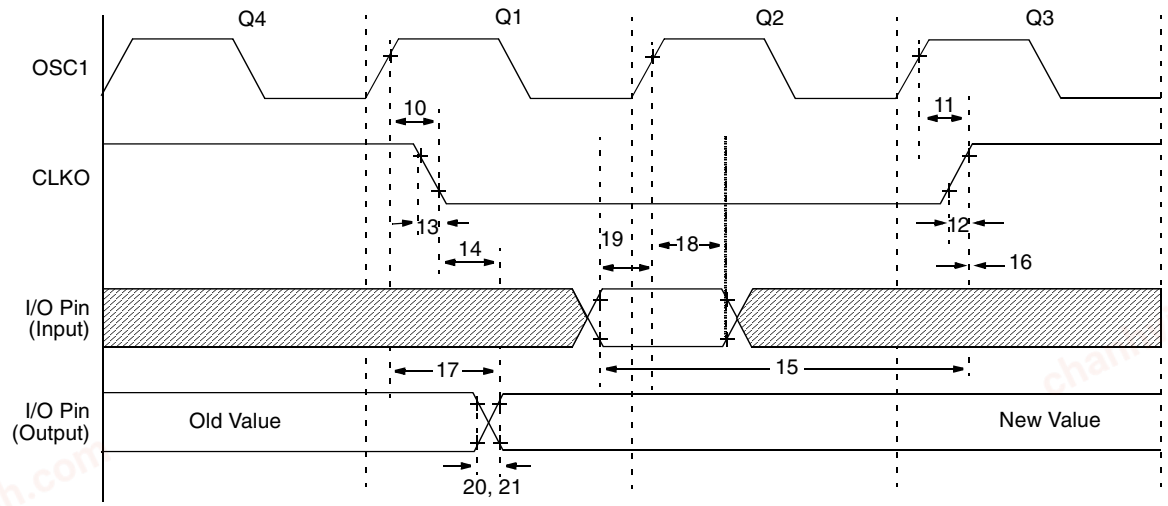


**TABLE 27-7: PLL CLOCK TIMING SPECIFICATIONS ($V_{DD}$ = 4.2 TO 5.5V)**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| — | FOSC | Oscillator Frequency Range | 4 | — | 10 | MHz | HS mode only |
| — | FSYS | On-Chip VCO System Frequency | 16 | — | 40 | MHz | HS mode only |
| — | $t_{rc}$ | PLL Start-up Time (Lock Time) | — | — | 2 | ms | |
| — | ΔCLK | CLKO Stability (Jitter) | -2 | — | +2 | % | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 27-7: CLKO AND I/O TIMING**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-8: CLKO AND I/O TIMING REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 10 | TosH2ckL | OSC1 ↑ to CLKO ↓ | | — | 75 | 200 | ns | **(1)** |
| 11 | TosH2ckH | OSC1 ↑ to CLKO ↑ | | — | 75 | 200 | ns | **(1)** |
| 12 | TckR | CLKO Rise Time | | — | 35 | 100 | ns | **(1)** |
| 13 | TckF | CLKO Fall Time | | — | 35 | 100 | ns | **(1)** |
| 14 | TckL2ioV | CLKO ↓ to Port Out Valid | | — | — | 0.5 TCY + 20 | ns | **(1)** |
| 15 | TioV2ckH | Port In Valid before CLKO ↑ | | 0.25 TCY + 25 | — | — | ns | **(1)** |
| 16 | TckH2ioI | Port In Hold after CLKO ↑ | | 0 | — | — | ns | **(1)** |
| 17 | TosH2ioV | OSC1 ↑ (Q1 cycle) to Port Out Valid | | — | 50 | 150 | ns | |
| 18 | TosH2ioI | OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time) | PIC18**F**XX8 | 100 | — | — | ns | |
| 18A | | | PIC18**LF**XX8 | 200 | — | — | ns | |
| 19 | TioV2osH | Port Input Valid to OSC1 ↑ (I/O in setup time) | | 0 | — | — | ns | |
| 20 | TioR | Port Output Rise Time | PIC18**F**XX8 | — | 10 | 25 | ns | |
| 20A | | | PIC18**LF**XX8 | — | — | 60 | ns | |
| 21 | TioF | Port Output Fall Time | PIC18**F**XX8 | — | 10 | 25 | ns | |
| 21A | | | PIC18**LF**XX8 | — | — | 60 | ns | |
| 22† | TINP | INT pin High or Low Time | | TCY | — | — | ns | |
| 23† | TRBP | RB7:RB4 Change INT High or Low Time | | TCY | — | — | 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.

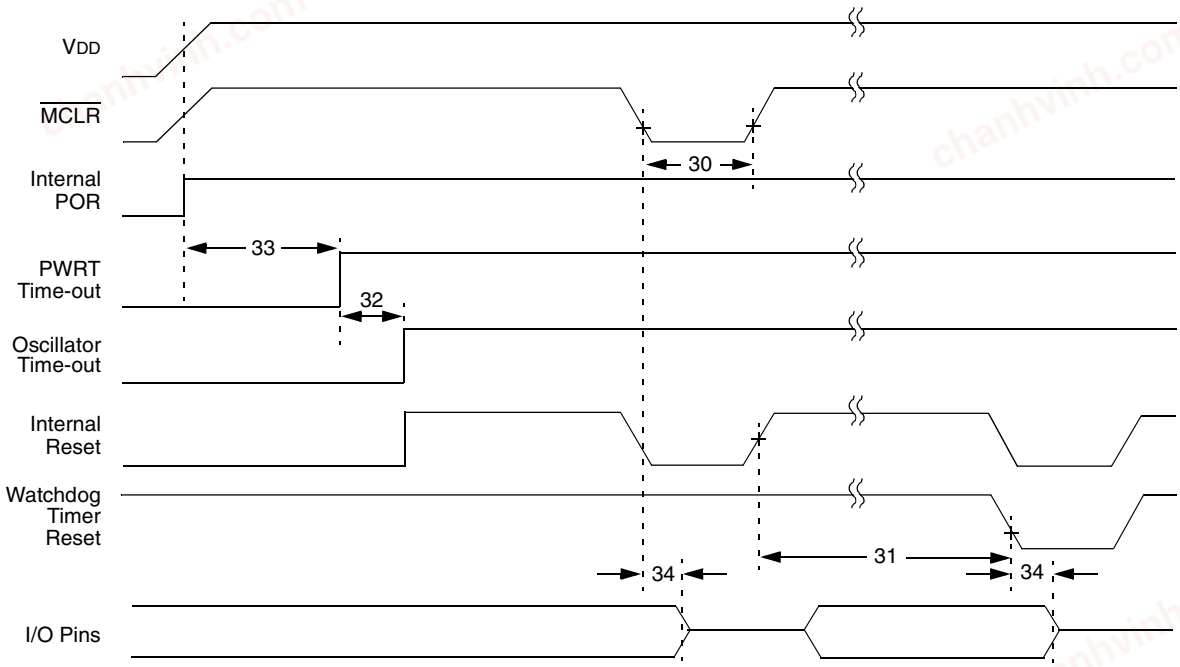
**Note 1:** Measurements are taken in RC mode where CLKO pin output is 4 x TOSC.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**




**Note:** Refer to Figure 27-5 for load conditions.

**FIGURE 27-9: BROWN-OUT RESET AND LOW-VOLTAGE DETECT TIMING**

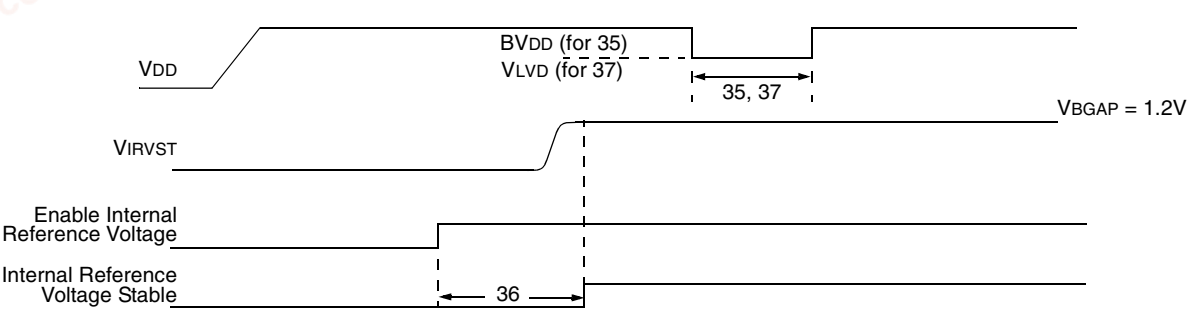



**TABLE 27-9: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, BROWN-OUT RESET AND LOW-VOLTAGE DETECT REQUIREMENTS**

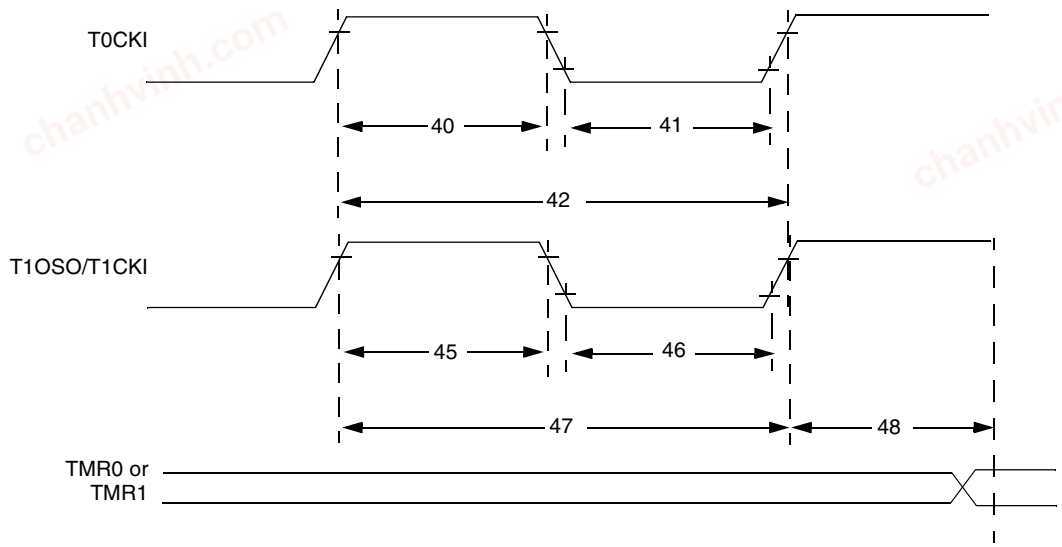
| Param No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 30 | TmcL | $\overline{MCLR}$ Pulse Width (low) | 2 | — | — | µs | |
| 31 | TWDT | Watchdog Timer Time-out Period (no prescaler) | 7 | 18 | 33 | ms | |
| 32 | TOST | Oscillation Start-up Timer Period | 1024 TOSC | — | 1024 TOSC | — | TOSC = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | 28 | 72 | 132 | ms | |
| 34 | TIOZ | I/O High-Impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset | — | 2 | — | µs | |
| 35 | TBOR | Brown-out Reset Pulse Width | 200 | — | — | µs | For VDD ≤ BVDD (see D005) |
| 36 | TIRVST | Time for Internal Reference Voltage to become stable | — | 20 | 50 | µs | |
| 37 | TLVD | Low-Voltage Detect Pulse Width | 200 | — | — | µs | For VDD ≤ VLVD (see D420) |

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-10: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

| Param No. | Symbol | Characteristic | | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 40 | Tt0H | T0CKI High Pulse Width | | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | | With prescaler | 10 | — | ns | |
| 41 | Tt0L | T0CKI Low Pulse Width | | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | | With prescaler | 10 | — | ns | |
| 42 | Tt0P | T0CKI Period | | No prescaler | TCY + 10 | — | ns | |
| | | | | With prescaler | Greater of: 20 ns or TCY + 40 / N | — | ns | N = prescale value (1, 2, 4,..., 256) |
| 45 | Tt1H | T1CKI High Time | Synchronous, no prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | Synchronous, with prescaler | PIC18**F**XX8 | 10 | — | ns | |
| | | | | PIC18**LF**XX8 | 25 | — | ns | |
| | | | Asynchronous | PIC18**F**XX8 | 30 | — | ns | |
| | | | | PIC18**LF**XX8 | 50 | — | ns | |
| 46 | Tt1L | T1CKI Low Time | Synchronous, no prescaler | | 0.5 TCY + 5 | — | ns | |
| | | | Synchronous, with prescaler | PIC18**F**XX8 | 10 | — | ns | |
| | | | | PIC18**LF**XX8 | 25 | — | ns | |
| | | | Asynchronous | PIC18**F**XX8 | 30 | — | ns | |
| | | | | PIC18**LF**XX8 | TBD | TBD | ns | |
| 47 | Tt1P | T1CKI Input Period | Synchronous | | Greater of: 20 ns or TCY + 40 / N | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | | 60 | — | ns | |
| | Ft1 | T1CKI Oscillator Input Frequency Range | | | DC | 50 | kHz | |
| 48 | Tcke2tmrl | Delay from External T1CKI Clock Edge to Timer Increment | | | 2 TOSC | 7 TOSC | — | |

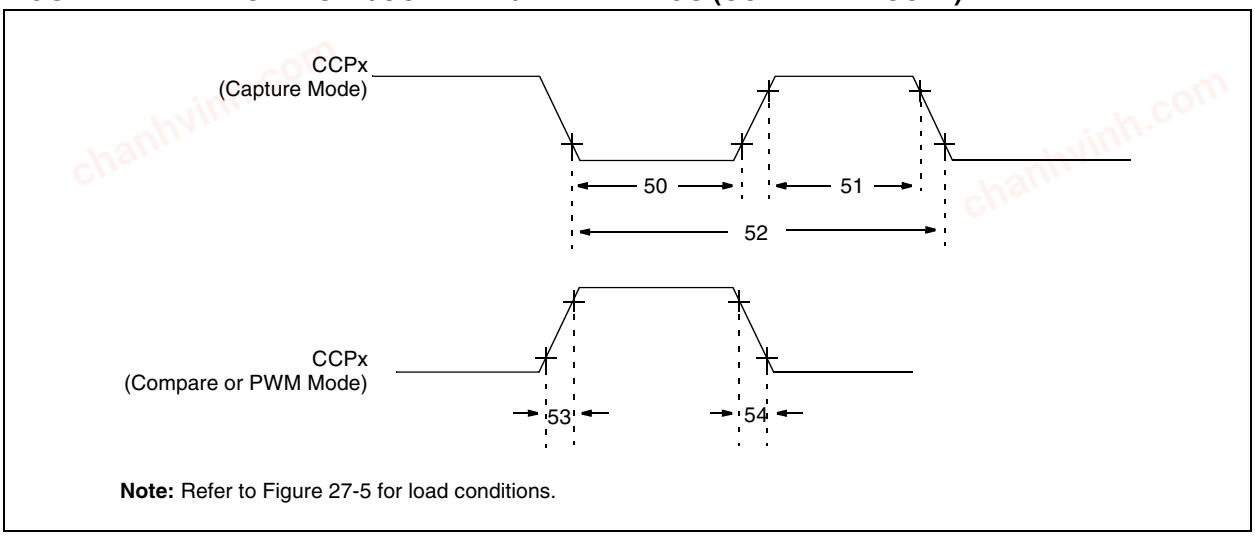
**Legend:** TBD = To Be Determined


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-11: CAPTURE/COMPARE/PWM TIMINGS (CCP1 AND ECCP1)**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-11: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP1 AND ECCP1)**

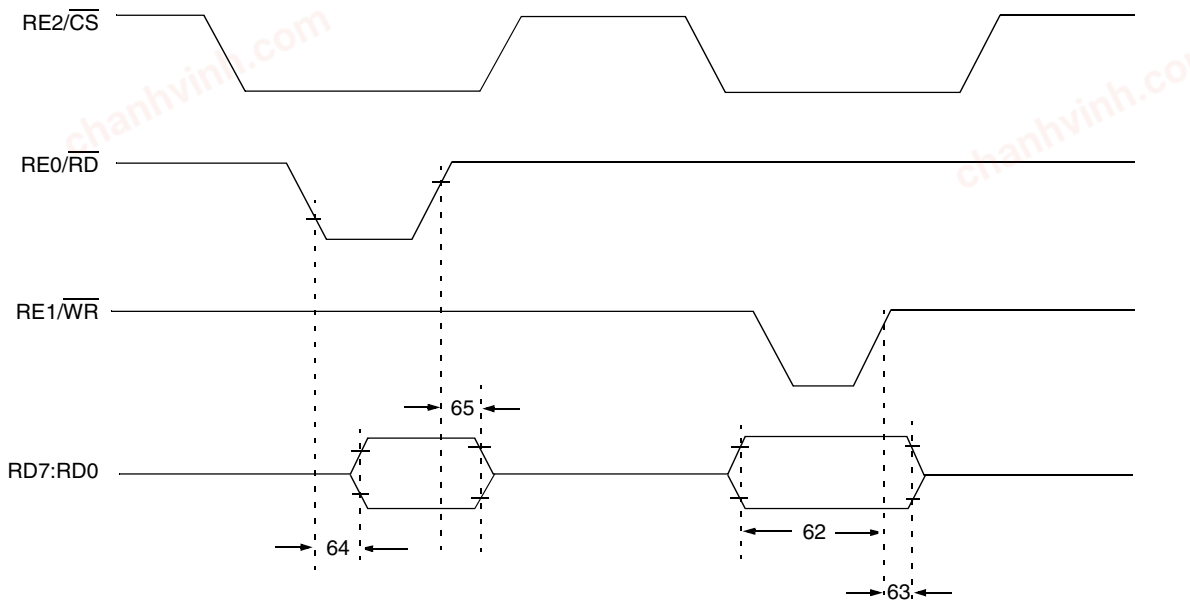
| Param No. | Symbol | Characteristic | | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 50 | TccL | CCPx Input Low Time | No prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | PIC18**F**XX8 | 10 | — | ns | |
| | | | | PIC18**LF**XX8 | 20 | — | ns | |
| 51 | TccH | CCPx Input High Time | No prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | PIC18**F**XX8 | 10 | — | ns | |
| | | | | PIC18**LF**XX8 | 20 | — | ns | |
| 52 | TccP | CCPx Input Period | | | $\frac{3\ T_{CY} + 40}{N}$ | — | ns | N = prescale value (1, 4 or 16) |
| 53 | TccR | CCPx Output Fall Time | | PIC18**F**XX8 | — | 25 | ns | |
| | | | | PIC18**LF**XX8 | — | 45 | ns | |
| 54 | TccF | CCPx Output Fall Time | | PIC18**F**XX8 | — | 25 | ns | |
| | | | | PIC18**LF**XX8 | — | 45 | ns | |

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-12: PARALLEL SLAVE PORT TIMING (PIC18F248 AND PIC18F458)**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-12: PARALLEL SLAVE PORT REQUIREMENTS (PIC18F248 AND PIC18F458)**

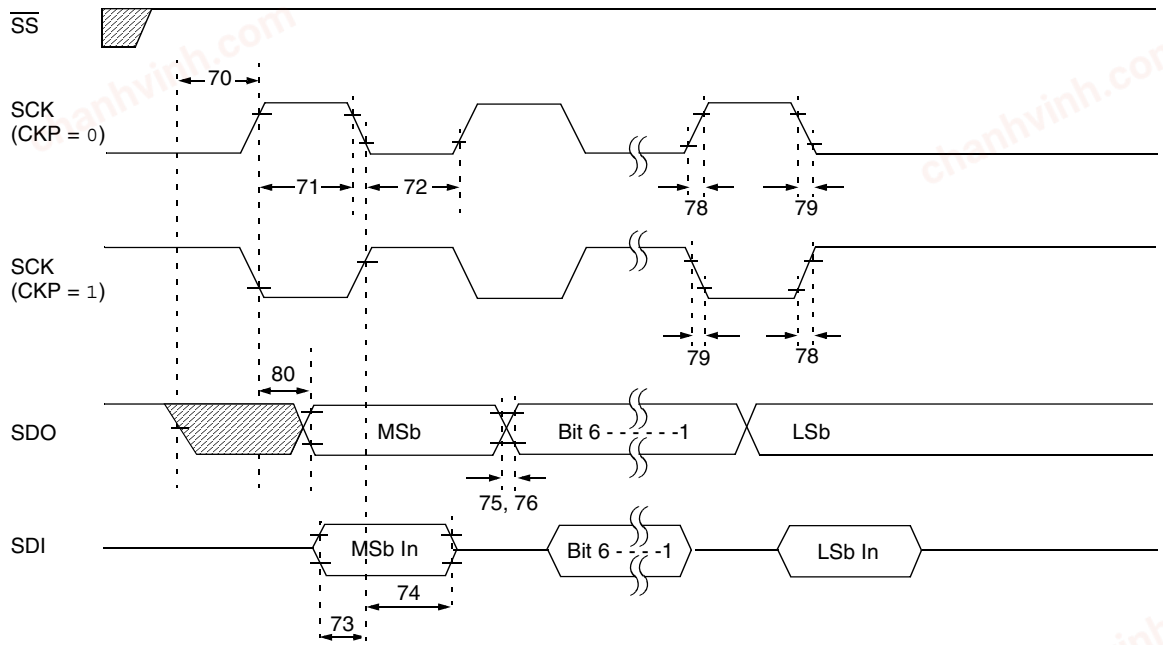
| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 62 | TdtV2wrH | Data-In Valid before $\overline{WR}$ ↑ or $\overline{CS}$ ↑ (setup time) | | 20<br>25 | —<br>— | ns<br>ns | <br>Extended Temp. range |
| 63 | TwrH2dtI | $\overline{WR}$ ↑ or $\overline{CS}$ ↑ to Data-In Invalid (hold time) | PIC18**F**XX8 | 20 | — | ns | |
| | | | PIC18**LF**XX8 | 35 | — | ns | |
| 64 | TrdL2dtV | $\overline{RD}$ ↓ and $\overline{CS}$ ↓ to Data-Out Valid | | —<br>— | 80<br>90 | ns<br>ns | <br>Extended Temp. range |
| 65 | TrdH2dtI | $\overline{RD}$ ↑ or $\overline{CS}$ ↓ to Data-Out Invalid | | 10 | 30 | ns | |
| 66 | TibfINH | Inhibit the IBF flag bit being cleared from $\overline{WR}$ ↑ or $\overline{CS}$ ↑ | | — | 3 TCY | ns | |


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-13: EXAMPLE SPI™ MASTER MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-13: EXAMPLE SPI™ MODE REQUIREMENTS (MASTER MODE, CKE = 0)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input | | TCY | — | ns | |
| 71 | TscH | SCK Input High Time (Slave mode) | Continuous | 1.25 TCY + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCK Input Low Time (Slave mode) | Continuous | 1.25 TCY + 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 Byte 1 to the 1st Clock Edge of Byte 2 | | 1.5 TCY + 40 | — | ns | **(Note 2)** |
| 74 | TscH2diL, TscL2diL | Hold Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 45 | ns | |
| 76 | TdoF | SDO Data Output Fall Time | | — | 25 | ns | |
| 78 | TscR | SCK Output Rise Time (Master mode) | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 45 | ns | |
| 79 | TscF | SCK Output Fall Time (Master mode) | | — | 25 | ns | |
| 80 | TscH2doV, TscL2doV | SDO Data Output Valid after SCK Edge | PIC18**F**XX8 | — | 50 | ns | |
| | | | PIC18**LF**XX8 | — | 100 | ns | |

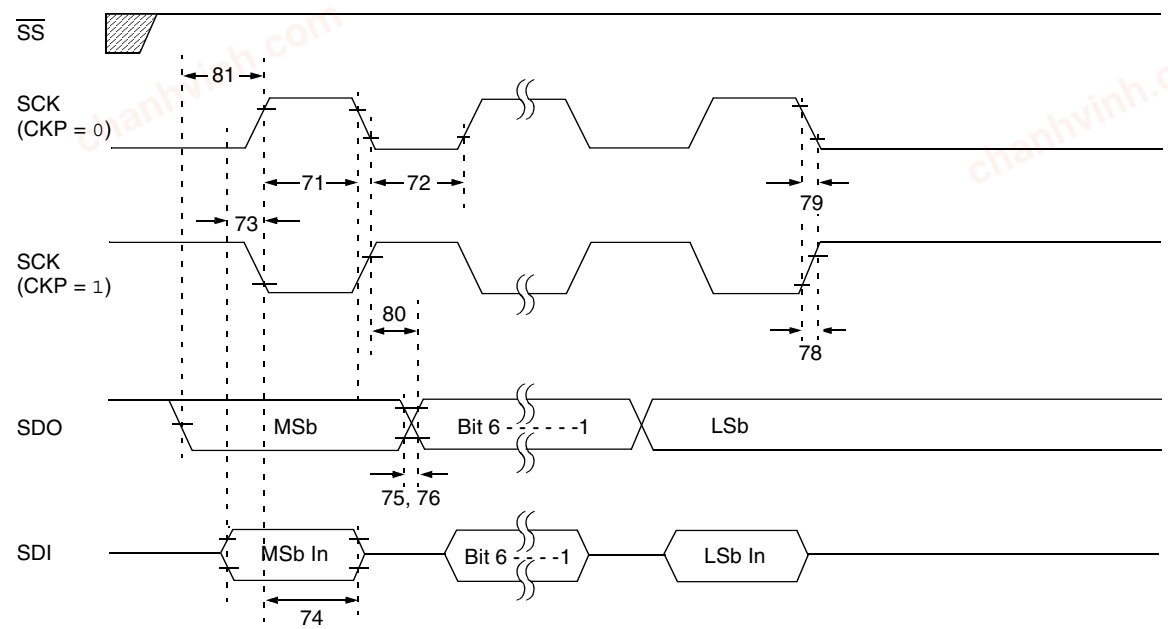
**Note 1:** Requires the use of parameter #73A.

**2:** Only if parameter #71A and #72A are used.

chanhvinh.com



**FIGURE 27-14: EXAMPLE SPI™ MASTER MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-14: 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 | 1.25 TCY + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCK Input Low Time (Slave mode) | Continuous | 1.25 TCY + 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 Byte 1 to the 1st Clock Edge of Byte 2 | | 1.5 TCY + 40 | — | ns | **(Note 2)** |
| 74 | TscH2diL, TscL2diL | Hold Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 45 | ns | |
| 76 | TdoF | SDO Data Output Fall Time | | — | 25 | ns | |
| 78 | TscR | SCK Output Rise Time (Master mode) | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 45 | ns | |
| 79 | TscF | SCK Output Fall Time (Master mode) | | — | 25 | ns | |
| 80 | TscH2doV, TscL2doV | SDO Data Output Valid after SCK Edge | PIC18**F**XX8 | — | 50 | ns | |
| | | | PIC18**LF**XX8 | — | 100 | ns | |
| 81 | TdoV2scH, TdoV2scL | SDO Data Output Setup to SCK Edge | | TCY | — | ns | |

**Note 1:** Requires the use of parameter #73A.

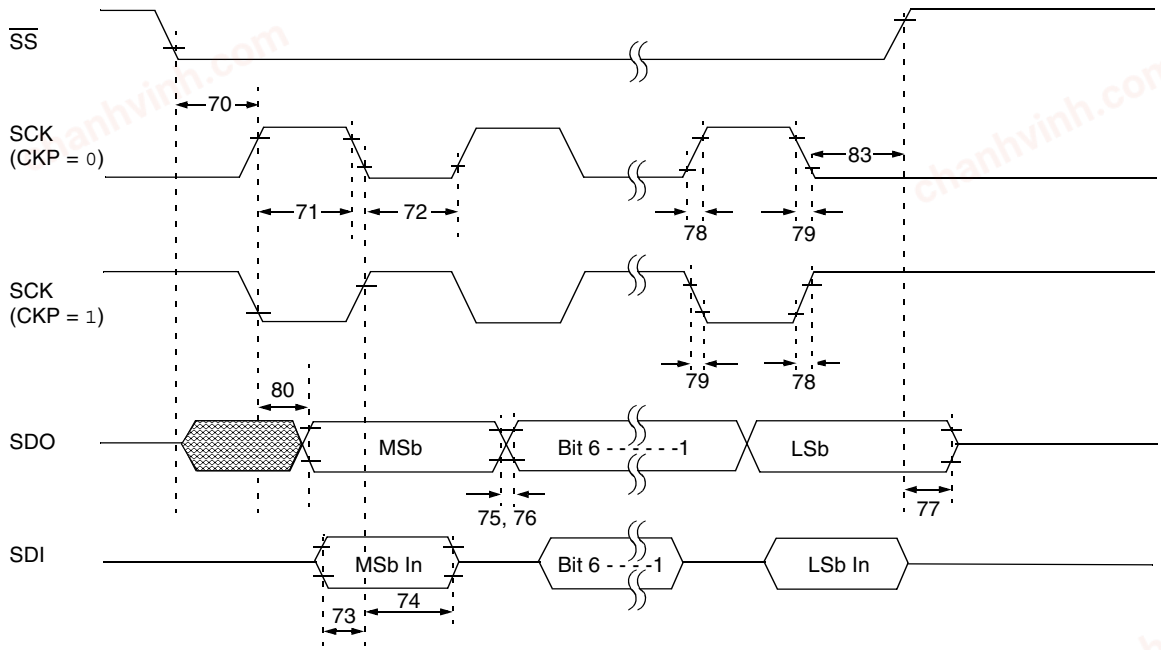
**2:** Only if parameter #71A and #72A are used.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-15: EXAMPLE SPI™ SLAVE MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-15: EXAMPLE SPI™ MODE REQUIREMENTS, SLAVE MODE TIMING (CKE = 0)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input | | TCY | — | ns | |
| 71 | TscH | SCK Input High Time (Slave mode) | Continuous | 1.25 TCY + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCK Input Low Time (Slave mode) | Continuous | 1.25 TCY + 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 Byte 1 to the 1st Clock Edge of Byte 2 | | 1.5 TCY + 40 | — | ns | **(Note 2)** |
| 74 | TscH2diL, TscL2diL | Hold Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | | 45 | ns | |
| 76 | TdoF | SDO Data Output Fall Time | | — | 25 | ns | |
| 77 | TssH2doZ | $\overline{SS}$ ↑ to SDO Output High-Impedance | | 10 | 50 | ns | |
| 78 | TscR | SCK Output Rise Time (Master mode) | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | | 45 | ns | |
| 79 | TscF | SCK Output Fall Time (Master mode) | | — | 25 | ns | |
| 80 | TscH2doV, TscL2doV | SDO Data Output Valid after SCK Edge | PIC18**F**XX8 | — | 50 | ns | |
| | | | PIC18**LF**XX8 | | 100 | ns | |
| 83 | TscH2ssH, TscL2ssH | $\overline{SS}$ ↑ after SCK Edge | | 1.5 TCY + 40 | — | ns | |

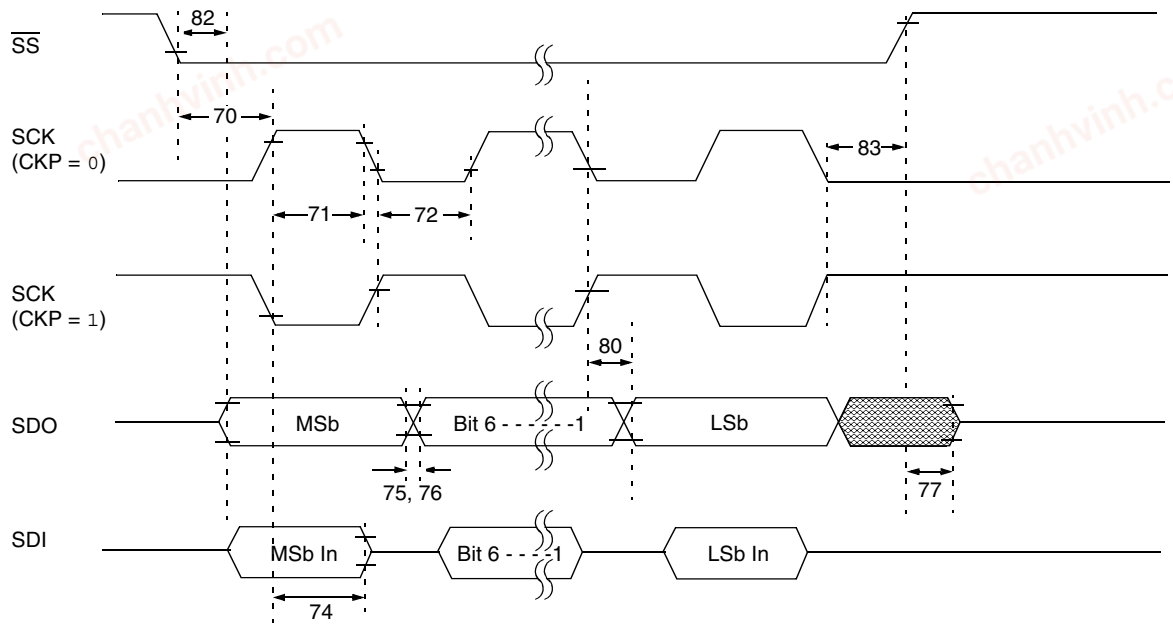
**Note 1:** Requires the use of parameter #73A.
**2:** Only if parameter #71A and #72A are used.

© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-16: EXAMPLE SPI™ SLAVE MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-16: EXAMPLE SPI™ SLAVE MODE REQUIREMENTS (CKE = 1)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input | | TCY | — | ns | |
| 71 | TscH | SCK Input High Time (Slave mode) | Continuous | 1.25 TCY + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCK Input Low Time (Slave mode) | Continuous | 1.25 TCY + 30 | — | ns | |
| 72A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 73A | TB2B | Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2 | | 1.5 TCY + 40 | — | ns | **(Note 2)** |
| 74 | TscH2diL, TscL2diL | Hold Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 45 | ns | |
| 76 | TdoF | SDO Data Output Fall Time | | — | 25 | ns | |
| 77 | TssH2doZ | $\overline{SS}$ ↑ to SDO Output High-Impedance | | 10 | 50 | ns | |
| 78 | TscR | SCK Output Rise Time (Master mode) | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 45 | ns | |
| 79 | TscF | SCK Output Fall Time (Master mode) | | — | 25 | ns | |
| 80 | TscH2doV, TscL2doV | SDO Data Output Valid after SCK Edge | PIC18**F**XX8 | — | 50 | ns | |
| | | | PIC18**LF**XX8 | — | 100 | ns | |
| 82 | TssL2doV | SDO Data Output Valid after $\overline{SS}$ ↓ Edge | PIC18**F**XX8 | — | 50 | ns | |
| | | | PIC18**LF**XX8 | — | 100 | ns | |
| 83 | TscH2ssH, TscL2ssH | $\overline{SS}$ ↑ after SCK Edge | | 1.5 TCY + 40 | — | ns | |

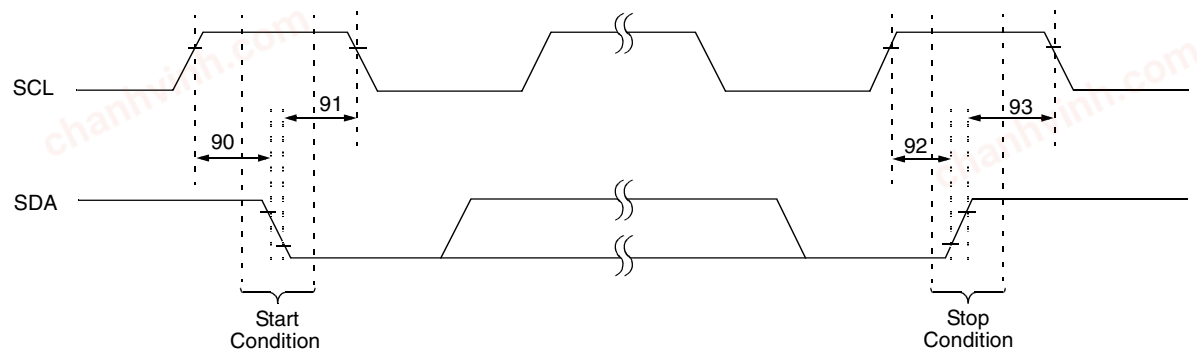
**Note 1:** Requires the use of parameter #73A.
**2:** Only if parameter #71A and #72A are used.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-17: I²C™ BUS START/STOP BITS TIMING**

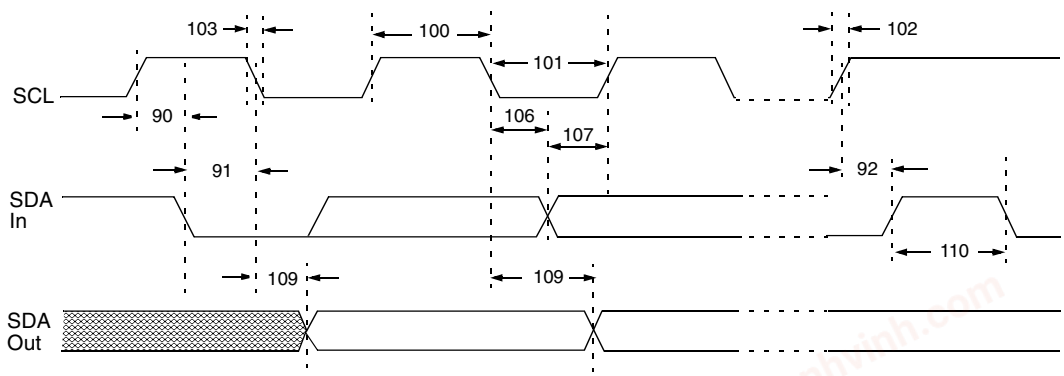



**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-17: I²C™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 90 | TSU:STA | Start Condition | 100 kHz mode | 4700 | — | ns | Only relevant for Repeated Start condition |
| | | Setup Time | 400 kHz mode | 600 | — | | |
| 91 | THD:STA | Start Condition | 100 kHz mode | 4000 | — | ns | After this period, the first clock pulse is generated |
| | | Hold Time | 400 kHz mode | 600 | — | | |
| 92 | TSU:STO | Stop Condition | 100 kHz mode | 4700 | — | ns | |
| | | Setup Time | 400 kHz mode | 600 | — | | |
| 93 | THD:STO | Stop Condition | 100 kHz mode | 4000 | — | ns | |
| | | Hold Time | 400 kHz mode | 600 | — | | |

**FIGURE 27-18: I²C™ BUS DATA TIMING**




**Note:** Refer to Figure 27-5 for load conditions.

chanhvinh.com



**TABLE 27-18: I²C™ BUS DATA REQUIREMENTS (SLAVE MODE)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 4.0 | — | µs | PIC18FXX8 must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 0.6 | — | µs | PIC18FXX8 must operate at a minimum of 10 MHz |
| | | | SSP Module | 1.5 TCY | — | | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 4.7 | — | µs | PIC18FXX8 must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1.3 | — | µs | PIC18FXX8 must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5 TCY | — | ns | |
| 102 | TR | SDA and SCL Rise Time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 103 | TF | SDA and SCL Fall Time | 100 kHz mode | — | 300 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 4.7 | — | µs | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 0.6 | — | µs | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 4.0 | — | µs | After this period the first clock pulse is generated |
| | | | 400 kHz mode | 0.6 | — | µs | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | µs | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 4.7 | — | µs | |
| | | | 400 kHz mode | 0.6 | — | µs | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | **(Note 1)** |
| | | | 400 kHz mode | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | µs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | µs | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

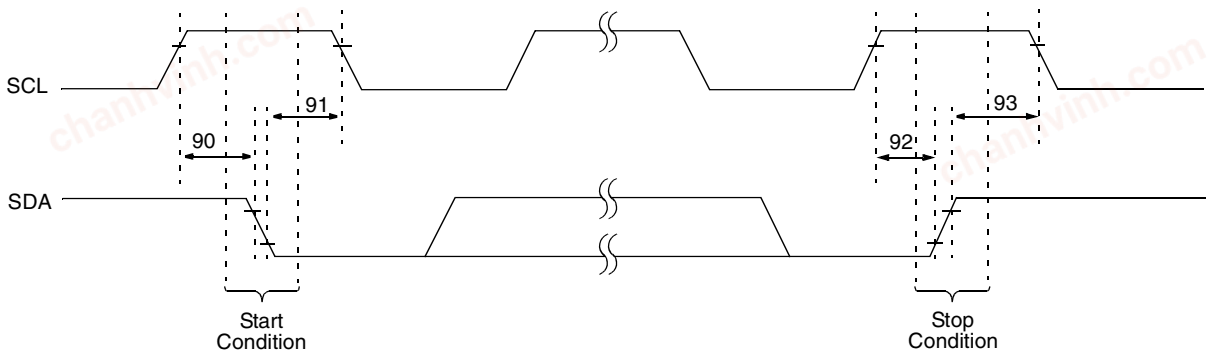
**2:** A Fast mode I²C™ bus device can be used in a Standard mode I²C bus system, but the requirement TSU;DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line.
Before the SCL line is released, TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification).


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-19: MASTER SSP I²C™ BUS START/STOP BITS TIMING WAVEFORMS**



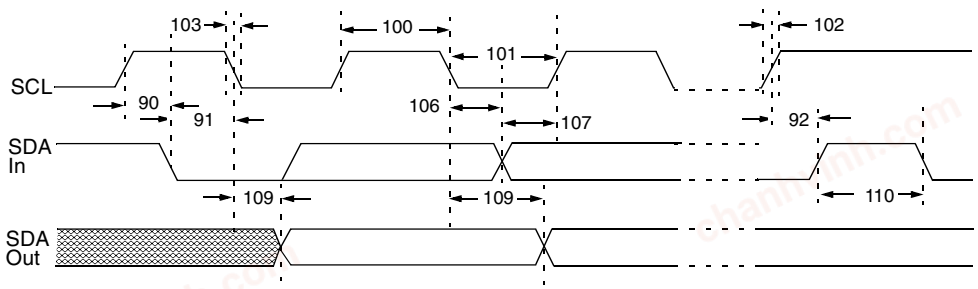

**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-19: MASTER SSP I²C™ BUS START/STOP BITS REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 93 | THD:STO | Stop Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C™ pins.

**FIGURE 27-20: MASTER SSP I²C™ BUS DATA TIMING**




**Note:** Refer to Figure 27-5 for load conditions.

chanhvinh.com



**TABLE 27-20: MASTER SSP I²C™ BUS DATA REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 102 | TR | SDA and SCL Rise Time | 100 kHz mode | — | 1000 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode(1) | — | 300 | ns | |
| 103 | TF | SDA and SCL Fall Time | 100 kHz mode | — | 300 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode(1) | — | 100 | ns | |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | ms | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | |
| | | | 400 kHz mode | — | 1000 | ns | |
| | | | 1 MHz mode(1) | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | ms | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | ms | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C™ pins.

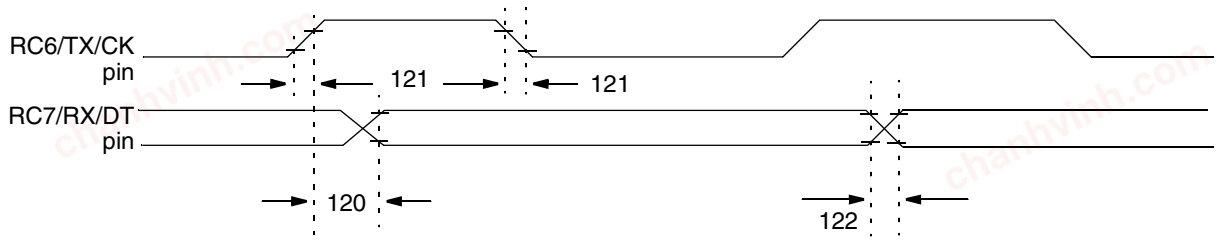
**2:** A Fast mode I²C bus device can be used in a Standard mode I²C bus system, but parameter #107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line.
Before the SCL line is released, parameter #102 + parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode).


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-21: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

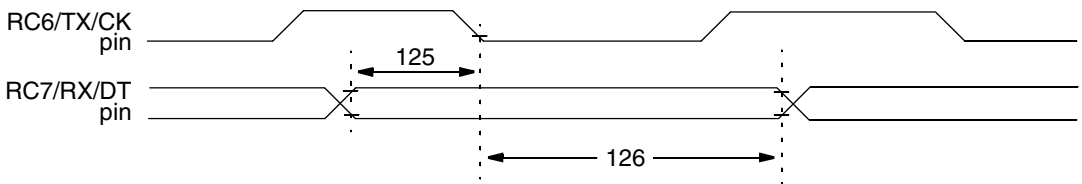



**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-21: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 120 | TckH2dtV | SYNC XMIT (Master & Slave)<br>Clock High to Data-Out Valid | PIC18**F**XX8 | — | 50 | ns | |
| | | | PIC18**LF**XX8 | — | 150 | ns | |
| 121 | Tckrf | Clock Out Rise Time and Fall Time (Master mode) | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 60 | ns | |
| 122 | Tdtrf | Data-Out Rise Time and Fall Time | PIC18**F**XX8 | — | 25 | ns | |
| | | | PIC18**LF**XX8 | — | 60 | ns | |

**FIGURE 27-22: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**




**Note:** Refer to Figure 27-5 for load conditions.

**TABLE 27-22: USART SYNCHRONOUS RECEIVE REQUIREMENTS**

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 125 | TdtV2ckl | SYNC RCV (Master & Slave)<br>Data-Hold before CK ↓ (DT hold time) | 10 | — | ns | |
| 126 | TckL2dtl | Data-Hold after CK ↓ (DT hold time) | 15 | — | ns | |

chanhvinh.com



**TABLE 27-23: A/D CONVERTER CHARACTERISTICS: PIC18FXX8 (INDUSTRIAL, EXTENDED) PIC18LFXX8 (INDUSTRIAL)**

| Param No. | Symbol | Characteristic | | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| A01 | NR | Resolution | | — | — | 10 | bit | VREF = VDD ≥ 3.0V |
| A03 | EIL | Integral Linearity Error | | — | — | <±1 | LSb | VREF = VDD ≥ 3.0V |
| A04 | EDL | Differential Linearity Error | | — | — | <±1 | LSb | VREF = VDD ≥ 3.0V |
| A05 | EFS | Full Scale Error | | — | — | <±1 | LSb | VREF = VDD ≥ 3.0V |
| A06 | EOFF | Offset Error | | — | — | <±1.5 | LSb | VREF = VDD ≥ 3.0V |
| A10 | — | Monotonicity(3) | | guaranteed | | | — | VSS ≤ VAIN ≤ VREF |
| A20 | VREF | Reference Voltage (VREFH – VREFL) | | 0V | — | — | V | |
| A20A | | | | 3V | — | — | V | For 10-bit resolution |
| A21 | VREFH | Reference Voltage High | | VSS | — | VDD + 0.3V | V | |
| A22 | VREFL | Reference Voltage Low | | VSS – 0.3V | — | VDD | V | |
| A25 | VAIN | Analog Input Voltage | | VSS – 0.3V | — | VREF + 0.3V | V | |
| A30 | ZAIN | Recommended Impedance of Analog Voltage Source | | — | — | 10.0 | kΩ | |
| A40 | IAD | A/D Conversion Current (VDD) | PIC18**F**XX8 | — | 180 | — | µA | Average current consumption when A/D is on **(Note 1)** |
| | | | PIC18**LF**XX8 | — | 90 | — | µA | |
| A50 | IREF | VREF Input Current **(Note 2)** | | 0 | — | 5 | µA | During VAIN acquisition. Based on differential of VHOLD to VAIN. To charge CHOLD. |
| | | | | — | — | 150 | µA | During A/D conversion cycle. |

**Note 1:** When A/D is off, it will not consume any current other than minor leakage current. The power-down current specification includes any such leakage from the A/D module.
VREF current is from RA2/AN2/VREF- and RA3/AN3/VREF+ pins or VDD and VSS pins, whichever is selected as reference input.

**2:** VSS ≤ VAIN ≤ VREF

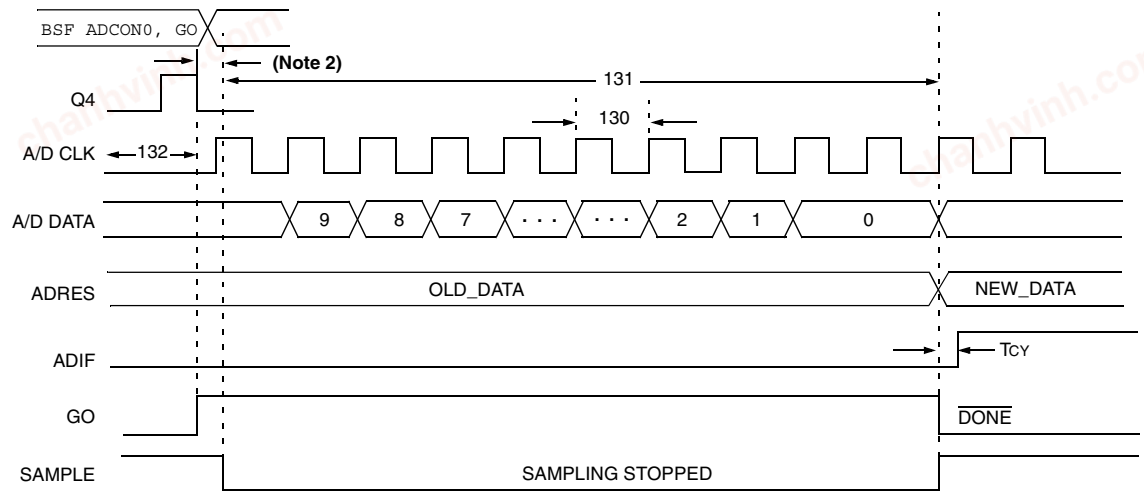
**3:** The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 27-23: A/D CONVERSION TIMING**




**Note 1:** If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.
**2:** This is a minimal RC delay (typically 100 ns) which also disconnects the holding capacitor from the analog input.

**TABLE 27-24: A/D CONVERSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 130 | TAD | A/d Clock Period | PIC18**F**XX8 | 1.6 | 20(5) | µs | TOSC based, VREF ≥ 3.0V |
| | | | PIC18**LF**XX8 | 3.0 | 20(5) | µs | TOSC based, VREF full range |
| | | | PIC18**F**XX8 | 2.0 | 6.0 | µs | A/D RC mode |
| | | | PIC18**LF**XX8 | 3.0 | 9.0 | µs | A/D RC mode |
| 131 | TCNV | Conversion Time (not including acquisition time) **(Note 1)** | | 11 | 12 | TAD | |
| 132 | TACQ | Acquisition Time **(Note 3)** | | 15<br>10 | —<br>— | µs<br>µs | -40°C ≤ Temp ≤ +125°C<br>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 TCY cycle.
**2:** See **Section 20.0 "Compatible 10-Bit Analog-to-Digital Converter (A/D) Module"** for minimum conditions when input voltage has changed more than 1 LSb.
**3:** The time for the holding capacitor to acquire the "New" input voltage when the voltage changes full scale after the conversion (AVDD to AVSS or AVSS to AVDD). The source impedance (*RS*) on the input channels is 50Ω.
**4:** On the next Q4 cycle of the device clock.
**5:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

© 2006 Microchip Technology Inc.

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## 28.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

**Note:** The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3σ) or (mean – 3σ) respectively, where σ is a standard deviation, over the whole temperature range.

**FIGURE 28-1: TYPICAL IDD vs. FOSC OVER VDD (HS MODE)**

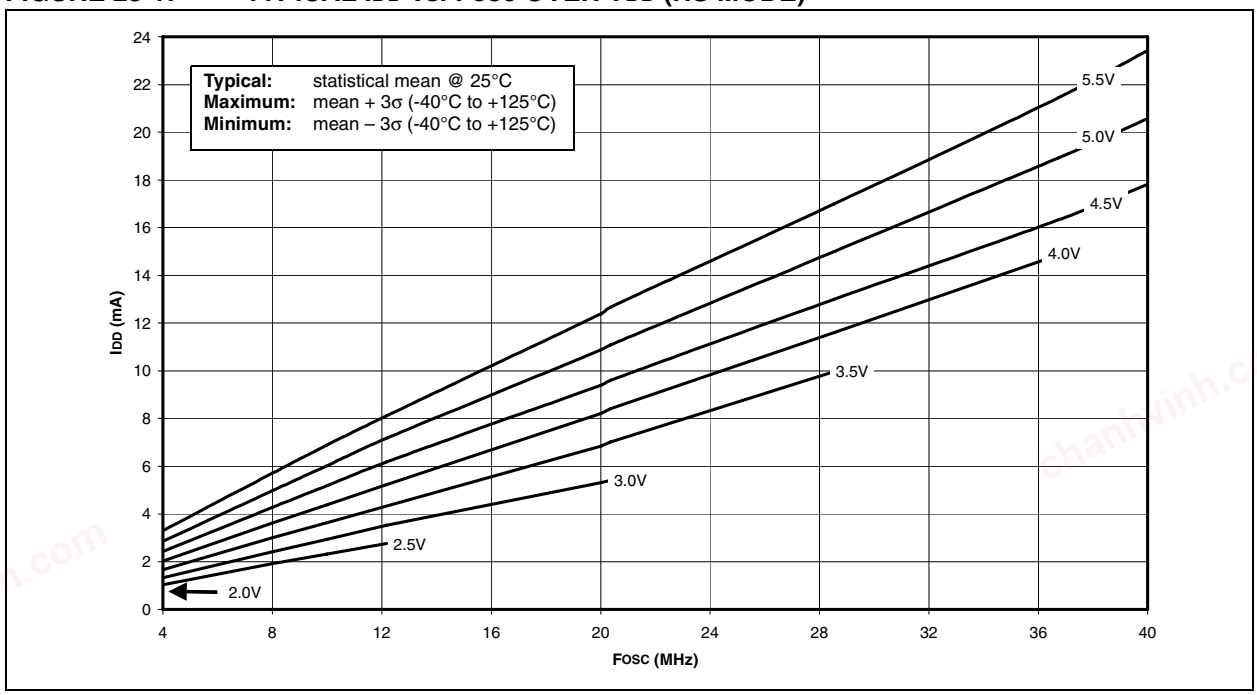
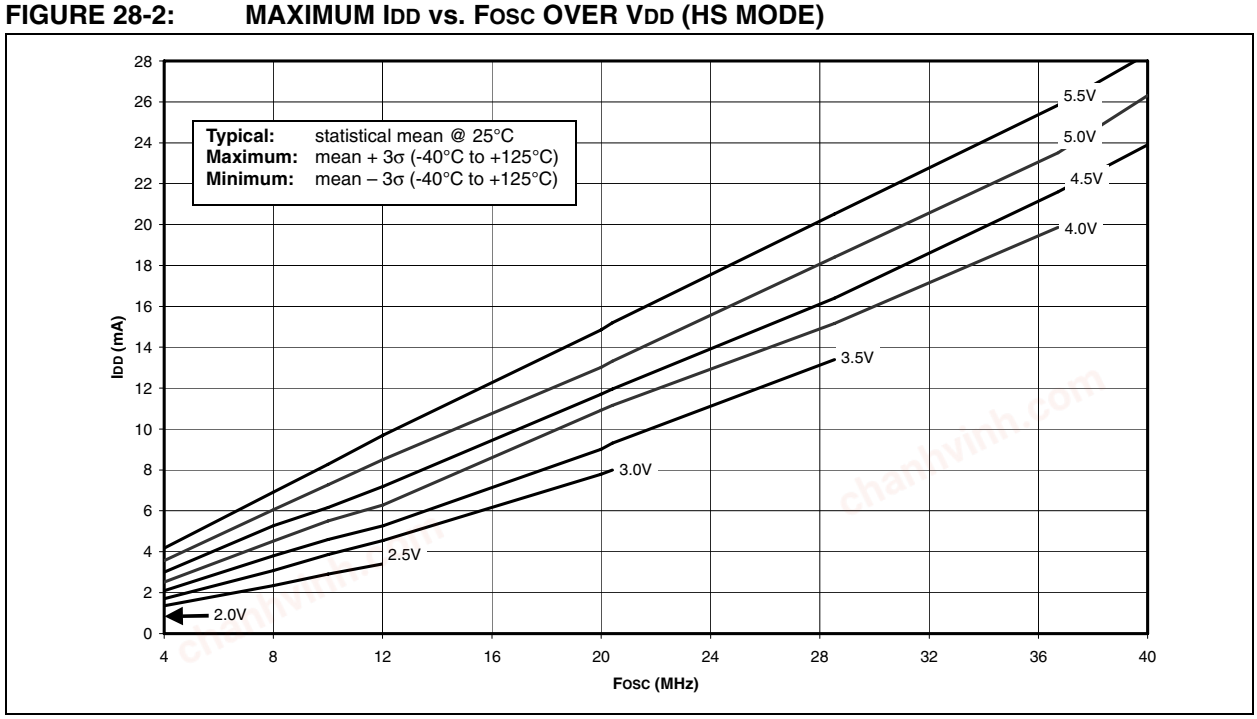



**FIGURE 28-2: MAXIMUM IDD vs. FOSC OVER VDD (HS MODE)**



chanhvinh.com



**FIGURE 28-3: TYPICAL IDD vs. FOSC OVER VDD (HS/PLL MODE)**

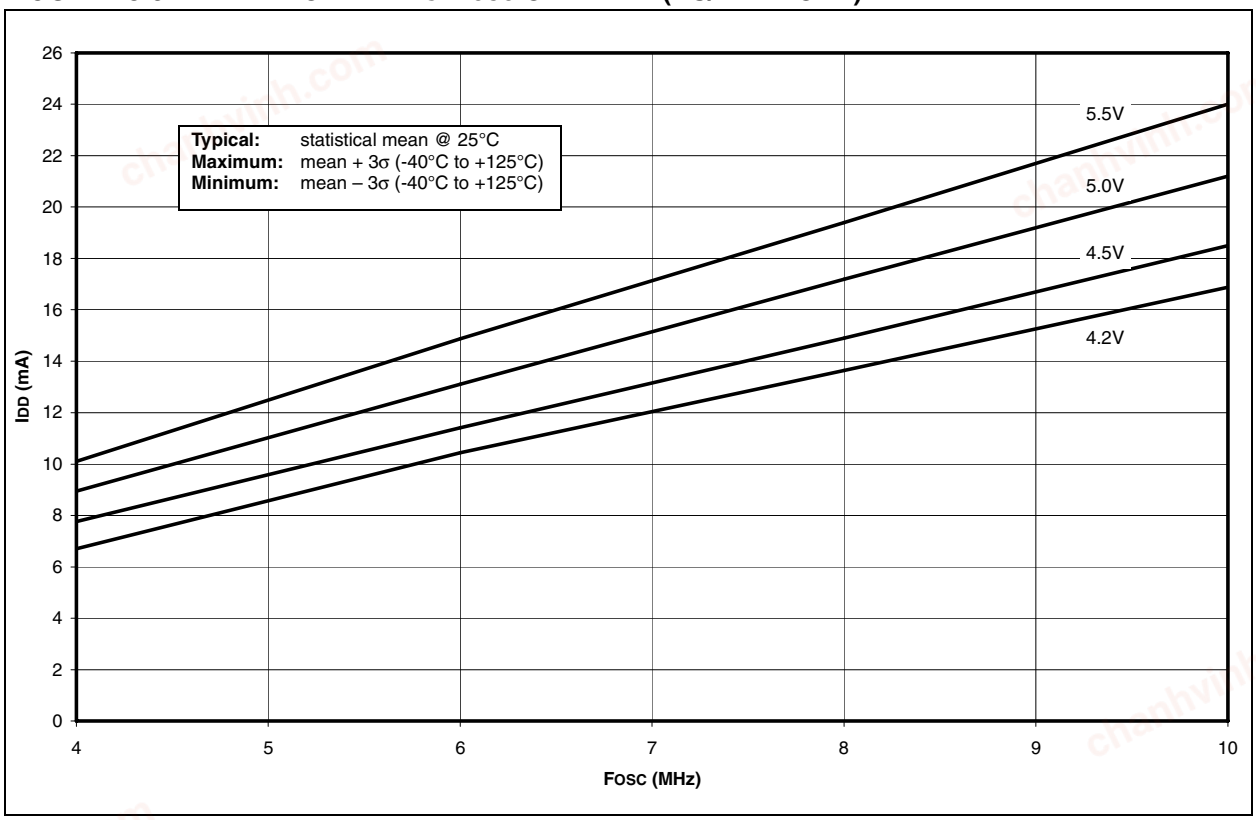
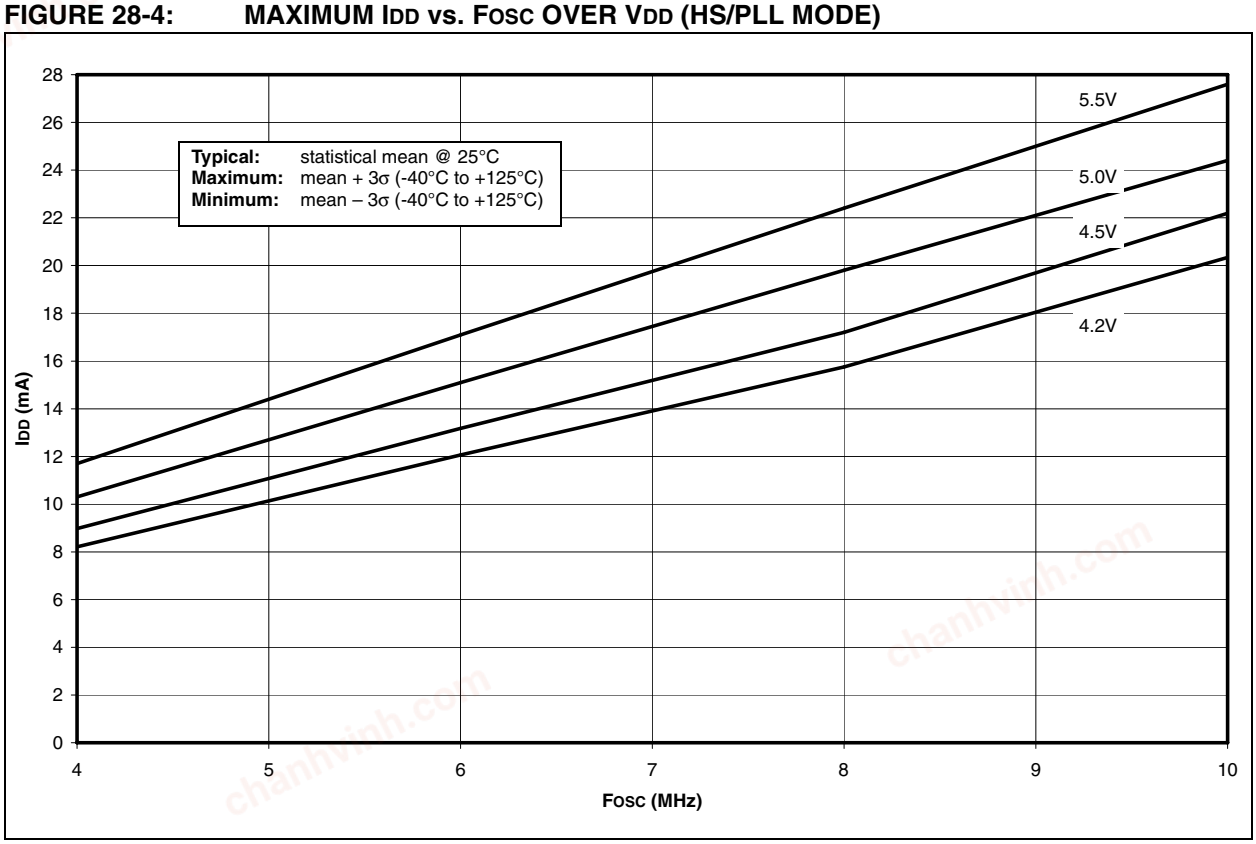



**FIGURE 28-4: MAXIMUM IDD vs. FOSC OVER VDD (HS/PLL MODE)**





© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 28-5: TYPICAL IDD vs. FOSC OVER VDD (XT MODE)**

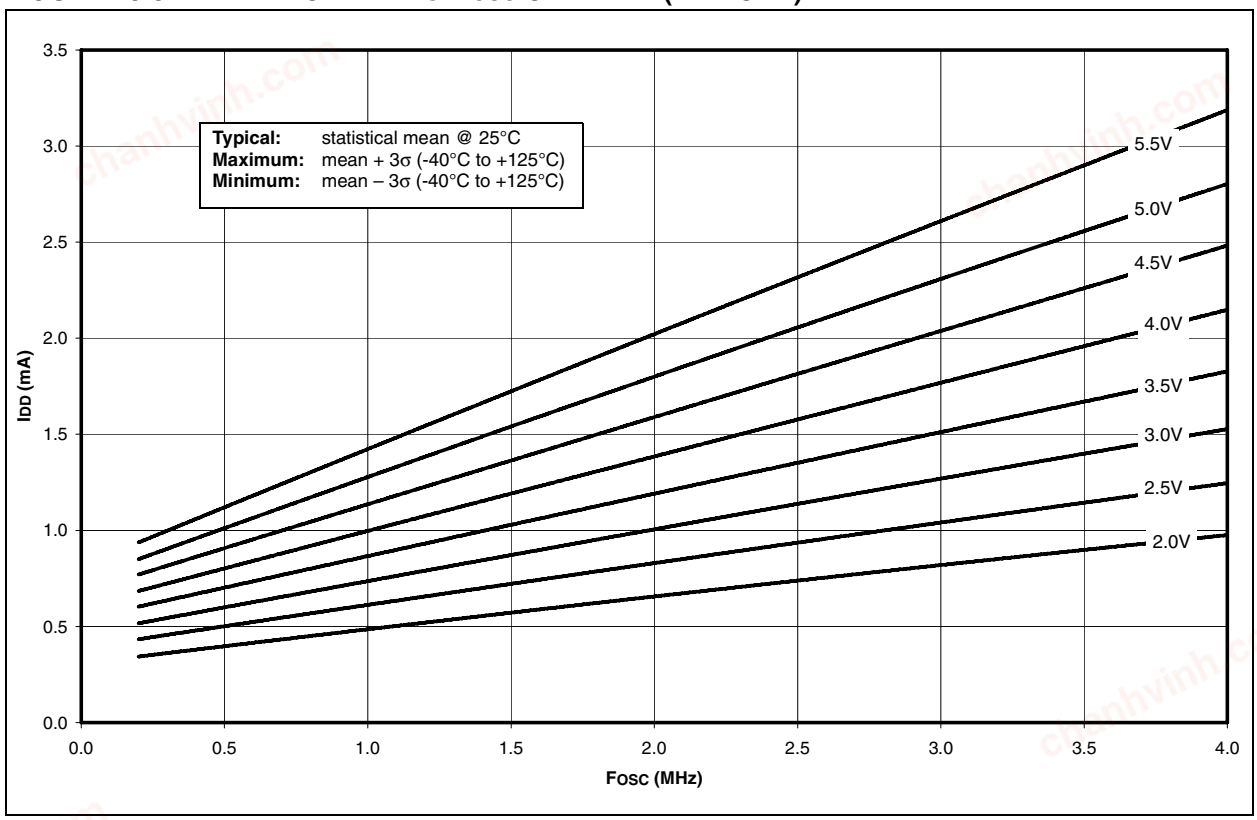
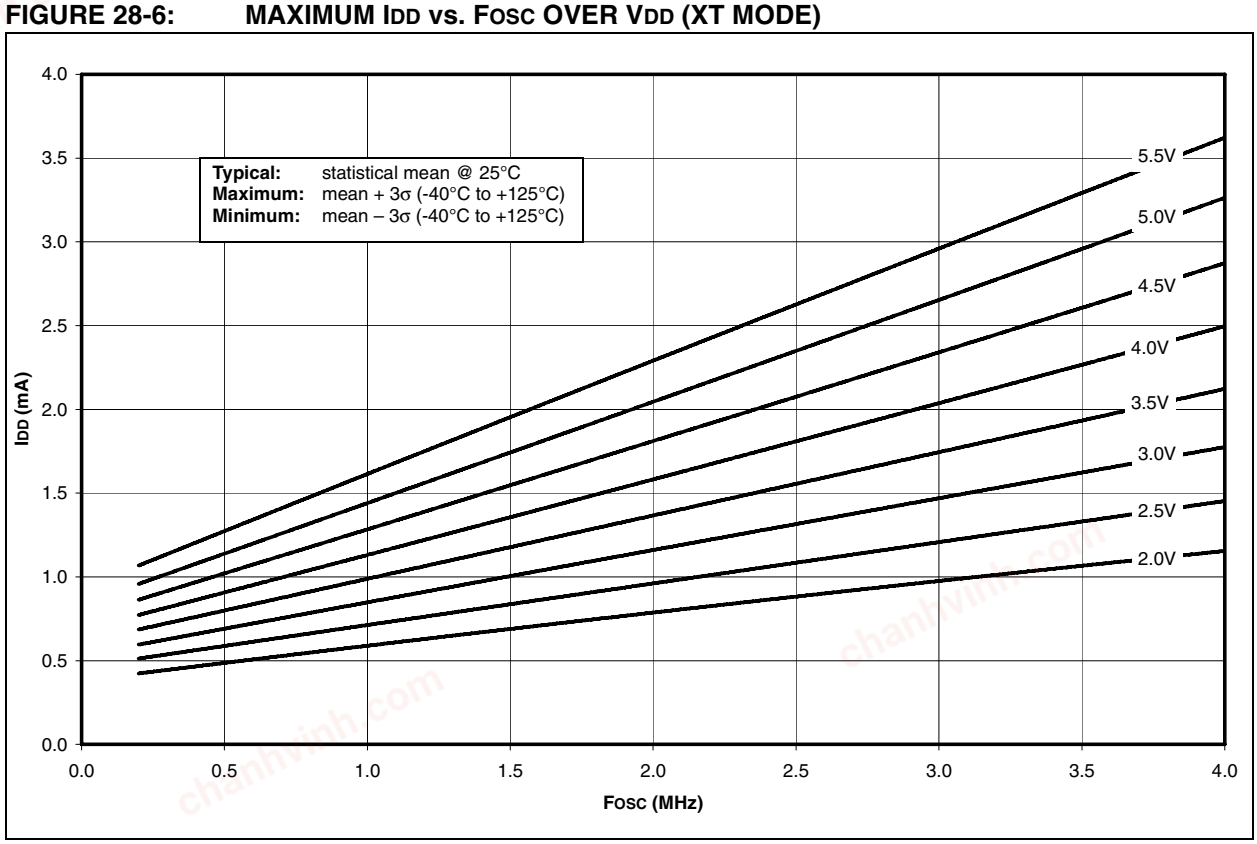



**FIGURE 28-6: MAXIMUM IDD vs. FOSC OVER VDD (XT MODE)**



chanhvinh.com



**FIGURE 28-7: TYPICAL IDD vs. FOSC OVER VDD (LP MODE)**

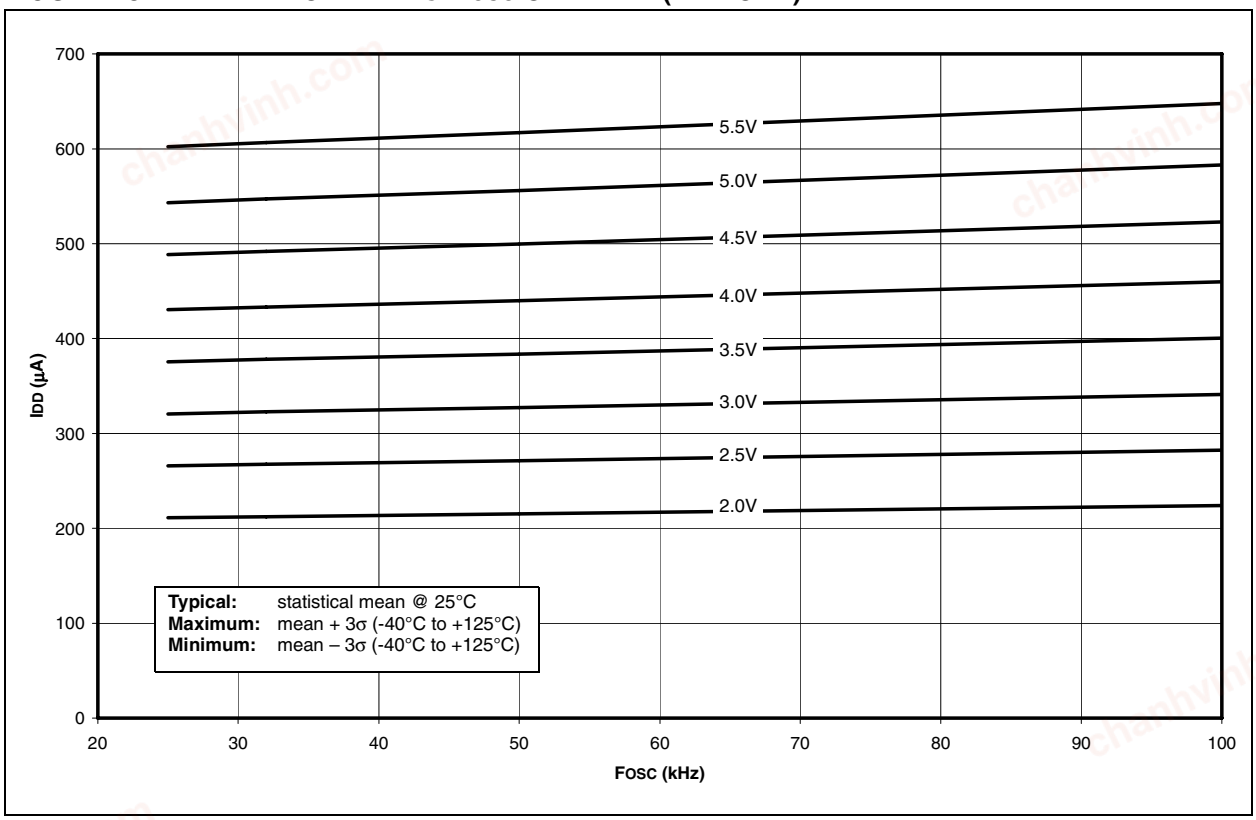
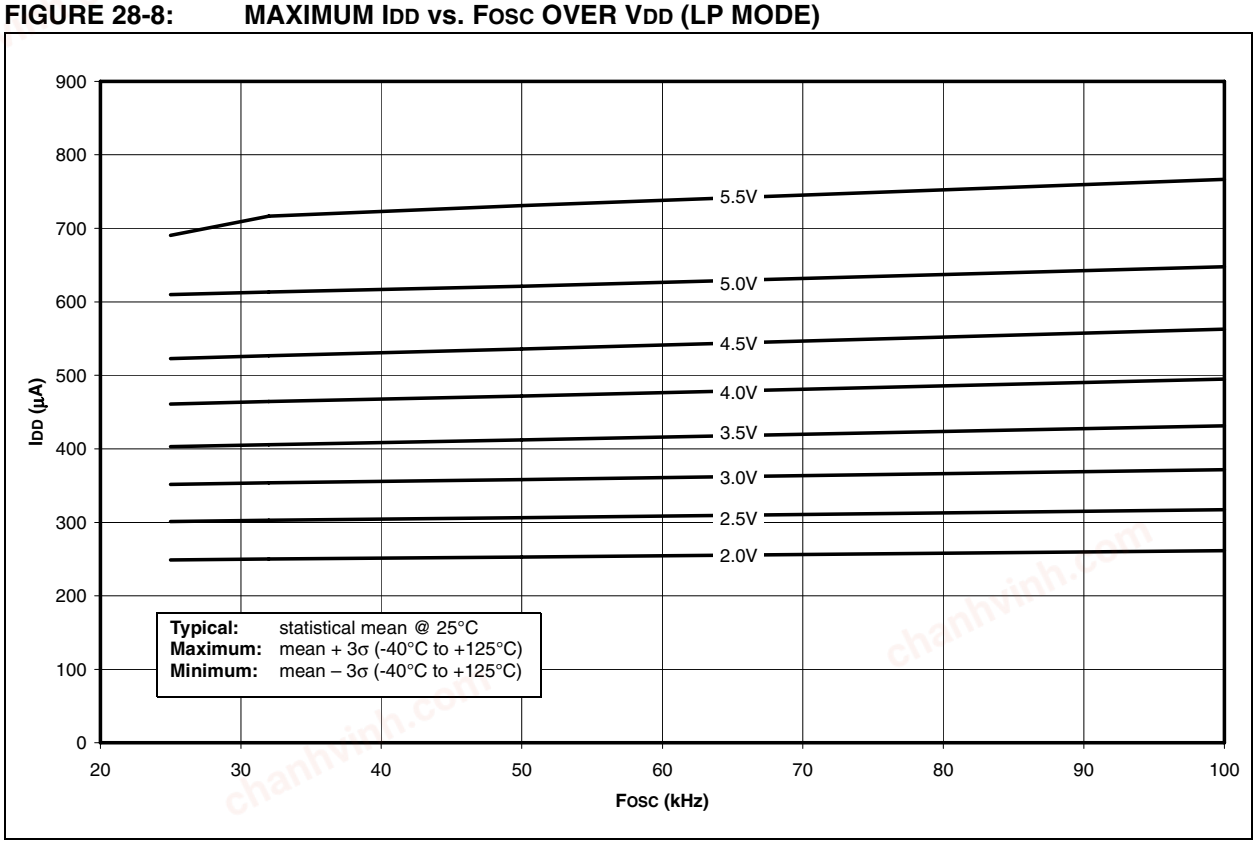



**FIGURE 28-8: MAXIMUM IDD vs. FOSC OVER VDD (LP MODE)**





© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 28-9: TYPICAL IDD vs. FOSC OVER VDD (EC MODE)**

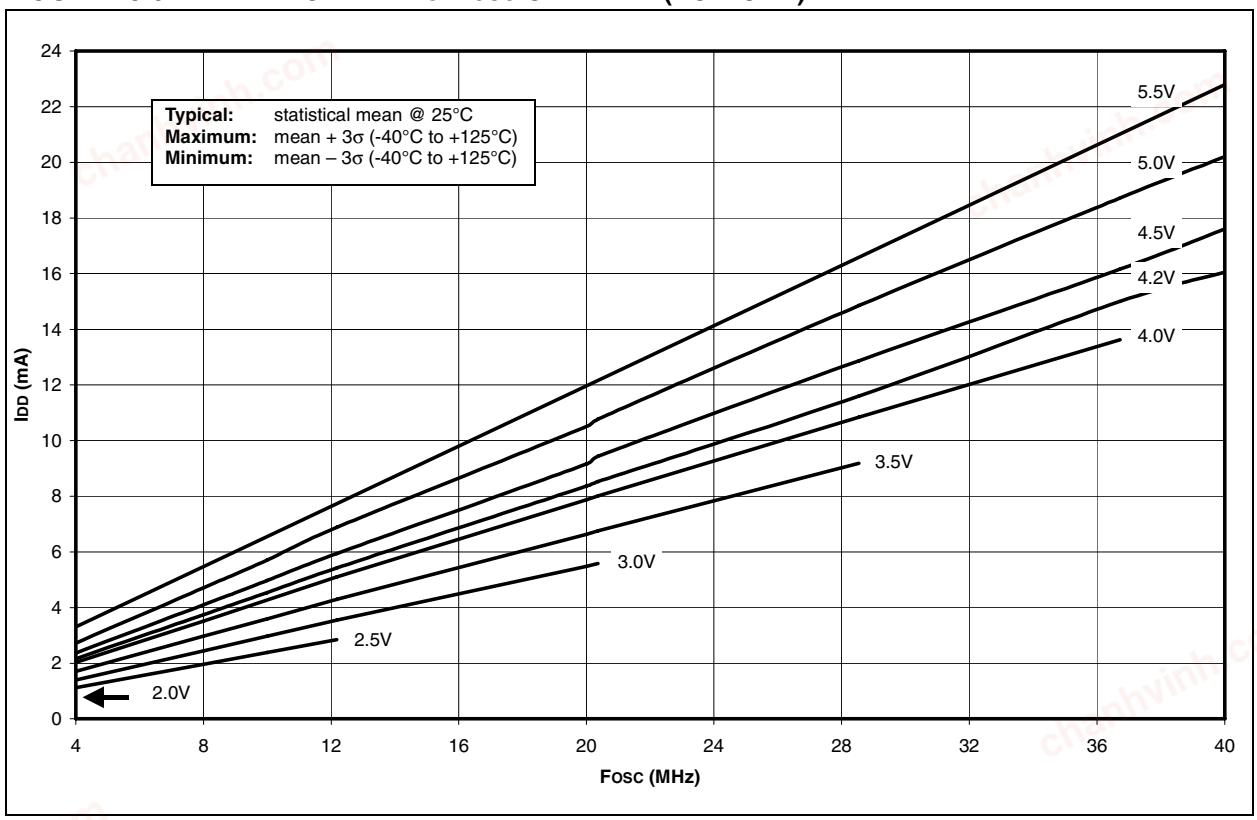
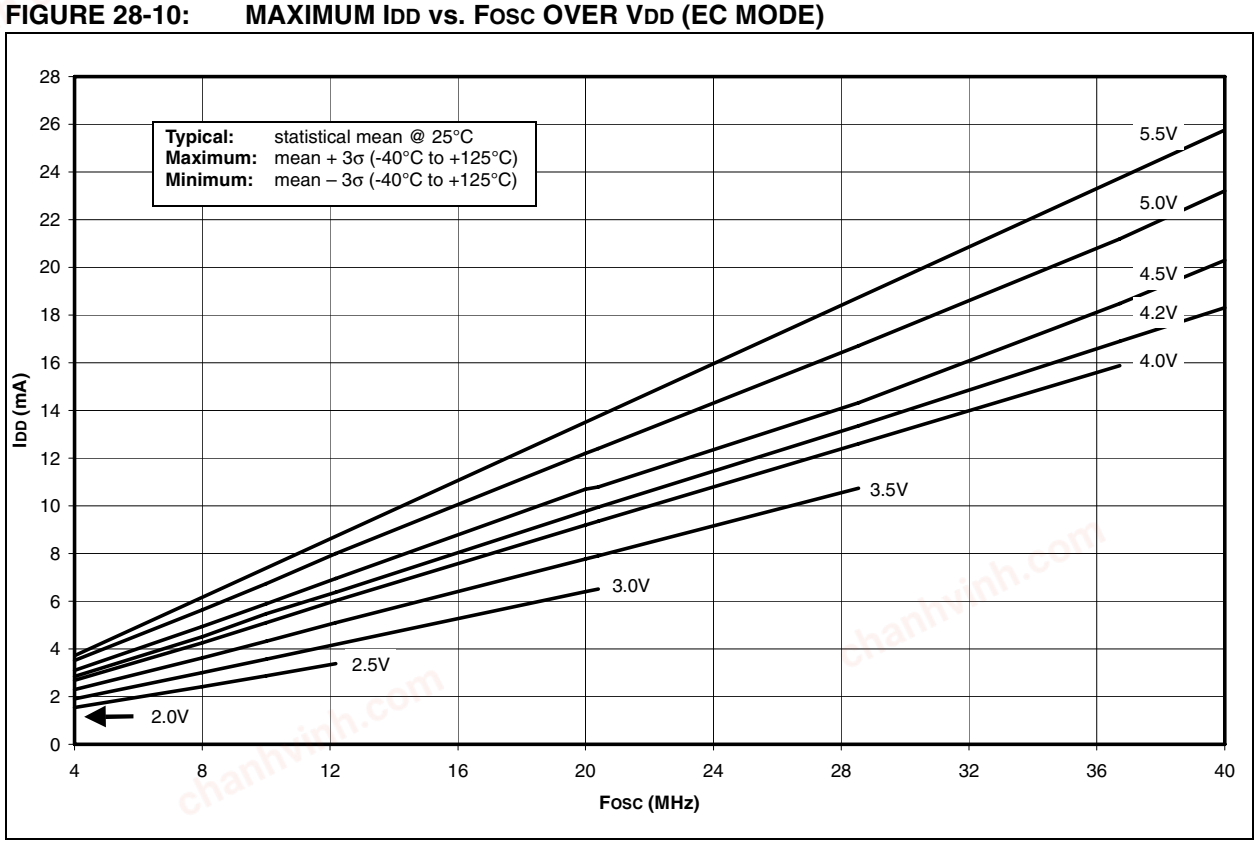



**FIGURE 28-10: MAXIMUM IDD vs. FOSC OVER VDD (EC MODE)**



chanhvinh.com



**FIGURE 28-11: TYPICAL AND MAXIMUM IDD vs. VDD (TIMER1 AS MAIN OSCILLATOR 32.768 kHz, C1 AND C2 = 47 pF)**

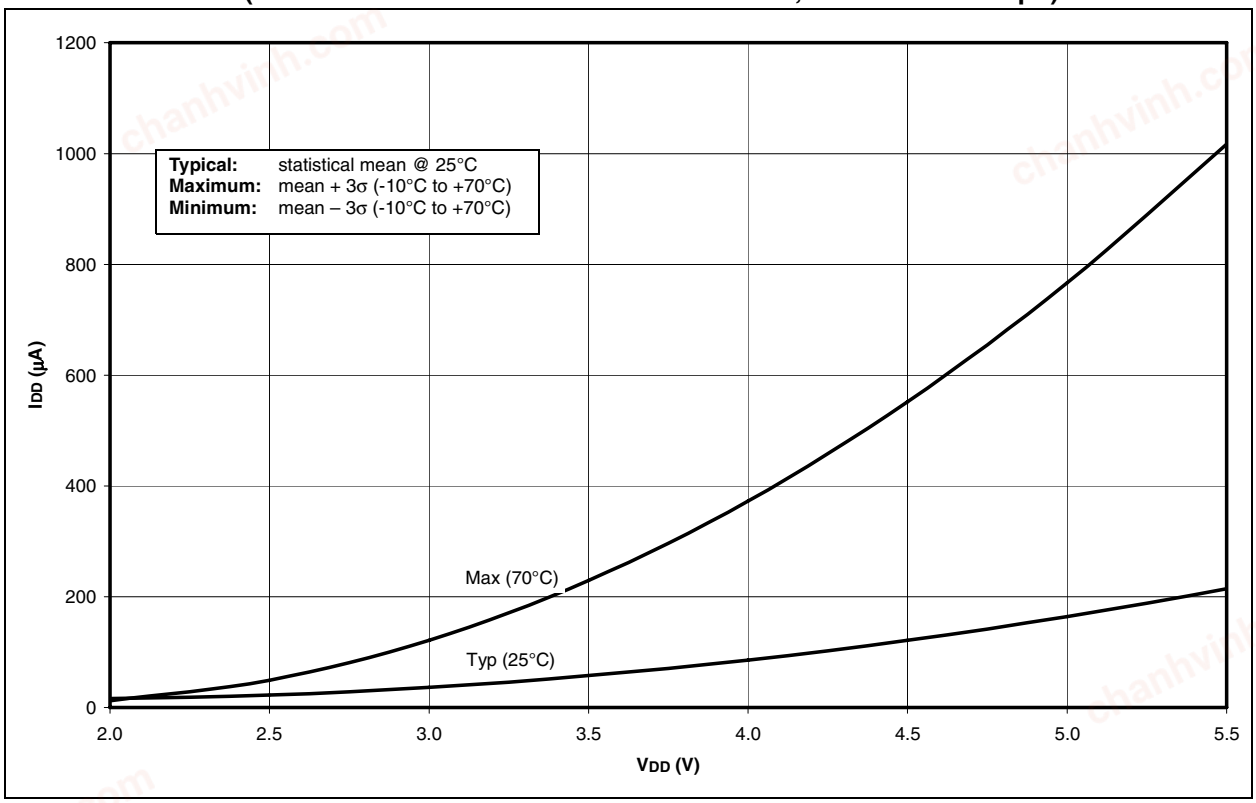
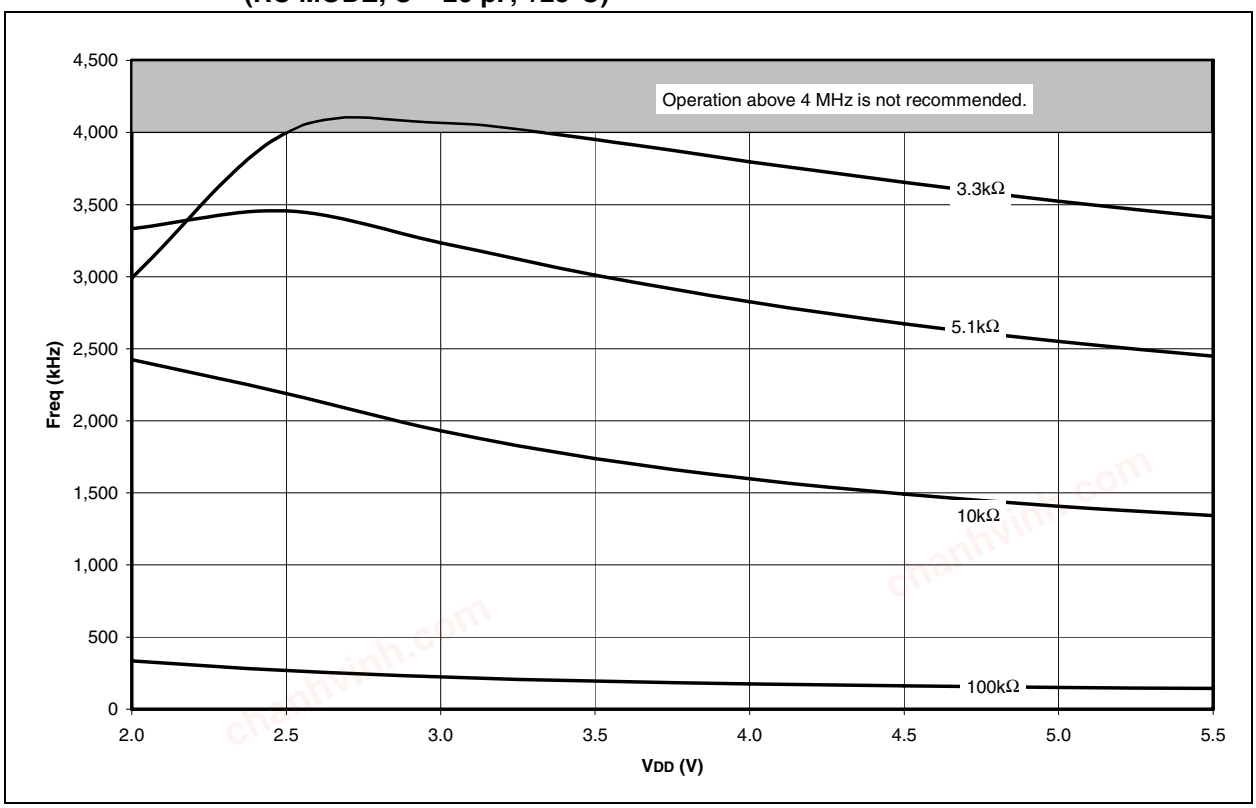



**FIGURE 28-12: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 20 pF, +25°C)**





© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 28-13: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 100 pF, +25°C)**

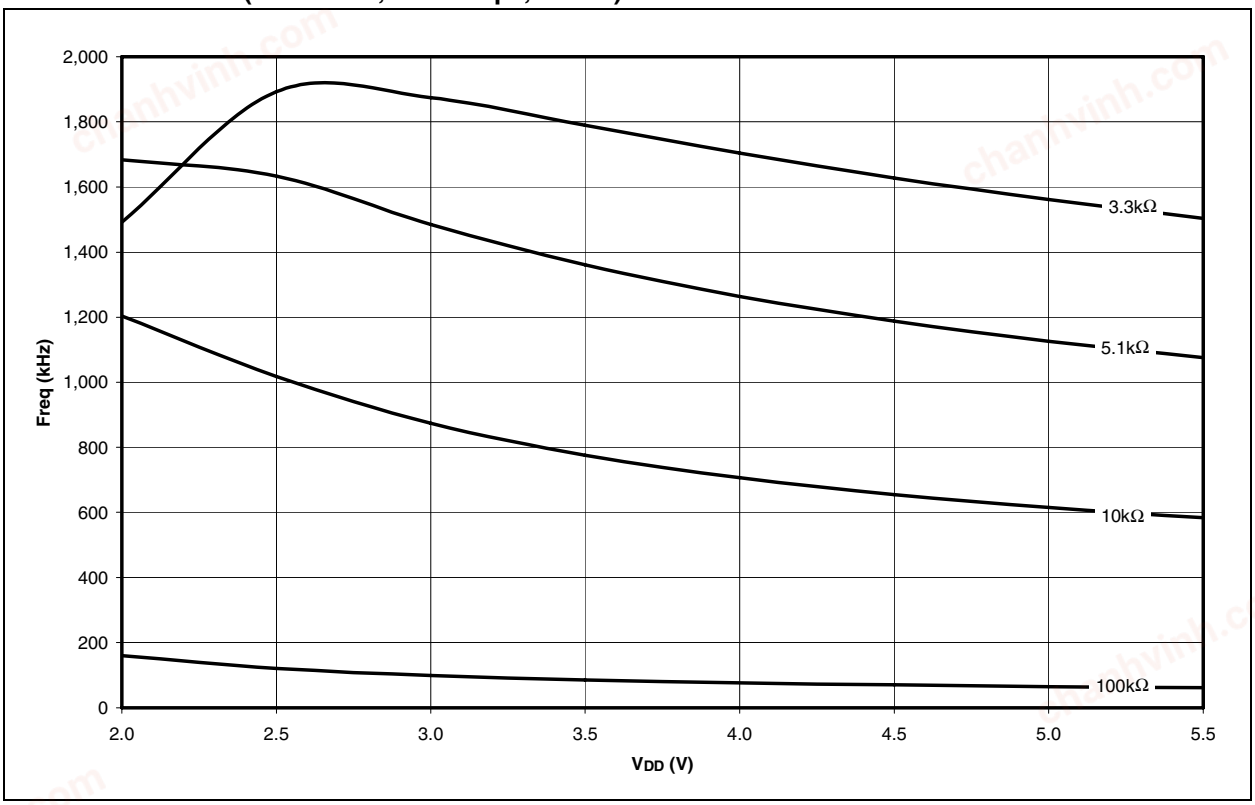
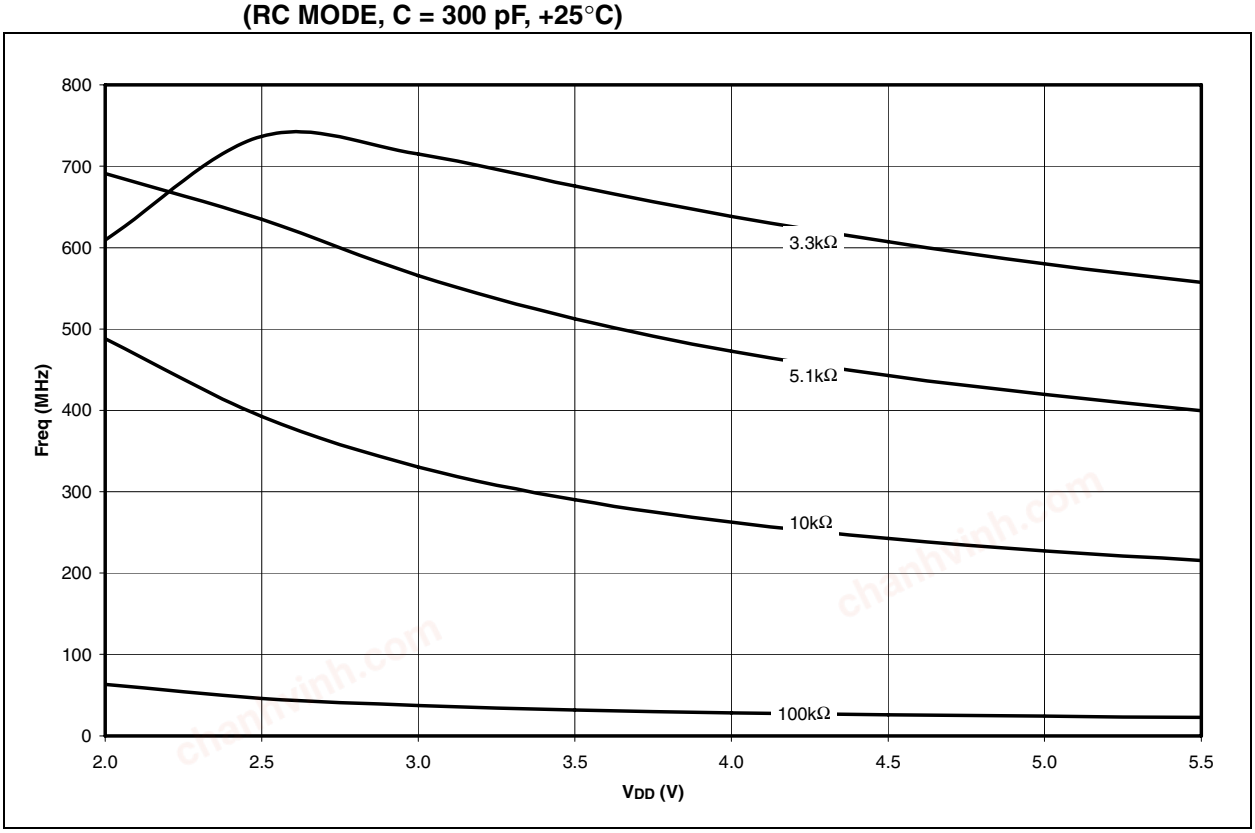



**FIGURE 28-14: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 300 pF, +25°C)**



chanhvinh.com



**FIGURE 28-15: $I_{PD}$ vs. $V_{DD}$, -40°C TO +125°C (SLEEP MODE, ALL PERIPHERALS DISABLED)**

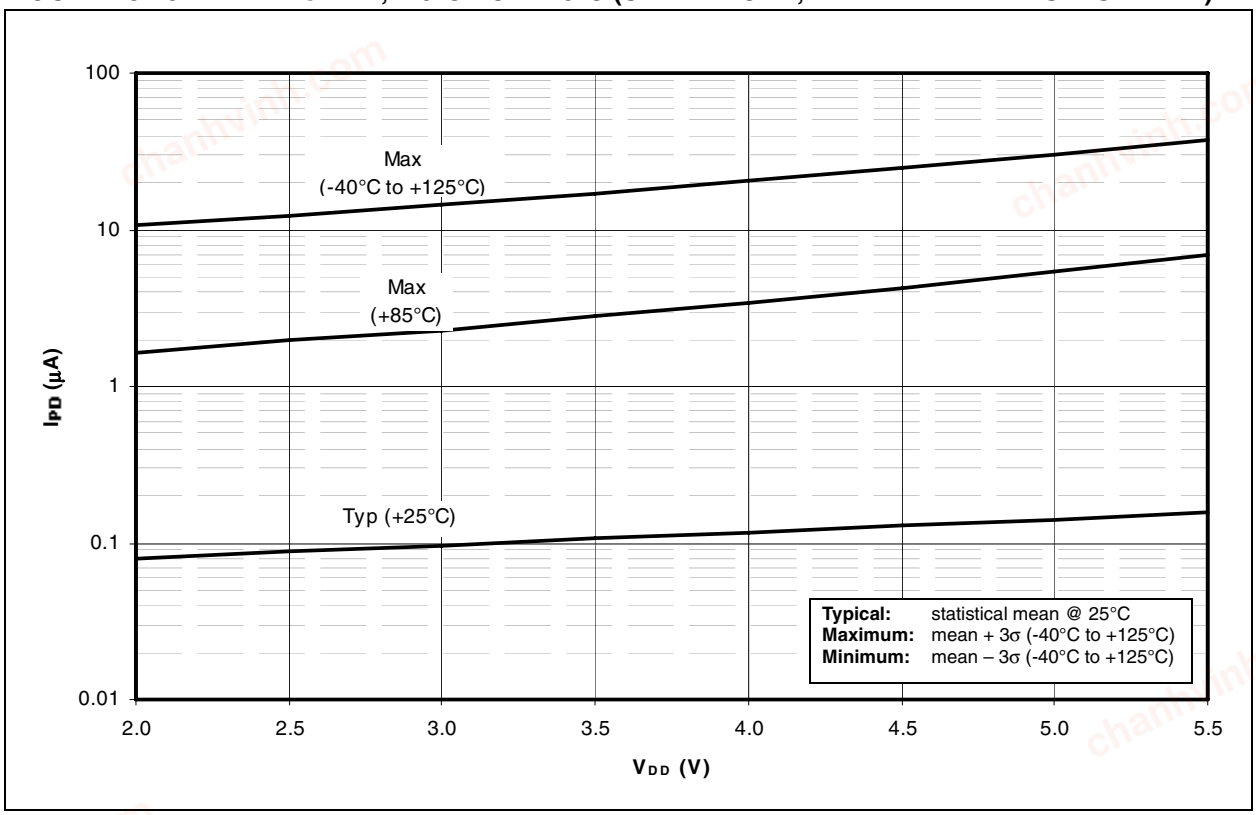
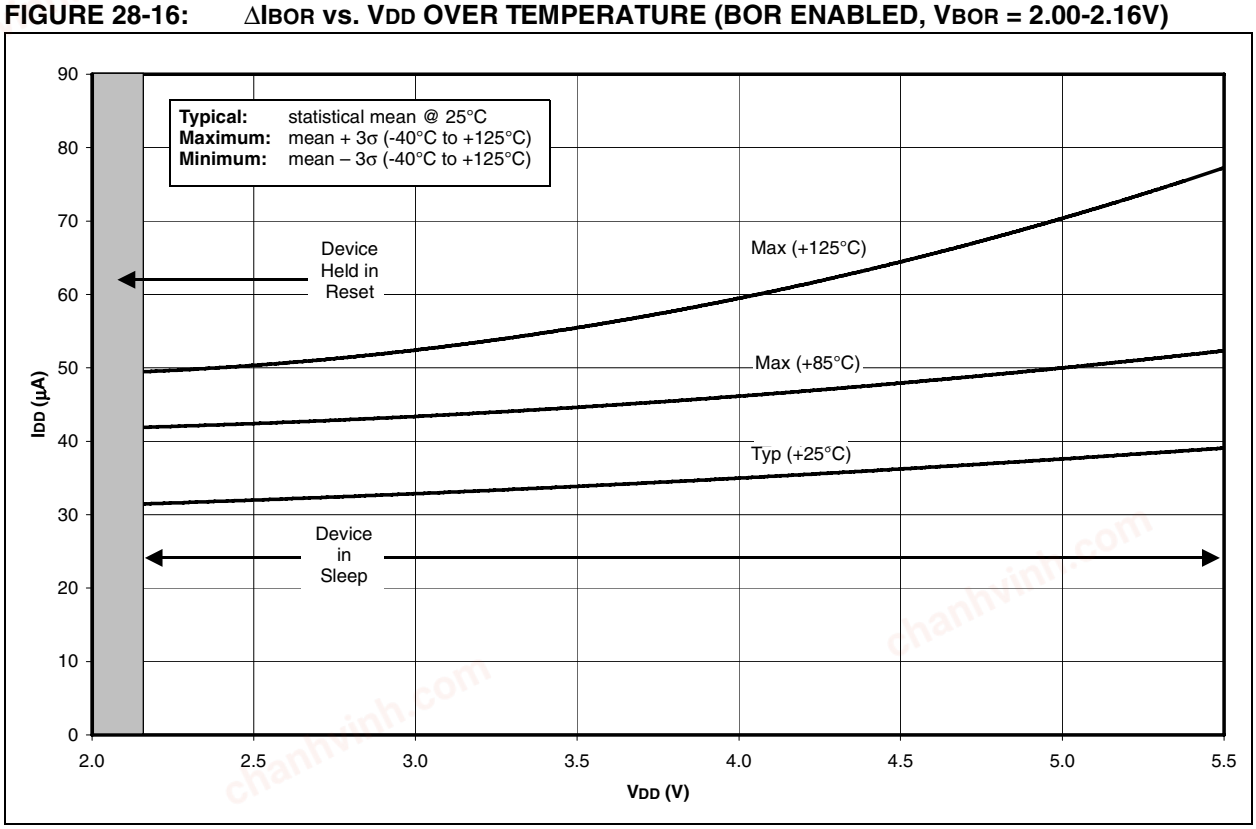



**FIGURE 28-16: $\Delta I_{BOR}$ vs. $V_{DD}$ OVER TEMPERATURE (BOR ENABLED, $V_{BOR}$ = 2.00-2.16V)**





© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 28-17: TYPICAL AND MAXIMUM ΔITMR1 vs. VDD OVER TEMPERATURE (-10°C TO +70°C, TIMER1 WITH OSCILLATOR, XTAL = 32 kHz, C1 AND C2 = 47 pF)**

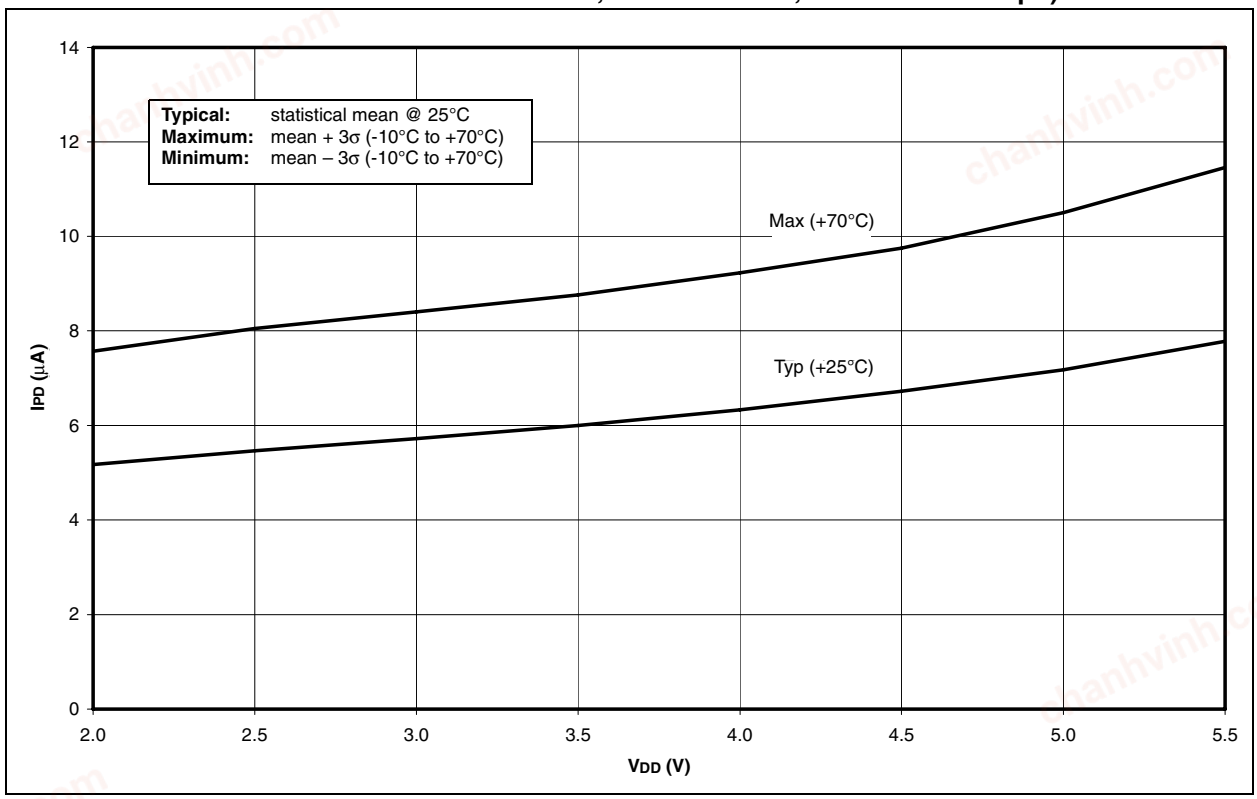
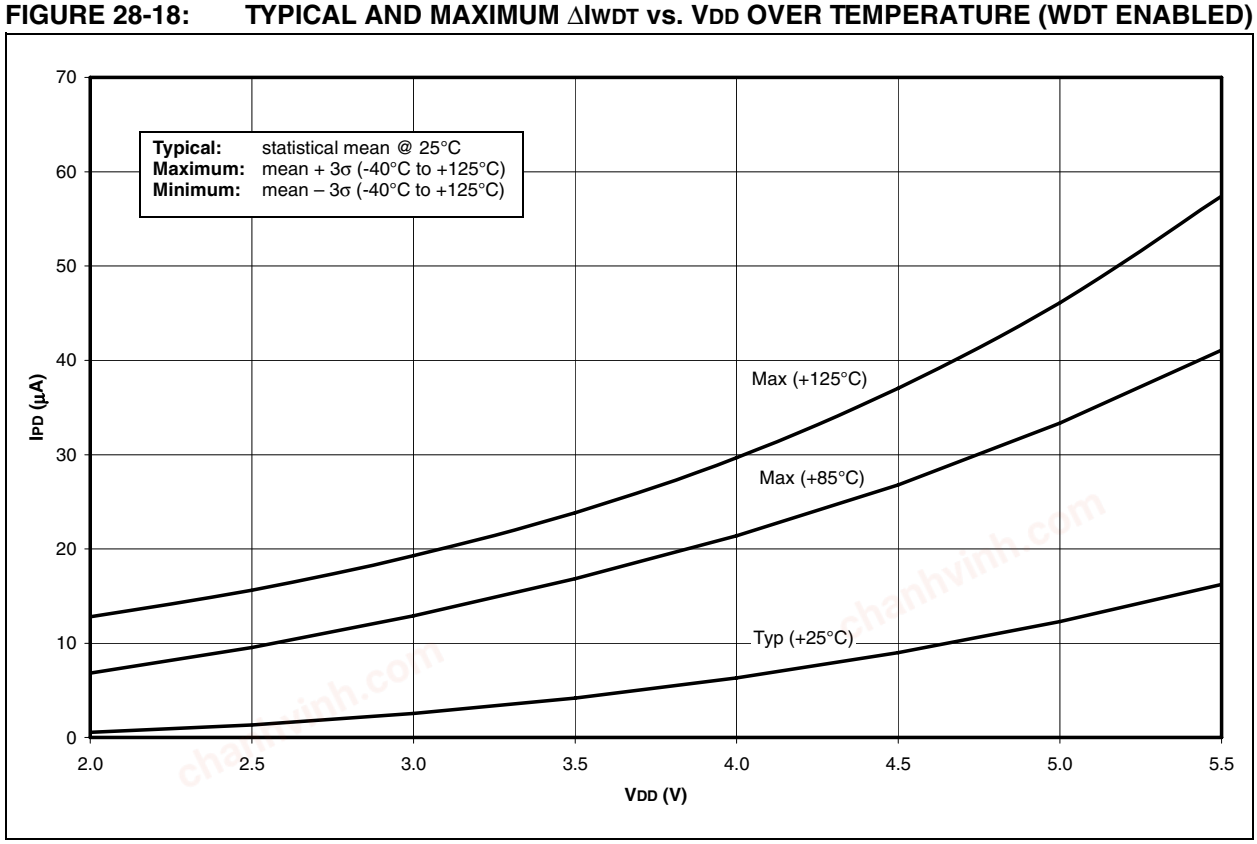



**FIGURE 28-18: TYPICAL AND MAXIMUM ΔIWDT vs. VDD OVER TEMPERATURE (WDT ENABLED)**



chanhvinh.com



**FIGURE 28-19: TYPICAL, MINIMUM AND MAXIMUM WDT PERIOD vs. VDD (-40°C TO +125°C)**

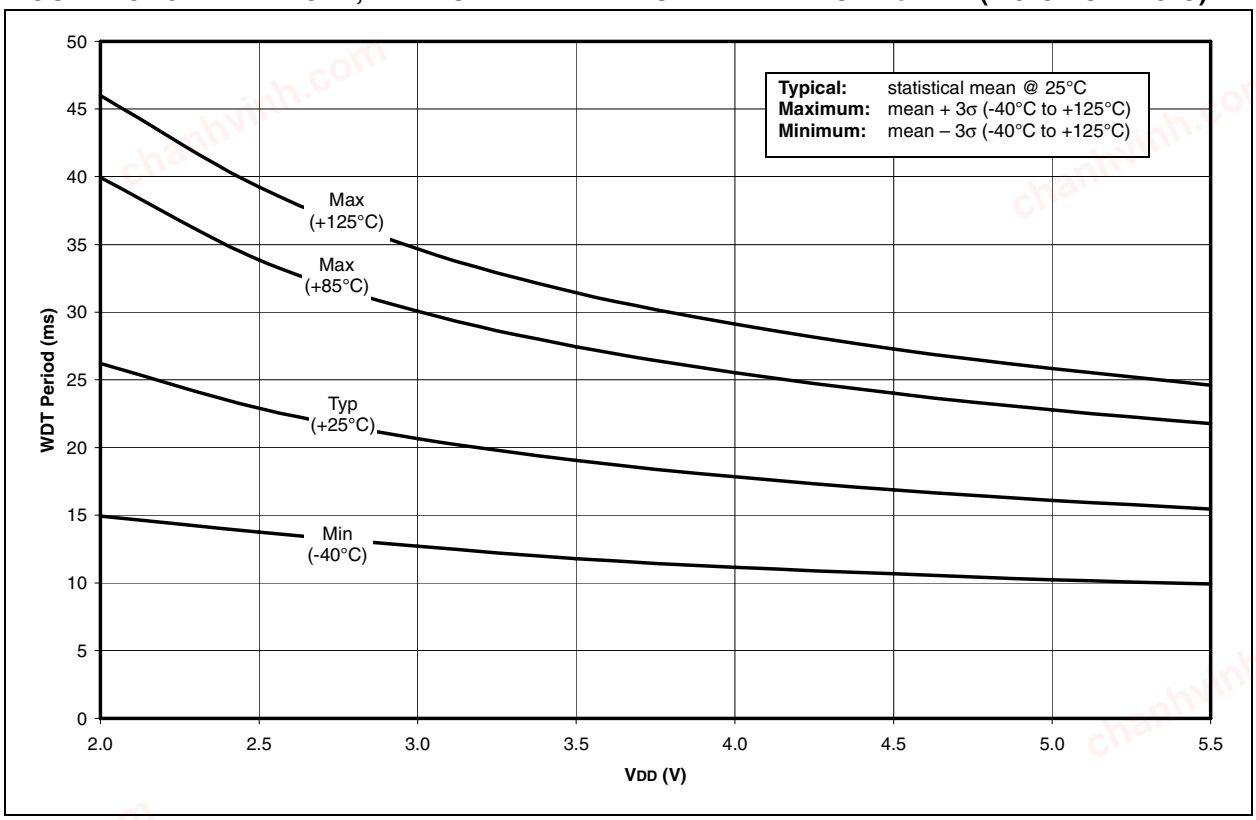
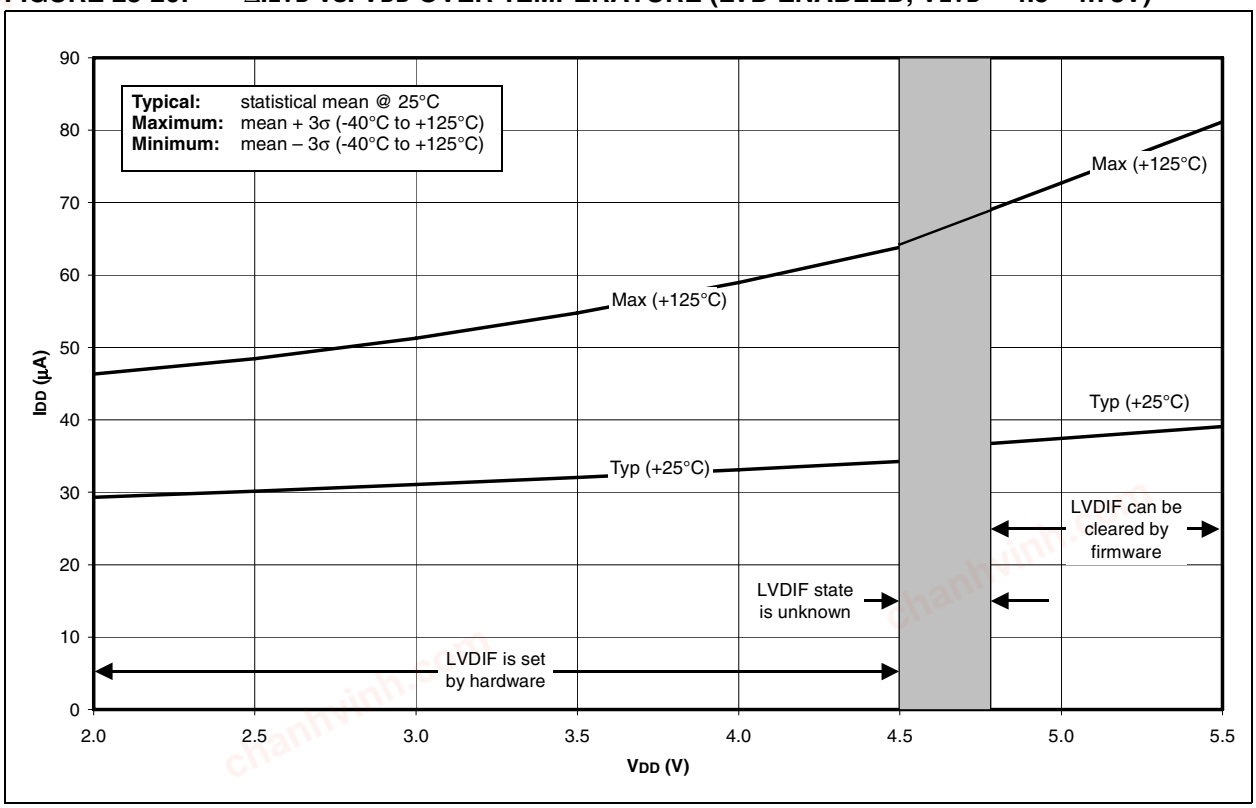



**FIGURE 28-20: ΔILVD vs. VDD OVER TEMPERATURE (LVD ENABLED, VLVD = 4.5 - 4.78V)**





© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 28-21: TYPICAL, MINIMUM AND MAXIMUM $V_{OH}$ vs. $I_{OH}$ ($V_{DD}$ = 5V, -40°C TO +125°C)**

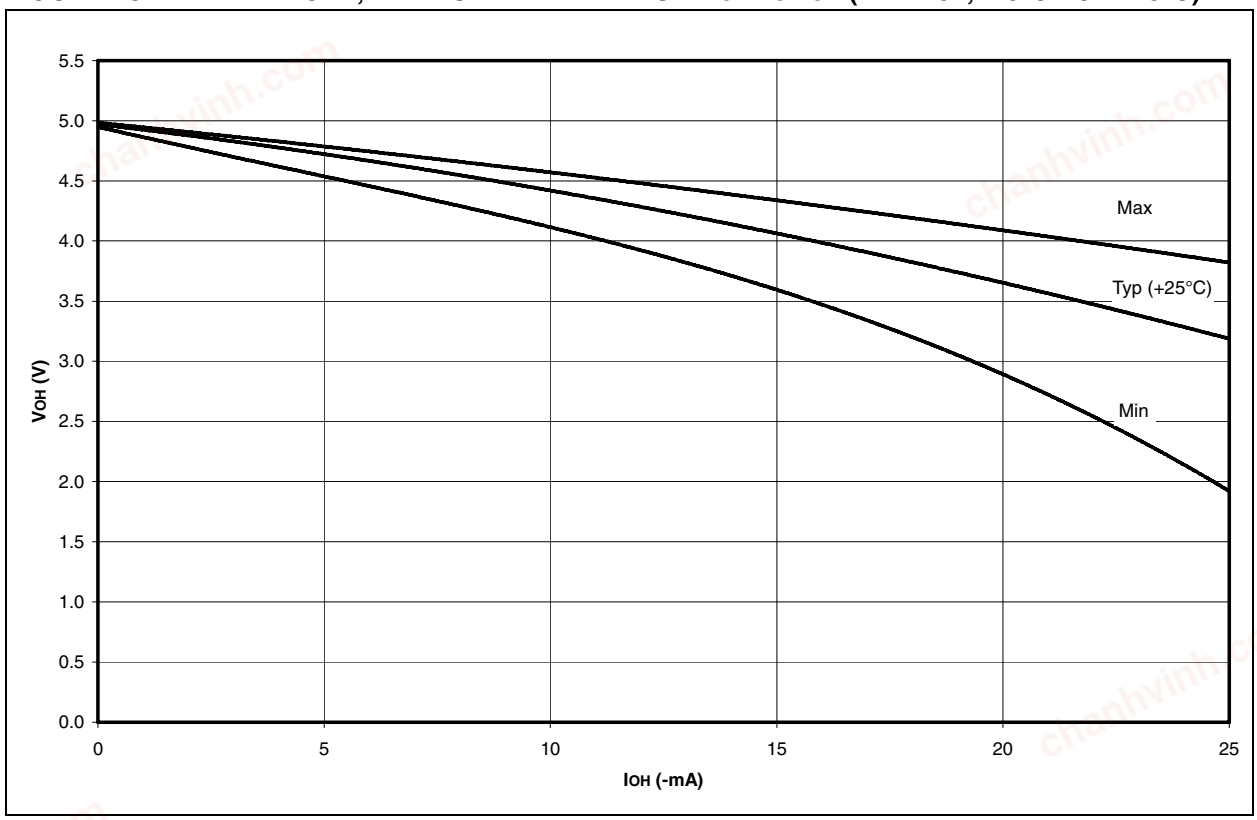
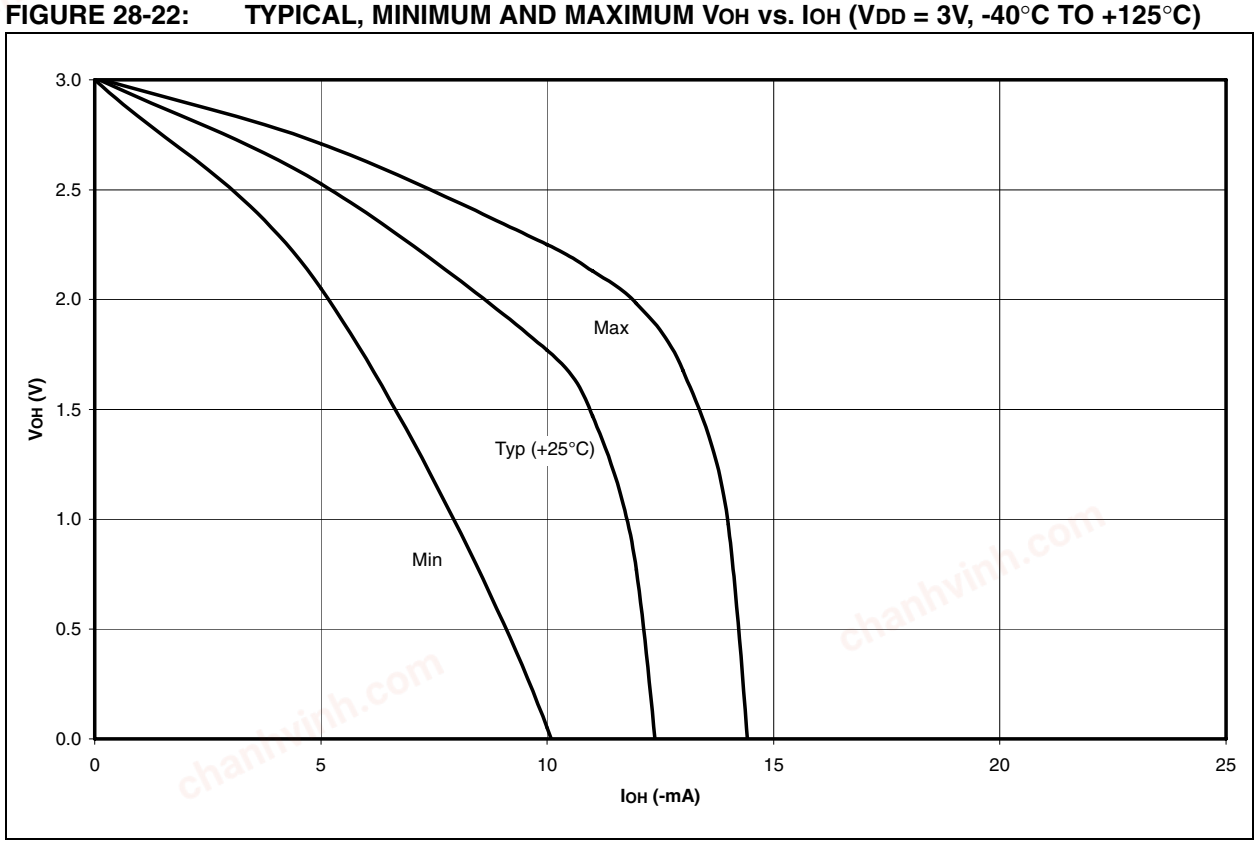



**FIGURE 28-22: TYPICAL, MINIMUM AND MAXIMUM $V_{OH}$ vs. $I_{OH}$ ($V_{DD}$ = 3V, -40°C TO +125°C)**



chanhvinh.com



**FIGURE 28-23: TYPICAL AND MAXIMUM VOL vs. IOL (VDD = 5V, -40°C TO +125°C)**

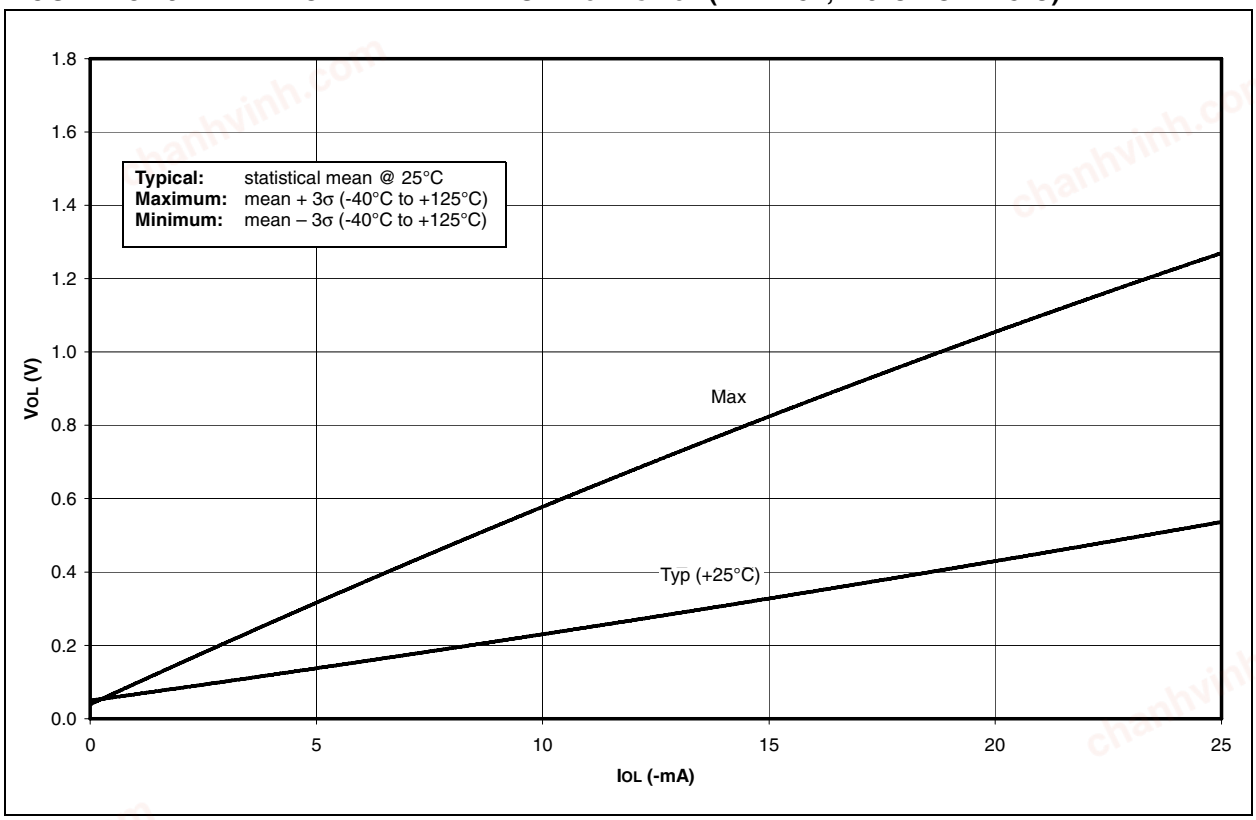
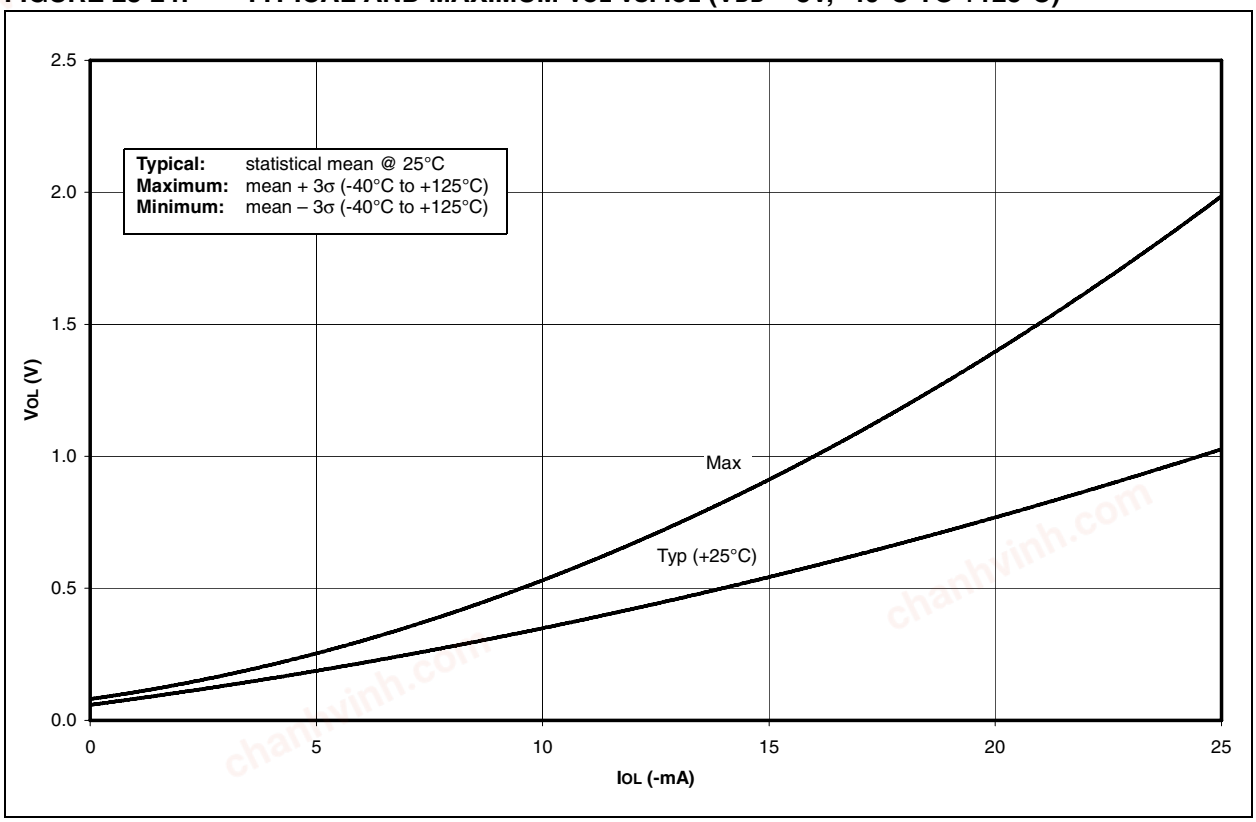



**FIGURE 28-24: TYPICAL AND MAXIMUM VOL vs. IOL (VDD = 3V, -40°C TO +125°C)**





© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 28-25: MINIMUM AND MAXIMUM VIN vs. VDD (ST INPUT, -40°C TO +125°C)**

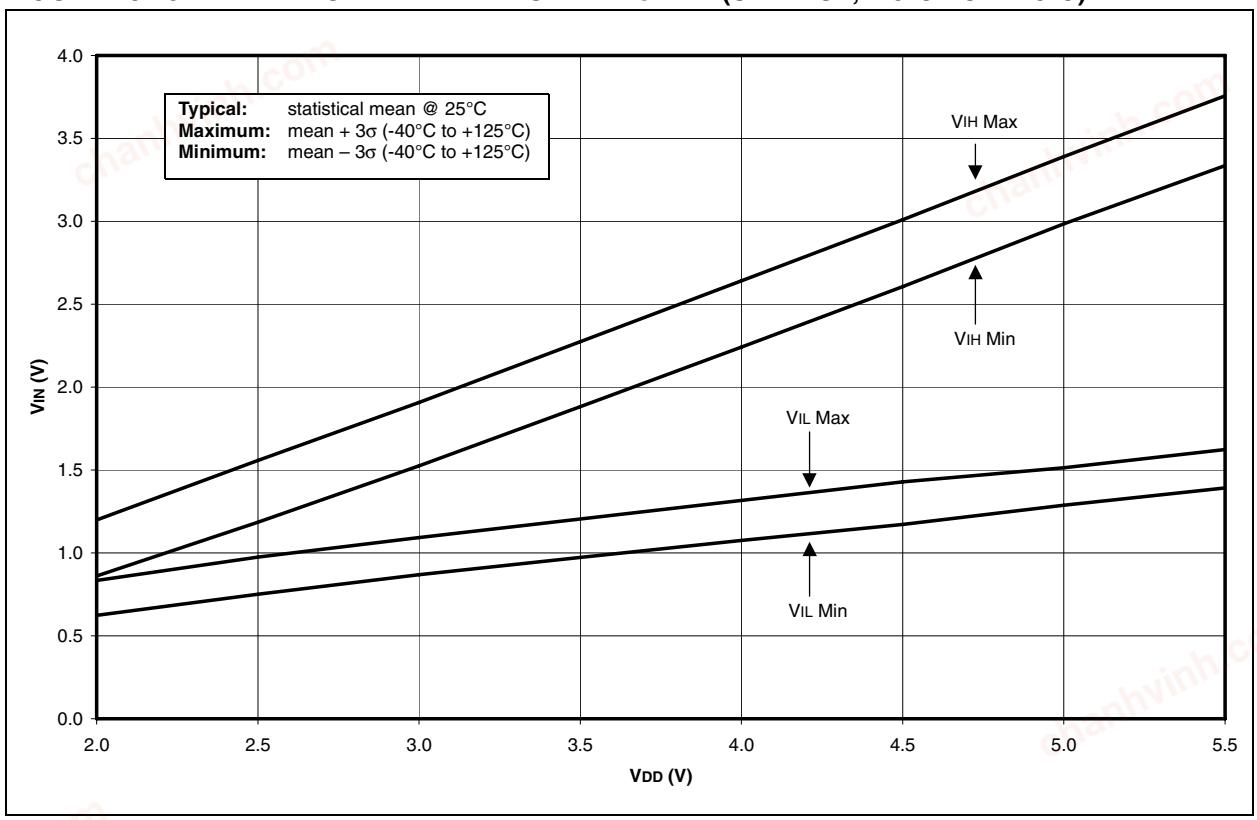
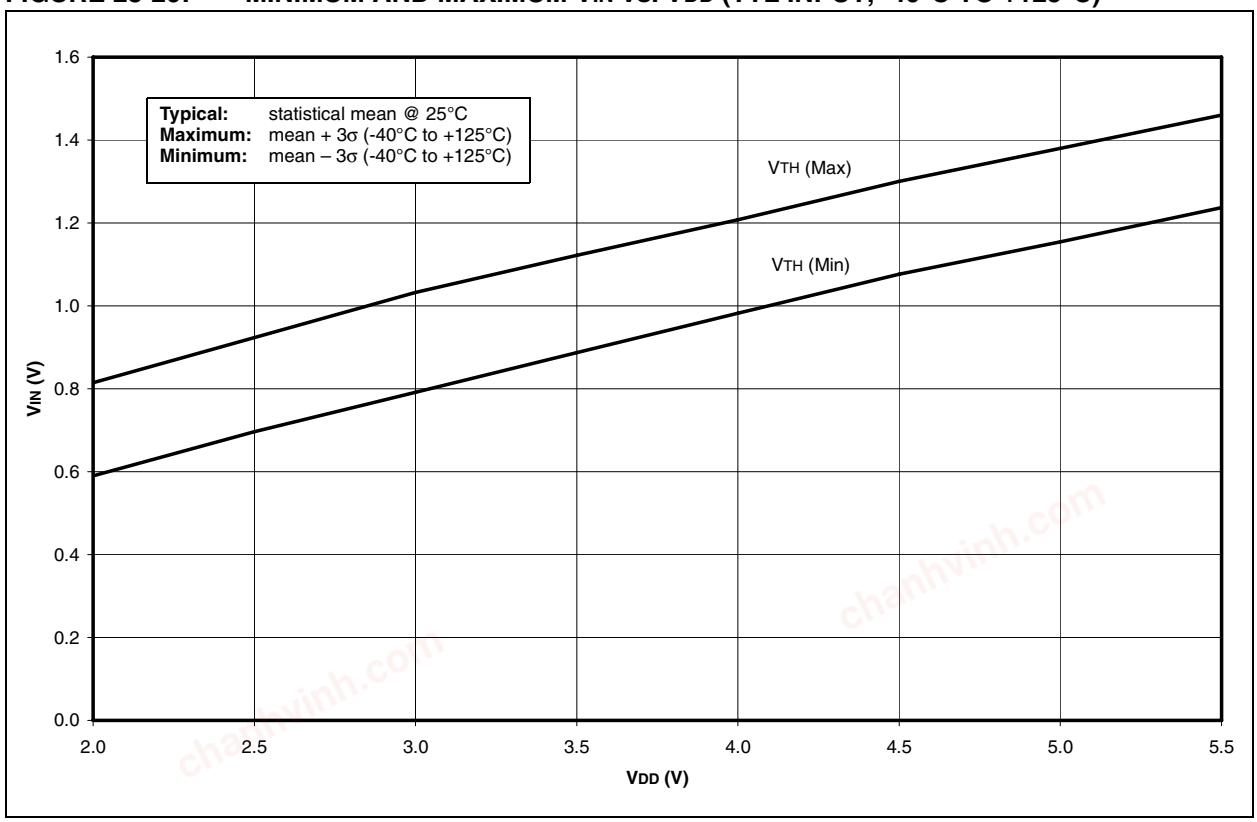



**FIGURE 28-26: MINIMUM AND MAXIMUM VIN vs. VDD (TTL INPUT, -40°C TO +125°C)**



chanhvinh.com



**FIGURE 28-27: MINIMUM AND MAXIMUM VIN vs. VDD (I2C™ INPUT, -40°C TO +125°C)**

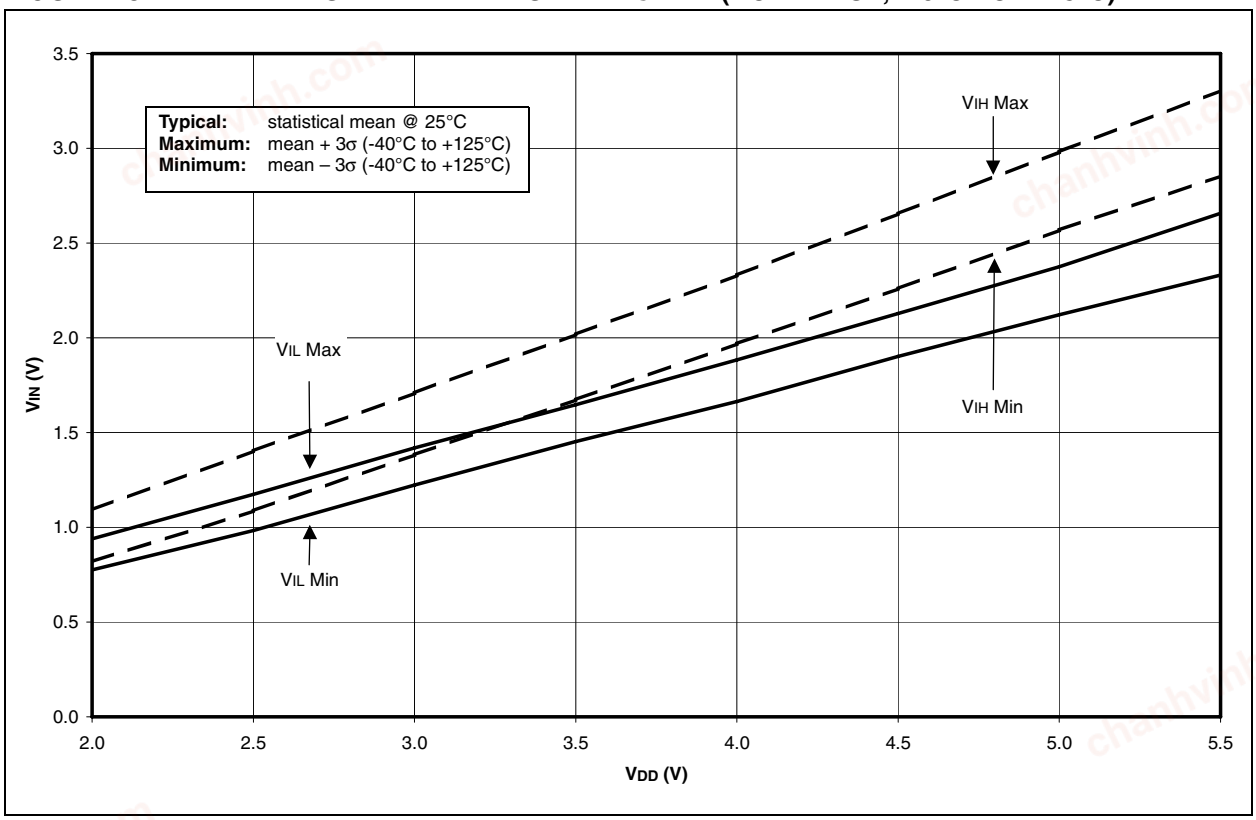
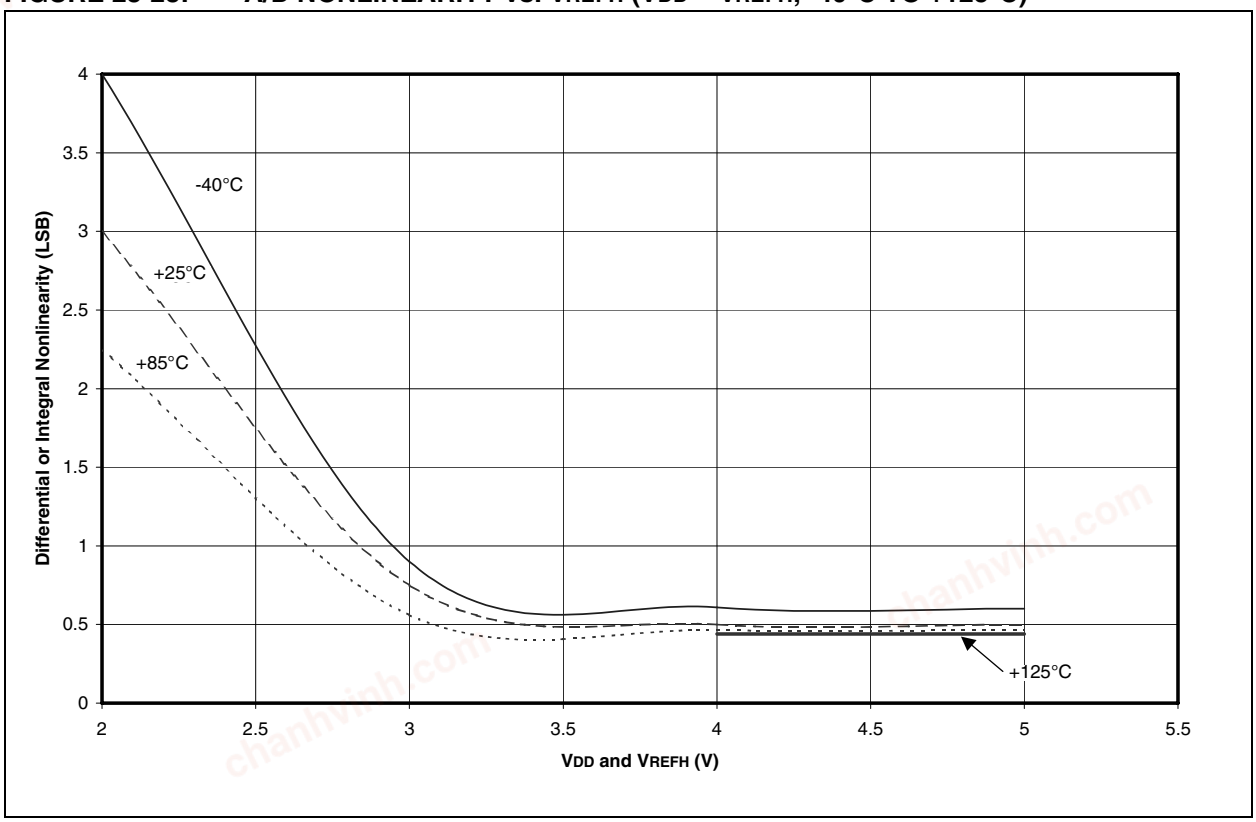



**FIGURE 28-28: A/D NONLINEARITY vs. VREFH (VDD = VREFH, -40°C TO +125°C)**



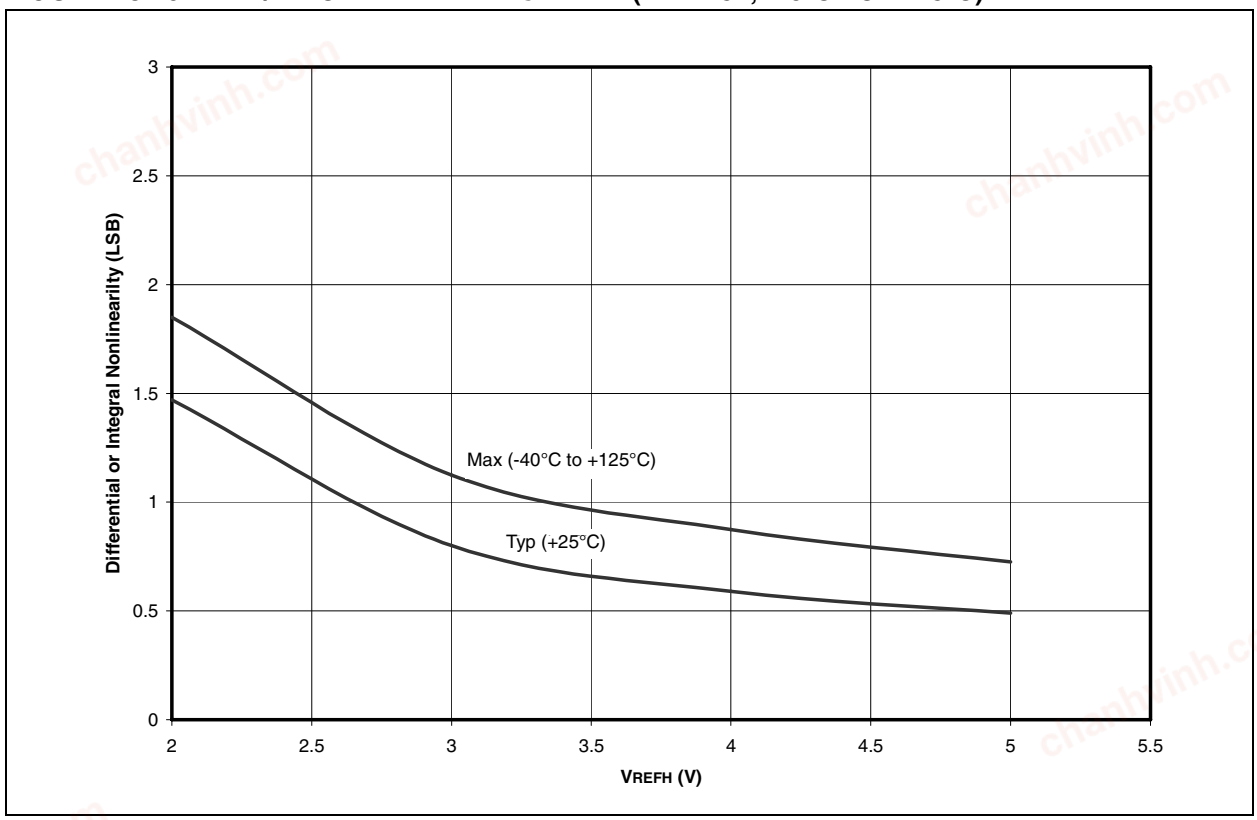


© 2006 Microchip Technology Inc.

chanhvinh.com



**FIGURE 28-29: A/D NONLINEARITY vs. VREFH (VDD = 5V, -40°C TO +125°C)**




chanhvinh.com

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

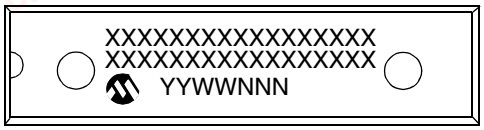
chanhvinh.com



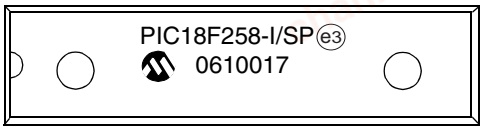
## 29.0 PACKAGING INFORMATION

### 29.1 Package Marking Information

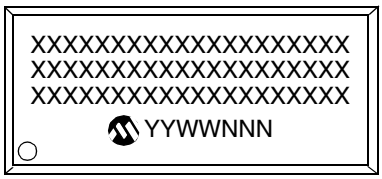
28-Lead SPDIP



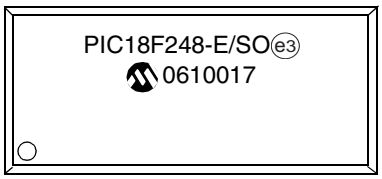

Example



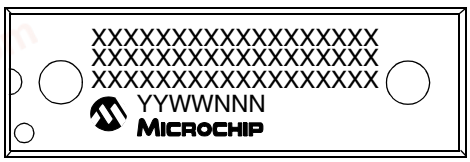

28-Lead SOIC



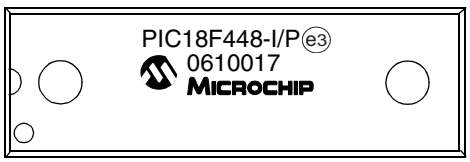

Example




40-Lead PDIP




Example




| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week '01') |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

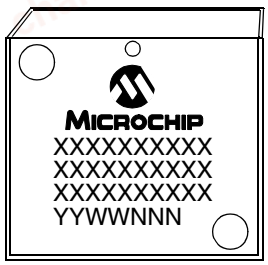
**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

chanhvinh.com

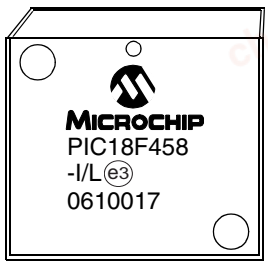


## 29.1 Package Marking Information (Continued)

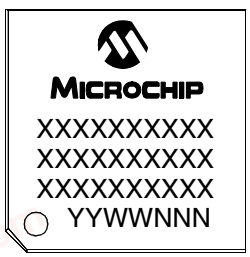
44-Lead PLCC



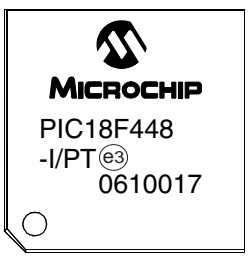

Example




44-Lead TQFP




Example





© 2006 Microchip Technology Inc.

chanhvinh.com

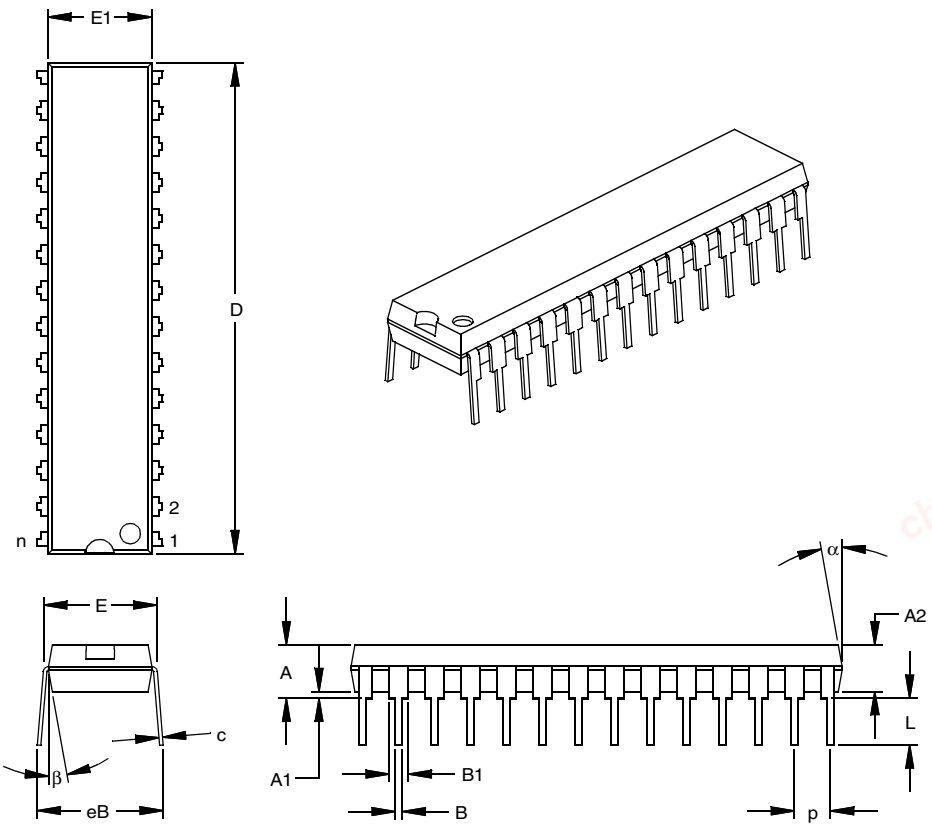


## 29.2 Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-line (SP) – 300 mil Body (PDIP)

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




| Units | | INCHES* | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 28 | | | 28 | |
| Pitch | p | | .100 | | | 2.54 | |
| Top to Seating Plane | A | .140 | .150 | .160 | 3.56 | 3.81 | 4.06 |
| Molded Package Thickness | A2 | .125 | .130 | .135 | 3.18 | 3.30 | 3.43 |
| Base to Seating Plane | A1 | .015 | | | 0.38 | | |
| Shoulder to Shoulder Width | E | .300 | .310 | .325 | 7.62 | 7.87 | 8.26 |
| Molded Package Width | E1 | .275 | .285 | .295 | 6.99 | 7.24 | 7.49 |
| Overall Length | D | 1.345 | 1.365 | 1.385 | 34.16 | 34.67 | 35.18 |
| Tip to Seating Plane | L | .125 | .130 | .135 | 3.18 | 3.30 | 3.43 |
| Lead Thickness | c | .008 | .012 | .015 | 0.20 | 0.29 | 0.38 |
| Upper Lead Width | B1 | .040 | .053 | .065 | 1.02 | 1.33 | 1.65 |
| Lower Lead Width | B | .016 | .019 | .022 | 0.41 | 0.48 | 0.56 |
| Overall Row Spacing § | eB | .320 | .350 | .430 | 8.13 | 8.89 | 10.92 |
| Mold Draft Angle Top | α | 5 | 10 | 15 | 5 | 10 | 15 |
| Mold Draft Angle Bottom | β | 5 | 10 | 15 | 5 | 10 | 15 |

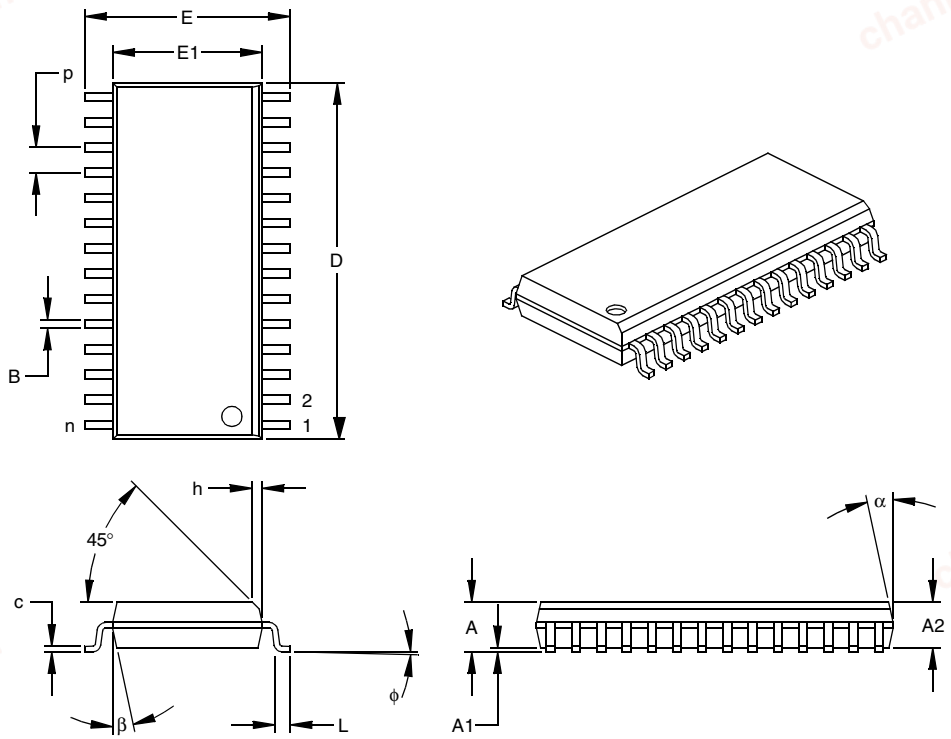
* Controlling Parameter
§ Significant Characteristic
**Notes:**
Dimension D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-095
Drawing No. C04-070

chanhvinh.com



## 28-Lead Plastic Small Outline (SO) – Wide, 300 mil Body (SOIC)

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




| Units | | INCHES* | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 28 | | | 28 | |
| Pitch | p | | .050 | | | 1.27 | |
| Overall Height | A | .093 | .099 | .104 | 2.36 | 2.50 | 2.64 |
| Molded Package Thickness | A2 | .088 | .091 | .094 | 2.24 | 2.31 | 2.39 |
| Standoff § | A1 | .004 | .008 | .012 | 0.10 | 0.20 | 0.30 |
| Overall Width | E | .394 | .407 | .420 | 10.01 | 10.34 | 10.67 |
| Molded Package Width | E1 | .288 | .295 | .299 | 7.32 | 7.49 | 7.59 |
| Overall Length | D | .695 | .704 | .712 | 17.65 | 17.87 | 18.08 |
| Chamfer Distance | h | .010 | .020 | .029 | 0.25 | 0.50 | 0.74 |
| Foot Length | L | .016 | .033 | .050 | 0.41 | 0.84 | 1.27 |
| Foot Angle Top | ϕ | 0 | 4 | 8 | 0 | 4 | 8 |
| Lead Thickness | c | .009 | .011 | .013 | 0.23 | 0.28 | 0.33 |
| Lead Width | B | .014 | .017 | .020 | 0.36 | 0.42 | 0.51 |
| Mold Draft Angle Top | α | 0 | 12 | 15 | 0 | 12 | 15 |
| Mold Draft Angle Bottom | β | 0 | 12 | 15 | 0 | 12 | 15 |

* 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: MS-013
Drawing No. C04-052

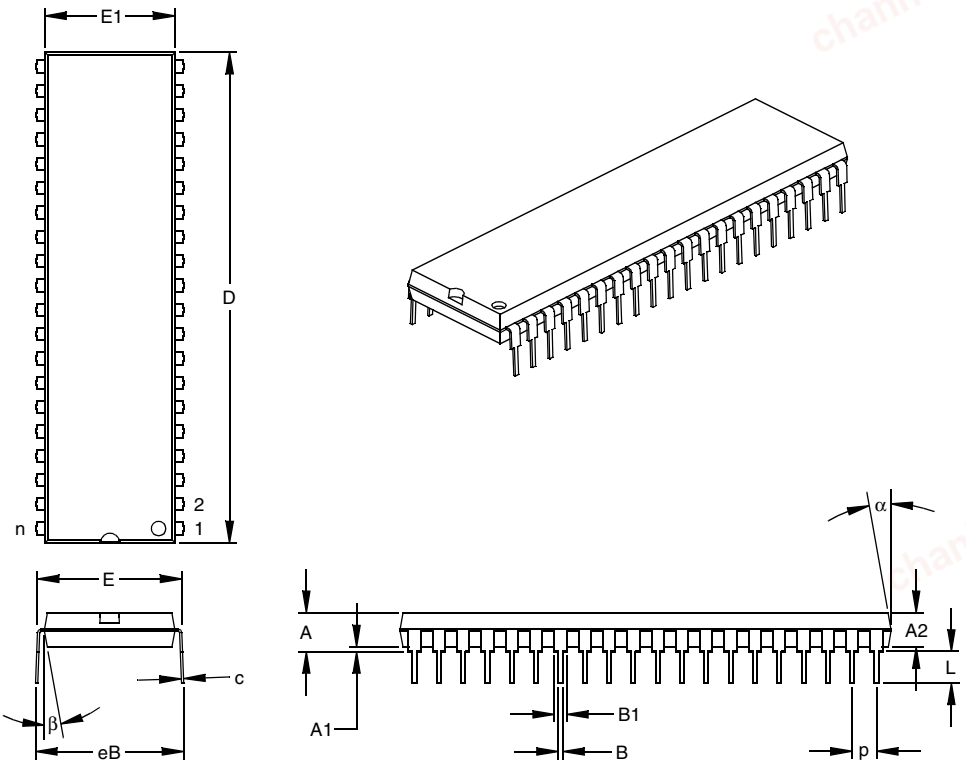

© 2006 Microchip Technology Inc.

chanhvinh.com



## 40-Lead Plastic Dual In-line (P) – 600 mil Body (PDIP)

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




| Units | | INCHES* | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 40 | | | 40 | |
| Pitch | p | | .100 | | | 2.54 | |
| Top to Seating Plane | A | .160 | .175 | .190 | 4.06 | 4.45 | 4.83 |
| Molded Package Thickness | A2 | .140 | .150 | .160 | 3.56 | 3.81 | 4.06 |
| Base to Seating Plane | A1 | .015 | | | 0.38 | | |
| Shoulder to Shoulder Width | E | .595 | .600 | .625 | 15.11 | 15.24 | 15.88 |
| Molded Package Width | E1 | .530 | .545 | .560 | 13.46 | 13.84 | 14.22 |
| Overall Length | D | 2.045 | 2.058 | 2.065 | 51.94 | 52.26 | 52.45 |
| Tip to Seating Plane | L | .120 | .130 | .135 | 3.05 | 3.30 | 3.43 |
| Lead Thickness | c | .008 | .012 | .015 | 0.20 | 0.29 | 0.38 |
| Upper Lead Width | B1 | .030 | .050 | .070 | 0.76 | 1.27 | 1.78 |
| Lower Lead Width | B | .014 | .018 | .022 | 0.36 | 0.46 | 0.56 |
| Overall Row Spacing § | eB | .620 | .650 | .680 | 15.75 | 16.51 | 17.27 |
| 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 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-011

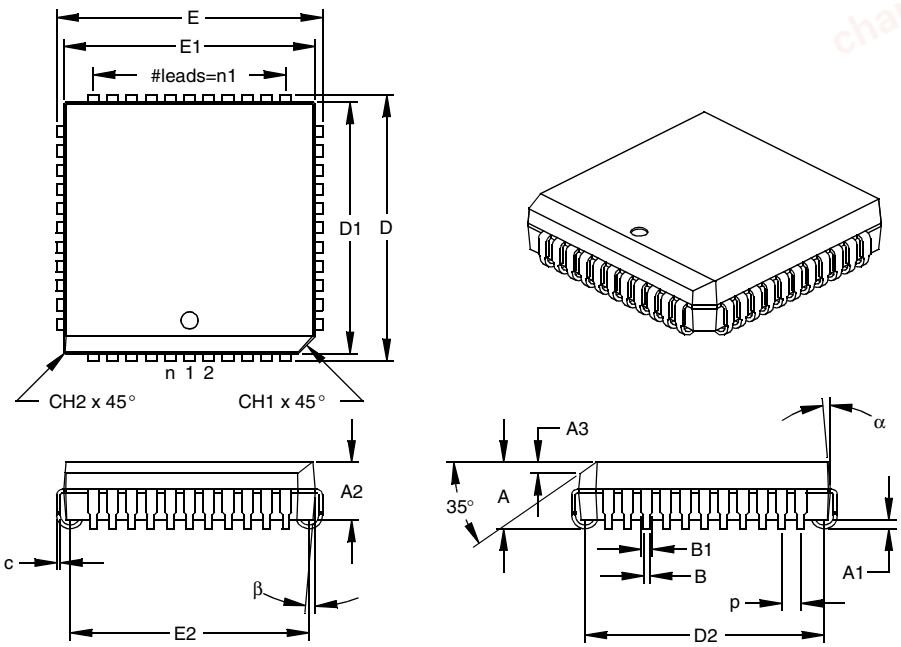
Drawing No. C04-016

chanhvinh.com



## 44-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)

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




| Units | | INCHES* | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 44 | | | 44 | |
| Pitch | p | | .050 | | | 1.27 | |
| Pins per Side | n1 | | 11 | | | 11 | |
| 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 | .685 | .690 | .695 | 17.40 | 17.53 | 17.65 |
| Overall Length | D | .685 | .690 | .695 | 17.40 | 17.53 | 17.65 |
| Molded Package Width | E1 | .650 | .653 | .656 | 16.51 | 16.59 | 16.66 |
| Molded Package Length | D1 | .650 | .653 | .656 | 16.51 | 16.59 | 16.66 |
| Footprint Width | E2 | .590 | .620 | .630 | 14.99 | 15.75 | 16.00 |
| Footprint Length | D2 | .590 | .620 | .630 | 14.99 | 15.75 | 16.00 |
| 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. C04-048

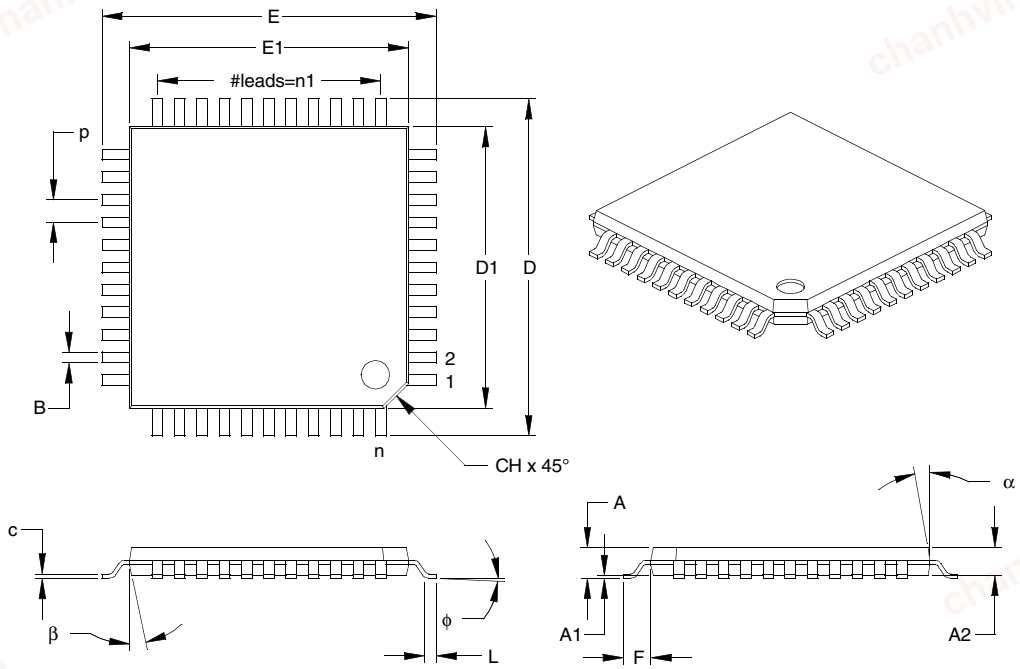

© 2006 Microchip Technology Inc.

chanhvinh.com



## 44-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

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




| Units | | INCHES | | | MILLIMETERS* | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 44 | | | 44 | |
| Pitch | p | | .031 | | | 0.80 | |
| Pins per Side | n1 | | 11 | | | 11 | |
| 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 REF. | | | 1.00 REF. | |
| Foot Angle | ϕ | 0 | 3.5 | 7 | 0 | 3.5 | 7 |
| Overall Width | E | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Overall Length | D | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Molded Package Width | E1 | .390 | .394 | .398 | 9.90 | 10.00 | 10.10 |
| Molded Package Length | D1 | .390 | .394 | .398 | 9.90 | 10.00 | 10.10 |
| Lead Thickness | c | .004 | .006 | .008 | 0.09 | 0.15 | 0.20 |
| Lead Width | B | .012 | .015 | .017 | 0.30 | 0.38 | 0.44 |
| 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

**Notes:**

Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

REF: Reference Dimension, usually without tolerance, for information purposes only.

See ASME Y14.5M

JEDEC Equivalent: MS-026

Drawing No. C04-076

Revised 07-22-05

chanhvinh.com



**NOTES:**

chanhvinh.com



© 2006 Microchip Technology Inc.

chanhvinh.com



## APPENDIX A: DATA SHEET REVISION HISTORY

### Revision A (June 2001)

Original data sheet for the PIC18FXX8 family.

### Revision B (May 2002)

Updated information on CAN module, device memory and register maps, I/O ports and Enhanced CCP.

### Revision C (January 2003)

This revision includes the DC and AC Characteristics Graphs and Tables (see **Section 28.0 "DC and AC Characteristics Graphs and Tables"**), **Section 27.0 "Electrical Characteristics"** have been updated and CAN certification information has been added.

### Revision D (September 2004)

Data Sheet Errata (DS80134 and DS80161) issues have been addressed and corrected along with minor corrections to the data sheet text.

### Revision E (October 2006)

Packaging diagrams updated.

## APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

**TABLE B-1: DEVICE DIFFERENCES**

| Features | | PIC18F248 | PIC18F258 | PIC18F448 | PIC18F458 |
|---|---|---|---|---|---|
| Internal Program Memory | Bytes | 16K | 32K | 16K | 32K |
| | # of Single-Word Instructions | 8192 | 16384 | 8192 | 16384 |
| Data Memory (Bytes) | | 768 | 1536 | 768 | 1536 |
| I/O Ports | | Ports A, B, C | Ports A, B, C | Ports A, B, C, D, E | Ports A, B, C, D, E |
| Enhanced Capture/Compare/PWM Modules | | — | — | 1 | 1 |
| Parallel Slave Port | | No | No | Yes | Yes |
| 10-bit Analog-to-Digital Converter | | 5 input channels | 5 input channels | 8 input channels | 8 input channels |
| Analog Comparators | | No | No | 2 | 2 |
| Analog Comparators VREF Output | | N/A | N/A | Yes | Yes |
| Packages | | 28-pin SPDIP<br>28-pin SOIC | 28-pin SPDIP<br>28-pin SOIC | 40-pin PDIP<br>44-pin PLCC<br>44-pin TQFP | 40-pin PDIP<br>44-pin PLCC<br>44-pin TQFP |

© 2006 Microchip Technology Inc.

chanhvinh.com



## 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 PIC18FXX8 family of devices.

### D.1 PIC16CXXX to PIC18FXX8

See Application Note *AN716 "Migrating Designs from PIC16C74A/74B to PIC18C442"* (DS00716).

### D.2 PIC17CXXX to PIC18FXX8

See Application Note *AN726 "PIC17CXXX to PIC18CXXX Migration"* (DS00726).


© 2006 Microchip Technology Inc.

chanhvinh.com



# INDEX

## A


A/D ..... 241
A/D Converter Flag (ADIF Bit) ..... 243
A/D Converter Interrupt, Configuring ..... 244
Acquisition Requirements ..... 244
Acquisition Time ..... 245
ADCON0 Register ..... 241
ADCON1 Register ..... 241
ADRESH Register ..... 241
ADRESH/ADRESL Registers ..... 243
ADRESL Register ..... 241
Analog Port Pins, Configuring ..... 246
Associated Registers Summary ..... 248
Calculating the Minimum Required Acquisition Time ..... 245
Configuring the Module ..... 244
Conversion Clock (TAD) ..... 246
Conversion Status (GO/DONE Bit) ..... 243
Conversion TAD Cycles ..... 248
Conversions ..... 247
Minimum Charging Time ..... 245
Result Registers ..... 247
Selecting the Conversion Clock ..... 246
Special Event Trigger (CCP) ..... 126
Special Event Trigger (ECCP) ..... 133, 248
TAD vs. Device Operating Frequencies (For Extended, LF Devices) (table) ..... 246
TAD vs. Device Operating Frequencies (table) ..... 246
Use of the ECCP Trigger ..... 248
Absolute Maximum Ratings ..... 329
AC (Timing) Characteristics ..... 341
Parameter Symbology ..... 341
Access Bank ..... 54
ACKSTAT ..... 173
ACKSTAT Status Flag ..... 173
ADCON0 Register ..... 241
GO/DONE Bit ..... 243
ADCON1 Register ..... 241
ADDLW ..... 287
Addressable Universal Synchronous Asynchronous Receiver Transmitter. *See* USART.
ADDWF ..... 287
ADDWFC ..... 288
ADRESH Register ..... 241
ADRESH/ADRESL Registers ..... 243
ADRESL Register ..... 241
Analog-to-Digital Converter. *See* A/D.
ANDLW ..... 288
ANDWF ..... 289
Assembler
MPASM Assembler ..... 323
Associated Registers ..... 192, 197


## B


Bank Select Register (BSR) ..... 54
Baud Rate Generator ..... 169
BC ..... 289
BCF ..... 290
BF ..... 173
BF Status Flag ..... 173
Block Diagrams
A/D ..... 243
Analog Input Model ..... 244, 253
Baud Rate Generator ..... 169
CAN Buffers and Protocol Engine ..... 200
CAN Receive Buffer ..... 230
CAN Transmit Buffer ..... 227
Capture Mode Operation ..... 125
Comparator I/O Operating Modes ..... 250
Comparator Output ..... 252
Comparator Voltage Reference Output Buffer Example ..... 257
Compare (CCP Module) Mode Operation ..... 126
Enhanced PWM ..... 134
Interrupt Logic ..... 78
Low-Voltage Detect (LVD) ..... 260
Low-Voltage Detect with External Input ..... 260
MSSP ($I^2C$ Master Mode) ..... 167
MSSP ($I^2C$ Mode) ..... 152
MSSP (SPI Mode) ..... 143
On-Chip Reset Circuit ..... 25
OSC2/CLKO/RA6 Pin ..... 94
PIC18F248/258 Architecture ..... 8
PIC18F448/458 Architecture ..... 9
PLL ..... 19
PORTC (Peripheral Output Override) ..... 100
PORTD and PORTE (Parallel Slave Port) ..... 107
PORTD in I/O Port Mode ..... 102
PORTE ..... 104
PWM (CCP Module) ..... 128
RA3:RA0 and RA5 Pins ..... 94
RA4/T0CKI Pin ..... 94
RB1:RB0 Pins ..... 97
RB2/CANTX/INT2 Pin ..... 98
RB3/CANRX Pin ..... 98
RB7:RB4 Pins ..... 97
Reads from Flash Program Memory ..... 69
Table Read Operation ..... 65
Table Write Operation ..... 66
Table Writes to Flash Program Memory ..... 71
Timer0 in 16-bit Mode ..... 110
Timer0 in 8-bit Mode ..... 110
Timer1 ..... 114
Timer1 (16-bit Read/Write Mode) ..... 114
Timer2 ..... 118
Timer3 ..... 120
Timer3 (16-bit Read/Write Mode) ..... 120
USART Receive ..... 191
USART Transmit ..... 189
Voltage Reference ..... 256
Watchdog Timer ..... 273
BN ..... 290
BNC ..... 291
BNN ..... 291
BNOV ..... 292
BNZ ..... 292
BOR. *See* Brown-out Reset.
BOV ..... 295
BRA ..... 293
BRG. *See* Baud Rate Generator.
Brown-out Reset (BOR) ..... 26, 265

chanhvinh.com



BSF ..... 293
BTFSC ..... 294
BTFSS ..... 294
BTG ..... 295
BZ ..... 296

## C

C Compilers
MPLAB C17 ..... 324
MPLAB C18 ..... 324
MPLAB C30 ..... 324
CALL ..... 296
CAN Module
Aborting Transmission ..... 228
Acknowledge Error ..... 237
Baud Rate Registers ..... 218
Baud Rate Setting ..... 233
Bit Error ..... 237
Bit Time Partitioning (diagram) ..... 233
Bit Timing Configuration Registers ..... 236
BRGCON1 ..... 236
BRGCON2 ..... 236
BRGCON3 ..... 236
Calculating TQ, Nominal Bit Rate and Nominal Bit Time ..... 234
Configuration Mode ..... 226
Control and Status Registers ..... 201
Controller Register Map ..... 225
CRC Error ..... 237
Disable Mode ..... 226
Error Detection ..... 237
Error Modes and Error Counters ..... 237
Error Modes State (diagram) ..... 238
Error Recognition Mode ..... 227
Error States ..... 237
Filter/Mask Truth (table) ..... 232
Form Error ..... 237
Hard Synchronization ..... 235
I/O Control Register ..... 221
Information Processing Time ..... 234
Initiating Transmission ..... 228
Internal Message Reception Flowchart ..... 231
Internal Transmit Message Flowchart ..... 229
Interrupt Acknowledge ..... 239
Interrupt Registers ..... 222
Interrupts ..... 238
Bus Activity Wake-up ..... 239
Bus-Off ..... 239
Code Bits ..... 238
Error ..... 239
Message Error ..... 239
Receive ..... 238
Receiver Bus Passive ..... 239
Receiver Overflow ..... 239
Receiver Warning ..... 239
Transmit ..... 238
Transmitter Bus Passive ..... 239
Transmitter Warning ..... 239
Lengthening a Bit Period (diagram) ..... 235
Listen Only Mode ..... 226
Loopback Mode ..... 227
Message Acceptance Filters and Masks ..... 215, 232
Message Acceptance Mask and Filter Operation (diagram) ..... 232
Message Reception ..... 230
Message Time-Stamping ..... 230
Message Transmission ..... 227
Modes of Operation ..... 226
Normal Mode ..... 226
Oscillator Tolerance ..... 236
Overview ..... 199
Phase Buffer Segments ..... 234
Programming Time Segments ..... 236
Propagation Segment ..... 234
Receive Buffer Registers ..... 210
Receive Buffers ..... 230
Receive Message Buffering ..... 230
Receive Priority ..... 230
Registers ..... 201
Resynchronization ..... 235
Sample Point ..... 234
Shortening a Bit Period (diagram) ..... 236
Stuff Bit Error ..... 237
Synchronization ..... 235
Synchronization Rules ..... 235
Synchronization Segment ..... 234
Time Quanta ..... 234
Transmit Buffer Registers ..... 206
Transmit Buffers ..... 227
Transmit Priority ..... 227
Transmit/Receive Buffers ..... 199
Values for ICODE (table) ..... 239
Capture (CCP Module) ..... 124
CAN Message Time-Stamp ..... 125
CCP Pin Configuration ..... 124
CCP1 Prescaler ..... 125
CCPR1H:CCPR1L Registers ..... 124
Software Interrupt ..... 125
Timer1/Timer3 Mode Selection ..... 124
Capture (ECCP Module) ..... 133
CAN Message Time-Stamp ..... 133
Capture/Compare/PWM (CCP) ..... 123
Capture Mode. *See* Capture (CCP Module).
CCP1 Module ..... 124
Timer Resources ..... 124
CCPR1H Register ..... 124
CCPR1L Register ..... 124
Compare Mode. *See* Compare (CCP Module).
Interaction of CCP1 and ECCP1 Modules ..... 124
PWM Mode. *See* PWM (CCP Module).
Ceramic Resonators
Ranges Tested ..... 17
Clocking Scheme ..... 41
CLRF ..... 297
CLRWDT ..... 297


© 2006 Microchip Technology Inc.

chanhvinh.com




Code Examples
16 x 16 Signed Multiply Routine ..... 76
16 x 16 Unsigned Multiply Routine ..... 76
8 x 8 Signed Multiply Routine ..... 75
8 x 8 Unsigned Multiply Routine ..... 75
Changing Between Capture Prescalers ..... 125
Data EEPROM Read ..... 61
Data EEPROM Refresh Routine ..... 62
Data EEPROM Write ..... 61
Erasing a Flash Program Memory Row ..... 70
Fast Register Stack ..... 40
How to Clear RAM (Bank 1) Using Indirect Addressing ..... 55
Initializing PORTA ..... 93
Initializing PORTB ..... 96
Initializing PORTC ..... 100
Initializing PORTD ..... 102
Initializing PORTE ..... 104
Loading the SSPBUF Register ..... 146
Reading a Flash Program Memory Word ..... 69
Saving Status, WREG and BSR Registers in RAM ..... 92
WIN and ICODE Bits Usage in Interrupt Service Routine to Access TX/RX Buffers ..... 203
Writing to Flash Program Memory ..... 72–73
Code Protection ..... 265
COMF ..... 298
Comparator Module ..... 249
Analog Input Connection Considerations ..... 253
Associated Registers ..... 254
Configuration ..... 250
Effects of a Reset ..... 253
External Reference Signal ..... 251
Internal Reference Signal ..... 251
Interrupts ..... 252
Operation ..... 251
Operation During Sleep ..... 253
Outputs ..... 251
Reference ..... 251
Response Time ..... 251
Comparator Specifications ..... 340
Comparator Voltage Reference Module ..... 255
Accuracy/Error ..... 256
Associated Registers ..... 257
Configuring ..... 255
Connection Considerations ..... 256
Effects of a Reset ..... 256
Operation During Sleep ..... 256
Compare (CCP Module) ..... 126
CCP1 Pin Configuration ..... 126
CCPR1 and ECCPR1 Registers ..... 126
Registers Associated with Capture, Compare, Timer1 and Timer3 ..... 127
Software Interrupt ..... 126
Special Event Trigger ..... 115, 121, 126, 248
Timer1/Timer3 Mode Selection ..... 126
Compare (ECCP Module) ..... 133
Registers Associated with Enhanced Capture, Compare, Timer1 and Timer3 ..... 133
Special Event Trigger ..... 133
Compatible 10-Bit Analog-to-Digital Converter (A/D) Module. *See* A/D.
Configuration Mode (CAN Module) ..... 226
CPFSEQ ..... 298
CPFSGT ..... 299
CPFSLT ..... 299
Crystal Oscillator
Capacitor Selection ..... 18

**D**

Data EEPROM Memory ..... 59
Associated Registers ..... 63
EEADR Register ..... 59
EECON1 Register ..... 59
EECON2 Register ..... 59
Operation During Code-Protect ..... 62
Protection Against Spurious Writes ..... 62
Reading ..... 61
Usage ..... 62
Write Verify ..... 62
Writing to ..... 61
Data Memory ..... 44
General Purpose Registers ..... 44
Special Function Registers ..... 44
Data Memory Map
PIC18F248/448 ..... 45
PIC18F258/458 ..... 46
DAW ..... 300
DC and AC Characteristics
Graphs and Tables ..... 361
DC Characteristics ..... 332
EEPROM and Enhanced Flash ..... 339
PIC18FXX8 (Ind., Ext.) and PIC18LFXX8 (Ind.) ..... 336
DCFSNZ ..... 301
DECF ..... 300
DECFSZ ..... 301
Demonstration Boards
PICDEM 1 ..... 326
PICDEM 17 ..... 327
PICDEM 18R ..... 327
PICDEM 2 Plus ..... 326
PICDEM 3 ..... 326
PICDEM 4 ..... 326
PICDEM LIN ..... 327
PICDEM USB ..... 327
PICDEM.net Internet/Ethernet ..... 326
Development Support ..... 323
Device Differences ..... 385
Device Migrations ..... 386
Device Overview ..... 7
Features ..... 7
Direct Addressing ..... 56
Disable Mode (CAN Module) ..... 226

**E**

Electrical Characteristics ..... 329
Enhanced Capture/Compare/PWM (ECCP) ..... 131
Auto-Shutdown ..... 142
Capture Mode. *See* Capture (ECCP Module).
Compare Mode. *See* Compare (ECCP Module).
ECCPR1H Register ..... 132
ECCPR1L Register ..... 132
Interaction of CCP1 and ECCP1 Modules ..... 132
Pin Assignments for Various Modes ..... 132
PWM Mode. *See* PWM (ECCP Module).
Timer Resources ..... 132

chanhvinh.com



Enhanced CCP Auto-Shutdown ..... 142
Enhanced PWM Mode.
*See* PWM (ECCP Module).
Errata ..... 5
Error Recognition Mode (CAN Module) ..... 226
Evaluation and Programming Tools ..... 327
External Clock Input ..... 19

**F**

Firmware Instructions ..... 281
Flash Program Memory ..... 65
Associated Registers ..... 74
Control Registers ..... 66
Erase Sequence ..... 70
Erasing ..... 70
Operation During Code-Protect ..... 73
Reading ..... 69
TABLAT (Table Latch) Register ..... 68
Table Pointer
Boundaries Based on Operation ..... 68
Table Pointer Boundaries ..... 68
Table Reads and Table Writes ..... 65
TBLPTR (Table Pointer) Register ..... 68
Write Sequence ..... 71
Writing to ..... 71
Protection Against Spurious Writes ..... 73
Unexpected Termination ..... 73
Write Verify ..... 73

**G**

GOTO ..... 302

**H**

Hardware Multiplier ..... 75
Operation ..... 75
Performance Comparison (table) ..... 75
HS4 (PLL) ..... 19

**I**

I/O Ports ..... 93
$I^2C$ Mode ..... 152
$\overline{ACK}$ Pulse ..... 156, 157
Acknowledge Sequence Timing ..... 176
Baud Rate Generator ..... 169
Bus Collision During a
Repeated Start Condition ..... 180
Bus Collision During a Start Condition ..... 178
Bus Collision During a Stop Condition ..... 181
Clock Arbitration ..... 170
Clock Stretching ..... 162
Effect of a Reset ..... 177
General Call Address Support ..... 166
Master Mode ..... 167
Operation ..... 168
Reception ..... 173
Repeated Start Condition Timing ..... 172
Start Condition Timing ..... 171
Transmission ..... 173
Multi-Master Mode ..... 177
Communication, Bus Collision
and Bus Arbitration ..... 177
Operation ..... 156
Read/$\overline{Write}$ Bit Information (R/$\overline{W}$ Bit) ..... 156, 157
Registers ..... 152
Serial Clock (RC3/SCK/SCL) ..... 157
Slave Mode ..... 156
Addressing ..... 156
Reception ..... 157
Transmission ..... 157
Sleep Operation ..... 177
Stop Condition Timing ..... 176
ID Locations ..... 265, 279
INCF ..... 302
INCFSZ ..... 303
In-Circuit Debugger ..... 279
In-Circuit Serial Programming (ICSP) ..... 265, 279
Indirect Addressing ..... 56
FSR Register ..... 55
INDF Register ..... 55
Operation ..... 55
INFSNZ ..... 303
Initialization Conditions for All Registers ..... 30
Instruction Cycle ..... 41
Instruction Flow/Pipelining ..... 41
Instruction Format ..... 283
Instruction Set ..... 281
ADDLW ..... 287
ADDWF ..... 287
ADDWFC ..... 288
ANDLW ..... 288
ANDWF ..... 289
BC ..... 289
BCF ..... 290
BN ..... 290
BNC ..... 291
BNN ..... 291
BNOV ..... 292
BNZ ..... 292
BOV ..... 295
BRA ..... 293
BSF ..... 293
BTFSC ..... 294
BTFSS ..... 294
BTG ..... 295
BZ ..... 296
CALL ..... 296
CLRF ..... 297
CLRWDT ..... 297
COMF ..... 298
CPFSEQ ..... 298
CPFSGT ..... 299
CPFSLT ..... 299
DAW ..... 300
DCFSNZ ..... 301
DECF ..... 300
DECFSZ ..... 301
GOTO ..... 302
INCF ..... 302
INCFSZ ..... 303
INFSNZ ..... 303
IORLW ..... 304
IORWF ..... 304
LFSR ..... 305
MOVF ..... 305
MOVFF ..... 306
MOVLB ..... 306
MOVLW ..... 307
MOVWF ..... 307
MULLW ..... 308
MULWF ..... 308


© 2006 Microchip Technology Inc.

chanhvinh.com




NEGF ..... 309
NOP ..... 309
POP ..... 310
PUSH ..... 310
RCALL ..... 311
RESET ..... 311
RETFIE ..... 312
RETLW ..... 312
RETURN ..... 313
RLCF ..... 313
RLNCF ..... 314
RRCF ..... 314
RRNCF ..... 315
SETF ..... 315
SLEEP ..... 316
SUBFWB ..... 316
SUBLW ..... 317
SUBWF ..... 317
SUBWFB ..... 318
SWAPF ..... 318
TBLRD ..... 319
TBLWT ..... 320
TSTFSZ ..... 321
XORLW ..... 321
XORWF ..... 322
Summary Table ..... 284
INTCON Register
RBIF Bit ..... 96
Inter-Integrated Circuit. *See* I²C.
Interrupt Sources
A/D Conversion Complete ..... 244
CAN Module ..... 238
Capture Complete (CCP) ..... 125
Compare Complete (CCP) ..... 126
Interrupt-on-Change (RB7:RB4) ..... 96
TMR0 Overflow ..... 111
TMR1 Overflow ..... 113, 115
TMR2 to PR2 Match ..... 118
TMR2 to PR2 Match (PWM) ..... 117, 128
TMR3 Overflow ..... 119, 121
Interrupt-on-Change (RB7:RB4) Flag
(RBIF Bit) ..... 96
Interrupts ..... 77, 265
Context Saving During ..... 92
Enable Registers ..... 85
Flag Registers ..... 82
INT ..... 92
PORTB Interrupt-on-Change ..... 92
Priority Registers ..... 88
TMR0 ..... 92
Interrupts, Flag Bits
A/D Converter Flag (ADIF Bit) ..... 243
CCP1 Flag (CCP1IF Bit) ..... 124, 125, 126
IORLW ..... 304
IORWF ..... 304


## L


LFSR ..... 305
Listen Only Mode (CAN Module) ..... 226
Look-up Tables ..... 43
Computed GOTO ..... 43
Table Reads/Table Writes ..... 43
Loopback Mode (CAN Module) ..... 226
Low-Voltage Detect ..... 259
Characteristics ..... 338
Current Consumption ..... 263
Effects of a Reset ..... 263
Operation ..... 262
Operation During Sleep ..... 263
Reference Voltage Set Point ..... 263
Typical Application ..... 259
Low-Voltage ICSP Programming ..... 279
LVD. *See* Low-Voltage Detect.


## M


Master Synchronous Serial Port (MSSP).
*See* MSSP.
Master Synchronous Serial Port.
*See* MSSP.
Memory Organization ..... 37
Data Memory ..... 44
Internal Program Memory Operation ..... 37
Program Memory ..... 37
Migrating from other PICmicro Devices ..... 386
MOVF ..... 305
MOVFF ..... 306
MOVLB ..... 306
MOVLW ..... 307
MOVWF ..... 307
MPLAB ASM30 Assembler,
Linker, Librarian ..... 324
MPLAB ICD 2 In-Circuit Debugger ..... 325
MPLAB ICE 2000 High-Performance
Universal In-Circuit Emulator ..... 325
MPLAB ICE 4000 High-Performance
Universal In-Circuit Emulator ..... 325
MPLAB Integrated Development
Environment Software ..... 323
MPLAB PM3 Device Programmer ..... 325
MPLINK Object Linker/
MPLIB Object Librarian ..... 324
MSSP ..... 143
Control Registers ..... 143
Enabling SPI I/O ..... 147
Operation ..... 146
Overview ..... 143
SPI Master Mode ..... 148
SPI Master/Slave Connection ..... 147
SPI Mode ..... 143
SPI Slave Mode ..... 149
TMR2 Output for Clock Shift ..... 117, 118
Typical Connection ..... 147
MSSP. *See also* I²C Mode, SPI Mode.
MULLW ..... 308
MULWF ..... 308


## N


NEGF ..... 309
NOP ..... 309
Normal Operation Mode (CAN Module) ..... 226

chanhvinh.com



## O


Opcode Field Descriptions ..... 282
Oscillator
  Effects of Sleep Mode ..... 23
  Power-up Delays ..... 23
  Switching Feature ..... 20
  System Clock Switch Bit ..... 20
  Transitions ..... 21
Oscillator Configurations ..... 17
  Crystal Oscillator, Ceramic Resonators ..... 17
  EC ..... 17
  ECIO ..... 17
  HS ..... 17
  HS4 ..... 17
  LP ..... 17
  RC ..... 17, 18
  RCIO ..... 17
  XT ..... 17
Oscillator Selection ..... 265
Oscillator, Timer1 ..... 113, 115, 121
Oscillator, WDT ..... 272


## P


Packaging Information ..... 377
  Details ..... 379
  Marking ..... 377
Parallel Slave Port (PSP) ..... 102, 107
  Associated Registers ..... 108
  PORTD ..... 107
  PSP Mode Select (PSPMODE) Bit ..... 102
  RE2/AN7/$\overline{CS}$/C2OUT ..... 107
PIC18FXX8 Voltage-Frequency
  Graph (Industrial) ..... 330
PIC18LFXX8 Voltage-Frequency
  Graph (Industrial) ..... 331
PICkit 1 Flash Starter Kit ..... 327
PICSTART Plus Development
  Programmer ..... 326
Pin Functions
  $\overline{MCLR}$/VPP ..... 10
  OSC1/CLKI ..... 10
  OSC2/CLKO/RA6 ..... 10
  RA0/AN0/CVREF ..... 11
  RA1/AN1 ..... 11
  RA2/AN2/VREF- ..... 11
  RA3/AN3/VREF+ ..... 11
  RA4/T0CKI ..... 11
  RA5/AN4/$\overline{SS}$/LVDIN ..... 11
  RA6 ..... 11
  RB0/INT0 ..... 12
  RB1/INT1 ..... 12
  RB2/CANTX/INT2 ..... 12
  RB3/CANRX ..... 12
  RB4 ..... 12
  RB5/PGM ..... 12
  RB6/PGC ..... 12
  RB7/PGD ..... 12
  RC0/T1OSO/T1CKI ..... 13
  RC1/T1OSI ..... 13
  RC2/CCP1 ..... 13
  RC3/SCK/SCL ..... 13
  RC4/SDI/SDA ..... 13
  RC5/SDO ..... 13
  RC6/TX/CK ..... 13
  RC7/RX/DT ..... 13
  RD0/PSP0/C1IN+ ..... 14
  RD1/PSP1/C1IN- ..... 14
  RD2/PSP2/C2IN+ ..... 14
  RD3/PSP3/C2IN- ..... 14
  RD4/PSP4/ECCP1/P1A ..... 14
  RD5/PSP5/P1B ..... 14
  RD6/PSP6/P1C ..... 14
  RD7/PSP7/P1D ..... 14
  RE0/AN5/$\overline{RD}$ ..... 15
  RE1/AN6/$\overline{WR}$/C1OUT ..... 15
  RE2/AN7/$\overline{CS}$/C2OUT ..... 15
  VDD ..... 15
  VSS ..... 15
Pinout I/O Descriptions ..... 10
Pointer, FSRn ..... 55
POP ..... 310
POR. *See* Power-on Reset.
PORTA
  Associated Register Summary ..... 95
  Functions ..... 95
  LATA Register ..... 93
  PORTA Register ..... 93
  TRISA Register ..... 93
PORTB
  Associated Register Summary ..... 99
  Functions ..... 99
  LATB Register ..... 96
  PORTB Register ..... 96
  RB7:RB4 Interrupt-on-Change
    Flag (RBIF Bit) ..... 96
  TRISB Register ..... 96
PORTC
  Associated Register Summary ..... 101
  Functions ..... 101
  LATC Register ..... 100
  PORTC Register ..... 100
  RC3/SCK/SCL Pin ..... 157
  RC7/RX/DT pin ..... 185
  TRISC Register ..... 100, 183
PORTD
  Associated Register Summary ..... 103
  Functions ..... 103
  LATD Register ..... 102
  Parallel Slave Port (PSP) Function ..... 102
  PORTD Register ..... 102
  TRISD Register ..... 102
PORTE
  Associated Register Summary ..... 106
  Functions ..... 106
  LATE Register ..... 104
  PORTE Register ..... 104
  PSP Mode Select (PSPMODE) Bit ..... 102
  RE2/AN7/$\overline{CS}$/C2OUT ..... 107
  TRISE Register ..... 104
Power-Down Mode. *See* Sleep.
Power-on Reset (POR) ..... 26, 265
  $\overline{MCLR}$ ..... 26
  Oscillator Start-up Timer (OST) ..... 26, 265
  PLL Lock Time-out ..... 26
  Power-up Timer (PWRT) ..... 26, 265
  Time-out Sequence ..... 27
Power-up Delays
  OSC1 and OSC2 Pin States
    in Sleep Mode ..... 23
Prescaler, Timer0 ..... 111



© 2006 Microchip Technology Inc.

chanhvinh.com




Prescaler, Timer2 ..... 128
PRO MATE II Universal Device Programmer ..... 325
Program Counter
PCL Register ..... 40
PCLATH Register ..... 40
PCLATU Register ..... 40
Program Memory ..... 37
Fast Register Stack ..... 40
Instructions ..... 41
Two-Word ..... 43
Map and Stack for PIC18F248/448 ..... 37
Map and Stack for PIC18F258/458 ..... 37
PUSH and POP Instructions ..... 40
Return Address Stack ..... 38
Return Stack Pointer (STKPTR) ..... 38
Stack Full/Underflow Resets ..... 40
Top-of-Stack Access ..... 38
Program Verification and Code Protection ..... 276
Associated Registers Summary ..... 276
Configuration Register Protection ..... 279
Data EEPROM Code Protection ..... 279
Program Memory Code Protection ..... 277
Programming, Device Instructions ..... 281
PUSH ..... 310
PWM (CCP Module) ..... 128
CCPR1H:CCPR1L Registers ..... 128
Duty Cycle ..... 128
Example Frequencies/Resolutions ..... 129
Period ..... 128
Registers Associated with PWM and Timer2 ..... 129
Setup for PWM Operation ..... 129
TMR2 to PR2 Match ..... 117, 128
PWM (ECCP Module) ..... 134
Full-Bridge Application Example ..... 138
Full-Bridge Mode ..... 137
Direction Change ..... 138
Half-Bridge Mode ..... 136
Half-Bridge Output Mode Applications Example ..... 136
Output Configurations ..... 134
Output Polarity Configuration ..... 140
Output Relationships Diagram ..... 135
Programmable Dead-Band Delay ..... 140
Registers Associated with Enhanced PWM and Timer2 ..... 141
Setup for PWM Operation ..... 141
Standard Mode ..... 134
Start-up Considerations ..... 140
System Implementation ..... 140

## Q

Q Clock ..... 128

## R

RAM. *See* Data Memory.
RCALL ..... 311
RCON Register
Significance of Status Bits vs. Initialization Condition ..... 27
RCSTA Register ..... 183
SPEN Bit ..... 183
Receiver Warning ..... 239
Register File ..... 44
Register File Summary ..... 49
Registers
ADCON0 (A/D Control 0) ..... 241
ADCON1 (A/D Control 1) ..... 242
BRGCON1 (Baud Rate Control 1) ..... 218
BRGCON2 (Baud Rate Control 2) ..... 219
BRGCON3 (Baud Rate Control 3) ..... 220
CANCON (CAN Control) ..... 201
CANSTAT (CAN Status) ..... 202
CCP1CON (CCP1 Control) ..... 123
CIOCON (CAN I/O Control) ..... 221
CMCON (Comparator Control) ..... 249
COMSTAT (CAN Communication Status) ..... 205
CONFIG1H (Configuration 1 High) ..... 266
CONFIG2H (Configuration 2 High) ..... 267
CONFIG2L (Configuration 2 Low) ..... 266
CONFIG4L (Configuration 4 Low) ..... 267
CONFIG5H (Configuration 5 High) ..... 268
CONFIG5L (Configuration 5 Low) ..... 268
CONFIG6H (Configuration 6 High) ..... 269
CONFIG6L (Configuration 6 Low) ..... 269
CONFIG7H (Configuration 7 High) ..... 270
CONFIG7L (Configuration 7 Low) ..... 270
CVRCON (Comparator Voltage Reference Control) ..... 255
DEVID1 (Device ID 1) ..... 271
DEVID2 (Device ID 2) ..... 271
ECCP1CON (ECCP1 Control) ..... 131
ECCP1DEL (PWM Delay) ..... 140
ECCPAS (Enhanced Capture/Compare/PWM Auto-Shutdown Control) ..... 142
EECON1 (EEPROM Control 1) ..... 60, 67
INTCON (Interrupt Control) ..... 79
INTCON2 (Interrupt Control 2) ..... 80
INTCON3 (Interrupt Control 3) ..... 81
IPR1 (Peripheral Interrupt Priority 1) ..... 88
IPR2 (Peripheral Interrupt Priority 2) ..... 89
IPR3 (Peripheral Interrupt Priority 3) ..... 90, 224
LVDCON (LVD Control) ..... 261
OSCCON (Oscillator Control) ..... 20
PIE1 (Peripheral Interrupt Enable 1) ..... 85
PIE2 (Peripheral Interrupt Enable 2) ..... 86
PIE3 (Peripheral Interrupt Enable 3) ..... 87, 223
PIR1 (Peripheral Interrupt Request (Flag) 1) ..... 82
PIR2 (Peripheral Interrupt Request (Flag) 2) ..... 83
PIR3 (Peripheral Interrupt Request (Flag) 3) ..... 84, 222
RCON (Reset Control) ..... 58, 91
RCSTA (Receive Status and Control) ..... 184
RXB0CON (Receive Buffer 0 Control) ..... 210
RXB1CON (Receive Buffer 1 Control) ..... 211
RXBnDLC (Receive Buffer n Data Length Code) ..... 213
RXBnDm (Receive Buffer n Data Field Byte m) ..... 214
RXBnEIDH (Receive Buffer n Extended Identifier, High Byte) ..... 212
RXBnEIDL (Receive Buffer n Extended Identifier, Low Byte) ..... 213
RXBnSIDH (Receive Buffer n Standard Identifier, High Byte) ..... 212

chanhvinh.com




RXBnSIDL (Receive Buffer n Standard Identifier, Low Byte) ..... 212
RXERRCNT (Receive Error Count) ..... 214
RXFnEIDH (Receive Acceptance Filter n Extended Identifier, High Byte) ..... 216
RXFnEIDL (Receive Acceptance Filter n Extended Identifier, Low Byte) ..... 216
RXFnSIDH (Receive Acceptance Filter n Standard Identifier Filter, High Byte) ..... 215
RXFnSIDL (Receive Acceptance Filter n Standard Identifier Filter, Low Byte) ..... 215
RXMnEIDH (Receive Acceptance Mask n Extended Identifier Mask, High Byte) ..... 217
RXMnEIDL (Receive Acceptance Mask n Extended Identifier Mask, Low Byte) ..... 217
RXMnSIDH (Receive Acceptance Mask n Standard Identifier Mask, High Byte) ..... 216
RXMnSIDL (Receive Acceptance Mask n Standard Identifier Mask, Low Byte) ..... 217
SSPCON1 (MSSP Control 1, I2C Mode) ..... 154
SSPCON1 (MSSP Control 1, SPI Mode) ..... 145
SSPCON2 (MSSP Control 2, I2C Mode) ..... 155
SSPSTAT (MSSP Status, I2C Mode) ..... 153
SSPSTAT (MSSP Status, SPI Mode) ..... 144
Status ..... 57
STKPTR (Stack Pointer) ..... 39
T0CON (Timer0 Control) ..... 109
T1CON (Timer1 Control) ..... 113
T2CON (Timer2 Control) ..... 117
T3CON (Timer3 Control) ..... 119
TRISE (PORTE Direction/PSP Control) ..... 105
TXBnCON (Transmit Buffer n Control) ..... 206
TXBnDLC (Transmit Buffer n Data Length Code) ..... 209
TXBnDm (Transmit Buffer n Data Field Byte m) ..... 208
TXBnEIDH (Transmit Buffer n Extended Identifier, High Byte) ..... 207
TXBnEIDL (Transmit Buffer n Extended Identifier, Low Byte) ..... 208
TXBnSIDH (Transmit Buffer n Standard Identifier, High Byte) ..... 207
TXBnSIDL (Transmit Buffer n Standard Identifier, Low Byte) ..... 207
TXERRCNT (Transmit Error Count) ..... 209
TXSTA (Transmit Status and Control) ..... 183
WDTCON (Watchdog Timer Control) ..... 272
RESET ..... 311
Reset ..... 25, 265
MCLR Reset During Normal Operation ..... 25
MCLR Reset During Sleep ..... 25
Power-on Reset (POR) ..... 25
Programmable Brown-out Reset (PBOR) ..... 25
RESET Instruction ..... 25
Stack Full Reset ..... 25
Stack Underflow Reset ..... 25
Watchdog Timer (WDT) Reset ..... 25
RETFIE ..... 312
RETLW ..... 312
RETURN ..... 313
Revision History ..... 385
RLCF ..... 313
RLNCF ..... 314
RRCF ..... 314
RRNCF ..... 315


## S


SCI. *See* USART.
SCK Pin ..... 143
SDI Pin ..... 143
SDO Pin ..... 143
Serial Clock (SCK) Pin ..... 143
Serial Communication Interface. *See* USART.
Serial Peripheral Interface. *See* SPI.
SETF ..... 315
Slave Select (SS) Pin ..... 143
Slave Select, SS Pin ..... 143
SLEEP ..... 316
Sleep ..... 265, 274
Software Simulator (MPLAB SIM) ..... 324
Software Simulator (MPLAB SIM30) ..... 324
Special Event Trigger. *See* Compare.
Special Features of the CPU ..... 265
Configuration Bits ..... 265
Configuration Bits and Device IDs ..... 265
Configuration Registers ..... 266–271
Special Function Register Map ..... 47
Special Function Registers ..... 44
SPI Mode
Associated Registers ..... 151
Bus Mode Compatibility ..... 151
Effects of a Reset ..... 151
Master Mode ..... 148
Master/Slave Connection ..... 147
Registers ..... 144
Serial Clock ..... 143
Serial Data In (SDI) Pin ..... 143
Serial Data Out (SDO) Pin ..... 143
Slave Select ..... 143
Slave Select Synchronization ..... 149
Sleep Operation ..... 151
SPI Clock ..... 148
SSPBUF Register ..... 148
SSPSR Register ..... 148
SSPOV ..... 173
SSPOV Status Flag ..... 173
SSPSTAT Register
R/W Bit ..... 156, 157
SUBFWB ..... 316
SUBLW ..... 317
SUBWF ..... 317
SUBWFB ..... 318
SWAPF ..... 318


## T


Table Pointer Operations (table) ..... 68
TBLRD ..... 319
TBLWT ..... 320
Timer0 ..... 109
16-bit Mode Timer Reads and Writes ..... 111
Associated Registers ..... 111
Operation ..... 111
Overflow Interrupt ..... 111
Prescaler ..... 111
Prescaler. *See* Prescaler, Timer0.
Switching Prescaler Assignment ..... 111



© 2006 Microchip Technology Inc.

chanhvinh.com




Timer1 ..... 113
16-bit Read/Write Mode ..... 115
Associated Registers ..... 116
Operation ..... 114
Oscillator ..... 113, 115
Overflow Interrupt ..... 113, 115
Special Event Trigger (CCP) ..... 115, 126
Special Event Trigger (ECCP) ..... 133
TMR1H Register ..... 113
TMR1L Register ..... 113
TMR3L Register ..... 119
Timer2 ..... 117
Associated Registers ..... 118
Operation ..... 117
Postscaler. *See* Postscaler, Timer2.
PR2 Register ..... 117, 128
Prescaler. *See* Prescaler, Timer2.
SSP Clock Shift ..... 117, 118
TMR2 Register ..... 117
TMR2 to PR2 Match Interrupt ..... 117, 118, 128
Timer3 ..... 119
Associated Registers ..... 121
Operation ..... 120
Oscillator ..... 121
Overflow Interrupt ..... 119, 121
Special Event Trigger (CCP) ..... 121
TMR3H Register ..... 119
Timing Conditions ..... 342
Load Conditions for Device Timing Specifications ..... 342
Temperature and Voltage Specifications – AC ..... 342
Timing Diagrams
A/D Conversion ..... 359
Acknowledge Sequence ..... 176
Baud Rate Generator with Clock Arbitration ..... 170
BRG Reset Due to SDA Arbitration During Start Condition ..... 179
Brown-out Reset (BOR) and Low-Voltage Detect ..... 345
Bus Collision During a Repeated Start Condition (Case 1) ..... 180
Bus Collision During a Repeated Start Condition (Case2) ..... 180
Bus Collision During a Stop Condition (Case 1) ..... 181
Bus Collision During a Stop Condition (Case 2) ..... 181
Bus Collision During Start Condition (SCL = 0) ..... 179
Bus Collision During Start Condition (SDA Only) ..... 178
Bus Collision for Transmit and Acknowledge ..... 177
Capture/Compare/PWM (CCP1 and ECCP1) ..... 347
CLKO and I/O ..... 344
Clock Synchronization ..... 163
Clock/Instruction Cycle ..... 41
External Clock ..... 343
First Start Bit ..... 171
Full-Bridge PWM Output ..... 137
Half-Bridge PWM Output ..... 136
I²C Bus Data ..... 353
I²C Bus Start/Stop Bits ..... 353
I²C Master Mode (Reception, 7-bit Address) ..... 175
I²C Master Mode (Transmission, 7 or 10-bit Address) ..... 174
I²C Slave Mode (Transmission, 10-bit Address) ..... 161
I²C Slave Mode (Transmission, 7-bit Address) ..... 159
I²C Slave Mode with SEN = 0 (Reception, 10-bit Address) ..... 160
I²C Slave Mode with SEN = 0 (Reception, 7-bit Address) ..... 158
I²C Slave Mode with SEN = 1 (Reception, 10-bit Address) ..... 165
I²C Slave Mode with SEN = 1 (Reception, 7-bit Address) ..... 164
Low-Voltage Detect ..... 262
Master SSP I²C Bus Data ..... 355
Master SSP I²C Bus Start/Stop Bits ..... 355
Parallel Slave Port (PIC18F248 and PIC18F458) ..... 348
Parallel Slave Port Read ..... 108
Parallel Slave Port Write ..... 107
PWM Direction Change ..... 139
PWM Direction Change at Near 100% Duty Cycle ..... 139
PWM Output ..... 128
Repeated Start Condition ..... 172
Reset, Watchdog Timer (WDT), Oscillator Start-up Timer (OST), Power-up Timer (PWRT) ..... 345
Slave Mode General Call Address Sequence (7 or 10-bit Address Mode) ..... 166
Slave Synchronization ..... 149
Slow Rise Time ($\overline{MCLR}$ Tied to VDD) ..... 29
SPI Master Mode ..... 148
SPI Master Mode Example (CKE = 0) ..... 349
SPI Master Mode Example (CKE = 1) ..... 350
SPI Slave Mode (with CKE = 0) ..... 150
SPI Slave Mode (with CKE = 1) ..... 150
SPI Slave Mode Example (CKE = 0) ..... 351
SPI Slave Mode Example (CKE = 1) ..... 352
Stop Condition Receive or Transmit Mode ..... 176
Time-out Sequence on POR w/PLL Enabled ($\overline{MCLR}$ Tied to VDD) ..... 29
Time-out Sequence on Power-up ($\overline{MCLR}$ Not Tied to VDD)
Case 1 ..... 28
Case 2 ..... 28
Time-out Sequence on Power-up ($\overline{MCLR}$ Tied to VDD) ..... 28
Timer0 and Timer1 External Clock ..... 346
Transition Between Timer1 and OSC1 (HS with PLL) ..... 22
Transition Between Timer1 and OSC1 (HS, XT, LP) ..... 21
Transition Between Timer1 and OSC1 (RC, EC) ..... 22

chanhvinh.com




Transition from OSC1 to Timer1 Oscillator ..... 21
USART Asynchronous Reception ..... 192
USART Asynchronous Transmission ..... 190
USART Asynchronous Transmission (Back to Back) ..... 190
USART Synchronous Receive (Master/Slave) ..... 357
USART Synchronous Reception (Master Mode, SREN) ..... 195
USART Synchronous Transmission ..... 194
USART Synchronous Transmission (Master/Slave) ..... 357
USART Synchronous Transmission (Through TXEN) ..... 194
Wake-up from Sleep via Interrupt ..... 275
Timing Diagrams and Specifications ..... 343
A/D Conversion Requirements ..... 359
A/D Converter Characteristics ..... 358
Capture/Compare/PWM Requirements (CCP1 and ECCP1) ..... 347
CLKO and I/O Timing Requirements ..... 344
Example SPI Mode Requirements (Master Mode, CKE = 0) ..... 349
Example SPI Mode Requirements (Master Mode, CKE = 1) ..... 350
Example SPI Mode Requirements (Slave Mode, CKE = 0) ..... 351
Example SPI Slave Mode Requirements (CKE = 1) ..... 352
External Clock Timing Requirements ..... 343
I²C Bus Data Requirements (Slave Mode) ..... 354
I²C Bus Start/Stop Bits Requirements (Slave Mode) ..... 353
Master SSP I²C Bus Data Requirements ..... 356
Master SSP I²C Bus Start/Stop Bits Requirements ..... 355
Parallel Slave Port Requirements (PIC18F248 and PIC18F458) ..... 348
PLL Clock ..... 344
Reset, Watchdog Timer, Oscillator Start-up Timer, Power-up Timer, Brown-out Reset and Low-Voltage Detect Requirements ..... 345
Timer0 and Timer1 External Clock Requirements ..... 346
USART Synchronous Receive Requirements ..... 357
USART Synchronous Transmission Requirements ..... 357
TSTFSZ ..... 321
TXSTA Register
BRGH Bit ..... 185


## U


USART ..... 183
Asynchronous Mode ..... 189
Reception ..... 191
Setting Up 9-Bit Mode with Address Detect ..... 191
Transmission ..... 189
Asynchronous Reception ..... 192
Asynchronous Transmission
Associated Registers ..... 190
Baud Rate Generator (BRG) ..... 185
Associated Registers ..... 185
Baud Rate Error, Calculating ..... 185
Baud Rate Formula ..... 185
Baud Rates for Asynchronous Mode (BRGH = 0) ..... 187
Baud Rates for Asynchronous Mode (BRGH = 1) ..... 188
Baud Rates for Synchronous Mode ..... 186
High Baud Rate Select (BRGH Bit) ..... 185
Sampling ..... 185
Serial Port Enable (SPEN) Bit ..... 183
Synchronous Master Mode ..... 193
Reception ..... 195
Transmission ..... 193
Synchronous Master Reception
Associated Registers ..... 195
Synchronous Master Transmission
Associated Registers ..... 193
Synchronous Slave Mode ..... 196
Reception ..... 196
Transmission ..... 196
Synchronous Slave Reception ..... 197
Synchronous Slave Transmission
Associated Registers ..... 197


## V


Voltage Reference Specifications ..... 340


## W


Wake-up from Sleep ..... 265, 274
Using Interrupts ..... 274
Watchdog Timer (WDT) ..... 265, 272
Associated Registers ..... 273
Control Register ..... 272
Postscaler ..... 273
Programming Considerations ..... 272
RC Oscillator ..... 272
Time-out Period ..... 272
WCOL ..... 171, 172, 173, 176
WCOL Status Flag ..... 171, 172, 173, 176
WDT. *See* Watchdog Timer.
WWW, On-Line Support ..... 5


## X


XORLW ..... 321
XORWF ..... 322



© 2006 Microchip Technology Inc.

chanhvinh.com



## THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- Development Systems Information Line

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the web site at: http://support.microchip.com**

chanhvinh.com



## READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager 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: PIC18FXX8 Literature Number: DS41159E

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?

 © 2006 Microchip Technology Inc.

chanhvinh.com



## PIC18FXX8 PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| PART NO. | X | /XX | XXX |
|---|---|---|---|
| Device | Temperature Range | Package | Pattern |

| | | |
|---|---|---|
| Device | PIC18F248/258[1], PIC18F448/458[1], PIC18F248/258T[2], PIC18F448/458T[2]; VDD range 4.2V to 5.5V<br>PIC18LF248/258[1], PIC18LF448/458[1], PIC18LF248/258T[2], PIC18LF448/458T[2]; VDD range 2.0V to 5.5V | |
| Temperature Range | I<br>E | = -40°C to +85°C (Industrial)<br>= -40°C to +125°C (Extended) |
| Package | PT<br>L<br>SO<br>SP<br>P | = TQFP (Thin Quad Flatpack)<br>= PLCC<br>= SOIC<br>= Skinny Plastic DIP<br>= PDIP |
| Pattern | QTP, SQTP, Code or Special Requirements (blank otherwise) | |

**Examples:**

a) PIC18LF258-I/L 301 = Industrial temp., PLCC package, Extended VDD limits, QTP pattern #301.

b) PIC18LF458-I/PT = Industrial temp., TQFP package, Extended VDD limits.

c) PIC18F258-E/L = Extended temp., PLCC package, normal VDD limits.

**Note 1:** F = Standard Voltage Range
LF = Wide Voltage Range
**2:** T = in tape and reel PLCC and TQFP packages only.

chanhvinh.com

# WORLDWIDE SALES AND SERVICE

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

**Atlanta**
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

**Kokomo**
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**Santa Clara**
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

**Toronto**
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

## ASIA/PACIFIC

**Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Habour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Fuzhou**
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

**China - Hong Kong SAR**
Tel: 852-2401-1200
Fax: 852-2401-3431

**China - Qingdao**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**China - Shanghai**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**China - Shenyang**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**China - Shenzhen**
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

**China - Shunde**
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xian**
Tel: 86-29-8833-7250
Fax: 86-29-8833-7256

## ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

**Japan - Yokohama**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

**Korea - Gumi**
Tel: 82-54-473-4301
Fax: 82-54-473-4302

**Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

**Malaysia - Penang**
Tel: 60-4-646-8870
Fax: 60-4-646-5086

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-572-9526
Fax: 886-3-572-6459

**Taiwan - Kaohsiung**
Tel: 886-7-536-4818
Fax: 886-7-536-4803

**Taiwan - Taipei**
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

**Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## EUROPE

**Austria - Wels**
Tel: 43-7242-2244-3910
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

08/29/06


© 2006 Microchip Technology Inc.