



# **PIC18F1230/1330**

## **Data Sheet**

High-Performance Microcontrollers  
with 10-bit A/D and nanoWatt Technology

---

**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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL 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, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICKit, PICTail, PIC<sup>32</sup> logo, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, 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.

© 2009, 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 design centers in California and India. The Company's quality system processes and procedures are for its PIC<sup>®</sup> MCUs and dsPIC<sup>®</sup> DSCs, KEELOQ<sup>®</sup> 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.*



## 18/20/28-Pin Enhanced Flash Microcontrollers with nanoWatt Technology, High-Performance PWM and A/D

### Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Ultra Low 50 nA Input Leakage
- Run mode currents down to 15  $\mu$ A, typical
- Idle mode currents down to 3.7  $\mu$ A, typical
- Sleep mode current down to 100 nA, typical
- Timer1 Oscillator: 1.8  $\mu$ A, typical; 32 kHz; 2V
- Watchdog Timer (WDT): 1.4  $\mu$ A, typical; 2V
- Two-Speed Oscillator Start-up

### 14-Bit Power Control PWM Module:

- Up to 6 PWM Channel Outputs
  - Complementary or independent outputs
- Edge or Center-Aligned Operation
- Flexible Dead-Band Generator
- Hardware Fault Protection Input
- Simultaneous Update of Duty Cycle and Period:
  - Flexible Special Event Trigger output

### Flexible Oscillator Structure:

- Four Crystal modes, up to 40 MHz
- 4x Phase Lock Loop (PLL) – Available for Crystal and Internal Oscillators
- Two External RC modes, up to 4 MHz
  - Fast wake-up from Sleep and Idle, 1  $\mu$ s, typical
- Two External Clock modes, up to 40 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies from 31 kHz to 8 MHz
  - Provides a complete range of clock speeds from 31 kHz to 32 MHz when used with PLL
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops

### Peripheral Highlights:

- High-Current Sink/Source 25 mA/25 mA
- Up to 4 Programmable External Interrupts
- Four Input Change Interrupts
- Enhanced Addressable USART module:
  - Supports RS-485, RS-232 and LIN/J2602
  - RS-232 operation using internal oscillator block (no external crystal required)
  - Auto-wake-up on Start bit
  - Auto-Baud Detect
- 10-Bit, up to 4-Channel Analog-to-Digital Converter module (A/D):
  - Auto-acquisition capability
  - Conversion available during Sleep
- Up to 3 Analog Comparators
- Programmable Reference Voltage for Comparators
- Programmable, 15-Level Low-Voltage Detection (LVD) module:
  - Supports interrupt on Low-Voltage Detection

### Special Microcontroller Features:

- C Compiler Optimized Architecture with Optional Extended Instruction Set
- Flash Memory Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Programmable Code Protection
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins
- Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit ADC Channel	EUSART	Analog Comparator	14-Bit PWM (ch)	Timers 16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)						
PIC18F1230	4096	2048	256	128	16	4	Yes	3	6	2
PIC18F1330	8192	4096	256	128	16	4	Yes	3	6	2

# PIC18F1230/1330

## Pin Diagrams

### 18-Pin PDIP, SOIC



### 20-Pin SSOP

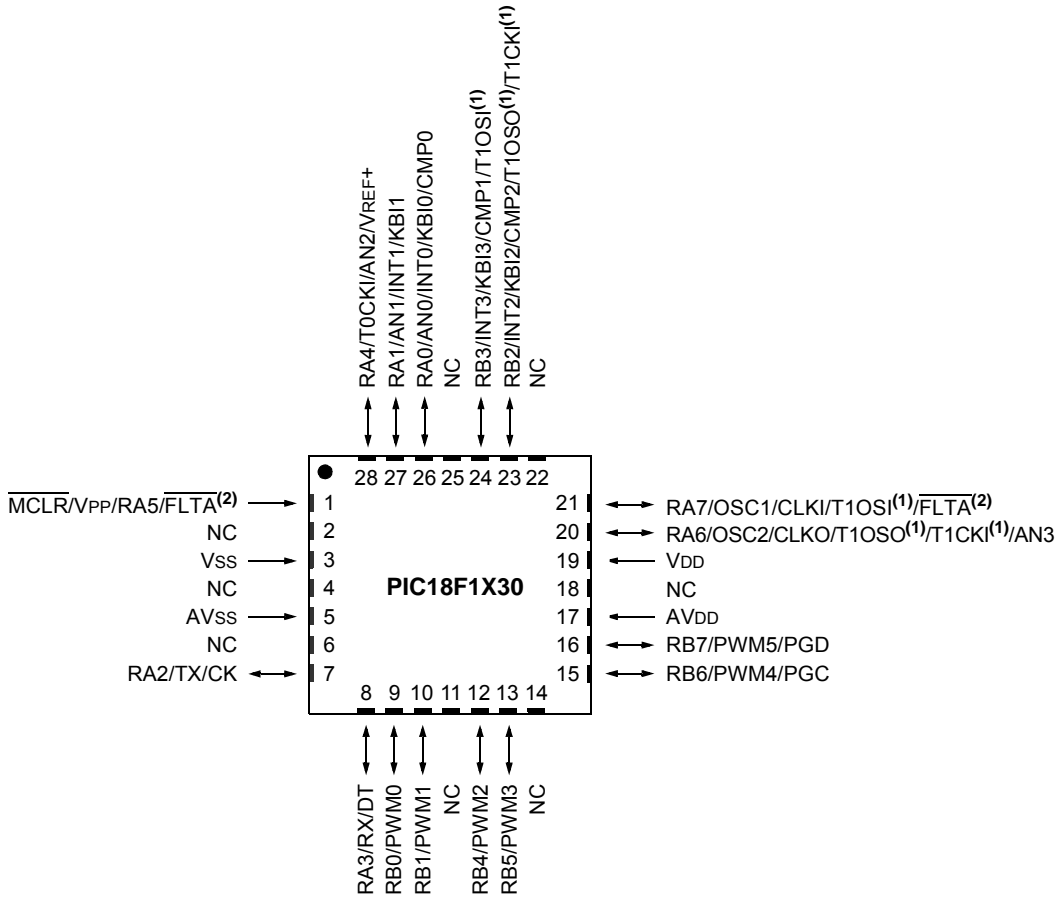


**Note 1:** Placement of T1OSI and T1OSO/T1CKI depends on the value of Configuration bit, T1OSCMX, of CONFIG3H.

**Note 2:** Placement of FLTA depends on the value of Configuration bit, FLTAMX, of CONFIG3H.

## Pin Diagrams (Continued)

### 28-Pin QFN<sup>(3)</sup>



- Note 1:** Placement of T1OSI and T1OSO/T1CKI depends on the value of Configuration bit, T1OSCMX, of CONFIG3H.
- Note 2:** Placement of FLTA depends on the value of Configuration bit, FLTAMX, of CONFIG3H.
- Note 3:** It is recommended that the user connect the center metal pad for this device package to the ground.

# PIC18F1230/1330

---

## Table of Contents

1.0	Device Overview .....	9
2.0	Guidelines for Getting Started with PIC18F Microcontrollers .....	17
3.0	Oscillator Configurations .....	21
4.0	Power-Managed Modes .....	31
5.0	Reset .....	39
6.0	Memory Organization .....	51
7.0	Flash Program Memory .....	71
8.0	Data EEPROM Memory .....	81
9.0	8 x 8 Hardware Multiplier .....	85
10.0	I/O Ports .....	87
11.0	Interrupts .....	93
12.0	Timer0 Module .....	107
13.0	Timer1 Module .....	111
14.0	Power Control PWM Module .....	117
15.0	Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) .....	147
16.0	10-Bit Analog-to-Digital Converter (A/D) Module .....	169
17.0	Comparator Module .....	179
18.0	Comparator Voltage Reference Module .....	183
19.0	Low-Voltage Detect (LVD) .....	187
20.0	Special Features of the CPU191 .....	
21.0	Development Support .....	211
22.0	Instruction Set Summary .....	215
23.0	Electrical Characteristics .....	265
24.0	Packaging Information .....	295
	Appendix A: Revision History .....	303
	Appendix B: Device Differences .....	304
	Appendix C: Conversion Considerations .....	305
	Appendix D: Migration from Baseline to Enhanced Devices .....	305
	Appendix E: Migration from Mid-Range TO Enhanced Devices .....	306
	Appendix F: Migration from High-End to Enhanced Devices .....	306
	Index .....	307

## 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](mailto: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](http://www.microchip.com) to receive the most current information on all of our products.

# PIC18F1230/1330

---

NOTES:



## 1.0 DEVICE OVERVIEW

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

- PIC18F1230
- PIC18F1330
- PIC18LF1230
- PIC18LF1330

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance Enhanced Flash program memory. On top of these features, the PIC18F1230/1330 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power control and motor control applications.

Peripheral highlights include:

- 14-bit resolution Power Control PWM module (PCPWM) with programmable dead-time insertion

The PCPWM can generate up to six complementary PWM outputs with dead-band time insertion. Overdrive current is detected by off-chip analog comparators or the digital Fault input ( $\overline{FLTA}$ ).

PIC18F1230/1330 devices also feature Flash program memory and an internal RC oscillator.

### 1.1 New Core Features

#### 1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F1230/1330 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 23.0 "Electrical Characteristics"** for values.

#### 1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F1230/1330 family offer ten different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes, using crystals or ceramic resonators.
- Two External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O).
- Two External RC Oscillator modes with the same pin options as the External Clock modes.
- An internal oscillator block which provides an 8 MHz clock and an INTRC source (approximately 31 kHz), as well as a range of six user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of eight clock frequencies. This option frees the two oscillator pins for use as additional general purpose I/Os.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the High-Speed Crystal and Internal Oscillator modes, which allows clock speeds of up to 40 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 32 MHz, all without using an external crystal or clock circuit.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

# PIC18F1230/1330

---

## 1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-Programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F1230/1330 family introduces an optional extension to the PIC18 instruction set, which adds eight new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Power Control PWM Module:** This module provides up to six modulated outputs for controlling half-bridge and full-bridge drivers. Other features include auto-shutdown on Fault detection and auto-restart to reactivate outputs once the condition has cleared.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN/J2602 bus protocol. Other enhancements include automatic Baud Rate Detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reducing code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 23.0 “Electrical Characteristics”** for time-out periods.

## 1.3 Details on Individual Family Members

Devices in the PIC18F1230/1330 family are available in 18-pin, 20-pin and 28-pin packages.

The devices are differentiated from each other in one way:

1. Flash program memory (4 Kbytes for PIC18F1230, 8 Kbytes for PIC18F1330).

All other features for devices in this family are identical. These are summarized in Table 1-1.

A block diagram of the PIC18F1220/1320 device architecture is provided in Figure 1-1. The pinouts for this device family are listed in Table 1-2.

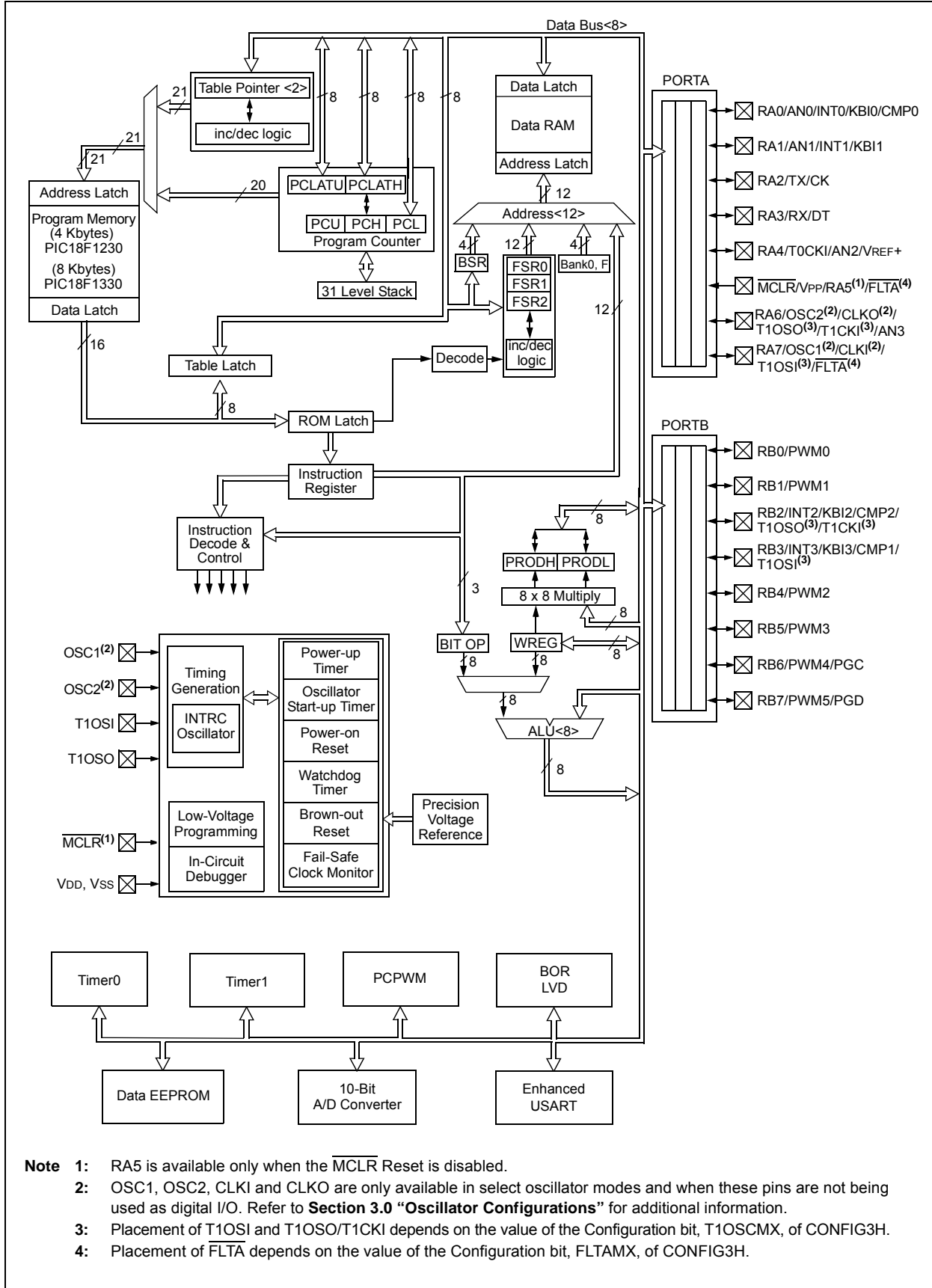
Like all Microchip PIC18 devices, members of the PIC18F1230/1330 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an “F” in the part number (such as PIC18F1330), accommodate an operating V<sub>DD</sub> range of 4.2V to 5.5V. Low-voltage parts, designated by “LF” (such as PIC18LF1330), function over an extended V<sub>DD</sub> range of 2.0V to 5.5V.

**TABLE 1-1: DEVICE FEATURES**

Features	PIC18F1230	PIC18F1330
Operating Frequency	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	4096	8192
Program Memory (Instructions)	2048	4096
Data Memory (Bytes)	256	256
Data EEPROM Memory (Bytes)	128	128
Interrupt Sources	17	17
I/O Ports	Ports A, B	Ports A, B
Timers	2	2
Power Control PWM Module	6 Channels	6 Channels
Serial Communications	Enhanced USART	Enhanced USART
10-Bit Analog-to-Digital Module	4 Input Channels	4 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes
Programmable Brown-out Reset	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	18-Pin PDIP 18-Pin SOIC 20-Pin SSOP 28-Pin QFN	18-Pin PDIP 18-Pin SOIC 20-Pin SSOP 28-Pin QFN

# PIC18F1230/1330

**FIGURE 1-1: PIC18F1230/1330 (18-PIN) BLOCK DIAGRAM**













## 2.0 GUIDELINES FOR GETTING STARTED WITH PIC18F MICROCONTROLLERS

### 2.1 Basic Connection Requirements

Getting started with the PIC18F1230/1330 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins (see **Section 2.2 “Power Supply Pins”**)
- All AVDD and AVSS pins, regardless of whether or not the analog device features are used (see **Section 2.2 “Power Supply Pins”**)
- MCLR pin (see **Section 2.3 “Master Clear (MCLR) Pin”**)

These pins must also be connected if they are being used in the end application:

- PGC/PGD pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see **Section 2.4 “ICSP Pins”**)
- OSCI and OSCO pins when an external oscillator source is used (see **Section 2.5 “External Oscillator Pins”**)

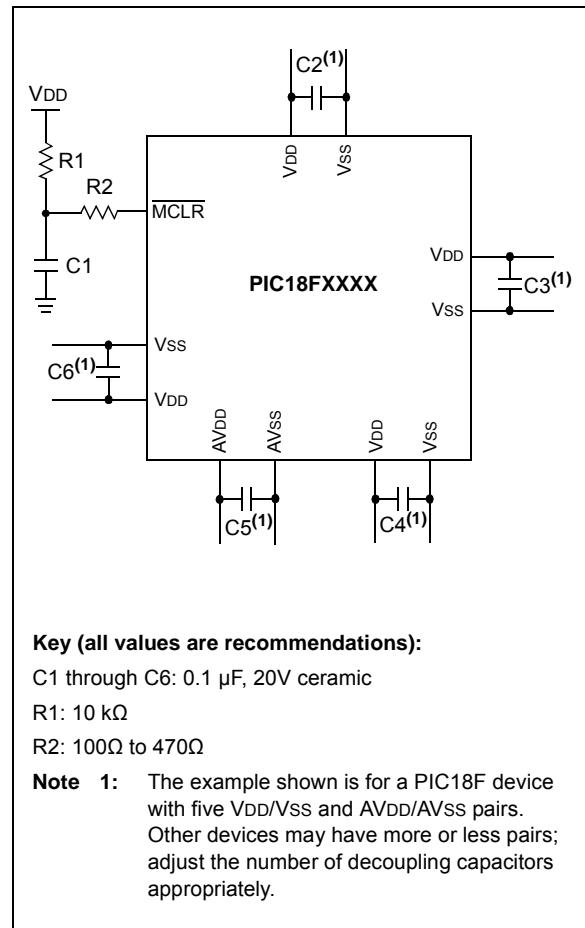
Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

**Note:** The AVDD and AVSS pins must always be connected, regardless of whether any of the analog modules are being used.

The minimum mandatory connections are shown in Figure 2-1.

**FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS**



# PIC18F1230/1330

## 2.2 Power Supply Pins

### 2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins, such as VDD, VSS, AVDD and AVSS, is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1  $\mu\text{F}$  (100 nF), 10-20V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01  $\mu\text{F}$  to 0.001  $\mu\text{F}$ . Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1  $\mu\text{F}$  in parallel with 0.001  $\mu\text{F}$ ).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

### 2.2.2 TANK CAPACITORS

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7  $\mu\text{F}$  to 47  $\mu\text{F}$ .

### 2.2.3 CONSIDERATIONS WHEN USING BOR

When the Brown-out Reset (BOR) feature is enabled, a sudden change in VDD may result in a spontaneous BOR event. This can happen when the microcontroller is operating under normal operating conditions, regardless of what the BOR set point has been programmed to, and even if VDD does not approach the set point. The precipitating factor in these BOR events is a rise or fall in VDD with a slew rate faster than 0.15V/ $\mu\text{s}$ .

An application that incorporates adequate decoupling between the power supplies will not experience such rapid voltage changes. Additionally, the use of an electrolytic tank capacitor across VDD and VSS, as described above, will be helpful in preventing high slew rate transitions.

If the application has components that turn on or off, and share the same VDD circuit as the microcontroller, the BOR can be disabled in software by using the SBOREN bit before switching the component. Afterwards, allow a small delay before re-enabling the BOR. By doing this, it is ensured that the BOR is disabled during the interval that might cause high slew rate changes of VDD.

**Note:** Not all devices incorporate software BOR control. See **Section 5.0 “Reset”** for device-specific information.

## 2.3 Master Clear ( $\overline{\text{MCLR}}$ ) Pin

The  $\overline{\text{MCLR}}$  pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to  $V_{DD}$  may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 2-1. Other circuit designs may be implemented, depending on the application's requirements.

During programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the  $\overline{\text{MCLR}}$  pin. Consequently, specific voltage levels ( $V_{IH}$  and  $V_{IL}$ ) and fast signal transitions must not be adversely affected. Therefore, specific values of  $R1$  and  $C1$  will need to be adjusted based on the application and PCB requirements. For example, it is recommended that the capacitor,  $C1$ , be isolated from the  $\overline{\text{MCLR}}$  pin during programming and debugging operations by using a jumper (Figure 2-2). The jumper is replaced for normal run-time operations.

Any components associated with the  $\overline{\text{MCLR}}$  pin should be placed within 0.25 inch (6 mm) of the pin.

## 2.4 ICSP Pins

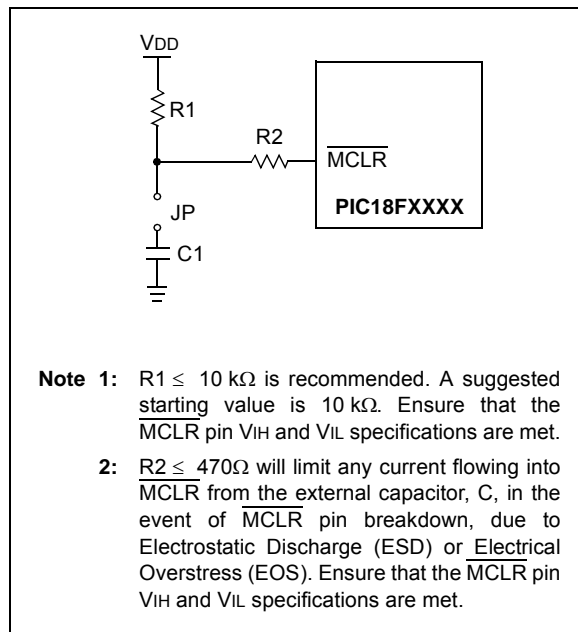
The PGC and PGD pins are used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of ohms, not to exceed 100Ω.

Pull-up resistors, series diodes, and capacitors on the PGC and PGD pins are not recommended as they will interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they should be removed from the circuit during programming and debugging. Alternatively, refer to the AC/DC characteristics and timing requirements information in the respective device Flash programming specification for information on capacitive loading limits and pin input voltage high ( $V_{IH}$ ) and input low ( $V_{IL}$ ) requirements.

For device emulation, ensure that the “Communication Channel Select” (i.e., PGCx/PGDx pins) programmed into the device matches the physical connections for the ICSP to the Microchip debugger/emulator tool.

For more information on available Microchip development tools connection requirements, refer to **Section 21.0 “Development Support”**.

**FIGURE 2-2: EXAMPLE OF  $\overline{\text{MCLR}}$  PIN CONNECTIONS**



## 2.5 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 3.0 “Oscillator Configurations”** for details).

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

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

Layout suggestions are shown in Figure 2-4. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

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

For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site ([www.microchip.com](http://www.microchip.com)):

- AN826, “Crystal Oscillator Basics and Crystal Selection for rPIC™ and PICmicro® Devices”
- AN849, “Basic PICmicro® Oscillator Design”
- AN943, “Practical PICmicro® Oscillator Analysis and Design”
- AN949, “Making Your Oscillator Work”

## 2.6 Unused I/Os

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

**FIGURE 2-3: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT**



## 3.0 OSCILLATOR CONFIGURATIONS

### 3.1 Oscillator Types

PIC18F1230/1330 devices can be operated in ten different oscillator modes. The user can program the Configuration bits, FOSC3:FOSC0, in Configuration Register 1H to select one of these ten modes:

1. LP Low-Power Crystal
2. XT Crystal/Resonator
3. HS High-Speed Crystal/Resonator
4. HSPLL High-Speed Crystal/Resonator with PLL enabled
5. RC External Resistor/Capacitor with Fosc/4 output on RA6
6. RCIO External Resistor/Capacitor with I/O on RA6
7. INTIO1 Internal Oscillator with Fosc/4 output on RA6 and I/O on RA7
8. INTIO2 Internal Oscillator with I/O on RA6 and RA7
9. EC External Clock with Fosc/4 output
10. ECIO External Clock with I/O on RA6

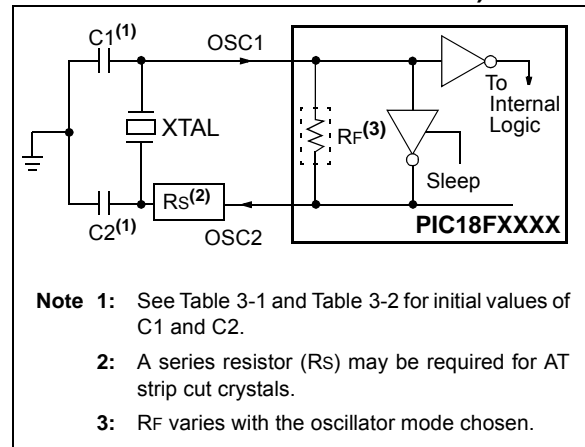
### 3.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 3-1 shows the pin connections.

The oscillator design requires the use of a parallel resonant crystal.

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

**FIGURE 3-1: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)**



**TABLE 3-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS**

Typical Capacitor Values Used:			
Mode	Freq	OSC1	OSC2
XT	3.58 MHz	15 pF	15 pF
	4.19 MHz	15 pF	15 pF
	4 MHz	30 pF	30 pF
	4 MHz	50 pF	50 pF

**Capacitor values are for design guidance only.**  
 Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.  
 See the notes following Table 3-2 for additional information.

# PIC18F1230/1330

**TABLE 3-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR**

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
LP	32 kHz	30 pF	30 pF
XT	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	10 MHz	15 pF	15 pF
	20 MHz	15 pF	15 pF
	25 MHz	15 pF	15 pF

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

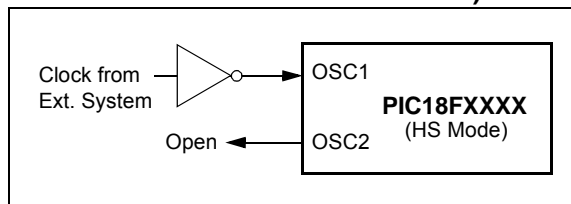
Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following this table for additional information.

- Note 1:** Higher capacitance increases the stability of the oscillator but also increases the start-up time.
- When operating below 3V VDD, or when using certain ceramic resonators at any voltage, it may be necessary to use the HS mode or switch to a crystal oscillator.
  - Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
  - Rs may be required to avoid overdriving crystals with low drive level specification.
  - Always verify oscillator performance over the VDD and temperature range that is expected for the application.

An external clock source may also be connected to the OSC1 pin in the HS mode, as shown in Figure 3-2.

**FIGURE 3-2: EXTERNAL CLOCK INPUT OPERATION (HS OSCILLATOR CONFIGURATION)**

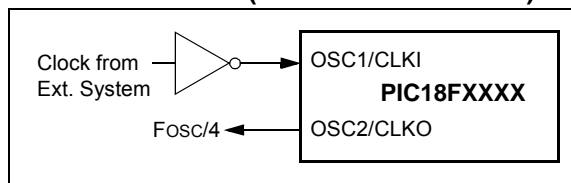


### 3.3 External Clock Input

The EC and ECIO Oscillator modes require an external clock source to be connected to the OSC1 pin. There is no oscillator start-up time required after a Power-on Reset or after an exit from Sleep mode.

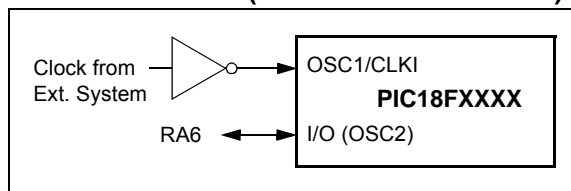
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-3 shows the pin connections for the EC Oscillator mode.

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



The ECIO Oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6). Figure 3-4 shows the pin connections for the ECIO Oscillator mode.

**FIGURE 3-4: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)**



## 3.4 RC Oscillator

For timing insensitive applications, the “RC” and “RCIO” device options offer additional cost savings. The actual oscillator frequency is a function of several factors:

- supply voltage
- values of the external resistor (R<sub>EXT</sub>) and capacitor (C<sub>EXT</sub>)
- operating temperature

Given the same device, operating voltage and temperature and component values, there will also be unit-to-unit frequency variations. These are due to factors such as:

- normal manufacturing variation
- difference in lead frame capacitance between package types (especially for low C<sub>EXT</sub> values)
- variations within the tolerance of limits of R<sub>EXT</sub> and C<sub>EXT</sub>

In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-5 shows how the R/C combination is connected.

**FIGURE 3-5: RC OSCILLATOR MODE**



The RCIO Oscillator mode (Figure 3-6) 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).

**FIGURE 3-6: RCIO OSCILLATOR MODE**



## 3.5 PLL Frequency Multiplier

A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower frequency oscillator circuit or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals or users who require higher clock speeds from an internal oscillator.

### 3.5.1 HSPLL OSCILLATOR MODE

The HSPLL mode makes use of the HS mode oscillator for frequencies up to 10 MHz. A PLL then multiplies the oscillator output frequency by 4 to produce an internal clock frequency up to 40 MHz. The PLEN bit is not available in this oscillator mode.

The PLL is only available to the crystal oscillator when the FOSC3:FOSC0 Configuration bits are programmed for HSPLL mode (= 0110).

**FIGURE 3-7: PLL BLOCK DIAGRAM (HS MODE)**



### 3.5.2 PLL AND INTOSC

The PLL is also available to the internal oscillator block in selected oscillator modes. In this configuration, the PLL is enabled in software and generates a clock output of up to 32 MHz. The operation of INTOSC with the PLL is described in **Section 3.6.4 “PLL in INTOSC Modes”**.

# PIC18F1230/1330

## 3.6 Internal Oscillator Block

The PIC18F1230/1330 devices include an internal oscillator block which generates two different clock signals; either can be used as the microcontroller's clock source. This may eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins.

The main output (INTOSC) is an 8 MHz clock source, which can be used to directly drive the device clock. It also drives a postscaler, which can provide a range of clock frequencies from 31 kHz to 4 MHz. The INTOSC output is enabled when a clock frequency from 125 kHz to 8 MHz is selected.

The other clock source is the internal RC oscillator (INTRC), which provides a nominal 31 kHz output. INTRC is enabled if it is selected as the device clock source; it is also enabled automatically when any of the following are enabled:

- Power-up Timer
- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in greater detail in **Section 20.0 "Special Features of the CPU"**.

The clock source frequency (INTOSC direct, INTRC direct or INTOSC postscaler) is selected by configuring the IRCF bits of the OSCCON register (page 28).

### 3.6.1 INTIO MODES

Using the internal oscillator as the clock source eliminates the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct configurations are available:

- In INTIO1 mode, the OSC2 pin outputs  $F_{osc}/4$ , while OSC1 functions as RA7 for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6, both for digital input and output.

### 3.6.2 INTOSC OUTPUT FREQUENCY

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8.0 MHz.

The INTRC oscillator operates independently of the INTOSC source. Any changes in INTOSC across voltage and temperature are not necessarily reflected by changes in INTRC and vice versa.

### 3.6.3 OSCTUNE REGISTER

The internal oscillator's output has been calibrated at the factory but can be adjusted in the user's application. This is done by writing to the OSCTUNE register (Register 3-1). The tuning sensitivity is constant throughout the tuning range.

When the OSCTUNE register is modified, the INTOSC frequency will begin shifting to the new frequency. Code execution continues during this shift. There is no indication that the shift has occurred.

The OSCTUNE register also implements the INTSRC and PLEN bits, which control certain features of the internal oscillator block. The INTSRC bit allows users to select which internal oscillator provides the clock source when the 31 kHz frequency option is selected. This is covered in greater detail in **Section 3.7.1 "Oscillator Control Register"**.

The PLEN bit controls the operation of the frequency multiplier, PLL, in internal oscillator modes.

### 3.6.4 PLL IN INTOSC MODES

The 4x frequency multiplier can be used with the internal oscillator block to produce faster device clock speeds than are normally possible with an internal oscillator. When enabled, the PLL produces a clock speed of up to 32 MHz.

Unlike HSPLL mode, the PLL is controlled through software. The control bit, PLEN (OSCTUNE<6>), is used to enable or disable its operation. If PLL is enabled and a Two-Speed Start-up from wake is performed, execution is delayed until the PLL starts.

The PLL is available when the device is configured to use the internal oscillator block as its primary clock source (FOSC3:FOSC0 = 1001 or 1000). Additionally, the PLL will only function when the selected output frequency is either 4 MHz or 8 MHz (OSCCON<6:4> = 111 or 110). If both of these conditions are not met, the PLL is disabled.

The PLEN control bit is only functional in those internal oscillator modes where the PLL is available. In all other modes, it is forced to '0' and is effectively unavailable.

### 3.6.5 INTOSC FREQUENCY DRIFT

The factory calibrates the internal oscillator block output (INTOSC) for 8 MHz. However, this frequency may drift as VDD or temperature changes, which can affect the controller operation in a variety of ways. It is possible to adjust the INTOSC frequency by modifying the value in the OSCTUNE register. This has no effect on the INTRC clock source frequency.

Tuning the INTOSC source requires knowing when to make the adjustment, in which direction it should be made and in some cases, how large a change is needed. Two compensation techniques are discussed in **Section 3.6.5.1 "Compensating with the EUSART"** and **Section 3.6.5.2 "Compensating with the Timers"**, but other techniques may be used.





# PIC18F1230/1330

## 3.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F1230/1330 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate low-frequency clock source. PIC18F1230/1330 devices offer two alternate clock sources. When an alternate clock source is enabled, the various power-managed operating modes are available.

Essentially, there are three clock sources for these devices:

- Primary oscillators
- Secondary oscillators
- Internal oscillator block

The **primary oscillators** include the External Crystal and Resonator modes, the External RC modes, the External Clock modes and the internal oscillator block. The particular mode is defined by the FOSC3:FOSC0 Configuration bits. The details of these modes are covered earlier in this chapter.

The **secondary oscillators** are those external sources not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power-managed mode.

PIC18F1230/1330 devices offer the Timer1 oscillator as a secondary oscillator. This oscillator, in all power-managed modes, is often the time base for functions such as a real-time clock.

Most often, a 32.768 kHz watch crystal is connected between the T1OSO/T1CKI and T1OSI pins. Like the LP mode oscillator circuit, loading capacitors are also connected from each pin to ground. The Timer1 oscillator is discussed in greater detail in **Section 13.2 “Timer1 Oscillator”**.

In addition to being a primary clock source, the **internal oscillator block** is available as a power-managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor.

The clock sources for the PIC18F1230/1330 devices are shown in Figure 3-8. See **Section 20.0 “Special Features of the CPU”** for Configuration register details.

FIGURE 3-8: PIC18F1230/1330 CLOCK DIAGRAM



## 3.7.1 OSCILLATOR CONTROL REGISTER

The OSCCON register (Register 3-2) controls several aspects of the device clock's operation, both in full power operation and in power-managed modes.

The System Clock Select bits, SCS1:SCS0, select the clock source. The available clock sources are the primary clock (defined by the FOSC3:FOSC0 Configuration bits), the secondary clock (Timer1 oscillator) and the internal oscillator block. The clock source changes immediately after one or more of the bits is written to, following a brief clock transition interval. The SCS bits are cleared on all forms of Reset.

The Internal Oscillator Frequency Select bits (IRCF2:IRCF0) select the frequency output of the internal oscillator block to drive the device clock. The choices are the INTRC source, the INTOSC source (8 MHz) or one of the frequencies derived from the INTOSC postscaler (31.25 kHz to 4 MHz). If the internal oscillator block is supplying the device clock, changing the states of these bits will have an immediate change on the internal oscillator's output. On device Resets, the default output frequency of the internal oscillator block is set at 1 MHz.

When a nominal output frequency of 31 kHz is selected (IRCF2:IRCF0 = 000), users may choose which internal oscillator acts as the source. This is done with the INTSRC bit in the OSCTUNE register (OSCTUNE<7>). Setting this bit selects INTOSC as a 31.25 kHz clock source by enabling the divide-by-256 output of the INTOSC postscaler. Clearing INTSRC selects INTRC (nominally 31 kHz) as the clock source.

This option allows users to select the tunable and more precise INTOSC as a clock source, while maintaining power savings with a very low clock speed. Regardless of the setting of INTSRC, INTRC always remains the clock source for features such as the Watchdog Timer and the Fail-Safe Clock Monitor.

The OSTS, IOFS and T1RUN bits indicate which clock source is currently providing the device clock. The OSTS bit indicates that the Oscillator Start-up Timer has timed out and the primary clock is providing the device clock in primary clock modes. The IOFS bit indicates when the internal oscillator block has stabilized and is providing the device clock in RC Clock modes. The T1RUN bit (T1CON<6>) indicates when the Timer1 oscillator is providing the device clock in secondary clock modes. In power-managed modes, only one of these three bits will be set at any time. If none of these bits are set, the INTRC is providing the clock or the internal oscillator block has just started and is not yet stable.

The IDLEN bit determines if the device goes into Sleep mode or one of the Idle modes when the SLEEP instruction is executed.

The use of the flag and control bits in the OSCCON register is discussed in more detail in **Section 4.0 "Power-Managed Modes"**.

**Note 1:** The Timer1 oscillator must be enabled to select the secondary clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 Control register (T1CON<3>). If the Timer1 oscillator is not enabled, then any attempt to select a secondary clock source will be ignored.

**2:** It is recommended that the Timer1 oscillator be operating and stable before selecting the secondary clock source or a very long delay may occur while the Timer1 oscillator starts.

## 3.7.2 OSCILLATOR TRANSITIONS

PIC18F1230/1330 devices contain circuitry to prevent clock "glitches" when switching between clock sources. A short pause in the device clock occurs during the clock switch. The length of this pause is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Clock transitions are discussed in greater detail in **Section 4.1.2 "Entering Power-Managed Modes"**.

# PIC18F1230/1330

## REGISTER 3-2: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-1	R/W-0	R/W-0	R <sup>(1)</sup>	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
bit 7							bit 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

- bit 7                      **IDLEN:** Idle Enable bit  
1 = Device enters Idle mode on SLEEP instruction  
0 = Device enters Sleep mode on SLEEP instruction
- bit 6-4                      **IRCF2:IRCF0:** Internal Oscillator Frequency Select bits  
111 = 8 MHz (INTOSC drives clock directly)  
110 = 4 MHz  
101 = 2 MHz  
100 = 1 MHz<sup>(3)</sup>  
011 = 500 kHz  
010 = 250 kHz  
001 = 125 kHz  
000 = 31 kHz (from either INTOSC/256 or INTRC directly)<sup>(2)</sup>
- bit 3                      **OSTS:** Oscillator Start-up Time-out Status bit<sup>(1)</sup>  
1 = Oscillator Start-up Timer time-out has expired; primary oscillator is running  
0 = Oscillator Start-up Timer time-out is running; primary oscillator is not ready
- bit 2                      **IOFS:** INTOSC Frequency Stable bit  
1 = INTOSC frequency is stable  
0 = INTOSC frequency is not stable
- bit 1-0                      **SCS1:SCS0:** System Clock Select bits  
1x = Internal oscillator block  
01 = Secondary (Timer1) oscillator  
00 = Primary oscillator

- Note 1:** Reset state depends on state of the IESO Configuration bit.  
**Note 2:** Source selected by the INTSRC bit (OSCTUNE<7>), see text.  
**Note 3:** Default output frequency of INTOSC on Reset.

## 3.8 Effects of Power-Managed Modes on the Various Clock Sources

When PRI\_IDLE mode is selected, the designated primary oscillator continues to run without interruption. For all other power-managed modes, the oscillator using the OSC1 pin is disabled. The OSC1 pin (and OSC2 pin, if used by the oscillator) will stop oscillating.

In secondary clock modes (SEC\_RUN and SEC\_IDLE), the Timer1 oscillator is operating and providing the device clock. The Timer1 oscillator may also run in all power-managed modes if required to clock Timer1 or Timer3.

In internal oscillator modes (RC\_RUN and RC\_IDLE), the internal oscillator block provides the device clock source. The 31 kHz INTRC output can be used directly to provide the clock and may be enabled to support various special features, regardless of the power-managed mode (see **Section 20.2 “Watchdog Timer (WDT)”**, **Section 20.3 “Two-Speed Start-up”** and **Section 20.4 “Fail-Safe Clock Monitor”** for more information on WDT, Fail-Safe Clock Monitor and Two-Speed Start-up). The INTOSC output at 8 MHz may be used directly to clock the device or may be divided down by the postscaler. The INTOSC output is disabled if the clock is provided directly from the INTRC output.

If the Sleep mode is selected, all clock sources are stopped. Since all the transistor switching currents have been stopped, 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 INTRC is required to support WDT operation. The Timer1 oscillator may be operating to support a real-

time clock. Other features may be operating that do not require a device clock source (i.e., INTx pins and others). Peripherals that may add significant current consumption are listed in **Section 23.0 “Electrical Characteristics”**.

## 3.9 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 is stable under normal circumstances and the primary clock is operating and stable. For additional information on power-up delays, see **Section 5.5 “Device Reset Timers”**.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 23-10). It is enabled by clearing (= 0) the PWRTEN Configuration bit.

The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (LP, XT and HS modes). The OST does this by counting 1024 oscillator cycles before allowing the oscillator to clock the device.

When the HSPLL Oscillator mode is selected, the device is kept in Reset for an additional 2 ms, following the HS mode OST delay, so the PLL can lock to the incoming clock frequency.

There is a delay of interval TcSD (parameter 38, Table 23-10), following POR, while the controller becomes ready to execute instructions. This delay runs concurrently with any other delays. This may be the only delay that occurs when any of the EC, RC or INTIO modes are used as the primary clock source.

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

Oscillator Mode	OSC1 Pin	OSC2 Pin
RC, INTIO1	Floating, external resistor should pull high	At logic low (clock/4 output)
RCIO	Floating, external resistor should pull high	Configured as PORTA, bit 6
INTIO2	Configured as PORTA, bit 7	Configured as PORTA, bit 6
ECIO	Floating, pulled by external clock	Configured as PORTA, bit 6
EC	Floating, pulled by external clock	At logic low (clock/4 output)
LP, XT and HS	Feedback inverter disabled at quiescent voltage level	Feedback inverter disabled at quiescent voltage level

**Note:** See Table 5-2 in **Section 5.0 “Reset”** for time-outs due to Sleep and MCLR Reset.

# PIC18F1230/1330

---

NOTES:

## 4.0 POWER-MANAGED MODES

PIC18F1230/1330 devices offer a total of seven operating modes for more efficient power management. These modes provide a variety of options for selective power conservation in applications where resources may be limited (i.e., battery-powered devices).

There are three categories of power-managed modes:

- Run modes
- Idle modes
- Sleep mode

These categories define which portions of the device are clocked and sometimes, what speed. The Run and Idle modes may use any of the three available clock sources (primary, secondary or internal oscillator block); the Sleep mode does not use a clock source.

The power-managed modes include several power-saving features offered on previous PIC® devices. One is the clock switching feature, offered in other PIC18 devices, allowing the controller to use the Timer1 oscillator in place of the primary oscillator. Also included is the Sleep mode, offered by all PIC devices, where all device clocks are stopped.

### 4.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires two decisions: if the CPU is to be clocked or not and the selection of a clock source. The IDLEN bit (OSCCON<7>) controls CPU clocking, while the SCS1:SCS0 bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 4-1.

#### 4.1.1 CLOCK SOURCES

The SCS1:SCS0 bits allow the selection of one of three clock sources for power-managed modes. They are:

- the primary clock, as defined by the FOSC3:FOSC0 Configuration bits
- the secondary clock (the Timer1 oscillator)
- the internal oscillator block (for RC modes)

#### 4.1.2 ENTERING POWER-MANAGED MODES

Switching from one power-managed mode to another begins by loading the OSCCON register. The SCS1:SCS0 bits select the clock source and determine which Run or Idle mode is to be used. Changing these bits causes an immediate switch to the new clock source, assuming that it is running. The switch may also be subject to clock transition delays. These are discussed in **Section 4.1.3 “Clock Transitions and Status Indicators”** and subsequent sections.

Entry to the power-managed Idle or Sleep modes is triggered by the execution of a SLEEP instruction. The actual mode that results depends on the status of the IDLEN bit.

Depending on the current mode and the mode being switched to, a change to a power-managed mode does not always require setting all of these bits. Many transitions may be done by changing the oscillator select bits, or changing the IDLEN bit, prior to issuing a SLEEP instruction. If the IDLEN bit is already configured correctly, it may only be necessary to perform a SLEEP instruction to switch to the desired mode.

**TABLE 4-1: POWER-MANAGED MODES**

Mode	OSCCON Bits		Module Clocking		Available Clock and Oscillator Source
	IDLEN<7> <sup>(1)</sup>	SCS1:SCS0<1:0>	CPU	Peripherals	
Sleep	0	N/A	Off	Off	None – All clocks are disabled
PRI_RUN	N/A	00	Clocked	Clocked	Primary – LP, XT, HS, HSPLL, RC, EC and Internal Oscillator Block <sup>(2)</sup> . This is the normal full power execution mode.
SEC_RUN	N/A	01	Clocked	Clocked	Secondary – Timer1 Oscillator
RC_RUN	N/A	1x	Clocked	Clocked	Internal Oscillator Block <sup>(2)</sup>
PRI_IDLE	1	00	Off	Clocked	Primary – LP, XT, HS, HSPLL, RC, EC
SEC_IDLE	1	01	Off	Clocked	Secondary – Timer1 Oscillator
RC_IDLE	1	1x	Off	Clocked	Internal Oscillator Block <sup>(2)</sup>

**Note 1:** IDLEN reflects its value when the SLEEP instruction is executed.

**2:** Includes INTOSC and INTOSC postscaler, as well as the INTRC source.

# PIC18F1230/1330

## 4.1.3 CLOCK TRANSITIONS AND STATUS INDICATORS

The length of the transition between clock sources is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Three bits indicate the current clock source and its status. They are:

- OSTS (OSCCON<3>)
- IOFS (OSCCON<2>)
- T1RUN (T1CON<6>)

In general, only one of these bits will be set while in a given power-managed mode. When the OSTS bit is set, the primary clock is providing the device clock. When the IOFS bit is set, the INTOSC output is providing a stable 8 MHz clock source to a divider that actually drives the device clock. When the T1RUN bit is set, the Timer1 oscillator is providing the clock. If none of these bits are set, then either the INTRC clock source is clocking the device, or the INTOSC source is not yet stable.

If the internal oscillator block is configured as the primary clock source by the FOSC3:FOSC0 Configuration bits, then both the OSTS and IOFS bits may be set when in PRI\_RUN or PRI\_IDLE modes. This indicates that the primary clock (INTOSC output) is generating a stable 8 MHz output. Entering another power-managed RC mode at the same frequency would clear the OSTS bit.

- Note 1:** Caution should be used when modifying a single IRCF bit. If VDD is less than 3V, it is possible to select a higher clock speed than is supported by the low VDD. Improper device operation may result if the VDD/FOSC specifications are violated.
- 2:** Executing a SLEEP instruction does not necessarily place the device into Sleep mode. It acts as the trigger to place the controller into either the Sleep mode or one of the Idle modes, depending on the setting of the IDLEN bit.

## 4.1.4 MULTIPLE SLEEP COMMANDS

The power-managed mode that is invoked with the SLEEP instruction is determined by the setting of the IDLEN bit at the time the instruction is executed. If another SLEEP instruction is executed, the device will enter the power-managed mode specified by IDLEN at that time. If IDLEN has changed, the device will enter the new power-managed mode specified by the new setting.

## 4.2 Run Modes

In the Run modes, clocks to both the core and peripherals are active. The difference between these modes is the clock source.

### 4.2.1 PRI\_RUN MODE

The PRI\_RUN mode is the normal, full power execution mode of the microcontroller. This is also the default mode upon a device Reset unless Two-Speed Start-up is enabled (see **Section 20.3 “Two-Speed Start-up”** for details). In this mode, the OSTS bit is set. The IOFS bit may be set if the internal oscillator block is the primary clock source (see **Section 3.7.1 “Oscillator Control Register”**).

### 4.2.2 SEC\_RUN MODE

The SEC\_RUN mode is the compatible mode to the “clock switching” feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the Timer1 oscillator. This gives users the option of lower power consumption while still using a high accuracy clock source.

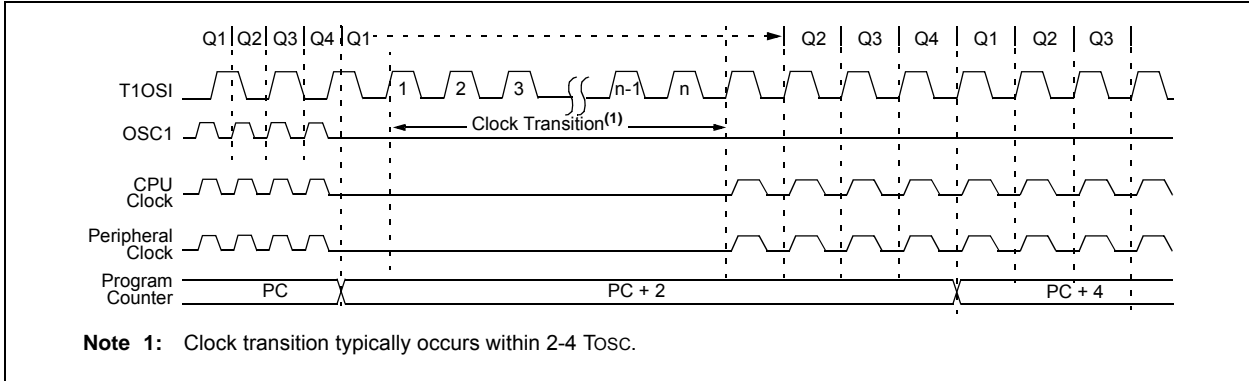
SEC\_RUN mode is entered by setting the SCS1:SCS0 bits to ‘01’. The device clock source is switched to the Timer1 oscillator (see Figure 4-1), the primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

**Note:** The Timer1 oscillator should already be running prior to entering SEC\_RUN mode. If the T1OSCFEN bit is not set when the SCS1:SCS0 bits are set to ‘01’, entry to SEC\_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, device clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result.

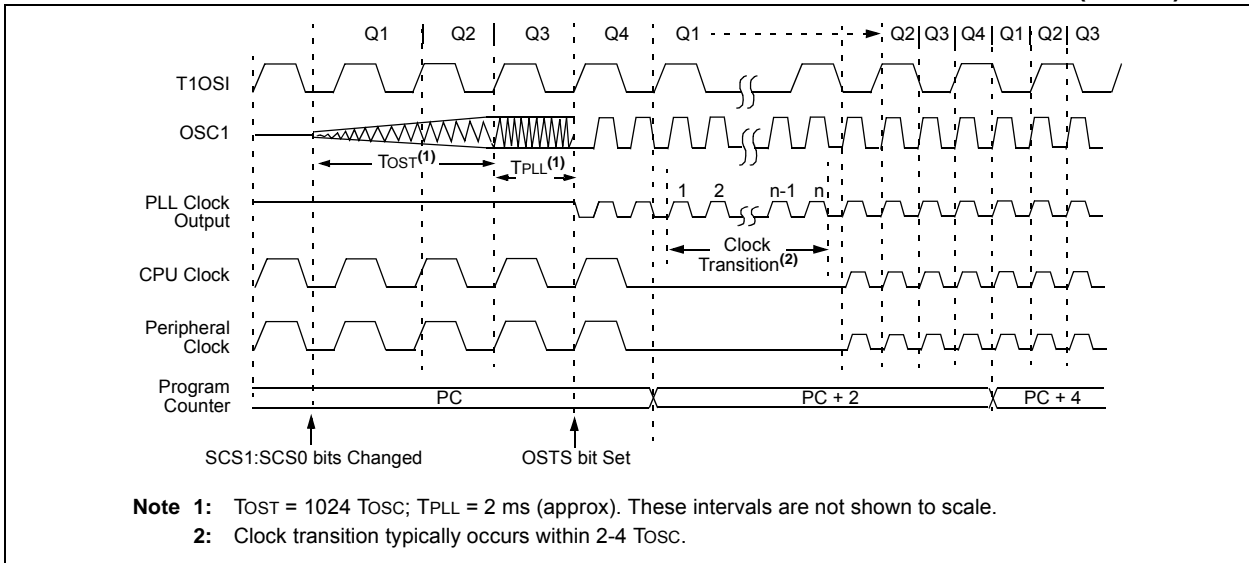
On transitions from SEC\_RUN to PRI\_RUN mode, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switch back to the primary clock occurs (see Figure 4-2). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the clock. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run.



**FIGURE 4-1: TRANSITION TIMING FOR ENTRY TO SEC\_RUN MODE**



**FIGURE 4-2: TRANSITION TIMING FROM SEC\_RUN MODE TO PRI\_RUN MODE (HSPLL)**



### 4.2.3 RC\_RUN MODE

In RC\_RUN mode, the CPU and peripherals are clocked from the internal oscillator block using the INTOSC multiplexer. In this mode, the primary clock is shut down. When using the INTRC source, this mode provides the best power conservation of all the Run modes, while still executing code. It works well for user applications which are not highly timing sensitive or do not require high-speed clocks at all times.

If the primary clock source is the internal oscillator block (either INTRC or INTOSC), there are no distinguishable differences between PRI\_RUN and RC\_RUN modes during execution. However, a clock switch delay will occur during entry to and exit from RC\_RUN mode. Therefore, if the primary clock source is the internal oscillator block, the use of RC\_RUN mode is not recommended.

This mode is entered by setting the SCS1 bit to '1'. Although it is ignored, it is recommended that the SCS0 bit also be cleared; this is to maintain software compatibility with future devices. When the clock source is switched to the INTOSC multiplexer (see Figure 4-3), the primary oscillator is shut down and the OST bit is cleared. The IRCF bits may be modified at any time to immediately change the clock speed.

**Note:** Caution should be used when modifying a single IRCF bit. If V<sub>DD</sub> is less than 3V, it is possible to select a higher clock speed than is supported by the low V<sub>DD</sub>. Improper device operation may result if the V<sub>DD</sub>/F<sub>OSC</sub> specifications are violated.

# PIC18F1230/1330

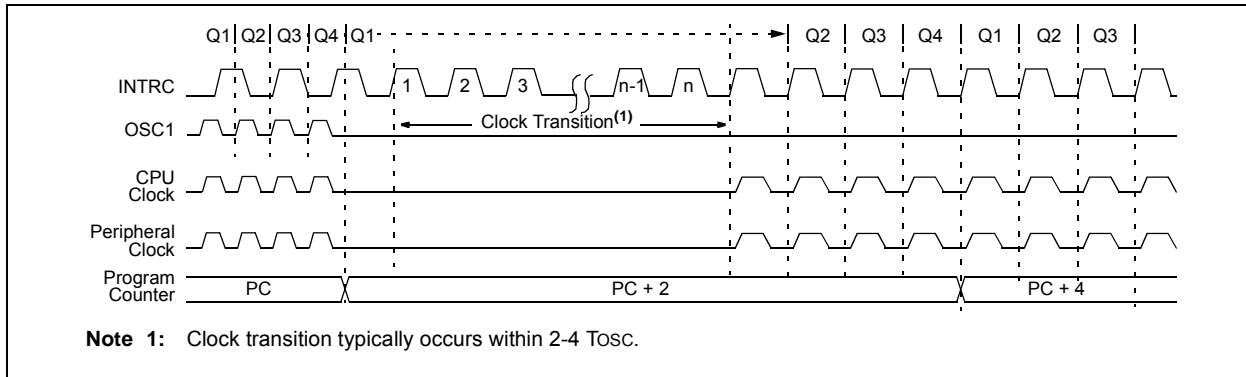
If the IRCF bits and the INTSRC bit are all clear, the INTOSC output is not enabled and the IOFS bit will remain clear; there will be no indication of the current clock source. The INTRC source is providing the device clocks.

If the IRCF bits are changed from all clear (thus, enabling the INTOSC output), or if INTSRC is set, the IOFS bit becomes set after the INTOSC output becomes stable. Clocks to the device continue while the INTOSC source stabilizes after an interval of TIOBST.

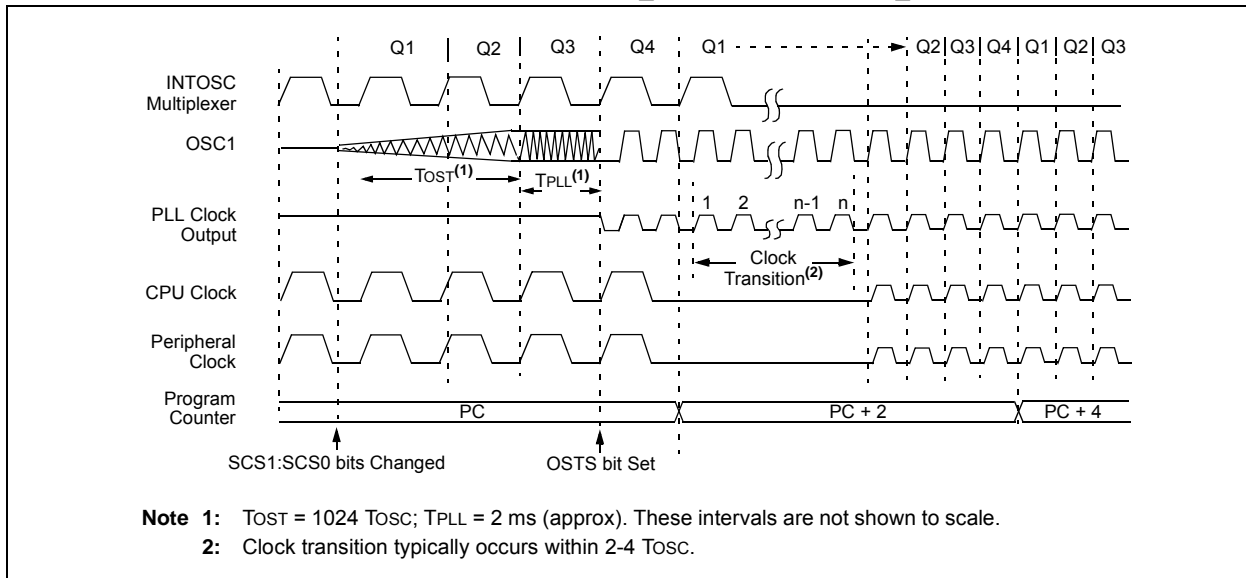
If the IRCF bits were previously at a non-zero value, or if INTSRC was set before setting SCS1 and the INTOSC source was already stable, the IOFS bit will remain set.

On transitions from RC\_RUN mode to PRI\_RUN mode, the device continues to be clocked from the INTOSC multiplexer while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 4-4). When the clock switch is complete, the IOFS bit is cleared, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCS bits are not affected by the switch. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

**FIGURE 4-3: TRANSITION TIMING TO RC\_RUN MODE**



**FIGURE 4-4: TRANSITION TIMING FROM RC\_RUN MODE TO PRI\_RUN MODE**



## 4.3 Sleep Mode

The power-managed Sleep mode in the PIC18F1230/1330 devices is identical to the legacy Sleep mode offered in all other PIC devices. It is entered by clearing the IDLEN bit (the default state on device Reset) and executing the `SLEEP` instruction. This shuts down the selected oscillator (Figure 4-5). All clock source status bits are cleared.

Entering the Sleep mode from any other mode does not require a clock switch. This is because no clocks are needed once the controller has entered Sleep. If the WDT is selected, the INTRC source will continue to operate. If the Timer1 oscillator is enabled, it will also continue to run.

When a wake event occurs in Sleep mode (by interrupt, Reset or WDT time-out), the device will not be clocked until the clock source selected by the SCS1:SCS0 bits becomes ready (see Figure 4-6), or it will be clocked from the internal oscillator block if either the Two-Speed Start-up or the Fail-Safe Clock Monitor are enabled (see **Section 20.0 “Special Features of the CPU”**). In either case, the OSTS bit is set when the primary clock is providing the device clocks. The IDLEN and SCS bits are not affected by the wake-up.

## 4.4 Idle Modes

The Idle modes allow the controller’s CPU to be selectively shut down while the peripherals continue to operate. Selecting a particular Idle mode allows users to further manage power consumption.

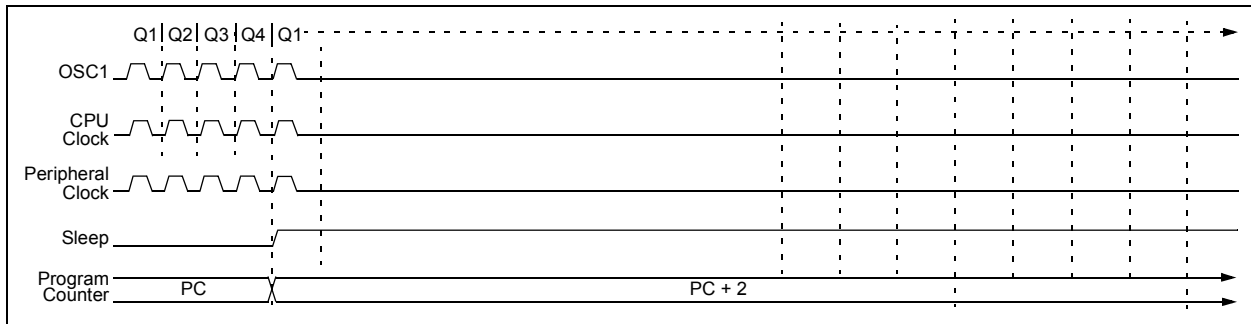
If the IDLEN bit is set to a ‘1’ when a `SLEEP` instruction is executed, the peripherals will be clocked from the clock source selected using the SCS1:SCS0 bits; however, the CPU will not be clocked. The clock source status bits are not affected. Setting IDLEN and executing a `SLEEP` instruction provides a quick method of switching from a given Run mode to its corresponding Idle mode.

If the WDT is selected, the INTRC source will continue to operate. If the Timer1 oscillator is enabled, it will also continue to run.

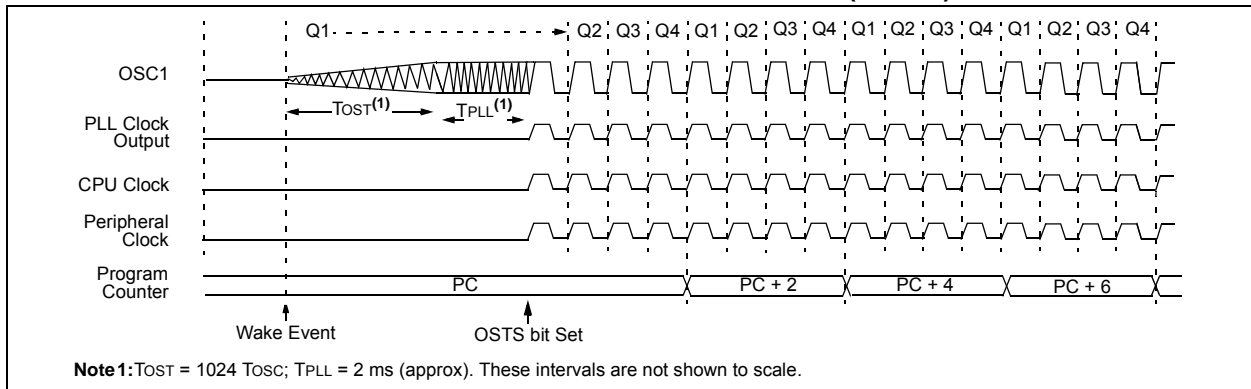
Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset. When a wake event occurs, CPU execution is delayed by an interval of  $T_{csd}$  (parameter 38, Table 23-10) while it becomes ready to execute code. When the CPU begins executing code, it resumes with the same clock source for the current Idle mode. For example, when waking from RC\_IDLE mode, the internal oscillator block will clock the CPU and peripherals (in other words, RC\_RUN mode). The IDLEN and SCS bits are not affected by the wake-up.

While in any Idle mode or the Sleep mode, a WDT time-out will result in a WDT wake-up to the Run mode currently specified by the SCS1:SCS0 bits.

**FIGURE 4-5: TRANSITION TIMING FOR ENTRY TO SLEEP MODE**



**FIGURE 4-6: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)**



# PIC18F1230/1330

## 4.4.1 PRI\_IDLE MODE

This mode is unique among the three low-power Idle modes, in that it does not disable the primary device clock. For timing sensitive applications, this allows for the fastest resumption of device operation with its more accurate primary clock source, since the clock source does not have to “warm-up” or transition from another oscillator.

PRI\_IDLE mode is entered from PRI\_RUN mode by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then clear the SCS bits and execute SLEEP. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC3:FOSC0 Configuration bits. The OSTS bit remains set (see Figure 4-7).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval T<sub>CSD</sub> is required between the wake event and when code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 4-8).

## 4.4.2 SEC\_IDLE MODE

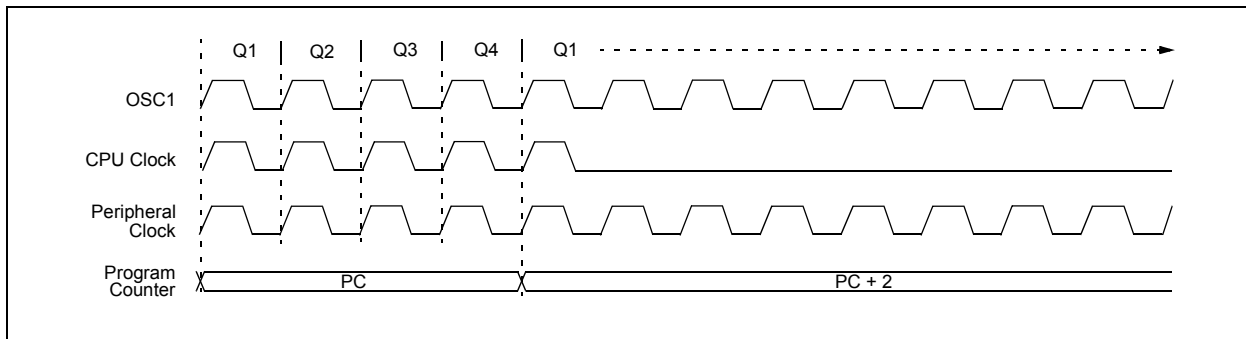
In SEC\_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the Timer1 oscillator. This mode is entered from SEC\_RUN by

setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set the IDLEN bit first, then set the SCS1:SCS0 bits to ‘01’ and execute SLEEP. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the T1RUN bit is set.

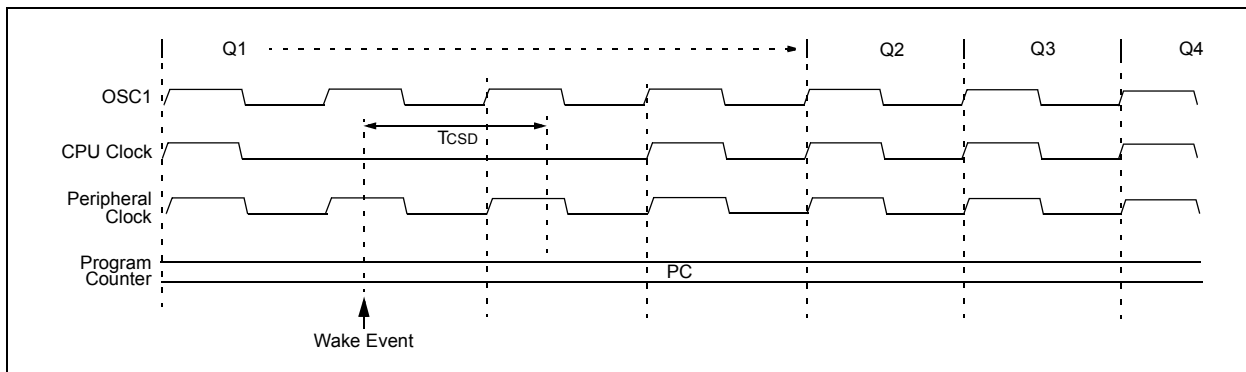
When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of T<sub>CSD</sub> following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 4-8).

**Note:** The Timer1 oscillator should already be running prior to entering SEC\_IDLE mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the SLEEP instruction will be ignored and entry to SEC\_IDLE mode will not occur. If the Timer1 oscillator is enabled but not yet running, peripheral clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result.

**FIGURE 4-7: TRANSITION TIMING FOR ENTRY TO IDLE MODE**



**FIGURE 4-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE**



## 4.4.3 RC\_IDLE MODE

In RC\_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the internal oscillator block using the INTOSC multiplexer. This mode allows for controllable power conservation during Idle periods.

From RC\_RUN, this mode is entered by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, first set IDLEN, then set the SCS1 bit and execute SLEEP. Although its value is ignored, it is recommended that SCS0 also be cleared; this is to maintain software compatibility with future devices. The INTOSC multiplexer may be used to select a higher clock frequency by modifying the IRCF bits before executing the SLEEP instruction. When the clock source is switched to the INTOSC multiplexer, the primary oscillator is shut down and the OSTS bit is cleared.

If the IRCF bits are set to any non-zero value, or the INTSRC bit is set, the INTOSC output is enabled. The IOFS bit becomes set, after the INTOSC output becomes stable, after an interval of TIOBST (parameter 39, Table 23-10). Clocks to the peripherals continue while the INTOSC source stabilizes. If the IRCF bits were previously at a non-zero value, or INTSRC was set before the SLEEP instruction was executed and the INTOSC source was already stable, the IOFS bit will remain set. If the IRCF bits and INTSRC are all clear, the INTOSC output will not be enabled, the IOFS bit will remain clear and there will be no indication of the current clock source.

When a wake event occurs, the peripherals continue to be clocked from the INTOSC multiplexer. After a delay of TCSD following the wake event, the CPU begins executing code being clocked by the INTOSC multiplexer. The IDLEN and SCS bits are not affected by the wake-up. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

## 4.5 Exiting Idle and Sleep Modes

An exit from Sleep mode or any of the Idle modes is triggered by an interrupt, a Reset or a WDT time-out. This section discusses the triggers that cause exits from power-managed modes. The clocking subsystem actions are discussed in each of the power-managed modes (see **Section 4.2 “Run Modes”**, **Section 4.3 “Sleep Mode”** and **Section 4.4 “Idle Modes”**).

### 4.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle mode or the Sleep mode to a Run mode. To enable this functionality, an interrupt source must be enabled by setting its enable bit in one of the INTCON or PIE registers. The exit sequence is initiated when the corresponding interrupt flag bit is set.

On all exits from Idle or Sleep modes by interrupt, code execution branches to the interrupt vector if the GIE/GIEH bit (INTCON<7>) is set. Otherwise, code execution continues or resumes without branching (see **Section 11.0 “Interrupts”**).

A fixed delay of interval TCSD following the wake event is required when leaving Sleep and Idle modes. This delay is required for the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

### 4.5.2 EXIT BY WDT TIME-OUT

A WDT time-out will cause different actions depending on which power-managed mode the device is in when the time-out occurs.

If the device is not executing code (all Idle modes and Sleep mode), the time-out will result in an exit from the power-managed mode (see **Section 4.2 “Run Modes”** and **Section 4.3 “Sleep Mode”**). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 20.2 “Watchdog Timer (WDT)”**).

The WDT timer and postscaler are cleared by executing a SLEEP or CLRWDT instruction, the loss of a currently selected clock source (if the Fail-Safe Clock Monitor is enabled) and modifying the IRCF bits in the OSCCON register if the internal oscillator block is the device clock source.

### 4.5.3 EXIT BY RESET

Normally, the device is held in Reset by the Oscillator Start-up Timer (OST) until the primary clock becomes ready. At that time, the OSTS bit is set and the device begins executing code. If the internal oscillator block is the new clock source, the IOFS bit is set instead.

The exit delay time from Reset to the start of code execution depends on both the clock sources before and after the wake-up and the type of oscillator if the new clock source is the primary clock. Exit delays are summarized in Table 4-2.

Code execution can begin before the primary clock becomes ready. If either the Two-Speed Start-up (see **Section 20.3 “Two-Speed Start-up”**) or Fail-Safe Clock Monitor (see **Section 20.4 “Fail-Safe Clock Monitor”**) is enabled, the device may begin execution as soon as the Reset source has cleared. Execution is clocked by the INTOSC multiplexer driven by the internal oscillator block. Execution is clocked by the internal oscillator block until either the primary clock becomes ready or a power-managed mode is entered before the primary clock becomes ready; the primary clock is then shut down.

# PIC18F1230/1330

## 4.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

Certain exits from power-managed modes do not invoke the OST at all. There are two cases:

- PRI\_IDLE mode, where the primary clock source is not stopped; and
- the primary clock source is not any of the LP, XT, HS or HSPLL modes.

In these instances, the primary clock source either does not require an oscillator start-up delay since it is already running (PRI\_IDLE), or normally does not require an oscillator start-up delay (RC, EC and INTIO Oscillator modes). However, a fixed delay of interval TcSD following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

**TABLE 4-2: EXIT DELAY ON WAKE-UP BY RESET FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)**

Clock Source before Wake-up	Clock Source after Wake-up	Exit Delay	Clock Ready Status Bit (OSCCON)
Primary Device Clock (PRI_IDLE mode)	LP, XT, HS	TcSD <sup>(1)</sup>	OSTS
	HSPLL		
	EC, RC		
	INTOSC <sup>(2)</sup>		IOFS
T1OSC	LP, XT, HS	TOST <sup>(3)</sup>	OSTS
	HSPLL	TOST + t <sub>rc</sub> <sup>(3)</sup>	
	EC, RC	TcSD <sup>(1)</sup>	
	INTOSC <sup>(1)</sup>	TIOBST <sup>(4)</sup>	IOFS
INTOSC <sup>(3)</sup>	LP, XT, HS	TOST <sup>(4)</sup>	OSTS
	HSPLL	TOST + t <sub>rc</sub> <sup>(3)</sup>	
	EC, RC	TcSD <sup>(1)</sup>	
	INTOSC <sup>(1)</sup>	None	IOFS
None (Sleep mode)	LP, XT, HS	TOST <sup>(3)</sup>	OSTS
	HSPLL	TOST + t <sub>rc</sub> <sup>(3)</sup>	
	EC, RC	TcSD <sup>(1)</sup>	
	INTOSC <sup>(1)</sup>	TIOBST <sup>(4)</sup>	IOFS

**Note 1:** TcSD (parameter 38) is a required delay when waking from Sleep and all Idle modes and runs concurrently with any other required delays (see **Section 4.4 “Idle Modes”**). On Reset, INTOSC defaults to 1 MHz.

**2:** Includes both the INTOSC 8 MHz source and postscaler derived frequencies.

**3:** TOST is the Oscillator Start-up Timer (parameter 32). t<sub>rc</sub> is the PLL Lock-out Timer (parameter F12); it is also designated as TPLL.

**4:** Execution continues during TIOBST (parameter 39), the INTOSC stabilization period.

## 5.0 RESET

The PIC18F1230/1330 devices differentiate between various kinds of Reset:

- Power-on Reset (POR)
- $\overline{\text{MCLR}}$  Reset during normal operation
- $\overline{\text{MCLR}}$  Reset during power-managed modes
- Watchdog Timer (WDT) Reset (during execution)
- Programmable Brown-out Reset (BOR)
- RESET Instruction
- Stack Full Reset
- Stack Underflow Reset

This section discusses Resets generated by  $\overline{\text{MCLR}}$ , POR and BOR and covers the operation of the various start-up timers. Stack Reset events are covered in **Section 6.1.2.4 “Stack Full and Underflow Resets”**. WDT Resets are covered in **Section 20.2 “Watchdog Timer (WDT)”**.

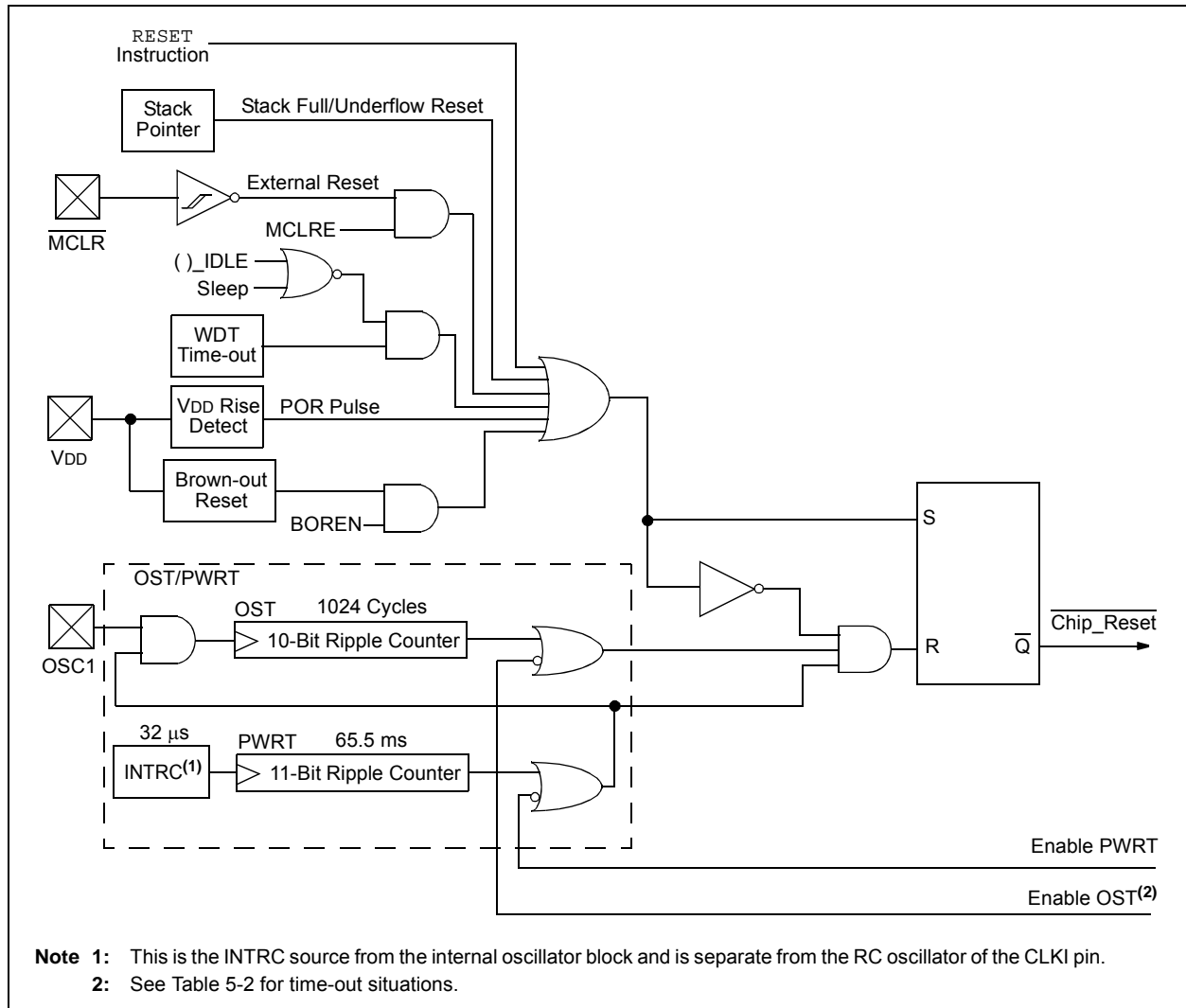
A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 5-1.

## 5.1 RCON Register

Device Reset events are tracked through the RCON register (Register 5-1). The lower five bits of the register indicate that a specific Reset event has occurred. In most cases, these bits can only be cleared by the event and must be set by the application after the event. The state of these flag bits, taken together, can be read to indicate the type of Reset that just occurred. This is described in more detail in **Section 5.6 “Reset State of Registers”**.

The RCON register also has control bits for setting interrupt priority (IPEN) and software control of the BOR (SBOREN). Interrupt priority is discussed in **Section 11.0 “Interrupts”**. BOR is covered in **Section 5.4 “Brown-out Reset (BOR)”**.

**FIGURE 5-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**



# PIC18F1230/1330

## REGISTER 5-1: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1 <sup>(1)</sup>	U-0	R/W-1	R-1	R-1	R/W-0 <sup>(2)</sup>	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 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

- bit 7            **IPEN:** Interrupt Priority Enable bit  
                   1 = Enable priority levels on interrupts  
                   0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
- bit 6            **SBOREN:** BOR Software Enable bit<sup>(1)</sup>  
                   If BOREN1:BOREN0 = 01:  
                   1 = BOR is enabled  
                   0 = BOR is disabled  
                   If BOREN1:BOREN0 = 00, 10 or 11:  
                   Bit is disabled and read as '0'.
- bit 5            **Unimplemented:** Read as '0'
- bit 4             **$\overline{RI}$ :** RESET Instruction Flag bit  
                   1 = The RESET instruction was not executed (set by firmware only)  
                   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 = Set by power-up, CLRWDT instruction or SLEEP instruction  
                   0 = A WDT time-out occurred
- bit 2             **$\overline{PD}$ :** Power-Down Detection Flag bit  
                   1 = Set by power-up or by the CLRWDT instruction  
                   0 = Set by execution of the SLEEP instruction
- bit 1             **$\overline{POR}$ :** Power-on Reset Status bit<sup>(2)</sup>  
                   1 = A Power-on Reset has not occurred (set by firmware only)  
                   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 (set by firmware only)  
                   0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

- Note 1:** If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'.  
**Note 2:** The actual Reset value of  $\overline{POR}$  is determined by the type of device Reset. See the notes following this register and **Section 5.6 "Reset State of Registers"** for additional information.

**Note 1:** 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.

**Note 2:** Brown-out Reset is said to have occurred when  $\overline{BOR}$  is '0' and  $\overline{POR}$  is '1' (assuming that  $\overline{POR}$  was set to '1' by software immediately after a Power-on Reset).



## 5.2 Master Clear ( $\overline{\text{MCLR}}$ )

The  $\overline{\text{MCLR}}$  pin provides a method for triggering an external Reset of the device. A Reset is generated by holding the pin low. These devices have a noise filter in the  $\overline{\text{MCLR}}$  Reset path which detects and ignores small pulses.

The  $\overline{\text{MCLR}}$  pin is not driven low by any internal Resets, including the WDT.

In PIC18F1230/1330 devices, the  $\overline{\text{MCLR}}$  input can be disabled with the MCLRE Configuration bit. When  $\overline{\text{MCLR}}$  is disabled, the pin becomes a digital input. See **Section 10.1 “PORTA, TRISA and LATA Registers”** for more information.

## 5.3 Power-on Reset (POR)

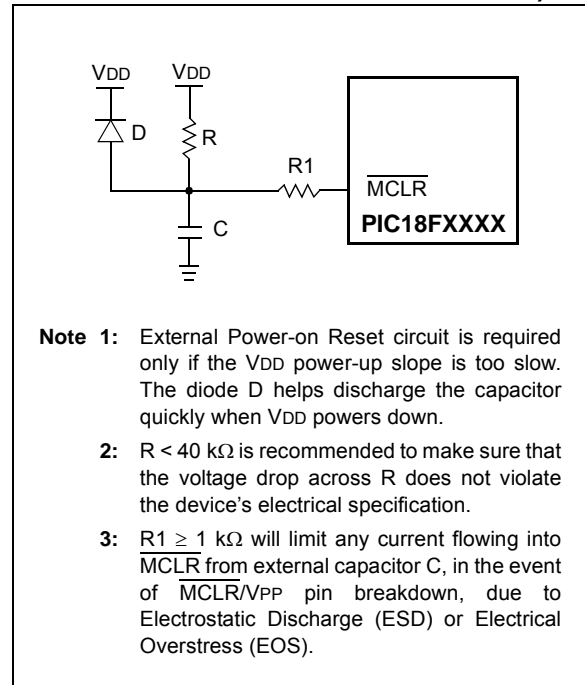
A Power-on Reset pulse is generated on-chip whenever  $V_{DD}$  rises above a certain threshold. This allows the device to start in the initialized state when  $V_{DD}$  is adequate for operation.

To take advantage of the POR circuitry, tie the  $\overline{\text{MCLR}}$  pin through a resistor ( $1\text{ k}\Omega$  to  $10\text{ k}\Omega$ ) to  $V_{DD}$ . This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for  $V_{DD}$  is specified (parameter D004). For a slow rise time, see Figure 5-2.

When the device starts normal operation (i.e., exits the Reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in Reset until the operating conditions are met.

Power-on Reset events are captured by the  $\overline{\text{POR}}$  bit (RCON<1>). The state of the bit is set to ‘0’ whenever a Power-on Reset occurs; it does not change for any other Reset event.  $\overline{\text{POR}}$  is not reset to ‘1’ by any hardware event. To capture multiple events, the user manually resets the bit to ‘1’ in software following any Power-on Reset.

**FIGURE 5-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW  $V_{DD}$  POWER-UP)**



# PIC18F1230/1330

## 5.4 Brown-out Reset (BOR)

PIC18F1230/1330 devices implement a BOR circuit that provides the user with a number of configuration and power-saving options. The BOR is controlled by the BORV1:BORV0 and BOREN1:BOREN0 Configuration bits. There are a total of four BOR configurations which are summarized in Table 5-1.

The BOR threshold is set by the BORV1:BORV0 bits. If BOR is enabled (any values of BOREN1:BOREN0 except '00'), any drop of VDD below VBOR (parameter D005) for greater than TBOR (parameter 35) will reset the device. A Reset may or may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR.

If the Power-up Timer is enabled, it will be invoked after VDD rises above VBOR; it then will keep the chip in Reset for an additional time delay, TPWRT (parameter 33). If VDD drops below VBOR 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 VBOR, the Power-up Timer will execute the additional time delay.

BOR and the Power-on Timer (PWRT) are independently configured. Enabling Brown-out Reset does not automatically enable the PWRT.

### 5.4.1 SOFTWARE ENABLED BOR

When BOREN1:BOREN0 = 01, the BOR can be enabled or disabled by the user in software. This is done with the control bit, SBOREN (RCON<6>). Setting SBOREN enables the BOR to function as previously described. Clearing SBOREN disables the BOR entirely. The SBOREN bit operates only in this mode; otherwise it is read as '0'.

Placing the BOR under software control gives the user the additional flexibility of tailoring the application to its environment without having to reprogram the device to

change BOR configuration. It also allows the user to tailor device power consumption in software by eliminating the incremental current that the BOR consumes. While the BOR current is typically very small, it may have some impact in low-power applications.

**Note:** Even when BOR is under software control, the Brown-out Reset voltage level is still set by the BORV1:BORV0 Configuration bits. It cannot be changed in software.

### 5.4.2 DETECTING BOR

When Brown-out Reset is enabled, the  $\overline{\text{BOR}}$  bit always resets to '0' on any Brown-out Reset or Power-on Reset event. This makes it difficult to determine if a Brown-out Reset event has occurred just by reading the state of  $\overline{\text{BOR}}$  alone. A more reliable method is to simultaneously check the state of both  $\overline{\text{POR}}$  and  $\overline{\text{BOR}}$ . This assumes that the  $\overline{\text{POR}}$  bit is reset to '1' in software immediately after any Power-on Reset event. If BOR is '0' while  $\overline{\text{POR}}$  is '1', it can be reliably assumed that a Brown-out Reset event has occurred.

### 5.4.3 DISABLING BOR IN SLEEP MODE

When BOREN1:BOREN0 = 10, the BOR remains under hardware control and operates as previously described. Whenever the device enters Sleep mode, however, the BOR is automatically disabled. When the device returns to any other operating mode, BOR is automatically re-enabled.

This mode allows for applications to recover from brown-out situations, while actively executing code, when the device requires BOR protection the most. At the same time, it saves additional power in Sleep mode by eliminating the small incremental BOR current.

**TABLE 5-1: BOR CONFIGURATIONS**

BOR Configuration		Status of SBOREN (RCON<6>)	BOR Operation
BOREN1	BOREN0		
0	0	Unavailable	BOR disabled; must be enabled by reprogramming the Configuration bits.
0	1	Available	BOR enabled in software; operation controlled by SBOREN.
1	0	Unavailable	BOR enabled in hardware in Run and Idle modes, disabled during Sleep mode.
1	1	Unavailable	BOR enabled in hardware; must be disabled by reprogramming the Configuration bits.

## 5.5 Device Reset Timers

PIC18F1230/1330 devices incorporate three separate on-chip timers that help regulate the Power-on Reset process. Their main function is to ensure that the device clock is stable before code is executed. These timers are:

- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- PLL Lock Time-out

### 5.5.1 POWER-UP TIMER (PWRT)

The Power-up Timer (PWRT) of PIC18F1230/1330 devices is an 11-bit counter which uses the INTRC source as the clock input. This yields an approximate time interval of  $2048 \times 32 \mu\text{s} = 65.6 \text{ ms}$ . While the PWRT is counting, the device is held in Reset.

The power-up time delay depends on the INTRC clock and will vary from chip to chip due to temperature and process variation. See DC parameter 33 for details.

The PWRT is enabled by clearing the  $\overline{\text{PWRTEN}}$  Configuration bit.

### 5.5.2 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 33). This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP, HS and HSPLL modes and only on Power-on Reset, or on exit from most power-managed modes.

### 5.5.3 PLL LOCK TIME-OUT

With the PLL enabled in its PLL mode, the time-out sequence following a Power-on Reset is slightly different from other oscillator modes. A separate 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 (T<sub>PLL</sub>) is typically 2 ms and follows the oscillator start-up time-out.

### 5.5.4 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows:

1. After the POR pulse has cleared, PWRT time-out is invoked (if enabled).
2. Then, the OST is activated.

The total time-out will vary based on oscillator configuration and the status of the PWRT. Figure 5-3, Figure 5-4, Figure 5-5, Figure 5-6 and Figure 5-7 all depict time-out sequences on power-up, with the Power-up Timer enabled and the device operating in HS Oscillator mode. Figures 5-3 through 5-6 also apply to devices operating in XT or LP modes. For devices in RC mode and with the PWRT disabled, there will be no time-out at all.

Since the time-outs occur from the POR pulse, if  $\overline{\text{MCLR}}$  is kept low long enough, all time-outs will expire. Bringing  $\overline{\text{MCLR}}$  high will begin execution immediately (Figure 5-5). This is useful for testing purposes or to synchronize more than one PIC18FXXXX device operating in parallel.

**TABLE 5-2: TIME-OUT IN VARIOUS SITUATIONS**

Oscillator Configuration	Power-up <sup>(2)</sup> and Brown-out Reset		Exit from Power-Managed Mode
	$\overline{\text{PWRTEN}} = 0$	$\overline{\text{PWRTEN}} = 1$	
HSPLL	$66 \text{ ms}^{(1)} + 1024 \text{ T}_{\text{osc}} + 2 \text{ ms}^{(2)}$	$1024 \text{ T}_{\text{osc}} + 2 \text{ ms}^{(2)}$	$1024 \text{ T}_{\text{osc}} + 2 \text{ ms}^{(2)}$
HS, XT, LP	$66 \text{ ms}^{(1)} + 1024 \text{ T}_{\text{osc}}$	$1024 \text{ T}_{\text{osc}}$	$1024 \text{ T}_{\text{osc}}$
EC, ECIO	$66 \text{ ms}^{(1)}$	—	—
RC, RCIO	$66 \text{ ms}^{(1)}$	—	—
INTIO1, INTIO2	$66 \text{ ms}^{(1)}$	—	—

**Note 1:** 66 ms (65.5 ms) is the nominal Power-up Timer (PWRT) delay.

**Note 2:** 2 ms is the nominal time required for the PLL to lock.

# PIC18F1230/1330

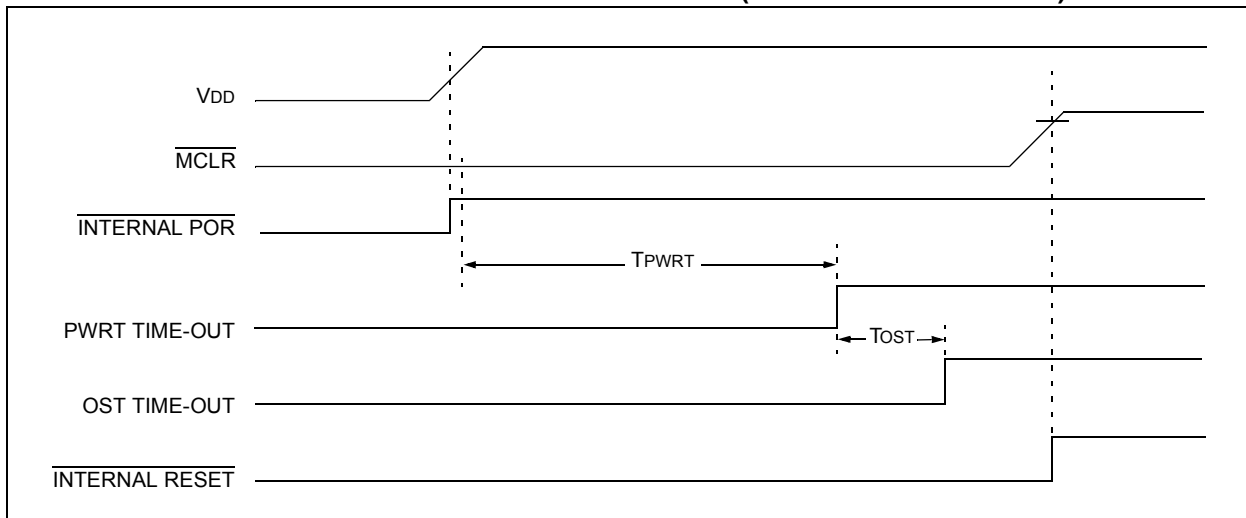
**FIGURE 5-3: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ ,  $V_{DD}$  RISE <  $T_{PWRT}$ )**



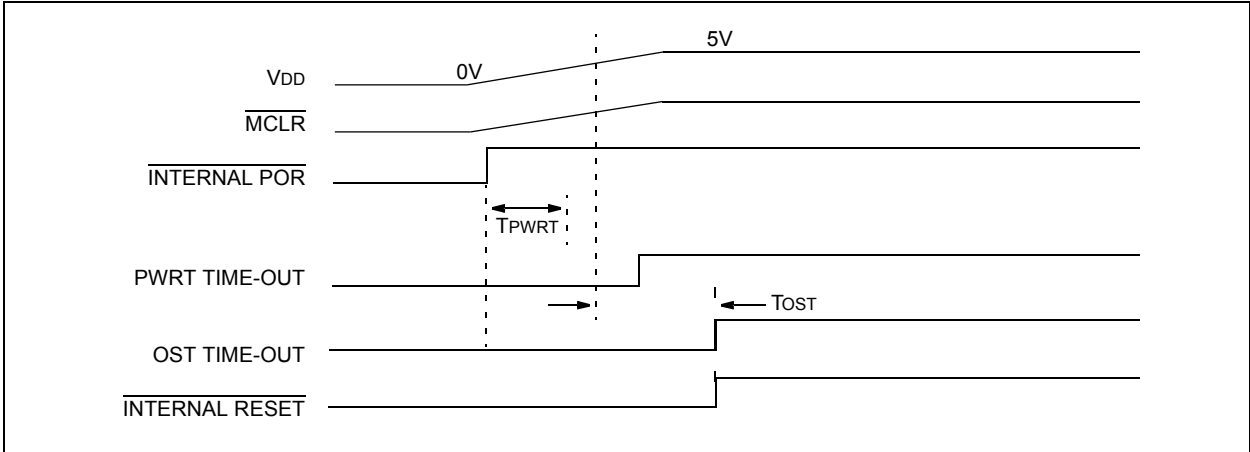
**FIGURE 5-4: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 1**



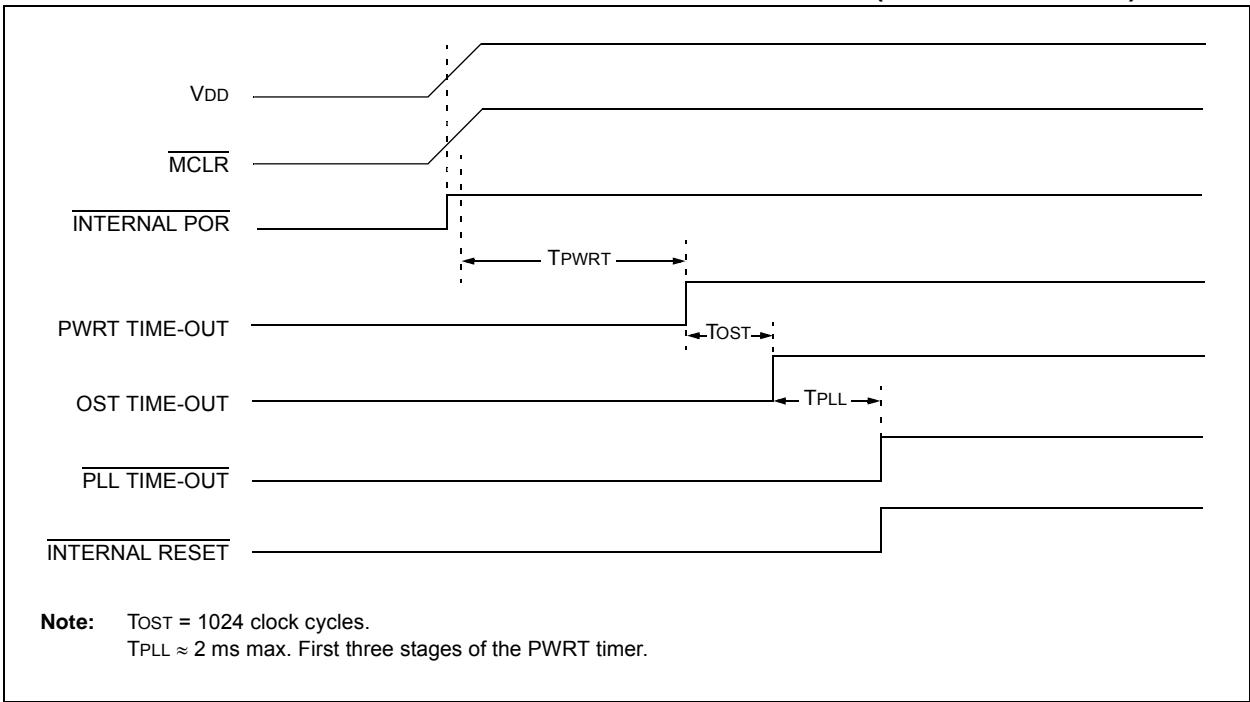
**FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{DD}$ ): CASE 2**



**FIGURE 5-6: SLOW RISE TIME ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ ,  $V_{DD}$  RISE  $>$   $T_{PWRT}$ )**



**FIGURE 5-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED ( $\overline{\text{MCLR}}$  TIED TO  $V_{DD}$ )**



# PIC18F1230/1330

## 5.6 Reset State of Registers

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” depending on the type of Reset that occurred.

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,  $\overline{RI}$ ,  $\overline{TO}$ ,  $\overline{PD}$ ,  $\overline{POR}$  and  $\overline{BOR}$ , are set or cleared differently in different Reset situations, as indicated in Table 5-3. These bits are used in software to determine the nature of the Reset.

Table 5-4 describes the Reset states for all of the Special Function Registers. These are categorized by Power-on and Brown-out Resets, Master Clear and WDT Resets and WDT wake-ups.

**TABLE 5-3: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

Condition	Program Counter	RCON Register						STKPTR Register	
		SBOREN	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	STKFUL	STKUNF
Power-on Reset	0000h	1	1	1	1	0	0	0	0
RESET Instruction	0000h	u <sup>(2)</sup>	0	u	u	u	u	u	u
Brown-out Reset	0000h	u <sup>(2)</sup>	1	1	1	u	0	u	u
$\overline{MCLR}$ during Power-Managed Run Modes	0000h	u <sup>(2)</sup>	u	1	u	u	u	u	u
$\overline{MCLR}$ during Power-Managed Idle Modes and Sleep Mode	0000h	u <sup>(2)</sup>	u	1	0	u	u	u	u
WDT Time-out during Full Power or Power-Managed Run Mode	0000h	u <sup>(2)</sup>	u	0	u	u	u	u	u
$\overline{MCLR}$ during Full Power Execution	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	u
Stack Full Reset (STVREN = 1)	0000h	u <sup>(2)</sup>	u	u	u	u	u	1	u
Stack Underflow Reset (STVREN = 1)	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	1
Stack Underflow Error (not an actual Reset, STVREN = 0)	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	1
WDT Time-out during Power-Managed Idle or Sleep Modes	PC + 2	u <sup>(2)</sup>	u	0	0	u	u	u	u
Interrupt Exit from Power-Managed Modes	PC + 2 <sup>(1)</sup>	u <sup>(2)</sup>	u	u	0	u	u	u	u

**Legend:** u = unchanged

**Note 1:** When the wake-up is due to an interrupt and the GIEH or GIEL bit is set, the PC is loaded with the interrupt vector (008h or 0018h).

**2:** Reset state is '1' for POR and unchanged for all other Resets when software BOR is enabled (BOREN1:BOREN0 Configuration bits = 01 and SBOREN = 1); otherwise, the Reset state is '0'.

**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
TOSU	1230	1330	---0 0000	---0 0000	---0 uuuu <sup>(3)</sup>
TOSH	1230	1330	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
TOSL	1230	1330	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
STKPTR	1230	1330	00-0 0000	uu-0 0000	uu-u uuuu <sup>(3)</sup>
PCLATU	1230	1330	---0 0000	---0 0000	---u uuuu
PCLATH	1230	1330	0000 0000	0000 0000	uuuu uuuu
PCL	1230	1330	0000 0000	0000 0000	PC + 2 <sup>(2)</sup>
TBLPTRU	1230	1330	--00 0000	--00 0000	--uu uuuu
TBLPTRH	1230	1330	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	1230	1330	0000 0000	0000 0000	uuuu uuuu
TABLAT	1230	1330	0000 0000	0000 0000	uuuu uuuu
PRODH	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	1230	1330	0000 000x	0000 000u	uuuu uuuu <sup>(1)</sup>
INTCON2	1230	1330	1111 1111	1111 1111	uuuu uuuu <sup>(1)</sup>
INTCON3	1230	1330	1100 0000	1100 0000	uuuu uuuu <sup>(1)</sup>
INDF0	1230	1330	N/A	N/A	N/A
POSTINC0	1230	1330	N/A	N/A	N/A
POSTDEC0	1230	1330	N/A	N/A	N/A
PREINC0	1230	1330	N/A	N/A	N/A
PLUSW0	1230	1330	N/A	N/A	N/A
FSR0H	1230	1330	---- 0000	---- 0000	---- uuuu
FSR0L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	1230	1330	N/A	N/A	N/A
POSTINC1	1230	1330	N/A	N/A	N/A
POSTDEC1	1230	1330	N/A	N/A	N/A
PREINC1	1230	1330	N/A	N/A	N/A
PLUSW1	1230	1330	N/A	N/A	N/A
FSR1H	1230	1330	---- 0000	---- 0000	---- uuuu
FSR1L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	1230	1330	---- 0000	---- 0000	---- uuuu

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

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

- 2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4: See Table 5-3 for Reset value for specific condition.
- 5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as '0'.
- 6: Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.

# PIC18F1230/1330

**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
INDF2	1230	1330	N/A	N/A	N/A
POSTINC2	1230	1330	N/A	N/A	N/A
POSTDEC2	1230	1330	N/A	N/A	N/A
PREINC2	1230	1330	N/A	N/A	N/A
PLUSW2	1230	1330	N/A	N/A	N/A
FSR2H	1230	1330	---- 0000	---- 0000	---- uuuu
FSR2L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	1230	1330	---x xxxx	---u uuuu	---u uuuu
TMR0H	1230	1330	0000 0000	0000 0000	uuuu uuuu
TMR0L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	1230	1330	1111 1111	1111 1111	uuuu uuuu
OSCCON	1230	1330	0100 q000	0100 q000	uuuu uuqu
LVDCON	1230	1330	--00 0101	--00 0101	--uu uuuu
WDTCON	1230	1330	---- ---0	---- ---0	---- ---u
RCON <sup>(4)</sup>	1230	1330	0q-1 11q0	0q-q qqnu	uq-u qqnu
TMR1H	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	1230	1330	0000 0000	u0uu uuuu	uuuu uuuu
ADRESH	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	1230	1330	0--- 0000	0--- 0000	u--- uuuu
ADCON1	1230	1330	---0 1111	---0 1111	---u uuuu
ADCON2	1230	1330	0-00 0000	0-00 0000	u-uu uuuu
BAUDCON	1230	1330	01-00 0-00	01-00 0-00	uu-uu u-uu
CVRCON	1230	1330	0-00 0000	0-00 0000	u-uu uuuu
CMCON	1230	1330	000- -000	000- -000	uuu- -uuu
SPBRGH	1230	1330	0000 0000	0000 0000	uuuu uuuu
SPBRG	1230	1330	0000 0000	0000 0000	uuuu uuuu
RCREG	1230	1330	0000 0000	0000 0000	uuuu uuuu
TXREG	1230	1330	0000 0000	0000 0000	uuuu uuuu
TXSTA	1230	1330	0000 0010	0000 0010	uuuu uuuu
RCSTA	1230	1330	0000 000x	0000 000x	uuuu uuuu

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

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

- 2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4: See Table 5-3 for Reset value for specific condition.
- 5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as '0'.
- 6: Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.



**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
EEADR	1230	1330	0000 0000	0000 0000	uuuu uuuu
EEDATA	1230	1330	0000 0000	0000 0000	uuuu uuuu
EECON2	1230	1330	0000 0000	0000 0000	0000 0000
EECON1	1230	1330	xx-0 x000	uu-0 u000	uu-0 u000
IPR3	1230	1330	---1 ----	---1 ----	---u ----
PIR3	1230	1330	---0 ----	---0 ----	---u ----
PIE3	1230	1330	---0 ----	---0 ----	---u ----
IPIR2	1230	1330	1--1 -1--	1--1 -1--	u--u -u--
PIR2	1230	1330	0--0 -0--	0--0 -0--	u--u -u-- <sup>(1)</sup>
PIE2	1230	1330	0--0 -0--	0--0 -0--	u--u -u--
IPR1	1230	1330	-111 1111	-111 1111	-uuu uuuu
PIR1	1230	1330	-000 0000	-000 0000	-uuu uuuu <sup>(1)</sup>
PIE1	1230	1330	-000 0000	-000 0000	-uuu uuuu
OSCTUNE	1230	1330	00-0 0000	00-0 0000	uu-u uuuu
PTCON0	1230	1330	0000 0000	uuuu uuuu	uuuu uuuu
PTCON1	1230	1330	00-- ----	00-- ----	uu-- ----
PTMRL	1230	1330	0000 0000	0000 0000	uuuu uuuu
PTMRH	1230	1330	---- 0000	---- 0000	---- uuuu
PTPERL	1230	1330	1111 1111	1111 1111	uuuu uuuu
PTPERH	1230	1330	---- 1111	---- 1111	---- uuuu
TRISB	1230	1330	1111 1111	1111 1111	uuuu uuuu
TRISA	1230	1330	1111 1111 <sup>(5)</sup>	1111 1111 <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>
PDC0L	1230	1330	0000 0000	0000 0000	uuuu uuuu
PDC0H	1230	1330	--00 0000	--00 0000	--uu uuuu
PDC1L	1230	1330	0000 0000	0000 0000	uuuu uuuu
PDC1H	1230	1330	--00 0000	--00 0000	--uu uuuu
PDC2L	1230	1330	0000 0000	0000 0000	uuuu uuuu
PDC2H	1230	1330	--00 0000	--00 0000	--uu uuuu
FLTCONFIG	1230	1330	0--- -000	0--- -000	u--- -uuu
LATB	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA	1230	1330	xxxx xxxx <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>
SEVTCMPL	1230	1330	0000 0000	0000 0000	uuuu uuuu

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

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

**2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

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

**4:** See Table 5-3 for Reset value for specific condition.

**5:** Bits 6 and 7 of PORTA, LATA and TRISA are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as '0'.

**6:** Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.

# PIC18F1230/1330

**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
SEVTCMPH	1230	1330	---- 0000	---- 0000	---- uuuu
PWMCON0	1230	1330	-100 -000 <sup>(6)</sup>	-100 -000 <sup>(6)</sup>	-uuu -uuu <sup>(6)</sup>
			-000 -000 <sup>(6)</sup>	-000 -000 <sup>(6)</sup>	-uuu -uuu <sup>(6)</sup>
PWMCON1	1230	1330	0000 0-00	0000 0-00	uuuu u-uu
DTCON	1230	1330	0000 0000	0000 0000	uuuu uuuu
OVDCOND	1230	1330	--11 1111	--11 1111	--uu uuuu
OVDCONS	1230	1330	--00 0000	--00 0000	--uu uuuu
PORTB	1230	1330	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	1230	1330	xx0x xxxx <sup>(5)</sup>	uu0u uuuu <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>

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

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4:** See Table 5-3 for Reset value for specific condition.
- 5:** Bits 6 and 7 of PORTA, LATA and TRISA are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as '0'.
- 6:** Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.

## 6.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 Enhanced microcontroller devices:

- Program Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate busses; this allows for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Flash program memory is provided in **Section 7.0 “Flash Program Memory”**. Data EEPROM is discussed separately in **Section 8.0 “Data EEPROM Memory”**.

## 6.1 Program Memory Organization

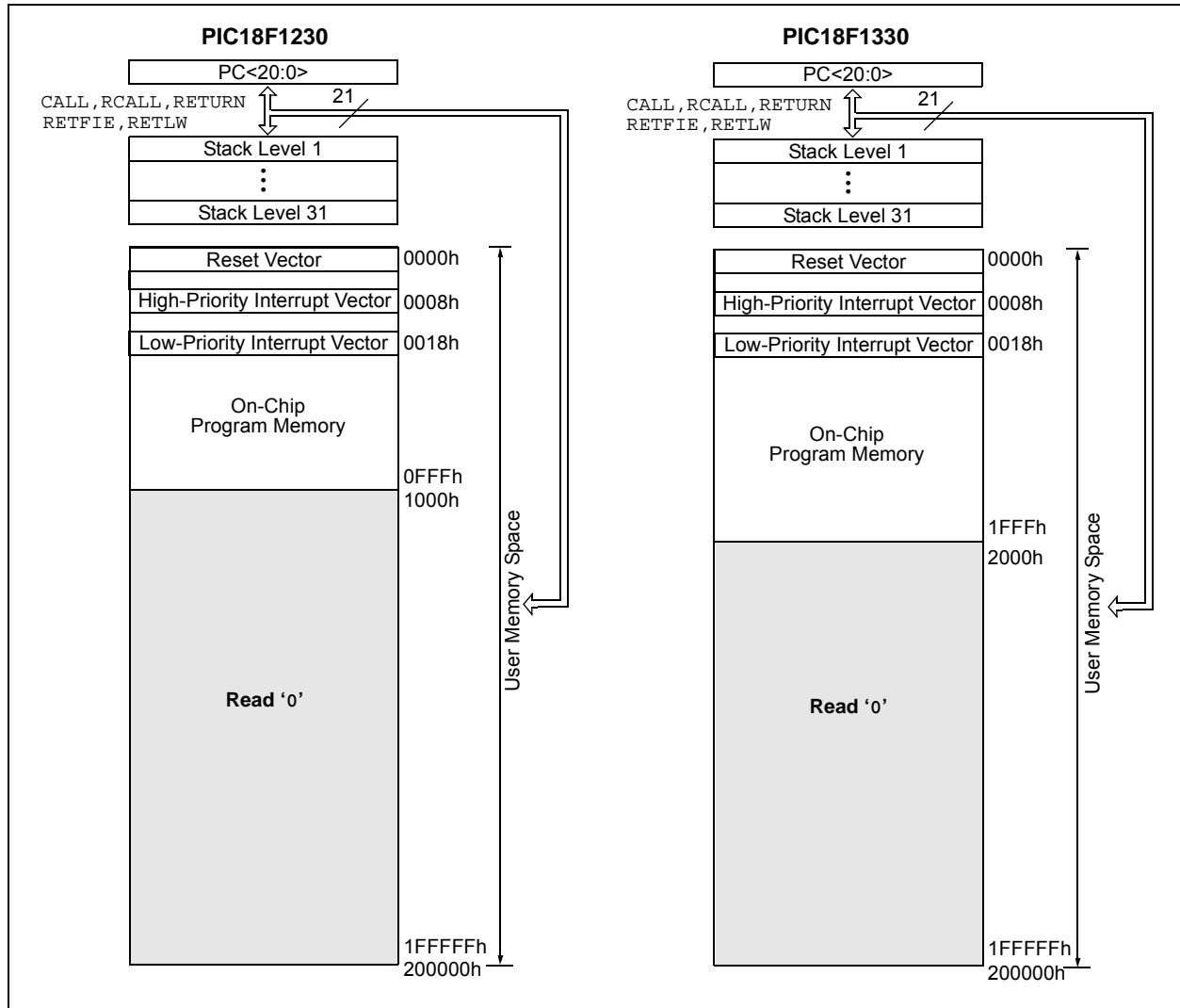
PIC18 microcontrollers implement a 21-bit program counter, which is capable of addressing a 2-Mbyte program memory space. Accessing a location between the upper boundary of the physically implemented memory and the 2-Mbyte address will return all ‘0’s (a NOP instruction).

The PIC18F1230 has 4 Kbytes of Flash memory and can store up to 2,048 single-word instructions. The PIC18F1330 has 8 Kbytes of Flash memory and can store up to 4,096 single-word instructions.

PIC18 devices have two interrupt vectors. The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

The program memory maps for PIC18F1230 and PIC18F1330 devices are shown in Figure 6-1.

**FIGURE 6-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F1230/1330 DEVICES**



# PIC18F1230/1330

## 6.1.1 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes to the PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads the PCL. This is useful for computed offsets to the PC (see **Section 6.1.4.1 “Computed GOTO”**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit 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.

## 6.1.2 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a 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 or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, STKPTR. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack Special Function Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

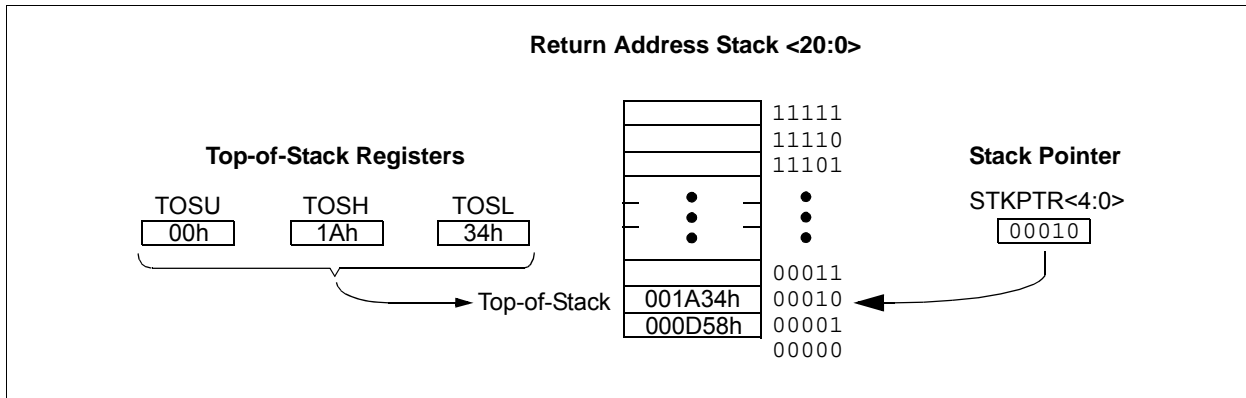
The Stack Pointer is initialized to ‘00000’ after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of ‘00000’; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

### 6.1.2.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 6-2). 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:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the global interrupt enable bits while accessing the stack to prevent inadvertent stack corruption.

**FIGURE 6-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



## 6.1.2.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 6-1) contains the Stack Pointer value, the STKFUL (Stack Full) status bit and the STKUNF (Stack Underflow) status bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) 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 is cleared by 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 20.1 “Configuration Bits”** 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 zero.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the Stack Pointer will increment to 31. Any additional pushes will not overwrite the 31st push and STKPTR will remain 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 zero. The STKUNF bit will remain set until cleared by software or until 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. This is not the same as a Reset, as the contents of the SFRs are not affected.

## 6.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack 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 feature. The PIC18 instruction set includes two instructions, `PUSH` and `POP`, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The `PUSH` instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto 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.

## REGISTER 6-1: 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 <sup>(1)</sup>	STKUNF <sup>(1)</sup>	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

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

- bit 7      **STKFUL:** Stack Full Flag bit<sup>(1)</sup>  
1 = Stack became full or overflowed  
0 = Stack has not become full or overflowed
- bit 6      **STKUNF:** Stack Underflow Flag bit<sup>(1)</sup>  
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 1:** Bit 7 and bit 6 are cleared by user software or by a POR.

# PIC18F1230/1330

## 6.1.2.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a full or underflow will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit but not cause a device Reset. The STKFUL or STKUNF bit is cleared by the user software or a Power-on Reset.

## 6.1.3 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers, to provide a “fast return” option for interrupts. The stack for each register is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the Stack registers. The values in the registers are then loaded back into their associated registers if the RETFIE, FAST instruction is used to return from the interrupt.

If both low and high-priority interrupts are enabled, the Stack registers cannot be used reliably to return from 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. In these cases, users must save the key registers in software during a low-priority interrupt.

If interrupt priority is not used, all interrupts may use the Fast Register Stack for returns from 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 CALL label, FAST instruction must be executed to save the STATUS, WREG and BSR registers to the Fast Register Stack. A RETURN, FAST instruction is then executed to restore these registers from the Fast Register Stack.

Example 6-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

### EXAMPLE 6-1: 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
```

## 6.1.4 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

### 6.1.4.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 6-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register 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 nn instructions that returns the value ‘nn’ to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of 2 (LSb = 0).

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

### EXAMPLE 6-2: COMPUTED GOTO USING AN OFFSET VALUE

```
MOVWF OFFSET, W
CALL TABLE
ORG nn00h
TABLE ADDWF PCL
      RETLW nnh
      RETLW nnh
      RETLW nnh
      .
      .
      .
```

### 6.1.4.2 Table Reads and Table Writes

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

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

Table read and table write operations are discussed further in Section 7.1 “Table Reads and Table Writes”.

## 6.2 PIC18 Instruction Cycle

### 6.2.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the program counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the Instruction Register (IR) during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 6-3.

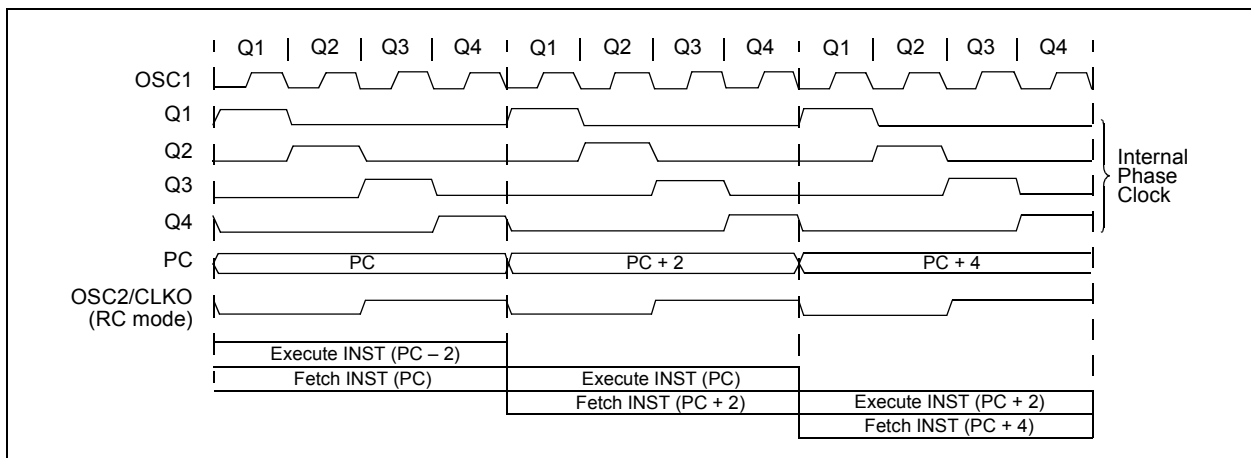
### 6.2.2 INSTRUCTION FLOW/PIPELINING

An "Instruction Cycle" consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the 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), then two cycles are required to complete the instruction (Example 6-3).

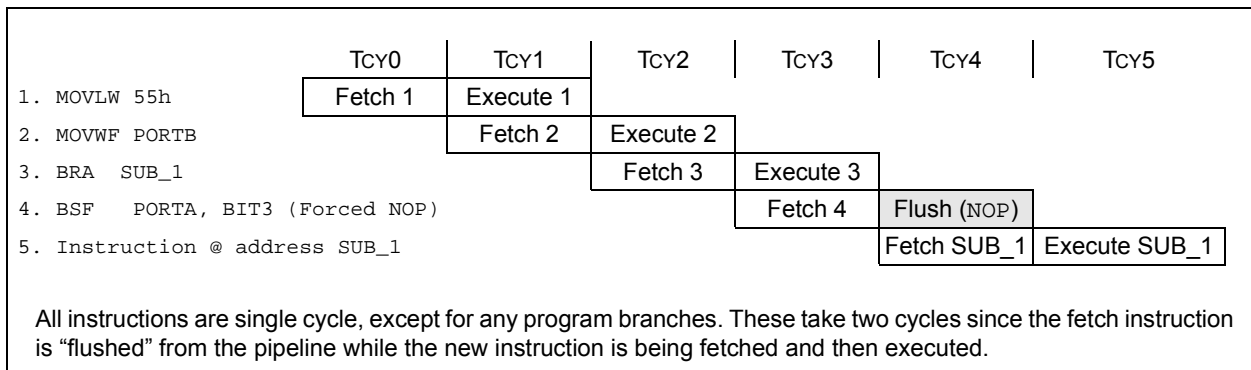
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

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).

**FIGURE 6-3: CLOCK/INSTRUCTION CYCLE**



**EXAMPLE 6-3: INSTRUCTION PIPELINE FLOW**



# PIC18F1230/1330

## 6.2.3 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). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSb will always read '0' (see **Section 6.1.1 "Program Counter"**).

Figure 6-4 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 6-4 shows how the instruction, GOTO 0006h, is encoded in the program memory. Program branch instructions, which 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 that the PC will be offset by. **Section 22.0 "Instruction Set Summary"** provides further details of the instruction set.

**FIGURE 6-4: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →			Word Address	
			LSB = 1	LSB = 0
				000000h
				000002h
				000004h
				000006h
Instruction 1:	MOVLW	055h	0Fh	55h
Instruction 2:	GOTO	0006h	EFh	03h
			F0h	00h
Instruction 3:	MOVFF	123h, 456h	C1h	23h
			F4h	56h
				000010h
				000012h
				000014h

## 6.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: CALL, MOVFF, GOTO and LSRF. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSBs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed

and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 6-4 shows how this works.

**Note:** See **Section 6.6 "PIC18 Instruction Execution and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

**EXAMPLE 6-4: 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, skip this word
1111 0100 0101 0110		; Execute this word as a NOP
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, execute this word
1111 0100 0101 0110		; 2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3 ; continue code



## 6.3 Data Memory Organization

**Note:** The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 6.5 “Data Memory and the Extended Instruction Set”** for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each; PIC18F1230/1330 devices implement 1 bank. Figure 6-5 shows the data memory organization for the PIC18F1230/1330 devices.

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. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 6.3.2 “Access Bank”** provides a detailed description of the Access RAM.

### 6.3.1 BANK SELECT REGISTER (BSR)

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the four Most Significant bits of a location's address; the instruction itself includes the eight Least Significant bits. Only the four lower bits of the BSR are implemented (BSR3:BSR0). The upper four bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the `MOVLB` instruction.

The value of the BSR indicates the bank in data memory. The 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 6-6.

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h, while the BSR is 0Fh, will end up resetting the program counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory map in Figure 6-5 indicates which banks are implemented.

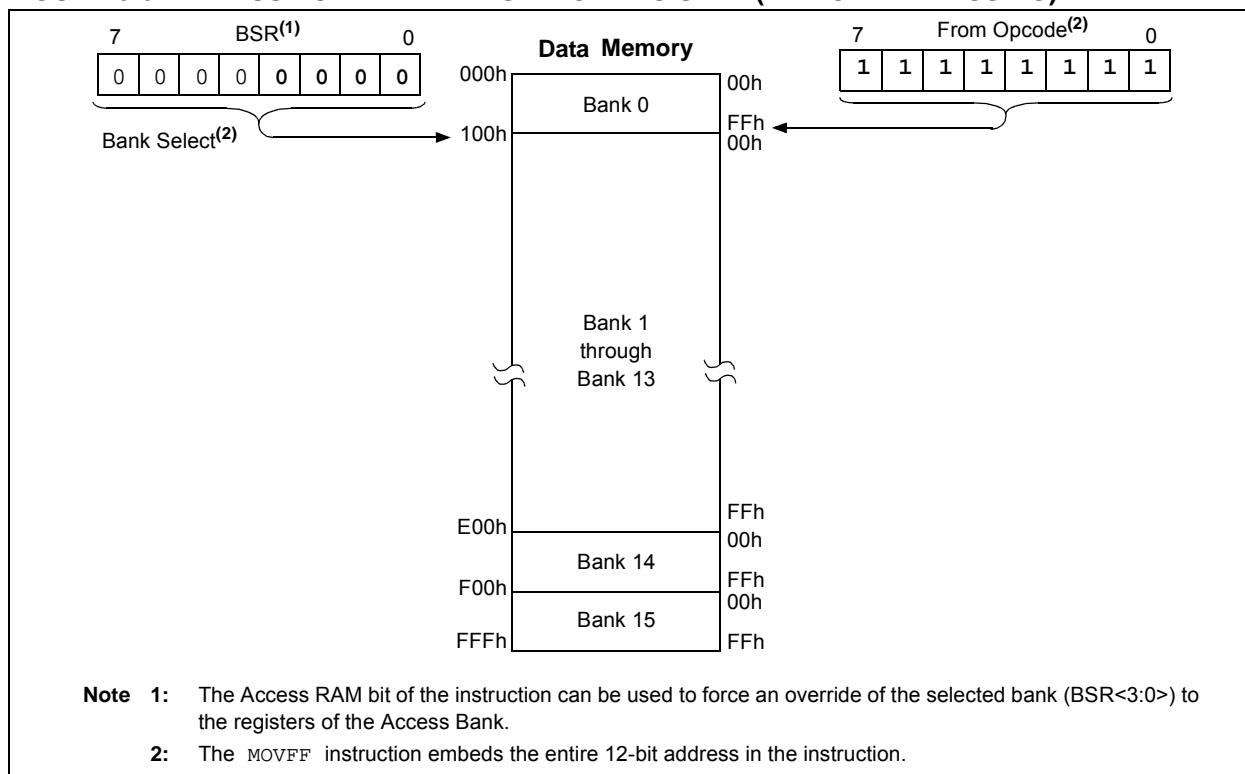
In the core PIC18 instruction set, only the `MOVFF` instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

# PIC18F1230/1330

**FIGURE 6-5: DATA MEMORY MAP FOR PIC18F1230/1330 DEVICES**



**FIGURE 6-6: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)**



## 6.3.2 ACCESS BANK

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 128 bytes of memory (00h-7Fh) in Bank 0 and the last 128 bytes of memory (80h-FFh) in Block 15. The lower half is known as the "Access RAM" and is composed of GPRs. The upper half is where the device's SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 6-5).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the 'a' parameter in the instruction). When 'a' is equal to '1', the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When 'a' is '0',

however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

Using this "forced" addressing allows the instruction to operate on a data address in a single cycle without updating the BSR first. For 8-bit addresses of 80h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 80h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in **Section 6.5.3 "Mapping the Access Bank in Indexed Literal Offset Addressing Mode"**.

## 6.3.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

# PIC18F1230/1330

## 6.3.4 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. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top half of Bank 15 (F80h to FFFh). A list of these registers is given in Table 6-1 and Table 6-2.

The SFRs can be classified into two sets: those associated with the “core” device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and Interrupt registers are described in their respective chapters, while the ALU’s STATUS register is described later in this section. Registers related to the operation of a peripheral feature are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as ‘0’s.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F1230/1330 DEVICES**

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDfh	INDF2 <sup>(1)</sup>	FBFh	— <sup>(2)</sup>	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2 <sup>(1)</sup>	FBEh	— <sup>(2)</sup>	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2 <sup>(1)</sup>	FBDh	— <sup>(2)</sup>	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2 <sup>(1)</sup>	FBCh	— <sup>(2)</sup>	F9Ch	— <sup>(2)</sup>
FFBh	PCLATU	FDBh	PLUSW2 <sup>(1)</sup>	FBBh	— <sup>(2)</sup>	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	— <sup>(2)</sup>	F9Ah	PTCON0
FF9h	PCL	FD9h	FSR2L	FB9h	— <sup>(2)</sup>	F99h	PTCON1
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	PTMRL
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	— <sup>(2)</sup>	F97h	PTMRH
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	— <sup>(2)</sup>	F96h	PTPERL
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	PTPERH
FF4h	PRODH	FD4h	— <sup>(2)</sup>	FB4h	CMCON	F94h	— <sup>(2)</sup>
FF3h	PRODL	FD3h	OSCCON	FB3h	— <sup>(2)</sup>	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	— <sup>(2)</sup>	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	— <sup>(2)</sup>	F91h	PDC0L
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	PDC0H
FEFh	INDF0 <sup>(1)</sup>	FCFh	TMR1H	FAFh	SPBRG	F8Fh	PDC1L
FEeh	POSTINC0 <sup>(1)</sup>	FCEh	TMR1L	FAEh	RCREG	F8Eh	PDC1H
FEDh	POSTDEC0 <sup>(1)</sup>	FCDh	T1CON	FADh	TXREG	F8Dh	PDC2L
FECh	PREINC0 <sup>(1)</sup>	FCCh	— <sup>(2)</sup>	FACH	TXSTA	F8Ch	PDC2H
FEbh	PLUSW0 <sup>(1)</sup>	FCBh	— <sup>(2)</sup>	FABh	RCSTA	F8Bh	FLTCONFIG
FEAh	FSR0H	FCAh	— <sup>(2)</sup>	FAAh	— <sup>(2)</sup>	F8Ah	LATB
FE9h	FSR0L	FC9h	— <sup>(2)</sup>	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	— <sup>(2)</sup>	FA8h	EEDATA	F88h	SEVTCMPL
FE7h	INDF1 <sup>(1)</sup>	FC7h	— <sup>(2)</sup>	FA7h	EECON2 <sup>(1)</sup>	F87h	SEVTCMPH
FE6h	POSTINC1 <sup>(1)</sup>	FC6h	— <sup>(2)</sup>	FA6h	EECON1	F86h	PWMCON0
FE5h	POSTDEC1 <sup>(1)</sup>	FC5h	— <sup>(2)</sup>	FA5h	IPR3	F85h	PWMCON1
FE4h	PREINC1 <sup>(1)</sup>	FC4h	ADRESH	FA4h	PIR3	F84h	DTCON
FE3h	PLUSW1 <sup>(1)</sup>	FC3h	ADRESL	FA3h	PIE3	F83h	OVDCOND
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	OVDCONS
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

- Note 1:** This is not a physical register.  
**Note 2:** Unimplemented registers are read as ‘0’.

# PIC18F1230/1330

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F1230/1330)**

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	47, 52
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	47, 52
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	47, 52
STKPTR	STKFUL <sup>(5)</sup>	STKUNF <sup>(5)</sup>	—	SP4	SP3	SP2	SP1	SP0	00-0 0000	47, 53
PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000	47, 52
PCLATH	Holding Register for PC<15:8>								0000 0000	47, 52
PCL	PC Low Byte (PC<7:0>)								0000 0000	47, 52
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	47, 74
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	47, 74
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	47, 74
TABLAT	Program Memory Table Latch								0000 0000	47, 74
PRODH	Product Register High Byte								xxxx xxxx	47, 85
PRODL	Product Register Low Byte								xxxx xxxx	47, 85
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	47, 95
INTCON2	$\overline{\text{RBP}}\text{U}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	1111 1111	47, 96
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	1100 0000	47, 97
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								N/A	47, 66
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								N/A	47, 66
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								N/A	47, 66
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								N/A	47, 66
PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W								N/A	47, 66
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte				---- 0000	47, 66
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	47, 66
WREG	Working Register								xxxx xxxx	47, 54
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								N/A	47, 66
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								N/A	47, 66
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								N/A	47, 66
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								N/A	47, 66
PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W								N/A	47, 66
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte				---- 0000	47, 66
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	47, 66
BSR	—	—	—	—	Bank Select Register				---- 0000	47, 57
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								N/A	48, 66
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								N/A	48, 66
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								N/A	48, 66
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								N/A	48, 66
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W								N/A	48, 66
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte				---- 0000	48, 66
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	48, 66

**Legend:** x = unknown, u = unchanged, - = unimplemented,  $\alpha$  = value depends on condition

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 “Brown-out Reset (BOR)”**.

**2:** The PLLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 “PLL in INTOSC Modes”**.

**3:** The RA5 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RA5 reads as '0'. This bit is read-only.

**4:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**5:** Bit 7 and bit 6 are cleared by user software or by a POR.

**6:** Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.

**7:** This bit has no effect if the Configuration bit, WDTEN, is enabled.

# PIC18F1230/1330

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F1230/1330) (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:
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxxx	48, 64
TMR0H	Timer0 Register High Byte								0000 0000	48, 109
TMR0L	Timer0 Register Low Byte								xxxx xxxxx	48, 109
T0CON	TMR0ON	T016BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	48, 107
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0100 q000	48, 28
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	48, 187
WDTCON	—	—	—	—	—	—	—	SWDTEN <sup>(7)</sup>	---- --0	48, 203
RCON	IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR	BOR	0q-1 11q0	48, 40
TMR1H	Timer1 Register High Byte								xxxx xxxxx	48, 115
TMR1L	Timer1 Register Low Byte								xxxx xxxxx	48, 115
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCN	T1SYNC	TMR1CS	TMR1ON	0000 0000	48, 111
ADRESH	A/D Result Register High Byte								xxxx xxxxx	48, 178
ADRESL	A/D Result Register Low Byte								xxxx xxxxx	48, 178
ADCON0	SEVTEN	—	—	—	CHS1	CHS0	GO/DONE	ADON	0--- 0000	48, 169
ADCON1	—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	---0 1111	48, 170
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	48, 171
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	01-0 00-00	48, 150
CVRCON	CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0-00 0000	48, 184
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	000- -000	48, 179
SPBRGH	EUSART Baud Rate Generator Register High Byte								0000 0000	48, 152
SPBRG	EUSART Baud Rate Generator Register Low Byte								0000 0000	48, 152
RCREG	EUSART Receive Register								0000 0000	48, 160
TXREG	EUSART Transmit Register								0000 0000	48, 157
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	48, 148
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	48, 149
EEADR	EEPROM Address Register								0000 0000	49, 81
EEDATA	EEPROM Data Register								0000 0000	49, 81
EECON2	EEPROM Control Register 2 (not a physical register)								0000 0000	49, 72
EECON1	EEPGD	CFGFS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	48, 73
IPR3	—	—	—	PTIP	—	—	—	—	---1 ----	49, 103
PIR3	—	—	—	PTIF	—	—	—	—	---0 ----	49, 99
PIE3	—	—	—	PTIE	—	—	—	—	---0 ----	49, 101
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	1--1 -1--	49, 103
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	0--0 -0--	49, 99
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	0--0 -0--	49, 101
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	-111 1111	49, 102
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	-000 0000	49, 98
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	-000 0000	49, 100
OSCTUNE	INTSRC	PLLEN <sup>(2)</sup>	—	TUN4	TUN3	TUN2	TUN1	TUN0	00-0 0000	49, 25
PTCON0	PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0	0000 0000	49, 122
PTCON1	PTEN	PTDIR	—	—	—	—	—	—	00-- ----	49, 122

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 “Brown-out Reset (BOR)”**.

**2:** The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 “PLL in INTOSC Modes”**.

**3:** The RA5 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RA5 reads as '0'. This bit is read-only.

**4:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**5:** Bit 7 and bit 6 are cleared by user software or by a POR.

**6:** Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.

**7:** This bit has no effect if the Configuration bit, WDTEN, is enabled.

# PIC18F1230/1330

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F1230/1330) (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:	
PTMRL	PWM Time Base Register (lower 8 bits)								0000 0000	49, 125	
PTMRH	—	—	—	—	PWM Time Base Register (upper 4 bits)				---- 0000	49, 125	
PTPERL	PWM Time Base Period Register (lower 8 bits)								1111 1111	49, 125	
PTPERH	—	—	—	—	PWM Time Base Period Register (upper 4 bits)				---- 1111	49, 125	
TRISB	PORTB Data Direction Control Register								1111 1111	49, 90	
TRISA	TRISA7 <sup>(4)</sup>	TRISA6 <sup>(4)</sup>	PORTA Data Direction Control Register						1111 1111	49, 87	
PDC0L	PWM Duty Cycle #0L Register (lower 8 bits)								0000 0000	49, 131	
PDC0H	—	—	PWM Duty Cycle #0H Register (upper 6 bits)				—	—	--00 0000	49, 131	
PDC1L	PWM Duty Cycle #1L Register (lower 8 bits)								0000 0000	49, 131	
PDC1H	—	—	PWM Duty Cycle #1H Register (upper 6 bits)				—	—	--00 0000	49, 131	
PDC2L	PWM Duty Cycle #2L Register (lower 8 bits)								0000 0000	49, 131	
PDC2H	—	—	PWM Duty Cycle #2H Register (upper 6 bits)				—	—	--00 0000	49, 131	
FLTCONFIG	BRFEN	—	—	—	—	—	FLTAS	FLTAMOD	FLTAEN	0--- -000	49, 143
LATB	PORTB Output Latch Register (Read and Write to Data Latch)								xxxx xxxx	49, 90	
LATA	LATA7 <sup>(4)</sup>	LATA6 <sup>(4)</sup>	PORTA Output Latch Register (Read and Write to Data Latch)						xxxx xxxx	49, 87	
SEVTCMPL	PWM Special Event Compare Register (lower 8 bits)								0000 0000	49, 144	
SEVTCMPH	—	—	—	—	PWM Special Event Compare Register (upper 4 bits)				---- 0000	50, 144	
PWMCON0	—	PWMEN2 <sup>(6)</sup>	PWMEN1 <sup>(6)</sup>	PWMEN0 <sup>(6)</sup>	—	PMOD2	PMOD1	PMOD0	-100 -000 -000 -000	50, 123	
PWMCON1	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC	0000 0-00	50, 124	
DTCON	DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0	0000 0000	50, 136	
OVDCOND	—	—	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0	--11 1111	50, 140	
OVDCONS	—	—	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0	--00 0000	50, 140	
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	50, 90	
PORTA	RA7 <sup>(4)</sup>	RA6 <sup>(4)</sup>	RA5 <sup>(3)</sup>	RA4	RA3	RA2	RA1	RA0	xx0x xxxx	50, 87	

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**2:** The PLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**3:** The RA5 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RA5 reads as '0'. This bit is read-only.

**4:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**5:** Bit 7 and bit 6 are cleared by user software or by a POR.

**6:** Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.

**7:** This bit has no effect if the Configuration bit, WDTEN, is enabled.

# PIC18F1230/1330

## 6.3.5 STATUS REGISTER

The STATUS register, shown in Register 6-2, contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the STATUS register is updated according to the instruction performed. Therefore, the result of an instruction with the STATUS register as its destination may be different than intended. As an example, `CLRF STATUS` will set the Z bit and leave the remaining Status bits unchanged ('000u u1uu').

It is recommended 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 in the STATUS register.

For other instructions that do not affect Status bits, see the instruction set summaries in Table 22-2 and Table 22-3.

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

### REGISTER 6-2: 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 <sup>(1)</sup>	C <sup>(2)</sup>
bit 7							bit 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

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 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 of the result) 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<sup>(1)</sup>  
 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

bit 0 **C:** Carry/borrow bit<sup>(2)</sup>  
 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 1:** 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 bit 4 or bit 3 of the source register.
- Note 2:** 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.



## 6.4 Data Addressing Modes

**Note:** The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 6.5 “Data Memory and the Extended Instruction Set”** for more information.

The data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in **Section 6.5.1 “Indexed Addressing with Literal Offset”**.

### 6.4.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include `SLEEP`, `RESET` and `DAW`.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include `ADDLW` and `MOVLW`, which respectively, add or move a literal value to the W register. Other examples include `CALL` and `GOTO`, which include a 20-bit program memory address.

### 6.4.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 6.3.3 “General Purpose Register File”**) or a location in the Access Bank (**Section 6.3.2 “Access Bank”**) as the data source for the instruction.

The Access RAM bit ‘a’ determines how the address is interpreted. When ‘a’ is ‘1’, the contents of the BSR (**Section 6.3.1 “Bank Select Register (BSR)”**) are used with the address to determine the complete 12-bit address of the register. When ‘a’ is ‘0’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as `MOVFF`, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation’s results is determined by the destination bit ‘d’. When ‘d’ is ‘1’, the results are stored back in the source register, overwriting its original contents. When ‘d’ is ‘0’, the results are stored in the W register. Instructions without the ‘d’ argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

### 6.4.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as Special Function Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 6-5.

#### EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 0) USING INDIRECT ADDRESSING

```

NEXT    LFSR    FSR0, 00h    ;
        CLRF   POSTINC0    ; Clear INDF
        ; register then
        ; inc pointer
        BTFSS  FSR0H, 0    ; All done with
        ; Bank0?
        BRA    NEXT        ; NO, clear next
CONTINUE        ; YES, continue
    
```

# PIC18F1230/1330

## 6.4.3.1 FSR Registers and the INDF Operand

At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. The four upper bits of the FSRnH register are not used so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect Addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as “virtual” registers: they are mapped in the SFR space but are not physically implemented. Reading or writing to a particular INDF register actually accesses its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L. Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction’s target. The INDF operand is just a convenient way of using the pointer.

Because Indirect Addressing uses a full 12-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

## 6.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action on its stored value. They are:

- POSTDEC: accesses the FSR value, then automatically decrements it by 1 afterwards
- POSTINC: accesses the FSR value, then automatically increments it by 1 afterwards
- PREINC: increments the FSR value by 1, then uses it in the operation
- PLUSW: adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the new value in the operation.

In this context, accessing an INDF register uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value offset by that in the W register; neither value is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR registers.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, roll-overs of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

**FIGURE 6-7: INDIRECT ADDRESSING**



The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

### 6.4.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains FE7h, the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a NOP.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses Indirect Addressing.

Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

## 6.5 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect Addressing with FSR0 and FSR1 also remains unchanged.

### 6.5.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced ('a' = 0); and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in Direct Addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

### 6.5.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is '1'), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in Figure 6-8.

Those who desire to use bit-oriented or byte-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 22.2.1 “Extended Instruction Syntax”**.

# PIC18F1230/1330

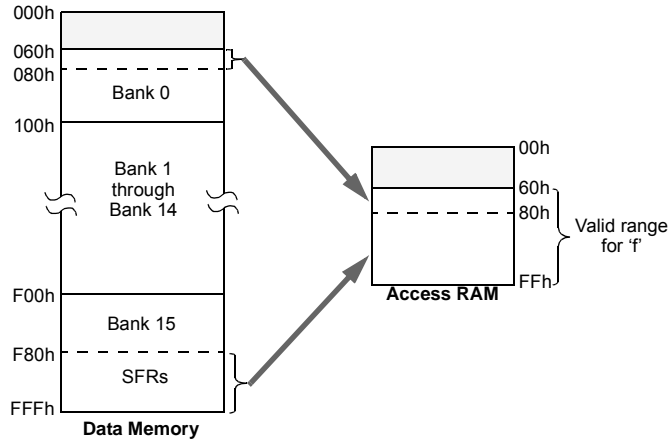
**FIGURE 6-8: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

**EXAMPLE INSTRUCTION:** `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

**When 'a' = 0 and f ≥ 60h:**

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations 060h to 07Fh (Bank 0) and F80h to FFFh (Bank 15) of data memory.

Locations below 60h are not available in this addressing mode.

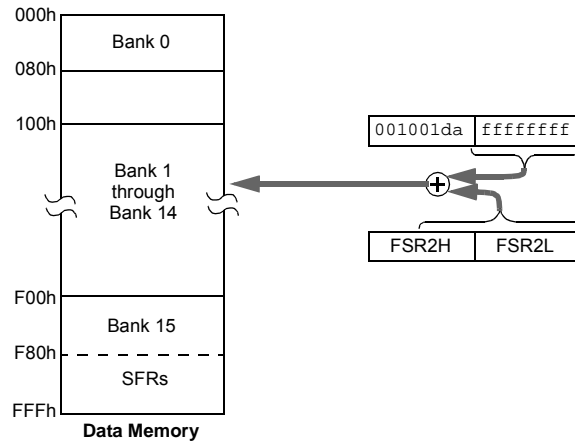


**When 'a' = 0 and f ≤ 5Fh:**

The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

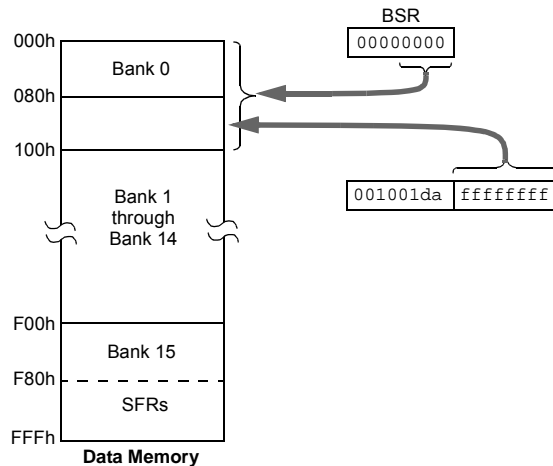
Note that in this mode, the correct syntax is now:

```
ADDWF [k], d
where 'k' is the same as 'f'.
```



**When 'a' = 1 (all values of f):**

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



## 6.5.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET ADDRESSING MODE

The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom half of Bank 0, this mode maps the contents from Bank 0 and a user-defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 6.3.2 “Access Bank”**). An example of Access Bank remapping in this addressing mode is shown in Figure 6-9.

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset Addressing mode. Operations that use the BSR (Access RAM bit is ‘1’) will continue to use Direct Addressing as before.

## 6.6 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds eight additional commands to the existing PIC18 instruction set. These instructions are executed as described in **Section 22.2 “Extended Instruction Set”**.

**FIGURE 6-9: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING MODE**



# PIC18F1230/1330

---

NOTES:

## 7.0 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.

## 7.1 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)

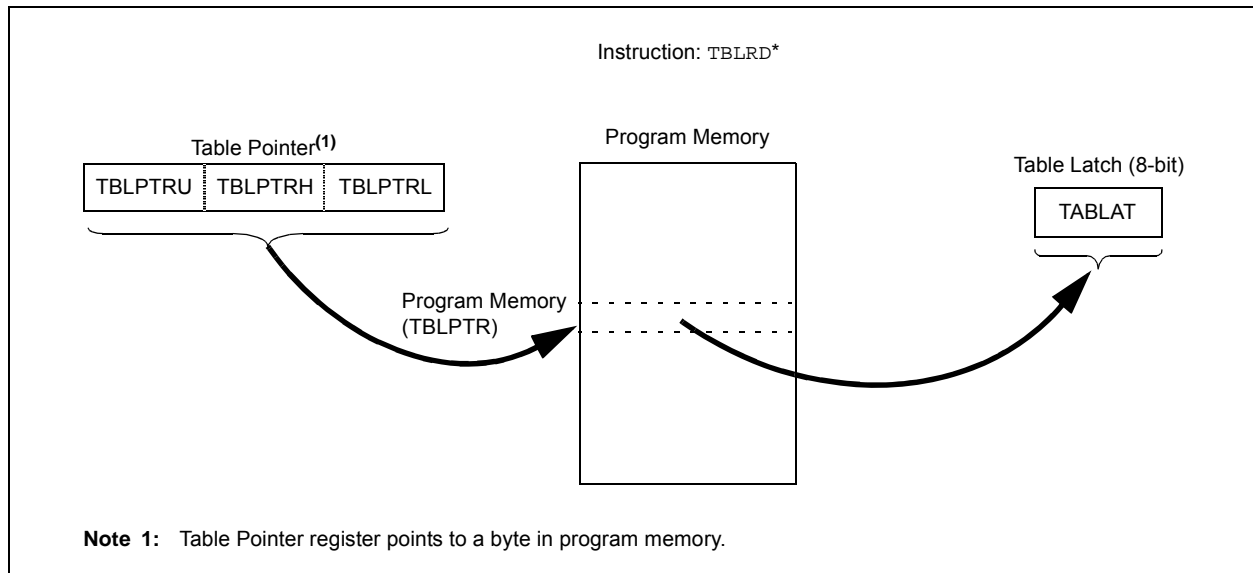
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 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 7.5 “Writing to Flash Program Memory”**. Figure 7-2 shows the operation of a table write with program memory and data RAM.

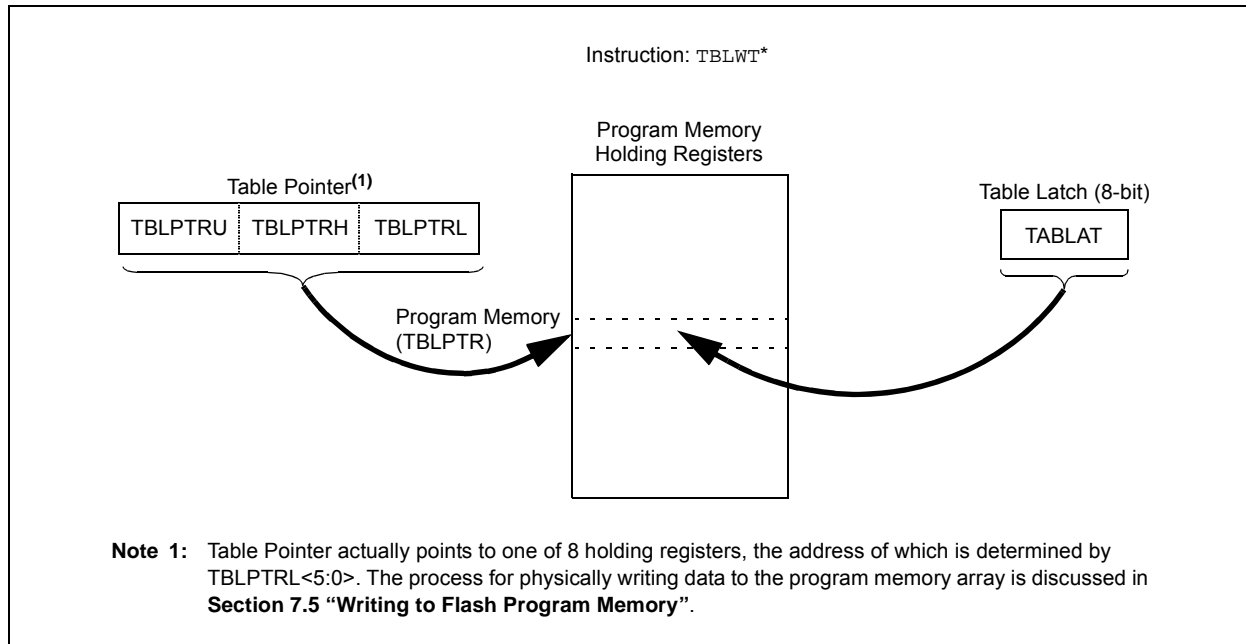
Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions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

**FIGURE 7-1: TABLE READ OPERATION**



# PIC18F1230/1330

FIGURE 7-2: TABLE WRITE OPERATION



## 7.2 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

### 7.2.1 EECON1 AND EECON2 REGISTERS

The EECON1 register (Register 7-1) is the control register for memory accesses. The EECON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading EECON2 will read all '0's.

The EEPGD control bit 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.

The CFGS control bit 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 20.0 "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 FREE is set, the erase operation is initiated on the next 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 in hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

**Note:** During normal operation, the WRERR may read as '1'. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly.

The WR control bit initiates write operations. The bit cannot be cleared, only set, in software; it is cleared in hardware at the completion of the write operation.

**Note:** The EEIF interrupt flag bit (PIR2<4>) is set when the write is complete. It must be cleared in software.



## REGISTER 7-1: 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 <sup>(1)</sup>	WREN	WR	RD
bit 7							bit 0

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

- bit 7      **EEPGD:** Flash Program or Data EEPROM Memory Select bit  
 1 = Access Flash program memory  
 0 = Access data EEPROM memory
- bit 6      **CFGS:** Flash Program/Data EEPROM or Configuration Select bit  
 1 = Access Configuration registers  
 0 = Access Flash program 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 WR command  
 (cleared by completion of erase operation)  
 0 = Perform write-only
- bit 3      **WRERR:** Flash Program/Data EEPROM Error Flag bit<sup>(1)</sup>  
 1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal  
 operation, or an improper write attempt)  
 0 = The write operation completed
- bit 2      **WREN:** Flash Program/Data EEPROM Write Enable bit  
 1 = Allows write cycles to Flash program/data EEPROM  
 0 = Inhibits write cycles to Flash program/data EEPROM
- bit 1      **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 WR bit can only be set (not cleared) in software.)  
 0 = Write cycle to the EEPROM is complete
- bit 0      **RD:** Read Control bit  
 1 = Initiates an EEPROM read. (Read takes one cycle. RD is cleared in hardware. The RD bit can only  
 be set (not cleared) in software. RD bit cannot be set when EEGD = 1 or CFGS = 1.)  
 0 = Does not initiate an EEPROM read

**Note 1:** When a WRERR occurs, the EEGD and CFGS bits are not cleared. This allows tracing of the error condition.

# PIC18F1230/1330

## 7.2.2 TABLAT – TABLE LATCH REGISTER

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

## 7.2.3 TBLPTR – TABLE POINTER REGISTER

The Table Pointer (TBLPTR) register 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 register, 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 7-1. These operations on the TBLPTR only affect the low-order 21 bits.

## 7.2.4 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 TBLPTR determine which byte is read from program memory into TABLAT.

When the timed write to program memory begins (via the WR bit), the 19 MSBs of the TBLPTR (TBLPTR<21:3>) determine which program memory block of 8 bytes is written to. The Table Pointer register's three LSBs (TBLPTR<2:0>) are ignored. For more detail, see **Section 7.5 “Writing to Flash Program Memory”**.

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

Figure 7-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

**TABLE 7-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS**

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

**FIGURE 7-3: TABLE POINTER BOUNDARIES BASED ON OPERATION**



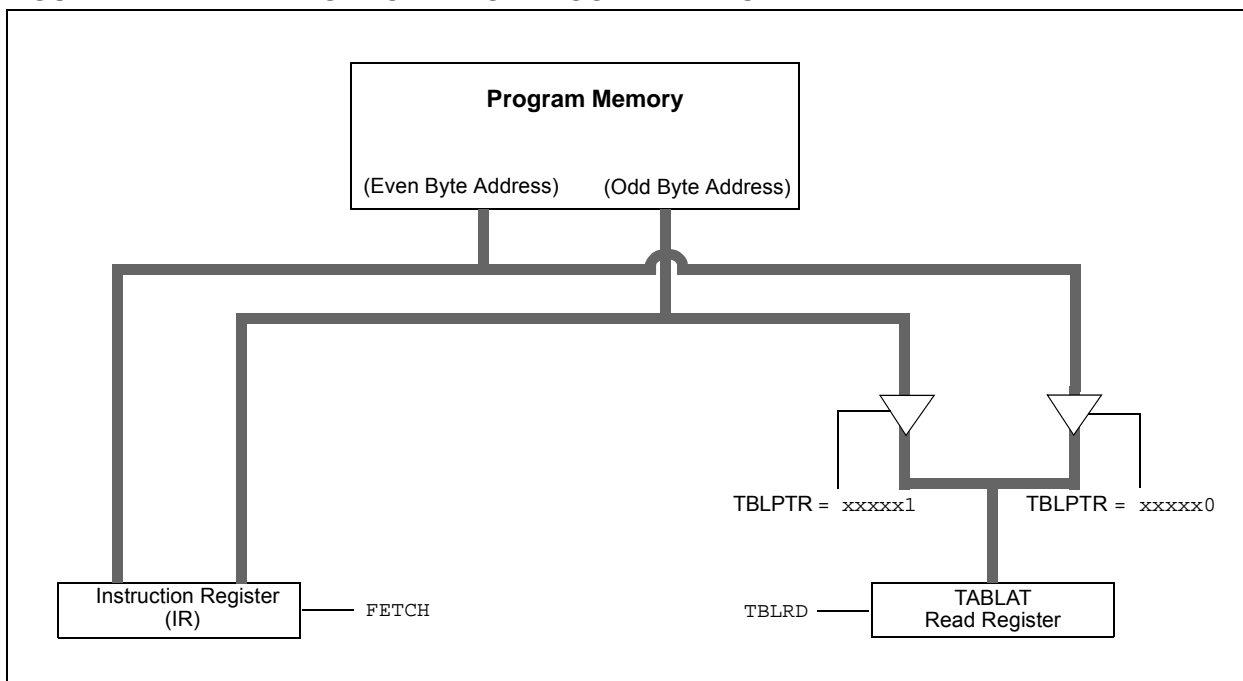
## 7.3 Reading the Flash Program Memory

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 7-4 shows the interface between the internal program memory and the `TABLAT`.

**FIGURE 7-4: READS FROM FLASH PROGRAM MEMORY**



**EXAMPLE 7-1: READING A FLASH PROGRAM MEMORY WORD**

```

MOV LW    CODE_ADDR_UPPER    ; Load TBLPTR with the base
MOV WF    TBLPTRU             ; address of the word
MOV LW    CODE_ADDR_HIGH
MOV WF    TBLPTRH
MOV LW    CODE_ADDR_LOW
MOV WF    TBLPTRL

READ_WORD
TBLRD*+   ; read into TABLAT and increment
MOV F    TABLAT, W           ; get data
MOV WF    WORD_EVEN

TBLRD*+   ; read into TABLAT and increment
MOV F    TABLAT, W           ; get data
MOV WF    WORD_ODD
    
```

# PIC18F1230/1330

## 7.4 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 microcontroller 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.

### 7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

1. Load Table Pointer register with address of row being erased.
2. Set the EECON1 register for the erase operation:
  - set EEPGD bit to point to program memory;
  - clear the CFGS bit to access program memory;
  - set WREN bit to enable writes;
  - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the 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 7-2: ERASING A FLASH PROGRAM MEMORY ROW

	MOVLW	CODE_ADDR_UPPER	; load TBLPTR with the base
	MOVWF	TBLPTRU	; address of the memory block
	MOVLW	CODE_ADDR_HIGH	
	MOVWF	TBLPTRH	
	MOVLW	CODE_ADDR_LOW	
	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
<b>Required Sequence</b>	MOVLW	55h	
	MOVWF	EECON2	; write 55h
	MOVLW	0AAh	
	MOVWF	EECON2	; write 0AAh
	BSF	EECON1, WR	; start erase (CPU stall)
	BSF	INTCON, GIE	; re-enable interrupts

## 7.5 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.

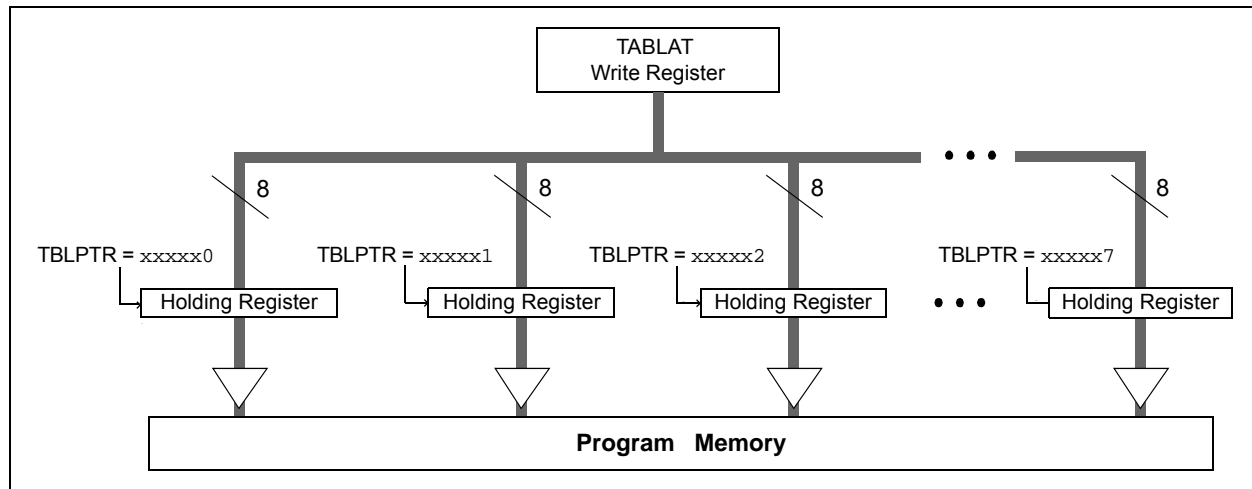
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 8 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the 8 holding registers,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

**Note:** The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all 8 holding registers before executing a write operation.

**FIGURE 7-5: TABLE WRITES TO FLASH PROGRAM MEMORY**



### 7.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

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

1. Read 8 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the row erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the 8 bytes into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
  - set EEPGD bit to point to program memory;
  - clear the CFGS bit to access program memory;
  - set WREN to enable byte writes.

8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the 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. Verify the memory (table read).

This procedure will require about 6 ms to update one row of 8 bytes of memory. An example of the required code is given in Example 7-3.

**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 register.

# PIC18F1230/1330

## EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY

```
        MOVLW    D'88                ; number of bytes in erase block
        MOVWF    COUNTER
        MOVLW    BUFFER_ADDR_HIGH    ; point to buffer
        MOVWF    FSR0H
        MOVLW    BUFFER_ADDR_LOW
        MOVWF    FSR0L
        MOVLW    CODE_ADDR_UPPER     ; Load TBLPTR with the base
        MOVWF    TBLPTRU             ; address of the memory block
        MOVLW    CODE_ADDR_HIGH
        MOVWF    TBLPTRH
        MOVLW    CODE_ADDR_LOW
        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    CODE_ADDR_UPPER     ; load TBLPTR with the base
        MOVWF    TBLPTRU             ; address of the memory block
        MOVLW    CODE_ADDR_HIGH
        MOVWF    TBLPTRH
        MOVLW    CODE_ADDR_LOW
        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

Required
Sequence
        MOVLW    55h                ; write 55h
        MOVWF    EECON2
        MOVLW    0AAh
        MOVWF    EECON2             ; write 0AAh
        BSF     EECON1, WR          ; start erase (CPU stall)
        BSF     INTCON, GIE         ; re-enable interrupts
        TBLRD*-                       ; dummy read decrement
        MOVLW    BUFFER_ADDR_HIGH    ; point to buffer
        MOVWF    FSR0H
        MOVLW    BUFFER_ADDR_LOW
        MOVWF    FSR0L

WRITE_BUFFER_BACK
        MOVLW    D'8                ; number of bytes in holding register
        MOVWF    COUNTER

WRITE_BYTE_TO_HREGS
        MOVFF    POSTINC0, WREG      ; 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
```

## EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)

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	
<b>Required Sequence</b>		MOVWF	EECON2	; write 55h
		MOVLW	0AAh	
		MOVWF	EECON2	; write 0AAh
		BSF	EECON1, WR	; start program (CPU stall)
		BSF	INTCON, GIE	; re-enable interrupts
		BCF	EECON1, WREN	; disable write to memory

### 7.5.2 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.

### 7.5.3 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. If the write operation is interrupted by a MCLR Reset or a WDT Time-out Reset during normal operation, the user can check the WRERR bit and rewrite the location(s) as needed.

### 7.5.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 20.0 “Special Features of the CPU”** for more detail.

## 7.6 Flash Program Operation During Code Protection

See **Section 20.5 “Program Verification and Code Protection”** for details on code protection of Flash program memory.

**TABLE 7-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					47
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								47
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								47
TABLAT	Program Memory Table Latch								47
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
EECON2	EEPROM Control Register 2 (not a physical register)								49
EECON1	EEPGD	CFGFS	—	FREE	WRERR	WREN	WR	RD	49
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	49
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	49
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	49

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during Flash/EEPROM access.

# PIC18F1230/1330

---

NOTES:



## 8.0 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. These devices have 128 bytes of data EEPROM with an address range from 00h to FFh.

The EEPROM data memory is rated for high erase/write cycle endurance. 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 parameter D122 (Table in **Section 23.0 “Electrical Characteristics”**) for exact limits.

### 8.1 EEADR Register

The EEPROM Address register can address 256 bytes of data EEPROM.

## 8.2 EECON1 and EECON2 Registers

Access to the data EEPROM is controlled by two registers: EECON1 and EECON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The EECON1 register (Register 7-1) is the control register for data and program memory access. Control bit EEPGD determines if the access will be to program or data EEPROM memory. When clear, operations will access the data EEPROM memory. When set, program memory is accessed.

Control bit, CFGS, determines if the access will be to the Configuration registers or to program memory/data EEPROM memory. When set, subsequent operations access Configuration registers. When CFGS is clear, the EEPGD bit selects either program Flash or data EEPROM memory.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set in hardware when the WREN bit is set and cleared when the internal programming timer expires and the write operation is complete.

**Note 1:** During normal operation, the WRERR bit is read as ‘1’. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly. The WR control bit initiates write operations. The bit cannot be cleared, only set, in software; it is cleared in hardware at the completion of the write operation.

**2:** The Interrupt Flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in the software Control bits RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

The RD bit cannot be set when accessing program memory (EEPGD = 1). Program memory is read using table read instructions. See **Section 7.1 “Table Reads and Table Writes”** regarding table reads.

**Note:** The EECON2 register is not a physical register. It is used exclusively in the memory write and erase sequences. Reading EECON2 will read all ‘0’s.

# PIC18F1230/1330

## REGISTER 8-1: 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 <sup>(1)</sup>	WREN	WR	RD
bit 7							bit 0

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

- bit 7      **EEPGD:** Flash Program or Data EEPROM Memory Select bit  
 1 = Access Flash program memory  
 0 = Access data EEPROM memory
- bit 6      **CFGS:** Flash Program/Data EEPROM or Configuration Select bit  
 1 = Access Configuration registers  
 0 = Access Flash program 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 WR command  
 (cleared by completion of erase operation)  
 0 = Perform write-only
- bit 3      **WRERR:** EEPROM Error Flag bit<sup>(1)</sup>  
 1 = A write operation is prematurely terminated  
 (MCLR or WDT Reset during self-timed erase or program operation)  
 0 = The write operation completed
- bit 2      **WREN:** Erase/Write Enable bit  
 1 = Allows erase/write cycles  
 0 = Inhibits erase/write cycles
- bit 1      **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 WR bit can only be set (not cleared) in software.)  
 0 = Write cycle to is completed
- bit 0      **RD:** Read Control bit  
 1 = Initiates a memory read. (Read takes one cycle. RD is cleared in hardware. The RD bit can only be  
 set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)  
 0 = Read completed

**Note 1:** When a WRERR occurs, the EEPGD or FREE bit is not cleared. This allows tracing of the error condition.

## 8.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD control bit (EECON1<7>) and then set control bit RD (EECON1<0>). The data is available for the very next instruction cycle; therefore, the EEDATA register 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).

## 8.4 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. The sequence in Example 8-2 must be followed to initiate the write cycle.

The write will not begin if this sequence is not exactly followed (write 55h to EECON2, write 0AAh to EECON2, then set 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, EECON1, EEADR and EEDATA cannot be modified. The 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 WR and WREN cannot be set with the same instruction.

At the completion of the write cycle, the WR bit is cleared in hardware and the EEPROM Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt or poll this bit. EEIF must be cleared by software.

## 8.5 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.

## 8.6 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 help prevent an accidental write during brown-out, power glitch, or software malfunction.

### EXAMPLE 8-1: DATA EEPROM READ

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

### EXAMPLE 8-2: DATA EEPROM WRITE

	MOVLW	DATA_EE_ADDR	;
	MOVWF	EEADR	; Data Memory Address to write
	MOVLW	DATA_EE_DATA	;
	MOVWF	EEDATA	; Data Memory Value to write
	BCF	EECON1, EEPGD	; Point to DATA memory
	BSF	EECON1, WREN	; Enable writes
	BCF	INTCON, GIE	; Disable Interrupts
	MOVLW	55h	;
Required	MOVWF	EECON2	; Write 55h
Sequence	MOVLW	0AAh	;
	MOVWF	EECON2	; Write 0AAh
	BSF	EECON1, WR	; Set WR bit to begin write
	BSF	INTCON, GIE	; Enable Interrupts
	BTFSC	EECON1, WR	; Wait for write to complete
	BRA	\$-2	
	SLEEP		; Wait for interrupt to signal write complete
	BCF	EECON1, WREN	; Disable writes

# PIC18F1230/1330

## 8.7 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in Configuration Words. 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 20.0 “Special Features of the CPU”** for additional information.

## 8.8 Using the Data EEPROM

The data EEPROM is a high-endurance, byte addressable 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. 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 8-3.

**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.

### EXAMPLE 8-3: 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
Required	MOVLW	55h		;
Sequence	MOVWF	EECON2		; Write 55h
	MOVLW	0AAh		;
	MOVWF	EECON2		; Write 0AAh
	BSF	EECON1, WR		; Set WR bit to begin write
	BTFSC	EECON1, WR		; Wait for write to complete
	BRA	\$-2		
	INCF	EEADR, F		; Increment address
	BRA	LOOP		; Not zero, do it again
	BCF	EECON1, WREN		; Disable writes
	BSF	INTCON, GIE		; Enable interrupts

**TABLE 8-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
EEADR	EEPROM Address Register								49
EEDATA	EEPROM Data Register								49
EECON2	EEPROM Control Register 2 (not a physical register)								49
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	49
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	49
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	49
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	49

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during Flash/EEPROM access.

## 9.0 8 x 8 HARDWARE MULTIPLIER

### 9.1 Introduction

All PIC18 devices include an 8 x 8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the Product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 9-1.

### 9.2 Operation

Example 9-1 shows the instruction sequence for an 8 x 8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 9-2 shows the sequence to do an 8 x 8 signed multiplication. 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 9-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

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

#### EXAMPLE 9-2: 8 x 8 SIGNED MULTIPLY ROUTINE

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

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

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

**TABLE 9-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

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 $\mu$ s	27.6 $\mu$ s	69 $\mu$ s
	Hardware multiply	1	1	100 ns	400 ns	1 $\mu$ s
8 x 8 signed	Without hardware multiply	33	91	9.1 $\mu$ s	36.4 $\mu$ s	91 $\mu$ s
	Hardware multiply	6	6	600 ns	2.4 $\mu$ s	6 $\mu$ s
16 x 16 unsigned	Without hardware multiply	21	242	24.2 $\mu$ s	96.8 $\mu$ s	242 $\mu$ s
	Hardware multiply	28	28	2.8 $\mu$ s	11.2 $\mu$ s	28 $\mu$ s
16 x 16 signed	Without hardware multiply	52	254	25.4 $\mu$ s	102.6 $\mu$ s	254 $\mu$ s
	Hardware multiply	35	40	4.0 $\mu$ s	16.0 $\mu$ s	40 $\mu$ s

# PIC18F1230/1330

Example 9-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 9-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

## EQUATION 9-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

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

## EXAMPLE 9-3: 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, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;

MOVF ARG1H, W    ;
MULWF ARG2L      ; ARG1H * ARG2L->
                  ; PRODH:PRODL

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

```

Example 9-4 shows the sequence to do a 16 x 16 signed multiply. Equation 9-2 shows the algorithm used. The 32-bit result is stored in four registers (RES3:RES0). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

## EQUATION 9-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

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

## EXAMPLE 9-4: 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, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;

MOVF ARG1H, W    ;
MULWF ARG2L      ; ARG1H * ARG2L ->
                  ; PRODH:PRODL

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

BTFSS ARG2H, 7   ; ARG2H:ARG2L neg?
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
:

```



# PIC18F1230/1330

**TABLE 10-1: PORTA I/O SUMMARY**

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RA0/AN0/INT0/ KBI0/CMP0	RA0	0	O	DIG	LATA<0> data output; not affected by analog input.
		1	I	TTL	PORTA<0> data input; disabled when analog input enabled.
	AN0	1	I	ANA	Analog input 0.
	INT0	1	I	ST	External interrupt 0.
	KBI0	1	I	TTL	Interrupt-on-change pin.
RA1/AN1/INT1/ KBI1	RA1	0	O	DIG	LATA<1> data output; not affected by analog input.
		1	I	TTL	PORTA<1> data input; disabled when analog input enabled.
	AN1	1	I	ANA	Analog input 1.
	INT1	1	I	ST	External interrupt 1.
	KBI1	1	I	TTL	Interrupt-on-change pin.
RA2/TX/CK	RA2	0	O	DIG	LATA<2> data output; not affected by analog input. Disabled when CVREF output enabled.
		1	I	TTL	PORTA<2> data input. Disabled when analog functions enabled; disabled when CVREF output enabled.
	TX	0	O	DIG	EUSART asynchronous transmit.
	CK	0	O	DIG	EUSART synchronous clock.
		1	I	ST	
RA3/RX/DT	RA3	0	O	DIG	LATA<3> data output; not affected by analog input.
		1	I	TTL	PORTA<3> data input; disabled when analog input enabled.
	RX	1	I	ANA	EUSART asynchronous receive.
	DT	0	O	DIG	EUSART synchronous data.
		1	I	TTL	
RA4/T0CKI/AN2/ VREF+	RA4	0	O	DIG	LATA<4> data output.
		1	I	ST	PORTA<4> data input; default configuration on POR.
	T0CKI	1	I	ST	Timer0 external clock input.
	AN2	1	I	ANA	Analog input 2.
	VREF+	1	I	ANA	A/D reference voltage (high) input.
MCLR/VPP/RA5/ FLTA	MCLR	1	I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
	VPP	1	I	ANA	Programming voltage input.
	RA5	1	I	ST	Digital input.
	FLTA <sup>(1)</sup>	1	I	ST	Fault detect input for PWM.
RA6/OSC2/CLKO/ T1OSO/T1CKI/AN3	RA6	0	O	DIG	LATA<6> data output. Enabled in RCIO, INTIO2 and ECIO modes only.
		1	I	ST	PORTA<6> data input. Enabled in RCIO, INTIO2 and ECIO modes only.
	OSC2	0	O	ANA	Oscillator crystal output or external clock source output.
	CLKO	0	O	ANA	Oscillator crystal output.
	T1OSO <sup>(2)</sup>	0	O	ANA	Timer1 oscillator output.
	T1CKI <sup>(2)</sup>	1	I	ST	Timer1 clock input.
	AN3	1	I	ANA	Analog input 3.
RA7/OSC1/CLKI/ T1OSI/FLTA	RA7	0	O	DIG	LATA<7> data output. Disabled in external oscillator modes.
		1	I	TTL	PORTA<7> data input. Disabled in external oscillator modes.
	OSC1	1	I	ANA	Oscillator crystal input or external clock source input.
	CLKI	1	I	ANA	External clock source input.
	T1OSI <sup>(2)</sup>	1	I	ANA	Timer1 oscillator input.
	FLTA <sup>(1)</sup>	1	I	ST	Fault detect input for PWM.

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Placement of FLTA depends on the value of Configuration bit, FLTAMX, of CONFIG3H.

**Note 2:** Placement of T1OSI and T1OSO/T1CKI depends on the value of Configuration bit, T1OSCMX, of CONFIG3H.



**TABLE 10-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	RA4	RA3	RA2	RA1	RA0	50
LATA	LATA7 <sup>(1)</sup>	LATA6 <sup>(1)</sup>	PORTA Output Latch Register (Read and Write to Data Latch)						49
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Control Register						49
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	47
ADCON1	—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	48
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	48
CVRCON	CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	48

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

**Note 1:** RA7:RA6 and their associated latch and data direction bits are enabled as I/O pins based on oscillator configuration; otherwise, they are read as '0'.

# PIC18F1230/1330

## 10.2 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).

The Output Latch register (LATB) is also memory mapped. Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

### EXAMPLE 10-2: INITIALIZING PORTB

```
CLRF   PORTB   ; Initialize PORTB by
              ; clearing output
              ; data latches
CLRF   LATB    ; Alternate method
              ; to clear output
              ; data latches
MOVLW  0Fh    ; Set RB<4:0> as
MOVWF  ADCON1 ; digital I/O pins
              ; (required if config bit
              ; PBDEN is set)
MOVLW  0CFh   ; Value used to
              ; initialize data
              ; direction
MOVWF  TRISB  ; Set RB<3:0> as inputs
              ; RB<5:4> as outputs
              ; RB<7:6> 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<7>). 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.

**Note:** On a Power-on Reset, PORTB is configured as digital inputs except for RB2 and RB3.

RB2 and RB3 are configured as analog inputs when the T1OSCMX bit of Configuration Register 3H is cleared. Otherwise, RB2 and RB3 are also configured as digital inputs.

Pins RB0, RB1 and RB4:RB7 are multiplexed with the Power Control PWM outputs.

Pins RB2 and RB3 are multiplexed with external interrupt inputs, interrupt-on-change input, the analog comparator inputs and the Timer1 oscillator input and output to become RB2/INT2/KBI2/CMP2/T1OSO/T1CKI and RB3/INT3/KNBI3/CMP1/T1OSI, respectively.

When the interrupt-on-change feature is enabled, only pins configured as inputs can cause this interrupt to occur (i.e., any RB2, RB3, RA0 and RA1 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (RB2, RB3, RA0 and RA1) are compared with the old value latched on the last read of PORTA and PORTB. The “mismatch” outputs of these pins are ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from Sleep mode, or any of the Idle modes. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- Any read or write of PORTB (except with the MOVFF (ANY), PORTB instruction).
- 1 T<sub>CY</sub>
- Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB and waiting 1 T<sub>CY</sub> will end the mismatch condition and allow flag bit, RBIF, to be cleared. Additionally, if the port pin returns to its original state, the mismatch condition will be cleared.

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

**TABLE 10-3: PORTB I/O SUMMARY**

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB0/PWM0	RB0	0	O	DIG	LATB<0> data output; not affected by analog input.
		1	I	TTL	PORTB<0> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	PWM0	0	O	DIG	PWM module output PWM0.
RB1PWM1	RB1	0	O	DIG	LATB<1> data output; not affected by analog input.
		1	I	TTL	PORTB<1> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	PWM1	0	O	DIG	PWM module output PWM1.
RB2/INT2/KBI2/ CMP2/T1OSO/ T1CKI	RB2	0	O	DIG	LATB<2> data output; not affected by analog input.
		1	I	TTL	PORTB<2> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	INT2	1	I	ST	External interrupt 2 input.
	KBI2	1	I	TTL	Interrupt-on-change pin.
	CMP2	1	I	ANA	Comparator 2 input.
	T1OSO <sup>(2)</sup>	0	O	ANA	Timer1 oscillator output.
	T1CKI <sup>(2)</sup>	1	I	ST	Timer1 clock input.
RB3/INT3/KBI3/ CMP1/T1OSI	RB3	0	O	DIG	LATB<3> data output; not affected by analog input.
		1	I	TTL	PORTB<3> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	INT3	1	I	ST	External interrupt 3 input.
	KBI3	1	I	TTL	Interrupt-on-change pin.
	CMP1	1	I	ANA	Comparator 1 input.
	T1OSI <sup>(2)</sup>	1	I	ANA	Timer1 oscillator input.
RB4/PWM2	RB4	0	O	DIG	LATB<4> data output; not affected by analog input.
		1	I	TTL	PORTB<4> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	PWM2	0	O	DIG	PWM module output PWM2.
RB5/PWM3	RB5	0	O	DIG	LATB<5> data output.
		1	I	TTL	PORTB<5> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	PWM3	0	O	DIG	PWM module output PWM3.
RB6/PWM4/PGC	RB6	0	O	DIG	LATB<6> data output.
		1	I	TTL	PORTB<6> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	PWM4	0	O	DIG	PWM module output PWM4.
	PGC	1	I	ST	In-Circuit Debugger and ICSP™ programming clock pin.
RB7/PWM5/PGD	RB7	0	O	DIG	LATB<7> data output.
		1	I	TTL	PORTB<7> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	PWM5	0	O	TTL	PWM module output PWM4.
	PGD	0	O	DIG	In-Circuit Debugger and ICSP programming data pin.

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Configuration on POR is determined by the PBAEN Configuration bit. Pins are configured as analog inputs by default when PBAEN is set and digital inputs when PBAEN is cleared.

**2:** Placement of T1OSI and T1OSO/T1CKI depends on the value of Configuration bit, T1OSCMX, of CONFIG3H.

# PIC18F1230/1330

**TABLE 10-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	50
LATB	PORTB Output Latch Register (Read and Write to Data Latch)								49
TRISB	PORTB Data Direction Control Register								49
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
INTCON2	$\overline{\text{RBP}}\text{U}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	47
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	47
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	48

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

## 11.0 INTERRUPTS

The PIC18F1230/1330 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. High-priority interrupt events will interrupt any low-priority interrupts that may be in progress.

There are thirteen registers which 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.

In general, interrupt sources have three bits to control their operation. They 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<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 0008h or 0018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit, which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit, which enables/disables all interrupt sources. All interrupts branch to address 0008h 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. Low-priority interrupts are not processed while high-priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). 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.

<b>Note:</b> Do not use the <code>MOVFF</code> instruction to modify any of the interrupt control registers while <b>any</b> interrupt is enabled. Doing so may cause erratic microcontroller behavior.
---

# PIC18F1230/1330

FIGURE 11-1: PIC18 INTERRUPT LOGIC



## 11.1 INTCON Registers

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

**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 for software polling.

### REGISTER 11-1: 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 <sup>(1)</sup>
bit 7							bit 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

- bit 7      **GIE/GIEH:** Global Interrupt Enable bit  
When IPEN = 0:  
 1 = Enables all unmasked interrupts  
 0 = Disables all interrupts  
When IPEN = 1:  
 1 = Enables all high-priority interrupts  
 0 = Disables all interrupts
  
- bit 6      **PEIE/GIEL:** Peripheral Interrupt Enable bit  
When IPEN = 0:  
 1 = Enables all unmasked peripheral interrupts  
 0 = Disables all peripheral interrupts  
When IPEN = 1:  
 1 = Enables all low-priority peripheral interrupts  
 0 = Disables all 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)  
 0 = The INT0 external interrupt did not occur
  
- bit 0      **RBIF:** RB Port Change Interrupt Flag bit<sup>(1)</sup>  
 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 1:** A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

# PIC18F1230/1330

## REGISTER 11-2: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBP}}\text{U}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
bit 7							bit 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

- bit 7  **$\overline{\text{RBP}}\text{U}$** : PORTB Pull-up Enable bit  
1 = All PORTB pull-ups are disabled  
0 = PORTB pull-ups are enabled by individual port latch values
- bit 6 **INTEDG0**: External Interrupt 0 Edge Select bit  
1 = Interrupt on rising edge  
0 = Interrupt on falling edge
- bit 5 **INTEDG1**: External Interrupt 1 Edge Select bit  
1 = Interrupt on rising edge  
0 = Interrupt on falling edge
- bit 4 **INTEDG2**: External Interrupt 2 Edge Select bit  
1 = Interrupt on rising edge  
0 = Interrupt on falling edge
- bit 3 **INTEDG3**: External Interrupt 3 Edge Select bit  
1 = Interrupt on rising edge  
0 = Interrupt on falling edge
- bit 2 **TMR0IP**: TMR0 Overflow Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 1 **INT3IP**: INT3 External Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 0 **RBIP**: RB Port Change Interrupt Priority bit  
1 = High priority  
0 = Low priority

**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 for software polling.



## REGISTER 11-3: INTCON3: INTERRUPT CONTROL REGISTER 3

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
bit 7							bit 0

### 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

- bit 7      **INT2IP:** INT2 External Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 6      **INT1IP:** INT1 External Interrupt Priority bit  
           1 = High priority  
           0 = Low priority
- bit 5      **INT3IE:** INT3 External Interrupt Enable bit  
           1 = Enables the INT3 external interrupt  
           0 = Disables the INT3 external interrupt
- bit 4      **INT2IE:** INT2 External Interrupt Enable bit  
           1 = Enables the INT2 external interrupt  
           0 = Disables the INT2 external interrupt
- bit 3      **INT1IE:** INT1 External Interrupt Enable bit  
           1 = Enables the INT1 external interrupt  
           0 = Disables the INT1 external interrupt
- bit 2      **INT3IF:** INT3 External Interrupt Flag bit  
           1 = The INT3 external interrupt occurred (must be cleared in software)  
           0 = The INT3 external interrupt did not occur
- bit 1      **INT2IF:** INT2 External Interrupt Flag bit  
           1 = The INT2 external interrupt occurred (must be cleared in software)  
           0 = The INT2 external interrupt did not occur
- bit 0      **INT1IF:** INT1 External Interrupt Flag bit  
           1 = The INT1 external interrupt occurred (must be cleared in software)  
           0 = The INT1 external interrupt did not occur

**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 for software polling.

# PIC18F1230/1330

## 11.2 PIR Registers

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

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

### REGISTER 11-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF
bit 7							bit 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

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

bit 6 **ADIF:** A/D Converter Interrupt Flag bit

1 = An A/D conversion completed (must be cleared in software)

0 = The A/D conversion is not complete

bit 5 **RCIF:** EUSART Receive Interrupt Flag bit

1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)

0 = The EUSART receive buffer is empty

bit 4 **TXIF:** EUSART Transmit Interrupt Flag bit

1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)

0 = The EUSART transmit buffer is full

bit 3 **CMP2IF:** Analog Comparator 2 Flag bit

1 = The output of CMP2 has changed since last read

0 = The output of CMP2 has not changed since last read

bit 2 **CMP1IF:** Analog Comparator 1 Flag bit

1 = The output of CMP1 has changed since last read

0 = The output of CMP1 has not changed since last read

bit 1 **CMP0IF:** Analog Comparator 0 Flag bit

1 = The output of CMP0 has changed since last read

0 = The output of CMP0 has not changed since last read

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit

1 = TMR1 register overflowed (must be cleared in software)

0 = TMR1 register did not overflow

## REGISTER 11-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

R/W-0	U-0	U-0	R/W-0	U-0	R/W-0	U-0	U-0
OSCFIF	—	—	EEIF	—	LVDIF	—	—
bit 7							bit 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

- bit 7      **OSCFIF:** Oscillator Fail Interrupt Flag bit  
 1 = Device oscillator failed, clock input has changed to INTOSC (must be cleared in software)  
 0 = Device clock operating
- bit 6-5    **Unimplemented:** Read as '0'
- bit 4      **EEIF:** Data EEPROM/Flash Write Operation Interrupt Flag bit  
 1 = The write operation is complete (must be cleared in software)  
 0 = The write operation is not complete or has not been started
- bit 3      **Unimplemented:** Read as '0'
- bit 2      **LVDIF:** Low-Voltage Detect Interrupt Flag bit  
 1 = A low-voltage condition occurred  
 0 = A low-voltage condition has not occurred
- bit 1-0    **Unimplemented:** Read as '0'

## REGISTER 11-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3

U-0	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0
—	—	—	PTIF	—	—	—	—
bit 7							bit 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

- bit 7-5    **Unimplemented:** Read as '0'
- bit 4      **PTIF:** PWM Time Base Interrupt bit  
 1 = PWM time base matched the value in PTPER register. Interrupt is issued according to the postscaler settings. PTIF must be cleared in software.  
 0 = PWM time base has not matched the value in PTPER register
- bit 3-0    **Unimplemented:** Read as '0'

# PIC18F1230/1330

## 11.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2 and PIE3). When IPEN = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

### REGISTER 11-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE
bit 7							bit 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

- bit 7      **Unimplemented:** Read as '0'
- bit 6      **ADIE:** A/D Converter Interrupt Enable bit  
1 = Enables the A/D interrupt  
0 = Disables the A/D interrupt
- bit 5      **RCIE:** EUSART Receive Interrupt Enable bit  
1 = Enables the EUSART receive interrupt  
0 = Disables the EUSART receive interrupt
- bit 4      **TXIE:** EUSART Transmit Interrupt Enable bit  
1 = Enables the EUSART transmit interrupt  
0 = Disables the EUSART transmit interrupt
- bit 3      **CMP2IE:** Analog Comparator 2 Interrupt Enable bit  
1 = Enables the CMP2 interrupt  
0 = Disables the CMP2 interrupt
- bit 2      **CMP1IE:** Analog Comparator 1 Interrupt Enable bit  
1 = Enables the CMP1 interrupt  
0 = Disables the CMP1 interrupt
- bit 1      **CMP0IE:** Analog Comparator 0 Interrupt Enable bit  
1 = Enables the CMP0 interrupt  
0 = Disables the CMP0 interrupt
- bit 0      **TMR1IE:** TMR1 Overflow Interrupt Enable bit  
1 = Enables the TMR1 overflow interrupt  
0 = Disables the TMR1 overflow interrupt

## REGISTER 11-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

R/W-0	U-0	U-0	R/W-0	U-0	R/W-0	U-0	U-0
OSCFIE	—	—	EEIE	—	LVDIE	—	—
bit 7							bit 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

bit 7      **OSCFIE:** Oscillator Fail Interrupt Enable bit

1 = Enabled

0 = Disabled

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

bit 4      **EEIE:** Data EEPROM/Flash Write Operation Interrupt Enable bit

1 = Enabled

0 = Disabled

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

bit 2      **LVDIE:** Low-Voltage Detect Interrupt Enable bit

1 = Enabled

0 = Disabled

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

## REGISTER 11-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

U-0	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0
—	—	—	PTIE	—	—	—	—
bit 7							bit 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

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

bit 4      **PTIE:** PWM Time Base Interrupt Enable bit

1 = PWM enabled

0 = PWM disabled

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

# PIC18F1230/1330

## 11.4 IPR Registers

The 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). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

### REGISTER 11-10: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP
bit 7							bit 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

- bit 7      **Unimplemented:** Read as '0'
- bit 6      **ADIP:** A/D Converter Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 5      **RCIP:** EUSART Receive Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 4      **TXIP:** EUSART Transmit Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 3      **CMP2IP:** Analog Comparator 2 Interrupt Priority bit  
1 = CMP2 is high priority  
0 = CMP2 is low priority
- bit 2      **CMP1IP:** Analog Comparator 1 Interrupt Priority bit  
1 = CMP1 is high priority  
0 = CMP1 is low priority
- bit 1      **CMP0IP:** Analog Comparator 0 Interrupt Priority bit  
1 = CMP0 is high priority  
0 = CMP0 is low priority
- bit 0      **TMR1IP:** TMR1 Overflow Interrupt Priority bit  
1 = High priority  
0 = Low priority

## REGISTER 11-11: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

R/W-1	U-0	U-0	R/W-1	U-0	R/W-1	U-0	U-0
OSCFIP	—	—	EEIP	—	LVDIP	—	—
bit 7							bit 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

bit 7                      **OSCFIP:** Oscillator Fail Interrupt Priority bit

1 = High priority

0 = Low priority

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

bit 4                      **EEIP:** Data EEPROM/Flash Write Operation Interrupt Priority bit

1 = High priority

0 = Low priority

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

bit 2                      **LVDIP:** Low-Voltage Detect Interrupt Priority bit

1 = High priority

0 = Low priority

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

## REGISTER 11-12: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

U-0	U-0	U-0	R/W-1	U-0	U-0	U-0	U-0
—	—	—	PTIP	—	—	—	—
bit 7							bit 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

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

bit 4                      **PTIP:** PWM Time Base Interrupt Priority bit

1 = High priority

0 = Low priority

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

# PIC18F1230/1330

## 11.5 RCON Register

The RCON register contains flag bits which are used to determine the cause of the last Reset or wake-up from Idle or Sleep modes. RCON also contains the IPEN bit which enables interrupt priorities.

The operation of the SBOREN bit and the Reset flag bits is discussed in more detail in **Section 5.1 “RCON Register”**.

### REGISTER 11-13: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1 <sup>(1)</sup>	U-0	R/W-1	R-1	R-1	R/W-0 <sup>(2)</sup>	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 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

- bit 7      **IPEN:** Interrupt Priority Enable bit  
1 = Enable priority levels on interrupts  
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
- bit 6      **SBOREN:** BOR Software Enable bit<sup>(1)</sup>  
For details of bit operation, see Register 5-1.
- bit 5      **Unimplemented:** Read as '0'
- bit 4       **$\overline{RI}$ :** RESET Instruction Flag bit  
For details of bit operation, see Register 5-1.
- bit 3       **$\overline{TO}$ :** Watchdog Time-out Flag bit  
For details of bit operation, see Register 5-1.
- bit 2       **$\overline{PD}$ :** Power-Down Detection Flag bit  
For details of bit operation, see Register 5-1.
- bit 1       **$\overline{POR}$ :** Power-on Reset Status bit<sup>(2)</sup>  
For details of bit operation, see Register 5-1.
- bit 0       **$\overline{BOR}$ :** Brown-out Reset Status bit  
For details of bit operation, see Register 5-1.

- Note 1:** If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'. See Register 5-1 for additional information.
- Note 2:** The actual Reset value of  $\overline{POR}$  is determined by the type of device Reset. See Register 5-1 for additional information.



## 11.6 INTx Pin Interrupts

External interrupts on the RA0/INT0, RA1/INT1, RB2/INT2 and RB3/INT3 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the pin, the corresponding flag bit, INTxIF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxIE. Flag bit, INTxIF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1, INT2 and INT3) can wake-up the processor from Idle or Sleep modes if bit INTxIE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1, INT2 and INT3 is determined by the value contained in the interrupt priority bits, INT1IP (INTCON3<6>), INT2IP (INTCON3<7>) and INT3IP (INTCON2<1>). There is no priority bit associated with INT0. It is always a high-priority interrupt source.

## 11.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 12.0 “Timer0 Module”** for further details on the Timer0 module.

## 11.8 Interrupt-on-Change

An input change on PORTA<1:0> and/or PORTB<2:3> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<3>). Interrupt priority for interrupt-on-change is determined by the value contained in the interrupt priority bit, RBIP (INTCON2<0>).

## 11.9 Context Saving During Interrupts

During interrupts, the return PC address 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 6.3 “Data Memory Organization”**), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user’s application, other registers may also need to be saved. Example 11-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

### EXAMPLE 11-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```

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

# PIC18F1230/1330

---

NOTES:

## 12.0 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

Figure 12-1 shows a simplified block diagram of the Timer0 module in 8-bit mode and Figure 12-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

The T0CON register (Register 12-1) is a readable and writable register that controls all the aspects of Timer0, including the prescale selection.

### REGISTER 12-1: 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	T016BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

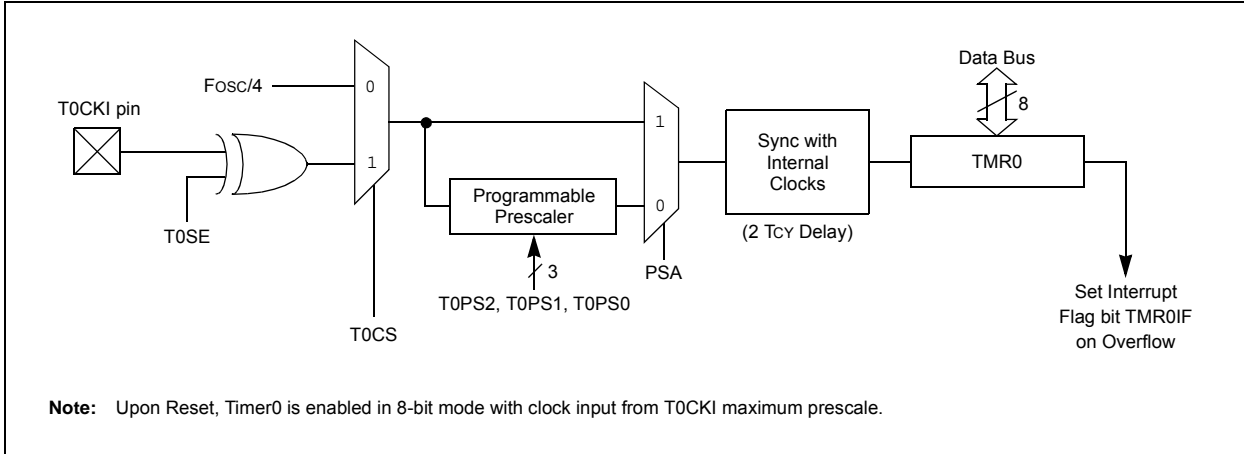
'0' = Bit is cleared

x = Bit is unknown

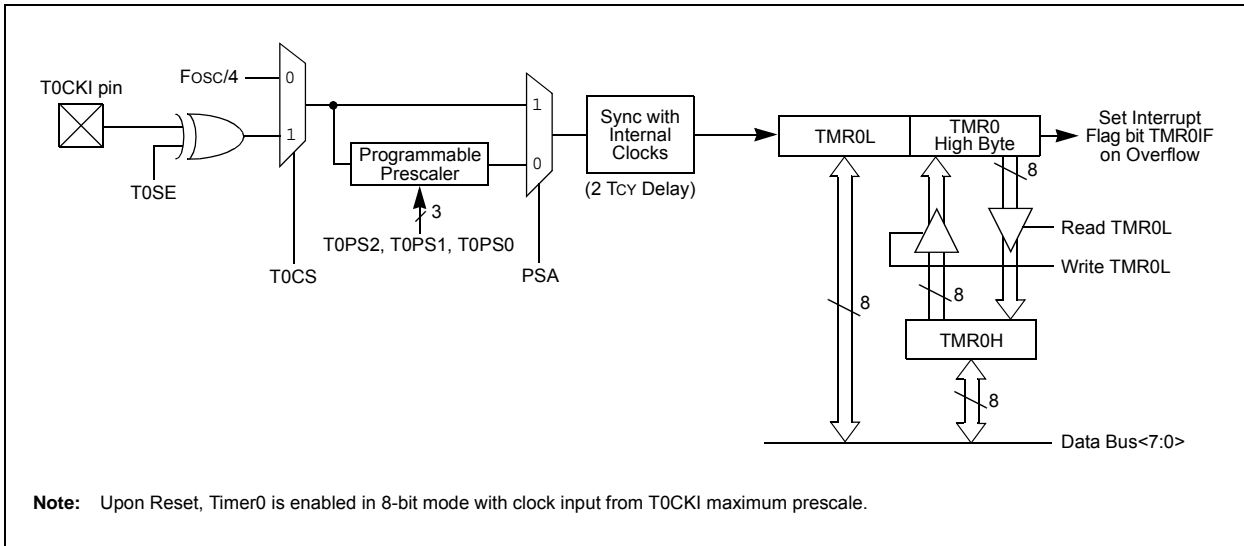
- bit 7      **TMR0ON:** Timer0 On/Off Control bit  
 1 = Enables Timer0  
 0 = Stops Timer0
- bit 6      **T016BIT:** Timer0 16-Bit Control bit  
 1 = Timer0 is configured as an 8-bit timer/counter  
 0 = Timer0 is configured as a 16-bit timer/counter
- bit 5      **T0CS:** Timer0 Clock Source Select bit  
 1 = Transition on T0CKI pin input edge  
 0 = Internal clock ( $F_{osc}/4$ )
- 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

# PIC18F1230/1330

**FIGURE 12-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE**



**FIGURE 12-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE**



## 12.1 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 TMR0 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 TMR0 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/AN2/VREF+. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge.

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.

## 12.2 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.

### 12.2.1 SWITCHING PRESCALER ASSIGNMENT

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

### 12.3 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 mode, since the timer requires clock cycles even when T0CS is set.

### 12.4 16-Bit Mode Timer Reads and Writes

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 12-2). The high byte of the Timer0 counter/timer is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid due to a rollover between successive reads of the high and low byte.

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

**TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER0**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
TMR0L	Timer0 Register Low Byte								48
TMR0H	Timer0 Register High Byte								48
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
T0CON	TMR0ON	T016BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	48
TRISA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	PORTA Data Direction Control Register						49

**Legend:** — = unimplemented locations read as ‘0’. Shaded cells are not used by Timer0.

**Note 1:** RA6 and RA7 are enabled as I/O pins depending on the oscillator mode selected in CONFIG1H.

# PIC18F1230/1330

---

NOTES:

## 13.0 TIMER1 MODULE

The Timer1 timer/counter module 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
- Status of system clock operation

Figure 13-1 is a simplified block diagram of the Timer1 module.

Register 13-1 details the Timer1 Control register. This register controls the operating mode of the Timer1 module and contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

The Timer1 oscillator can be used as a secondary clock source in power-managed modes. When the T1RUN bit is set, the Timer1 oscillator provides the system clock. If the Fail-Safe Clock Monitor is enabled and the Timer1 oscillator fails while providing the system clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications with only a minimal addition of external components and code overhead.

### REGISTER 13-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
bit 7							bit 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

- 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      **T1RUN:** Timer1 System Clock Status bit  
 1 = Device clock is derived from Timer1 oscillator  
 0 = Device clock is derived from another source
- bit 5-4    **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits  
 11 = 1:8 Prescale value  
 10 = 1:4 Prescale value  
 01 = 1:2 Prescale value  
 00 = 1:1 Prescale value
- bit 3      **T1OSCEN:** Timer1 Oscillator Enable bit  
 1 = Timer1 oscillator is enabled  
 0 = Timer1 oscillator is shut off  
 The oscillator inverter and feedback resistor are turned off to eliminate power drain.
- bit 2      **T1SYNC:** Timer1 External Clock Input Synchronization Select bit  
When TMR1CS = 1:  
 1 = Do not synchronize external clock input  
 0 = Synchronize external clock input  
When TMR1CS = 0:  
 This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.
- bit 1      **TMR1CS:** Timer1 Clock Source Select bit  
 1 = External clock from T1OSO/T1CKI (on the rising edge)<sup>(1)</sup>  
 0 = Internal clock (FOSC/4)
- bit 0      **TMR1ON:** Timer1 On bit  
 1 = Enables Timer1  
 0 = Stops Timer1

**Note 1:** Placement of T1OSI and T1OSO/T1CKI depends on the value of the Configuration bit, T1OSCMX, of CONFIG3H.

# PIC18F1230/1330

## 13.1 Timer1 Operation

Timer1 can operate in one of these modes:

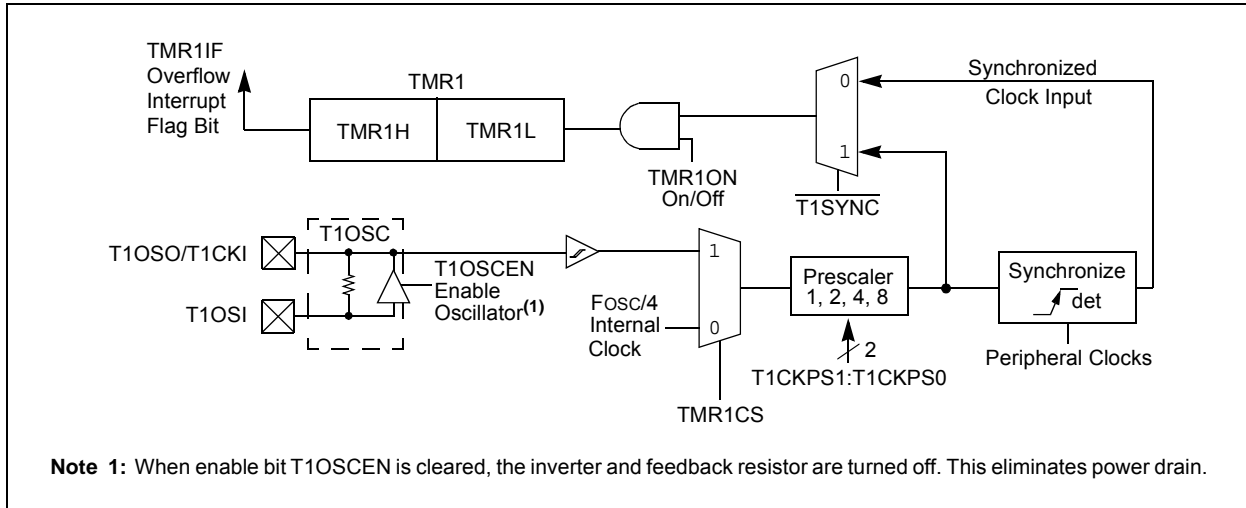
- As a timer
- As a synchronous counter
- As an asynchronous counter

The operating mode is determined by the Clock Select bit, TMR1CS (T1CON<1>).

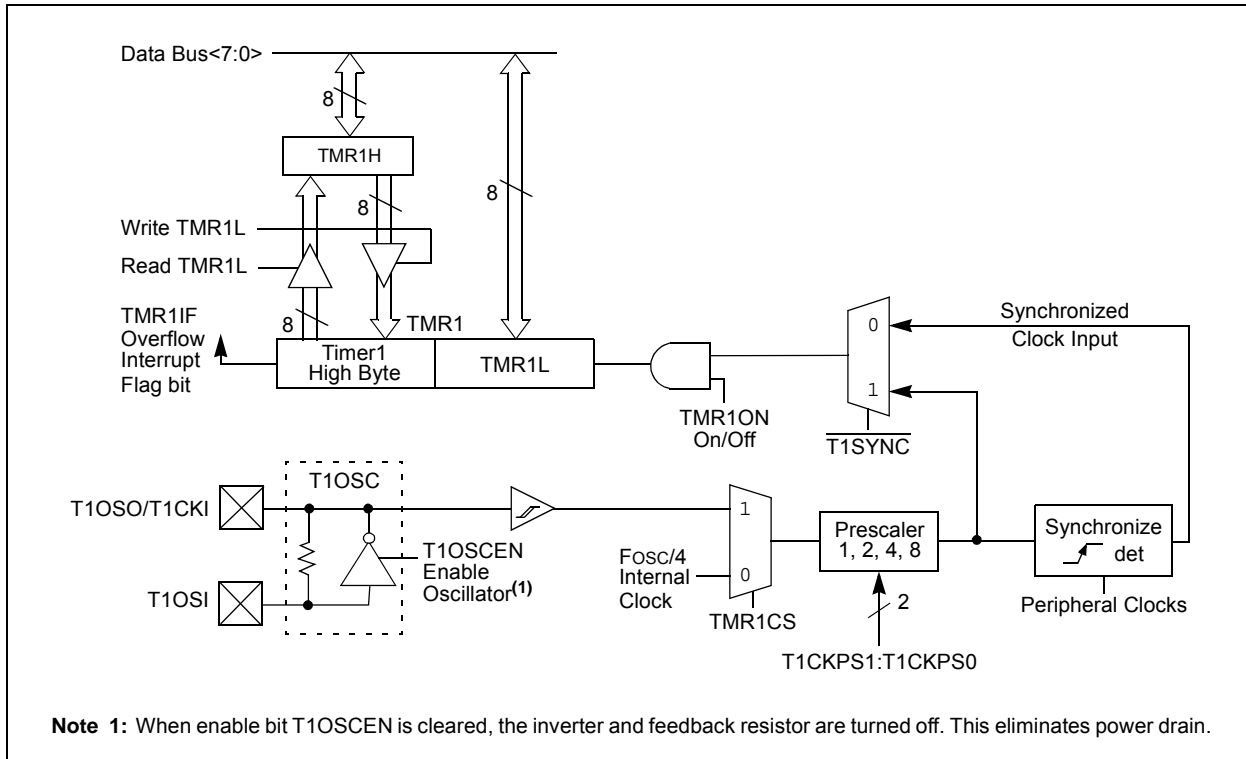
When TMR1CS = 0, Timer1 increments every instruction cycle. When TMR1CS = 1, Timer1 increments on every rising edge of the external clock input or the Timer1 oscillator, if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the T1OSI and T1OSO/T1CKI pins become inputs. That is, the corresponding TRISA bit value is ignored, and the pins are read as '0'.

**FIGURE 13-1: TIMER1 BLOCK DIAGRAM**



**FIGURE 13-2: TIMER1 BLOCK DIAGRAM: 16-BIT READ/WRITE MODE**



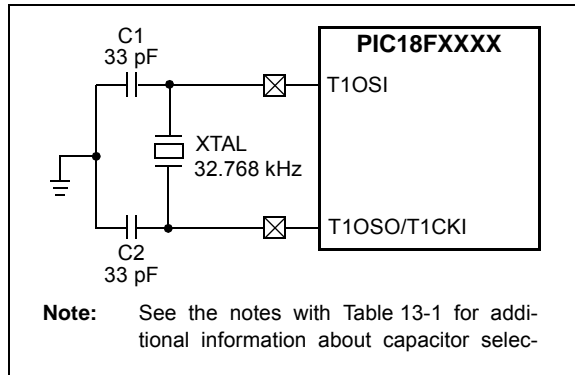


## 13.2 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO/TICKI (amplifier output). The placement of these pins depends on the value of Configuration bit, T1OSCMX (see **Section 20.1 “Configuration Bits”**). It is enabled by setting control bit T1OSCEN (T1CON<3>). The oscillator is a low-power oscillator rated for 32 kHz crystals. It will continue to run during all power-managed modes. The circuit for a typical LP oscillator is shown in Figure 13-3. Table 13-1 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.

**FIGURE 13-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**



**TABLE 13-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR**

Osc Type	Freq	C1	C2
LP	32 kHz	27 pF <sup>(1)</sup>	27 pF <sup>(1)</sup>

**Note 1:** Microchip suggests this value 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.

### 13.2.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power-managed modes. By setting the System Clock Select bits, SCS1:SCS0 (OSCCON<1:0>), to '01', the device switches to SEC\_RUN mode; both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a SLEEP instruction is executed, the device enters SEC\_IDLE mode. Additional details are available in **Section 4.0 “Power-Managed Modes”**.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 system clock status flag, T1RUN (T1CON<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source being currently used by the Fail-Safe Clock Monitor. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

## 13.3 Timer1 Oscillator Layout Considerations

The oscillator circuit, shown in Figure 13-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator (such as the PWM pin, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 13-4, may be helpful when used on a single-sided PCB, or in addition to a ground plane.

**FIGURE 13-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**



# PIC18F1230/1330

---

## 13.4 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing Timer1 interrupt enable bit, TMR1IE (PIE1<0>).

## 13.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 13-2). When the RD16 control bit (T1CON<7>) 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.

## 13.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 13.2 “Timer1 Oscillator”**), gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or super capacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, *RTCisr*, shown in Example 13-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow triggers the interrupt and calls the routine, which increments the seconds counter by one. Additional counters for minutes and hours are incremented as the previous counter overflow.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take 2 seconds. To force the overflow at the required one-second intervals, it is necessary to preload it. The simplest method is to set the MSb of TMR1H with a *BSF* instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1), as shown in the routine, *RTCinit*. The Timer1 oscillator must also be enabled and running at all times.

## EXAMPLE 13-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```

RTCinit
    MOVLW    0x80                ; Preload TMR1 register pair
    MOVWF    TMR1H              ; for 1 second overflow
    CLRF     TMR1L
    MOVLW    b'00001111'       ; Configure for external clock,
    MOVWF    T1CON              ; Asynchronous operation, external oscillator
    CLRF     secs               ; Initialize timekeeping registers
    CLRF     mins
    MOVLW    .12
    MOVWF    hours
    BSF      PIE1, TMR1IE      ; Enable Timer1 interrupt
    RETURN

RTCisr
    BSF      TMR1H, 7          ; Preload for 1 sec overflow
    BCF      PIR1, TMR1IF     ; Clear interrupt flag
    INCF     secs, F           ; Increment seconds
    MOVLW    .59              ; 60 seconds elapsed?
    CPFSGT   secs
    RETURN                    ; No, done
    CLRF     secs             ; Clear seconds
    INCF     mins, F          ; Increment minutes
    MOVLW    .59              ; 60 minutes elapsed?
    CPFSGT   mins
    RETURN                    ; No, done
    CLRF     mins            ; clear minutes
    INCF     hours, F         ; Increment hours
    MOVLW    .23              ; 24 hours elapsed?
    CPFSGT   hours
    RETURN                    ; No, done
    MOVLW    .01              ; Reset hours to 1
    MOVWF    hours
    RETURN                    ; Done
    
```

**TABLE 13-2: 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	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
TMR1L	Timer1 Register Low Byte								48
TMR1H	Timer1 Register High Byte								48
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	48

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

# PIC18F1230/1330

---

NOTES:

## 14.0 POWER CONTROL PWM MODULE

The Power Control PWM module simplifies the task of generating multiple, synchronized Pulse-Width Modulated (PWM) outputs for use in the control of motor controllers and power conversion applications. In particular, the following power and motion control applications are supported by the PWM module:

- Three-Phase and Single-Phase AC Induction Motors
- Switched Reluctance Motors
- Brushless DC (BLDC) Motors
- Uninterruptible Power Supplies (UPS)
- Multiple DC Brush Motors

The PWM module has the following features:

- Up to six PWM I/O pins with three duty cycle generators. Pins can be paired to acquire a complete half-bridge control.
- Up to 14-bit resolution, depending upon the PWM period.
- “On-the-fly” PWM frequency changes.
- Edge and Center-Aligned Output modes.
- Single-Pulse Generation mode.
- Programmable dead-time control between paired PWMs.
- Interrupt support for asymmetrical updates in Center-Aligned mode.
- Output override for Electrically Commutated Motor (ECM) operation; for example, BLDC.
- Special Event Trigger comparator for triggering A/D conversion.
- PWM outputs disable feature sets PWM outputs to their inactive state when in Debug mode.

The Power Control PWM module supports three PWM generators and six output channels on PIC18F1230/1330 devices. A simplified block diagram of the module is shown in Figure 14-1. Figure 14-2 and Figure 14-3 show how the module hardware is configured for each PWM output pair for the Complementary and Independent Output modes.

Each functional unit of the PWM module will be discussed in subsequent sections.

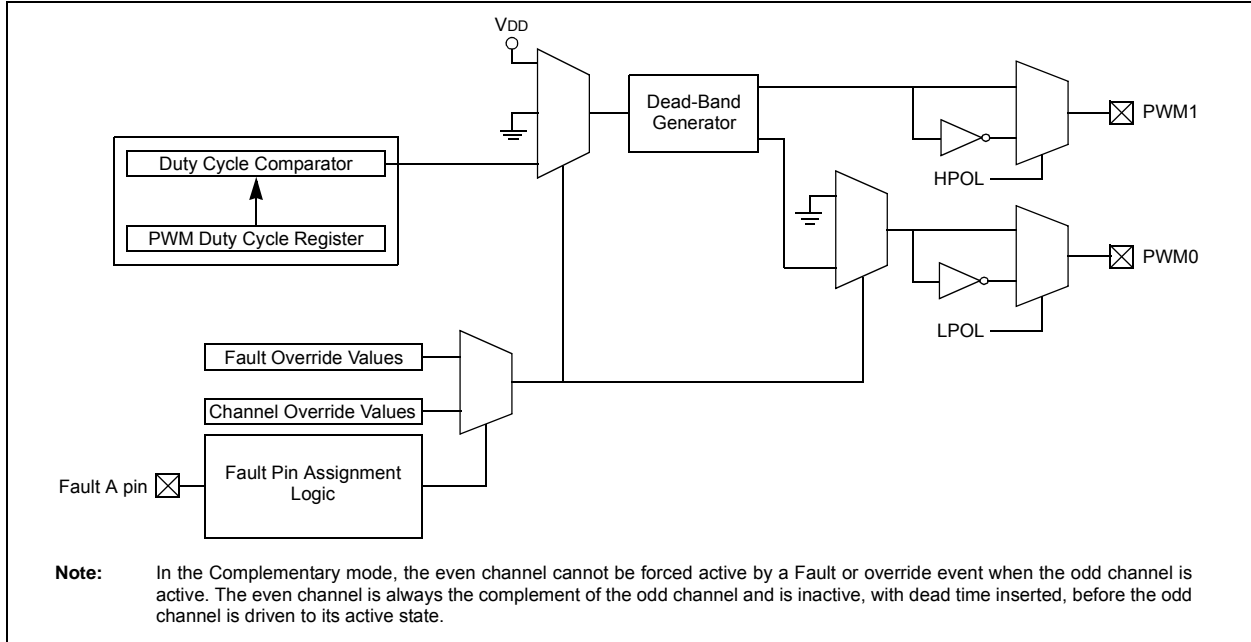
# PIC18F1230/1330

**FIGURE 14-1: POWER CONTROL PWM MODULE BLOCK DIAGRAM**

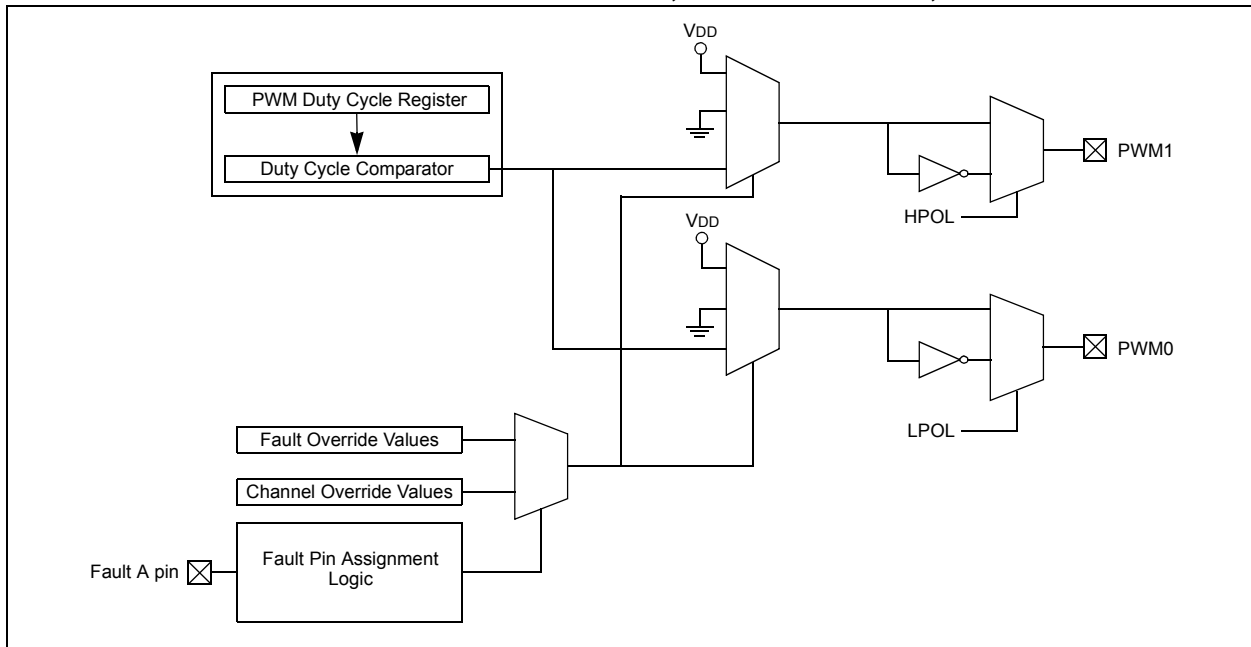


**Note 1:** Only PWM Generator 2 is shown in detail. The other generators are identical; their details are omitted for clarity.

**FIGURE 14-2: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, COMPLEMENTARY MODE**



**FIGURE 14-3: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, INDEPENDENT MODE**



This module contains three duty cycle generators, numbered 0 through 2. The module has six PWM output pins, numbered 0 through 5. The six PWM outputs are grouped into output pairs of even and odd numbered outputs. In Complementary modes, the even PWM pins must always be the complement of the corresponding odd PWM pins. For example, PWM0 will be the complement of PWM1 and PWM2 will be the complement of PWM3. The dead-time generator

inserts an OFF period called “dead time” between the going OFF of one pin to the going ON of the complementary pin of the paired pins. This is to prevent damage to the power switching devices that will be connected to the PWM output pins.

The time base for the PWM module is provided by its own 12-bit timer, which also incorporates selectable prescaler and postscaler options.

# PIC18F1230/1330

---

## 14.1 Control Registers

The operation of the PWM module is controlled by a total of 20 registers. Eight of these are used to configure the features of the module:

- PWM Timer Control Register 0 (PTCON0)
- PWM Timer Control Register 1 (PTCON1)
- PWM Control Register 0 (PWMCON0)
- PWM Control Register 1 (PWMCON1)
- Dead-Time Control Register (DTCON)
- Output Override Control Register (OVDCOND)
- Output State Register (OVDCONS)
- Fault Configuration Register (FLTCONFIG)

There are also 12 registers that are configured as six register pairs of 16 bits. These are used for the configuration values of specific features. They are:

- PWM Time Base Registers (PTMRH and PTMRL)
- PWM Time Base Period Registers (PTPERH and PTPERL)
- PWM Special Event Compare Registers (SEVTCMPH and SEVTCMPL)
- PWM Duty Cycle #0 Registers (PDC0H and PDC0L)
- PWM Duty Cycle #1 Registers (PDC1H and PDC1L)
- PWM Duty Cycle #2 Registers (PDC2H and PDC2L)

All of these register pairs are double-buffered.

## 14.2 Module Functionality

The PWM module supports several modes of operation that are beneficial for specific power and motor control applications. Each mode of operation is described in subsequent sections.

The PWM module is composed of several functional blocks. The operation of each is explained separately in relation to the several modes of operation:

- PWM Time Base
- PWM Time Base Interrupts
- PWM Period
- PWM Duty Cycle
- Dead-Time Generators
- PWM Output Overrides
- PWM Fault Inputs
- PWM Special Event Trigger

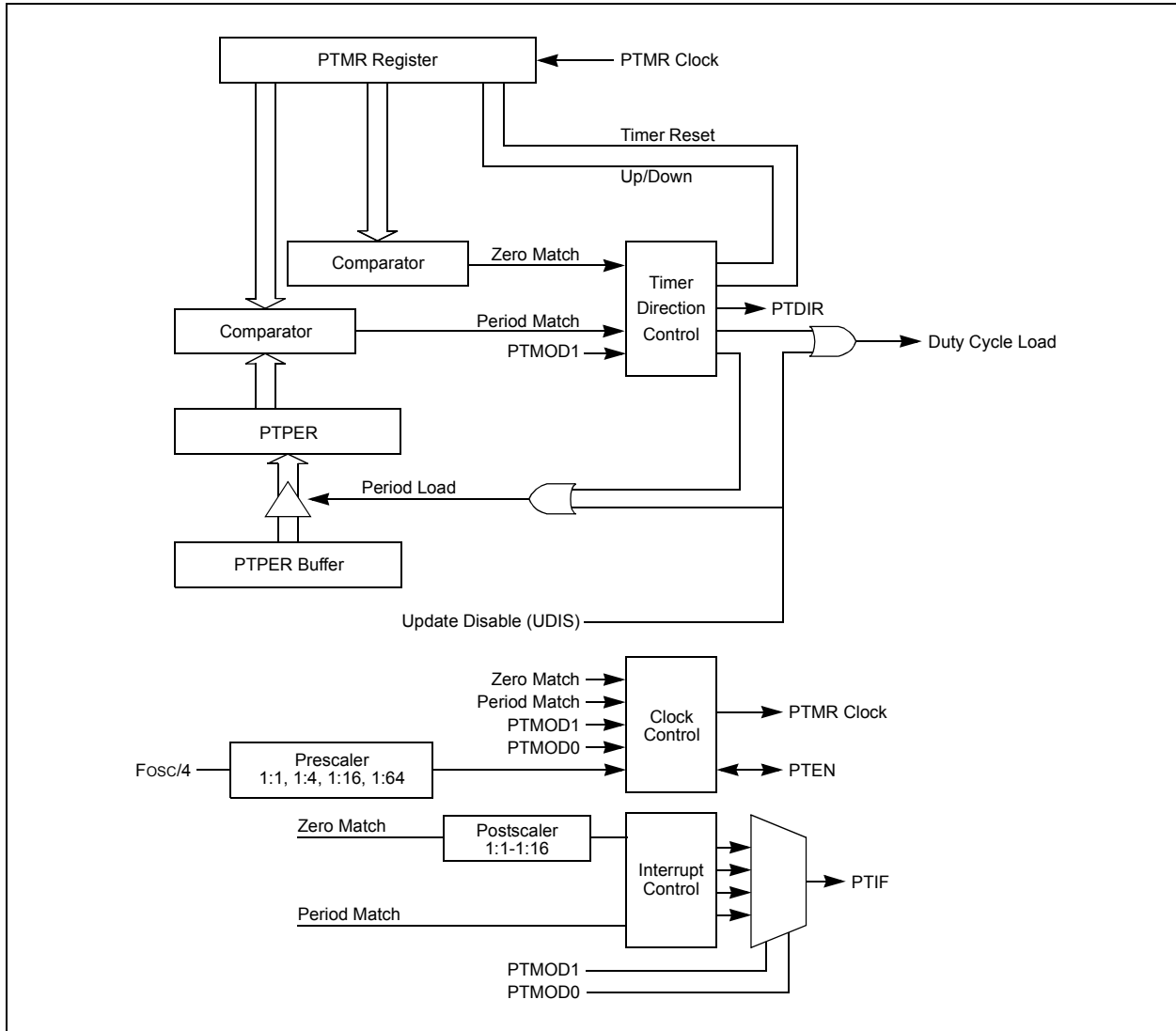
## 14.3 PWM Time Base

The PWM time base is provided by a 12-bit timer with prescaler and postscaler functions. A simplified block diagram of the PWM time base is shown in Figure 14-4. The PWM time base is configured through the PTCON0 and PTCON1 registers. The time base is enabled or disabled by respectively setting or clearing the PTEN bit in the PTCON1 register.

<b>Note:</b> The PTMR register pair (PTMRL:PTMRH) is not cleared when the PTEN bit is cleared in software.
--



**FIGURE 14-4: PWM TIME BASE BLOCK DIAGRAM**



The PWM time base can be configured for four different modes of operation:

- Free-Running mode
- Single-Shot mode
- Continuous Up/Down Count mode
- Continuous Up/Down Count mode with interrupts for double updates

These four modes are selected by the PTMOD1:PTMOD0 bits in the PTCO<sub>0</sub> register. The Free-Running mode produces edge-aligned PWM generation. The Continuous Up/Down Count modes produce center-aligned PWM generation. The Single-Shot mode allows the PWM module to support pulse control of certain Electronically Commutated Motors (ECMs) and produces edge-aligned operation.

# PIC18F1230/1330

## REGISTER 14-1: PTCON0: PWM TIMER CONTROL REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0
bit 7							bit 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

bit 7-4                      **PTOPS3:PTOPS0:** PWM Time Base Output Postscale Select bits

0000 = 1:1 Postscale

0001 = 1:2 Postscale

.

.

.

1111 = 1:16 Postscale

bit 3-2                      **PTCKPS1:PTCKPS0:** PWM Time Base Input Clock Prescale Select bits

00 = PWM time base input clock is Fosc/4 (1:1 prescale)

01 = PWM time base input clock is Fosc/16 (1:4 prescale)

10 = PWM time base input clock is Fosc/64 (1:16 prescale)

11 = PWM time base input clock is Fosc/256 (1:64 prescale)

bit 1-0                      **PTMOD1:PTMOD0:** PWM Time Base Mode Select bits

11 = PWM time base operates in a Continuous Up/Down Count mode with interrupts for double PWM updates

10 = PWM time base operates in a Continuous Up/Down Count mode

01 = PWM time base configured for Single-Shot mode

00 = PWM time base operates in a Free-Running mode

## REGISTER 14-2: PTCON1: PWM TIMER CONTROL REGISTER 1

R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
PTEN	PTDIR	—	—	—	—	—	—
bit 7							bit 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

bit 7                      **PTEN:** PWM Time Base Timer Enable bit

1 = PWM time base is on

0 = PWM time base is off

bit 6                      **PTDIR:** PWM Time Base Count Direction Status bit

1 = PWM time base counts down

0 = PWM time base counts up

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

## REGISTER 14-3: PWMCON0: PWM CONTROL REGISTER 0

U-0	R/W-1 <sup>(1)</sup>	R/W-1 <sup>(1)</sup>	R/W-1 <sup>(1)</sup>	U-0	R/W-0	R/W-0	R/W-0
—	PWMEN2	PWMEN1	PWMEN0	—	PMOD2	PMOD1	PMOD0
bit 7				bit 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

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

bit 6-4                      **PWMEN2:PWMEN0:** PWM Module Enable bits<sup>(1)</sup>

- 111 = All odd PWM I/O pins enabled for PWM output
- 110 = PWM1, PWM3 pins enabled for PWM output
- 10x = All PWM I/O pins enabled for PWM output
- 011 = PWM0, PWM1, PWM2 and PWM3 I/O pins enabled for PWM output
- 010 = PWM0 and PWM1 pins enabled for PWM output
- 001 = PWM1 pin is enabled for PWM output
- 000 = PWM module disabled; all PWM I/O pins are general purpose I/O

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

bit 2-0                      **PMOD2:PMOD0:** PWM Output Pair Mode bits

For PMOD0:

- 1 = PWM I/O pin pair (PWM0, PWM1) is in the Independent mode
- 0 = PWM I/O pin pair (PWM0, PWM1) is in the Complementary mode

For PMOD1:

- 1 = PWM I/O pin pair (PWM2, PWM3) is in the Independent mode
- 0 = PWM I/O pin pair (PWM2, PWM3) is in the Complementary mode

For PMOD2:

- 1 = PWM I/O pin pair (PWM4, PWM5) is in the Independent mode
- 0 = PWM I/O pin pair (PWM4, PWM5) is in the Complementary mode

**Note 1:** Reset condition of PWMEN bits depends on the PWMPIN Configuration bit of CONFIG3L.

# PIC18F1230/1330

## REGISTER 14-4: PWMCON1: PWM CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC
bit 7							bit 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

bit 7-4      **SEVOPS3:SEVOPS0:** PWM Special Event Trigger Output Postscale Select bits

0000 = 1:1 Postscale

0001 = 1:2 Postscale

.

.

.

1111 = 1:16 Postscale

bit 3      **SEVTDIR:** Special Event Trigger Time Base Direction bit

1 = A Special Event Trigger will occur when the PWM time base is counting downwards

0 = A Special Event Trigger will occur when the PWM time base is counting upwards

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

bit 1      **UDIS:** PWM Update Disable bit

1 = Updates from Duty Cycle and Period Buffer registers are disabled

0 = Updates from Duty Cycle and Period Buffer registers are enabled

bit 0      **OSYNC:** PWM Output Override Synchronization bit

1 = Output overrides via the OVDCON register are synchronized to the PWM time base

0 = Output overrides via the OVDCON register are asynchronous

## 14.3.1 FREE-RUNNING MODE

In the Free-Running mode, the PWM time base (PTMRL and PTMRH) will begin counting upwards until the value in the PWM Time Base Period register, PTPER (PTPERL and PTPERH), is matched. The PTMR registers will be reset on the following input clock edge and the time base will continue counting upwards as long as the PTEN bit remains set.

## 14.3.2 SINGLE-SHOT MODE

In the Single-Shot mode, the PWM time base will begin counting upwards when the PTEN bit is set. When the value in the PTMR register matches the PTPER register, the PTMR register will be reset on the following input clock edge and the PTEN bit will be cleared by the hardware to halt the time base.

## 14.3.3 CONTINUOUS UP/DOWN COUNT MODES

In Continuous Up/Down Count modes, the PWM time base counts upwards until the value in the PTPER register matches the PTMR register. On the following input clock edge, the timer counts downwards. The PTDIR bit in the PTCOEN1 register is read-only and indicates the counting direction. The PTDIR bit is set when the timer counts downwards.

**Note:** Since the PWM compare outputs are driven to the active state when the PWM time-base is counting downwards and matches the duty cycle value, the PWM outputs are held inactive during the first half of the first period of the Continuous Up/Down Count mode until the PTMR begins to count down from the PTPER value.

## 14.3.4 PWM TIME BASE PRESCALER

The input clock to PTMR ( $F_{osc}/4$ ) has prescaler options of 1:1, 1:4, 1:16 or 1:64. These are selected by control bits, PTCKPS<1:0>, in the PTCOEN0 register. The prescaler counter is cleared when any of the following occurs:

- Write to the PTMR register
- Write to the PTCOEN (PTCOEN0 or PTCOEN1) register
- Any device Reset

**Note:** The PTMR register is not cleared when PTCOENx is written.

Table 14-1 shows the minimum PWM frequencies that can be generated with the PWM time base and the prescaler. An operating frequency of 40 MHz ( $F_{CYC} = 10$  MHz) and PTPER = 0xFFFF are assumed in the table. The PWM module must be capable of generating PWM signals at the line frequency (50 Hz or 60 Hz) for certain power control applications.

**TABLE 14-1: MINIMUM PWM FREQUENCY**

Minimum PWM Frequencies vs. Prescaler Value for $F_{CYC} = 10$ MIPS (PTPER = 0FFFh)		
Prescale	PWM Frequency Edge-Aligned	PWM Frequency Center-Aligned
1:1	2441 Hz	1221 Hz
1:4	610 Hz	305 Hz
1:16	153 Hz	76 Hz
1:64	38 Hz	19 Hz

## 14.3.5 PWM TIME BASE POSTSCALER

The match output of PTMR can optionally be postscaled through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate an interrupt. The postscaler counter is cleared when any of the following occurs:

- Write to the PTMR register
- Write to the PTCOENx register
- Any device Reset

The PTMR register is not cleared when PTCOENx is written.

## 14.4 PWM Time Base Interrupts

The PWM timer can generate interrupts based on the modes of operation selected by the PTMOD<1:0> bits and the postscaler bits (PTOPS<3:0>).

### 14.4.1 INTERRUPTS IN FREE-RUNNING MODE

When the PWM time base is in the Free-Running mode (PTMOD<1:0> = 00), an interrupt event is generated each time a match with the PTPER register occurs. The PTMR register is reset to zero in the following clock edge.

Using a postscaler selection other than 1:1 will reduce the frequency of interrupt events.

# PIC18F1230/1330

**FIGURE 14-5: PWM TIME BASE INTERRUPT TIMING, FREE-RUNNING MODE**



## 14.4.2 INTERRUPTS IN SINGLE-SHOT MODE

When the PWM time base is in the Single-Shot mode (PTMOD<1:0> = 01), an interrupt event is generated when a match with the PTPER register occurs. The PWM Time Base register (PTMR) is reset to zero on the following input clock edge and the PTEN bit is cleared. The postscaler selection bits have no effect in this Timer mode.

## 14.4.3 INTERRUPTS IN CONTINUOUS UP/DOWN COUNT MODE

In the Continuous Up/Down Count mode (PTMOD<1:0> = 10), an interrupt event is generated each time the value of the PTMR register becomes zero and the PWM time base begins to count upwards. The postscaler selection bits may be used in this Timer mode to reduce the frequency of the interrupt events. Figure 14-7 shows the interrupts in Continuous Up/Down Count mode.

**FIGURE 14-6: PWM TIME BASE INTERRUPT TIMING, SINGLE-SHOT MODE**



**FIGURE 14-7: PWM TIME BASE INTERRUPTS, CONTINUOUS UP/DOWN COUNT MODE**



# PIC18F1230/1330

## 14.4.4 INTERRUPTS IN DOUBLE UPDATE MODE

This mode is available in Continuous Up/Down Count mode. In the Double Update mode (PTMOD<1:0> = 11), an interrupt event is generated each time the PTMR register is equal to zero and each time the PTMR matches the PTPER register. Figure 14-8 shows the interrupts in Continuous Up/Down Count mode with double updates.

The Double Update mode provides two additional functions to the user in Center-Aligned mode.

1. The control loop bandwidth is doubled because the PWM duty cycles can be updated twice per period.

2. Asymmetrical center-aligned PWM waveforms can be generated, which are useful for minimizing output waveform distortion in certain motor control applications.

**Note:** Do not change the PTMOD bits while PTEN is active. It will yield unexpected results. To change the PWM Timer mode of operation, first clear the PTEN bit, load PTMOD bits with required data and then set PTEN.

**FIGURE 14-8: PWM TIME BASE INTERRUPTS, CONTINUOUS UP/DOWN COUNT MODE WITH DOUBLE UPDATES**





## 14.5 PWM Period

The PWM period is defined by the PTPER register pair (PTPERL and PTPERH). The PWM period has 12-bit resolution by combining 4 LSBs of PTPERH and 8 bits of PTPERL. PTPER is a double-buffered register used to set the counting period for the PWM time base.

The PTPER buffer contents are loaded into the PTPER register at the following times:

- Free-Running and Single-Shot modes: When the PTMR register is reset to zero after a match with the PTPER register.
- Continuous Up/Down Count modes: When the PTMR register is zero. The value held in the PTPER buffer is automatically loaded into the PTPER register when the PWM time base is disabled (PTEN = 0). Figure 14-9 and Figure 14-10 indicate the times when the contents of the PTPER buffer are loaded into the actual PTPER register.

The PWM period can be calculated from the following formulas:

### EQUATION 14-1: PWM PERIOD FOR FREE-RUNNING MODE

$$T_{\text{PWM}} = \frac{(\text{PTPER} + 1) \times \text{PTMRPS}}{\text{FOSC}/4}$$

### EQUATION 14-2: PWM PERIOD FOR CONTINUOUS UP/DOWN COUNT MODE

$$T_{\text{PWM}} = \frac{(2 \times \text{PTPER}) \times \text{PTMRPS}}{\frac{\text{FOSC}}{4}}$$

The PWM frequency is the inverse of period; or

### EQUATION 14-3: PWM FREQUENCY

$$\text{PWM Frequency} = \frac{1}{\text{PWM Period}}$$

The maximum resolution (in bits) for a given device oscillator and PWM frequency can be determined from the following formula:

### EQUATION 14-4: PWM RESOLUTION

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

The PWM resolutions and frequencies are shown for a selection of execution speeds and PTPER values in Table 14-2. The PWM frequencies in Table 14-2 are calculated for Edge-Aligned PWM mode. For Center-Aligned mode, the PWM frequencies will be approximately one-half the values indicated in this table.

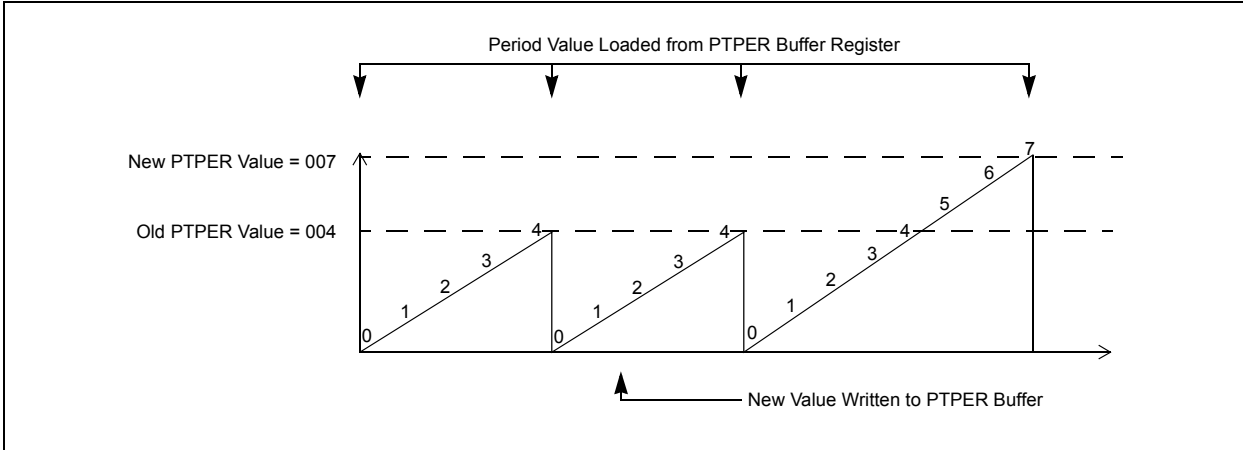
**TABLE 14-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS**

PWM Frequency = 1/T <sub>PWM</sub>				
Fosc	MIPS	PTPER Value	PWM Resolution	PWM Frequency
40 MHz	10	0FFFh	14 bits	2.4 kHz
40 MHz	10	07FFh	13 bits	4.9 kHz
40 MHz	10	03FFh	12 bits	9.8 kHz
40 MHz	10	01FFh	11 bits	19.5 kHz
40 MHz	10	FFh	10 bits	39.0 kHz
40 MHz	10	7Fh	9 bits	78.1 kHz
40 MHz	10	3Fh	8 bits	156.2 kHz
40 MHz	10	1Fh	7 bits	312.5 kHz
40 MHz	10	0Fh	6 bits	625 kHz
25 MHz	6.25	0FFFh	14 bits	1.5 kHz
25 MHz	6.25	03FFh	12 bits	6.1 kHz
25 MHz	6.25	FFh	10 bits	24.4 kHz
10 MHz	2.5	0FFFh	14 bits	610 Hz
10 MHz	2.5	03FFh	12 bits	2.4 kHz
10 MHz	2.5	FFh	10 bits	9.8 kHz
5 MHz	1.25	0FFFh	14 bits	305 Hz
5 MHz	1.25	03FFh	12 bits	1.2 kHz
5 MHz	1.25	FFh	10 bits	4.9 kHz
4 MHz	1	0FFFh	14 bits	244 Hz
4 MHz	1	03FFh	12 bits	976 Hz
4 MHz	1	FFh	10 bits	3.9 kHz

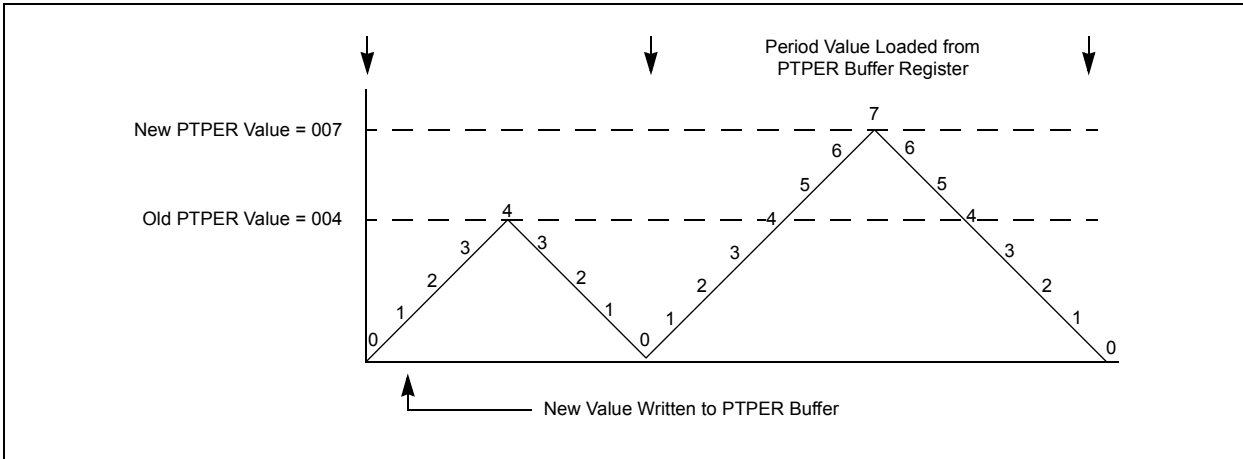
**Note:** For center-aligned operation, PWM frequencies will be approximately 1/2 the value indicated in the table.

# PIC18F1230/1330

**FIGURE 14-9: PWM PERIOD BUFFER UPDATES IN FREE-RUNNING MODE**



**FIGURE 14-10: PWM PERIOD BUFFER UPDATES IN CONTINUOUS UP/DOWN COUNT MODES**



## 14.6 PWM Duty Cycle

PWM duty cycle is defined by the PDCx (PDCxL and PDCxH) registers. There are a total of three PWM Duty Cycle registers for four pairs of PWM channels. The Duty Cycle registers have 14-bit resolution by combining the six LSBs of PDCxH with the 8 bits of PDCxL. PDCx is a double-buffered register used to set the counting period for the PWM time base.

### 14.6.1 PWM DUTY CYCLE REGISTERS

There are three 14-bit Special Function Registers used to specify duty cycle values for the PWM module:

- PDC0 (PDC0L and PDC0H)
- PDC1 (PDC1L and PDC1H)
- PDC2 (PDC2L and PDC2H)

The value in each Duty Cycle register determines the amount of time that the PWM output is in the active state. The upper 12 bits of PDCx hold the actual duty cycle value from PTMRH/L<11:0>, while the lower two bits control which internal Q clock the duty cycle match will occur. This 2-bit value is decoded from the Q clocks, as shown in Figure 14-11, when the prescaler is 1:1 (PTCKPS<1:0> = 00).

In Edge-Aligned mode, the PWM period starts at Q1 and ends when the Duty Cycle register matches the PTMR register as follows. The duty cycle match is considered when the upper 12 bits of the PDCx are equal to the

PTMR and the lower 2 bits are equal to Q1, Q2, Q3 or Q4, depending on the lower two bits of the PDCx (when the prescaler is 1:1 or PTCKPS<1:0> = 00).

**Note:** When the prescaler is not 1:1 (PTCKPS<1:0> ≠ ~00), the duty cycle match occurs at the Q1 clock of the instruction cycle when the PTMR and PDCx match occurs.

Each compare unit has logic that allows override of the PWM signals. This logic also ensures that the PWM signals will complement each other (with dead-time insertion) in Complementary mode (see Section 14.7 “Dead-Time Generators”).

**Note:** To get the correct PWM duty cycle, always multiply the calculated PWM duty cycle value by four before writing it to the PWM Duty Cycle registers. This is due to the two additional LSBs in the PWM Duty Cycle registers which are compared against the internal Q clock for the PWM duty cycle match.

**FIGURE 14-11: DUTY CYCLE COMPARISON**



# PIC18F1230/1330

## 14.6.2 DUTY CYCLE REGISTER BUFFERS

The three PWM Duty Cycle registers are double-buffered to allow glitchless updates of the PWM outputs. For each duty cycle block, there is a Duty Cycle Buffer register that is accessible by the user and a second Duty Cycle register that holds the actual compare value used in the present PWM period.

In Edge-Aligned PWM Output mode, a new duty cycle value will be updated whenever a PTMR match with the PTPER register occurs and PTMR is reset, as shown in Figure 14-12. Also, the contents of the duty cycle buffers are automatically loaded into the Duty Cycle registers when the PWM time base is disabled (PTEN = 0).

When the PWM time base is in the Continuous Up/Down Count mode, new duty cycle values will be updated when the value of the PTMR register is zero and the PWM time base begins to count upwards. The contents of the duty cycle buffers are automatically loaded into the Duty Cycle registers when the PWM time base is disabled (PTEN = 0). Figure 14-13 shows the timings when the duty cycle update occurs for the Continuous Up/Down Count mode. In this mode, up to one entire PWM period is available for calculating and loading the new PWM duty cycle before changes take effect.

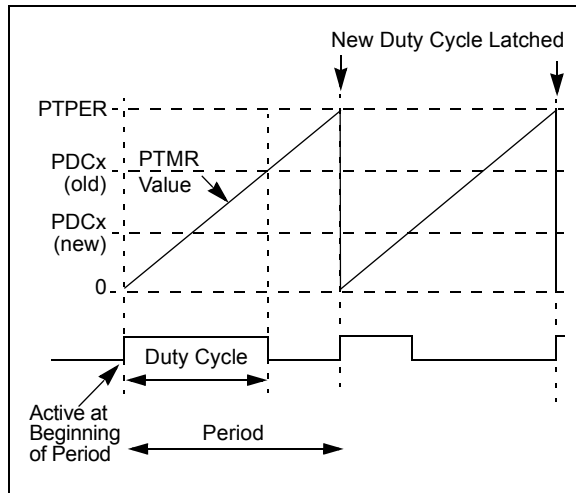
When the PWM time base is in the Continuous Up/Down Count mode with double updates, new duty cycle values will be updated when the value of the PTMR register is zero and when the value of the PTMR register matches the value in the PTPER register. The contents of the duty cycle buffers are automatically loaded into the Duty Cycle registers during both of the previously described conditions. Figure 14-14 shows the duty cycle updates for Continuous Up/Down Count mode with double updates. In this mode, up to half of a PWM period is available for calculating and loading the new PWM duty cycle before changes take effect.

## 14.6.3 EDGE-ALIGNED PWM

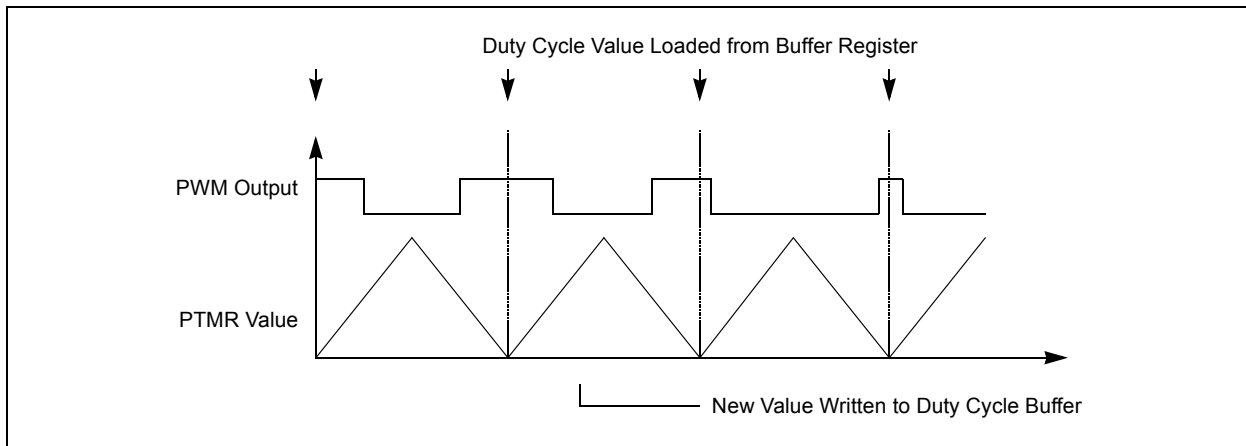
Edge-aligned PWM signals are produced by the module when the PWM time base is in the Free-Running mode or the Single-Shot mode. For edge-aligned PWM outputs, the output for a given PWM channel has a period specified by the value loaded in PTPER and a duty cycle specified by the appropriate Duty Cycle register (see Figure 14-12). The PWM output is driven active at the beginning of the period (PTMR = 0) and is driven inactive when the value in the Duty Cycle register matches PTMR. A new cycle is started when PTMR matches the PTPER, as explained in the PWM period section.

If the value in a particular Duty Cycle register is zero, then the output on the corresponding PWM pin will be inactive for the entire PWM period. In addition, the output on the PWM pin will be active for the entire PWM period if the value in the Duty Cycle register is greater than the value held in the PTPER register.

**FIGURE 14-12: EDGE-ALIGNED PWM**



**FIGURE 14-13: DUTY CYCLE UPDATE TIMES IN CONTINUOUS UP/DOWN COUNT MODE**



**FIGURE 14-14: DUTY CYCLE UPDATE TIMES IN CONTINUOUS UP/DOWN COUNT MODE WITH DOUBLE UPDATES**



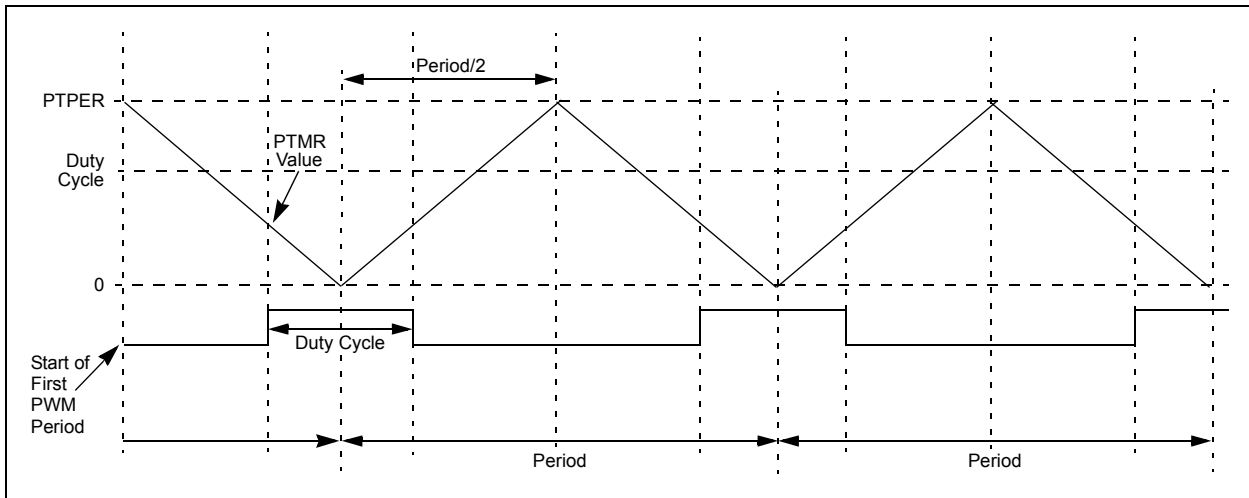
### 14.6.4 CENTER-ALIGNED PWM

Center-aligned PWM signals are produced by the module when the PWM time base is configured in a Continuous Up/Down Count mode (see Figure 14-15). The PWM compare output is driven to the active state when the value of the Duty Cycle register matches the value of PTMR and the PWM time base is counting downwards (PTDIR = 1). The PWM compare output will be driven to the inactive state when the PWM time base is counting upwards (PTDIR = 0) and the value in the PTMR register matches the duty cycle value. If the value in a particular Duty Cycle register is zero, then the output on the corresponding PWM pin will be

inactive for the entire PWM period. In addition, the output on the PWM pin will be active for the entire PWM period if the value in the Duty Cycle register is equal to or greater than the value in the PTPER register.

**Note:** When the PWM is started in Center-Aligned mode, the PWM Time Base Period register (PTPER) is loaded into the PWM Time Base register (PTMR) and the PTMR is configured automatically to start down counting. This is done to ensure that all the PWM signals don't start at the same time.

**FIGURE 14-15: START OF CENTER-ALIGNED PWM**



# PIC18F1230/1330

## 14.6.5 COMPLEMENTARY PWM OPERATION

The Complementary mode of PWM operation is useful to drive one or more power switches in half-bridge configuration, as shown in Figure 14-16. This inverter topology is typical for a 3-phase induction motor, brushless DC motor or 3-phase Uninterruptible Power Supply (UPS) control applications.

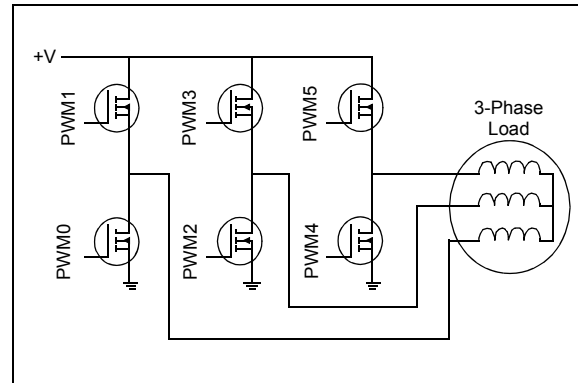
Each upper/lower power switch pair is fed by a complementary PWM signal. Dead time may be optionally inserted during device switching, where both outputs are inactive for a short period (see **Section 14.7 “Dead-Time Generators”**).

In Complementary mode, the duty cycle comparison units are assigned to the PWM outputs as follows:

- PDC0 register controls PWM1/PWM0 outputs
- PDC1 register controls PWM3/PWM2 outputs
- PDC2 register controls PWM5/PWM4 outputs

PWM1/3/5 are the main PWMs that are controlled by the PDCx registers and PWM0/2/4 are the complemented outputs. When using the PWMs to control the half-bridge, the odd number PWMs can be used to control the upper power switch and the even numbered PWMs can be used for the lower switches.

**FIGURE 14-16: TYPICAL LOAD FOR COMPLEMENTARY PWM OUTPUTS**



The Complementary mode is selected for each PWM I/O pin pair by clearing the appropriate PMODx bit in the PWMCON0 register. The PWM I/O pins are set to Complementary mode by default upon all kinds of device Resets.

## 14.7 Dead-Time Generators

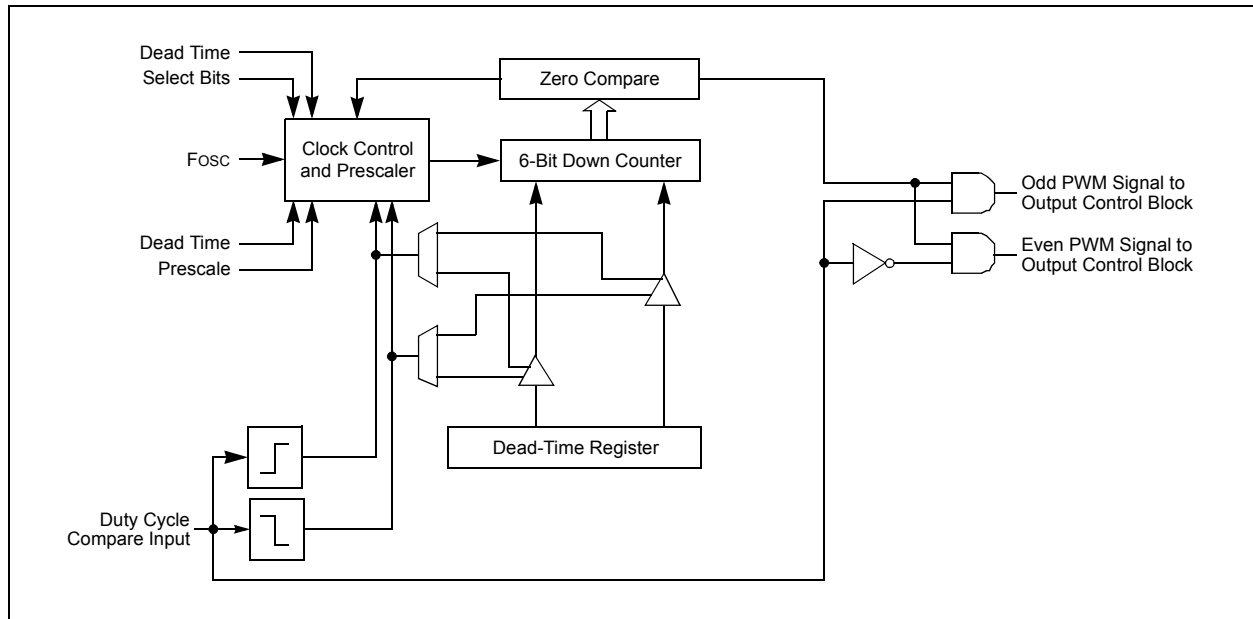
In power inverter applications, where the PWMs are used in Complementary mode to control the upper and lower switches of a half-bridge, a dead-time insertion is highly recommended. The dead-time insertion keeps both outputs in inactive state for a brief time. This avoids any overlap in the switching during the state change of the power devices due to TON and TOFF characteristics.

Because the power output devices cannot switch instantaneously, some amount of time must be provided between the turn-off event of one PWM output in a complementary pair and the turn-on event of the other transistor. The PWM module allows dead time to be programmed. The following sections explain the dead-time block in detail.

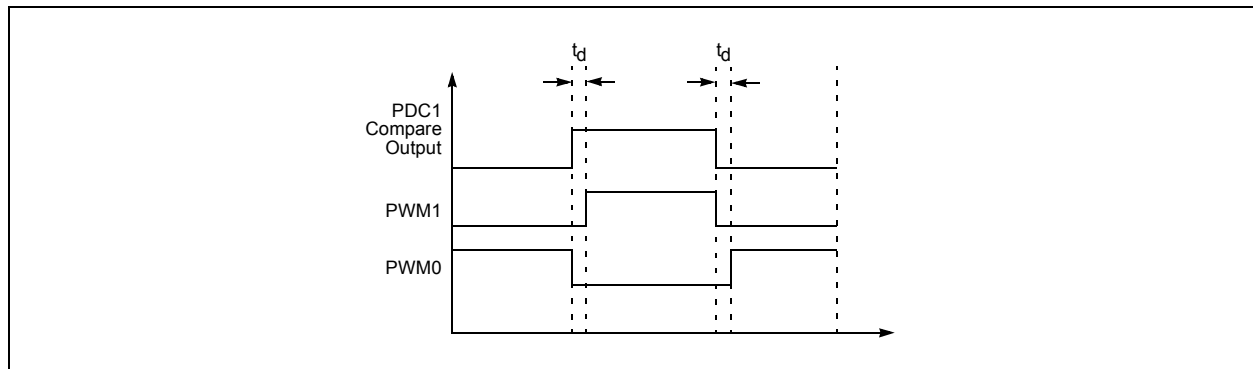
### 14.7.1 DEAD-TIME INSERTION

Each complementary output pair for the PWM module has a 6-bit down counter used to produce the dead-time insertion. As shown in Figure 14-17, each dead-time unit has a rising and falling edge detector connected to the duty cycle comparison output. The dead time is loaded into the timer on the detected PWM edge event. Depending on whether the edge is rising or falling, one of the transitions on the complementary outputs is delayed until the timer counts down to zero. A timing diagram, indicating the dead-time insertion for one pair of PWM outputs, is shown in Figure 14-18.

**FIGURE 14-17: DEAD-TIME CONTROL UNIT BLOCK DIAGRAM FOR ONE PWM OUTPUT PAIR**



**FIGURE 14-18: DEAD-TIME INSERTION FOR COMPLEMENTARY PWM**



# PIC18F1230/1330

## REGISTER 14-5: DTCON: DEAD-TIME 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
DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0
bit 7							bit 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

bit 7-6 **DTPS1:DTPS0:** Dead-Time Unit A Prescale Select bits

- 11 = Clock source for dead-time unit is  $F_{osc}/16$
- 10 = Clock source for dead-time unit is  $F_{osc}/8$
- 01 = Clock source for dead-time unit is  $F_{osc}/4$
- 00 = Clock source for dead-time unit is  $F_{osc}/2$

bit 5-0 **DT5:DT0:** Unsigned 6-Bit Dead-Time Value for Dead-Time Unit bits

### 14.7.2 DEAD-TIME RANGES

The amount of dead time provided by the dead-time unit is selected by specifying the input clock prescaler value and a 6-bit unsigned value defined in the DTCON register. Four input clock prescaler selections have been provided to allow a suitable range of dead times based on the device operating frequency.  $F_{osc}/2$ ,  $F_{osc}/4$ ,  $F_{osc}/8$  and  $F_{osc}/16$  are the clock prescaler options available using the DTPS1:DTPS0 control bits in the DTCON register.

After selecting an appropriate prescaler value, the dead time is adjusted by loading a 6-bit unsigned value into DTCON<5:0>. The dead-time unit prescaler is cleared on any of the following events:

- On a load of the down timer due to a duty cycle comparison edge event;
- On a write to the DTCON register; or
- On any device Reset.

### 14.7.3 DECREMENTING THE DEAD-TIME COUNTER

The dead-time counter is clocked from any of the Q clocks based on the following conditions.

1. The dead-time counter is clocked on Q1 when:
  - The DTPS bits are set to any of the following dead-time prescaler settings:  $F_{osc}/4$ ,  $F_{osc}/8$ ,  $F_{osc}/16$
  - The PWM Time Base Prescale bits (PTCKPS<1:0>) are set to any of the following prescale ratios:  $F_{osc}/16$ ,  $F_{osc}/64$ ,  $F_{osc}/256$
2. The dead-time counter is clocked by a pair of Q clocks when the PWM Time Base Prescale bits are set to 1:1 (PTCKPS<1:0> = 00,  $F_{osc}/4$ ) and the dead-time counter is clocked by the  $F_{osc}/2$  (DTPS<1:0> = 00).
3. The dead-time counter is clocked using every other Q clock, depending on the two LSBs in the Duty Cycle registers:
  - If the PWM duty cycle match occurs on Q1 or Q3, then the dead-time counter is clocked using every Q1 and Q3
  - If the PWM duty cycles match occurs on Q2 or Q4, then the dead-time counter is clocked using every Q2 and Q4
4. When the DTPS<1:0> bits are set to any of the other dead-time prescaler settings (i.e.,  $F_{osc}/4$ ,  $F_{osc}/8$  or  $F_{osc}/16$ ) and the PWM time base prescaler is set to 1:1, the dead-time counter is clocked by the Q clock corresponding to the Q clocks on which the PWM duty cycle match occurs.



The actual dead time is calculated from the DTCON register as follows:

$$\text{Dead Time} = \text{Dead-Time Value} / (\text{Fosc} / \text{Prescaler})$$

Table 14-3 shows example dead-time ranges as a function of the input clock prescaler selected and the device operating frequency.

**TABLE 14-3: EXAMPLE DEAD-TIME RANGES**

Fosc (MHz)	MIPS	Prescaler Selection	Dead-Time Min	Dead-Time Max
40	10	Fosc/2	50 ns	3.2 μs
40	10	Fosc/4	100 ns	6.4 μs
40	10	Fosc/8	200 ns	12.8 μs
40	10	Fosc/16	400 ns	25.6 μs
32	8	Fosc/2	62.5 ns	4 μs
32	8	Fosc/4	125 ns	8 μs
32	8	Fosc/8	250 ns	16 μs
32	8	Fosc/16	500 ns	32 μs
25	6.25	Fosc/2	80 ns	5.12 μs
25	6.25	Fosc/4	160 ns	10.2 μs
25	6.25	Fosc/8	320 ns	20.5 μs
25	6.25	Fosc/16	640 ns	41 μs
20	5	Fosc/2	100 ns	6.4 μs
20	5	Fosc/4	200 ns	12.8 μs
20	5	Fosc/8	400 ns	25.6 μs
20	5	Fosc/16	800 ns	51.2 μs
10	2.5	Fosc/2	200 ns	12.8 μs
10	2.5	Fosc/4	400 ns	25.6 μs
10	2.5	Fosc/8	800 ns	51.2 μs
10	2.5	Fosc/16	1.6 μs	102.4 μs
5	1.25	Fosc/2	400 ns	25.6 μs
5	1.25	Fosc/4	800 ns	51.2 μs
5	1.25	Fosc/8	1.6 μs	102.4 μs
5	1.25	Fosc/16	3.2 μs	204.8 μs
4	1	Fosc/2	0.5 μs	32 μs
4	1	Fosc/4	1 μs	64 μs
4	1	Fosc/8	2 μs	128 μs
4	1	Fosc/16	4 μs	256 μs

## 14.7.4 DEAD-TIME DISTORTION

**Note 1:** For small PWM duty cycles, the ratio of dead time to the active PWM time may become large. In this case, the inserted dead time will introduce distortion into waveforms produced by the PWM module. The user can ensure that dead-time distortion is minimized by keeping the PWM duty cycle at least three times larger than the dead time. A similar effect occurs for duty cycles at or near 100%. The maximum duty cycle used in the application should be chosen such that the minimum inactive time of the signal is at least three times larger than the dead time. If the dead time is greater or equal to the duty cycle of one of the PWM output pairs, then that PWM pair will be inactive for the whole period.

**2:** Changing the dead-time values in DTCON when the PWM is enabled may result in an undesirable situation. Disable the PWM (PTEN = 0) before changing the dead-time value.

## 14.8 Independent PWM Output

Independent PWM mode is used for driving the loads (as shown in Figure 14-19) that drive one winding of a switched reluctance motor. A particular PWM output pair is configured in the Independent Output mode when the corresponding PMODx bit in the PWMCON0 register is set. No dead-time control is implemented between the PWM I/O pins when the module is operating in the Independent PWM mode and both I/O pins are allowed to be active simultaneously. This mode can also be used to drive stepper motors.

### 14.8.1 DUTY CYCLE ASSIGNMENT IN THE INDEPENDENT PWM MODE

In the Independent PWM mode, each duty cycle generator is connected to both PWM output pins in a given PWM output pair. The odd and the even PWM output pins are driven with a single PWM duty cycle generator. PWM1 and PWM0 are driven by the PWM channel which uses the PDC0 register to set the duty cycle, PWM3 and PWM2 with PDC1, and PWM5 and PWM4 with PDC2 (see Figure 14-3 and Register 14-3).

# PIC18F1230/1330

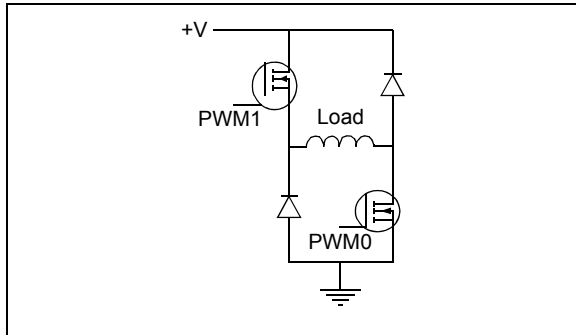
## 14.8.2 PWM CHANNEL OVERRIDE

PWM output may be manually overridden for each PWM channel by using the appropriate bits in the OVDCOND and OVDCONS registers. The user may select the following signal output options for each PWM output pin operating in the Independent PWM mode:

- I/O pin outputs PWM signal
- I/O pin inactive
- I/O pin active

Refer to **Section 14.10 “PWM Output Override”** for details for all the override functions.

**FIGURE 14-19: CENTER CONNECTED LOAD**



## 14.9 Single-Pulse PWM Operation

The single-pulse PWM operation is available only in Edge-Aligned mode. In this mode, the PWM module will produce single-pulse output. Single-pulse operation is configured when the PTMOD1:PTMOD0 bits are set to '01' in the PTCON0 register. This mode of operation is useful for driving certain types of ECMs.

In Single-Pulse mode, the PWM I/O pin(s) are driven to the active state when the PTEN bit is set. When the PWM timer match with Duty Cycle register occurs, the PWM I/O pin is driven to the inactive state. When the PWM timer match with the PTPER register occurs, the PTMR register is cleared, all active PWM I/O pins are driven to the inactive state, the PTEN bit is cleared and an interrupt is generated if the corresponding interrupt bit is set.

**Note:** PTPER and PDCx values are held as they are after the single-pulse output. To have another cycle of single pulse, only PTEN has to be enabled.

## 14.10 PWM Output Override

The PWM output override bits allow the user to manually drive the PWM I/O pins to specified logic states, independent of the duty cycle comparison units. The PWM override bits are useful when controlling various types of ECMs, like a BLDC motor.

OVDCOND and OVDCONS registers are used to define the PWM override options. The OVDCOND register contains six bits, POVD5:POVD0, that determine which PWM I/O pins will be overridden. The OVDCONS register contains six bits, POUT5:POUT0, that determine the state of the PWM I/O pins when a particular output is overridden via the POVD bits.

The POVD bits are active-low control bits. When the POVD bits are set, the corresponding POUT bit will have no effect on the PWM output. In other words, the pins corresponding to POVD bits that are set will have the duty PWM cycle set by the PDCx registers. When one of the POVD bits is cleared, the output on the corresponding PWM I/O pin will be determined by the state of the POUT bit. When a POUT bit is set, the PWM pin will be driven to its active state. When the POUT bit is cleared, the PWM pin will be driven to its inactive state.

### 14.10.1 COMPLEMENTARY OUTPUT MODE

The even numbered PWM I/O pins have override restrictions when a pair of PWM I/O pins are operating in the Complementary mode (PMDx = 0). In Complementary mode, if the even numbered pin is driven active by clearing the corresponding POVD bit and by setting the POUT bits in the OVDCOND and OVDCONS registers, the output signal is forced to be the complement of the odd numbered I/O pin in the pair (see Figure 14-2 for details).

### 14.10.2 OVERRIDE SYNCHRONIZATION

If the OSYNC bit in the PWMCON1 register is set, all output overrides performed via the OVDCOND and OVDCONS registers will be synchronized to the PWM time base. Synchronous output overrides will occur on the following conditions:

- When the PWM is in Edge-Aligned mode, synchronization occurs when PTMR is zero.
- When the PWM is in Center-Aligned mode, synchronization occurs when PTMR is zero and when the value of PTMR matches PTPER.

**Note 1:** In the Complementary mode, the even channel cannot be forced active by a Fault or override event when the odd channel is active. The even channel is always the complement of the odd channel, with dead-time inserted, before the odd channel can be driven to its active state as shown in Figure 14-20.

- 2: Dead time inserted in the PWM channels even when they are in Override mode.

**FIGURE 14-20: OVERRIDE BITS IN COMPLEMENTARY MODE**



# PIC18F1230/1330

## 14.10.3 OUTPUT OVERRIDE EXAMPLES

Figure 14-21 shows an example of a waveform that might be generated using the PWM output override feature. The figure shows a six-step commutation sequence for a BLDC motor. The motor is driven through a 3-phase inverter as shown in Figure 14-16. When the appropriate rotor position is detected, the PWM outputs are switched to the next commutation state in the sequence. In this example, the PWM outputs are driven to specific logic states. The OVDCOND and OVDCONS register values used to generate the signals in Figure 14-21 are given in Table 14-4.

The PWM Duty Cycle registers may be used in conjunction with the OVDCOND and OVDCONS registers. The Duty Cycle registers control the average voltage across the load and the OVDCOND and OVDCONS registers control the commutation sequence. Figure 14-22 shows the waveforms, while Table 14-4 and Table 14-5 show the OVDCOND and OVDCONS register values used to generate the signals.

### REGISTER 14-6: OVDCOND: OUTPUT OVERRIDE CONTROL REGISTER

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0
bit 7							bit 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

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

bit 5-0                      **POVD5:POVD0:** PWM Output Override bits

- 1 = Output on PWM I/O pin is controlled by the value in the Duty Cycle register and the PWM time base
- 0 = Output on PWM I/O pin is controlled by the value in the corresponding POUTx bit

### REGISTER 14-7: OVDCONS: OUTPUT STATE REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
bit 7							bit 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

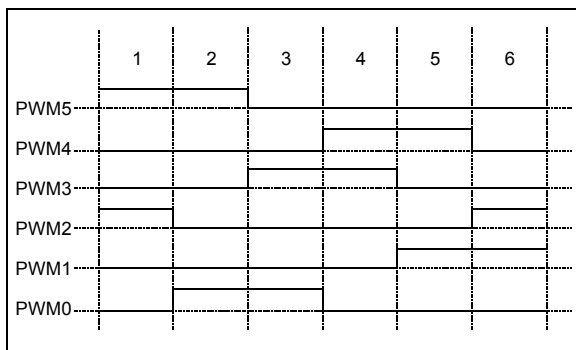
bit 7-6                      **Unimplemented:** Read as '0'

bit 5-0                      **POUT5:POUT0:** PWM Manual Output bits<sup>(1)</sup>

- 1 = Output on PWM I/O pin is active when the corresponding PWM output override bit is cleared
- 0 = Output on PWM I/O pin is inactive when the corresponding PWM output override bit is cleared

**Note 1:** With PWMs configured in complementary mode, even PWM (PWM0, 2, 4) outputs will be complementary of the odd PWM (PWM1, 3, 5) outputs, irrespective of the POUT bit setting.

**FIGURE 14-21: PWM OUTPUT OVERRIDE EXAMPLE #1**



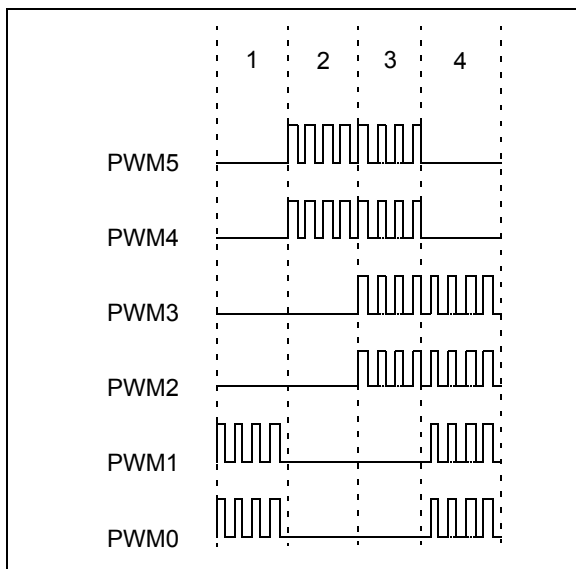
**TABLE 14-4: PWM OUTPUT OVERRIDE EXAMPLE #1**

State	OVDCOND (POVD)	OVDCONS (POUT)
1	00000000b	00100100b
2	00000000b	00100001b
3	00000000b	00001001b
4	00000000b	00011000b
5	00000000b	00010010b
6	00000000b	00000110b

**TABLE 14-5: PWM OUTPUT OVERRIDE EXAMPLE #2**

State	OVDCOND (POVD)	OVDCONS (POUT)
1	00000011b	00000000b
2	00110000b	00000000b
3	00111100b	00000000b
4	00001111b	00000000b

**FIGURE 14-22: PWM OUTPUT OVERRIDE EXAMPLE #2**



## 14.11 PWM Output and Polarity Control

There are three device Configuration bits associated with the PWM module that provide PWM output pin control defined in the CONFIG3L register. They are:

- HPOL
- LPOL
- PWMPIN

These three Configuration bits work in conjunction with the three PWM Enable bits (PWMEN2:PWMEN0) in the PWMCON0 register. The Configuration bits and PWM enable bits ensure that the PWM pins are in the correct states after a device Reset occurs.

### 14.11.1 OUTPUT PIN CONTROL

The PWMEN2:PWMEN0 control bits enable each PWM output pin as required in the application.

All PWM I/O pins are general purpose I/O. When a pair of pins is enabled for PWM output, the PORT and TRIS registers controlling the pins are disabled. Refer to Figure 14-23 for details.

### 14.11.2 OUTPUT POLARITY CONTROL

The polarity of the PWM I/O pins is set during device programming via the HPOL and LPOL Configuration bits in the CONFIG3L register. The HPOL Configuration bit sets the output polarity for the high side PWM outputs: PWM1, PWM3 and PWM5. The polarity is active-high when HPOL is set (= 1) and active-low when it is cleared (= 0).

The LPOL Configuration bit sets the output polarity for the low side PWM outputs: PWM0, PWM2 and PWM4. As with HPOL, they are active-high when LPOL is set and active-low when cleared.

All output signals generated by the PWM module are referenced to the polarity control bits, including those generated by Fault inputs or manual override (see **Section 14.10 “PWM Output Override”**).

The default polarity Configuration bits have the PWM I/O pins in active-high output polarity.

# PIC18F1230/1330

**FIGURE 14-23: PWM I/O PIN BLOCK DIAGRAM**



### 14.11.3 PWM OUTPUT PIN RESET STATES

The PWMPIN Configuration bit determines the PWM output pins to be PWM output pins, or digital I/O pins, after the device comes out of Reset. If the PWMPIN Configuration bit is unprogrammed (default), the PWMEN2:PWMEN0 control bits will be cleared on a device Reset. Consequently, all PWM outputs will be tri-stated and controlled by the corresponding PORT and TRIS registers. If the PWMPIN Configuration bit is programmed low, the PWMEN2:PWMEN0 control bits will be set to '100' on a device Reset:

All PWM pins will be enabled for PWM output and will have the output polarity defined by the HPOL and LPOL Configuration bits.

## 14.12 PWM Fault Input

There is one Fault input associated with the PWM module. The main purpose of the input Fault pin is to disable the PWM output signals and drive them into an inactive state. The action of the Fault input is performed

directly in hardware so that when a Fault occurs, it can be managed quickly and the PWMs outputs are put into an inactive state to save the power devices connected to the PWMs.

The PWM Fault input is  $\overline{FLT\bar{A}}$ , which can come from I/O pins, the CPU or another module. The  $\overline{FLT\bar{A}}$  pin is an active-low input so it is easy to "OR" many sources to the same input.

The FLTCONFIG register (Register 14-8) defines the settings of the  $\overline{FLT\bar{A}}$  input.

**Note:** The inactive state of the PWM pins is dependent on the HPOL and LPOL Configuration bit settings, which define the active and inactive state for PWM outputs.

### 14.12.1 FAULT PIN ENABLE BIT

By setting the bit FLTAEN in the FLTCONFIG register, the corresponding Fault input is enabled. If FLTAEN bit is cleared, then the Fault input has no effect on the PWM module.

## 14.12.2 FAULT INPUT MODE

The FLTAMOD bit in the FLTCONFIG register determines whether the PWM I/O pins are deactivated when they are overridden by a Fault input.

FLTAS bit in the FLTCONFIG register gives the status of the Fault A input.

The Fault input has two modes of operation:

- **Inactive Mode (FLTAMOD = 0)**

This is a catastrophic Fault Management mode. When the Fault occurs in this mode, the PWM outputs are deactivated. The PWM pins will remain in Inactivated mode until the Fault is cleared (Fault input is driven high) and the corresponding Fault status bit has been cleared in software. The PWM outputs are enabled immediately at the beginning of the following PWM period, after Fault status bit (FLTAS) is cleared.

- **Cycle-by-Cycle Mode (FLTAMOD = 1)**

When the Fault occurs in this mode, the PWM outputs are deactivated. The PWM outputs will remain in the defined Fault states (all PWM outputs inactive) for as long as the Fault pin is held low. After the Fault pin is driven high, the PWM outputs will return to normal operation at the beginning of the following PWM period and the FLTAS bit is automatically cleared.

## 14.12.3 PWM OUTPUTS WHILE IN FAULT CONDITION

While in the Fault state (i.e.,  $\overline{FLTA}$  input is active), the PWM output signals are driven into their inactive states.

## 14.12.4 PWM OUTPUTS IN DEBUG MODE

The BRFFEN bit in the FLTCONFIG register controls the simulation of Fault condition when a breakpoint is hit, while debugging the application using an In-Circuit Debugger (ICD). Setting the BRFFEN bit to high enables the Fault condition on breakpoint, thus driving the PWM outputs to inactive state. This is done to avoid any continuous keeping of status on the PWM pin, which may result in damage of the power devices connected to the PWM outputs.

If BRFFEN = 0, the Fault condition on breakpoint is disabled.

**Note:** It is highly recommended to enable the Fault condition on breakpoint if a debugging tool is used while developing the firmware and the high-power circuitry is used. When the device is ready to program after debugging the firmware, the BRFFEN bit can be disabled.

## REGISTER 14-8: FLTCONFIG: FAULT CONFIGURATION REGISTER

R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
BRFFEN	—	—	—	—	FLTAS	FLTAMOD	FLTAEN
bit 7						bit 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

bit 7	<p><b>BRFFEN:</b> Breakpoint Fault Enable bit</p> <p>1 = Enable Fault condition on a breakpoint</p> <p>0 = Disable Fault condition</p>
bit 6-3	<p><b>Unimplemented:</b> Read as '0'</p>
bit 2	<p><b>FLTAS:</b> Fault A Status bit</p> <p>1 = <math>\overline{FLTA}</math> is asserted:</p> <p style="padding-left: 20px;">if FLTAMOD = 0, cleared by the user;</p> <p style="padding-left: 20px;">if FLTAMOD = 1, cleared automatically at beginning of the new period when <math>\overline{FLTA}</math> is deasserted</p> <p>0 = No Fault</p>
bit 1	<p><b>FLTAMOD:</b> Fault A Mode bit</p> <p>1 = Cycle-by-Cycle mode: Pins are inactive for the remainder of the current PWM period or until <math>\overline{FLTA}</math> is deasserted; FLTAS is cleared automatically</p> <p>0 = Inactive mode: Pins are deactivated (catastrophic failure) until <math>\overline{FLTA}</math> is deasserted and FLTAS is cleared by the user only</p>
bit 0	<p><b>FLTAEN:</b> Fault A Enable bit</p> <p>1 = Enable Fault A</p> <p>0 = Disable Fault A</p>

## 14.13 PWM Update Lockout

For a complex PWM application, the user may need to write up to four Duty Cycle registers and the PWM Time Base Period Register, PTPER, at a given time. In some applications, it is important that all buffer registers be written before the new duty cycle and period values are loaded for use by the module.

A PWM update lockout feature may optionally be enabled so the user may specify when new duty cycle buffer values are valid. The PWM update lockout feature is enabled by setting the control bit, UDIS, in the PWMCON1 register. This bit affects all Duty Cycle Buffer registers and the PWM Time Base Period register, PTPER.

To perform a PWM update lockout:

1. Set the UDIS bit.
2. Write all Duty Cycle registers and PTPER, if applicable.
3. Clear the UDIS bit to re-enable updates.
4. With this, when UDIS bit is cleared, the buffer values will be loaded to the actual registers. This makes a synchronous loading of the registers.

## 14.14 PWM Special Event Trigger

The PWM module has a Special Event Trigger capability that allows A/D conversions to be synchronized to the PWM time base. The A/D sampling and conversion time may be programmed to occur at any point within the PWM period. The Special Event Trigger allows the user to minimize the delay between the time when A/D conversion results are acquired and the time when the duty cycle value is updated.

The PWM 16-bit Special Event Trigger register, SEVTCMP (high and low), and five control bits in the PWMCON1 register are used to control its operation.

The PTMR value for which a Special Event Trigger should occur is loaded into the SEVTCMP register pair. SEVTDIR bit in PWMCON1 register specifies the counting phase when the PWM time base is in a Continuous Up/Down Count mode.

If the SEVTDIR bit is cleared, the Special Event Trigger will occur on the upward counting cycle of the PWM time base. If SEVTDIR is set, the Special Event Trigger will occur on the downward count cycle of the PWM time base. The SEVTDIR bit only effects this operation when the PWM timer is in the Continuous Up/Down Count mode.

<b>Note:</b> The Special Event Trigger will take place only for non-zero values in the SEVTCMP registers.
---

### 14.14.1 SPECIAL EVENT TRIGGER ENABLE

The PWM module will always produce Special Event Trigger pulses. This signal may optionally be used by the A/D module. Refer to **Chapter 16.0 "10-Bit Analog-to-Digital Converter (A/D) Module"** for details.

### 14.14.2 SPECIAL EVENT TRIGGER POSTSCALER

The PWM Special Event Trigger has a postscaler that allows a 1:1 to 1:16 postscale ratio. The postscaler is configured by writing the SEVOPS3:SEVOPS0 control bits in the PWMCON1 register.

The Special Event Trigger output postscaler is cleared on any write to the SEVTCMP register pair, or on any device Reset.



**TABLE 14-6: REGISTERS ASSOCIATED WITH THE POWER CONTROL PWM MODULE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
IPR3	—	—	—	PTIP	—	—	—	—	49
PIE3	—	—	—	PTIE	—	—	—	—	49
PIR3	—	—	—	PTIF	—	—	—	—	49
PTCON0	PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0	49
PTCON1	PTEN	PTDIR	—	—	—	—	—	—	49
PTMRL <sup>(1)</sup>	PWM Time Base Register (lower 8 bits)								49
PTMRH <sup>(1)</sup>	—	—	—	—	PWM Time Base Register (upper 4 bits)				49
PTPERL <sup>(1)</sup>	PWM Time Base Period Register (lower 8 bits)								49
PTPERH <sup>(1)</sup>	—	—	—	—	PWM Time Base Period Register (upper 4 bits)				49
SEVTCMPL <sup>(1)</sup>	PWM Special Event Compare Register (lower 8 bits)								49
SEVTCMPH <sup>(1)</sup>	—	—	—	—	PWM Special Event Compare Register (upper 4 bits)				50
PWMCON0	—	PWMEN2 <sup>(2)</sup>	PWMEN1 <sup>(2)</sup>	PWMEN0 <sup>(2)</sup>	—	PMOD2	PMOD1	PMOD0	50
PWMCON1	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC	50
DTCON	DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0	50
FLTCONFIG	BRFEN	—	—	—	—	FLTAS	FLTAMOD	FLTAEN	49
OVDCOND	—	—	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0	50
OVDCONS	—	—	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0	50
PDC0L <sup>(1)</sup>	PWM Duty Cycle #0L Register (lower 8 bits)								49
PDC0H <sup>(1)</sup>	—	—	PWM Duty Cycle #0H Register (upper 6 bits)						49
PDC1L <sup>(1)</sup>	PWM Duty Cycle #1L Register (lower 8 bits)								49
PDC1H <sup>(1)</sup>	—	—	PWM Duty Cycle #1H Register (upper 6 bits)						49
PDC2L <sup>(1)</sup>	PWM Duty Cycle #2L Register (lower 8 bits)								49
PDC2H <sup>(1)</sup>	—	—	PWM Duty Cycle #2H Register (upper 6 bits)						49

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used with the Power Control PWM.

**Note 1:** Double-buffered register pairs. Refer to text for explanation of how these registers are read and written to.

**2:** Reset condition of PWMEN bits depends on the PWMPIN Configuration bit.

# PIC18F1230/1330

---

NOTES:

## 15.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of the two serial I/O modules. (Generically, the USART is also known as a Serial Communications Interface or SCI.) The EUSART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also 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 Enhanced USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These features make it ideally suited for use in Local Interconnect Network bus (LIN/J2602 bus) systems.

The EUSART can be configured in the following modes:

- Asynchronous (full-duplex) with:
  - Auto-Wake-up on Character Reception
  - Auto-Baud Calibration
  - 12-Bit Break Character Transmission
- Synchronous – Master (half-duplex) with Selectable Clock Polarity
- Synchronous – Slave (half-duplex) with Selectable Clock Polarity

The pins of the Enhanced USART are multiplexed with PORTA. In order to configure RA2/TX/CK and RA3/RX/DT as an EUSART:

- bit SPEN (RCSTA<7>) must be set (= 1)
- bit TRISA<3> must be set (= 1)
- bit TRISA<2> must be set (= 1)

**Note:** The EUSART control will automatically reconfigure the pin from input to output as needed.

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

These are detailed on the following pages in Register 15-1, Register 15-2 and Register 15-3, respectively.

# PIC18F1230/1330

## REGISTER 15-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN <sup>(1)</sup>	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 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

- 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<sup>(1)</sup>  
1 = Transmit enabled  
0 = Transmit disabled
- bit 4      **SYNC:** EUSART Mode Select bit  
1 = Synchronous mode  
0 = Asynchronous mode
- bit 3      **SENDB:** Send Break Character bit  
Asynchronous mode:  
1 = Send Sync Break on next transmission (cleared by hardware upon completion)  
0 = Sync Break transmission completed  
Synchronous mode:  
Don't care.
- 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.

**Note 1:** SREN/CREN overrides TXEN in Sync mode.

## REGISTER 15-2: 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

### 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

- 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 (held in Reset)
- 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:  
 Don't care.
- bit 4      **CREN:** Continuous Receive Enable bit  
Asynchronous mode:  
 1 = Enables receiver  
 0 = Disables receiver  
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 loads 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  
Asynchronous mode 9-bit (RX9 = 0):  
 Don't care.
- bit 2      **FERR:** Framing Error bit  
 1 = Framing error (can be updated by reading RCREG register and receiving 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  
 This can be address/data bit or a parity bit and must be calculated by user firmware.

# PIC18F1230/1330

## REGISTER 15-3: BAUDCON: BAUD RATE CONTROL REGISTER

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 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

- bit 7      **ABDOVF**: Auto-Baud Acquisition Rollover Status bit  
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)  
0 = No BRG rollover has occurred
- bit 6      **RCIDL**: Receive Operation Idle Status bit  
1 = Receive operation is Idle  
0 = Receive operation is active
- bit 5      **RXDTP**: Received Data Polarity Select bit  
Asynchronous mode:  
1 = RX data is inverted  
0 = RX data is not inverted  
Synchronous mode:  
Unused in this mode.
- bit 4      **TXCKP**: Clock and Data Polarity Select bit  
Asynchronous mode:  
1 = Idle state for transmit (TX) is a low level  
0 = Idle state for transmit (TX) is a high level  
Synchronous mode:  
1 = Idle state for clock (CK) is a high level  
0 = Idle state for clock (CK) is a low level
- bit 3      **BRG16**: 16-Bit Baud Rate Register Enable bit  
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG  
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored
- bit 2      **Unimplemented**: Read as '0'
- bit 1      **WUE**: Wake-up Enable bit  
Asynchronous mode:  
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge  
0 = RX pin not monitored or rising edge detected  
Synchronous mode:  
Unused in this mode.
- bit 0      **ABDEN**: Auto-Baud Detect Enable bit  
Asynchronous mode:  
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion  
0 = Baud rate measurement disabled or completed  
Synchronous mode:  
Unused in this mode.

## 15.1 Baud Rate Generator (BRG)

The BRG is a dedicated 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCON<3>) selects 16-bit mode.

The SPBRGH:SPBRG register pair controls the period of a free-running timer. In Asynchronous mode, bits BRGH (TXSTA<2>) and BRG16 (BAUDCON<3>) also control the baud rate. In Synchronous mode, BRGH is ignored. Table 15-1 shows the formula for computation of the baud rate for different EUSART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 15-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 15-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 15-2. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGH:SPBRG registers 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.

**Note:** A BRG value of '0' is not supported.

### 15.1.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRG register pair.

### 15.1.2 SAMPLING

The data on the RX 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 when SYNC is clear or when both BRG16 and BRGH are not set. The data on the RX pin is sampled once when SYNC is set or when BRG16 and BRGH are both set.

**TABLE 15-1: BAUD RATE FORMULAS**

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	$F_{OSC}/[64 (n + 1)]$
0	0	1	8-bit/Asynchronous	$F_{OSC}/[16 (n + 1)]$
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	$F_{OSC}/[4 (n + 1)]$
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

**Legend:** x = Don't care, n = value of SPBRGH:SPBRG register pair

# PIC18F1230/1330

## EXAMPLE 15-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

Desired Baud Rate =  $F_{OSC}/(64 ([SPBRGH:SPBRG] + 1))$

Solving for SPBRGH:SPBRG:

$$X = ((F_{OSC}/\text{Desired Baud Rate})/64) - 1$$

$$= ((16000000/9600)/64) - 1$$

$$= [25.042] = 25$$

Calculated Baud Rate =  $16000000/(64 (25 + 1))$

$$= 9615$$

Error =  $(\text{Calculated Baud Rate} - \text{Desired Baud Rate})/\text{Desired Baud Rate}$

$$= (9615 - 9600)/9600 = 0.16\%$$

TABLE 15-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	48
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	48
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	48
SPBRGH	EUSART Baud Rate Generator Register High Byte								48
SPBRG	EUSART Baud Rate Generator Register Low Byte								48

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the BRG.



**TABLE 15-3: BAUD RATES FOR ASYNCHRONOUS MODES**

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.16	207	0.300	-0.16	103	0.300	-0.16	51
1.2	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

# PIC18F1230/1330

**TABLE 15-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)**

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	0.300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1.201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2.403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.04	832	0.300	-0.16	415	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	0.300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1.200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2.400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9.615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19.230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57.142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117.647	-2.12	16

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.01	3332	0.300	-0.04	1665	0.300	-0.04	832
1.2	1.200	0.04	832	1.201	-0.16	415	1.201	-0.16	207
2.4	2.404	0.16	415	2.403	-0.16	207	2.403	-0.16	103
9.6	9.615	0.16	103	9.615	-0.16	51	9.615	-0.16	25
19.2	19.231	0.16	51	19.230	-0.16	25	19.230	-0.16	12
57.6	58.824	2.12	16	55.555	3.55	8	—	—	—
115.2	111.111	-3.55	8	—	—	—	—	—	—

## 15.1.3 AUTO-BAUD RATE DETECT

The Enhanced USART module supports the automatic detection and calibration of baud rate. This feature is active only in Asynchronous mode and while the WUE bit is clear.

The automatic baud rate measurement sequence (Figure 15-1) begins whenever a Start bit is received and the ABDEN bit is set. The calculation is self-averaging.

In the Auto-Baud Rate Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. In ABD mode, the internal Baud Rate Generator is used as a counter to time the bit period of the incoming serial byte stream.

Once the ABDEN bit is set, the state machine will clear the BRG and look for a Start bit. The Auto-Baud Rate Detect must receive a byte with the value 55h (ASCII “U”, which is also the LIN/J2602 bus Sync character) in order to calculate the proper bit rate. The measurement is taken over both a low and a high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRG begins counting up, using the preselected clock source on the first rising edge of RX. After eight bits on the RX pin or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGH:SPBRG register pair. Once the 5th edge is seen (this should correspond to the Stop bit), the ABDEN bit is automatically cleared.

If a rollover of the BRG occurs (an overflow from FFFFh to 0000h), the event is trapped by the ABDOVF status bit (BAUDCON<7>). It is set in hardware by BRG rollovers and can be set or cleared by the user in software. ABD mode remains active after rollover events and the ABDEN bit remains set (Figure 15-2).

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock can be configured by the BRG16 and BRGH bits. The BRG16 bit must be set to use both SPBRG1 and SPBRGH1 as a 16-bit counter. This allows the user to verify that no carry occurred for 8-bit modes by checking for 00h in the SPBRGH register. Refer to Table 15-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSART state machine is held in Idle. The RCIF interrupt is set once the fifth rising edge on RX is detected. The value in the RCREG needs to be read to clear the RCIF interrupt. The contents of RCREG should be discarded.

**Note 1:** If the WUE bit is set with the ABDEN bit, Auto-Baud Rate Detection will occur on the byte *following* the Break character.

**2:** It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the Auto-Baud Rate Detection feature.

**3:** To maximize baud rate range, it is recommended to set the BRG16 bit if the auto-baud feature is used.

**TABLE 15-4: BRG COUNTER CLOCK RATES**

BRG16	BRGH	BRG Counter Clock
0	0	Fosc/512
0	1	Fosc/128
1	0	Fosc/128
1	1	Fosc/32

### 15.1.3.1 ABD and EUSART Transmission

Since the BRG clock is reversed during ABD acquisition, the EUSART transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREG cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.

# PIC18F1230/1330

**FIGURE 15-1: AUTOMATIC BAUD RATE CALCULATION**



**FIGURE 15-2: BRG OVERFLOW SEQUENCE**



## 15.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA<4>). In this mode, the EUSART 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/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART'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 and BRG16 bits (TXSTA<2> and BAUDCON<3>). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.

In Asynchronous mode, clock polarity is selected with the TXCKP bit (BAUDCON<4>). Setting TXCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low. Data polarity is selected with the RXDTP bit (BAUDCON<5>).

Setting RXDTP inverts data on RX, while clearing the bit has no effect on received data.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-Bit Break Character Transmit
- Auto-Baud Rate Detection

### 15.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 15-3. 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 Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available).

Once the TXREG register transfers the data to the TSR register (occurs in one T<sub>cy</sub>), the TXREG register is empty and the TXIF flag bit (PIR1<4>) is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF will be set regardless of the state of TXIE; it cannot be cleared in software. TXIF is also not cleared immediately upon loading TXREG but becomes valid in the second instruction cycle following the load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

While TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. 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.

To set up an Asynchronous Transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
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).
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

# PIC18F1230/1330

**FIGURE 15-3: EUSART TRANSMIT BLOCK DIAGRAM**



**FIGURE 15-4: ASYNCHRONOUS TRANSMISSION**



**FIGURE 15-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**



**TABLE 15-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	48
TXREG	EUSART Transmit Register								48
TXSTA	CSRC	TX9	TXEN	SYNC	SENDER	BRGH	TRMT	TX9D	48
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	48
SPBRGH	EUSART Baud Rate Generator Register High Byte								48
SPBRG	EUSART Baud Rate Generator Register Low Byte								48

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

# PIC18F1230/1330

## 15.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 15-6. The data is received on the RX 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.

To set up an Asynchronous Reception:

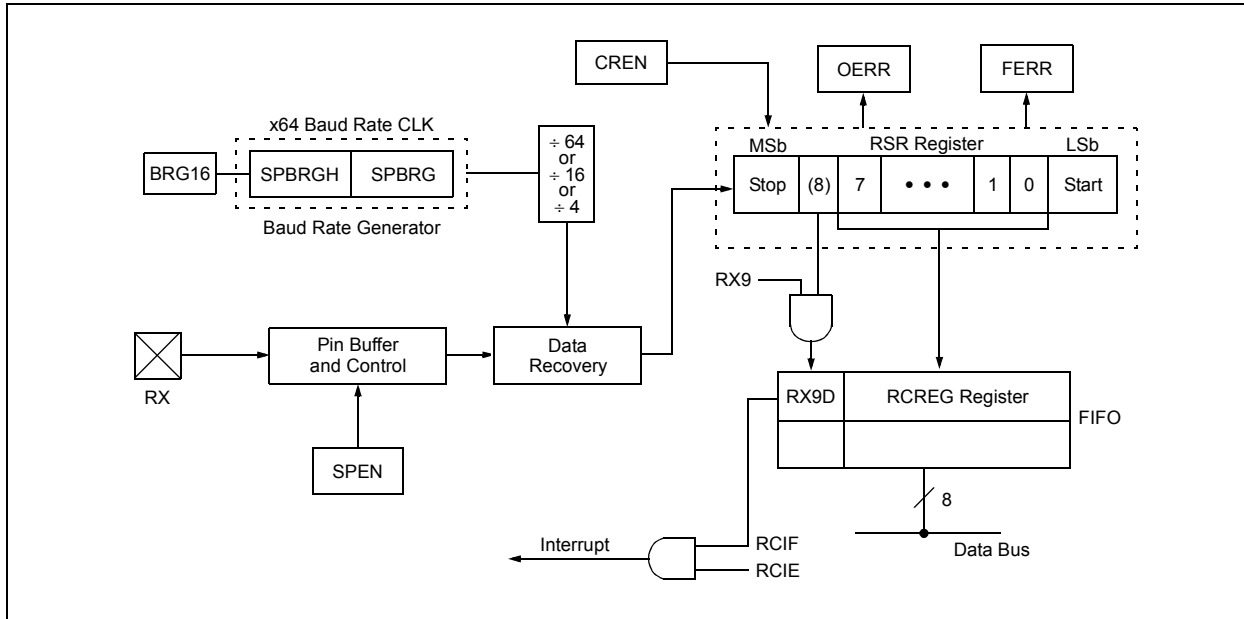
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCIE.
4. If 9-bit 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 9th 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.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

## 15.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
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 15-6: EUSART RECEIVE BLOCK DIAGRAM**





**FIGURE 15-7: ASYNCHRONOUS RECEPTION**



**TABLE 15-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	48
RCREG	EUSART Receive Register								48
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	48
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	48
SPBRGH	EUSART Baud Rate Generator Register High Byte								48
SPBRG	EUSART Baud Rate Generator Register Low Byte								48

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

### 15.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RX/DT line while the EUSART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCON<1>). Once set, the typical receive sequence on RX/DT is disabled and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a Wake-up Signal character for the LIN/J2602 protocol.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 15-8) and asynchronously if the device is in Sleep mode (Figure 15-9). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

# PIC18F1230/1330

## 15.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false End-of-Character and cause data or framing errors. To work properly, therefore, the initial characters in the transmission must be all '0's. This can be 00h (8 bits) for standard RS-232 devices or 000h (12 bits) for LIN/J2602 bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., XT or HS mode). The Sync Break (or Wake-up Signal) character must be of sufficient length and be followed by a sufficient interval to allow enough time for the selected oscillator to start and provide proper initialization of the EUSART.

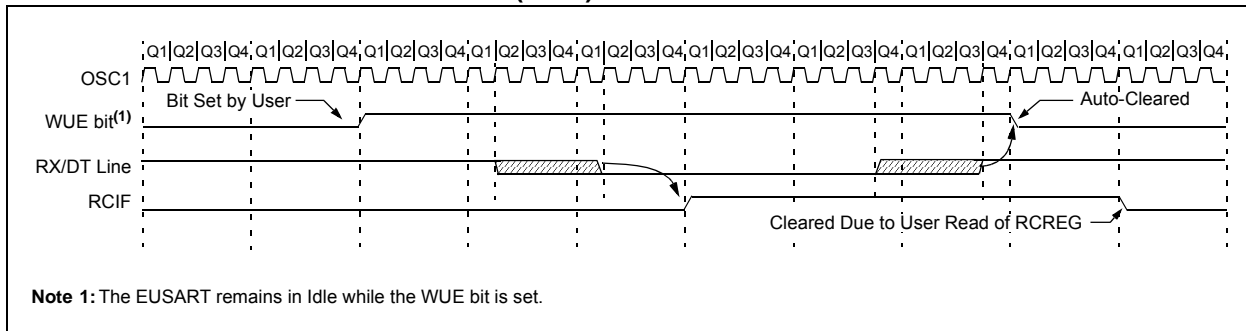
## 15.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

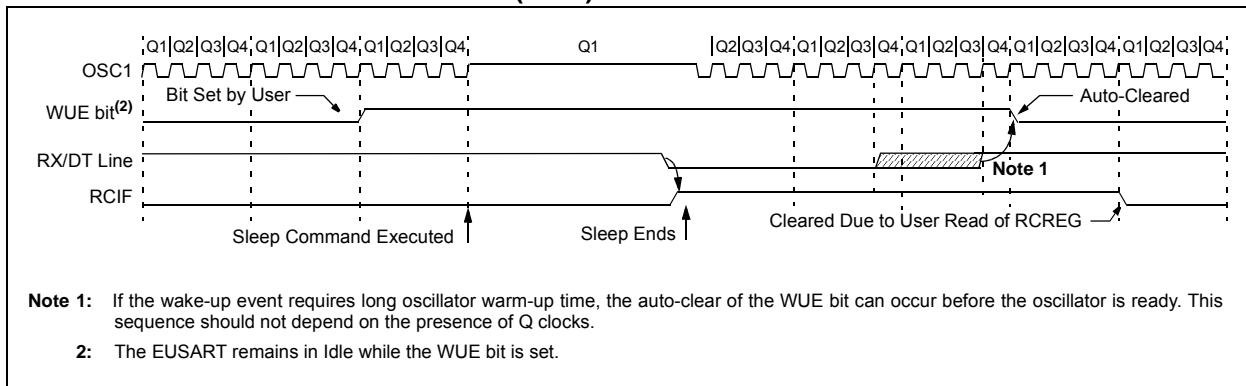
The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 15-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**



**FIGURE 15-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**



## 15.2.5 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN/J2602 bus standard. The Break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN/J2602 specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 15-10 for the timing of the Break character sequence.

### 15.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN/J2602 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.

3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

## 15.2.6 RECEIVING A BREAK CHARACTER

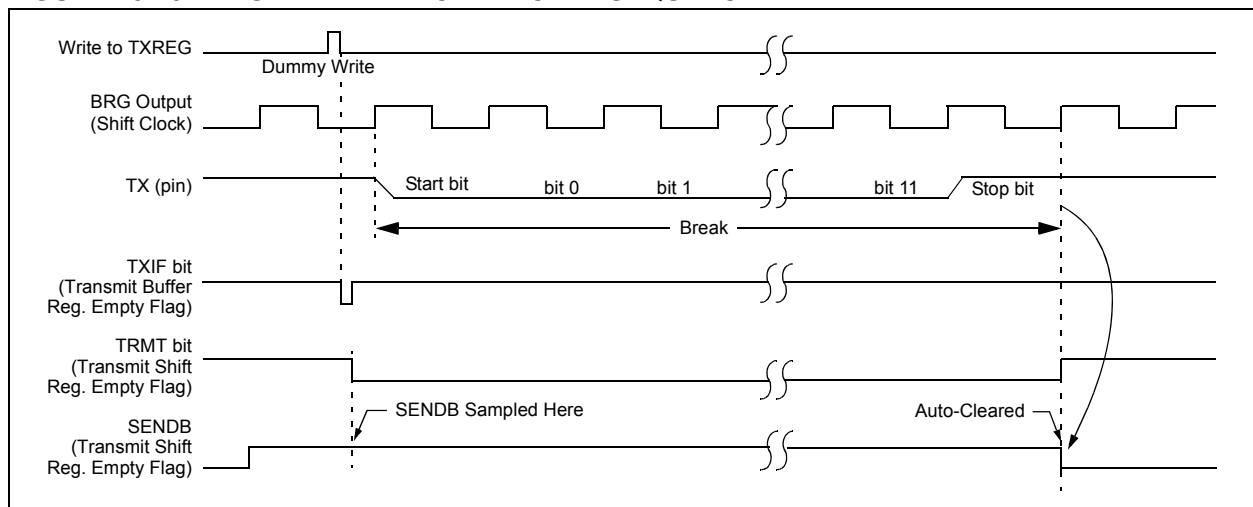
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 15.2.4 "Auto-wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABDEN bit once the TXIF interrupt is observed.

**FIGURE 15-10: SEND BREAK CHARACTER SEQUENCE**



# PIC18F1230/1330

## 15.3 EUSART Synchronous Master Mode

The Master mode indicates that the processor transmits the master clock on the CK line. The Synchronous Master mode is entered by setting the CSRC bit (TXSTA<7>). In this 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<4>). In addition, enable bit, SPEN (RCSTA<7>), is set in order to configure the TX and RX pins to CK (clock) and DT (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK line. Clock polarity is selected with the SCKP bit (BAUDCON<4>). Setting SCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low.

### 15.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 15-3. 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  $T_{CY}$ ), the TXREG is empty and the TXIF flag bit (PIR1<4>) is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF is set regardless of the state of enable bit, TXIE; it 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<1>), 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.

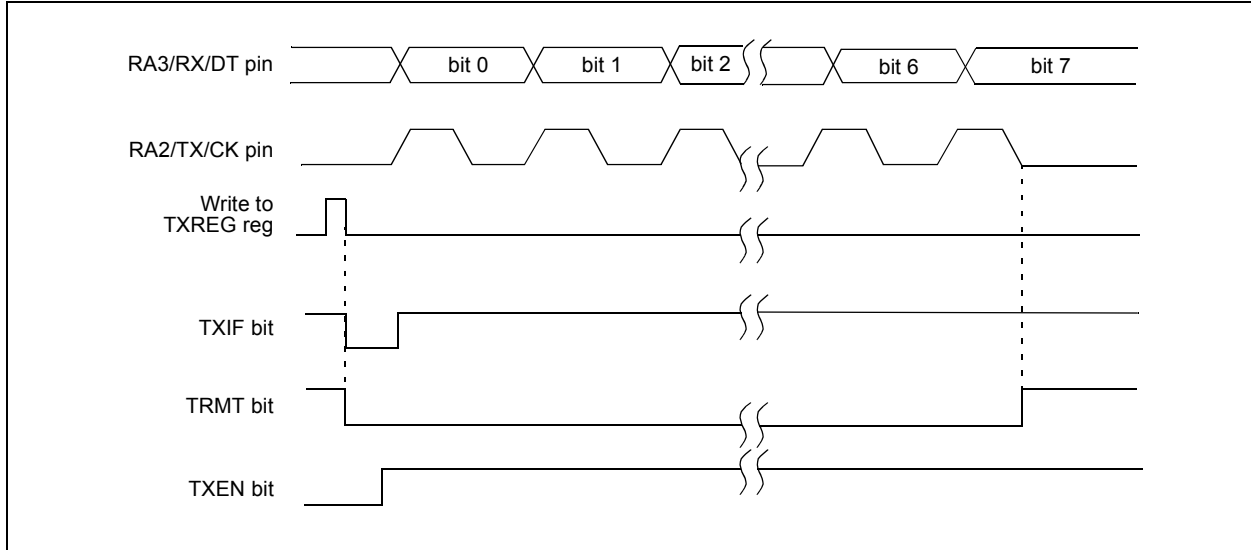
To set up a Synchronous Master Transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
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.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 15-11: SYNCHRONOUS TRANSMISSION**



**FIGURE 15-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**



**TABLE 15-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	48
TXREG	EUSART Transmit Register								48
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	48
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	48
SPBRGH	EUSART Baud Rate Generator Register High Byte								48
SPBRG	EUSART Baud Rate Generator Register Low Byte								48

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

# PIC18F1230/1330

## 15.3.2 EUSART SYNCHRONOUS MASTER RECEPTION

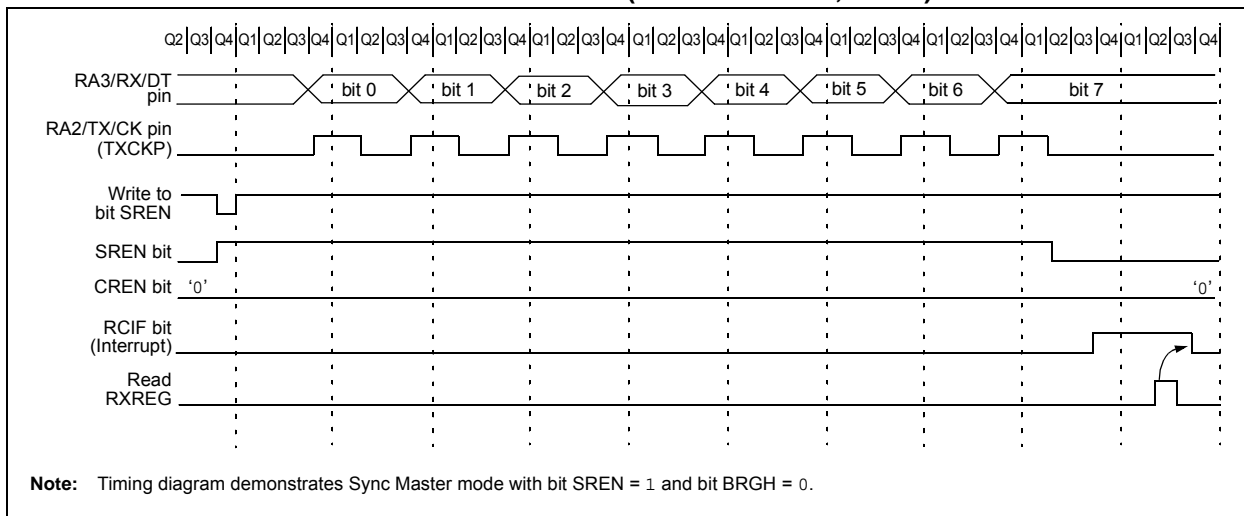
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTA<5>), or the Continuous Receive Enable bit, CREN (RCSTA<4>). Data is sampled on the RX pin on the falling edge of the clock.

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.

To set up a Synchronous Master Reception:

1. If any error occurred, clear the error by clearing bit, CREN.
2. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.
3. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
4. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
5. Ensure bits, CREN and SREN, are clear.
6. If the signal from the CK pin is to be inverted, set the TXCKP bit.
7. If interrupts are desired, set enable bit, RCIE.
8. If 9-bit reception is desired, set bit, RX9.
9. If a single reception is required, set bit, SREN. For continuous reception, set bit, CREN.
10. Interrupt flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if the enable bit, RCIE, was set.
11. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
12. Read the 8-bit received data by reading the RCREG register.

**FIGURE 15-13: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**



**TABLE 15-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	48
RCREG	EUSART Receive Register								48
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	48
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	48
SPBRGH	EUSART Baud Rate Generator Register High Byte								48
SPBRG	EUSART Baud Rate Generator Register Low Byte								48

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception

## 15.4 EUSART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit, CSRC (TXSTA<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

### 15.4.1 EUSART SYNCHRONOUS SLAVE TRANSMISSION

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 the 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 now 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.

To set 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 the signal from the CK pin is to be inverted, set the TXCKP bit.
5. If 9-bit transmission is desired, set bit, TX9.
6. Enable the transmission by setting enable bit, TXEN.
7. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
8. Start transmission by loading data to the TXREG register.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 15-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	48
TXREG	EUSART Transmit Register								48
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	48
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	48
SPBRGH	EUSART Baud Rate Generator Register High Byte								48
SPBRG	EUSART Baud Rate Generator Register Low Byte								48

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

# PIC18F1230/1330

## 15.4.2 EUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep, or any Idle mode and bit SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREG register; if the RCIE enable bit is set, the interrupt generated will wake the chip from the low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set 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 the signal from the CK pin is to be inverted, set the TXCKP bit.
4. If 9-bit reception is desired, set bit, RX9.
5. To enable reception, set enable bit, CREN.
6. Flag bit, RCIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCIE, was set.
7. Read the RCSTA register to get the 9th 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 bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 15-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	48
RCREG	EUSART Receive Register								48
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	48
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	48
SPBRGH	EUSART Baud Rate Generator Register High Byte								48
SPBRG	EUSART Baud Rate Generator Register Low Byte								48

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.



## 16.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has 4 inputs for the 18/20/28-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number in PIC18F1230/1330 devices.

The module has five registers:

- A/D Result Register High Byte (ADRESH)
- A/D Result Register Low Byte (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 16-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 16-2, configures the functions of the port pins. The ADCON2 register, shown in Register 16-3, configures the A/D clock source, programmed acquisition time and justification.

### REGISTER 16-1: ADCON0: A/D CONTROL REGISTER 0

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTEN	—	—	—	CHS1	CHS0	GO/DONE	ADON
bit 7							bit 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

- bit 7      **SEVTEN:** Special Event Trigger Enable bit  
 1 = Special Event Trigger from Power Control PWM module is enabled  
 0 = Special Event Trigger from Power Control PWM module is disabled (default)
- bit 6-4    **Unimplemented:** Read as '0'
- bit 3-2    **CHS1:CHS0:** Analog Channel Select bits  
 00 = Channel 0 (AN0)  
 01 = Channel 1 (AN1)  
 10 = Channel 2 (AN2)  
 11 = Channel 3 (AN3)
- bit 1      **GO/DONE:** A/D Conversion Status bit  
When ADON = 1:  
 1 = A/D conversion in progress  
 0 = A/D Idle
- bit 0      **ADON:** A/D On bit  
 1 = A/D Converter module is enabled  
 0 = A/D Converter module is disabled

# PIC18F1230/1330

## REGISTER 16-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 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

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

bit 4      **VCFG0:** Voltage Reference Configuration bit (VREF+ source)

1 = Positive reference for the A/D is VREF+

0 = Positive reference for the A/D is AVDD

bit 3      **PCFG3:** A/D Port Configuration bit for RA6/AN3

0 = Port is configured as AN3

1 = Port is configured as RA6

bit 2      **PCFG2:** A/D Port Configuration bit for RA4/AN2

0 = Port is configured as AN2

1 = Port is configured as RA4

bit 1      **PCFG1:** A/D Port Configuration bit for RA1/AN1

0 = Port is configured as AN1

1 = Port is configured as RA1

bit 0      **PCFG0:** A/D Port Configuration bit for RA0/AN0

0 = Port is configured as AN0

1 = Port is configured as RA0

## REGISTER 16-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 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

- bit 7      **ADFM:** A/D Result Format Select bit  
           1 = Right justified  
           0 = Left justified
- bit 6      **Unimplemented:** Read as '0'
- bit 5-3    **ACQT2:ACQT0:** A/D Acquisition Time Select bits  
           111 = 20 TAD  
           110 = 16 TAD  
           101 = 12 TAD  
           100 = 8 TAD  
           011 = 6 TAD  
           010 = 4 TAD  
           001 = 2 TAD  
           000 = 0 TAD<sup>(1)</sup>
- bit 2-0    **ADCS2:ADCS0:** A/D Conversion Clock Select bits  
           111 = FRC (clock derived from A/D RC oscillator)<sup>(1)</sup>  
           110 = FOSC/64  
           101 = FOSC/16  
           100 = FOSC/4  
           011 = FRC (clock derived from A/D RC oscillator)<sup>(1)</sup>  
           010 = FOSC/32  
           001 = FOSC/8  
           000 = FOSC/2

**Note 1:** If the A/D FRC clock source is selected, a delay of one T<sub>CY</sub> (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

# PIC18F1230/1330

The analog reference voltage is software selectable to the device's positive supply voltage (VDD), or the voltage level on the RA4/T0CKI/AN2/VREF+ pin.

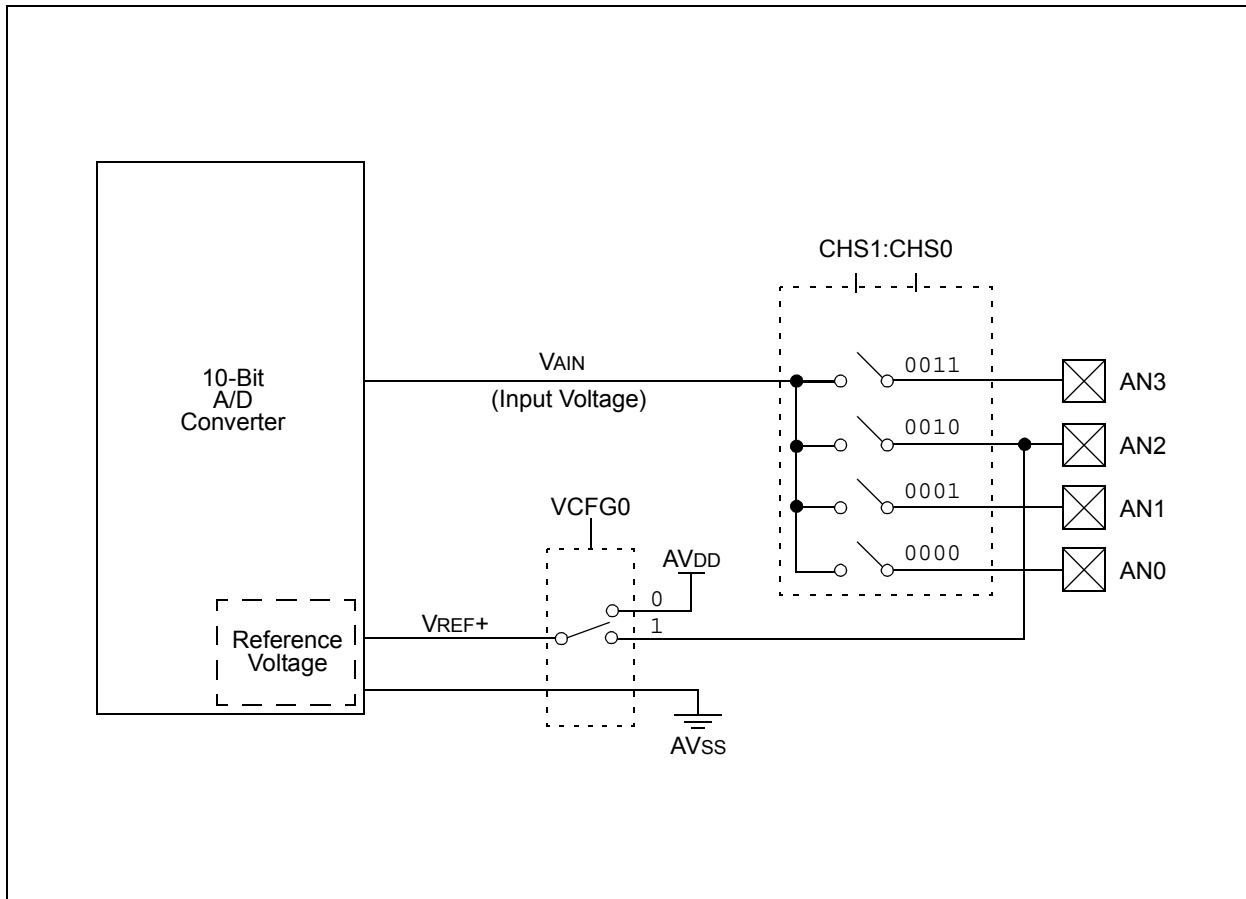
The A/D Converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D Converter's internal RC oscillator.

The output of the sample and hold is the input into the A/D 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 in progress is aborted.

Each port pin associated with the A/D Converter can be configured as an analog input 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 register pair, the GO/DONE bit (ADCON0 register) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 16-1.

**FIGURE 16-1: A/D BLOCK DIAGRAM**



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

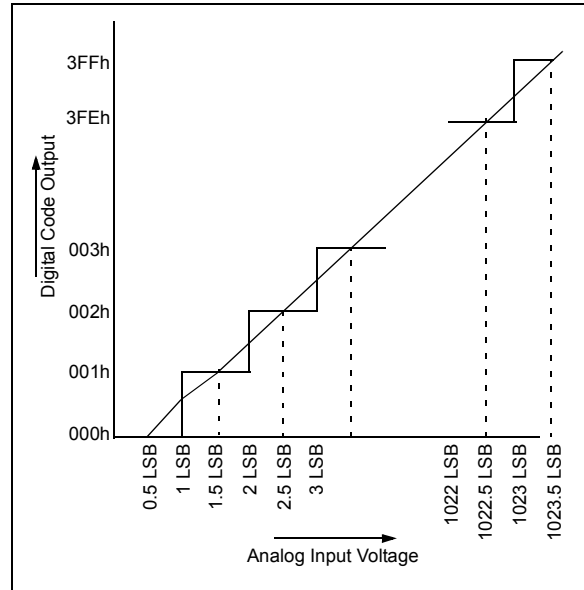
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs. To determine acquisition time, see **Section 16.2 “A/D Acquisition Requirements”**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/DONE bit and the actual start of the conversion.

The following steps should be followed to perform an A/D conversion:

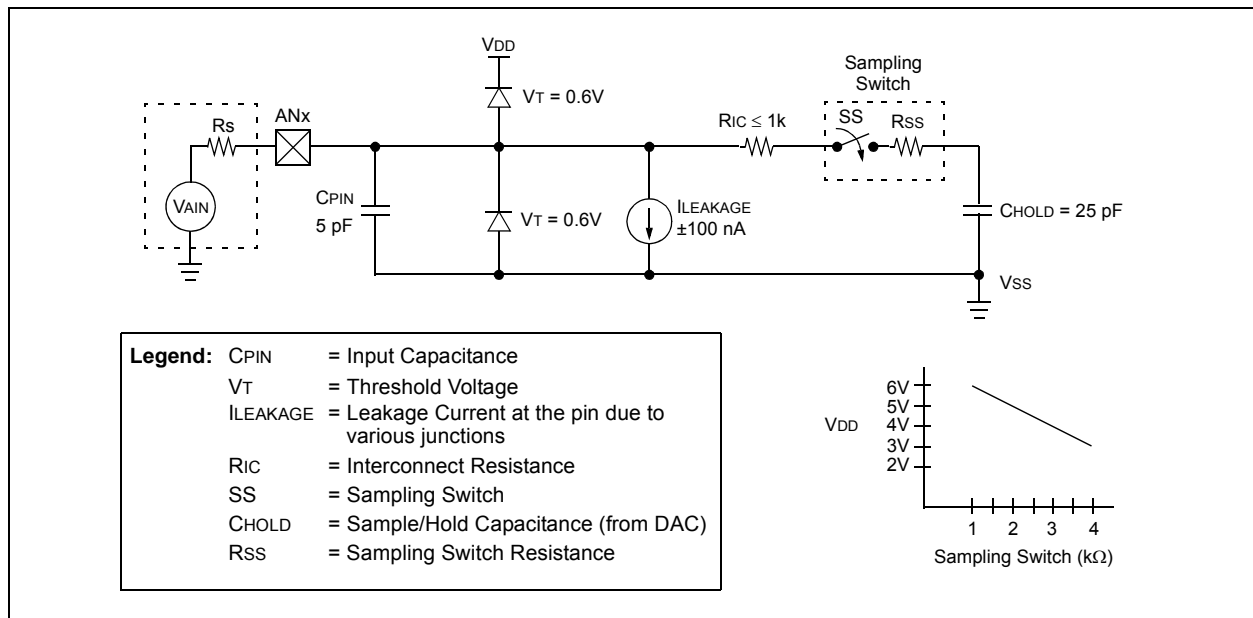
1. Configure the A/D module:
  - Configure analog pins, voltage reference and digital I/O (ADCON1)
  - Select A/D input channel (ADCON0)
  - Select A/D acquisition time (ADCON2)
  - Select A/D conversion clock (ADCON2)
  - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
  - Clear ADIF bit
  - Set ADIE bit
  - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
  - Set GO/DONE bit (ADCON0 register)

5. Wait for A/D conversion to complete, by either:
  - Polling for the GO/DONE bit to be cleared
  - OR
  - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as T<sub>AD</sub>. A minimum wait of 2 T<sub>AD</sub> is required before the next acquisition starts.

**FIGURE 16-2: A/D TRANSFER FUNCTION**



**FIGURE 16-3: ANALOG INPUT MODEL**



# PIC18F1230/1330

## 16.1 Triggering A/D Conversions

The A/D conversion can be triggered by setting the GO/DONE bit. This bit can either be set manually by the programmer or by setting the SEVTEN bit of ADCON0. When the SEVTEN bit is set, the Special Event Trigger from the Power Control PWM module triggers the A/D conversion. For more information, see **Section 14.14 “PWM Special Event Trigger”**.

## 16.2 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 16-3. 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), the channel must be sampled for at least the minimum acquisition time before starting a conversion.

**Note:** When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 16-1 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 16-3 shows the calculation of the minimum required acquisition time, TACQ. This calculation is based on the following application system assumptions:

CHOLD	=	25 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSB
VDD	=	5V → Rss = 2 kΩ
Temperature	=	85°C (system max.)

### EQUATION 16-1: ACQUISITION TIME

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

### EQUATION 16-2: A/D MINIMUM CHARGING TIME

$$\begin{aligned} \text{V}_{\text{HOLD}} &= (\text{V}_{\text{REF}} - (\text{V}_{\text{REF}}/2048)) \cdot (1 - e^{-(\text{TC}/\text{CHOLD})(\text{RIC} + \text{R}_{\text{SS}} + \text{R}_{\text{S}})}) \\ \text{or} \\ \text{TC} &= -(\text{CHOLD})(\text{RIC} + \text{R}_{\text{SS}} + \text{R}_{\text{S}}) \ln(1/2048) \end{aligned}$$

### EQUATION 16-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{TAMP} + \text{TC} + \text{TCOFF} \\ \text{TAMP} &= 0.2 \mu\text{s} \\ \text{TCOFF} &= (\text{Temp} - 25^\circ\text{C})(0.02 \mu\text{s}/^\circ\text{C}) \\ &\quad (85^\circ\text{C} - 25^\circ\text{C})(0.02 \mu\text{s}/^\circ\text{C}) \\ &\quad 1.2 \mu\text{s} \end{aligned}$$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, TCOFF = 0 ms.

$$\begin{aligned} \text{TC} &= -(\text{CHOLD})(\text{RIC} + \text{R}_{\text{SS}} + \text{R}_{\text{S}}) \ln(1/2048) \\ &\quad -(25 \text{ pF})(1 \text{ k}\Omega + 2 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(0.0004883) \\ &\quad 1.05 \mu\text{s} \\ \text{TACQ} &= 0.2 \mu\text{s} + 1 \mu\text{s} + 1.2 \mu\text{s} \\ &\quad 2.4 \mu\text{s} \end{aligned}$$

## 16.3 Selecting and Configuring Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the  $\overline{\text{GO/DONE}}$  bit is set. It also gives users the option to use an automatically determined acquisition time.

Acquisition time may be set with the ACQT2:ACQT0 bits (ADCON2<5:3>), which provide a range of 2 to 20 TAD. When the  $\overline{\text{GO/DONE}}$  bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there may be no need to wait for an acquisition time between selecting a channel and setting the  $\overline{\text{GO/DONE}}$  bit.

Manual acquisition is selected when ACQT2:ACQT0 = 000. When the  $\overline{\text{GO/DONE}}$  bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the  $\overline{\text{GO/DONE}}$  bit. This option is also the default Reset state of the ACQT2:ACQT0 bits and is compatible with devices that do not offer programmable acquisition times.

In either case, when the conversion is completed, the  $\overline{\text{GO/DONE}}$  bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

## 16.4 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. There are seven possible options for TAD:

- 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 as short as possible, but greater than the minimum TAD (see parameter 130 for more information).

Table 16-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

**TABLE 16-1: TAD vs. DEVICE OPERATING FREQUENCIES**

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18F1230/1330	PIC18LF1230/1330 <sup>(4)</sup>
2 TOSC	000	2.86 MHz	1.43 MHz
4 TOSC	100	5.71 MHz	2.86 MHz
8 TOSC	001	11.43 MHz	5.72 MHz
16 TOSC	101	22.86 MHz	11.43 MHz
32 TOSC	010	40.0 MHz	22.86 MHz
64 TOSC	110	40.0 MHz	22.86 MHz
RC <sup>(3)</sup>	x11	1.00 MHz <sup>(1)</sup>	1.00 MHz <sup>(2)</sup>

**Note 1:** The RC source has a typical TAD time of 1.2  $\mu\text{s}$ .

**2:** The RC source has a typical TAD time of 2.5  $\mu\text{s}$ .

**3:** For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.

**4:** Low-power (PIC18LF1230/1330) devices only.

# PIC18F1230/1330

---

## 16.5 Operation in Power-Managed Modes

The selection of the automatic acquisition time and A/D conversion clock is determined in part by the clock source and frequency while in a power-managed mode.

If the A/D is expected to operate while the device is in a power-managed mode, the ACQT2:ACQT0 and ADCS2:ADCS0 bits in ADCON2 should be updated in accordance with the clock source to be used in that mode. After entering the mode, an A/D acquisition or conversion may be started. Once started, the device should continue to be clocked by the same clock source until the conversion has been completed.

If desired, the device may be placed into the corresponding Idle mode during the conversion. If the device clock frequency is less than 1 MHz, the A/D RC clock source should be selected.

Operation in Sleep mode requires the A/D FRC clock to be selected. If bits ACQT2:ACQT0 are set to '000' and a conversion is started, the conversion will be delayed one instruction cycle to allow execution of the *SLEEP* instruction and entry to Sleep mode. The IDLEN bit (OSCCON<7>) must have already been cleared prior to starting the conversion.

## 16.6 Configuring Analog Port Pins

The ADCON1 and TRISA registers configure the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS1: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 as analog inputs. Analog levels on a digitally configured input will be accurately converted.

**2:** Analog levels on any pin defined as a digital input may cause the digital input buffer to consume current out of the device's specification limits.



## 16.7 A/D Conversions

Figure 16-4 shows the operation of the A/D Converter after the  $\overline{\text{GO/DONE}}$  bit has been set and the ACQT2:ACQT0 bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 16-5 shows the operation of the A/D Converter after the  $\overline{\text{GO/DONE}}$  bit has been set, the ACQT2:ACQT0 bits are set to '010' and a 4 TAD acquisition time is selected before the conversion starts.

Clearing the  $\overline{\text{GO/DONE}}$  bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. This means that 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).

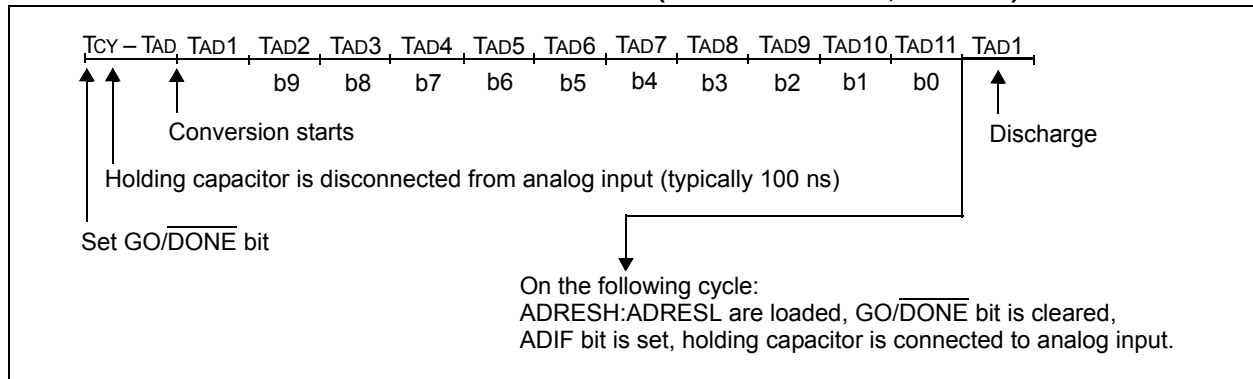
After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

**Note:** The  $\overline{\text{GO/DONE}}$  bit should **NOT** be set in the same instruction that turns on the A/D.

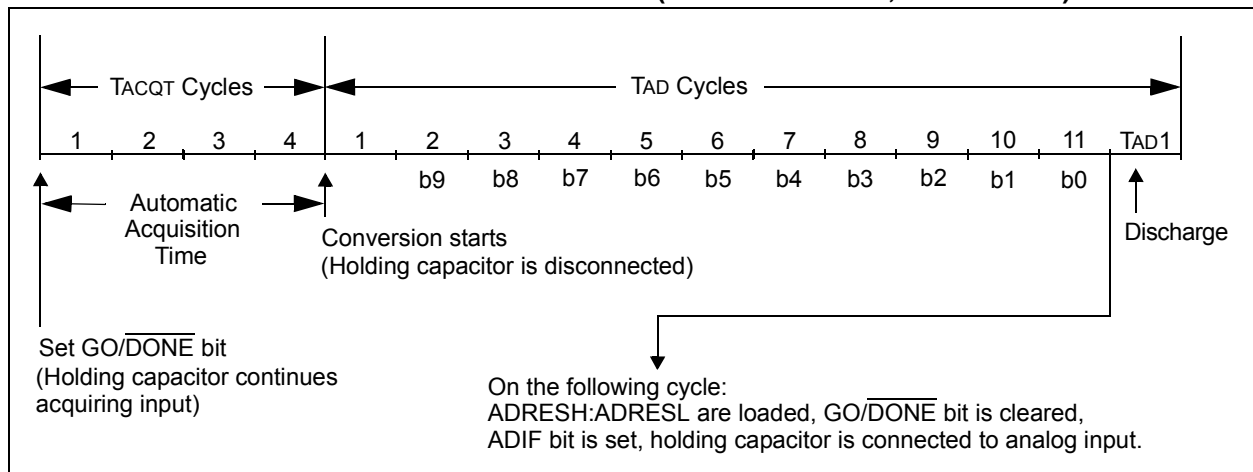
## 16.8 Discharge

The discharge phase is used to initialize the value of the capacitor array. The array is discharged before every sample. This feature helps to optimize the unity-gain amplifier, as the circuit always needs to charge the capacitor array, rather than charge/discharge based on previous measure values.

**FIGURE 16-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**



**FIGURE 16-5: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



# PIC18F1230/1330

**TABLE 16-2: REGISTERS ASSOCIATED WITH A/D OPERATION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
ADRESH	A/D Result Register High Byte								48
ADRESL	A/D Result Register Low Byte								48
ADCON0	SEVTEN	—	—	—	CHS1	CHS0	GO/DONE	ADON	48
ADCON1	—	—	—	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	48
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	48
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5 <sup>(2)</sup>	RA4	RA3	RA2	RA1	RA0	50
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Control Register						49

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** PORTA<7:6> and their direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**2:** The RA5 bit is only available when Master Clear Reset is disabled (MCLR Configuration bit = 0); otherwise, RA5 reads as '0'. This bit is read-only.

## 17.0 COMPARATOR MODULE

The analog comparator module contains three comparators. The inputs can be selected from the analog inputs multiplexed with pins RA0, RB2 and RB3, as well as the on-chip voltage reference (see

Section 18.0 “Comparator Voltage Reference Module”). The digital outputs are not available at the pin level and can only be read through the control register, CMCON (Register 17-1). CMCON also selects the comparator input.

**REGISTER 17-1: CMCON: COMPARATOR CONTROL REGISTER**

R-0	R-0	R-0	U-0	U-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0
bit 7							bit 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

- bit 7      **C2OUT:** Comparator 2 Output bit  
1 = C2 VIN+ > C2 VIN- (CVREF)  
0 = C2 VIN+ < C2 VIN- (CVREF)
- bit 6      **C1OUT:** Comparator 1 Output bit  
1 = C1 VIN+ > C1 VIN- (CVREF)  
0 = C1 VIN+ < C1 VIN- (CVREF)
- bit 5      **C0OUT:** Comparator 0 Output bit  
1 = C0 VIN+ > C0 VIN- (CVREF)  
0 = C0 VIN+ < C0 VIN- (CVREF)
- bit 4-3    **Unimplemented:** Read as '0'
- bit 2      **CMEN2:** Comparator 2 Enable bit  
1 = Comparator 2 is enabled  
0 = Comparator 2 is disabled
- bit 1      **CMEN1:** Comparator 1 Enable bit  
1 = Comparator 1 is enabled  
0 = Comparator 1 is disabled
- bit 0      **CMEN0:** Comparator 0 Enable bit  
1 = Comparator 0 is enabled  
0 = Comparator 0 is disabled

# PIC18F1230/1330

## 17.1 Comparator Configuration

For every analog comparator, there is a control bit called CMENx in the CMCON register. By setting the CMENx bit, the corresponding comparator can be enabled. If the Comparator mode is changed, the comparator output level may not be valid for the specified mode change delay shown in **Section 23.0 “Electrical Characteristics”**.

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

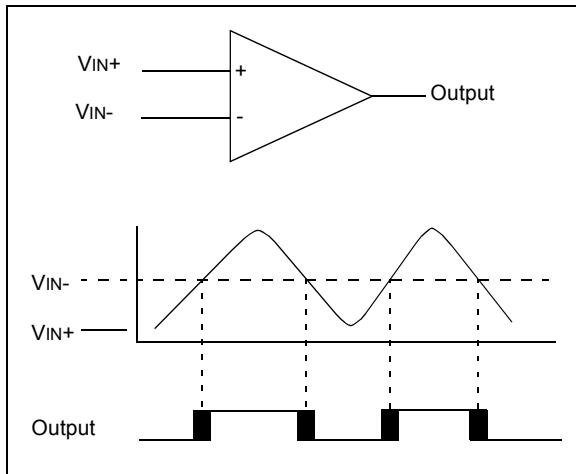
## 17.2 Comparator Operation

A single comparator is shown in Figure 17-1, along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ (CMPx) is less than the analog input VIN- (CVREF), the output of the comparator is a digital low level. When the analog input at VIN+ (CMPx) is greater than the analog input VIN- (CVREF), the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 17-1 represent the uncertainty due to input offsets and response time.

## 17.3 Comparator Reference

In this comparator module, an internal voltage reference is used (see **Section 18.0 “Comparator Voltage Reference Module”**).

**FIGURE 17-1: SINGLE COMPARATOR**



## 17.4 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 (see **Section 23.0 “Electrical Characteristics”**).

## 17.5 Comparator Outputs

The comparator outputs are read through the CxOUT bits of the CMCON register. These bits are read-only. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications.

**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.

## 17.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of the corresponding comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:5>, to determine the actual change that occurred. The CMPxIF bit (PIR1<3:1>) is the Comparator Interrupt Flag. The CMPxIF bit must be reset by clearing it. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

Both the CMPxIE bit (PIE1<3:1>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt for the corresponding comparator. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMPxIF bit will still be set if an interrupt condition occurs.

**Note:** If a change in the CMCON register (C2OUT, C1OUT or C0OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMPxIF (PIR1 register) interrupt flag may not get set.

The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- Any read or write of CMCON will end the mismatch condition.
- Clear flag bit CMPxIF.
- Input returning to original state.

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

## 17.7 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. 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 (CMEN2:CMEN0 = 000) before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

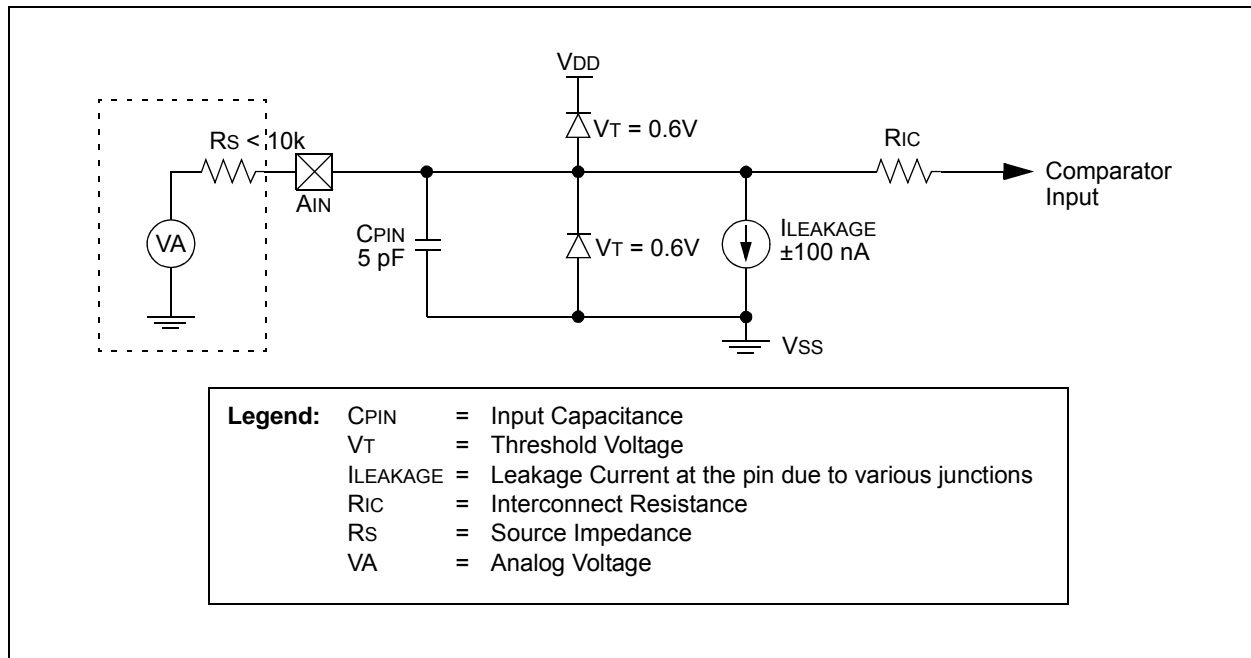
## 17.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state, causing the comparator modules to be turned off (CMEN2:CMEN0 = 000).

## 17.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 17-2. 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 Zener diode, should have very little leakage current.

**FIGURE 17-2: COMPARATOR ANALOG INPUT MODEL**



# PIC18F1230/1330

**TABLE 17-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	48
CVRCON	CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	48
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR1	—	ADIF	RCIF	TXIF	CMP2IF	CMP1IF	CMP0IF	TMR1IF	49
PIE1	—	ADIE	RCIE	TXIE	CMP2IE	CMP1IE	CMP0IE	TMR1IE	49
IPR1	—	ADIP	RCIP	TXIP	CMP2IP	CMP1IP	CMP0IP	TMR1IP	49
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5 <sup>(2)</sup>	RA4	RA3	RA2	RA1	RA0	50
LATA	LATA7 <sup>(1)</sup>	LATA6 <sup>(1)</sup>	PORTA Data Latch Register (Read and Write to Data Latch)						49
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Control Register						49
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	50
LATB	PORTB Data Latch Register (Read and Write to Data Latch)								49
TRISB	PORTB Data Direction Control Register								49

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

**Note 1:** PORTA<7:6> and their direction and latch bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**2:** The RA5 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RA5 reads as '0'. This bit is read-only.

## 18.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 16-tap resistor ladder network that provides a selectable reference voltage. Its purpose is to provide a reference for the analog comparators.

A block diagram of the module is shown in Figure 18-1. 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 module's supply reference can be provided from either device VDD/VSS or an external voltage reference.

### 18.1 Configuring the Comparator Voltage Reference

The voltage reference module is controlled through the CVRCON register (Register 18-1). The comparator voltage reference provides two ranges of output voltage, each with 16 distinct levels. The range to be used is selected by the CVRR bit (CVRCON<5>). The primary difference between the ranges is the size of the steps selected by the CVREF selection bits (CVR3:CVR0), with one range offering finer resolution. The equations used to calculate the output of the comparator voltage reference are as follows:

If CVRR = 1:

$$CVREF = ((CVR3:CVR0)/24) \times CVRSRC$$

If CVRR = 0:

$$CVREF = (CVRSRC \times 1/4) + (((CVR3:CVR0)/32) \times CVRSRC)$$

The comparator reference supply voltage can come from either AVDD or AVSS, or the external VREF+ that is multiplexed with RA4 and AVSS. The voltage source is selected by the CVRSS bit (CVRCON<4>).

Additionally, the voltage reference can select the unscaled VREF+ input for use by the comparators, bypassing the CVREF module. (See Table 18-1 and Figure 18-1.)

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see Table 23-3 in **Section 23.0 "Electrical Characteristics"**).

**TABLE 18-1: VOLTAGE REFERENCE OUTPUT**

CVREN	CVRSS	CVREF	Comparator Input
0	0	Disabled	No reference
0	1	Disabled	From VREF (CVREF bypassed)
1	0	Enabled	From CVREF
1	1	Enabled	From CVREF

# PIC18F1230/1330

## REGISTER 18-1: CVRCON: COMPARATOR VOLTAGE REFERENCE 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
CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

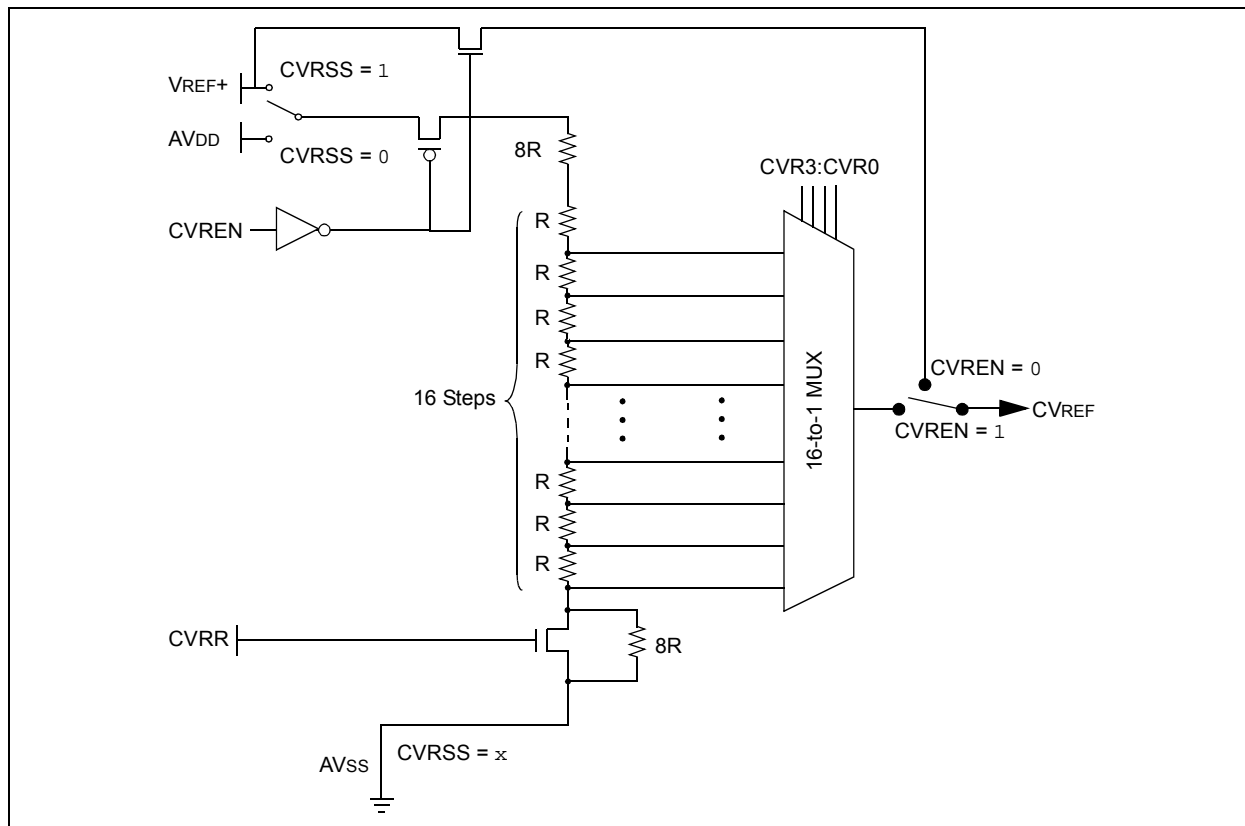
### Legend:

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

- bit 7      **CVREN:** Comparator Voltage Reference Enable bit  
 1 = CVREF circuit powered on  
 0 = CVREF circuit powered down
- bit 6      **Unimplemented:** Read as '0'
- bit 5      **CVRR:** Comparator VREF Range Selection bit  
 1 = 0 to 0.667 CVRSRC, with CVRSRC/24 step size (low range)  
 0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size (high range)
- bit 4      **CVRSS:** Comparator VREF Source Selection bit  
 When CVRR = 1  
 1 = Comparator reference source, CVRSRC = (VREF+) – (AVSS)  
 0 = Comparator reference source, CVRSRC = AVDD – AVSS  
 When CVRR = 0  
 1 = VREF+ input used directly, comparator voltage reference bypassed  
 0 = No reference is provided
- bit 3-0    **CVR3:CVR0:** Comparator VREF Value Selection bits ( $0 \leq (CVR3:CVR0) \leq 15$ )  
 When CVRR = 1:  
 $CVREF = ((CVR3:CVR0)/24) \cdot (CVRSRC)$   
 When CVRR = 0:  
 $CVREF = (CVRSRC/4) + ((CVR3:CVR0)/32) \cdot (CVRSRC)$



**FIGURE 18-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM**



## 18.2 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 18-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in **Section 23.0 “Electrical Characteristics”**.

## 18.3 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.

## 18.4 Effects of a Reset

A device Reset disables the voltage reference by clearing bit, CVREN (CVRCON<7>). This Reset selects the high-voltage range by clearing bit, CVRR (CVRCON<5>). The CVR value select bits are also cleared.

**TABLE 18-2: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
CVRCON	CVREN	—	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	48
CMCON	C2OUT	C1OUT	C0OUT	—	—	CMEN2	CMEN1	CMEN0	48

**Legend:** Shaded cells are not used with the comparator voltage reference.

# PIC18F1230/1330

---

NOTES:

## 19.0 LOW-VOLTAGE DETECT (LVD)

PIC18F1230/1330 devices have a Low-Voltage Detect module (LVD). This is a programmable circuit that allows the user to specify the device voltage trip point. If the device experiences an excursion past the trip 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 the interrupt.

The Low-Voltage Detect Control register (Register 19-1) completely controls the operation of the LVD module. This allows the circuitry to be “turned off” by the user under software control, which minimizes the current consumption for the device.

The block diagram for the LVD module is shown in Figure 19-1.

**REGISTER 19-1: 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	LV DEN	LV DL3 <sup>(1)</sup>	LV DL2 <sup>(1)</sup>	LV DL1 <sup>(1)</sup>	LV DL0 <sup>(1)</sup>
bit 7							bit 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

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

bit 5      **IRVST:** Internal Reference Voltage Stable Flag bit  
 1 = Indicates that the voltage detect logic will generate the interrupt flag at the specified voltage trip point  
 0 = Indicates that the voltage detect logic will not generate the interrupt flag at the specified voltage trip point and the LVD interrupt should not be enabled

bit 4      **LV DEN:** Low-Voltage Detect Power Enable bit  
 1 = LVD enabled  
 0 = LVD disabled

bit 3-0      **LV DL3:LV DL0:** Voltage Detection Limit bits<sup>(1)</sup>  
 1111 = Reserved  
 1110 = Maximum setting  
 .  
 .  
 .  
 0000 = Minimum setting

**Note 1:** See Table 23-4 in **Section 23.0 “Electrical Characteristics”** for the specifications.

# PIC18F1230/1330

The module is enabled by setting the LVDEN bit. Each time that the LVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit is a read-only bit and is used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

## 19.1 Operation

When the LVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a

trip point voltage. The “trip point” voltage is the voltage level at which the device detects a low-voltage event depending on the configuration of the module. 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 by setting the LVDIF bit.

The trip point voltage is software programmable to any 1 of 15 values. The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

**FIGURE 19-1: LVD MODULE BLOCK DIAGRAM**



## 19.2 LVD Setup

The following steps are needed to set up the LVD module:

1. Disable the module by clearing the LVDEN bit (LVDCON<4>).
2. Write the value to the LVDL3:LVDL0 bits that selects the desired LVD trip point.
3. Enable the LVD module by setting the LVDEN bit.
4. Clear the LVD interrupt flag (PIR2<2>) which may have been set from a previous interrupt.
5. Enable the LVD interrupt, if interrupts are desired, by setting the LVDIE and GIE bits (PIE2<2> and INTCON<7>). An interrupt will not be generated until the IRVST bit is set.

## 19.3 Current Consumption

When the module is enabled, the LVD comparator and voltage divider are enabled and will consume static current. The total current consumption, when enabled, is specified in electrical specification parameter D022B.

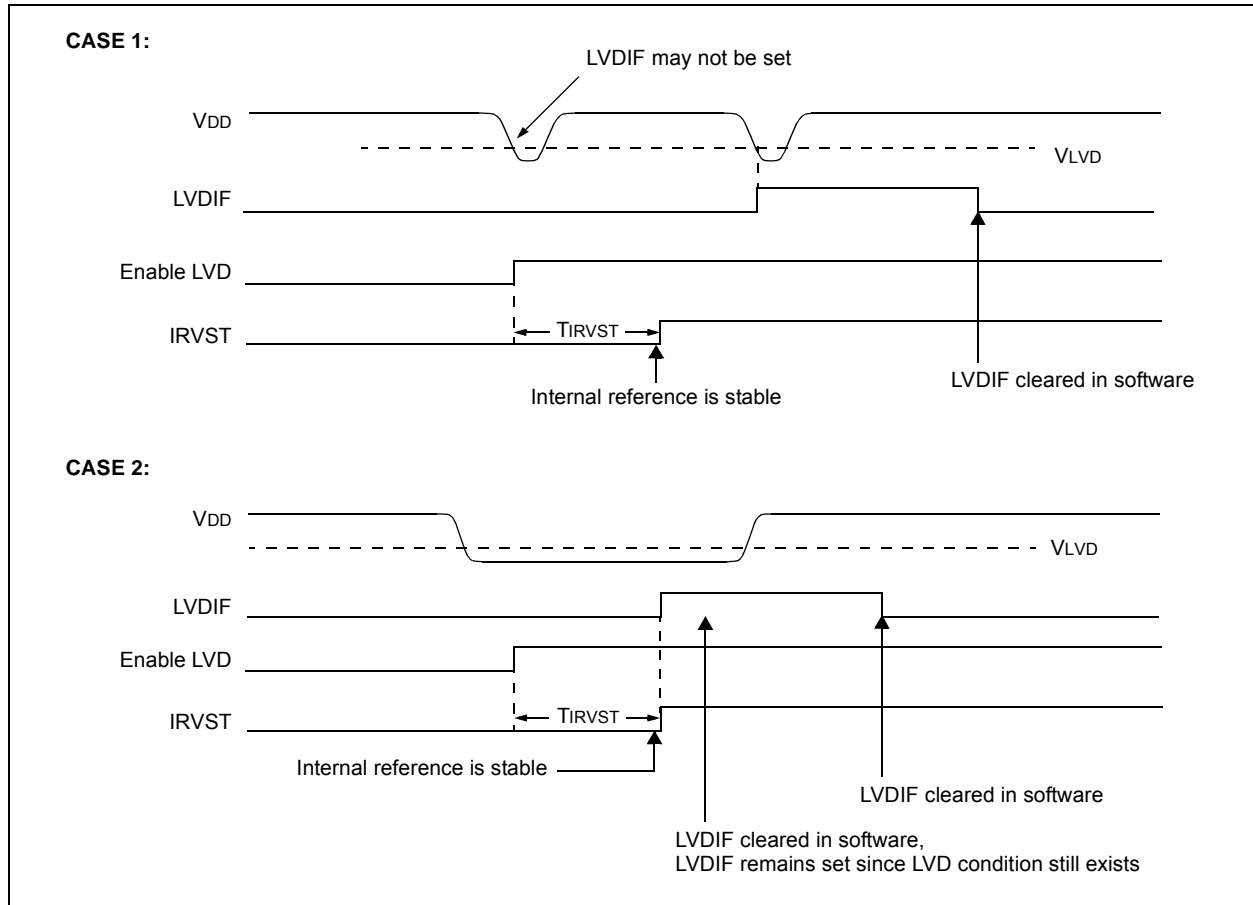
Depending on the application, the LVD module does not need to be operating constantly. To decrease the current requirements, the LVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the LVD module may be disabled.

## 19.4 LVD Start-up Time

The internal reference voltage of the LVD module, specified in electrical specification parameter D420, may be used by other internal circuitry, such as the programmable Brown-out Reset. If the LVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low-voltage condition can be reliably detected. This start-up time, T<sub>IRVST</sub>, is an interval that is independent of device clock speed. It is specified in electrical specification parameter 36.

The LVD interrupt flag is not enabled until T<sub>IRVST</sub> has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (refer to Figure 19-2).

**FIGURE 19-2: LOW-VOLTAGE DETECT OPERATION**



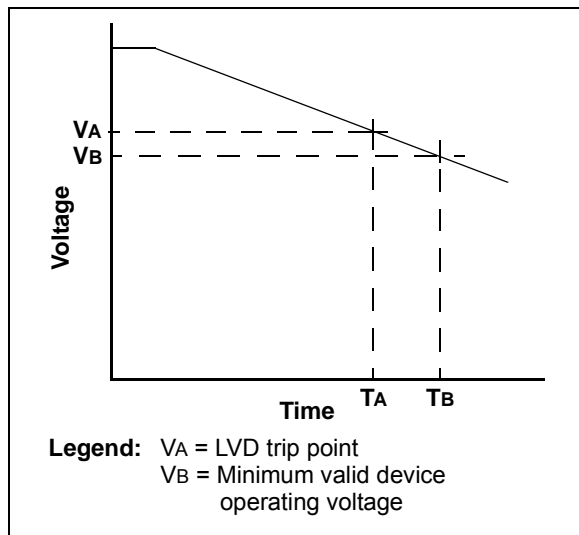
# PIC18F1230/1330

## 19.5 Applications

In many applications, the ability to detect a drop below a particular threshold is desirable.

For general battery applications, Figure 19-3 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage  $V_A$ , the LVD logic generates an interrupt at time  $T_A$ . The interrupt could cause the execution of an ISR, which would allow the application to perform “housekeeping tasks” and perform a controlled shutdown before the device voltage exits the valid operating range at  $T_B$ . The LVD, thus, would give the application a time window, represented by the difference between  $T_A$  and  $T_B$ , to safely exit.

**FIGURE 19-3: TYPICAL LOW-VOLTAGE DETECT APPLICATION**



## 19.6 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.

## 19.7 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the LVD module to be turned off.

**TABLE 19-1: REGISTERS ASSOCIATED WITH LOW-VOLTAGE DETECT MODULE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
LVDCON	—	—	IRVST	LV DEN	LV DL3	LV DL2	LV DL1	LV DL0	48
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	47
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	—	—	49
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	—	—	49
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	—	—	49

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are unused by the LVD module.

## 20.0 SPECIAL FEATURES OF THE CPU

PIC18F1230/1330 devices include several features intended to maximize reliability and minimize cost through elimination of external components. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- ID Locations
- In-Circuit Serial Programming

The oscillator can be configured for the application depending on frequency, power, accuracy and cost. All of the options are discussed in detail in **Section 3.0 “Oscillator Configurations”**.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet.

In addition to their Power-up and Oscillator Start-up Timers provided for Resets, PIC18F1230/1330 devices have a Watchdog Timer, which is either permanently enabled via the Configuration bits or software controlled (if configured as disabled).

The inclusion of an internal RC oscillator also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

### 20.1 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-3FFFFFFh) 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 WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction with the TBLPTR pointing to the Configuration register sets up the address and data for the Configuration register write. Setting the 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. For additional details on Flash programming, refer to **Section 7.5 “Writing to Flash Program Memory”**.

**TABLE 20-1: 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	IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0	00-- 0111
300002h	CONFIG2L	—	—	—	BORV1	BORV0	BOREN1	BOREN0	PWRTEN	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300004h	CONFIG3L	—	—	—	—	HPOL	LPOL	PWMPIN	—	---- 111-
300005h	CONFIG3H	MCLRE	—	—	—	T1OSCMX	—	—	FLTAMX	1--- 0--1
300006h	CONFIG4L	BKBUG	XINST	BBSIZ1	BBSIZ0	—	—	—	STVREN	1000 ---1
300008h	CONFIG5L	—	—	—	—	—	—	CP1	CP0	---- --11
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0	---- --11
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0	---- --11
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFFFh	DEVID1 <sup>(1)</sup>	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	See Table 20-2
3FFFFFFh	DEVID2 <sup>(1)</sup>	DEV10	DEV9	DEV8	DEV7	DEEV6	DEV5	DEV4	DEV3	See Table 20-2

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are unimplemented, read as ‘0’.

**Note 1:** DEVID registers are read-only and cannot be programmed by the user.

# PIC18F1230/1330

## REGISTER 20-1: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

R/P-0	R/P-0	U-0	U-0	R/P-0	R/P-1	R/P-1	R/P-1
IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0
bit 7							bit 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

bit 7      **IESO:** Internal/External Oscillator Switchover bit

1 = Oscillator Switchover mode enabled

0 = Oscillator Switchover mode disabled

bit 6      **FCMEN:** Fail-Safe Clock Monitor Enable bit

1 = Fail-Safe Clock Monitor enabled

0 = Fail-Safe Clock Monitor disabled

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

bit 3-0    **FOSC3:FOSC0:** Oscillator Selection bits

11xx = External RC oscillator, CLKO function on RA6

101x = External RC oscillator, CLKO function on RA6

1001 = Internal oscillator block, CLKO function on RA6, port function on RA7

1000 = Internal oscillator block, port function on RA6 and RA7

0111 = External RC oscillator, port function on RA6

0110 = HS oscillator, PLL enabled (Clock Frequency = 4 x FOSC1)

0101 = EC oscillator, port function on RA6

0100 = EC oscillator, CLKO function on RA6

0011 = External RC oscillator, CLKO function on RA6

0010 = HS oscillator

0001 = XT oscillator

0000 = LP oscillator



## REGISTER 20-2: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	BORV1 <sup>(1)</sup>	BORV0 <sup>(1)</sup>	BOREN1 <sup>(2)</sup>	BOREN0 <sup>(2)</sup>	PWRTEN <sup>(2)</sup>
bit 7							bit 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

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

bit 4-3      **BORV1:BORV0:** Brown-out Reset Voltage bits<sup>(1)</sup>

11 = Minimum setting

•

•

•

00 = Maximum setting

bit 2-1      **BOREN1:BOREN0:** Brown-out Reset Enable bits<sup>(2)</sup>

11 = Brown-out Reset enabled in hardware only (SBOREN is disabled)

10 = Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)

01 = Brown-out Reset enabled and controlled by software (SBOREN is enabled)

00 = Brown-out Reset disabled in hardware and software

bit 0      **PWRTEN:** Power-up Timer Enable bit<sup>(2)</sup>

1 = PWRT disabled

0 = PWRT enabled

**Note 1:** See **Section 23.1 “DC Characteristics”** for the specifications.

**Note 2:** The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.

# PIC18F1230/1330

## REGISTER 20-3: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7							bit 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

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

bit 4-1      **WDTPS3:WDTPS0:** Watchdog Timer Postscale Select bits

1111 = 1:32,768

1110 = 1:16,384

1101 = 1:8,192

1100 = 1:4,096

1011 = 1:2,048

1010 = 1:1,024

1001 = 1:512

1000 = 1:256

0111 = 1:128

0110 = 1:64

0101 = 1:32

0100 = 1:16

0011 = 1:8

0010 = 1:4

0001 = 1:2

0000 = 1:1

bit 0      **WDTEN:** Watchdog Timer Enable bit

1 = WDT enabled

0 = WDT disabled (control is placed on the SWDTEN bit)

## REGISTER 20-4: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300005h)

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	U-0
—	—	—	—	HPOL <sup>(1)</sup>	LPOL <sup>(1)</sup>	PWMPIN	—
bit 7							bit 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

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

bit 3 **HPOL:** High Side Transistors Polarity bit (Odd PWM Output Polarity Control bit)<sup>(1)</sup>

1 = PWM1, PWM3 and PWM5 are active-high (default)

0 = PWM1, PWM3 and PWM5 are active-low

bit 2 **LPOL:** Low Side Transistors Polarity bit (Even PWM Output Polarity Control bit)<sup>(1)</sup>

1 = PWM0, PWM2 and PWM4 are active-high (default)

0 = PWM0, PWM2 and PWM4 are active-low

bit 2 **PWMPIN:** PWM Output Pins Reset State Control bit

1 = PWM outputs disabled upon Reset

0 = PWM outputs drive active states upon Reset<sup>(2)</sup>

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

**Note 1:** Polarity control bits, HPOL and LPOL, define PWM signal output active and inactive states, PWM states generated by the Fault inputs or PWM manual override.

**2:** When PWMPIN = 0, PWMEN<2:0> = 100. PWM output polarity is defined by HPOL and LPOL.

# PIC18F1230/1330

## REGISTER 20-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

R/P-1	U-0	U-0	U-0	R/P-0	U-0	U-0	R/P-1
MCLRE	—	—	—	T1OSCMX	—	—	FLTAMX
bit 7							bit 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

- bit 7      **MCLRE:**  $\overline{\text{MCLR}}$  Pin Enable bit  
1 =  $\overline{\text{MCLR}}$  pin enabled, RA5 input pin disabled  
0 = RA5 input pin enabled,  $\overline{\text{MCLR}}$  pin disabled
- bit 6-4    **Unimplemented:** Read as '0'
- bit 3      **T1OSCMX:** T1OSO/T1CKI MUX bit  
1 = T1OSO/T1CKI pin resides on RA6  
0 = T1OSO/T1CKI pin resides on RB2
- bit 2-1    **Unimplemented:** Read as '0'
- bit 0      **FLTAMX:**  $\overline{\text{FLTA}}$  MUX bit  
1 =  $\overline{\text{FLTA}}$  is muxed onto RA5  
0 =  $\overline{\text{FLTA}}$  is muxed onto RA7

## REGISTER 20-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	R/P-0	R/P-0	R/P-0	U-0	U-0	U-0	R/P-1
$\overline{\text{BKBUG}}$	XINST	BBSIZ1	BBSIZ0	—	—	—	STVREN
bit 7							bit 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

- bit 7                       **$\overline{\text{BKBUG}}$** : 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                      **XINST**: Extended Instruction Set Enable bit  
 1 = Instruction set extension and Indexed Addressing mode enabled  
 0 = Instruction set extension and Indexed Addressing mode disabled
- bit 5-4                      **BBSIZ<1:0>**: Boot Block Size Select bits  
For PIC18F1330 device:  
 11 = 1 kW Boot Block size  
 10 = 1 kW Boot Block size  
 01 = 512W Boot Block size  
 00 = 256W Boot Block size  
For PIC18F1230 device:  
 11 = 512W Boot Block size  
 10 = 512W Boot Block size  
 01 = 512W Boot Block size  
 00 = 256W Boot Block size
- bit 3                      **Unimplemented**: Maintain as '0'
- bit 2-1                      **Unimplemented**: Read as '0'
- bit 0                      **STVREN**: Stack Overflow/Underflow Reset Enable bit  
 1 = Reset on stack overflow/underflow enabled  
 0 = Reset on stack overflow/underflow disabled

# PIC18F1230/1330

## REGISTER 20-7: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	CP1	CP0
bit 7							bit 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

- bit 7-2            **Unimplemented:** Read as '0'
- bit 1            **CP1:** Code Protection bit (Block 1 Code Memory Area)  
                   1 = Block 1 is not code-protected  
                   0 = Block 1 is code-protected
- bit 0            **CP0:** Code Protection bit (Block 0 Code Memory Area)  
                   1 = Block 0 is not code-protected  
                   0 = Block 0 is code-protected

## REGISTER 20-8: 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

### Legend:

R = Readable bit                      C = Clearable bit                      U = Unimplemented bit, read as '0'  
 -n = Value when device is unprogrammed                      u = Unchanged from programmed state

- bit 7            **CPD:** Code Protection bit (Data EEPROM)  
                   1 = Data EEPROM is not code-protected  
                   0 = Data EEPROM is code-protected
- bit 6            **CPB:** Code Protection bit (Boot Block Memory Area)  
                   1 = Boot Block is not code-protected  
                   0 = Boot Block is code-protected
- bit 5-0        **Unimplemented:** Read as '0'

## REGISTER 20-9: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	WRT1	WRT0
bit 7						bit 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

- bit 7-2            **Unimplemented:** Read as '0'
- bit 1            **WRT1:** Write Protection bit (Block 1 Code Memory Area)  
                   1 = Block 1 is not write-protected  
                   0 = Block 1 is write-protected
- bit 0            **WRT0:** Write Protection bit (Block 0 Code Memory Area)  
                   1 = Block 0 is not write-protected  
                   0 = Block 0 is write-protected

## REGISTER 20-10: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)

R/C-1	R/C-1	R-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC <sup>(1)</sup>	—	—	—	—	—
bit 7						bit 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

- bit 7            **WRTD:** Write Protection bit (Data EEPROM)  
                   1 = Data EEPROM is not write-protected  
                   0 = Data EEPROM is write-protected
- bit 6            **WRTB:** Write Protection bit (Boot Block Memory Area)  
                   1 = Boot Block is not write-protected  
                   0 = Boot Block is write-protected
- bit 5            **WRTC:** Write Protection bit (Configuration Registers)<sup>(1)</sup>  
                   1 = Configuration registers are not write-protected  
                   0 = Configuration registers are write-protected
- bit 4-0        **Unimplemented:** Read as '0'

**Note 1:** This bit is read-only in normal execution mode; it can be written only in Program mode.

# PIC18F1230/1330

## REGISTER 20-11: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	EBTR1 <sup>(1)</sup>	EBTR0 <sup>(1)</sup>
bit 7						bit 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

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

bit 1 **EBTR1:** Table Read Protection bit (Block 1 Code Memory Area)

1 = Block 1 is not protected from table reads executed in other blocks

0 = Block 1 is protected from table reads executed in other blocks

bit 0 **EBTR0:** Table Read Protection bit (Block 0 Code Memory Area)

1 = Block 0 is not protected from table reads executed in other blocks

0 = Block 0 is protected from table reads executed in other blocks

**Note 1:** It is recommended to enable the corresponding CPx bit to protect block from external read operations.

## REGISTER 20-12: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)

U-0	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB <sup>(1)</sup>	—	—	—	—	—	—
bit 7						bit 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

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

bit 6 **EBTRB:** Table Read Protection bit (Boot Block Memory Area)

1 = Boot Block is not protected from table reads executed in other blocks

0 = Boot Block is protected from table reads executed in other blocks

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

**Note 1:** It is recommended to enable the corresponding CPx bit to protect block from external read operations.



## REGISTER 20-13: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F1230/1330 DEVICES

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

### Legend:

R = Read-only bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value when device is unprogrammed                      u = Unchanged from programmed state

bit 7-5                      **DEV2:DEV0:** Device ID bits  
                                     000 = PIC18F1230  
                                     001 = PIC18F1330

bit 4-0                      **REV3:REV0:** Revision ID bits  
                                     These bits are used to indicate the device revision.

## REGISTER 20-14: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F1230/1330 DEVICES

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

### Legend:

R = Read-only bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value when device is unprogrammed                      u = Unchanged from programmed state

bit 7-0                      **DEV10:DEV3:** Device ID bits<sup>(1)</sup>  
                                     0001 1110 = PIC18F1230/1330 devices  
                                     These bits are used with the DEV2:DEV0 bits in the DEVID1 register to identify part number.

**Note 1:** The values for DEV10:DEV3 may be shared with other devices. A device can be identified by using the entire DEV10:DEV0 bit sequence.

# PIC18F1230/1330

## 20.2 Watchdog Timer (WDT)

For PIC18F1230/1330 devices, the WDT is driven by the INTRC source. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by bits in Configuration Register 2H. Available periods range from 4 ms to 131.072 seconds (2.18 minutes). The WDT and postscaler are cleared when any of the following events occur: a `SLEEP` or `CLRWDT` instruction is executed, the IRCF bits (`OSCCON<6:4>`) are changed or a clock failure has occurred.

**Note 1:** The `CLRWDT` and `SLEEP` instructions clear the WDT and postscaler counts when executed.

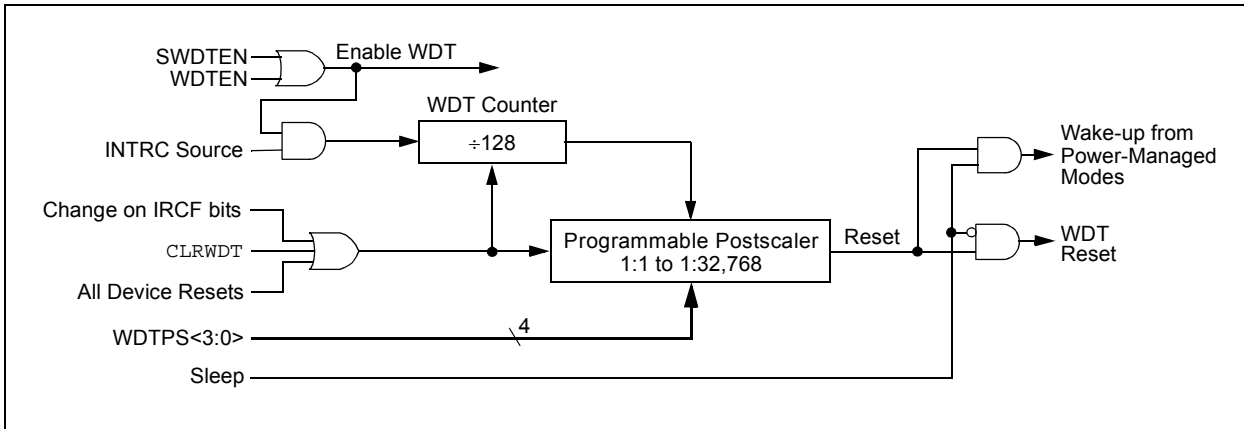
**2:** Changing the setting of the IRCF bits (`OSCCON<6:4>`) clears the WDT and postscaler counts.

**3:** When a `CLRWDT` instruction is executed, the postscaler count will be cleared.

### 20.2.1 CONTROL REGISTER

Register 20-15 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, but only if the Configuration bit has disabled the WDT.

**FIGURE 20-1: WDT BLOCK DIAGRAM**



## REGISTER 20-15: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN <sup>(1)</sup>
bit 7							bit 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

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

bit 0                      **SWDTEN:** Software Controlled Watchdog Timer Enable bit<sup>(1)</sup>

1 = Watchdog Timer is on

0 = Watchdog Timer is off

**Note 1:** This bit has no effect if the Configuration bit, WDTEN, is enabled.

**TABLE 20-2: SUMMARY OF WATCHDOG TIMER REGISTERS**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
RCON	IPEN	SBOREN <sup>(1)</sup>	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	48
WDTCON	—	—	—	—	—	—	—	SWDTEN <sup>(2)</sup>	48

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 “Brown-out Reset (BOR)”**.

**2:** This bit has no effect if the Configuration bit, WDTEN, is enabled.

# PIC18F1230/1330

## 20.3 Two-Speed Start-up

The Two-Speed Start-up feature helps to minimize the latency period from oscillator start-up to code execution by allowing the microcontroller to use the INTOSC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is LP, XT, HS or HSPLL (crystal-based modes). Other sources do not require an OST start-up delay; for these, Two-Speed Start-up should be disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer, after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI\_RUN mode.

To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IRCF2:IRCF0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:IRCF0 bits prior to entering Sleep mode.

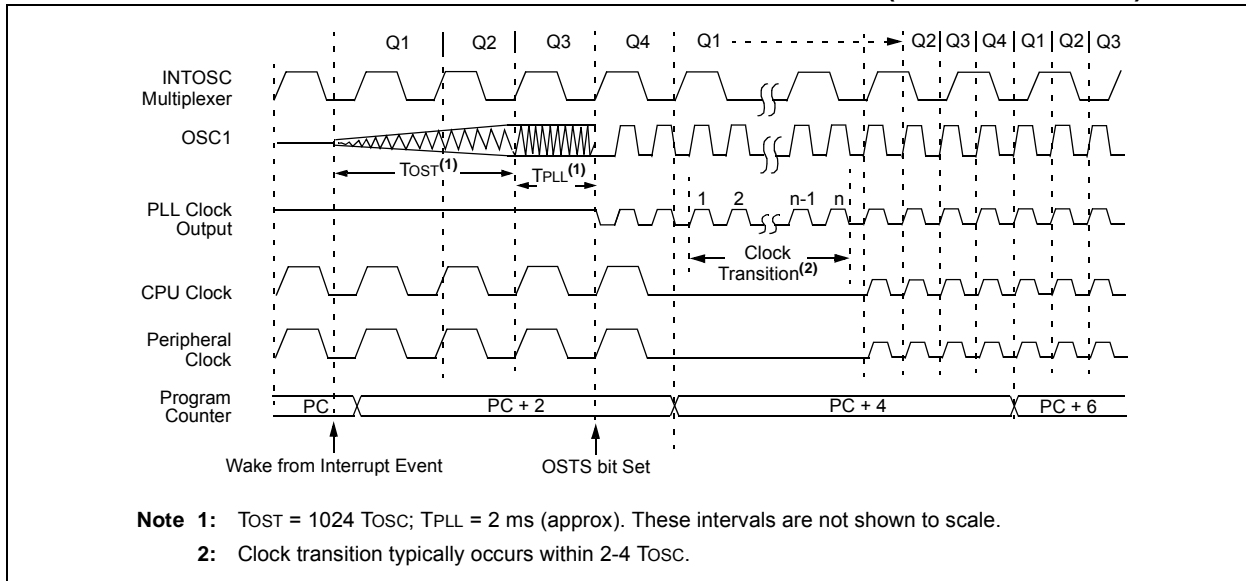
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 20.3.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTOSC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including multiple SLEEP instructions (refer to **Section 4.1.4 “Multiple Sleep Commands”**). In practice, this means that user code can change the SCS1:SCS0 bit settings or issue SLEEP instructions before the OST times out. This would allow an application to briefly wake-up, perform routine “housekeeping” tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OST bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

**FIGURE 20-2: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)**

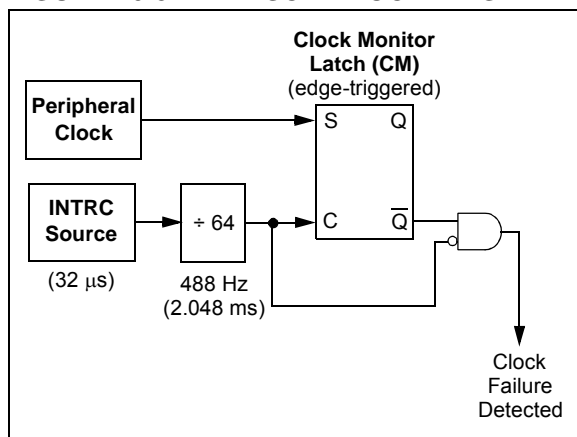


## 20.4 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the microcontroller to continue operation in the event of an external oscillator failure by automatically switching the device clock to the internal oscillator block. The FSCM function is enabled by setting the FCMEN Configuration bit.

When FSCM is enabled, the INTRC oscillator runs at all times to monitor clocks to peripherals and provide a backup clock in the event of a clock failure. Clock monitoring (shown in Figure 20-3) is accomplished by creating a sample clock signal, which is the INTRC output divided by 64. This allows ample time between FSCM sample clocks for a peripheral clock edge to occur. The peripheral device clock and the sample clock are presented as inputs to the Clock Monitor latch (CM). The CM is set on the falling edge of the device clock source, but cleared on the rising edge of the sample clock.

**FIGURE 20-3: FSCM BLOCK DIAGRAM**



Clock failure is tested for on the falling edge of the sample clock. If a sample clock falling edge occurs while CM is still set, a clock failure has been detected (Figure 20-4). This causes the following:

- The FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>).
- The device clock source is switched to the internal oscillator block (OSCCON is not updated to show the current clock source – this is the fail-safe condition).
- The WDT is reset.

During switchover, the postscaler frequency from the internal oscillator block may not be sufficiently stable for timing sensitive applications. In these cases, it may be desirable to select another clock configuration and enter an alternate power-managed mode. This can be done to attempt a partial recovery or execute a controlled shut-down. See **Section 4.1.4 “Multiple Sleep Commands”** and **Section 20.3.1 “Special Considerations for Using Two-Speed Start-up”** for more details.

To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IRCF2:IRCF0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:IRCF0 bits prior to entering Sleep mode.

The FSCM will detect failures of the primary or secondary clock sources only. If the internal oscillator block fails, no failure would be detected, nor would any action be possible.

### 20.4.1 FSCM AND THE WATCHDOG TIMER

Both the FSCM and the WDT are clocked by the INTRC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTRC oscillator when the FSCM is enabled.

As already noted, the clock source is switched to the INTOSC clock when a clock failure is detected. Depending on the frequency selected by the IRCF2:IRCF0 bits, this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed allows a WDT time-out to occur and a subsequent device Reset. For this reason, fail-safe clock events also reset the WDT and postscaler, allowing it to start timing from when execution speed was changed and decreasing the likelihood of an erroneous time-out.

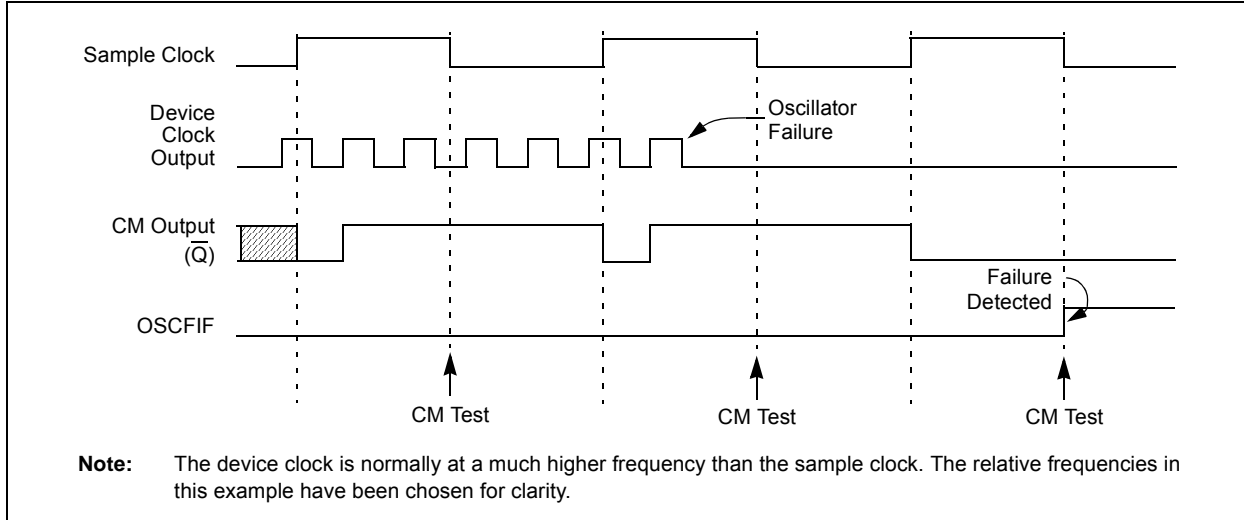
### 20.4.2 EXITING FAIL-SAFE OPERATION

The fail-safe condition is terminated by either a device Reset or by entering a power-managed mode. On Reset, the controller starts the primary clock source specified in Configuration Register 1H (with any required start-up delays that are required for the oscillator mode, such as the OST or PLL timer). The INTOSC multiplexer provides the device clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock source is then switched to the primary clock (indicated by the OSTS bit in the OSCCON register becoming set). The Fail-Safe Clock Monitor then resumes monitoring the peripheral clock.

The primary clock source may never become ready during start-up. In this case, operation is clocked by the INTOSC multiplexer. The OSCCON register will remain in its Reset state until a power-managed mode is entered.

# PIC18F1230/1330

**FIGURE 20-4: FSCM TIMING DIAGRAM**



### 20.4.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. Fail-Safe Clock Monitoring of the power-managed clock source resumes in the power-managed mode.

If an oscillator failure occurs during power-managed operation, the subsequent events depend on whether or not the oscillator failure interrupt is enabled. If enabled (OSCFIF = 1), code execution will be clocked by the INTOSC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTOSC source.

### 20.4.4 POR OR WAKE FROM SLEEP

The FSCM is designed to detect oscillator failure at any point after the device has exited Power-on Reset (POR) or low-power Sleep mode. When the primary device clock is EC, RC or INTRC modes, monitoring can begin immediately following these events.

For oscillator modes involving a crystal or resonator (HS, HSPLL, LP or XT), the situation is somewhat different. Since the oscillator may require a start-up

time considerably longer than the FSCM sample clock time, a false clock failure may be detected. To prevent this, the internal oscillator block is automatically configured as the device clock and functions until the primary clock is stable (the OST and PLL timers have timed out). This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTRC returns to its role as the FSCM source.

**Note:** The same logic that prevents false oscillator failure interrupts on POR, or wake from Sleep, will also prevent the detection of the oscillator's failure to start at all following these events. This can be avoided by monitoring the OSTs bit and using a timing routine to determine if the oscillator is taking too long to start. Even so, no oscillator failure interrupt will be flagged.

As noted in **Section 20.3.1 "Special Considerations for Using Two-Speed Start-up"**, it is also possible to select another clock configuration and enter an alternate power-managed mode while waiting for the primary clock to become stable. When the new power-managed mode is selected, the primary clock is disabled.

## 20.5 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PIC® devices.

The user program memory is divided into three blocks. One of these is a Boot Block of variable size (maximum 2 Kbytes). The remainder of the memory is divided into two blocks on binary boundaries.

Each of the three blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPx)
- Write-Protect bit (WRTx)
- External Block Table Read bit (EBTRx)

Figure 20-5 shows the program memory organization for 4 and 8-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 20-3.

**FIGURE 20-5: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F1230/1330**

MEMORY SIZE/DEVICE		Address Range	Block Code Protection Controlled By:
4 Kbytes (PIC18F1230)	8 Kbytes (PIC18F1330)		
Boot Block	Boot Block	000000h 0003FFh	CPB, WRTB, EBTRB
Block 0		000400h 0007FFh	CP0, WRT0, EBTR0
Block 1	Block 0	000800h 000FFFh	CP1, WRT1, EBTR1
Unimplemented Read '0's	Block 1	001000h 001FFFh	CP2, WRT2, EBTR2
Unimplemented Read '0's	Unimplemented Read '0's	002000h  1FFFFFFh	(Unimplemented Memory Space)

**TABLE 20-3: 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	—	—	—	—	—	—	CP1	CP0
300009h CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah CONFIG6L	—	—	—	—	—	—	WRT1	WRT0
30000Bh CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0
30000Dh CONFIG7H	—	EBTRB	—	—	—	—	—	—

**Legend:** Shaded cells are unimplemented.

# PIC18F1230/1330

## 20.5.1 PROGRAM MEMORY CODE PROTECTION

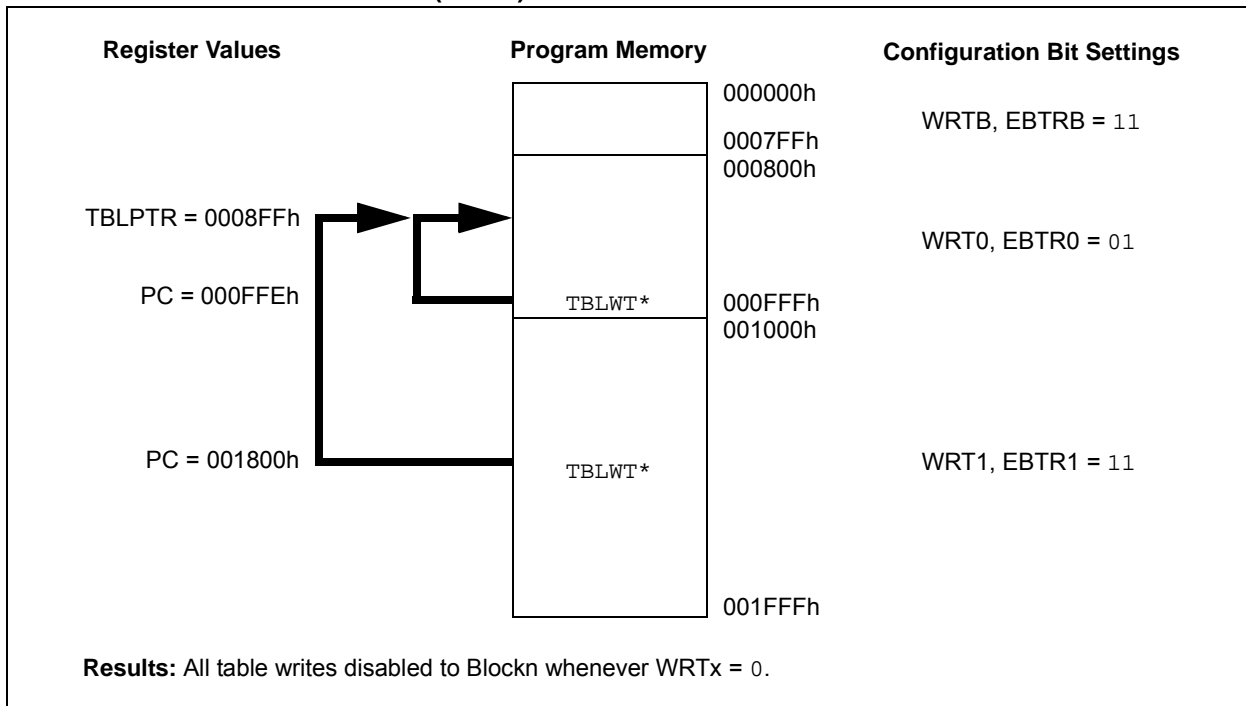
The program memory may be read to or written from any location using the table read and table write instructions. The Device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In normal execution mode, the CPx bits have no direct effect. CPx bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTx Configuration bit is '0'. The EBTRx bits control table reads. For a block of user memory with the EBTRx 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 20-6 through 20-8 illustrate table write and table read protection.

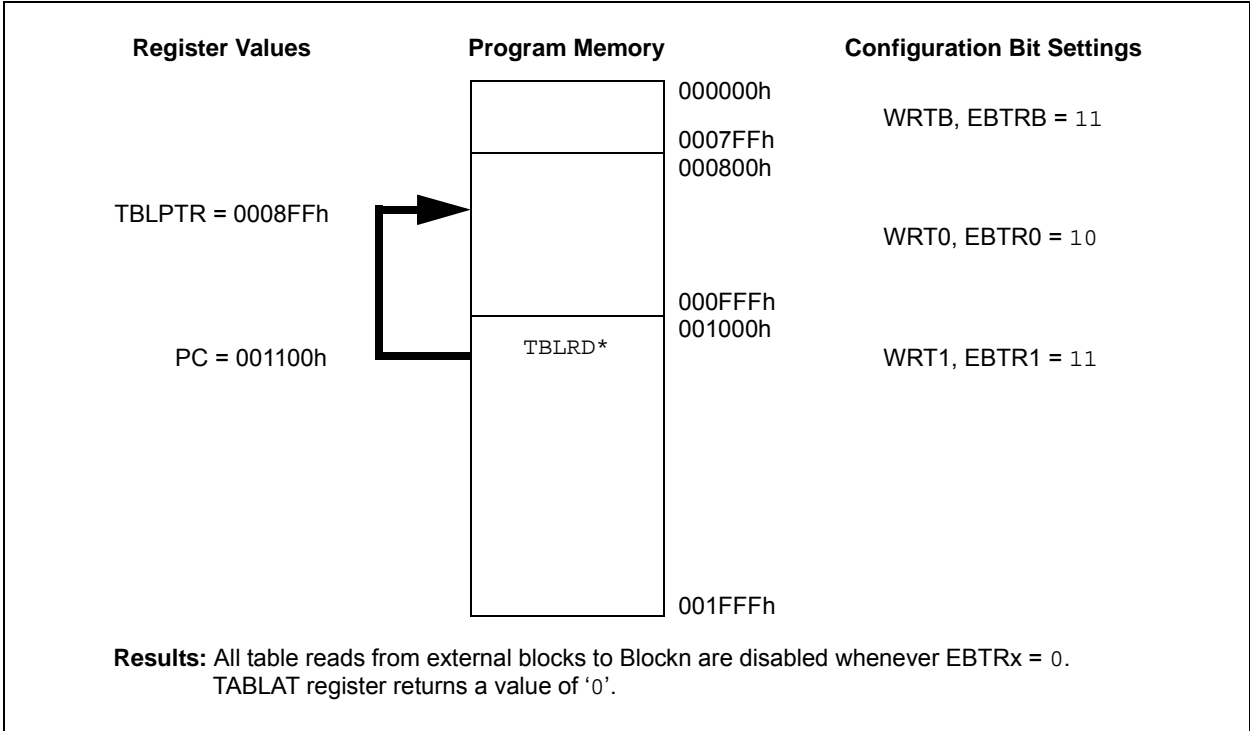
**Note:** Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP operation or an external programmer.

**FIGURE 20-6: TABLE WRITE (WRTx) DISALLOWED**

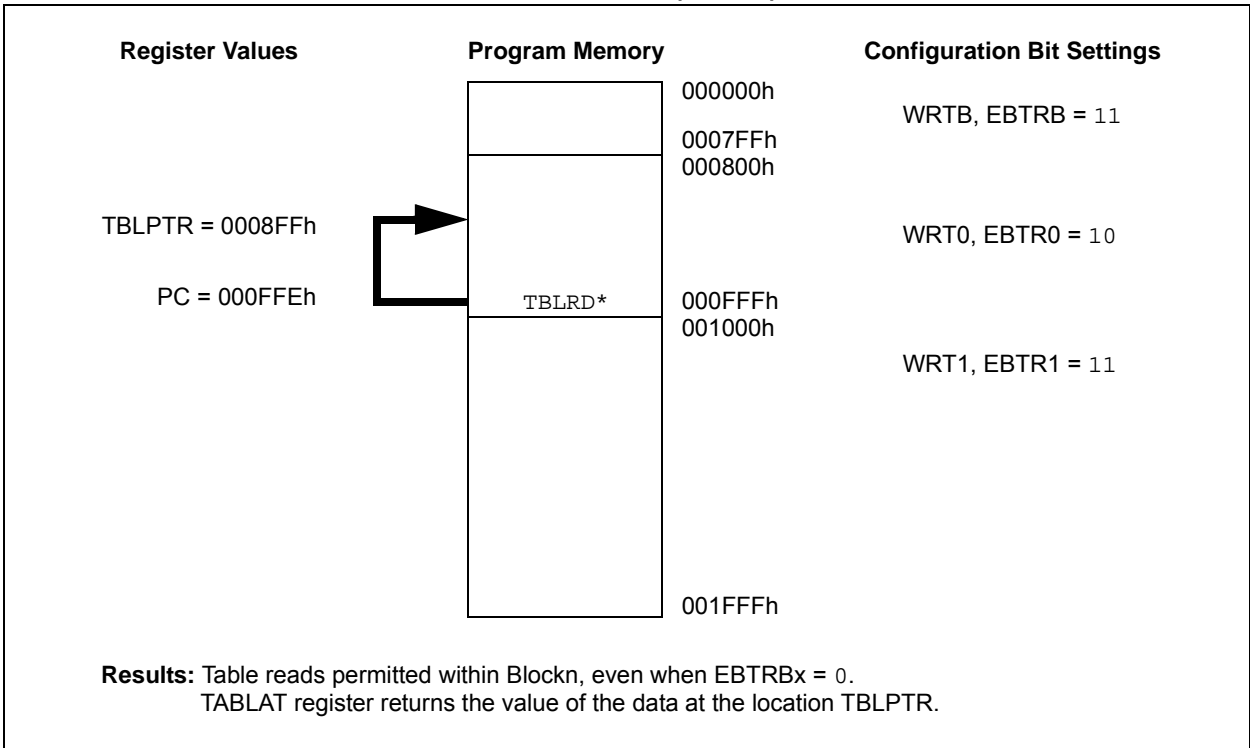




**FIGURE 20-7: EXTERNAL BLOCK TABLE READ (EBTRx) DISALLOWED**



**FIGURE 20-8: EXTERNAL BLOCK TABLE READ (EBTRx) ALLOWED**



# PIC18F1230/1330

## 20.5.2 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 internal and external writes to data EEPROM. The CPU can always read data EEPROM under normal operation, regardless of the protection bit settings.

## 20.5.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In normal execution mode, the WRTC bit is read-only. WRTC can only be written via ICSP operation or an external programmer.

## 20.6 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 both readable and writable 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.

## 20.7 In-Circuit Serial Programming

PIC18F1230/1330 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.

## 20.8 In-Circuit Debugger

When the  $\overline{\text{BKBUG}}$  Configuration bit 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 resources are not available for general use. Table 20-4 shows which resources are required by the background debugger.

**TABLE 20-4: DEBUGGER RESOURCES**

I/O pins:	RB6, RB7
Stack:	2 levels
Program Memory:	512 bytes
Data Memory:	10 bytes

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to  $\overline{\text{MCLR}}$ / $\overline{\text{VPP}}$ /RA5/FLTA, VDD, VSS, RB7/PWM5/PGD and RB6/PWM4/PGC. This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.

## 20.9 Single-Supply ICSP Programming

The PIC18F1230/1330 device family does not support Low-Voltage ICSP Programming or LVP. This device family can only be programmed using high-voltage ICSP programming. For more details, refer to the "PIC18F1230/1330 Flash Microcontroller Programming Specification" (DS39752).

Memory that is not code-protected can be erased using either a block erase, or erased row by row, then written at any specified VDD. If code-protected memory is to be erased, a block erase is required.

## 21.0 DEVELOPMENT SUPPORT

The PIC<sup>®</sup> microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB<sup>®</sup> IDE Software
- Assemblers/Compilers/Linkers
  - MPASM<sup>™</sup> Assembler
  - MPLAB C18 and MPLAB C30 C Compilers
  - MPLINK<sup>™</sup> Object Linker/  
MPLIB<sup>™</sup> Object Librarian
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB REAL ICE<sup>™</sup> In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PICSTART<sup>®</sup> Plus Development Programmer
  - MPLAB PM3 Device Programmer
  - PICKit<sup>™</sup> 2 Development Programmer
- Low-Cost Demonstration and Development Boards and Evaluation Kits

## 21.1 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<sup>®</sup> operating system-based application that contains:

- A single graphical interface to all 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
- Visual device initializer for easy register initialization
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as HI-TECH Software C Compilers and IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either assembly or C)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (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 increased flexibility and power.

# PIC18F1230/1330

---

## 21.2 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for all PIC 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

## 21.3 MPLAB C18 and MPLAB C30 C Compilers

The MPLAB C18 and MPLAB C30 Code Development Systems are complete ANSI C compilers for Microchip's PIC18 and PIC24 families of microcontrollers and the dsPIC30 and dsPIC33 family of digital signal controllers. 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.

## 21.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. 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

## 21.5 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 Assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

## 21.6 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C18 and MPLAB C30 C Compilers, and the MPASM and MPLAB ASM30 Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 21.7 MPLAB ICE 2000 High-Performance In-Circuit Emulator

The MPLAB ICE 2000 In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC 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 architecture of the MPLAB ICE 2000 In-Circuit Emulator allows expansion to support new PIC 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.

## 21.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC® and MCU devices. It debugs and programs PIC® and dsPIC® Flash microcontrollers with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The MPLAB REAL ICE probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with the popular MPLAB ICD 2 system (RJ11) or with the new high speed, noise tolerant, low-voltage differential signal (LVDS) interconnection (CAT5).

MPLAB REAL ICE is field upgradeable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added, such as software breakpoints and assembly code trace. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, real-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 21.9 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 PIC MCUs and can be used to develop for these and other PIC MCUs and dsPIC DSCs. 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, and 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 PIC devices.

## 21.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular, detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an SD/MMC card for file storage and secure data applications.

# PIC18F1230/1330

---

## 21.11 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 PIC devices in DIP packages up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus Development Programmer is CE compliant.

## 21.12 PICkit 2 Development Programmer

The PICkit™ 2 Development Programmer is a low-cost programmer and selected Flash device debugger with an easy-to-use interface for programming many of Microchip's baseline, mid-range and PIC18F families of Flash memory microcontrollers. The PICkit 2 Starter Kit includes a prototyping development board, twelve sequential lessons, software and HI-TECH's PICC™ Lite C compiler, and is designed to help get up to speed quickly using PIC® microcontrollers. The kit provides everything needed to program, evaluate and develop applications using Microchip's powerful, mid-range Flash memory family of microcontrollers.

## 21.13 Demonstration, Development and Evaluation Boards

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart® battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Check the Microchip web page ([www.microchip.com](http://www.microchip.com)) and the latest "Product Selector Guide" (DS00148) for the complete list of demonstration, development and evaluation kits.

## 22.0 INSTRUCTION SET SUMMARY

PIC18F1230/1330 devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of 8 new instructions for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

### 22.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC<sup>®</sup> MCU instruction sets, while maintaining an easy migration from these PIC MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four 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 22-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 22-1 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 four double-word instructions. These instructions were made double-word to contain the required information in 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  $\mu$ s. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2  $\mu$ s. Two-word branch instructions (if true) would take 3  $\mu$ s.

Figure 22-1 shows the general formats that the instructions can have. All examples use the convention 'nnh' to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 22-2, lists the standard instructions recognized by the Microchip MPASM<sup>™</sup> Assembler.

**Section 22.1.1 “Standard Instruction Set”** provides a description of each instruction.

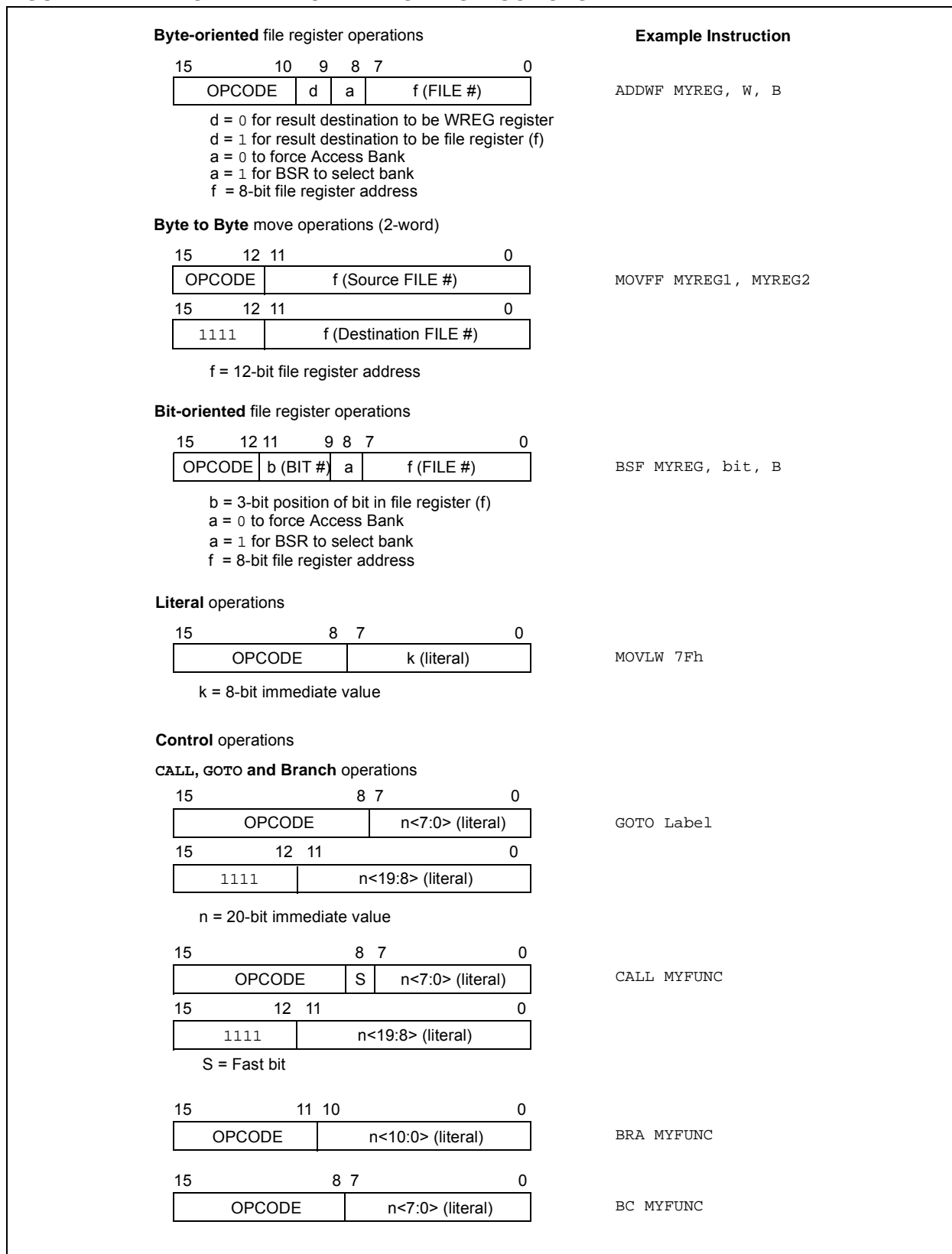
# PIC18F1230/1330

**TABLE 22-1: OPCODE FIELD DESCRIPTIONS**

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU Status bits: <b>C</b> arry, <b>D</b> igit Carry, <b>Z</b> ero, <b>O</b> verflow, <b>N</b> egative.
d	Destination select bit d = 0: store result in WREG d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
f <sub>s</sub>	12-bit Register file address (000h to FFFh). This is the source address.
f <sub>d</sub>	12-bit Register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions:
*	No change to register (such as TBLPTR with table reads and writes)
*+	Post-Increment register (such as TBLPTR with table reads and writes)
*-	Post-Decrement register (such as TBLPTR with table reads and writes)
+*	Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for Call/Branch and Return instructions.
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.
$\overline{PD}$	Power-Down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a program memory location).
TABLAT	8-bit Table Latch.
TO	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
x	Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
z <sub>s</sub>	7-bit offset value for indirect addressing of register files (source).
z <sub>d</sub>	7-bit offset value for indirect addressing of register files (destination).
{ }	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr]<n>	Specifies bit n of the register indicated by the pointer expr.
→	Assigned to.
< >	Register bit field.
∈	In the set of.
<i>italics</i>	User-defined term (font is Courier New).



**FIGURE 22-1: GENERAL FORMAT FOR INSTRUCTIONS**



# PIC18F1230/1330

**TABLE 22-2: PIC18FXXXX INSTRUCTION SET**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb			LSb			
<b>BYTE-ORIENTED OPERATIONS</b>									
ADDWF	f, d, a	Add WREG and f	1	0010	01da0	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 <sub>s</sub> , f <sub>d</sub>	Move f <sub>s</sub> (source) to 1st word f <sub>d</sub> (destination) 2nd word	2	1100	ffff	ffff	ffff	None	
				1111	ffff	ffff	ffff		
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
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	1, 2
SUBFWB	f, d, a	Subtract f from WREG with Borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with Borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
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	

- 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 the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-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.

**TABLE 22-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb	LSb					
<b>BIT-ORIENTED OPERATIONS</b>									
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
<b>CONTROL OPERATIONS</b>									
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	
BN OV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BN Z	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call subroutine	2	1110	110s	kkkk	kkkk	None	
		1st word							
		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	2	1110	1111	kkkk	kkkk	None	
		1st word							
		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 the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-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.

# PIC18F1230/1330

**TABLE 22-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb			LSb			
<b>LITERAL OPERATIONS</b>									
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 FSR(f) 1st word	2	1110	1110	00ff	kkkk	None	
MOVLB	k	Move Literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move Literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply Literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from Literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR Literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
<b>DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS</b>									
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	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 the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-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.

## 22.1.1 STANDARD INSTRUCTION SET

ADDLW	ADD Literal to W								
Syntax:	ADDLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	$(W) + k \rightarrow W$								
Status Affected:	N, OV, C, DC, Z								
Encoding:	<table border="1"> <tr> <td>0000</td> <td>1111</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	0000	1111	kkkk	kkkk				
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:	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write to W						

**Example:**           ADDLW 15h

Before Instruction  
W = 10h  
After Instruction  
W = 25h

ADDWF	ADD W to f				
Syntax:	ADDWF f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(W) + (f) \rightarrow \text{dest}$				
Status Affected:	N, OV, C, DC, Z				
Encoding:	<table border="1"> <tr> <td>0010</td> <td>01da</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0010	01da	ffff	ffff
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'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.				
Words:	1				
Cycles:	1				

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:**           ADDWF REG, 0, 0

Before Instruction  
W = 17h  
REG = 0C2h  
After Instruction  
W = 0D9h  
REG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

# PIC18F1230/1330

## ADDWFC      ADD W and Carry bit to f

Syntax:            ADDWFC    f {,d {,a}}

Operands:         $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

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 is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words:            1

Cycles:            1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:**            ADDWFC    REG, 0, 1

Before Instruction

Carry bit = 1  
REG = 02h  
W = 4Dh

After Instruction

Carry bit = 0  
REG = 02h  
W = 50h

## ANDLW      AND Literal with W

Syntax:            ANDLW    k

Operands:         $0 \leq k \leq 255$

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    05Fh

Before Instruction

W = A3h

After Instruction

W = 03h

## ANDWF

## AND W with f

**Syntax:** ANDWF f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**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'.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**Words:** 1  
**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** ANDWF REG, 0, 0

Before Instruction

W = 17h  
 REG = C2h

After Instruction

W = 02h  
 REG = C2h

## BC

## Branch if Carry

**Syntax:** BC n

**Operands:**  $-128 \leq n \leq 127$

**Operation:** if Carry bit is '1',  
 $(PC) + 2 + 2n \rightarrow PC$

**Status Affected:** None

**Encoding:**

1110	0010	nnnn	nnnn
------	------	------	------

**Description:** If the Carry bit is '1', then the program will branch.  
 The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is then a two-cycle instruction.

**Words:** 1  
**Cycles:** 1(2)

**Q Cycle Activity:**  
 If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

**Example:** HERE BC 5

Before Instruction

PC = address (HERE)

After Instruction

If Carry = 1;  
 PC = address (HERE + 12)  
 If Carry = 0;  
 PC = address (HERE + 2)

# PIC18F1230/1330

## BCF

### Bit Clear f

**Syntax:** BCF f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in \{0,1\}$

**Operation:**  $0 \rightarrow f \leftarrow b$

**Status Affected:** None

**Encoding:**

1001	bbba	ffff	ffff
------	------	------	------

**Description:** Bit 'b' in register 'f' is cleared.  
 If 'a' is '0', the Access Bank is selected.  
 If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**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, 0

Before Instruction  
 FLAG\_REG = C7h  
 After Instruction  
 FLAG\_REG = 47h

## BN

### Branch if Negative

**Syntax:** BN n

**Operands:**  $-128 \leq n \leq 127$

**Operation:** if Negative bit is '1',  
 $(PC) + 2 + 2n \rightarrow PC$

**Status Affected:** None

**Encoding:**

1110	0110	nnnn	nnnn
------	------	------	------

**Description:** If the Negative bit is '1', then the program will branch.  
 The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is then a two-cycle instruction.

**Words:** 1

**Cycles:** 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

**Example:** HERE BN Jump

Before Instruction  
 PC = address (HERE)  
 After Instruction  
 If Negative = 1;  
 PC = address (Jump)  
 If Negative = 0;  
 PC = address (HERE + 2)



# PIC18F1230/1330

## BNC Branch if Not Carry

Syntax: BNC n

Operands:  $-128 \leq n \leq 127$

Operation: if Carry bit is '0',  
 $(PC) + 2 + 2n \rightarrow PC$

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: BNN n

Operands:  $-128 \leq n \leq 127$

Operation: if Negative bit is '0',  
 $(PC) + 2 + 2n \rightarrow PC$

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)

# PIC18F1230/1330

## BNOV Branch if Not Overflow

Syntax: BNOV n

Operands:  $-128 \leq n \leq 127$

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: BNZ n

Operands:  $-128 \leq n \leq 127$

Operation: if Zero bit is '0',  
(PC) + 2 + 2n → PC

Status Affected: None

Encoding: 

1110	0001	nnnn	nnnn
------	------	------	------

Description: If the Zero bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

**Example:**                    HERE                    BNZ Jump

Before Instruction  
PC = address (HERE)

After Instruction  
If Zero = 0;  
PC = address (Jump)  
If Zero = 1;  
PC = address (HERE + 2)

## BRA Unconditional Branch

**Syntax:** BRA n

**Operands:**  $-1024 \leq n \leq 1023$

**Operation:**  $(PC) + 2 + 2n \rightarrow PC$

**Status Affected:** None

**Encoding:**

1101	0nnn	nnnn	nnnn
------	------	------	------

**Description:** Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is a two-cycle instruction.

**Words:** 1

**Cycles:** 2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC	
No operation	No operation	No operation	No operation	

**Example:**           HERE       BRA Jump

Before Instruction  
PC = address (HERE)

After Instruction  
PC = address (Jump)

## BSF Bit Set f

**Syntax:** BSF f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in [0,1]$

**Operation:**  $1 \rightarrow f<b>$

**Status Affected:** None

**Encoding:**

1000	bbba	ffff	ffff
------	------	------	------

**Description:** Bit 'b' in register 'f' is set.  
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

	Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'	

**Example:**           BSF       FLAG\_REG, 7, 1

Before Instruction  
FLAG\_REG = 0Ah

After Instruction  
FLAG\_REG = 8Ah

# PIC18F1230/1330

## BTFSC Bit Test File, Skip if Clear

**Syntax:** BTFSC f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in [0,1]$

**Operation:** skip if (f<b>) = 0

**Status Affected:** None

**Encoding:**

1011	bbba	ffff	ffff
------	------	------	------

**Description:** If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh).  
 See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**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, 0
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:** BTFSS f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b < 7$   
 $a \in [0,1]$

**Operation:** skip if (f<b>) = 1

**Status Affected:** None

**Encoding:**

1010	bbba	ffff	ffff
------	------	------	------

**Description:** If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh).  
 See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**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, 0
FALSE  :
TRUE   :
```

**Before Instruction**  
 PC = address (HERE)

**After Instruction**  
 If FLAG<1> = 0;  
 PC = address (FALSE)  
 If FLAG<1> = 1;  
 PC = address (TRUE)

**BTG**                      **Bit Toggle f**

---

Syntax:                    BTG f, b {,a}

Operands:                 $0 \leq f \leq 255$   
 $0 \leq b < 7$   
 $a \in [0,1]$

Operation:                 $(\overline{f\langle b \rangle}) \rightarrow f\langle b \rangle$

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 is selected.  
 If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words:                    1

Cycles:                   1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

**Example:**                    BTG     PORTC, 4, 0

Before Instruction:  
 PORTC = 0111 0101 [75h]  
 After Instruction:  
 PORTC = 0110 0101 [65h]

**BOV**                      **Branch if Overflow**

---

Syntax:                    BOV n

Operands:                 $-128 \leq n \leq 127$

Operation:                if Overflow bit is '1',  
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected:        None

Encoding:                

1110	0100	nnnn	nnnn
------	------	------	------

Description:             If the Overflow bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is then a two-cycle instruction.

Words:                    1

Cycles:                   1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

**Example:**                                       HERE                    BOV     Jump

Before Instruction  
 PC = address (HERE)  
 After Instruction  
 If Overflow = 1;  
           PC = address (Jump)  
 If Overflow = 0;  
           PC = address (HERE + 2)

# PIC18F1230/1330

## BZ Branch if Zero

**Syntax:** BZ n

**Operands:**  $-128 \leq n \leq 127$

**Operation:** if Zero bit is '1',  
(PC) + 2 + 2n → PC

**Status Affected:** None

**Encoding:**

1110	0000	nnnn	nnnn
------	------	------	------

**Description:** If the Zero bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

**Words:** 1

**Cycles:** 1(2)

**Q Cycle Activity:**  
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

**Example:**            HERE            BZ    Jump

Before Instruction  
PC = address (HERE)

After Instruction  
If Zero = 1;  
PC = address (Jump)  
If Zero = 0;  
PC = address (HERE + 2)

## CALL Subroutine Call

**Syntax:** CALL k {,s}

**Operands:**  $0 \leq k \leq 1048575$   
 $s \in [0,1]$

**Operation:** (PC) + 4 → TOS,  
k → PC<20:1>;  
if s = 1,  
(W) → WS,  
(STATUS) → STATUSS,  
(BSR) → BSRS

**Status Affected:** None

**Encoding:**

1110	110s	k <sub>7</sub> kkk	kkkk <sub>0</sub>
1111	k <sub>19</sub> kkk	kkkk	kkkk <sub>8</sub>

**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. Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-cycle instruction.

**Words:** 2

**Cycles:** 2

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>, PUSH PC to stack	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

**Example:**            HERE            CALL    THERE, 1

Before Instruction  
PC = address (HERE)

After Instruction  
PC = address (THERE)  
TOS = address (HERE + 4)  
WS = W  
BSRS = BSR  
STATUSS = STATUS

**CLRF**                      **Clear f**

---

Syntax:                      CLRF f{,a}

Operands:                     $0 \leq f \leq 255$   
 $a \in [0,1]$

Operation:                     $000h \rightarrow f$ ,  
 $1 \rightarrow Z$

Status Affected:            Z

Encoding:                    

0110	101a	ffff	ffff
------	------	------	------

Description:                Clears the contents of the specified register.  
If 'a' is '0', the Access Bank is selected.  
If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words:                        1

Cycles:                        1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example:                    CLRF                      FLAG\_REG, 1

Before Instruction  
FLAG\_REG = 5Ah

After Instruction  
FLAG\_REG = 00h

**CLRWDT**                    **Clear Watchdog Timer**

---

Syntax:                       CLRWDT

Operands:                    None

Operation:                     $000h \rightarrow$  WDT,  
 $000h \rightarrow$  WDT postscaler,  
 $1 \rightarrow \overline{TO}$ ,  
 $1 \rightarrow \overline{PD}$

Status Affected:             $\overline{TO}$ ,  $\overline{PD}$

Encoding:                    

0000	0000	0000	0100
------	------	------	------

Description:                CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits,  $\overline{TO}$  and  $\overline{PD}$ , are set.

Words:                        1

Cycles:                        1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example:                    CLRWDT

Before Instruction  
WDT Counter = ?

After Instruction  
WDT Counter = 00h  
WDT Postscaler = 0  
 $\overline{TO}$  = 1  
 $\overline{PD}$  = 1

# PIC18F1230/1330

## COMF Complement f

**Syntax:** COMF f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(\bar{f}) \rightarrow \text{dest}$

**Status Affected:** N, Z

**Encoding:**

0001	11da	ffff	ffff
------	------	------	------

**Description:** The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** COMF REG, 0, 0

Before Instruction  
 REG = 13h

After Instruction  
 REG = 13h  
 W = ECh

## CPFSEQ Compare f with W, Skip if f = W

**Syntax:** CPFSEQ f {,a}

**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**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, 0  
 NEQUAL :  
 EQUAL :

Before Instruction  
 PC Address = HERE  
 W = ?  
 REG = ?

After Instruction  
 If REG = W;  
 PC = Address (EQUAL)  
 If REG  $\neq$  W;  
 PC = Address (NEQUAL)



## CPFSGT Compare f with W, Skip if f > W

**Syntax:** CPFSGT f {,a}  
**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$   
**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 is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**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, 0  
 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:** CPFSLT f {,a}  
**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$   
**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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.

**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, 1  
 NLESS :  
 LESS :

**Before Instruction**  
 PC = Address (HERE)  
 W = ?  
**After Instruction**  
 If REG < W;  
 PC = Address (LESS)  
 If REG ≥ W;  
 PC = Address (NLESS)

# PIC18F1230/1330

## DAW Decimal Adjust W Register

Syntax: DAW

Operands: None

Operation: If  $[W<3:0> > 9]$  or  $[DC = 1]$  then,  
 $(W<3:0>) + 6 \rightarrow W<3:0>$ ;  
 else,  
 $(W<3:0>) \rightarrow W<3:0>$

If  $[W<7:4> + DC > 9]$  or  $[C = 1]$  then,  
 $(W<7:4>) + 6 + DC \rightarrow W<7:4>$ ;  
 else,  
 $(W<7:4>) + DC \rightarrow W<7:4>$

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	=	A5h
C	=	0
DC	=	0
After Instruction		
W	=	05h
C	=	1
DC	=	0

**Example 2:**

Before Instruction		
W	=	CEh
C	=	0
DC	=	0
After Instruction		
W	=	34h
C	=	1
DC	=	0

## DECF Decrement f

Syntax: DECF f{,d{,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(f) - 1 \rightarrow \text{dest}$

Status Affected: C, DC, N, OV, Z

Encoding: 

0000	01da	ffff	ffff
------	------	------	------

Description: Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:**

DECF CNT, 1, 0

Before Instruction		
CNT	=	01h
Z	=	0
After Instruction		
CNT	=	00h
Z	=	1

**DECFSZ**      **Decrement f, Skip if 0**

Syntax:      DECFSZ f {,d {,a}}

Operands:       $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:       $(f) - 1 \rightarrow \text{dest}$ ,  
 skip if result = 0

Status Affected:      None

Encoding:      

0010	11da	ffff	ffff
------	------	------	------

Description:      The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 1
            GOTO      LOOP
            CONTINUE
    
```

Before Instruction  
 PC = Address (HERE)

After Instruction  
 CNT = CNT - 1  
 If CNT = 0;  
   PC = Address (CONTINUE)  
 If CNT  $\neq$  0;  
   PC = Address (HERE + 2)

**DCFSNZ**      **Decrement f, Skip if Not 0**

Syntax:      DCFSNZ f {,d {,a}}

Operands:       $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:       $(f) - 1 \rightarrow \text{dest}$ ,  
 skip if result  $\neq$  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'. 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 0
ZERO      :
NZERO     :
    
```

Before Instruction  
 TEMP = ?

After Instruction  
 TEMP = TEMP - 1  
 If TEMP = 0;  
   PC = Address (ZERO)  
 If TEMP  $\neq$  0;  
   PC = Address (NZERO)

# PIC18F1230/1330

## GOTO Unconditional Branch

Syntax: GOTO k  
 Operands:  $0 \leq k \leq 1048575$   
 Operation:  $k \rightarrow PC<20:1>$   
 Status Affected: None

Encoding:

1110	1111	$k_7kkk$	$kkkk_0$
1111	$k_{19}kkk$	$kkkk$	$kkkk_8$

1st word ( $k<7:0>$ )  
 2nd word ( $k<19:8>$ )

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: INCF f,{d},{a}  
 Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(f) + 1 \rightarrow \text{dest}$

Status Affected: C, DC, N, OV, Z

Encoding:

0010	10da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1  
 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: INCF CNT, 1, 0

Before Instruction  
 CNT = FFh  
 Z = 0  
 C = ?  
 DC = ?

After Instruction  
 CNT = 00h  
 Z = 1  
 C = 1  
 DC = 1

**INCFSZ**      **Increment f, Skip if 0**

Syntax:      INCFSZ f {,d {,a}}

Operands:     $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:     $(f) + 1 \rightarrow \text{dest}$ ,  
skip if result = 0

Status Affected:    None

Encoding:    

0011	11da	ffff	ffff
------	------	------	------

Description:    The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'.  
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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 0  
                  NZERO    :  
                  ZERO     :

Before Instruction  
PC = Address (HERE)

After Instruction  
CNT = CNT + 1  
If CNT = 0;  
PC = Address (ZERO)  
If CNT  $\neq$  0;  
PC = Address (NZERO)

**INFSNZ**      **Increment f, Skip if Not 0**

Syntax:      INFSNZ f {,d {,a}}

Operands:     $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:     $(f) + 1 \rightarrow \text{dest}$ ,  
skip if result  $\neq$  0

Status Affected:    None

Encoding:    

0100	10da	ffff	ffff
------	------	------	------

Description:    The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'.  
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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 0  
                  ZERO    :  
                  NZERO   :

Before Instruction  
PC = Address (HERE)

After Instruction  
REG = REG + 1  
If REG  $\neq$  0;  
PC = Address (NZERO)  
If REG = 0;  
PC = Address (ZERO)

# PIC18F1230/1330

## IORLW Inclusive OR Literal with W

Syntax: IORLW k

Operands:  $0 \leq k \leq 255$

Operation: (W) .OR. k  $\rightarrow$  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 35h

Before Instruction

W = 9Ah

After Instruction

W = BFh

## IORWF Inclusive OR W with f

Syntax: IORWF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation: (W) .OR. (f)  $\rightarrow$  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'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** IORWF RESULT, 0, 1

Before Instruction

RESULT = 13h

W = 91h

After Instruction

RESULT = 13h

W = 93h

**LFSR**                      **Load FSR**

---

Syntax:                      LFSR f, k

Operands:                     $0 \leq f \leq 2$   
 $0 \leq k \leq 4095$

Operation:                     $k \rightarrow \text{FSRf}$

Status Affected:            None

Encoding:                    

1110	1110	00ff	$k_{11}kkk$
1111	0000	$k_7kkk$	$kkkk$

Description:                The 12-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.

Words:                        2

Cycles:                        2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH	
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL	

Example:                      LFSR 2, 3ABh

After Instruction

FSR2H	=	03h
FSR2L	=	ABh

**MOVF**                      **Move f**

---

Syntax:                      MOVF f {,d {,a}}

Operands:                     $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:                     $f \rightarrow \text{dest}$

Status Affected:            N, Z

Encoding:                    

0101	00da	ffff	ffff
------	------	------	------

Description:                The contents of register 'f' 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'. Location 'f' can be anywhere in the 256-byte bank.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words:                        1

Cycles:                        1

Q Cycle Activity:

	Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write W	

Example:                      MOVF REG, 0, 0

Before Instruction

REG	=	22h
W	=	FFh

After Instruction

REG	=	22h
W	=	22h

# PIC18F1230/1330

## MOVFF Move f to f

**Syntax:** MOVFF  $f_s, f_d$

**Operands:**  $0 \leq f_s \leq 4095$   
 $0 \leq f_d \leq 4095$

**Operation:**  $(f_s) \rightarrow f_d$

**Status Affected:** None

**Encoding:**

1100	ffff	ffff	ffff <sub>s</sub>
1111	ffff	ffff	ffff <sub>d</sub>

**1st word (source)**  
**2nd word (destin.)**

**Description:** The contents of source register 'f<sub>s</sub>' are moved to destination register 'f<sub>d</sub>'. Location of source 'f<sub>s</sub>' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f<sub>d</sub>' 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.

**Words:** 2

**Cycles:** 2 (3)

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

**Example:** MOVFF REG1, REG2

Before Instruction  
 REG1 = 33h  
 REG2 = 11h

After Instruction  
 REG1 = 33h  
 REG2 = 33h

## MOVLB Move Literal to Low Nibble in BSR

**Syntax:** MOVLW k

**Operands:**  $0 \leq k \leq 255$

**Operation:**  $k \rightarrow \text{BSR}$

**Status Affected:** None

**Encoding:**

0000	0001	kkkk	kkkk
------	------	------	------

**Description:** The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of k<sub>7:k<sub>4</sub></sub>.

**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 = 02h

After Instruction  
 BSR Register = 05h



## MOVLW Move Literal to W

Syntax: MOVLW k

Operands:  $0 \leq k \leq 255$

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 5Ah

After Instruction

W = 5Ah

## MOVWF Move W to f

Syntax: MOVWF f{,a}

Operands:  $0 \leq f \leq 255$   
 $a \in [0,1]$

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 is selected.  
If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: MOVWF REG, 0

Before Instruction

W = 4Fh

REG = FFh

After Instruction

W = 4Fh

REG = 4Fh

# PIC18F1230/1330

## MULLW Multiply Literal with W

Syntax: MULLW k

Operands:  $0 \leq k \leq 255$

Operation:  $(W) \times k \rightarrow \text{PRODH:PRODL}$

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 0C4h

Before Instruction

W = E2h  
PRODH = ?  
PRODL = ?

After Instruction

W = E2h  
PRODH = ADh  
PRODL = 08h

## MULWF Multiply W with f

Syntax: MULWF f {,a}

Operands:  $0 \leq f \leq 255$   
 $a \in [0,1]$

Operation:  $(W) \times (f) \rightarrow \text{PRODH:PRODL}$

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 is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH:PRODL

Example: MULWF REG, 1

Before Instruction

W = C4h  
REG = B5h  
PRODH = ?  
PRODL = ?

After Instruction

W = C4h  
REG = B5h  
PRODH = 8Ah  
PRODL = 94h

## NEGF

## Negate f

**Syntax:** NEGF f{,a}

**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**Operation:**  $(\bar{f}) + 1 \rightarrow f$

**Status Affected:** N, OV, C, DC, Z

**Encoding:**

0110	110a	ffff	ffff
------	------	------	------

**Description:** Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**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 [3Ah]

After Instruction

REG = 1100 0110 [C6h]

## NOP

## No Operation

**Syntax:** NOP

**Operands:** None

**Operation:** No operation

**Status Affected:** None

**Encoding:**

0000	0000	0000	0000
1111	xxxx	xxxx	xxxx

**Description:** No operation.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

**Example:**

None.

# PIC18F1230/1330

## POP Pop Top of Return Stack

Syntax: POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding: 

0000	0000	0000	0110
------	------	------	------

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

**Example:**

	POP	
	GOTO	NEW
Before Instruction		
TOS	=	0031A2h
Stack (1 level down)	=	014332h
After Instruction		
TOS	=	014332h
PC	=	NEW

## PUSH Push Top of Return Stack

Syntax: PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding: 

0000	0000	0000	0101
------	------	------	------

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then 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	=	345Ah
PC	=	0124h
After Instruction		
PC	=	0126h
TOS	=	0126h
Stack (1 level down)	=	345Ah

**RCALL**                      **Relative Call**

---

Syntax:                      RCALL n

Operands:                     $-1024 \leq n \leq 1023$

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' 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:                      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

# PIC18F1230/1330

## RETFIE Return from Interrupt

**Syntax:** RETFIE {s}

**Operands:**  $s \in [0,1]$

**Operation:** (TOS) → PC,  
 1 → GIE/GIEH or PEIE/GIEL;  
 if  $s = 1$ ,  
 (WS) → W,  
 (STATUS) → STATUS,  
 (BSRS) → BSR,  
 PCLATU, PCLATH are unchanged

**Status Affected:** 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, STATUS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs.

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	No operation	No operation	POP PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

**Example:** RETFIE 1

After Interrupt

PC	=	TOS
W	=	WS
BSR	=	BSRS
STATUS	=	STATUS
GIE/GIEH, PEIE/GIEL	=	1

## RETLW Return Literal to W

**Syntax:** RETLW k

**Operands:**  $0 \leq k \leq 255$

**Operation:**  $k \rightarrow W$ ,  
 (TOS) → PC,  
 PCLATU, PCLATH are unchanged

**Status Affected:** None

**Encoding:**

0000	1100	kkkk	kkkk
------	------	------	------

**Description:** W is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	POP PC from stack, Write to W
No operation	No operation	No operation	No operation

**Example:**

```
CALL TABLE ; 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 = 07h

After Instruction  
 W = value of kn

## RETURN Return from Subroutine

**Syntax:** RETURN {s}

**Operands:** s ∈ [0,1]

**Operation:** (TOS) → PC;  
if s = 1,  
(WS) → W,  
(STATUS) → STATUS,  
(BSRS) → BSR,  
PCLATU, PCLATH are unchanged

**Status Affected:** None

**Encoding:**

0000	0000	0001	001s
------	------	------	------

**Description:** Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's' = 1, the contents of the shadow registers, WS, STATUS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs.

**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 Instruction:  
PC = TOS

## RLCF Rotate Left f through Carry

**Syntax:** RLCF f {,d {,a}}

**Operands:** 0 ≤ f ≤ 255  
d ∈ [0,1]  
a ∈ [0,1]

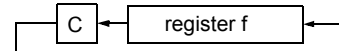
**Operation:** (f<n>) → dest<n + 1>,  
(f<7>) → C,  
(C) → dest<0>

**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'.  
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



**Words:** 1

**Cycles:** 1

Q Cycle Activity:

	Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination	

**Example:** RLCF REG, 0, 0

Before Instruction  
REG = 1110 0110  
C = 0

After Instruction  
REG = 1110 0110  
W = 1100 1100  
C = 1

# PIC18F1230/1330

## RLNCF Rotate Left f (No Carry)

Syntax: RLNCF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

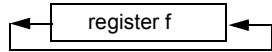
Operation:  $(f\langle n \rangle) \rightarrow \text{dest}\langle n + 1 \rangle$ ,  
 $(f\langle 7 \rangle) \rightarrow \text{dest}\langle 0 \rangle$

Status Affected: N, Z

Encoding: 

0100	01da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** RLNCF REG, 1, 0

Before Instruction  
 REG = 1010 1011  
 After Instruction  
 REG = 0101 0111

## RRCF Rotate Right f through Carry

Syntax: RRCF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

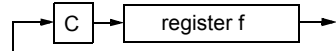
Operation:  $(f\langle n \rangle) \rightarrow \text{dest}\langle n - 1 \rangle$ ,  
 $(f\langle 0 \rangle) \rightarrow C$ ,  
 $(C) \rightarrow \text{dest}\langle 7 \rangle$

Status Affected: C, N, Z

Encoding: 

0011	00da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** RRCF REG, 0, 0

Before Instruction  
 REG = 1110 0110  
 C = 0  
 After Instruction  
 REG = 1110 0110  
 W = 0111 0011  
 C = 0



## RRNCF Rotate Right f (No Carry)

**Syntax:** RRNCF f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

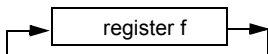
**Operation:** (f<n>) → dest<n – 1>,  
(f<0>) → dest<7>

**Status Affected:** N, Z

**Encoding:**

0100	00da	ffff	ffff
------	------	------	------

**Description:** The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'.  
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.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



**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, 0, 0

Before Instruction  
W = ?  
REG = 1101 0111  
After Instruction  
W = 1110 1011  
REG = 1101 0111

## SETF Set f

**Syntax:** SETF f {,a}

**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**Operation:** FFh → f

**Status Affected:** None

**Encoding:**

0110	100a	ffff	ffff
------	------	------	------

**Description:** The contents of the specified register are set to FFh.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

**Example:** SETF REG, 1

Before Instruction  
REG = 5Ah  
After Instruction  
REG = FFh

# PIC18F1230/1330

## SLEEP Enter Sleep mode

Syntax: 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: SUBFWB f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(W) - (f) - (\overline{C}) \rightarrow \text{dest}$

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'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example 1:** SUBFWB REG, 1, 0

Before Instruction

REG = 3  
W = 2  
C = 1

After Instruction

REG = FF  
W = 2  
C = 0  
Z = 0  
N = 1 ; 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

**SUBLW**                    **Subtract W from Literal**

---

Syntax:                    SUBLW k

Operands:                 $0 \leq k \leq 255$

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 02h

Before Instruction  
W = 01h  
C = ?

After Instruction  
W = 01h  
C = 1 ; result is positive  
Z = 0  
N = 0

**Example 2:**                    SUBLW 02h

Before Instruction  
W = 02h  
C = ?

After Instruction  
W = 00h  
C = 1 ; result is zero  
Z = 1  
N = 0

**Example 3:**                    SUBLW 02h

Before Instruction  
W = 03h  
C = ?

After Instruction  
W = FFh ; (2's complement)  
C = 0 ; result is negative  
Z = 0  
N = 1

**SUBWF**                    **Subtract W from f**

---

Syntax:                    SUBWF f {,d {,a}}

Operands:                 $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:                 $(f) - (W) \rightarrow \text{dest}$

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'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words:                    1

Cycles:                    1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example 1:**                    SUBWF REG, 1, 0

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, 0, 0

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, 1, 0

Before Instruction  
REG = 1  
W = 2  
C = ?

After Instruction  
REG = FFh ; (2's complement)  
W = 2  
C = 0 ; result is negative  
Z = 0  
N = 1

# PIC18F1230/1330

## SUBWFB Subtract W from f with Borrow

**Syntax:** SUBWFB f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(f) - (W) - (\overline{C}) \rightarrow \text{dest}$

**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'.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**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 = 19h (0001 1001)  
 W = 0Dh (0000 1101)  
 C = 1

**After Instruction**  
 REG = 0Ch (0000 1011)  
 W = 0Dh (0000 1101)  
 C = 1  
 Z = 0  
 N = 0 ; result is positive

**Example 2:** SUBWFB REG, 0, 0

**Before Instruction**  
 REG = 1Bh (0001 1011)  
 W = 1Ah (0001 1010)  
 C = 0

**After Instruction**  
 REG = 1Bh (0001 1011)  
 W = 00h  
 C = 1  
 Z = 1 ; result is zero  
 N = 0

**Example 3:** SUBWFB REG, 1, 0

**Before Instruction**  
 REG = 03h (0000 0011)  
 W = 0Eh (0000 1101)  
 C = 1

**After Instruction**  
 REG = F5h (1111 0100)  
 ; [2's comp]  
 W = 0Eh (0000 1101)  
 C = 0  
 Z = 0  
 N = 1 ; result is negative

## SWAPF Swap f

**Syntax:** SWAPF f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(f<3:0>) \rightarrow \text{dest}<7:4>$ ,  
 $(f<7:4>) \rightarrow \text{dest}<3:0>$

**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'.  
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** SWAPF REG, 1, 0

**Before Instruction**  
 REG = 53h

**After Instruction**  
 REG = 35h

## TBLRD Table Read

**Syntax:** TBLRD (\*; \*+; \*-; +\*)

**Operands:** None

**Operation:** if TBLRD \*,  
(Prog Mem (TBLPTR)) → TABLAT,  
TBLPTR – No Change;  
if TBLRD \*+,  
(Prog Mem (TBLPTR)) → TABLAT,  
(TBLPTR) + 1 → TBLPTR;  
if TBLRD \*-,  
(Prog Mem (TBLPTR)) → TABLAT,  
(TBLPTR) – 1 → TBLPTR;  
if TBLRD +\*,  
(TBLPTR) + 1 → TBLPTR,  
(Prog Mem (TBLPTR)) → TABLAT

**Status Affected:** None

Encoding:	0000	0000	0000	10nn
				nn=0 *
				=1 *+
				=2 *-
				=3 +*

**Description:** This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

	Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)	

## TBLRD Table Read (Continued)

**Example 1:** TBLRD \*+ ;

Before Instruction

TABLAT = 55h  
TBLPTR = 00A356h  
MEMORY (00A356h) = 34h

After Instruction

TABLAT = 34h  
TBLPTR = 00A357h

**Example 2:** TBLRD +\* ;

Before Instruction

TABLAT = AAh  
TBLPTR = 01A357h  
MEMORY (01A357h) = 12h  
MEMORY (01A358h) = 34h

After Instruction

TABLAT = 34h  
TBLPTR = 01A358h

# PIC18F1230/1330

## TBLWT Table Write

Syntax: 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 nn=0 * =1 ** =2 *- =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 7.0 “Flash Program Memory”** for additional details on programming Flash memory.) The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSB of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1  
Cycles: 2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation	No operation (Write to Holding Register)

## TBLWT Table Write (Continued)

Example 1: TBLWT \*\*;

Before Instruction

TABLAT	=	55h
TBLPTR	=	00A356h
HOLDING REGISTER (00A356h)	=	FFh

After Instructions (table write completion)

TABLAT	=	55h
TBLPTR	=	00A357h
HOLDING REGISTER (00A356h)	=	55h

Example 2: TBLWT \*\*;

Before Instruction

TABLAT	=	34h
TBLPTR	=	01389Ah
HOLDING REGISTER (01389Ah)	=	FFh
HOLDING REGISTER (01389Bh)	=	FFh

After Instruction (table write completion)

TABLAT	=	34h
TBLPTR	=	01389Bh
HOLDING REGISTER (01389Ah)	=	FFh
HOLDING REGISTER (01389Bh)	=	34h

**TSTFSZ**      **Test f, Skip if 0**

Syntax:      TSTFSZ f {,a}

Operands:     $0 \leq f \leq 255$   
 $a \in [0,1]$

Operation:    skip if  $f = 0$

Status Affected:    None

Encoding:    

0110	011a	ffff	ffff
------	------	------	------

Description:    If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1
NZERO    :
ZERO     :
```

Before Instruction

PC = Address (HERE)

After Instruction

```

If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)
```

**XORLW**      **Exclusive OR Literal with W**

Syntax:      XORLW k

Operands:     $0 \leq k \leq 255$

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    0AFh

Before Instruction

W = B5h

After Instruction

W = 1Ah

# PIC18F1230/1330

---

## XORWF Exclusive OR W with f

---

Syntax: XORWF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

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 the register 'f'.  
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 22.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** XORWF REG, 1, 0

Before Instruction

REG = AFh

W = B5h

After Instruction

REG = 1Ah

W = B5h



## 22.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F1230/1330 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set (with the exception of `CALLW`, `MOVSF` and `MOVSS`) can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- Dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- Function Pointer invocation
- Software Stack Pointer manipulation
- Manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 22-3. Detailed descriptions are provided in **Section 22.2.2 “Extended Instruction Set”**. The opcode field descriptions in Table 22-1 (page 216) apply to both the standard and extended PIC18 instruction sets.

**Note:** The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in the assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

### 22.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets (“[]”). This is done to indicate that the argument is used as an index or offset. The MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 22.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”**.

**Note:** In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces (“{ }”).

**TABLE 22-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected
			MSb			LSb	
ADDFSR f, k	Add Literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK k	Add Literal to FSR2 and Return	2	1110	1000	11kk	kkkk	None
CALLW	Call Subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF z <sub>s</sub> , f <sub>d</sub>	Move z <sub>s</sub> (source) to 1st word f <sub>d</sub> (destination) 2nd word	2	1110	1011	0zzz	zzzz	None
MOVSS z <sub>s</sub> , z <sub>d</sub>	Move z <sub>s</sub> (source) to 1st word z <sub>d</sub> (destination) 2nd word	2	1110	1011	1zzz	zzzz	None
PUSHL k	Store Literal at FSR2, Decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR f, k	Subtract Literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK k	Subtract Literal from FSR2 and Return	2	1110	1001	11kk	kkkk	None

# PIC18F1230/1330

## 22.2.2 EXTENDED INSTRUCTION SET

### ADDFSR Add Literal to FSR

Syntax: ADDFSR f, k

Operands:  $0 \leq k \leq 63$   
 $f \in [0, 1, 2]$

Operation:  $FSR(f) + k \rightarrow FSR(f)$

Status Affected: None

Encoding: 

1110	1000	ffkk	kkkk
------	------	------	------

Description: The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to FSR

**Example:** ADDFSR 2, 23h

Before Instruction  
 FSR2 = 03FFh

After Instruction  
 FSR2 = 0422h

### ADDULNK Add Literal to FSR2 and Return

Syntax: ADDULNK k

Operands:  $0 \leq k \leq 63$

Operation:  $FSR2 + k \rightarrow FSR2$ ,  
 (TOS)  $\rightarrow$  PC

Status Affected: None

Encoding: 

1110	1000	11kk	kkkk
------	------	------	------

Description: The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle. This may be thought of as a special case of the ADDFSR instruction, where  $f = 3$  (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to FSR
No Operation	No Operation	No Operation	No Operation

**Example:** ADDULNK 23h

Before Instruction  
 FSR2 = 03FFh  
 PC = 0100h

After Instruction  
 FSR2 = 0422h  
 PC = (TOS)

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

**CALLW Subroutine Call Using WREG**

Syntax: CALLW

Operands: None

Operation: (PC + 2) → TOS,  
(W) → PCL,  
(PCLATH) → PCH,  
(PCLATU) → PCU

Status Affected: None

Encoding: 

0000	0000	0001	0100
------	------	------	------

Description: First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched. Unlike CALL, there is no option to update W, STATUS or BSR.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read WREG	PUSH PC to stack	No operation
No operation	No operation	No operation	No operation

**Example:**                    HERE        CALLW

Before Instruction

PC        =    address (HERE)

PCLATH =   10h

PCLATU =   00h

W         =   06h

After Instruction

PC        =   001006h

TOS       =   address (HERE + 2)

PCLATH =   10h

PCLATU =   00h

W         =   06h

**MOVSF Move Indexed to f**

Syntax: MOVSF [z<sub>s</sub>], f<sub>d</sub>

Operands: 0 ≤ z<sub>s</sub> ≤ 127  
0 ≤ f<sub>d</sub> ≤ 4095

Operation: ((FSR2) + z<sub>s</sub>) → f<sub>d</sub>

Status Affected: None

Encoding: 

1110	1011	0zzz	zzzz <sub>s</sub>
1111	ffff	ffff	ffff <sub>d</sub>

Description: The contents of the source register are moved to destination register 'f<sub>d</sub>'. The actual address of the source register is determined by adding the 7-bit literal offset 'z<sub>s</sub>' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal 'f<sub>d</sub>' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh). The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register. If the resultant source address points to an indirect addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

**Example:**                    MOVSF    [05h], REG2

Before Instruction

FSR2       =   80h

Contents of 85h = 33h

REG2       =  11h

After Instruction

FSR2       =   80h

Contents of 85h = 33h

REG2       =  33h

# PIC18F1230/1330

## MOVSS Move Indexed to Indexed

**Syntax:** MOVSS [z<sub>s</sub>], [z<sub>d</sub>]

**Operands:** 0 ≤ z<sub>s</sub> ≤ 127  
0 ≤ z<sub>d</sub> ≤ 127

**Operation:** ((FSR2) + z<sub>s</sub>) → ((FSR2) + z<sub>d</sub>)

**Status Affected:** None

**Encoding:**

1110	1011	1zzz	zzzz <sub>s</sub>
1111	xxxx	xzzz	zzzz <sub>d</sub>

**1st word (source)**  
**2nd word (dest.)**

**Description**

The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets 'z<sub>s</sub>' or 'z<sub>d</sub>', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).

The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.

**Words:** 2  
**Cycles:** 2

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	Determine dest addr	Determine dest addr	Write to dest reg

**Example:** MOVSS [05h], [06h]

**Before Instruction**

FSR2 = 80h  
 Contents of 85h = 33h  
 Contents of 86h = 11h

**After Instruction**

FSR2 = 80h  
 Contents of 85h = 33h  
 Contents of 86h = 33h

## PUSHL Store Literal at FSR2, Decrement FSR2

**Syntax:** PUSHL k

**Operands:** 0 ≤ k ≤ 255

**Operation:** k → (FSR2),  
FSR2 – 1 → FSR2

**Status Affected:** None

**Encoding:**

1110	1010	kkkk	kkkk
------	------	------	------

**Description:** The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.

**Words:** 1  
**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process data	Write to destination

**Example:** PUSHL 08h

**Before Instruction**

FSR2H:FSR2L = 01ECh  
 Memory (01ECh) = 00h

**After Instruction**

FSR2H:FSR2L = 01EBh  
 Memory (01ECh) = 08h

## SUBFSR Subtract Literal from FSR

Syntax: SUBFSR f, k  
 Operands:  $0 \leq k \leq 63$   
 $f \in [0, 1, 2]$   
 Operation:  $FSR(f - k) \rightarrow FSR(f)$   
 Status Affected: None  
 Encoding: 

1110	1001	f f k k	k k k k
------	------	---------	---------

  
 Description: The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.  
 Words: 1  
 Cycles: 1  
 Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** SUBFSR 2, 23h

Before Instruction  
 FSR2 = 03FFh  
 After Instruction  
 FSR2 = 03DCh

## SUBULNK Subtract Literal from FSR2 and Return

Syntax: SUBULNK k  
 Operands:  $0 \leq k \leq 63$   
 Operation:  $FSR2 - k \rightarrow FSR2$ ,  
 (TOS)  $\rightarrow PC$   
 Status Affected: None  
 Encoding: 

1110	1001	11 k k	k k k k
------	------	--------	---------

  
 Description: The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle. This may be thought of as a special case of the SUBFSR instruction, where  $f = 3$  (binary '11'); it operates only on FSR2.  
 Words: 1  
 Cycles: 2  
 Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination
No Operation	No Operation	No Operation	No Operation

**Example:** SUBULNK 23h

Before Instruction  
 FSR2 = 03FFh  
 PC = 0100h  
 After Instruction  
 FSR2 = 03DCh  
 PC = (TOS)

# PIC18F1230/1330

---

## 22.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

**Note:** Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode (**Section 6.5.1 “Indexed Addressing with Literal Offset”**). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank (*'a' = 0*) or in a GPR bank designated by the BSR (*'a' = 1*). When the extended instruction set is enabled and *'a' = 0*, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see **Section 22.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”**).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing mode.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

### 22.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument, *'f'*, in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value, *'k'*. As already noted, this occurs only when *'f'* is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets (“[ ]”). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within brackets, will generate an error in the MPASM Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing mode, the Access RAM argument is never specified; it will automatically be assumed to be *'0'*. This is in contrast to standard operation (extended instruction set disabled) when *'a'* is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM Assembler.

The destination argument, *'d'*, functions as before.

In the latest versions of the MPASM Assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, */y*, or the PE directive in the source listing.

## 22.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F1230/1330, it is very important to consider the type of code. A large, re-entrant application that is written in *'C'* and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

## ADDWF **ADD W to Indexed (Indexed Literal Offset mode)**

**Syntax:** ADDWF [k] {,d}

**Operands:**  $0 \leq k \leq 95$   
 $d \in [0,1]$

**Operation:**  $(W) + ((FSR2) + k) \rightarrow \text{dest}$

**Status Affected:** N, OV, C, DC, Z

**Encoding:**

0010	01d0	kkkk	kkkk
------	------	------	------

**Description:** The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process Data	Write to destination

**Example:** ADDWF [OFST], 0

**Before Instruction**

W	=	17h
OFST	=	2Ch
FSR2	=	0A00h
Contents of 0A2Ch	=	20h

**After Instruction**

W	=	37h
Contents of 0A2Ch	=	20h

## BSF **Bit Set Indexed (Indexed Literal Offset mode)**

**Syntax:** BSF [k], b

**Operands:**  $0 \leq f \leq 95$   
 $0 \leq b \leq 7$

**Operation:**  $1 \rightarrow ((FSR2) + k) \langle b \rangle$

**Status Affected:** None

**Encoding:**

1000	bbb0	kkkk	kkkk
------	------	------	------

**Description:** Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

**Example:** BSF [FLAG\_OFST], 7

**Before Instruction**

FLAG_OFST	=	0Ah
FSR2	=	0A00h
Contents of 0A0Ah	=	55h

**After Instruction**

Contents of 0A0Ah	=	D5h
-------------------	---	-----

## SETF **Set Indexed (Indexed Literal Offset mode)**

**Syntax:** SETF [k]

**Operands:**  $0 \leq k \leq 95$

**Operation:**  $FFh \rightarrow ((FSR2) + k)$

**Status Affected:** None

**Encoding:**

0110	1000	kkkk	kkkk
------	------	------	------

**Description:** The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process Data	Write register

**Example:** SETF [OFST]

**Before Instruction**

OFST	=	2Ch
FSR2	=	0A00h
Contents of 0A2Ch	=	00h

**After Instruction**

Contents of 0A2Ch	=	FFh
-------------------	---	-----

# PIC18F1230/1330

---

## 22.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18F1230/1330 family of devices. This includes the MPLAB C18 C Compiler, MPASM Assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.



## 23.0 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings<sup>(†)</sup>

Ambient temperature under bias .....	-40°C to +125°C
Storage temperature .....	-65°C to +150°C
Voltage on any pin with respect to V <sub>SS</sub> (except V <sub>DD</sub> and $\overline{\text{MCLR}}$ ) .....	-0.3V to (V <sub>DD</sub> + 0.3V)
Voltage on V <sub>DD</sub> with respect to V <sub>SS</sub> .....	-0.3V to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to V <sub>SS</sub> ( <b>Note 2</b> ) .....	0V to +13.25V
Total power dissipation ( <b>Note 1</b> ) .....	1.0W
Maximum current out of V <sub>SS</sub> pin .....	300 mA
Maximum current into V <sub>DD</sub> pin .....	250 mA
Input clamp current, I <sub>IK</sub> (V <sub>I</sub> < 0 or V <sub>I</sub> > V <sub>DD</sub> ) .....	±20 mA
Output clamp current, I <sub>OK</sub> (V <sub>O</sub> < 0 or V <sub>O</sub> > V <sub>DD</sub> ) .....	±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 .....	200 mA
Maximum current sourced by all ports .....	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<sub>SS</sub> at the  $\overline{\text{MCLR}}$ /V<sub>PP</sub>/RA5/ $\overline{\text{FLTA}}$  pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a “low” level to the  $\overline{\text{MCLR}}$ /V<sub>PP</sub>/RA5/ $\overline{\text{FLTA}}$  pin, rather than pulling this pin directly to V<sub>SS</sub>.

† **NOTICE:** Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

# PIC18F1230/1330

FIGURE 23-1: PIC18F1230/1330 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

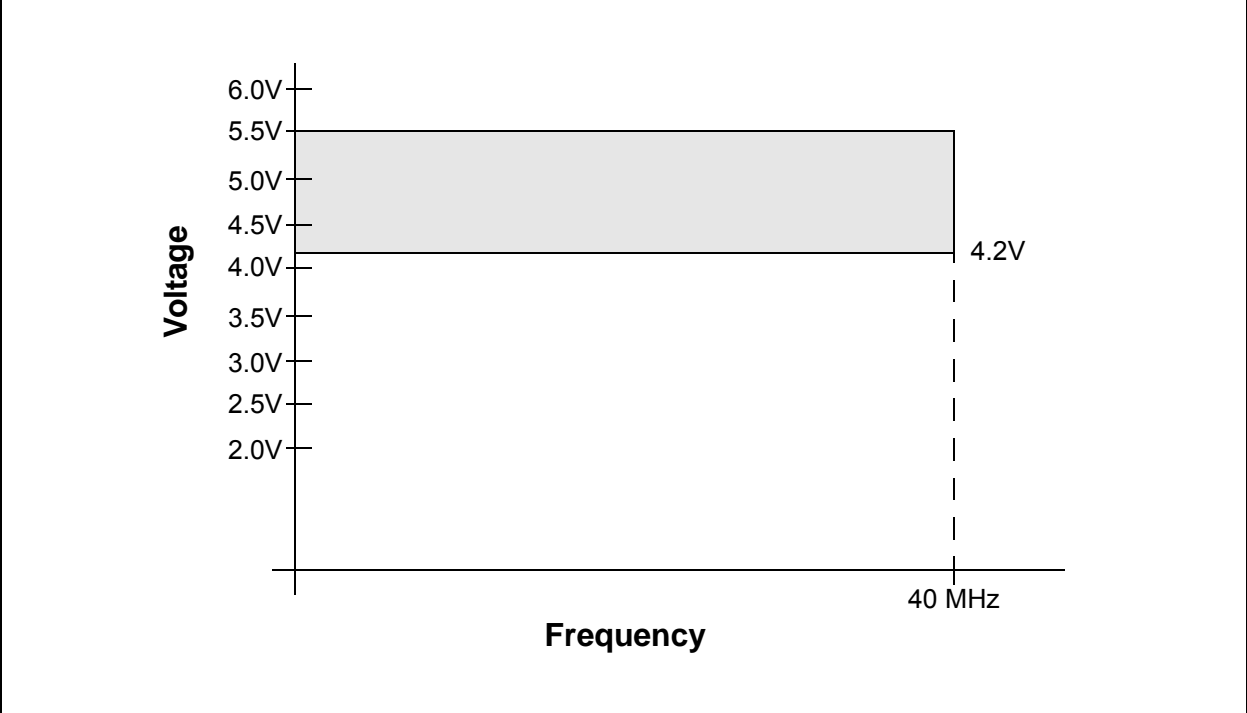


FIGURE 23-2: PIC18F1230/1330 VOLTAGE-FREQUENCY GRAPH (EXTENDED)

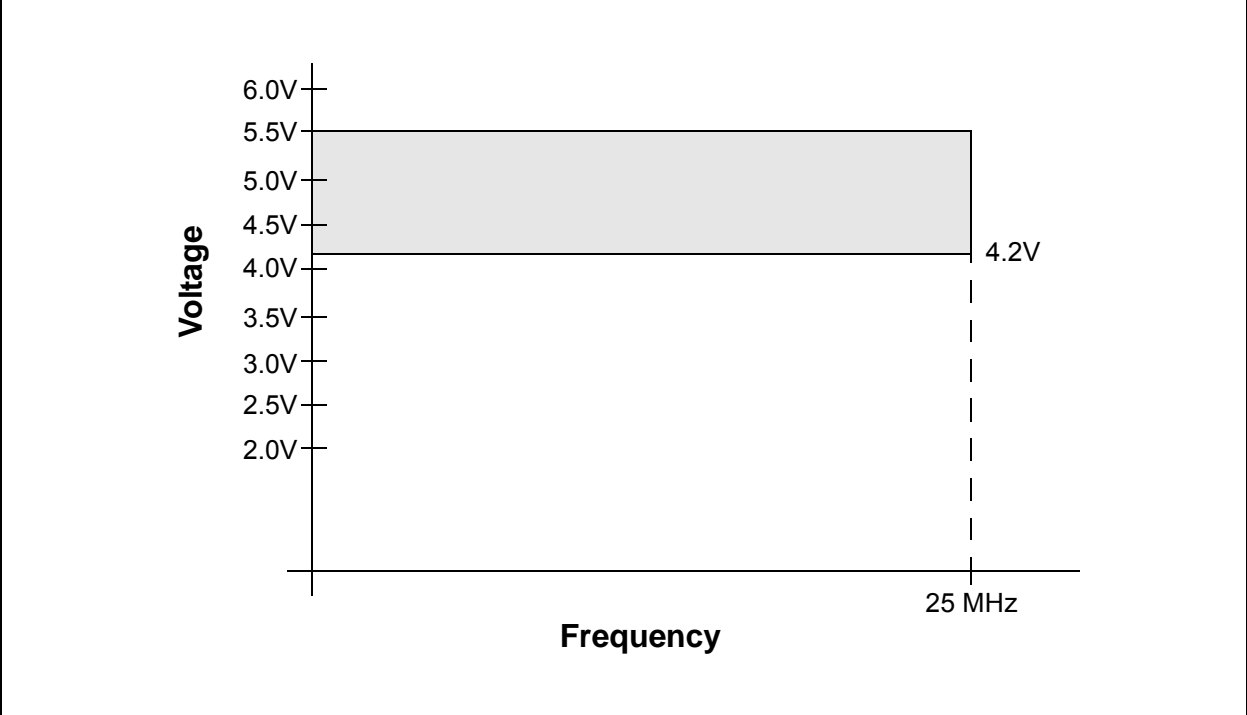


FIGURE 23-3: PIC18LF1230/1330 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



# PIC18F1230/1330

## 23.1 DC Characteristics: Supply Voltage PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
D001	VDD	<b>Supply Voltage</b>					
		PIC18LF1230/1330	2.0	—	5.5	V	HS, XT, RC and LP Oscillator modes
		PIC18F1230/1330	4.2	—	5.5	V	
D001C	AVDD	<b>Analog Supply Voltage</b>	VDD - 0.3	—	VDD + 0.3	V	
D001D	AVSS	<b>Analog Ground Voltage</b>	VSS - 0.3	—	VSS + 0.3	V	
D002	VDR	<b>RAM Data Retention Voltage<sup>(1)</sup></b>	1.5	—	—	V	
D003	VPOR	<b>VDD Start Voltage</b> to ensure internal Power-on Reset signal	—	—	0.7	V	See section on Power-on Reset for details
D004	SVDD	<b>VDD Rise Rate</b> to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See section on Power-on Reset for details
D005	VBOR	<b>Brown-out Reset Voltage</b>					
		PIC18LF1230/1330					
		BORV1:BORV0 = 11	2.00	2.05	2.16	V	
D005	VBOR	BORV1:BORV0 = 10	2.65	2.79	2.93	V	
		All devices					
		BORV1:BORV0 = 01	4.11 <sup>(2)</sup>	4.33	4.55	V	
		BORV1:BORV0 = 00	4.36	4.59	4.82	V	

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

**Note 2:** With BOR enabled, full-speed operation (FOSC = 40 MHz) is supported until a BOR occurs. This is valid although VDD may be below the minimum voltage for this frequency.

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial				
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Power-Down Current (IPD)<sup>(1)</sup></b>						
	PIC18LF1230/1330	100	742	nA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V (Sleep mode)
		0.1	0.742	μA	$+25^{\circ}\text{C}$	
		0.2	4.80	μA	$+85^{\circ}\text{C}$	
	PIC18LF1230/1330	0.1	1.20	μA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V (Sleep mode)
		0.1	1.20	μA	$+25^{\circ}\text{C}$	
		0.3	7.80	μA	$+85^{\circ}\text{C}$	
	All devices	0.1	7.79	μA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V (Sleep mode)
		0.1	7.79	μA	$+25^{\circ}\text{C}$	
		0.4	14.8	μA	$+85^{\circ}\text{C}$	
	Extended devices only	10	119	μA	$+125^{\circ}\text{C}$	

- Note 1:** 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 V<sub>DD</sub> or V<sub>SS</sub> and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all I<sub>DD</sub> measurements in active operation mode are:  
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>;  
MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F1230/1330

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

Param No.	Device	Typ	Max	Units	Conditions		
PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Supply Current ( $I_{DD}$ ) <sup>(2)</sup>							
PIC18LF1230/1330		15	28.1	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{DD} = 2.0\text{V}$	FOSC = 31 kHz (RC_RUN mode, INTRC source)
		15	28.1	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		15	28.1	$\mu\text{A}$	$+85^{\circ}\text{C}$		
PIC18LF1230/1330		40	54	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{DD} = 3.0\text{V}$	
		35	54	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		30	54	$\mu\text{A}$	$+85^{\circ}\text{C}$		
All devices		105	149	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{DD} = 5.0\text{V}$	
		90	149	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		80	149	$\mu\text{A}$	$+85^{\circ}\text{C}$		
Extended devices only		80	249	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18LF1230/1330		0.32	0.93	$\text{mA}$	$-40^{\circ}\text{C}$	$V_{DD} = 2.0\text{V}$	FOSC = 1 MHz (RC_RUN mode, INTOSC source)
		0.33	0.93	$\text{mA}$	$+25^{\circ}\text{C}$		
		0.33	0.93	$\text{mA}$	$+85^{\circ}\text{C}$		
PIC18LF1230/1330		0.6	1.03	$\text{mA}$	$-40^{\circ}\text{C}$	$V_{DD} = 3.0\text{V}$	
		0.55	1.03	$\text{mA}$	$+25^{\circ}\text{C}$		
		0.6	1.03	$\text{mA}$	$+85^{\circ}\text{C}$		
All devices		1.1	2.03	$\text{mA}$	$-40^{\circ}\text{C}$	$V_{DD} = 5.0\text{V}$	
		1.1	2.03	$\text{mA}$	$+25^{\circ}\text{C}$		
		1.0	2.03	$\text{mA}$	$+85^{\circ}\text{C}$		
Extended devices only		1	3.3	$\text{mA}$	$+125^{\circ}\text{C}$		

- Note 1:** 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  $V_{DD}$  or  $V_{SS}$  and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all  $I_{DD}$  measurements in active operation mode are:  
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to  $V_{DD}$  or  $V_{SS}$ ;  
MCLR =  $V_{DD}$ ; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (I<sub>DD</sub>)<sup>(2)</sup></b>							
	PIC18LF1230/1330	0.8	1.83	mA	-40°C	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 4 MHz (RC_RUN mode, INTOSC source)
		0.8	1.83	mA	+25°C		
		0.8	1.83	mA	+85°C		
	PIC18LF1230/1330	1.3	2.93	mA	-40°C	V <sub>DD</sub> = 3.0V	
		1.3	2.93	mA	+25°C		
		1.3	2.93	mA	+85°C		
	All devices	2.5	4.73	mA	-40°C	V <sub>DD</sub> = 5.0V	
		2.5	4.73	mA	+25°C		
		2.5	4.73	mA	+85°C		
	Extended devices only	2.5	10.0	mA	+125°C		
	PIC18LF1230/1330	2.9	7.6	μA	-40°C	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 31 kHz (RC_IDLE mode, INTRC source)
		3.1	7.6	μA	+25°C		
		3.6	10.6	μA	+85°C		
	PIC18LF1230/1330	4.5	10.6	μA	-40°C	V <sub>DD</sub> = 3.0V	
		4.8	10.6	μA	+25°C		
		5.8	14.6	μA	+85°C		
	All devices	9.2	15.6	μA	-40°C	V <sub>DD</sub> = 5.0V	
		9.8	15.6	μA	+25°C		
		11.4	35.6	μA	+85°C		
	Extended devices only	21	179	μA	+125°C		

- Note 1:** 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 V<sub>DD</sub> or V<sub>SS</sub> and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all I<sub>DD</sub> measurements in active operation mode are:  
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>;  
MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F1230/1330

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (I<sub>DD</sub>)<sup>(2)</sup></b>							
PIC18LF1230/1330		165	347	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 1 MHz (RC_IDLE mode, INTOSC source)
		175	347	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		190	347	$\mu\text{A}$	$+85^{\circ}\text{C}$		
PIC18LF1230/1330		250	497	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V	
		270	497	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		290	497	$\mu\text{A}$	$+85^{\circ}\text{C}$		
All devices		500	930	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V	
		520	930	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		550	930	$\mu\text{A}$	$+85^{\circ}\text{C}$		
Extended devices only		0.6	2.9	mA	$+125^{\circ}\text{C}$		
PIC18LF1230/1330		340	497	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 4 MHz (RC_IDLE mode, INTOSC source)
		350	497	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		360	497	$\mu\text{A}$	$+85^{\circ}\text{C}$		
PIC18LF1230/1330		520	830	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V	
		540	830	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		580	830	$\mu\text{A}$	$+85^{\circ}\text{C}$		
All devices		1.0	1.33	mA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V	
		1.1	1.33	mA	$+25^{\circ}\text{C}$		
		1.1	1.33	mA	$+85^{\circ}\text{C}$		
Extended devices only		1.1	5.0	mA	$+125^{\circ}\text{C}$		

- Note 1:** 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 V<sub>DD</sub> or V<sub>SS</sub> and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all I<sub>DD</sub> measurements in active operation mode are:  
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>;  
MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.



## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial						
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended						
Param No.	Device	Typ	Max	Units	Conditions			
<b>Supply Current (I<sub>DD</sub>)<sup>(2)</sup></b>								
PIC18LF1230/1330		250	497	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 1 MHz ( <b>PRI_RUN</b> , EC oscillator)	
		260	497	$\mu\text{A}$	$+25^{\circ}\text{C}$			
		250	497	$\mu\text{A}$	$+85^{\circ}\text{C}$			
PIC18LF1230/1330		550	750	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V		
		480	750	$\mu\text{A}$	$+25^{\circ}\text{C}$			
		460	750	$\mu\text{A}$	$+85^{\circ}\text{C}$			
All devices		1.2	3	mA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V		
		1.1	3	mA	$+25^{\circ}\text{C}$			
		1.0	3	mA	$+85^{\circ}\text{C}$			
Extended devices only		1.0	3.0	mA	$+125^{\circ}\text{C}$			
PIC18LF1230/1330		0.72	1.93	mA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 4 MHz ( <b>PRI_RUN</b> , EC oscillator)	
		0.74	1.93	mA	$+25^{\circ}\text{C}$			
		0.74	1.93	mA	$+85^{\circ}\text{C}$			
PIC18LF1230/1330		1.3	2.93	mA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V		
		1.3	2.93	mA	$+25^{\circ}\text{C}$			
		1.3	2.93	mA	$+85^{\circ}\text{C}$			
All devices		2.7	5.93	mA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V		
		2.6	5.93	mA	$+25^{\circ}\text{C}$			
		2.5	5.93	mA	$+85^{\circ}\text{C}$			
Extended devices only		2.6	7.0	mA	$+125^{\circ}\text{C}$			
Extended devices only		8.4	27.7	mA	$+125^{\circ}\text{C}$	V <sub>DD</sub> = 4.2V		F <sub>OSC</sub> = 25 MHz ( <b>PRI_RUN</b> , EC oscillator)
		11	27.7	mA	$+125^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V		
All devices		15	26	mA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 4.2V	F <sub>OSC</sub> = 40 MHz ( <b>PRI_RUN</b> , EC oscillator)	
		16	25	mA	$+25^{\circ}\text{C}$			
		16	24	mA	$+85^{\circ}\text{C}$			
All devices		21	39.3	mA	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V		
		21	39.3	mA	$+25^{\circ}\text{C}$			
		21	39.3	mA	$+85^{\circ}\text{C}$			

- Note 1:** 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 V<sub>DD</sub> or V<sub>SS</sub> and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all I<sub>DD</sub> measurements in active operation mode are:  
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>;  
MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F1230/1330

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated)						
		Operating temperature		-40°C ≤ TA ≤ +85°C for industrial				
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated)						
		Operating temperature		-40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	Device	Typ	Max	Units	Conditions			
<b>Supply Current (IDD)<sup>(2)</sup></b>								
	All devices	7.5	20.3	mA	-40°C	VDD = 4.2V	Fosc = 4 MHz, 16 MHz internal (PRI_RUN HS+PLL)	
		7.4	20.3	mA	+25°C			
		7.3	20.3	mA	+85°C			
Extended devices only	8.0	21	mA	+125°C				
All devices	10	20.3	mA	-40°C	VDD = 5.0V			Fosc = 4 MHz, 16 MHz internal (PRI_RUN HS+PLL)
	10	20.3	mA	+25°C				
	9.7	20.3	mA	+85°C				
Extended devices only	10	21	mA	+125°C				
All devices	17	40	mA	-40°C		VDD = 4.2V	Fosc = 10 MHz, 40 MHz internal (PRI_RUN HS+PLL)	
	17	40	mA	+25°C				
	17	40	mA	+85°C				
All devices	23	40	mA	-40°C	VDD = 5.0V			Fosc = 10 MHz, 40 MHz internal (PRI_RUN HS+PLL)
	23	40	mA	+25°C				
	23	40	mA	+85°C				

- Note 1:** 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- Note 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, 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 or VSS;  
MCLR = VDD; WDT enabled/disabled as specified.
- Note 3:** Low-power Timer1 oscillator selected.
- Note 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (I<sub>DD</sub>)<sup>(2)</sup></b>							
	PIC18LF1230/1330	65	112	μA	-40°C	V <sub>DD</sub> = 2.0V	Fosc = 1 MHz (PRI_IDLE mode, EC oscillator)
		65	112	μA	+25°C		
		70	112	μA	+85°C		
	PIC18LF1230/1330	120	237	μA	-40°C	V <sub>DD</sub> = 3.0V	
		120	237	μA	+25°C		
		130	237	μA	+85°C		
	All devices	300	360	μA	-40°C	V <sub>DD</sub> = 5.0V	
		240	360	μA	+25°C		
		300	360	μA	+85°C		
Extended devices only	320	865	μA	+125°C			
PIC18LF1230/1330	260	427	μA	-40°C	V <sub>DD</sub> = 2.0V	Fosc = 4 MHz (PRI_IDLE mode, EC oscillator)	
	255	427	μA	+25°C			
	270	427	μA	+85°C			
PIC18LF1230/1330	420	740	μA	-40°C	V <sub>DD</sub> = 3.0V		
	430	740	μA	+25°C			
	450	740	μA	+85°C			
All devices	0.9	1.23	mA	-40°C	V <sub>DD</sub> = 5.0V		
	0.9	1.23	mA	+25°C			
	0.9	1.23	mA	+85°C			
Extended devices only	1	1.2	mA	+125°C			
Extended devices only	2.8	10.7	mA	+125°C	V <sub>DD</sub> = 4.2V	Fosc = 25 MHz (PRI_IDLE mode, EC oscillator)	
	4.3	10.7	mA	+125°C	V <sub>DD</sub> = 5.0V		
All devices	6.0	9.5	mA	-40°C	V <sub>DD</sub> = 4.2V	Fosc = 40 MHz (PRI_IDLE mode, EC oscillator)	
	6.2	9.0	mA	+25°C			
	6.6	8.6	mA	+85°C			
All devices	8.1	17.3	mA	-40°C	V <sub>DD</sub> = 5.0V		
	9.1	17.3	mA	+25°C			
	8.3	17.3	mA	+85°C			

- Note 1:** 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 V<sub>DD</sub> or V<sub>SS</sub> and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all I<sub>DD</sub> measurements in active operation mode are:  
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>;  
MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F1230/1330

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (IDD)<sup>(2)</sup></b>							
	PIC18LF1230/1330	14	39.6	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 32 kHz <sup>(4)</sup> ( <b>SEC_RUN</b> mode, Timer1 as clock)
		15	39.6	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		16	39.6	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	PIC18LF1230/1330	40	64	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V	
		35	64	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		31	64	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	All devices	99	147	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V	
		81	147	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		75	147	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	PIC18LF1230/1330	2.5	11.6	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 32 kHz <sup>(4)</sup> ( <b>SEC_IDLE</b> mode, Timer1 as clock)
		3.7	11.6	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		4.5	11.6	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	PIC18LF1230/1330	5.0	14.6	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V	
		5.4	14.6	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		6.3	14.6	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	All devices	8.5	24.6	$\mu\text{A}$	$-40^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V	
		9.0	24.6	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		10.5	24.6	$\mu\text{A}$	$+85^{\circ}\text{C}$		

- Note 1:** 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 V<sub>DD</sub> or V<sub>SS</sub> and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, 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 V<sub>DD</sub> or V<sub>SS</sub>;  
MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

PIC18LF1230/1330 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial						
PIC18F1230/1330 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended						
Param No.	Device	Typ	Max	Units	Conditions			
D022 (ΔI <sub>WDT</sub> )	Watchdog Timer	Module Differential Currents (ΔI <sub>WDT</sub> , ΔI <sub>BOR</sub> , ΔI <sub>LVD</sub> , ΔI <sub>OSCB</sub> , ΔI <sub>AD</sub> )						
		1.3	4.8	μA	-40°C	V <sub>DD</sub> = 2.0V		
		1.4	5.4	μA	+25°C			
		2.0	5.4	μA	+85°C			
		V <sub>DD</sub> = 3.0V	1.9	5.6	μA	-40°C	V <sub>DD</sub> = 3.0V	
			2.0	6.2	μA	+25°C		
			2.8	6.2	μA	+85°C		
			4.0	9.6	μA	-40°C		
		V <sub>DD</sub> = 5.0V	5.5	9.6	μA	+25°C	V <sub>DD</sub> = 5.0V	
			5.6	9.6	μA	+85°C		
13	13		μA	+125°C				
D022A (ΔI <sub>BOR</sub> )	Brown-out Reset <sup>(4)</sup>	35	54.6	μA	-40°C to +85°C	V <sub>DD</sub> = 3.0V	Sleep mode, BOREN1:BOREN0 = 10	
		40	64.6	μA	-40°C to +85°C	V <sub>DD</sub> = 5.0V		
		55	44	μA	-40°C to +125°C			
		0	44	μA	-40°C to +85°C			
		0	44	μA	-40°C to +125°C			
D022B (ΔI <sub>LVD</sub> )	Low-Voltage Detect <sup>(4)</sup>	22	37.6	μA	-40°C to +85°C	V <sub>DD</sub> = 2.0V		
		25	39.6	μA	-40°C to +85°C	V <sub>DD</sub> = 3.0V		
		29	44.6	μA	-40°C to +85°C	V <sub>DD</sub> = 5.0V		
		30	54.6	μA	-40°C to +125°C			
D025 (ΔI <sub>OSCB</sub> )	Timer1 Oscillator	2.1	5.5	μA	-40°C	V <sub>DD</sub> = 2.0V	32 kHz on Timer1 <sup>(3)</sup>	
		1.8	6.1	μA	+25°C			
		2.1	6.1	μA	+85°C			
		V <sub>DD</sub> = 3.0V	2.2	7	μA	-40°C	V <sub>DD</sub> = 3.0V	32 kHz on Timer1 <sup>(3)</sup>
			2.6	7.6	μA	+25°C		
			2.9	7.6	μA	+85°C		
		V <sub>DD</sub> = 5.0V	3.0	7.6	μA	-40°C	V <sub>DD</sub> = 5.0V	32 kHz on Timer1 <sup>(3)</sup>
			3.2	7.6	μA	+25°C		
3.4	7.6		μA	+85°C				

**Note 1:** 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 V<sub>DD</sub> or V<sub>SS</sub> and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all I<sub>DD</sub> measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>;

MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.

**3:** Low-power Timer1 oscillator selected.

**4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F1230/1330

## 23.2 DC Characteristics: Power-Down and Supply Current PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

<b>PIC18LF1230/1330</b> (Industrial)		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
<b>PIC18F1230/1330</b> (Industrial, Extended)		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions		
D026 ( $\Delta I_{AD}$ )	<b>Module Differential Currents (<math>\Delta I_{WDT}</math>, <math>\Delta I_{BOR}</math>, <math>\Delta I_{LVD}</math>, <math>\Delta I_{OSCB}</math>, <math>\Delta I_{AD}</math>)</b>						
	<b>A/D Converter</b>	1.0	1.6	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{DD} = 2.0\text{V}$	A/D on, not converting
		1.0	1.6	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{DD} = 3.0\text{V}$	
		1.0	1.6	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{DD} = 5.0\text{V}$	
		2.0	7.6	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$		

- Note 1:** 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  $V_{DD}$  or  $V_{SS}$  and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all  $I_{DD}$  measurements in active operation mode are:  
 $\overline{\text{OSC1}}$  = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to  $V_{DD}$  or  $V_{SS}$ ;  
 $\overline{\text{MCLR}}$  =  $V_{DD}$ ; WDT enabled/disabled as specified.
- 3:** Low-power Timer1 oscillator selected.
- 4:** BOR and LVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

## 23.3 DC Characteristics: PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D030 D030A D031 D031A D031B D032 D033 D033A D033B D034	V <sub>IL</sub>	<b>Input Low Voltage</b> I/O ports: with TTL buffer  with Schmitt Trigger buffer RC3 and RC4  $\overline{\text{MCLR}}$ OSC1 OSC1 OSC1 T1CKI	V <sub>SS</sub> — V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub>	0.15 V <sub>DD</sub> 0.8 0.2 V <sub>DD</sub> 0.3 V <sub>DD</sub> 0.8 0.2 V <sub>DD</sub> 0.3 V <sub>DD</sub> 0.2 V <sub>DD</sub> 0.3 0.3	V V V V V V V V V V	V <sub>DD</sub> < 4.5V 4.5V ≤ V <sub>DD</sub> ≤ 5.5V  I <sup>2</sup> C™ enabled SMBus enabled  HS, HSPLL modes RC, EC modes <sup>(1)</sup> XT, LP modes
D040 D040A D041 D041A D041B D042 D043 D043A D043B D043C D044	V <sub>IH</sub>	<b>Input High Voltage</b> I/O ports: with TTL buffer  with Schmitt Trigger buffer RC3 and RC4  $\overline{\text{MCLR}}$ OSC1 OSC1 OSC1 T1CKI	0.25 V <sub>DD</sub> + 0.8V 2.0 0.8 V <sub>DD</sub> 0.7 V <sub>DD</sub> 2.1 0.8 V <sub>DD</sub> 0.7 V <sub>DD</sub> 0.8 V <sub>DD</sub> 0.9 V <sub>DD</sub> 1.6 1.6	V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub>	V V V V V V V V V V V	V <sub>DD</sub> < 4.5V 4.5V ≤ V <sub>DD</sub> ≤ 5.5V  I <sup>2</sup> C enabled I <sup>2</sup> C enabled  HS, HSPLL modes EC mode RC mode <sup>(1)</sup> XT, LP modes
D060  D061 D063	I <sub>IL</sub>	<b>Input Leakage Current<sup>(2,3)</sup></b> I/O ports  $\overline{\text{MCLR}}$ OSC1	—  — —	±200  ±50 ±1 ±1	nA  nA μA μA	V <sub>SS</sub> < 5.5V V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> Pin at high-impedance  V <sub>SS</sub> < 3V V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> Pin at high-impedance  V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub>
D070	I <sub>PU</sub> I <sub>PURB</sub>	<b>Weak Pull-up Current</b> PORTB weak pull-up current	50	400	μA	V <sub>DD</sub> = 5V, V <sub>PIN</sub> = V <sub>SS</sub>

**Note 1:** In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC® device be driven with an external clock while in RC mode.

**2:** The leakage current on the  $\overline{\text{MCLR}}$  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.

# PIC18F1230/1330

## 23.3 DC Characteristics: PIC18F1230/1330 (Industrial) PIC18LF1230/1330 (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D080	VOL	<b>Output Low Voltage</b> I/O ports	—	0.6	V	$I_{OL} = 8.5\text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D083		OSC2/CLKO (RC, RCIO, EC, ECIO modes)	—	0.6	V	$I_{OL} = 1.6\text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D090	VOH	<b>Output High Voltage<sup>(3)</sup></b> I/O ports	$V_{DD} - 0.7$	—	V	$I_{OH} = -3.0\text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D092		OSC2/CLKO (RC, RCIO, EC, ECIO modes)	$V_{DD} - 0.7$	—	V	$I_{OH} = -1.3\text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D100	COSC2	<b>Capacitive Loading Specs on Output Pins</b> OSC2 pin	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	CIO	All I/O pins and OSC2 (in RC mode)	—	50	pF	To meet the AC Timing Specifications

- Note 1:** In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC<sup>®</sup> device be driven with an external clock while in RC mode.
- 2:** The leakage current on the  $\overline{\text{MCLR}}$  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.



**TABLE 23-1: MEMORY PROGRAMMING REQUIREMENTS**

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
<b>Data EEPROM Memory</b>							
D120	ED	Byte Endurance	100K	1M	—	E/W	-40°C to +85°C
D121	VDRW	VDD for Read/Write	V <sub>MIN</sub>	—	5.5	V	Using EECON to read/write V <sub>MIN</sub> = Minimum operating voltage
D122	TDEW	Erase/Write Cycle Time	3.59	4.10	4.86	ms	
D123	TRETD	Characteristic Retention	40	—	—	Year	Provided no other specifications are violated
D124	TREF	Number of Total Erase/Write Cycles before Refresh <sup>(1)</sup>	1M	10M	—	E/W	-40°C to +85°C
D125	IDDP	Supply Current during Programming	—	10	—	mA	
<b>Program Flash Memory</b>							
D130	EP	Cell Endurance	10K	100K	—	E/W	-40°C to +85°C
D131	VPR	VDD for Read	V <sub>MIN</sub>	—	5.5	V	V <sub>MIN</sub> = Minimum operating voltage
D132B	VPEW	VDD for Self-Timed Write	V <sub>MIN</sub>	—	5.5	V	V <sub>MIN</sub> = Minimum operating voltage
D133A	TIW	Self-Timed Write Cycle Time	1.79	2.05	2.43	ms	
D134	TRETD	Characteristic Retention	40	100	—	Year	Provided no other specifications are violated
D135	IDDP	Supply Current during Programming	—	10	—	mA	

† Data in “Typ” column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Refer to Section 7.8 for a more detailed discussion on data EEPROM endurance.

# PIC18F1230/1330

**TABLE 23-2: COMPARATOR SPECIFICATIONS**

Operating Conditions: $3.0V < V_{DD} < 5.5V$ , $-40^{\circ}C < T_A < +125^{\circ}C$ (unless otherwise stated).							
Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D300	V <sub>IOFF</sub>	Input Offset Voltage	—	±5.0	±10	mV	
D301	V <sub>ICM</sub>	Input Common Mode Voltage	0	—	$V_{DD} - 1.5$	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
D303	T <sub>RESP</sub>	Response Time <sup>(1)</sup>	—	150	400	ns	PIC18FXXXX
D303A			—	150	600	ns	PIC18LFXXXX, $V_{DD} = 2.0V$
D304	T <sub>MC2OV</sub>	Comparator Mode Change to Output Valid	—	—	10	μs	

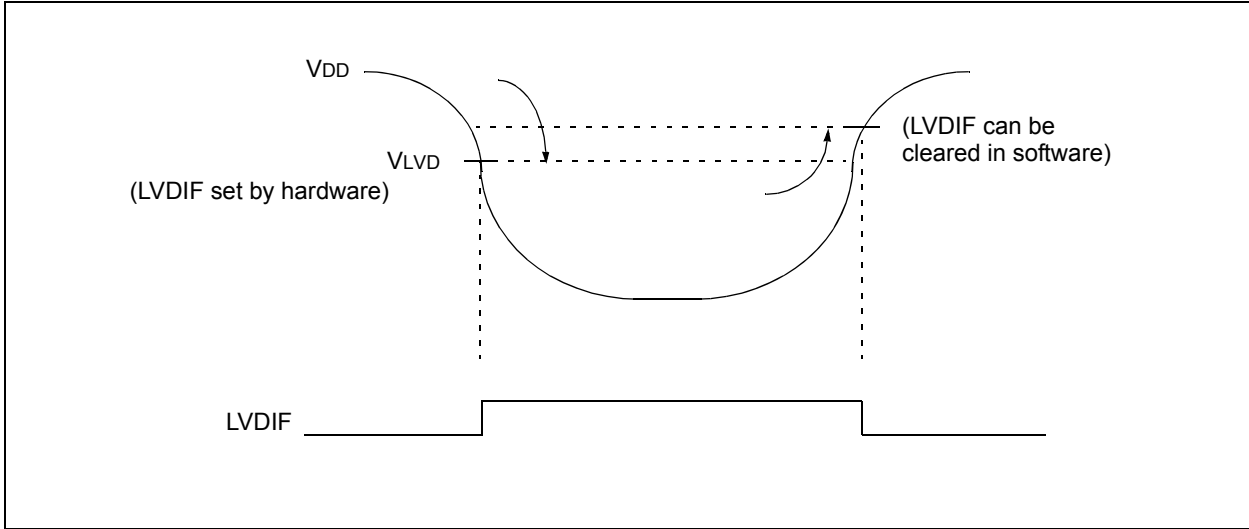
**Note 1:** Response time measured with one comparator input at  $(V_{DD} - 1.5)/2$ , while the other input transitions from V<sub>SS</sub> to V<sub>DD</sub>.

**TABLE 23-3: VOLTAGE REFERENCE SPECIFICATIONS**

Operating Conditions: $3.0V < V_{DD} < 5.5V$ , $-40^{\circ}C < T_A < +125^{\circ}C$ (unless otherwise stated).							
Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D310	V <sub>RES</sub>	Resolution	$V_{DD}/24$	—	$V_{DD}/32$	LSb	
D311	V <sub>RAA</sub>	Absolute Accuracy	—	—	1/2	LSb	
D312	V <sub>RUR</sub>	Unit Resistor Value (R)	—	2k	—	Ω	
D310	T <sub>SET</sub>	Settling Time <sup>(1)</sup>	—	—	10	μs	

**Note 1:** Settling time measured while CVRR = 1 and CVR3:CVR0 transitions from '0000' to '1111'.

**FIGURE 23-4: LOW-VOLTAGE DETECT CHARACTERISTICS**



**TABLE 23-4: LOW-VOLTAGE DETECT CHARACTERISTICS**

Standard Operating Conditions (unless otherwise stated)								
Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial								
$-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended								
Param No.	Sym	Characteristic		Min	Typ	Max	Units	Conditions
D420		LVD Voltage on VDD Transition High-to-Low	LVDL<3:0> = 0000	2.06	2.17	2.28	V	
			LVDL<3:0> = 0001	2.12	2.23	2.34	V	
			LVDL<3:0> = 0010	2.24	2.36	2.48	V	
			LVDL<3:0> = 0011	2.32	2.44	2.56	V	
			LVDL<3:0> = 0100	2.47	2.60	2.73	V	
			LVDL<3:0> = 0101	2.65	2.79	2.93	V	
			LVDL<3:0> = 0110	2.74	2.89	3.04	V	
			LVDL<3:0> = 0111	2.96	3.12	3.28	V	
			LVDL<3:0> = 1000	3.22	3.39	3.56	V	
			LVDL<3:0> = 1001	3.37	3.55	3.73	V	
			LVDL<3:0> = 1010	3.52	3.71	3.90	V	
			LVDL<3:0> = 1011	3.70	3.90	4.10	V	
			LVDL<3:0> = 1100	3.90	4.11	4.32	V	
			LVDL<3:0> = 1101	4.11	4.33	4.55	V	
			LVDL<3:0> = 1110	4.36	4.59	4.82	V	

# PIC18F1230/1330

## 23.4 AC (Timing) Characteristics

### 23.4.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created using one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I<sup>2</sup>C specifications only)
4. Ts (I<sup>2</sup>C specifications only)

<p>T</p> <p>F      Frequency</p>	<p>T      Time</p>
----------------------------------	--------------------

Lowercase letters (pp) and their meanings:

<p>pp</p> <p>cc      CCP1</p> <p>ck      CLKO</p> <p>cs      <math>\overline{CS}</math></p> <p>di      SDI</p> <p>do      SDO</p> <p>dt      Data in</p> <p>io      I/O port</p> <p>mc      <math>\overline{MCLR}</math></p>	<p>osc      OSC1</p> <p>rd      <math>\overline{RD}</math></p> <p>rw      <math>\overline{RD}</math> or <math>\overline{WR}</math></p> <p>sc      SCK</p> <p>ss      <math>\overline{SS}</math></p> <p>t0      T0CKI</p> <p>t1      T13CKI</p> <p>wr      <math>\overline{WR}</math></p>
--	--

Uppercase letters and their meanings:

<p>S</p> <p>F      Fall</p> <p>H      High</p> <p>I      Invalid (High-impedance)</p> <p>L      Low</p> <p>I<sup>2</sup>C only</p> <p>AA      output access</p> <p>BUF      Bus free</p>	<p>P      Period</p> <p>R      Rise</p> <p>V      Valid</p> <p>Z      High-impedance</p> <p>High    High</p> <p>Low     Low</p>
--	--

TCC:ST (I<sup>2</sup>C specifications only)

<p>CC</p> <p>HD      Hold</p> <p>ST</p> <p>DAT      DATA input hold</p> <p>STA      Start condition</p>	<p>SU      Setup</p> <p>STO      Stop condition</p>
---	--

## 23.4.2 TIMING CONDITIONS

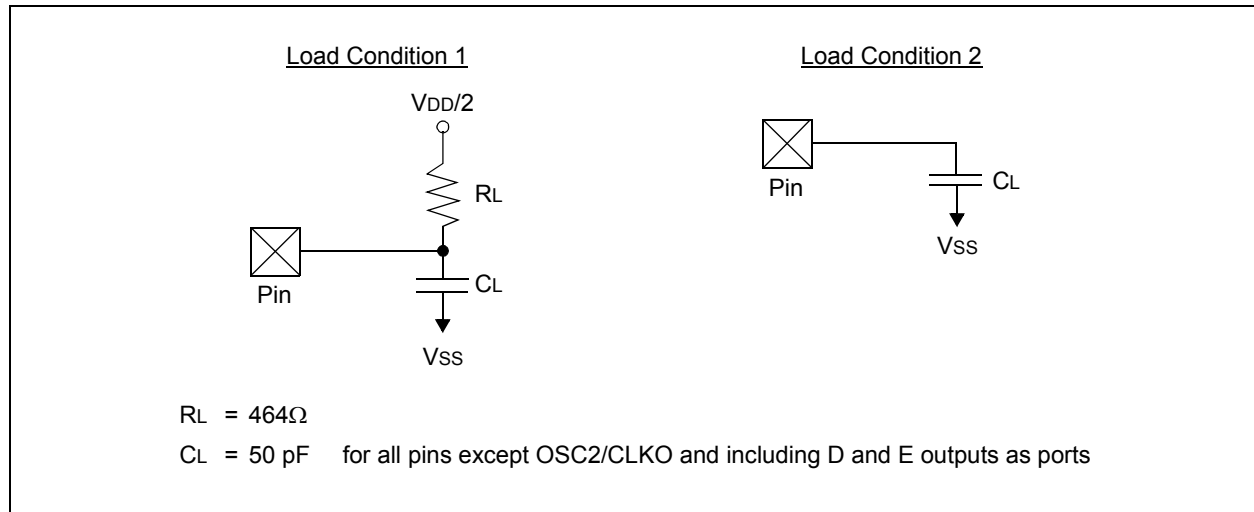
The temperature and voltages specified in Table 23-5 apply to all timing specifications unless otherwise noted. Figure 23-5 specifies the load conditions for the timing specifications.

**Note:** Because of space limitations, the generic terms “PIC18FXXXX” and “PIC18LFXXXX” are used throughout this section to refer to the PIC18F1230/1330 and PIC18LF1230/1330 families of devices specifically and only those devices.

**TABLE 23-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

<b>AC CHARACTERISTICS</b>	<b>Standard Operating Conditions (unless otherwise stated)</b>	
	Operating temperature	-40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended
	Operating voltage VDD range	as described in DC spec <b>Section 23.1</b> and <b>Section 23.3</b> .
		LF parts operate for industrial temperatures only.

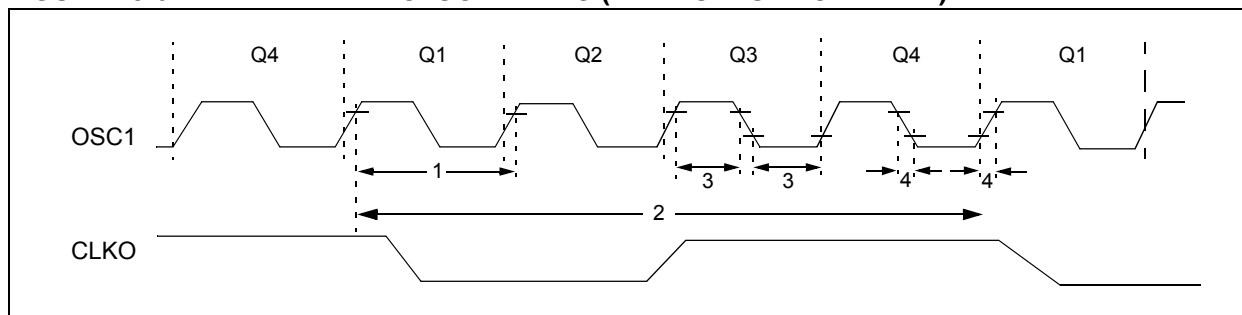
**FIGURE 23-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**



# PIC18F1230/1330

## 23.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 23-6: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)**



**TABLE 23-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	Fosc	External CLKI Frequency <sup>(1)</sup>  Oscillator Frequency <sup>(1)</sup>	DC	1	MHz	XT, RC Oscillator modes
			DC	40	MHz	EC Oscillator mode
			DC	31.25	kHz	LP Oscillator mode
			DC	4	MHz	RC Oscillator mode
			0.1	4	MHz	XT Oscillator mode
			4	20	MHz	HS Oscillator mode
			5	200	kHz	LP Oscillator mode
1	Tosc	External CLKI Period <sup>(1)</sup>  Oscillator Period <sup>(1)</sup>	1000	—	ns	XT, RC Oscillator modes
			50	—	ns	HS Oscillator mode
			25	—	ns	EC Oscillator mode
			250	—	ns	RC Oscillator mode
			250	1	μs	XT Oscillator mode
			50	250	ns	HS Oscillator mode
			100	250	ns	HS +PLL Oscillator mode
5	200	μs	LP Oscillator mode			
2	Tcy	Instruction Cycle Time <sup>(1)</sup>	100	—	ns	Tcy = 4/Fosc, Industrial
			160	—	ns	Tcy = 4/Fosc, Extended
3	TosL, TosH	External Clock in (OSC1) High or Low Time	30	—	ns	XT Oscillator mode
			2.5	—	μs	LP Oscillator mode
			10	—	ns	HS Oscillator mode
4	TosR, TosF	External Clock in (OSC1) Rise or Fall Time	—	20	ns	XT Oscillator mode
			—	50	ns	LP Oscillator mode
			—	7.5	ns	HS Oscillator mode

**Note 1:** Instruction cycle period (Tcy) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

**TABLE 23-7: PLL CLOCK TIMING SPECIFICATIONS (V<sub>DD</sub> = 4.2V TO 5.5V)**

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
F10	FOSC	Oscillator Frequency Range	4	—	10	MHz	HS mode only
F11	FSYS	On-Chip VCO System Frequency	16	—	40	MHz	HS mode only
F12	t <sub>rc</sub>	PLL Start-up Time (Lock Time)	—	—	2	ms	
F13	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.

**TABLE 23-8: AC CHARACTERISTICS: INTERNAL RC ACCURACY**

Standard Operating Conditions (unless otherwise stated)							
Operating temperature -40°C ≤ TA ≤ +85°C for industrial							
-40°C ≤ TA ≤ +125°C for extended							
Param No.	Device	Min	Typ	Max	Units	Conditions	
<b>INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz, 31 kHz<sup>(1)</sup></b>							
	PIC18LF1230/1330	-2	+/-1	2	%	+25°C	V <sub>DD</sub> = 2.7-3.3V
		-5	—	5	%	-10°C to +85°C	V <sub>DD</sub> = 2.7-3.3V
		-10	+/-1	10	%	-40°C to +85°C	V <sub>DD</sub> = 2.7-3.3V
	PIC18F1230/1330	-2	+/-1	2	%	+25°C	V <sub>DD</sub> = 4.5-5.5V
		-5	—	5	%	-10°C to +85°C	V <sub>DD</sub> = 4.5-5.5V
		-10	+/-1	10	%	-40°C to +85°C	V <sub>DD</sub> = 4.5-5.5V
<b>INTRC Accuracy @ Freq = 31 kHz<sup>(2,3)</sup></b>							
	PIC18LF1230/1330	26.562	—	35.938	kHz	-40°C to +85°C	V <sub>DD</sub> = 2.7-3.3V
	PIC18F1230/1330	26.562	—	35.938	kHz	-40°C to +85°C	V <sub>DD</sub> = 4.5-5.5V

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** Frequency calibrated at 25°C. OSCTUNE register can be used to compensate for temperature drift.

**2:** INTRC frequency after calibration.

**3:** Change of INTRC frequency as V<sub>DD</sub> changes.

# PIC18F1230/1330

**FIGURE 23-7: CLKO AND I/O TIMING**



**TABLE 23-9: CLKO AND I/O TIMING REQUIREMENTS**

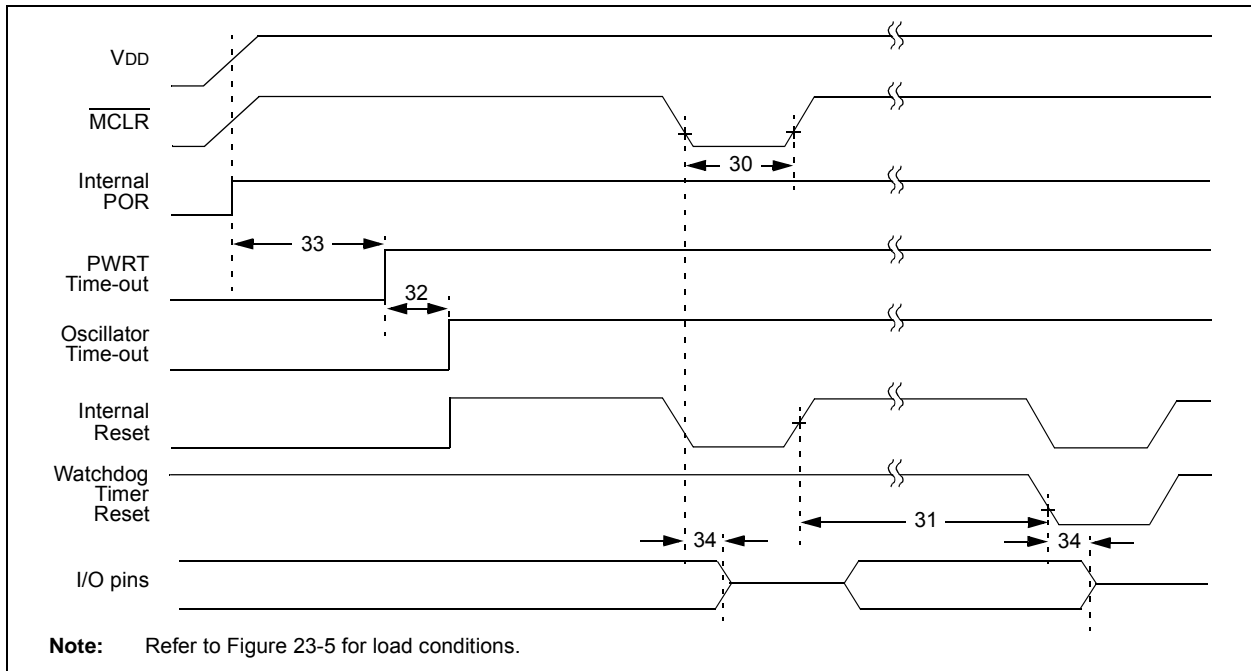
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions	
10	TosH2ckL	OSC1 ↑ to CLKO ↓	—	75	200	ns	(Note 1)	
11	TosH2ckH	OSC1 ↑ to CLKO ↑	—	75	200	ns	(Note 1)	
12	TckR	CLKO Rise Time	—	35	100	ns	(Note 1)	
13	TckF	CLKO Fall Time	—	35	100	ns	(Note 1)	
14	TckL2ioV	CLKO ↓ to Port Out Valid	—	—	0.5 T <sub>CY</sub> + 20	ns	(Note 1)	
15	TioV2ckH	Port In Valid before CLKO ↑	0.25 T <sub>CY</sub> + 25	—	—	ns	(Note 1)	
16	TckH2ioI	Port In Hold after CLKO ↑	0	—	—	ns	(Note 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)	PIC18FXXXX	100	—	—	ns	
18A			PIC18LFXXXX	200	—	—	ns	V <sub>DD</sub> = 2.0V
19	TioV2osH	Port Input Valid to OSC1 ↑ (I/O in setup time)	0	—	—	ns		
20	TioR	Port Output Rise Time	PIC18FXXXX	—	10	25	ns	
20A			PIC18LFXXXX	—	—	60	ns	V <sub>DD</sub> = 2.0V
21	TioF	Port Output Fall Time	PIC18FXXXX	—	10	25	ns	
21A			PIC18LFXXXX	—	—	60	ns	V <sub>DD</sub> = 2.0V
22†	TINP	INTx Pin High or Low Time	T <sub>CY</sub>	—	—	ns		
23†	TRBP	RB7:RB4 Change INTx High or Low Time	T <sub>CY</sub>	—	—	ns		

† These parameters are asynchronous events not related to any internal clock edges.

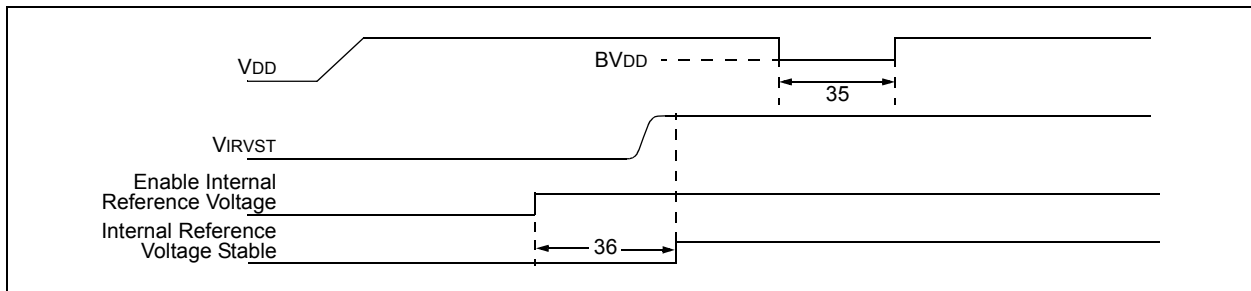
**Note 1:** Measurements are taken in RC mode, where CLKO output is 4 x Tosc.



**FIGURE 23-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**



**FIGURE 23-9: BROWN-OUT RESET TIMING**

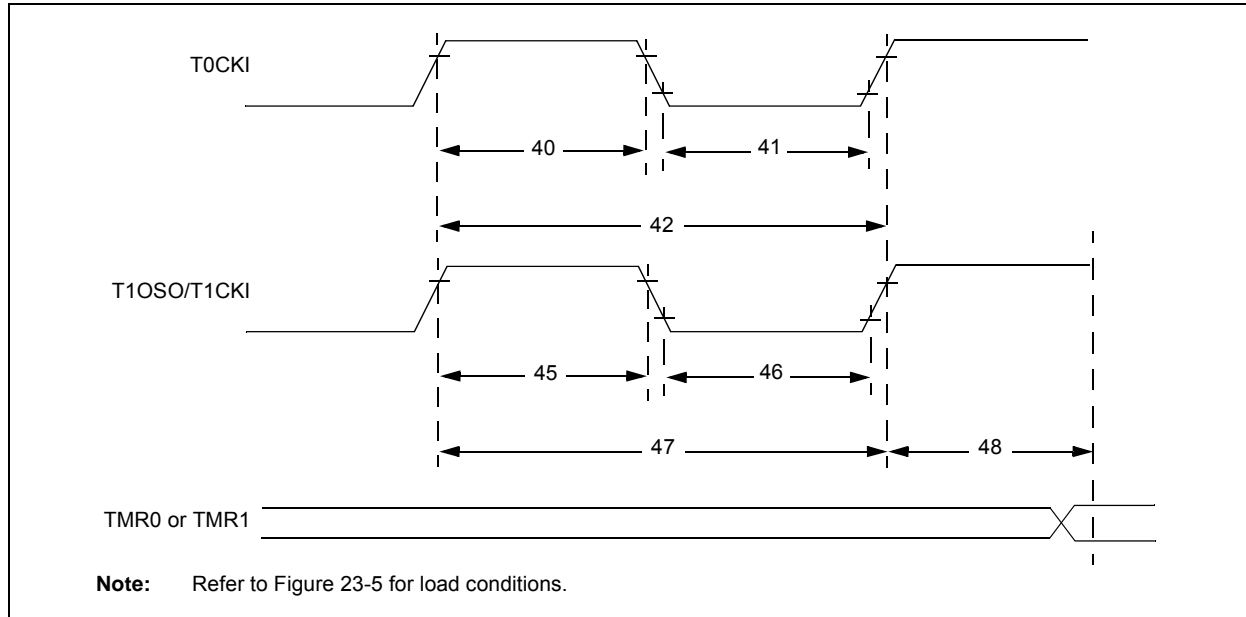


**TABLE 23-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TmCL	MCLR Pulse Width (low)	2	—	—	μs	
31	TWDT	Watchdog Timer Time-out Period (no postscaler)	3.4	4.0	4.6	ms	
32	TOST	Oscillation Start-up Timer Period	1024 T <sub>osc</sub>	—	1024 T <sub>osc</sub>	—	T <sub>osc</sub> = OSC1 period
33	TPWRT	Power-up Timer Period	55.6	65.5	75	ms	
34	TIOZ	I/O High-Impedance from MCLR Low or Watchdog Timer Reset	—	2	—	μs	
35	TBOR	Brown-out Reset Pulse Width	200	—	—	μs	VDD ≤ BVDD (see D005)
36	TI <sub>R</sub> VST	Time for Internal Reference Voltage to become Stable	—	20	50	μs	
37	TLVD	Low-Voltage Detect Pulse Width	200	—	—	μs	VDD ≤ VLVD
38	TCSD	CPU Start-up Time	—	10	—	μs	
39	TI <sub>O</sub> BST	Time for INTOSC to Stabilize	—	1	—	μs	

# PIC18F1230/1330

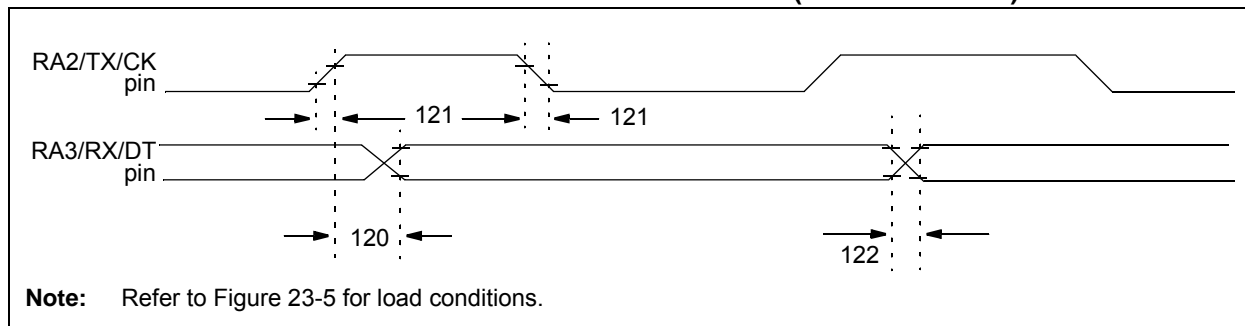
**FIGURE 23-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**



**TABLE 23-11: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions	
40	Tt0H	T0CKI High Pulse Width	No prescaler	$0.5 T_{CY} + 20$	—	ns		
			With prescaler	10	—	ns		
41	Tt0L	T0CKI Low Pulse Width	No prescaler	$0.5 T_{CY} + 20$	—	ns		
			With prescaler	10	—	ns		
42	Tt0P	T0CKI Period	No prescaler	$T_{CY} + 10$	—	ns		
			With prescaler	Greater of: 20 ns or $(T_{CY} + 40)/N$	—	ns		N = prescale value (1, 2, 4, ..., 256)
45	Tt1H	T1CKI High Time	Synchronous, no prescaler	$0.5 T_{CY} + 20$	—	ns		
			Synchronous, with prescaler	PIC18FXXXX	10	—		ns
				PIC18LFXXXX	25	—		ns
			Asynchronous	PIC18FXXXX	30	—		ns
PIC18LFXXXX	50	—		ns	$V_{DD} = 2.0V$			
46	Tt1L	T1CKI Low Time	Synchronous, no prescaler	$0.5 T_{CY} + 5$	—	ns		
			Synchronous, with prescaler	PIC18FXXXX	10	—		ns
				PIC18LFXXXX	25	—		ns
			Asynchronous	PIC18FXXXX	30	—		ns
PIC18LFXXXX	50	—		ns	$V_{DD} = 2.0V$			
47	Tt1P	T1CKI Input Period	Synchronous	Greater of: 20 ns or $(T_{CY} + 40)/N$	—	ns	N = prescale value (1, 2, 4, 8)	
			Asynchronous	60	—	ns		
	Ft1	T1CKI Oscillator Input Frequency Range		DC	50	kHz		
48	Tcke2tmr1	Delay from External T1CKI Clock Edge to Timer Increment		$2 T_{OSC}$	$7 T_{OSC}$	—		

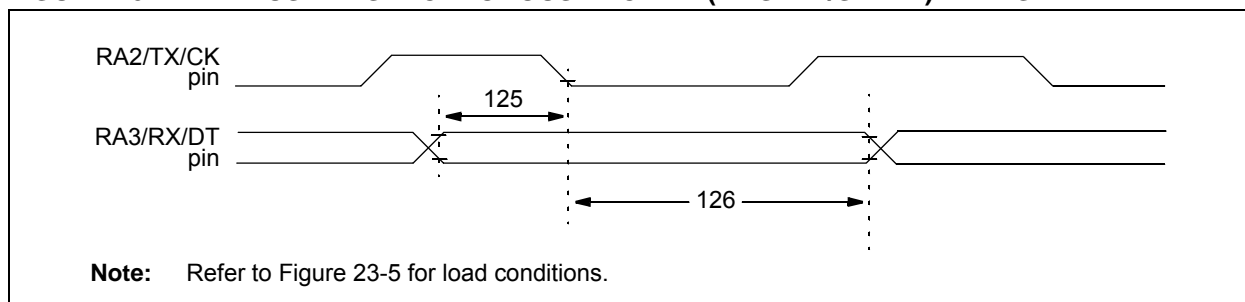
**FIGURE 23-11: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**



**TABLE 23-12: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions	
120	TckH2dtV	<u>SYNC XMIT (MASTER &amp; SLAVE)</u> Clock High to Data Out Valid	PIC18FXXXX	—	40	ns	
			PIC18LFXXXX	—	100	ns	VDD = 2.0V
121	Tckrf	Clock Out Rise Time and Fall Time (Master mode)	PIC18FXXXX	—	20	ns	
			PIC18LFXXXX	—	50	ns	VDD = 2.0V
122	Tdtrf	Data Out Rise Time and Fall Time	PIC18FXXXX	—	20	ns	
			PIC18LFXXXX	—	50	ns	VDD = 2.0V

**FIGURE 23-12: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



**TABLE 23-13: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TdtV2ckl	<u>SYNC RCV (MASTER &amp; SLAVE)</u> Data Hold before CK ↓ (DT hold time)	10	—	ns	
126	TckL2dtl	Data Hold after CK ↓ (DT hold time)	15	—	ns	

# PIC18F1230/1330

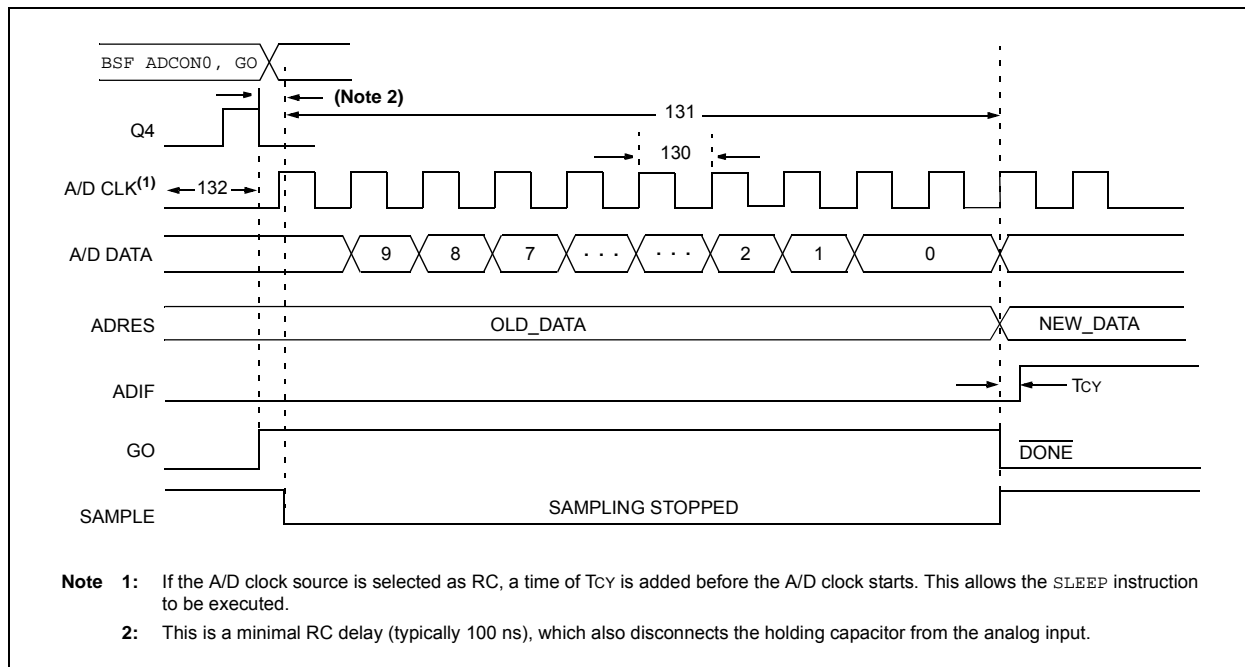
**TABLE 23-14: A/D CONVERTER CHARACTERISTICS**

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	—	—	10	bit	$\Delta V_{REF} \geq 3.0V$
A03	EIL	Integral Linearity Error	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$
A04	EDL	Differential Linearity Error	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$
A06	E <sub>OFF</sub>	Offset Error	—	—	$< \pm 2$	LSb	$\Delta V_{REF} \geq 3.0V$
A07	E <sub>GN</sub>	Gain Error	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$
A10	—	Monotonicity	Guaranteed <sup>(1)</sup>			—	$V_{SS} \leq V_{AIN} \leq V_{REF}$
A20	$\Delta V_{REF}$	Reference Voltage Range ( $V_{REF+} - V_{SS}$ )	1.8	—	—	V	$V_{DD} < 3.0V$
			3	—	—	V	$V_{DD} \geq 3.0V$
A21	$V_{REF+}$	Positive Reference Voltage	$V_{SS}$	—	$V_{REF+}$	V	
A22	$V_{REF-}$	Negative Reference Voltage	$V_{SS} - 0.3V$	—	$V_{DD} - 3.0V$	—	
A25	$V_{AIN}$	Analog Input Voltage	$V_{REF-}$	—	$V_{REF+}$	V	
A30	$Z_{AIN}$	Recommended Impedance of Analog Voltage Source	—	—	2.5	k $\Omega$	
A50	I <sub>REF</sub>	$V_{REF+}$ Input Current <sup>(2)</sup>	—	—	5	$\mu A$	During $V_{AIN}$ acquisition. During A/D conversion cycle.
			—	—	150	$\mu A$	

**Note 1:** The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

**Note 2:**  $V_{REF+}$  current is from RA4/T0CKI/AN2/ $V_{REF+}$  pin or  $V_{DD}$ , whichever is selected as the  $V_{REF+}$  source.

**FIGURE 23-13: A/D CONVERSION TIMING**



**TABLE 23-15: A/D CONVERSION REQUIREMENTS**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions	
130	TAD	A/D Clock Period	PIC18FXXXX	0.7	25.0 <sup>(1)</sup>	μs	TOSC based, VREF ≥ 3.0V
			PIC18LFXXXX	1.4	25.0 <sup>(1)</sup>	μs	VDD = 2.0V, TOSC based, VREF full range
		PIC18FXXXX	—	1	μs	A/D RC mode	
		PIC18LFXXXX	—	3	μs	VDD = 2.0V, A/D RC mode	
131	T <sub>cnv</sub>	Conversion Time (not including acquisition time) <sup>(2)</sup>	11	12	TAD		
132	T <sub>acq</sub>	Acquisition Time <sup>(3)</sup>	1.4	—	μs	-40°C to +85°C	
135	T <sub>swc</sub>	Switching Time from Convert → Sample	—	(Note 4)			
136	T <sub>dis</sub>	Discharge Time	0.2	—	μs		

**Note 1:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

**Note 2:** ADRES register may be read on the following T<sub>cy</sub> cycle.

**Note 3:** The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (R<sub>s</sub>) on the input channels is 50Ω.

**Note 4:** On the following cycle of the device clock.

# PIC18F1230/1330

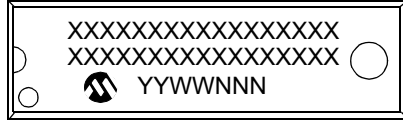
---

NOTES:

## 24.0 PACKAGING INFORMATION

### 24.1 Package Marking Information

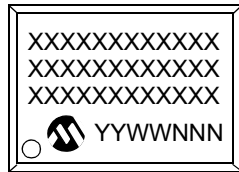
18-Lead PDIP



Example



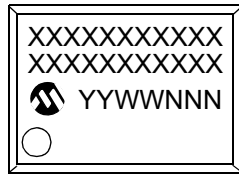
18-Lead SOIC



Example



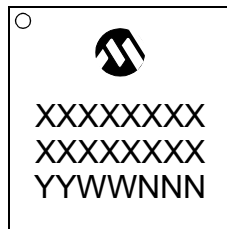
20-Lead SSOP



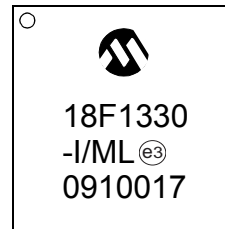
Example



28-Lead QFN



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.

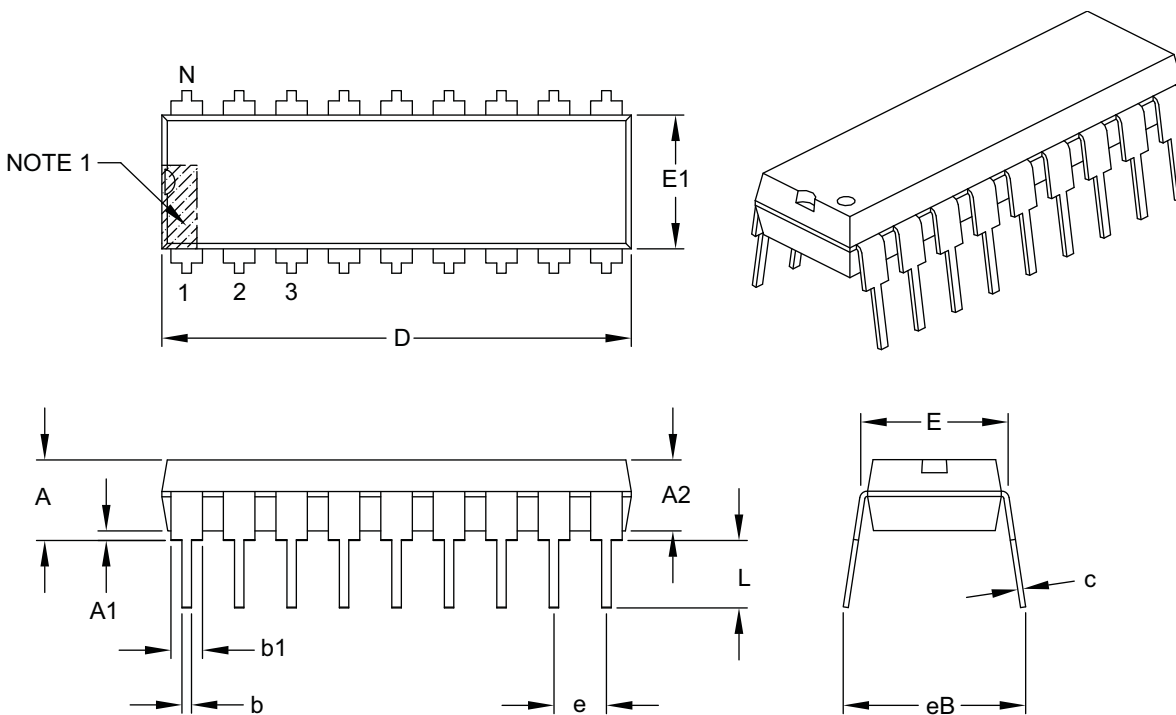
# PIC18F1230/1330

## 24.2 Package Details

The following sections give the technical details of the packages.

### 18-Lead Plastic Dual In-Line (P) – 300 mil Body [PDIP]

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



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	18		
Pitch	e	.100 BSC		
Top to Seating Plane	A	–	–	.210
Molded Package Thickness	A2	.115	.130	.195
Base to Seating Plane	A1	.015	–	–
Shoulder to Shoulder Width	E	.300	.310	.325
Molded Package Width	E1	.240	.250	.280
Overall Length	D	.880	.900	.920
Tip to Seating Plane	L	.115	.130	.150
Lead Thickness	c	.008	.010	.014
Upper Lead Width	b1	.045	.060	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	–	–	.430

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

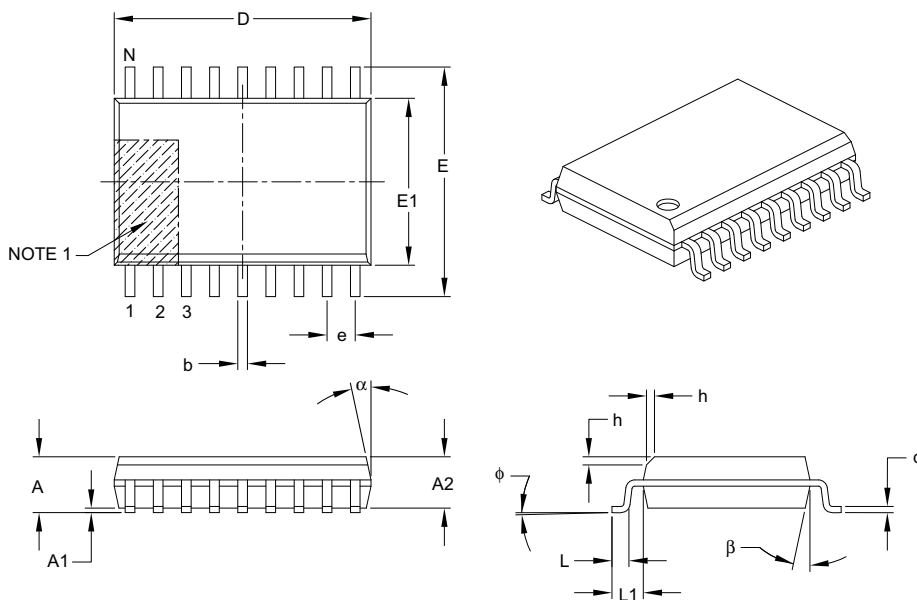
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-007B



## 18-Lead Plastic Small Outline (SO) – Wide, 7.50 mm Body [SOIC]

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



Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Pins	N	18		
Pitch	e	1.27 BSC		
Overall Height	A	–	–	2.65
Molded Package Thickness	A2	2.05	–	–
Standoff §	A1	0.10	–	0.30
Overall Width	E	10.30 BSC		
Molded Package Width	E1	7.50 BSC		
Overall Length	D	11.55 BSC		
Chamfer (optional)	h	0.25	–	0.75
Foot Length	L	0.40	–	1.27
Footprint	L1	1.40 REF		
Foot Angle	φ	0°	–	8°
Lead Thickness	c	0.20	–	0.33
Lead Width	b	0.31	–	0.51
Mold Draft Angle Top	α	5°	–	15°
Mold Draft Angle Bottom	β	5°	–	15°

**Notes:**

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- § Significant Characteristic.
- Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15 mm per side.
- Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

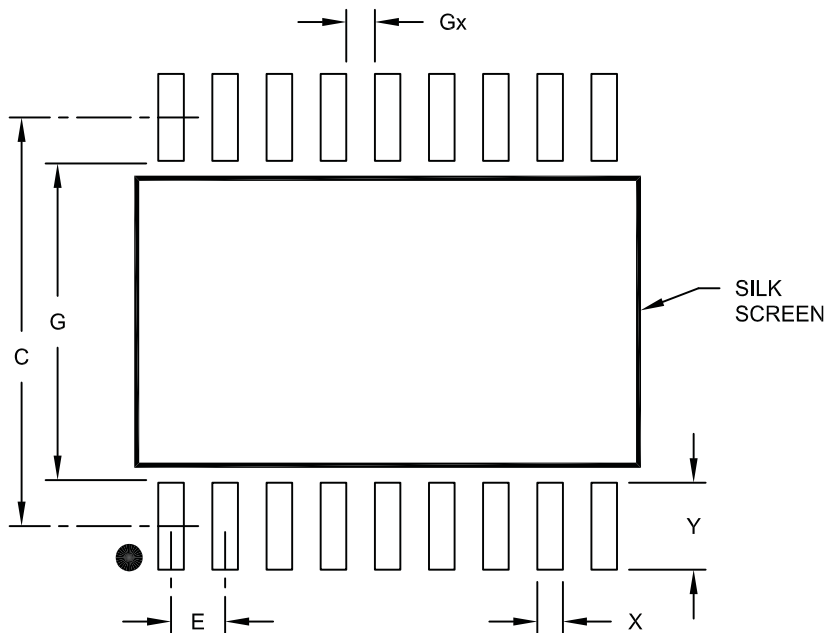
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-051B

# PIC18F1230/1330

18-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

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



RECOMMENDED LAND PATTERN

Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	1.27 BSC		
Contact Pad Spacing	C		9.40	
Contact Pad Width	X			0.60
Contact Pad Length	Y			2.00
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.40		

Notes:

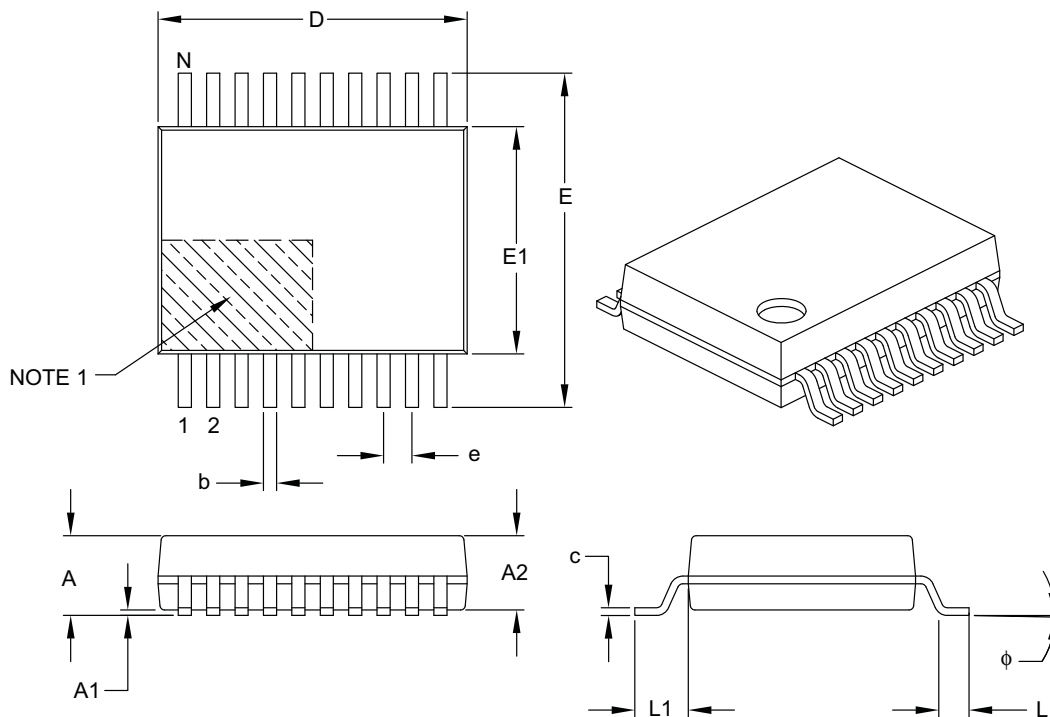
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A

## 20-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

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



Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Pins	N	20		
Pitch	e	0.65 BSC		
Overall Height	A	–	–	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	–	–
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	6.90	7.20	7.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1	1.25 REF		
Lead Thickness	c	0.09	–	0.25
Foot Angle	$\phi$	0°	4°	8°
Lead Width	b	0.22	–	0.38

**Notes:**

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
- Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

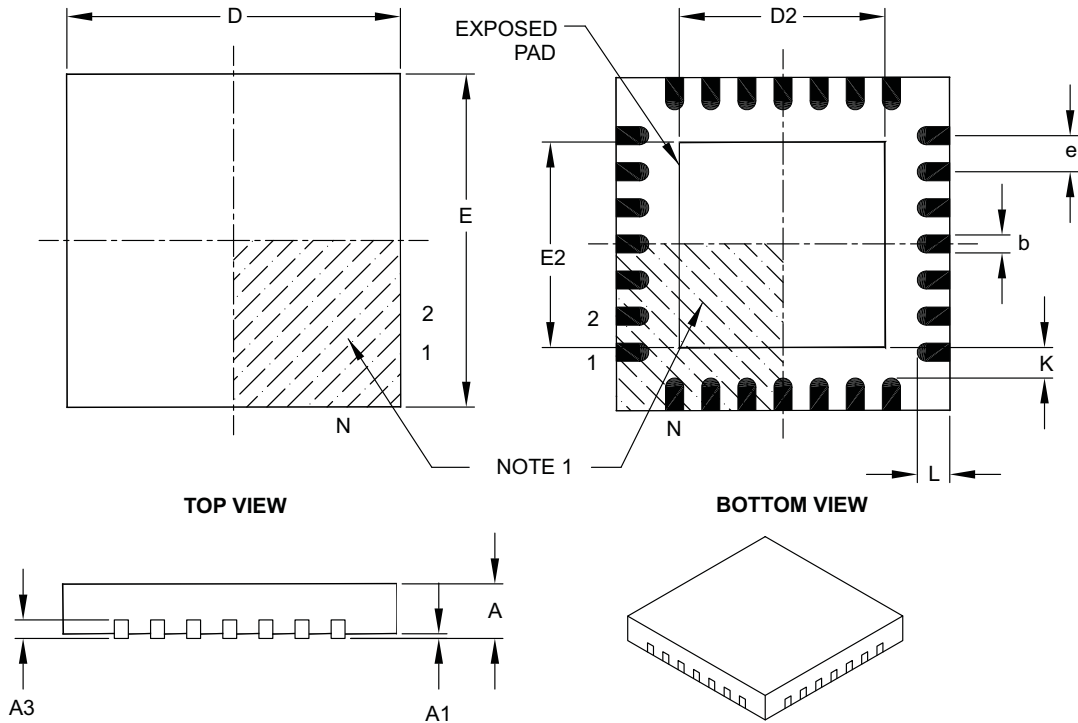
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072B

# PIC18F1230/1330

## 28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

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



Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	0.65 BSC		
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Contact Thickness	A3	0.20 REF		
Overall Width	E	6.00 BSC		
Exposed Pad Width	E2	3.65	3.70	4.20
Overall Length	D	6.00 BSC		
Exposed Pad Length	D2	3.65	3.70	4.20
Contact Width	b	0.23	0.30	0.35
Contact Length	L	0.50	0.55	0.70
Contact-to-Exposed Pad	K	0.20	-	-

**Notes:**

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Package is saw singulated.
- Dimensioning and tolerancing per ASME Y14.5M.

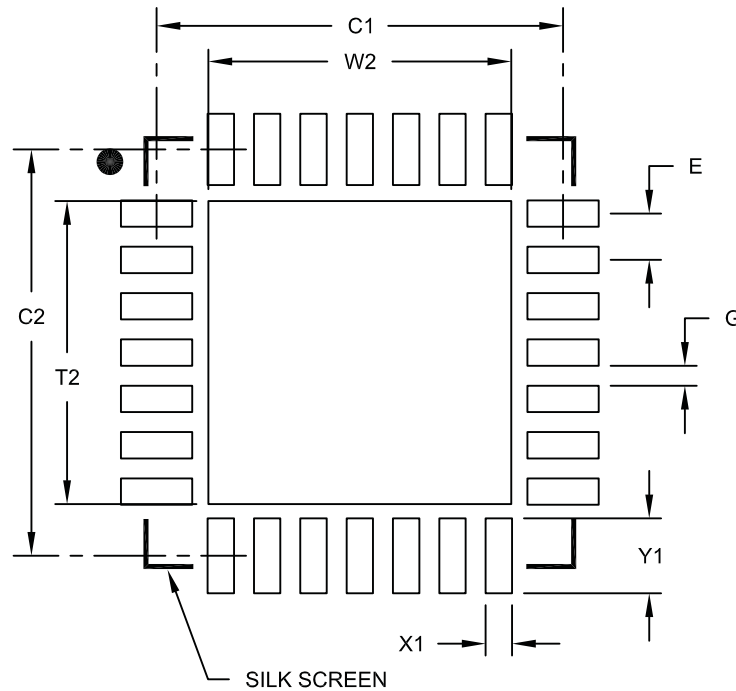
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-105B

## 28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

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



RECOMMENDED LAND PATTERN

Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Optional Center Pad Width	W2			4.25
Optional Center Pad Length	T2			4.25
Contact Pad Spacing	C1		5.70	
Contact Pad Spacing	C2		5.70	
Contact Pad Width (X28)	X1			0.37
Contact Pad Length (X28)	Y1			1.00
Distance Between Pads	G	0.20		

**Notes:**

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A

# PIC18F1230/1330

---

NOTES:

## APPENDIX A: REVISION HISTORY

### Revision A (November 2005)

Original data sheet for PIC18F1230/1330 devices.

### Revision B (February 2006)

Data bank information was updated and a note was added for calculating the PCPWM duty cycle.

### Revision C (March 2007)

Updated **Section 23.0 “Electrical Characteristics”** and **Section 24.0 “Packaging Information”**.

### Revision D (November 2009)

Updated LIN 1.2 to LIN/J2602 throughout document along with minor corrections throughout document. Added the PIC18LF1230 and PIC18LF1330 devices. Refer to Table A-1 for additional revision history.

**TABLE A-1: SECTION REVISION HISTORY**

Section Name	Update Description
<b>Section 1.0 “Device Overview”</b>	Updated Table 1-2
<b>Section 6.0 “Memory Organization”</b>	Updated Table 6-2
<b>Section 7.0 “Flash Program Memory”</b>	Updated <b>Section 7.2.4 “Table Pointer Boundaries”</b> , Figure 7-3
<b>Section 8.0 “Data EEPROM Memory”</b>	Updated <b>Section 8.2 “EECON1 and EECON2 Registers”</b> , <b>Section 8.8 “Using the Data EEPROM”</b>
<b>Section 10.0 “I/O Ports”</b>	Updated <b>Section 10.2 “PORTB, TRISB and LATB Registers”</b>
<b>Section 14.0 “Power Control PWM Module”</b>	Updated Register 14-6, <b>Section 14.11.2 “Output Polarity Control”</b>
<b>Section 15.0 “Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART)”</b>	Updated Register 15-3, <b>Section 15.1 “Baud Rate Generator (BRG)”</b> , Table 15-2, <b>Section 15.1.3 “Auto-Baud Rate Detect”</b> , <b>Section 15.2 “EUSART Asynchronous Mode”</b> , Table 15-5, Table 15-6, <b>Section 15.3 “EUSART Synchronous Master Mode”</b> , Figure 15-11, Table 15-7, Figure 15-13, Table 15-8, Table 15-9, Table 15-10
<b>Section 16.0 “10-Bit Analog-to-Digital Converter (A/D) Module”</b>	Updated Register 16-2
<b>Section 17.0 “Comparator Module”</b>	Updated Figure 17-2
<b>Section 18.0 “Comparator Voltage Reference Module”</b>	Updated <b>Section 18.1 “Configuring the Comparator Voltage Reference”</b> , Register 18-1, Figure 18-1
<b>Section 20.0 “Special Features of the CPU”</b>	Updated Register 20-6, Register 20-13, Register 20-14
<b>Section 22.0 “Instruction Set Summary”</b>	Updated Table 22-2
<b>Section 23.0 “Electrical Characteristics”</b>	Updated Table 23-1, Figure 23-3, Table 23-2, Table 23-3, Table 23-4, Table 23-5, Table 23-6, Table 23-8, Table 23-14, Table 23-15

# PIC18F1230/1330

---

## 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	PIC18F1230	PIC18F1330
Program Memory (Bytes)	4096	8192
Program Memory (Instructions)	2048	4096
Packages	18-Pin PDIP 18-Pin SOIC 20-Pin SSOP 28-Pin QFN	18-Pin PDIP 18-Pin SOIC 20-Pin SSOP 28-Pin QFN



## APPENDIX C: CONVERSION CONSIDERATIONS

This appendix discusses the considerations for converting from previous versions of a device to the ones listed in this data sheet. Typically, these changes are due to the differences in the process technology used. An example of this type of conversion is from a PIC16C74A to a PIC16C74B.

**Not Applicable**

## APPENDIX D: MIGRATION FROM BASELINE TO ENHANCED DEVICES

This section discusses how to migrate from a Baseline device (i.e., PIC16C5X) to an Enhanced MCU device (i.e., PIC18FXXX).

The following are the list of modifications over the PIC16C5X microcontroller family:

**Not Currently Available**

# PIC18F1230/1330

---

## **APPENDIX E: MIGRATION FROM MID-RANGE TO ENHANCED DEVICES**

A detailed discussion of the differences between the mid-range MCU devices (i.e., PIC16CXXX) and the Enhanced devices (i.e., PIC18FXXX) is provided in AN716, “*Migrating Designs from PIC16C74A/74B to PIC18C442*”. The changes discussed, while device specific, are generally applicable to all mid-range to Enhanced device migrations.

This Application Note is available as Literature Number DS00716.

## **APPENDIX F: MIGRATION FROM HIGH-END TO ENHANCED DEVICES**

A detailed discussion of the migration pathway and differences between the high-end MCU devices (i.e., PIC17CXXX) and the Enhanced devices (i.e., PIC18FXXX) is provided in AN726, “*PIC17CXXX to PIC18CXXX Migration*”.

This Application Note is available as Literature Number DS00726.

## INDEX

### A

A/D .....	169
A/D Converter Interrupt, Configuring .....	173
Acquisition Requirements .....	174
ADCON0 Register .....	169
ADCON1 Register .....	169
ADCON2 Register .....	169
ADRESH Register .....	169, 172
ADRESL Register .....	169
Analog Port Pins, Configuring .....	176
Associated Registers .....	178
Configuring the Module .....	173
Conversion Clock (TAD) .....	175
Conversion Requirements .....	293
Conversion Status (GO/DONE Bit) .....	172
Conversions .....	177
Converter Characteristics .....	292
Discharge .....	177
Operation in Power-Managed Modes .....	176
Selecting and Configuring Acquisition Time .....	175
Triggering Conversions .....	174
Absolute Maximum Ratings .....	265
AC (Timing) Characteristics .....	284
Conditions .....	285
Load Conditions for Device Timing Specifications .....	285
Parameter Symbology .....	284
Temperature and Voltage Specifications .....	285
AC Characteristics .....	
Internal RC Accuracy .....	287
Access Bank .....	
Mapping with Indexed Literal Offset Addressing Mode ..	69
Remapping with Indexed Literal Offset Addressing Mode ..	69
ADCON0 Register .....	169
GO/DONE Bit .....	172
ADCON1 Register .....	169
ADCON2 Register .....	169
ADDFSR .....	258
ADDLW .....	221
ADDULNK .....	258
ADDWF .....	221
ADDWFC .....	222
ADRESH Register .....	169
ADRESL Register .....	169, 172
Analog-to-Digital Converter. <i>See</i> A/D.	
ANDLW .....	222
ANDWF .....	223
Assembler .....	
MPASM Assembler .....	212

### B

BC .....	223
BCF .....	224
Block Diagrams .....	
A/D .....	172
Analog Input Model .....	173
Comparator Analog Input Model .....	181
Comparator Voltage Reference .....	185
Dead-Time Control Unit for One PWM Output Pair ..	135
Device Clock .....	26
EUSART Receive .....	160
EUSART Transmit .....	158

External Power-on Reset Circuit (Slow VDD Power-up)  
41

Fail-Safe Clock Monitor .....	205
Generic I/O Port .....	87
Interrupt Logic .....	94
Low-Voltage Detect .....	188
On-Chip Reset Circuit .....	39
PIC18F1230/1330 .....	12
PLL (HS Mode) .....	23
Power Control PWM .....	118
PWM (One Output Pair, Complementary Mode) .....	119
PWM (One Output Pair, Independent Mode) .....	119
PWM I/O Pin .....	142
PWM Time Base .....	121
Reads From Flash Program Memory .....	75
Single Comparator .....	180
Table Read Operation .....	71
Table Write Operation .....	72
Table Writes to Flash Program Memory .....	77
Timer0 in 16-Bit Mode .....	108
Timer0 in 8-Bit Mode .....	108
Timer1 .....	112
Timer1 (16-Bit Read/Write Mode) .....	112
Watchdog Timer .....	202
BN .....	224
BNC .....	225
BNN .....	225
BNOV .....	226
BNZ .....	226
BOR. <i>See</i> Brown-out Reset.	
BOV .....	229
BRA .....	227
Brown-out Reset (BOR) .....	42
Detecting .....	42
Disabling in Sleep Mode .....	42
Software Enabled .....	42
BSF .....	227
BTFSC .....	228
BTFSS .....	228
BTG .....	229
BZ .....	230

### C

C Compilers .....	
MPLAB C18 .....	212
MPLAB C30 .....	212
CALL .....	230
CALLW .....	259
Clock Sources .....	26
Selecting the 31 kHz Source .....	27
Selection Using OSCCON Register .....	27
CLRF .....	231
CLRWDT .....	231
Code Examples .....	
16 x 16 Signed Multiply Routine .....	86
16 x 16 Unsigned Multiply Routine .....	86
8 x 8 Signed Multiply Routine .....	85
8 x 8 Unsigned Multiply Routine .....	85
Computed GOTO Using an Offset Value .....	54
Data EEPROM Read .....	83
Data EEPROM Refresh Routine .....	84
Data EEPROM Write .....	83
Erasing a Flash Program Memory Row .....	76
Fast Register Stack .....	54

# PIC18F1230/1330

How to Clear RAM (Bank 0) Using Indirect Addressing . . . . .	65	Data Memory . . . . .	57
Implementing a Real-Time Clock Using a Timer1 Interrupt Service . . . . .	115	Access Bank . . . . .	59
Initializing PORTA . . . . .	87	and the Extended Instruction Set . . . . .	67
Initializing PORTB . . . . .	90	Bank Select Register (BSR) . . . . .	57
Reading a Flash Program Memory Word . . . . .	75	General Purpose Registers . . . . .	59
Saving STATUS, WREG and BSR Registers in RAM . . . . .	105	Map for PIC18F1230/1330 . . . . .	58
Writing to Flash Program Memory . . . . .	78–79	Special Function Registers . . . . .	60
Code Protection . . . . .	191, 207	DAW . . . . .	234
Associated Registers . . . . .	207	DC Characteristics . . . . .	279
Configuration Register Protection . . . . .	210	Power-Down and Supply Current . . . . .	269
Data EEPROM . . . . .	210	Supply Voltage . . . . .	268
Program Memory . . . . .	208	DCFSNZ . . . . .	235
COMF . . . . .	232	DECf . . . . .	234
Comparator . . . . .	179	DECFSZ . . . . .	235
Analog Input Connection Considerations . . . . .	181	Development Support . . . . .	211
Associated Registers . . . . .	182	Device Differences . . . . .	304
Configuration . . . . .	180	Device Overview . . . . .	9
Effects of a Reset . . . . .	181	Details on Individual Family Members . . . . .	10
Interrupts . . . . .	180	Features (table) . . . . .	11
Operation . . . . .	180	New Core Features . . . . .	9
Operation During Sleep . . . . .	181	Other Special Features . . . . .	10
Outputs . . . . .	180	Device Reset Timers . . . . .	43
Reference . . . . .	180	Oscillator Start-up Timer (OST) . . . . .	43
Response Time . . . . .	180	PLL Lock Time-out . . . . .	43
Comparator Specifications . . . . .	282	Power-up Timer (PWRT) . . . . .	43
Comparator Voltage Reference . . . . .	183	Time-out Sequence . . . . .	43
Accuracy and Error . . . . .	185	Direct Addressing . . . . .	66
Associated Registers . . . . .	185	<b>E</b>	
Configuring . . . . .	183	Effect on Standard PIC MCU Instructions . . . . .	262
Effects of a Reset . . . . .	185	Effects of Power-Managed Modes on Various Clock Sources . . . . .	29
Operation During Sleep . . . . .	185	Electrical Characteristics . . . . .	265
Computed GOTO . . . . .	54	Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART). See EUSART.	
Configuration Bits . . . . .	191	Equations	
Context Saving During Interrupts . . . . .	105	A/D Acquisition Time . . . . .	174
Conversion Considerations . . . . .	305	A/D Minimum Charging Time . . . . .	174
CPFSEQ . . . . .	232	Calculating the Minimum Required Acquisition Time . . . . .	174
CPFSGT . . . . .	233	PWM Frequency . . . . .	129
CPFSLT . . . . .	233	PWM Period for Continuous Up/Down Count Mode . . . . .	129
Crystal Oscillator/Ceramic Resonator . . . . .	21	PWM Period for Free-Running Mode . . . . .	129
Customer Change Notification Service . . . . .	314	PWM Resolution . . . . .	129
Customer Notification Service . . . . .	314	Errata . . . . .	7
Customer Support . . . . .	314	EUSART	
<b>D</b>		Asynchronous Mode . . . . .	157
Data Addressing Modes . . . . .	65	12-Bit Break Character Sequence . . . . .	163
Comparing Options with the Extended Instruction Set . . . . .		Associated Registers, Receive . . . . .	161
Enabled . . . . .	68	Associated Registers, Transmit . . . . .	159
Direct . . . . .	65	Auto-Wake-up on Sync Break Character . . . . .	161
Indexed Literal Offset . . . . .	67	Receiver . . . . .	160
Instructions Affected . . . . .	67	Receiving a Break Character . . . . .	163
Indirect . . . . .	65	Setting Up 9-Bit Mode with Address Detect . . . . .	160
Inherent and Literal . . . . .	65	Transmitter . . . . .	157
Data EEPROM Memory . . . . .	81	Baud Rate Generator	
Associated Registers . . . . .	84	Operation in Power-Managed Modes . . . . .	151
EEADR Register . . . . .	81	Baud Rate Generator (BRG) . . . . .	151
EECON1 and EECON2 Registers . . . . .	81	Associated Registers . . . . .	152
Operation During Code-Protect . . . . .	84	Auto-Baud Rate Detect . . . . .	155
Protection Against Spurious Write . . . . .	83	Baud Rate Error, Calculating . . . . .	152
Reading . . . . .	83	Baud Rates, Asynchronous Modes . . . . .	153
Using . . . . .	84	High Baud Rate Select (BRGH Bit) . . . . .	151
Write Verify . . . . .	83	Sampling . . . . .	151
Writing . . . . .	83	Synchronous Master Mode . . . . .	164
		Associated Registers, Receive . . . . .	166

Associated Registers, Transmit .....	165	<b>I</b>	
Reception .....	166	I/O Ports .....	87
Transmission .....	164	ID Locations .....	191, 210
Synchronous Slave Mode .....	167	INCF .....	236
Associated Registers, Receive .....	168	INCFSZ .....	237
Associated Registers, Transmit .....	167	In-Circuit Debugger .....	210
Reception .....	168	In-Circuit Serial Programming (ICSP) .....	191, 210
Transmission .....	167	Independent PWM Mode	
Extended Instruction Set		Duty Cycle Assignment .....	137
ADDFSR .....	258	Output .....	137
ADDULNK .....	258	Output, Channel Override .....	138
and Using MPLAB Tools .....	264	Indexed Literal Offset Addressing	
CALLW .....	259	and Standard PIC18 Instructions .....	262
Considerations for Use .....	262	Indexed Literal Offset Mode .....	262
MOVSF .....	259	Indirect Addressing .....	66
MOVSS .....	260	INFSNZ .....	237
PUSHL .....	260	Initialization Conditions for all Registers .....	47–50
SUBFSR .....	261	Instruction Cycle .....	55
SUBULNK .....	261	Clocking Scheme .....	55
Syntax .....	257	Flow/Pipelining .....	55
External Clock Input .....	22	Instruction Set .....	215
<b>F</b>		ADDLW .....	221
Fail-Safe Clock Monitor .....	191, 205	ADDWF .....	221
Exiting Operation .....	205	ADDWF (Indexed Literal Offset Mode) .....	263
Interrupts in Power-Managed Modes .....	206	ADDWFC .....	222
POR or Wake From Sleep .....	206	ANDLW .....	222
WDT During Oscillator Failure .....	205	ANDWF .....	223
Fast Register Stack .....	54	BC .....	223
Firmware Instructions .....	215	BCF .....	224
Flash Program Memory .....	71	BN .....	224
Associated Registers .....	79	BNC .....	225
Control Registers .....	72	BNN .....	225
EECON1 and EECON2 .....	72	BNOV .....	226
TABLAT (Table Latch) Register .....	74	BNZ .....	226
TBLPTR (Table Pointer) Register .....	74	BOV .....	229
Erase Sequence .....	76	BRA .....	227
Erasing .....	76	BSF .....	227
Operation During Code-Protect .....	79	BSF (Indexed Literal Offset Mode) .....	263
Reading .....	75	BTFSC .....	228
Table Pointer		BTFSS .....	228
Boundaries Based on Operation .....	74	BTG .....	229
Operations with TBLRD and TBLWT (table) .....	74	BZ .....	230
Table Pointer Boundaries .....	74	CALL .....	230
Table Reads and Table Writes .....	71	CLRf .....	231
Write Sequence .....	77	CLRWDt .....	231
Writing .....	77	COMF .....	232
Protection Against Spurious Writes .....	79	CPFSEQ .....	232
Unexpected Termination .....	79	CPFSGT .....	233
Write Verify .....	79	CPFSLT .....	233
FSCM. See Fail-Safe Clock Monitor.		DAW .....	234
<b>G</b>		DCFSNZ .....	235
GOTO .....	236	DECf .....	234
<b>H</b>		DECFSZ .....	235
Hardware Multiplier .....	85	Extended Instruction Set .....	257
Introduction .....	85	General Format .....	217
Operation .....	85	GOTO .....	236
Performance Comparison .....	85	INCF .....	236
		INCFSZ .....	237
		INFSNZ .....	237
		IORLW .....	238
		IORWF .....	238
		LFSR .....	239
		MOVf .....	239
		MOVFF .....	240
		MOVLB .....	240

# PIC18F1230/1330

MOVLW .....	241	<b>L</b>	
MOVWF .....	241	LFSR .....	239
MULLW .....	242	Low-Voltage Detect .....	187
MULWF .....	242	Applications .....	190
NEGF .....	243	Associated Registers .....	190
NOP .....	243	Characteristics .....	283
Opcode Field Descriptions .....	216	Current Consumption .....	189
POP .....	244	Effects of a Reset .....	190
PUSH .....	244	Operation .....	188
RCALL .....	245	During Sleep .....	190
RESET .....	245	Setup .....	189
RETFIE .....	246	Start-up Time .....	189
RETLW .....	246	Typical Application .....	190
RETURN .....	247	Low-Voltage ICSP Programming. See Single-Supply ICSP	
RLCF .....	247	Programming .....	
RLNCF .....	248	LVD. See Low-Voltage Detect. ....	187
RRCF .....	248	<b>M</b>	
RRNCF .....	249	Master Clear (MCLR) .....	41
SETF .....	249	Memory Organization .....	51
SETF (Indexed Literal Offset Mode) .....	263	Data Memory .....	57
SLEEP .....	250	Program Memory .....	51
Standard Instructions .....	215	Memory Programming Requirements .....	281
SUBFWB .....	250	Microchip Internet Web Site .....	314
SUBLW .....	251	Migration from Baseline to Enhanced Devices .....	305
SUBWF .....	251	Migration from High-End to Enhanced Devices .....	306
SUBWFB .....	252	Migration from Mid-Range to Enhanced Devices .....	306
SWAPF .....	252	MOVF .....	239
TBLRD .....	253	MOVFF .....	240
TBLWT .....	254	MOVLB .....	240
TSTFSZ .....	255	MOVLW .....	241
XORLW .....	255	MOVSF .....	259
XORWF .....	256	MOVSS .....	260
INTCON Registers .....	95–97	MOVWF .....	241
Internal Oscillator Block .....	24	MPLAB ASM30 Assembler, Linker, Librarian .....	212
Adjustment .....	24	MPLAB ICD 2 In-Circuit Debugger .....	213
INTIO Modes .....	24	MPLAB ICE 2000 High-Performance Universal In-Circuit Em-	
INTOSC Frequency Drift .....	24	ulator .....	213
INTOSC Output Frequency .....	24	MPLAB Integrated Development Environment Software ..	211
OSCTUNE Register .....	24	MPLAB PM3 Device Programmer .....	213
PLL in INTOSC Modes .....	24	MPLAB REAL ICE In-Circuit Emulator System .....	213
Internal RC Oscillator		MPLINK Object Linker/MPLIB Object Librarian .....	212
Use with WDT .....	202	MULLW .....	242
Internet Address .....	314	MULWF .....	242
Interrupt Sources .....	191	<b>N</b>	
A/D Conversion Complete .....	173	NEGF .....	243
INTx Pin .....	105	NOP .....	243
PORTB, Interrupt-on-Change .....	105	<b>O</b>	
TMR0 .....	105	Oscillator Configuration .....	21
TMR1 Overflow .....	111	EC .....	21
Interrupts .....	93	ECIO .....	21
Interrupts, Flag Bits		HS .....	21
Interrupt-on-Change Flag (RBIF Bit) .....	90	HSPLL .....	21
INTOSC, INTRC. See Internal Oscillator Block.		Internal Oscillator Block .....	24
IORLW .....	238	INTIO1 .....	21
IORWF .....	238	INTIO2 .....	21
IPR Registers .....	102	LP .....	21
		RC .....	21
		RCIO .....	21
		XT .....	21
		Oscillator Selection .....	191
		Oscillator Start-up Timer (OST) .....	29, 43
		Oscillator Switching .....	26

Oscillator Transitions .....	27	PRI_IDLE .....	36
Oscillator, Timer1 .....	111	RC_IDLE .....	37
<b>P</b>		SEC_IDLE .....	36
Packaging .....	295	Multiple Sleep Commands .....	32
Details .....	296	Run Modes .....	32
Marking Information .....	295	PRI_RUN .....	32
PICSTART Plus Development Programmer .....	214	RC_RUN .....	33
PIE Registers .....	100	SEC_RUN .....	32
Pin Functions		Selecting .....	31
AVDD .....	16	Sleep Mode .....	35
AVss .....	16	Summary (table) .....	31
MCLR/VPP/RA5/FLTA .....	13	Power-on Reset (POR) .....	41
NC .....	16	Time-out Sequence .....	43
RA0/AN0/INT0/KBI0/CMP0 .....	14	Power-up Delays .....	29
RA1/AN1/INT1/KBI1 .....	14	Power-up Timer (PWRT) .....	29
RA2/TX/CK .....	14	Prescaler, Timer0 .....	109
RA3/RX/DT .....	14	PRI_IDLE Mode .....	36
RA4/T0CKI/AN2/VREF+ .....	14	PRI_RUN Mode .....	32
RA6/OSC2/CLKO/T1OSO/T1CKI/AN3 .....	13	Program Counter .....	52
RA7/OSC1/CLKI/T1OSI/FLTA .....	13	PCL, PCH and PCU Registers .....	52
RB0/PWM0 .....	15	PCLATH and PCLATU Registers .....	52
RB1/PWM1 .....	15	Program Memory	
RB2/INT2/KBI2/CMP2/T1OSO/T1CKI .....	15	and Extended Instruction Set .....	69
RB3/INT3/KBI3/CMP1/T1OSI .....	15	Instructions .....	56
RB4/PWM2 .....	15	Two-Word .....	56
RB5/PWM3 .....	15	Interrupt Vector .....	51
RB6/PWM4/PGC .....	15	Look-up Tables .....	54
RB7/PWM5/PGD .....	15	Map and Stack (diagram) .....	51
VDD .....	16	Reset Vector .....	51
Vss .....	16	Program Verification .....	207
Pinout I/O Descriptions		Programming, Device Instructions .....	215
PIC18F1230/1330 .....	13	PUSH .....	244
PIR Registers .....	98	PUSH and POP Instructions .....	53
PLL Frequency Multiplier .....	23	PUSHL .....	260
HSPLL Oscillator Mode .....	23	PWM	
Use with INTOSC .....	23	Fault Input .....	142
POP .....	244	Output and Polarity Control .....	141
POR. See Power-on Reset.		Single-Pulse Operation .....	138
PORTA		Special Event Trigger .....	144
Associated Registers .....	89	Update Lockout .....	144
LATA Register .....	87	PWM Dead-Time	
PORTA Register .....	87	Decrementing the Counter .....	136
TRISA Register .....	87	Distortion .....	137
PORTB		Generators .....	135
Associated Registers .....	92	Insertion .....	135
Interrupt-on-Change Flag (RBIF Bit) .....	90	Ranges .....	136
LATB Register .....	90	PWM Duty Cycle .....	131
PORTB Register .....	90	Center-Aligned .....	133
TRISB Register .....	90	Complementary Operation .....	134
Power Control PWM .....	117	Edge-Aligned .....	132
Associated Registers .....	145	Register Buffers .....	132
Control Registers .....	120	Registers .....	131
Functionality .....	120	PWM Output Override .....	138
Power-Managed Modes .....	31	Complementary Mode .....	138
and A/D Operation .....	176	Examples .....	140
Clock Sources .....	31	Synchronization .....	138
Clock Transitions and Status Indicators .....	32	PWM Period .....	129
Effects on Clock Sources .....	29	PWM Time Base .....	120
Entering .....	31	Continuous Up/Down Count Modes .....	125
Exiting Idle and Sleep Modes .....	37	Free-Running Mode .....	125
By Interrupt .....	37	Interrupts .....	125
By Reset .....	37	In Continuous Up/Down Count Mode .....	126
By WDT Time-out .....	37	In Double Update Mode .....	128
Without an Oscillator Start-up Delay .....	38	In Free-Running Mode .....	125
Idle Modes .....	35	In Single-Shot Mode .....	126

# PIC18F1230/1330

Postscaler .....	125
Prescaler .....	125
Single-Shot Mode .....	125

## R

RAM. See Data Memory.

RBIF Bit .....	90
RC Oscillator .....	23
RCIO Oscillator Mode .....	23
RC_IDLE Mode .....	37
RC_RUN Mode .....	33
RCALL .....	245

### RCON Register

Bit Status During Initialization .....	46
--	----

Reader Response .....	315
-----------------------	-----

Register File Summary .....	61–63
-----------------------------	-------

### Registers

ADCON0 (A/D Control 0) .....	169
ADCON1 (A/D Control 1) .....	170
ADCON2 (A/D Control 2) .....	171
BAUDCON (Baud Rate Control) .....	150
CMCON (Comparator Control) .....	179
CONFIG1H (Configuration 1 High) .....	192
CONFIG2H (Configuration 2 High) .....	194
CONFIG2L (Configuration 2 Low) .....	193
CONFIG3H (Configuration 3 High) .....	196
CONFIG3L (Configuration 3 Low) .....	195
CONFIG4L (Configuration 4 Low) .....	197
CONFIG5H (Configuration 5 High) .....	198
CONFIG5L (Configuration 5 Low) .....	198
CONFIG6H (Configuration 6 High) .....	199
CONFIG6L (Configuration 6 Low) .....	199
CONFIG7H (Configuration 7 High) .....	200
CONFIG7L (Configuration 7 Low) .....	200
CVRCON (Comparator Voltage Reference Control) .....	184
DEVID1 (Device ID 1) .....	201
DEVID2 (Device ID 2) .....	201
DTCON (Dead-Time Control) .....	136
EECON1 (EEPROM Control 1) .....	73, 82
FLTCONFIG (Fault Configuration) .....	143
INTCON (Interrupt Control) .....	95
INTCON2 (Interrupt Control 2) .....	96
INTCON3 (Interrupt Control 3) .....	97
IPR1 (Peripheral Interrupt Priority 1) .....	102
IPR2 (Peripheral Interrupt Priority 2) .....	103
IPR3 (Peripheral Interrupt Priority 3) .....	103
LVDCON (Low-Voltage Detect Control) .....	187
OSCCON (Oscillator Control) .....	28
OSCTUNE (Oscillator Tuning) .....	25
OVDCOND (Output Override Control) .....	140
OVDCONS (Output State) .....	140
PIE1 (Peripheral Interrupt Enable 1) .....	100
PIE2 (Peripheral Interrupt Enable 2) .....	101
PIE3 (Peripheral Interrupt Enable 3) .....	101
PIR1 (Peripheral Interrupt Request (Flag) 1) .....	98
PIR2 (Peripheral Interrupt Request (Flag) 2) .....	99
PIR3 (Peripheral Interrupt Request (Flag) 3) .....	99
PTCON0 (PWM Timer Control 0) .....	122
PTCON1 (PWM Timer Control 1) .....	122
PWMCON0 (PWM Control 0) .....	123
PWMCON1 (PWM Control 1) .....	124
RCON (Reset Control) .....	40, 104
RCSTA (Receive Status and Control) .....	149
STATUS .....	64
STKPTR (Stack Pointer) .....	53
T0CON (Timer0 Control) .....	107

T1CON (Timer1 Control) .....	111
TXSTA (Transmit Status and Control) .....	148
WDTCON (Watchdog Timer Control) .....	203

RESET .....	245
-------------	-----

Reset State of Registers .....	46
--------------------------------	----

Resets .....	39, 191
--------------	---------

Brown-out Reset (BOR) .....	191
-----------------------------	-----

Oscillator Start-up Timer (OST) .....	191
---------------------------------------	-----

Power-on Reset (POR) .....	191
----------------------------	-----

Power-up Timer (PWRT) .....	191
-----------------------------	-----

RETFIE .....	246
--------------	-----

RETLW .....	246
-------------	-----

RETURN .....	247
--------------	-----

Return Address Stack .....	52
----------------------------	----

Associated Registers .....	52
----------------------------	----

Return Stack Pointer (STKPTR) .....	53
-------------------------------------	----

Revision History .....	303
------------------------	-----

RLCF .....	247
------------	-----

RLNCF .....	248
-------------	-----

RRCF .....	248
------------	-----

RRNCF .....	249
-------------	-----

## S

SEC_IDLE Mode .....	36
---------------------	----

SEC_RUN Mode .....	32
--------------------	----

SETF .....	249
------------	-----

Single-Supply ICSP Programming .....	210
--------------------------------------	-----

Single-Supply ICSP Programming .....	210
--------------------------------------	-----

SLEEP .....	250
-------------	-----

### Sleep

OSC1 and OSC2 Pin States .....	29
--------------------------------	----

Software Simulator (MPLAB SIM) .....	212
--------------------------------------	-----

Special Features of the CPU .....	191
-----------------------------------	-----

### Special Function Registers

Map .....	60
-----------	----

Stack Full/Underflow Resets .....	54
-----------------------------------	----

SUBFSR .....	261
--------------	-----

SUBFWB .....	250
--------------	-----

SUBLW .....	251
-------------	-----

SUBULNK .....	261
---------------	-----

SUBWFB .....	251
--------------	-----

SUBWFB .....	252
--------------	-----

SWAPF .....	252
-------------	-----

## T

Table Reads/Table Writes .....	54
--------------------------------	----

TBLRD .....	253
-------------	-----

TBLWT .....	254
-------------	-----

Time-out in Various Situations (table) .....	43
--	----

Timer0 .....	107
--------------	-----

16-Bit Mode Timer Reads and Writes .....	109
--	-----

Associated Registers .....	109
----------------------------	-----

Clock Source Edge Select (T0SE Bit) .....	109
---	-----

Clock Source Select (T0CS Bit) .....	109
--------------------------------------	-----

Interrupt .....	109
-----------------	-----

Operation .....	109
-----------------	-----

Prescaler .....	109
-----------------	-----

Switching the Assignment .....	109
--------------------------------	-----

Prescaler Assignment (PSA Bit) .....	109
--------------------------------------	-----

Prescaler Select (T0PS2:T0PS0 Bits) .....	109
---	-----

Prescaler. See Prescaler, Timer0.	
-----------------------------------	--



Timer1 .....	111	VDD Rise < TPWRT) .....	44
16-Bit Read/Write Mode .....	114	Timer0 and Timer1 External Clock .....	290
Associated Registers .....	115	Transition for Entry to Idle Mode .....	36
Interrupt .....	114	Transition for Entry to SEC_RUN Mode .....	33
Operation .....	112	Transition for Entry to Sleep Mode .....	35
Oscillator .....	111, 113	Transition for Two-Speed Start-up (INTOSC to HSPLL) 204	
Oscillator Layout Considerations .....	113	Transition for Wake From Idle to Run Mode .....	36
Overflow Interrupt .....	111	Transition for Wake From Sleep (HSPLL) .....	35
TMR1H Register .....	111	Transition from RC_RUN Mode to PRI_RUN Mode ..	34
TMR1L Register .....	111	Transition from SEC_RUN Mode to PRI_RUN Mode (HSPLL) .....	33
Use as a Clock Source .....	113	Transition to RC_RUN Mode .....	34
Use as a Real-Time Clock .....	114	Timing Diagrams and Specifications .....	286
Timing Diagrams		CLKO and I/O Requirements .....	288
A/D Conversion .....	293	EUSART Synchronous Receive Requirements .....	291
Asynchronous Reception .....	161	EUSART Synchronous Transmission Requirements .....	291
Asynchronous Transmission .....	158	External Clock Requirements .....	286
Asynchronous Transmission (Back-to-Back) .....	158	PLL Clock .....	287
Automatic Baud Rate Calculation .....	156	Reset, Watchdog Timer, Oscillator Start-up Timer, Pow- er-up Timer and Brown-out Reset Requirements ..	289
Auto-Wake-up Bit (WUE) During Normal Operation	162	Timer0 and Timer1 External Clock Requirements ...	290
Auto-Wake-up Bit (WUE) During Sleep .....	162	Top-of-Stack Access .....	52
BRG Overflow Sequence .....	156	TSTFSZ .....	255
Brown-out Reset (BOR) .....	289	Two-Speed Start-up .....	191, 204
CLKO and I/O .....	288	Two-Word Instructions	
Clock/Instruction Cycle .....	55	Example Cases .....	56
Dead-Time Insertion for Complementary PWM .....	135	TXSTA Register	
Duty Cycle Update Times in Continuous Up/Down Count Mode .....	132	BRGH Bit .....	151
Duty Cycle Update Times in Continuous Up/Down Count Mode with Double Updates .....	133	<b>V</b>	
Edge-Aligned PWM .....	132	Voltage Reference Specifications .....	282
EUSART Synchronous Receive (Master/Slave) .....	291	<b>W</b>	
EUSART Synchronous Transmission (Master/Slave) .... 291		Watchdog Timer (WDT) .....	191, 202
External Clock (All Modes Except PLL) .....	286	Associated Registers .....	203
Fail-Safe Clock Monitor .....	206	Control Register .....	202
Low-Voltage Detect Characteristics .....	283	During Oscillator Failure .....	205
Low-Voltage Detect Operation .....	189	Programming Considerations .....	202
Override Bits in Complementary Mode .....	139	WWW Address .....	314
PWM Output Override Example #1 .....	141	WWW, On-Line Support .....	7
PWM Output Override Example #2 .....	141	<b>X</b>	
PWM Period Buffer Updates in Continuous Up/Down Count Modes .....	130	XORLW .....	255
PWM Period Buffer Updates in Free-Running Mode	130	XORWF .....	256
PWM Time Base Interrupt (Free-Running Mode) ....	126		
PWM Time Base Interrupt (Single-Shot Mode) .....	127		
PWM Time Base Interrupts (Continuous Up/Down Count Mode with Double Updates) .....	128		
PWM Time Base Interrupts (Continuous Up/Down Count Mode) .....	127		
Reset, Watchdog Timer (WDT), Oscillator Start-up Timer (OST), Power-up Timer (PWRT) .....	289		
Send Break Character Sequence .....	163		
Slow Rise Time (MCLR Tied to VDD, VDD Rise > TPWRT) .....	45		
Start of Center-Aligned PWM .....	133		
Synchronous Reception (Master Mode, SREN) .....	166		
Synchronous Transmission .....	164		
Synchronous Transmission (Through TXEN) .....	165		
Time-out Sequence on POR w/PLL Enabled (MCLR Tied to VDD) .....	45		
Time-out Sequence on Power-up (MCLR Not Tied to VDD, Case 1) .....	44		
Time-out Sequence on Power-up (MCLR Not Tied to VDD, Case 2) .....	44		
Time-out Sequence on Power-up (MCLR Tied to VDD,			

# PIC18F1230/1330

---

NOTES:

## THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at [www.microchip.com](http://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](http://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>**

# PIC18F1230/1330

---

---

## READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager  
RE: Reader Response  
Total Pages Sent \_\_\_\_\_

From: Name \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City / State / ZIP / Country \_\_\_\_\_  
Telephone: (\_\_\_\_\_) \_\_\_\_\_ - \_\_\_\_\_ FAX: (\_\_\_\_\_) \_\_\_\_\_ - \_\_\_\_\_

Application (optional):

Would you like a reply? \_\_\_Y \_\_\_N

Device: PIC18F1230/1330 Literature Number: DS39758D

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?

---

---

## PIC18F1230/1330 PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

<u>PART NO.</u>	<u>X</u>	<u>/XX</u>	<u>XXX</u>
Device	Temperature Range	Package	Pattern
Device	PIC18F1230/1330 <sup>(1)</sup> PIC18F1230/1330T <sup>(2)</sup> VDD range 4.2V to 5.5V PIC18LF1230/1330 <sup>(1)</sup> PIC18LF1230/1330T <sup>(2)</sup> VDD range 2.0V to 5.5V		
Temperature Range	I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended)		
Package	SO = Plastic Small Outline (SOIC) SS = Plastic Shrink Small Outline (SSOP) P = Plastic Dual In-line (PDIP) ML = Plastic Quad Flat No Lead (QFN)		
Pattern	QTP, SQTP, Code or Special Requirements (blank otherwise)		

**Examples:**

a) PIC18LF1330-I/P 301 = Industrial temp., PDIP package, Extended VDD limits, QTP pattern #301.

b) PIC18LF1230-I/SO = Industrial temp., SOIC package, Extended VDD limits.

**Note 1:** F = Standard Voltage Range  
 LF = Wide Voltage Range

**2:** T = in tape and reel



---

---

## 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](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Cleveland**  
Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**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  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4080

**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 - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-6578-300  
Fax: 886-3-6578-370

**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: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820

03/26/09



Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits, General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



#### Как с нами связаться

**Телефон:** 8 (812) 309 58 32 (многоканальный)

**Факс:** 8 (812) 320-02-42

**Электронная почта:** [org@eplast1.ru](mailto:org@eplast1.ru)

**Адрес:** 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.