

## 28/40/44/64-Pin, Enhanced Flash Microcontrollers with ECAN™ and nanoWatt XLP Technology

### Power-Managed Modes:

- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Two-Speed Oscillator Start-up
- Fail-Safe Clock Monitor (FSCM)
- Power-Saving Peripheral Module Disable (PMD)
- Ultra Low-Power Wake-up
- Fast Wake-up, 1 µs, Typical
- Low-Power WDT, 300 nA, Typical
- Run mode Currents Down to Very Low 3.8 µA, Typical
- Idle mode Currents Down to Very Low 880 nA, Typical
- Sleep mode Current Down to Very Low 13 nA, Typical

### ECAN Bus Module Features:

- Conforms to CAN 2.0B Active Specification
- Three Operating modes:
  - Legacy mode (full backward compatibility with existing PIC18CXX8/FXX8 CAN modules)
  - Enhanced mode
  - FIFO mode or programmable TX/RX buffers
- Message Bit Rates up to 1 Mbps
- DeviceNet™ Data Byte Filter Support
- Six Programmable Receive/Transmit Buffers
- Three Dedicated Transmit Buffers with Prioritization
- Two Dedicated Receive Buffers

### ECAN Bus Module Features (Continued):

- 16 Full, 29-Bit Acceptance Filters with Dynamic Association
- Three Full, 29-Bit Acceptance Masks
- Automatic Remote Frame Handling
- Advanced Error Management Features

### Special Microcontroller Features:

- Operating Voltage Range: 1.8V to 5.5V
- On-Chip 3.3V Regulator
- Operating Speed up to 64 MHz
- Up to 64 Kbytes On-Chip Flash Program Memory:
  - 10,000 erase/write cycle, typical
  - 20 years minimum retention, typical
- 1,024 Bytes of Data EEPROM:
  - 100,000 Erase/write cycle data EEPROM memory, typical
- 3.6 Kbytes of General Purpose Registers (SRAM)
- Three Internal Oscillators: LF-INTOSC (31 KHz), MF-INTOSC (500 kHz) and HF-INTOSC (16 MHz)
- 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 4,194s
- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug via Two Pins
- Programmable BOR
- Programmable LVD

**TABLE 1: DEVICE COMPARISON**

Device	Program Memory	Data Memory (Bytes)	Data EE (Bytes)	Pins	I/O	CTMU	12-Bit A/D Channels	CCP/ECCP	Timers 8-Bit/16-Bit	EUSART	Comparators	ECAN™	MSSP	BORMV/LVD	DSM
PIC18F25K80	32 Kbytes	3,648	1,024	28	24	1	8-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18LF25K80	32 Kbytes	3,648	1,024	28	24	1	8-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18F26K80	64 Kbytes	3,648	1,024	28	24	1	8-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18LF26K80	64 Kbytes	3,648	1,024	28	24	1	8-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18F45K80	32 Kbytes	3,648	1,024	40/44	35	1	11-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18LF45K80	32 Kbytes	3,648	1,024	40/44	35	1	11-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18F46K80	64 Kbytes	3,648	1,024	40/44	35	1	11-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18LF46K80	64 Kbytes	3,648	1,024	40/44	35	1	11-ch	4/1	2/3	2	2	1	1	Yes	No
PIC18F65K80	32 Kbytes	3,648	1,024	64	54	1	11-ch	4/1	2/3	2	2	1	1	Yes	Yes
PIC18LF65K80	32 Kbytes	3,648	1,024	64	54	1	11-ch	4/1	2/3	2	2	1	1	Yes	Yes
PIC18F66K80	64 Kbytes	3,648	1,024	64	54	1	11-ch	4/1	2/3	2	2	1	1	Yes	Yes
PIC18LF66K80	64 Kbytes	3,648	1,024	64	54	1	11-ch	4/1	2/3	2	2	1	1	Yes	Yes

# PIC18F66K80 FAMILY

---

---

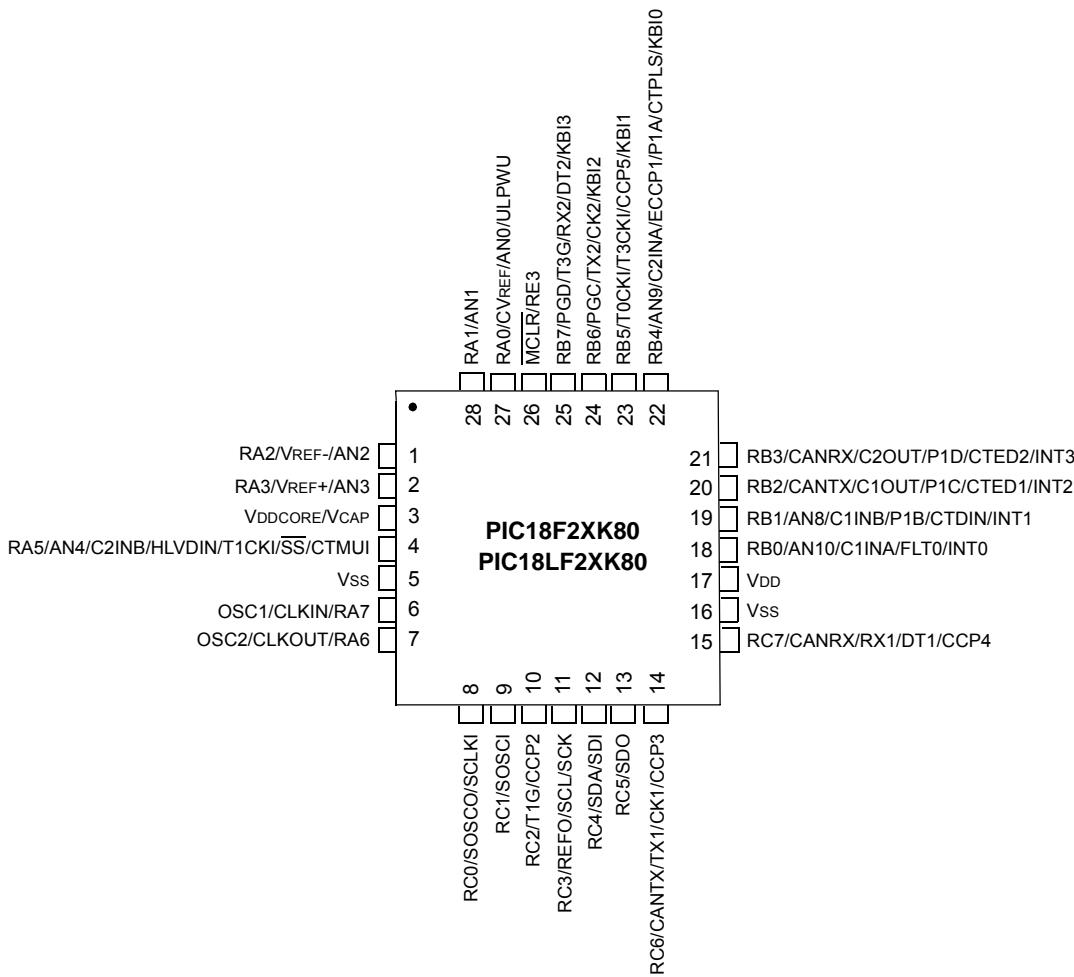
## Peripheral Highlights:

- Five CCP/ECCP modules:
  - Four Capture/Compare/PWM (CCP) modules
  - One Enhanced Capture/Compare/PWM (ECCP) module
- Five 8/16-Bit Timer/Counter modules:
  - Timer0: 8/16-bit timer/counter with 8-bit programmable prescaler
  - Timer1, Timer3: 16-bit timer/counter
  - Timer2, Timer4: 8-bit timer/counter
- Two Analog Comparators
- Configurable Reference Clock Output
- Charge Time Measurement Unit (CTMU):
  - Capacitance measurement
  - Time measurement with 1 ns typical resolution
  - Integrated voltage reference
- High-Current Sink/Source 25 mA/25 mA (PORTB and PORTC)
- Up to Four External Interrupts
- One Master Synchronous Serial Port (MSSP) module:
  - 3/4-wire SPI (supports all four SPI modes)
  - I<sup>2</sup>C™ Master and Slave modes
- Two Enhanced Addressable USART modules:
  - LIN/J2602 support
  - Auto-Baud Detect (ABD)
- 12-Bit A/D Converter with up to 11 Channels:
  - Auto-acquisition and Sleep operation
  - Differential Input mode of operation
- Data Signal Modulator module:
  - Select modulator and carrier sources from various module outputs
- Integrated Voltage Reference

# PIC18F66K80 FAMILY

## Pin Diagrams

### 28-Pin QFN<sup>(1)</sup>

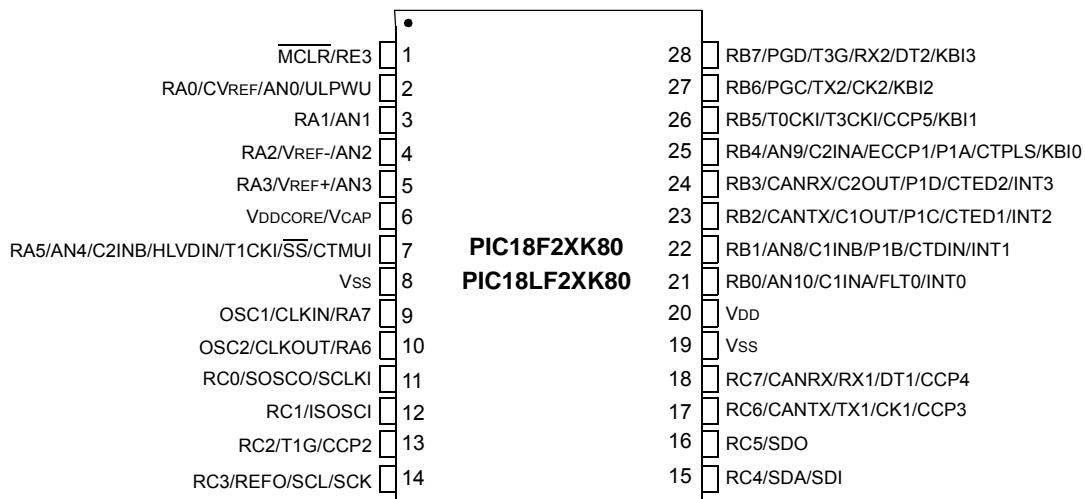


**Note 1:** For the QFN package, it is recommended that the bottom pad be connected to Vss.

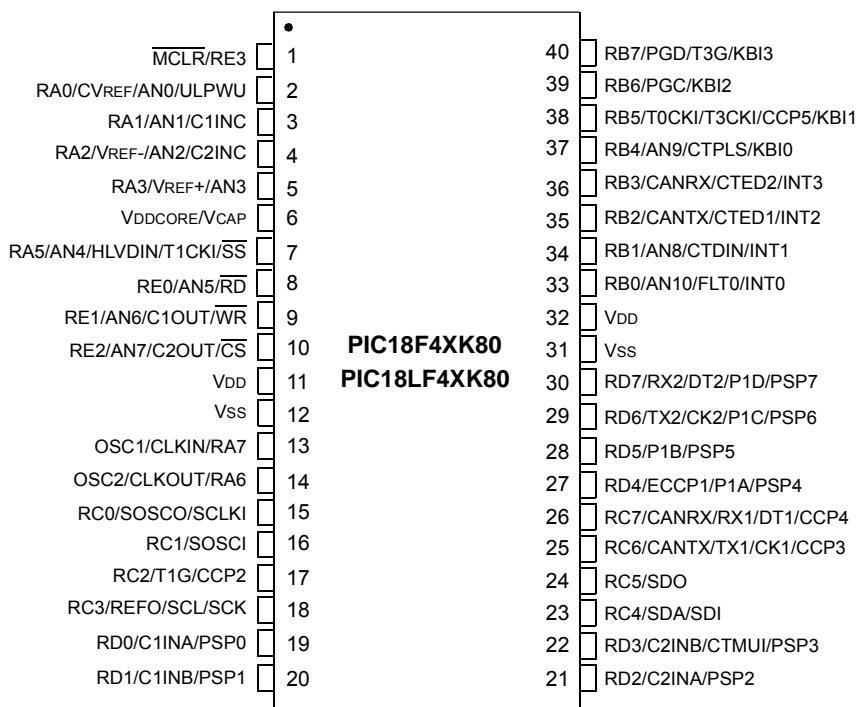
# PIC18F66K80 FAMILY

## Pin Diagrams (Continued)

### 28-Pin SSOP/SPDIP/SOIC



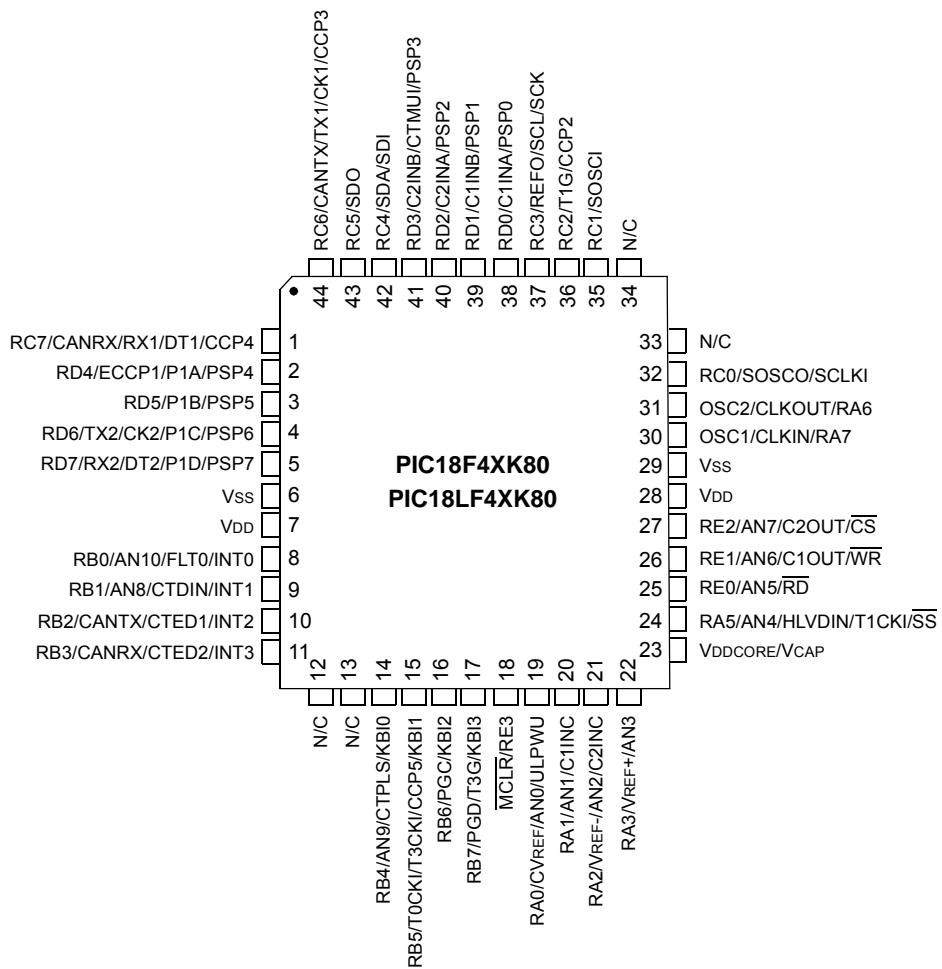
### 40-Pin PDIP



# PIC18F66K80 FAMILY

## Pin Diagrams (Continued)

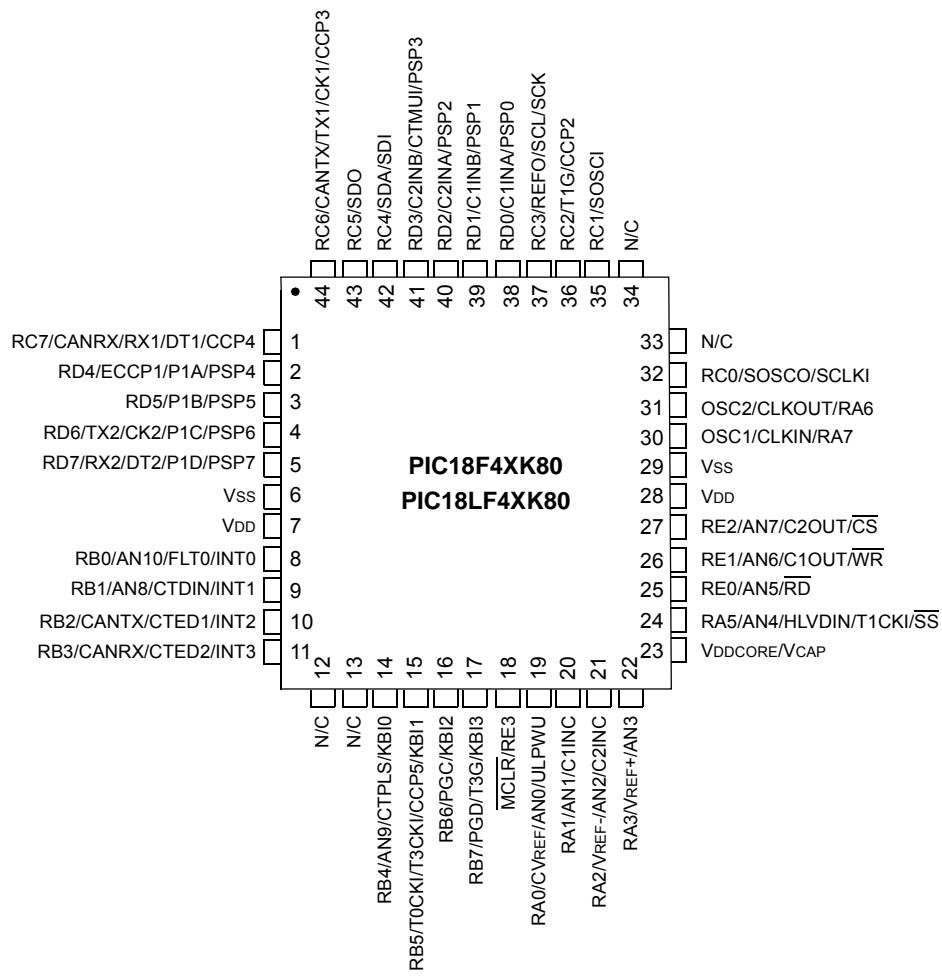
### 44-Pin TQFP



# PIC18F66K80 FAMILY

## Pin Diagrams (Continued)

### 44-Pin QFN<sup>(1)</sup>

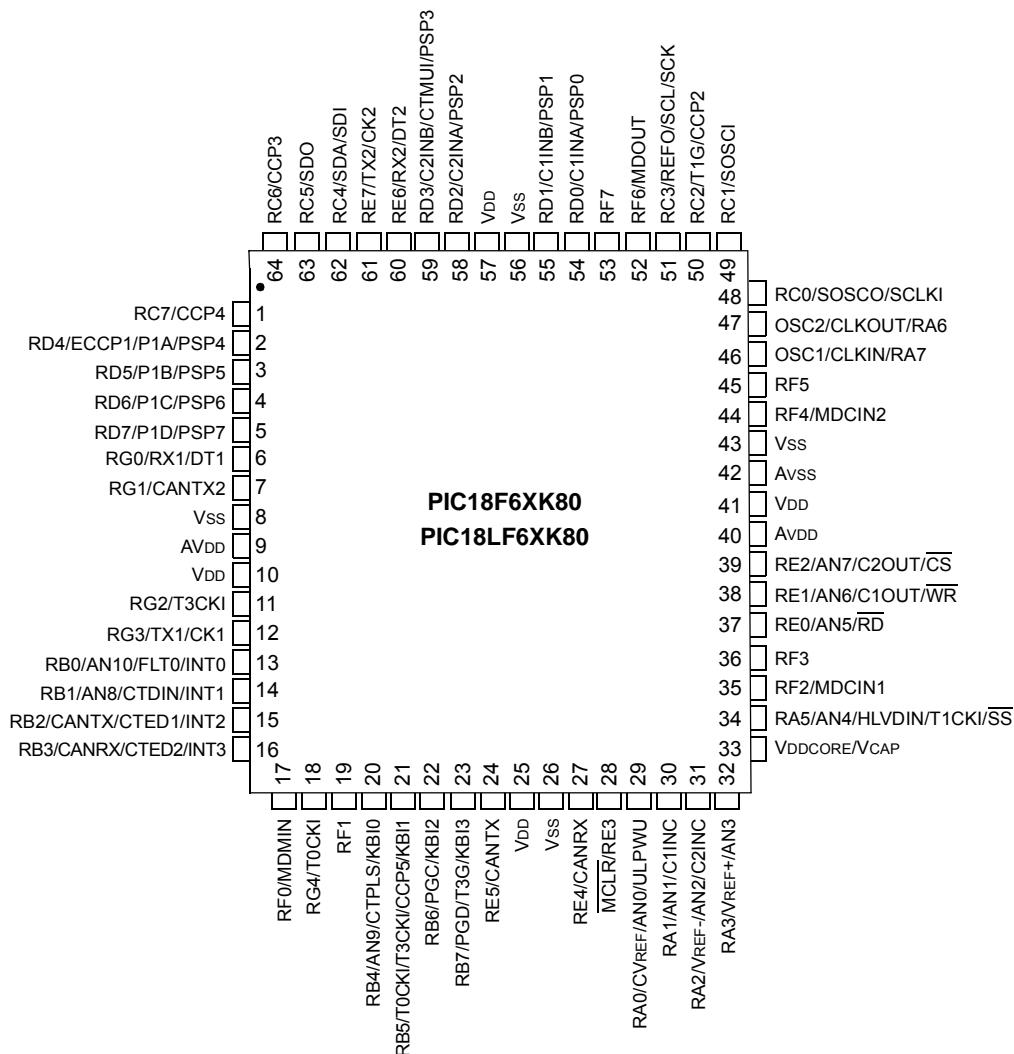


**Note 1:** For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F66K80 FAMILY

## Pin Diagrams (Continued)

64-Pin QFN<sup>(1)</sup>/TQFP



Note 1: For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F66K80 FAMILY

---

---

## Table of Contents

1.0	Device Overview .....	11
2.0	Guidelines for Getting Started with PIC18FXXKXX Microcontrollers .....	45
3.0	Oscillator Configurations .....	51
4.0	Power-Managed Modes .....	65
5.0	Reset .....	79
6.0	Memory Organization .....	101
7.0	Flash Program Memory .....	129
8.0	Data EEPROM Memory .....	139
9.0	8 x 8 Hardware Multiplier .....	145
10.0	Interrupts .....	147
11.0	I/O Ports .....	171
12.0	Data Signal Modulator .....	195
13.0	Timer0 Module .....	205
14.0	Timer1 Module .....	209
15.0	Timer2 Module .....	221
16.0	Timer3 Module .....	223
17.0	Timer4 Modules .....	233
18.0	Charge Time Measurement Unit (CTMU) .....	235
19.0	Capture/Compare/PWM (CCP) Modules .....	253
20.0	Enhanced Capture/Compare/PWM (ECCP) Module .....	265
21.0	Master Synchronous Serial Port (MSSP) Module .....	287
22.0	Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) .....	333
23.0	12-Bit Analog-to-Digital Converter (A/D) Module .....	357
24.0	Comparator Module .....	373
25.0	Comparator Voltage Reference Module .....	381
26.0	High/Low-Voltage Detect (HLVD) .....	385
27.0	ECAN Module .....	391
28.0	Special Features of the CPU .....	457
29.0	Instruction Set Summary .....	483
30.0	Development Support .....	533
31.0	Electrical Characteristics .....	537
32.0	Packaging Information .....	581
	Appendix A: Revision History .....	601
	Appendix B: Migration to PIC18F66K80 Family .....	602
	Index .....	605
	The Microchip Web Site .....	619
	Customer Change Notification Service .....	619
	Customer Support .....	619
	Reader Response .....	620
	Product Identification System .....	621

## 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.

# **PIC18F66K80 FAMILY**

---

---

**NOTES:**

## 1.0 DEVICE OVERVIEW

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

- PIC18F25K80
- PIC18F26K80
- PIC18F45K80
- PIC18F46K80
- PIC18F65K80
- PIC18F66K80
- PIC18LF25K80
- PIC18LF26K80
- PIC18LF45K80
- PIC18LF46K80
- PIC18LF65K80
- PIC18LF66K80

This family combines the traditional advantages of all PIC18 microcontrollers – namely, high computational performance and a rich feature set – with an extremely competitive price point. These features make the PIC18F66K80 family a logical choice for many high-performance applications where price is a primary consideration.

### 1.1 Core Features

#### 1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F66K80 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 RC oscillator, power consumption during code execution can be reduced.
- **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.
- **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.
- **nanoWatt XLP:** An extra low-power BOR and low-power Watchdog timer

#### 1.1.2 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F66K80 family offer different oscillator options, allowing users a range of choices in developing application hardware. These include:

- External Resistor/Capacitor (RC); RA6 available
- External Resistor/Capacitor with Clock Out (RCIO)
- Three External Clock modes:
  - External Clock (EC); RA6 available
  - External Clock with Clock Out (ECIO)
  - External Crystal (XT, HS, LP)

- A Phase Lock Loop (PLL) frequency multiplier, available to the external oscillator modes which allows clock speeds of up to 64 MHz. PLL can also be used with the internal oscillator.
- An internal oscillator block that provides a 16 MHz clock ( $\pm 2\%$  accuracy) and an INTOSC source (approximately 31 kHz, stable over temperature and VDD)
  - Operates as HF-INTOSC or MF-INTOSC when block is selected for 16 MHz or 500 kHz
  - Frees the two oscillator pins for use as additional general purpose I/O

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, 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.

#### 1.1.3 MEMORY OPTIONS

The PIC18F66K80 family provides ample room for application code, from 32 Kbytes to 64 Kbytes of code space. The Flash cells for program memory are rated to last up to 10,000 erase/write cycles. Data retention without refresh is conservatively estimated to be greater than 20 years.

The Flash program memory is readable and writable. During normal operation, the PIC18F66K80 family also provides plenty of room for dynamic application data with up to 3.6 Kbytes of data RAM.

#### 1.1.4 EXTENDED INSTRUCTION SET

The PIC18F66K80 family implements the optional extension to the PIC18 instruction set, adding eight new instructions and an Indexed Addressing mode. Enabled as a device configuration option, the extension has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as 'C'.

# PIC18F66K80 FAMILY

---

## 1.1.5 EASY MIGRATION

Regardless of the memory size, all devices share the same rich set of peripherals, allowing for a smooth migration path as applications grow and evolve.

The consistent pinout scheme used throughout the entire family also aids in migrating to the next larger device. This is true when moving between the 28-pin, 40-pin, 44-pin and 64-pin members, or even jumping from smaller to larger memory devices.

The PIC18F66K80 family is also largely pin compatible with other PIC18 families, such as the PIC18F4580, PIC18F4680 and PIC18F8680 families of microcontrollers with an ECAN module. This allows a new dimension to the evolution of applications, allowing developers to select different price points within Microchip's PIC18 portfolio, while maintaining a similar feature set.

## 1.2 Other Special Features

- **Communications:** The PIC18F66K80 family incorporates a range of serial communication peripherals, including two Enhanced USARTs that support LIN/J2602, one Master SSP module capable of both SPI and I<sup>2</sup>C™ (Master and Slave) modes of operation and an Enhanced CAN module.
- **CCP Modules:** PIC18F66K80 family devices incorporate four Capture/Compare/PWM (CCP) modules. Up to four different time bases can be used to perform several different operations at once.
- **ECCP Modules:** The PIC18F66K80 family has one Enhanced CCP (ECCP) module to maximize flexibility in control applications:
  - Up to four different time bases for performing several different operations at once
  - Up to four PWM outputs
  - Other beneficial features, such as polarity selection, programmable dead time, auto-shutdown and restart, and Half-Bridge and Full-Bridge Output modes
- **12-Bit A/D Converter:** The PIC18F66K80 family has a differential A/D. It 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.

- **Charge Time Measurement Unit (CTMU):** The CTMU is a flexible analog module that provides accurate differential time measurement between pulse sources, as well as asynchronous pulse generation.

Together with other on-chip analog modules, the CTMU can precisely measure time, measure capacitance or relative changes in capacitance, or generate output pulses that are independent of the system clock.

- **LP Watchdog Timer (WDT):** This enhanced version incorporates a 22-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See [Section 31.0 "Electrical Characteristics"](#) for time-out periods.

## 1.3 Details on Individual Family Members

Devices in the PIC18F66K80 family are available in 28-pin, 40/44-pin and 64-pin packages. Block diagrams for each package are shown in [Figure 1-1](#), [Figure 1-2](#) and [Figure 1-3](#), respectively.

The devices are differentiated from each other in these ways:

- Flash Program Memory:
  - PIC18FX5K80 (PIC18F25K80, PIC18F45K80 and PIC18F45K80) – 32 Kbytes
  - PIC18FX6K80 (PIC18F26K80, PIC18F46K80 and PIC18F66K80) – 64 Kbytes
- I/O Ports:
  - PIC18F2XK80 (28-pin devices) – Three bidirectional ports
  - PIC18F4XK80 (40/44-pin devices) – Five bidirectional ports
  - PIC18F6XK80 (64-pin devices) – Seven bidirectional ports

All other features for devices in this family are identical. These are summarized in [Table 1-1](#), [Table 1-2](#) and [Table 1-3](#).

The pinouts for all devices are listed in [Table 1-4](#), [Table 1-5](#) and [Table 1-6](#).

# PIC18F66K80 FAMILY

---

**TABLE 1-1: DEVICE FEATURES FOR THE PIC18F2XK80 (28-PIN DEVICES)**

Features	PIC18F25K80	PIC18F26K80
Operating Frequency		DC – 64 MHz
Program Memory (Bytes)	32K	64K
Program Memory (Instructions)	16,384	32,768
Data Memory (Bytes)		3.6K
Interrupt Sources		31
I/O Ports		Ports A, B, C
Parallel Communications		Parallel Slave Port (PSP)
Timers		Five
Comparators		Two
CTMU		Yes
Capture/Compare/PWM (CCP) Modules		Four
Enhanced CCP (ECCP) Modules		One
Serial Communications		One MSSP and Two Enhanced USARTs (EUSART)
12-Bit Analog-to-Digital Module		Eight Input Channels
Resets (and Delays)		POR, BOR, RESET Instruction, Stack Full, Stack Underflow, MCLR, WDT (PWRT, OST)
Instruction Set		75 Instructions, 83 with Extended Instruction Set Enabled
Packages		28-Pin QFN-S, SOIC, SPDIP and SSOP

**TABLE 1-2: DEVICE FEATURES FOR THE PIC18F4XK80 (40/44-PIN DEVICES)**

Features	PIC18F45K80	PIC18F46K80
Operating Frequency		DC – 64 MHz
Program Memory (Bytes)	32K	64K
Program Memory (Instructions)	16,384	32,768
Data Memory (Bytes)		3.6K
Interrupt Sources		32
I/O Ports		Ports A, B, C, D, E
Parallel Communications		Parallel Slave Port (PSP)
Timers		Five
Comparators		Two
CTMU		Yes
Capture/Compare/PWM (CCP) Modules		Four
Enhanced CCP (ECCP) Modules		One
Serial Communications		One MSSP and Two Enhanced USARTs (EUSART)
12-Bit Analog-to-Digital Module		Eleven Input Channels
Resets (and Delays)		POR, BOR, RESET Instruction, Stack Full, Stack Underflow, MCLR, WDT (PWRT, OST)
Instruction Set		75 Instructions, 83 with Extended Instruction Set Enabled
Packages		40-Pin PDIP and 44-Pin QFN and TQFP

# PIC18F66K80 FAMILY

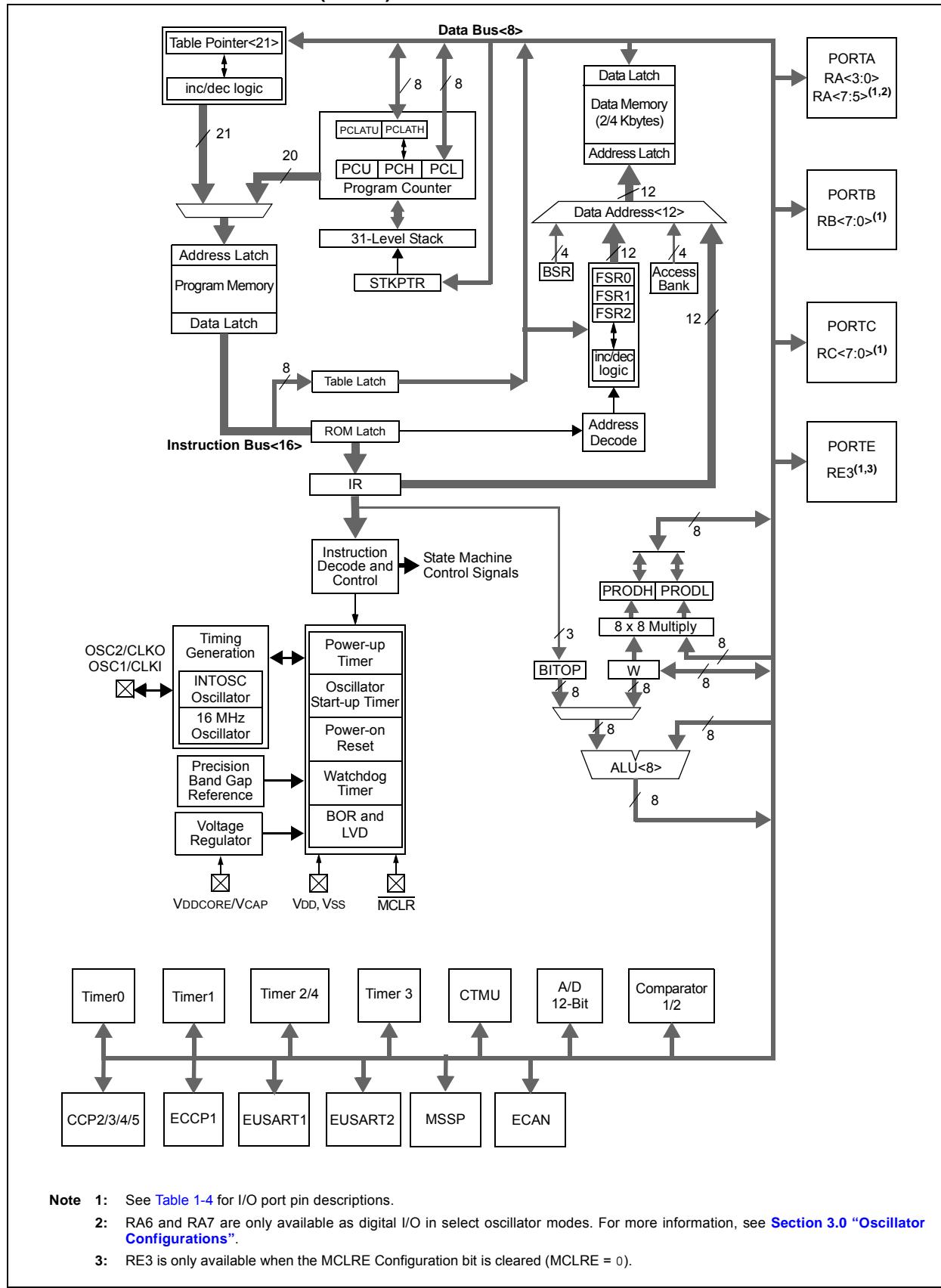
---

TABLE 1-3: DEVICE FEATURES FOR THE PIC18F6XK80 (64-PIN DEVICES)

Features	PIC18F65K80	PIC18F66K80
Operating Frequency	DC – 64 MHz	
Program Memory (Bytes)	32K	64K
Program Memory (Instructions)	16,384	32,768
Data Memory (Bytes)	3.6K	
Interrupt Sources	32	
I/O Ports	Ports A, B, C, D, E, F, G	
Parallel Communications	Parallel Slave Port (PSP)	
Timers	Five	
Comparators	Two	
CTMU	Yes	
Capture/Compare/PWM (CCP) Modules	Four	
Enhanced CCP (ECCP) Modules	One	
DSM	Yes	Yes
Serial Communications	One MSSP and Two Enhanced USARTs (EUSART)	
12-Bit Analog-to-Digital Module	Eleven Input Channels	
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow, MCLR, WDT (PWRT, OST)	
Instruction Set	75 Instructions, 83 with Extended Instruction Set Enabled	
Packages	64-Pin QFN and TQFP	

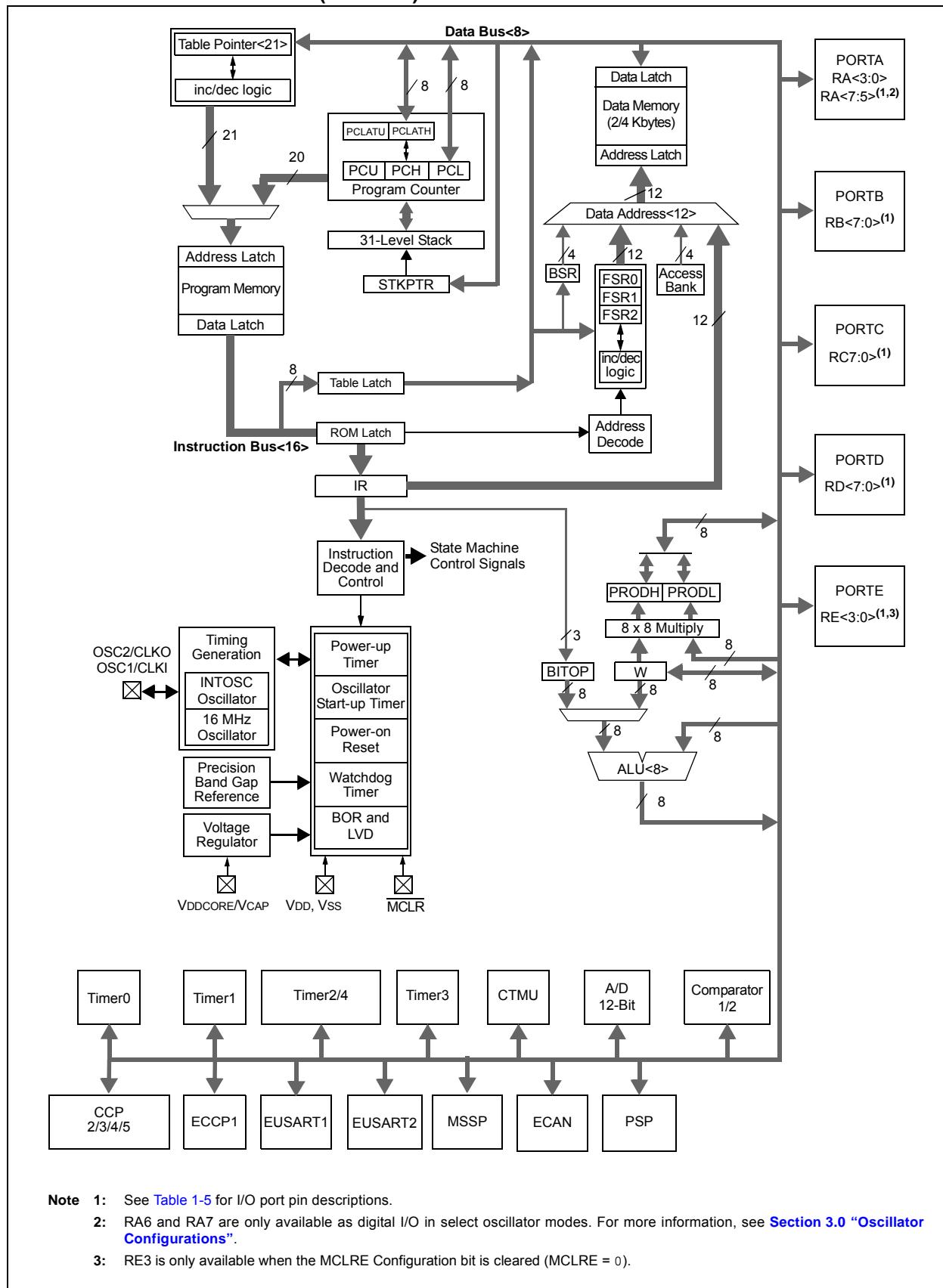
# PIC18F66K80 FAMILY

**FIGURE 1-1: PIC18F2XK80 (28-PIN) BLOCK DIAGRAM**



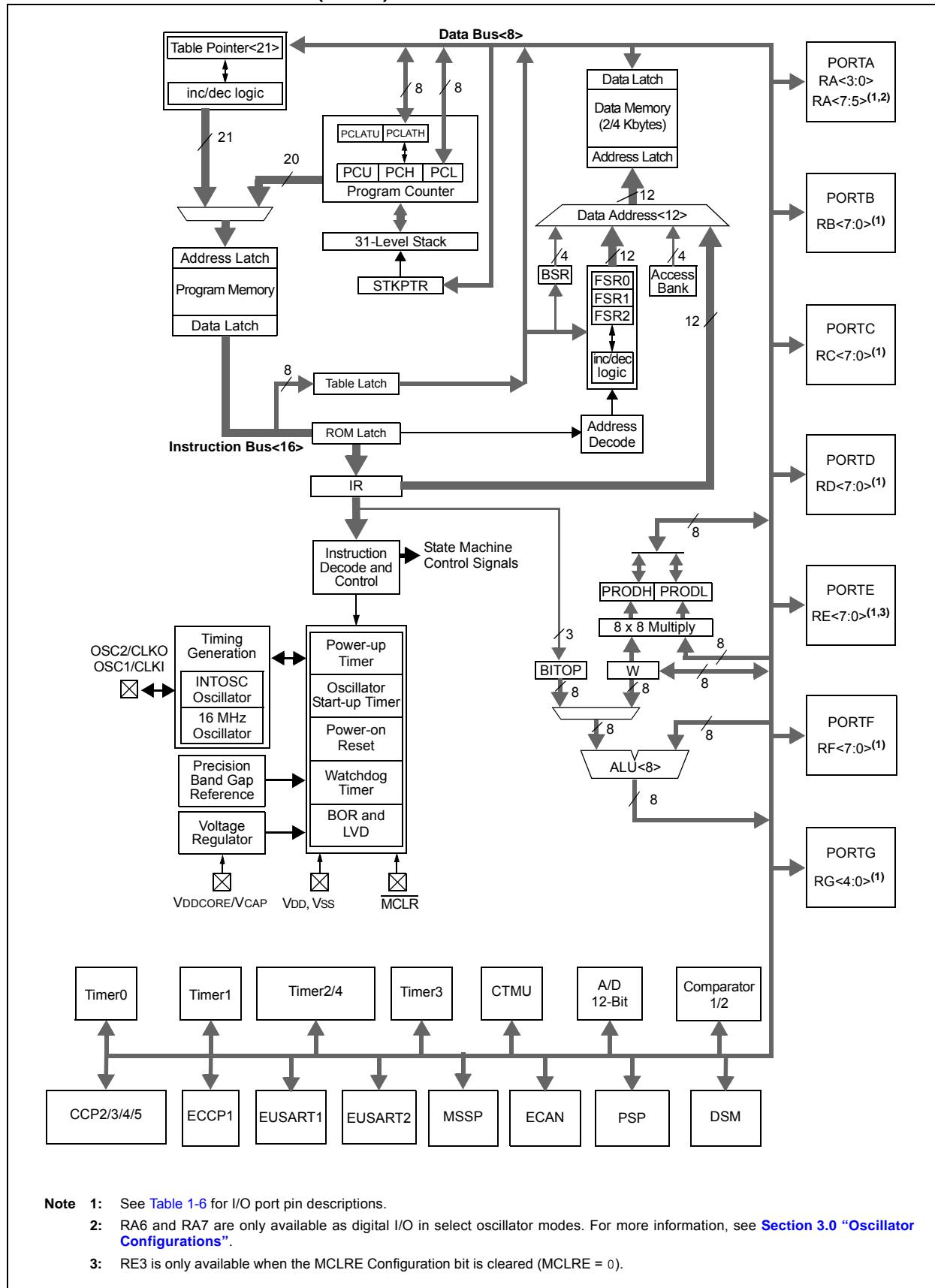
# PIC18F66K80 FAMILY

**FIGURE 1-2: PIC18F4XK80 (40/44-PIN) BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

**FIGURE 1-3: PIC18F6XK80 (64-PIN) BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

---

TABLE 1-4: PIC18F2XK80 I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	QFN	SSOP/ SPDIP /SOIC			
MCLR/RE3  MCLR  RE3	26	1	I	ST	Master Clear (input) or programming voltage (input). This pin is an active-low Reset to the device.
			I	ST	General purpose, input only pin.
OSC1/CLKIN/RA7  OSC1  CLKIN  RA7	6	9	I	ST	Oscillator crystal input.
			I	CMOS	External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
			I/O	ST/ CMOS	General purpose I/O pin.
OSC2/CLKOUT/RA6  OSC2  CLKOUT  RA6	7	10	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
			O	—	In certain oscillator modes, OSC2 pin outputs CLK0, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
			I/O	ST/ CMOS	General purpose I/O pin.

**Legend:** CMOS = CMOS compatible input or output      I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer  
 ST = Schmitt Trigger input with CMOS levels      Analog = Analog input  
 I = Input      O = Output  
 P = Power

# PIC18F66K80 FAMILY

TABLE 1-4: PIC18F2XK80 I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	QFN	SSOP/ SPDIP /SOIC			
RA0/CVREF/AN0/ULPWU	27	2	I/O	ST/ CMOS	PORTA is a bidirectional I/O port.
RA0			O	Analog	General purpose I/O pin.
CVREF			I	Analog	Comparator reference voltage output.
AN0			I	Analog	Analog Input 0.
ULPWU					Ultra Low-Power Wake-up input.
RA1/AN1	28	3	I/O	ST/ CMOS	Digital I/O.
RA1			I	Analog	Analog Input 1.
AN1					
RA2/VREF-/AN2	1	4	I/O	ST/ CMOS	Digital I/O.
RA2			I	Analog	A/D reference voltage (low) input.
VREF-			I	Analog	Analog Input 2.
AN2					
RA3/VREF+/AN3	2	5	I/O	ST/ CMOS	Digital I/O.
RA3			I	Analog	A/D reference voltage (high) input.
VREF+			I	Analog	Analog Input 3.
AN3					
RA5/AN4/C2INB/HLDIN/T1CKI/SS/CTMUI	4	7	I/O	ST/ CMOS	Digital I/O.
RA5			I	Analog	Analog Input 4.
AN4			I	Analog	Comparator 2 Input B.
C2INB			I	Analog	High/Low-Voltage Detect input.
HLDIN			I	ST	Timer1 clock input.
T1CKI			I	ST	SPI slave select input.
SS					
CTMUI					CTMU pulse generator charger for the C2INB.

**Legend:** CMOS = CMOS compatible input or output

$\text{I}^2\text{C}^{\text{TM}}$  =  $\text{I}^2\text{C}/\text{SMBus}$  input buffer

ST = Schmitt Trigger input with CMOS levels

Analog = Analog input

I = Input

O = Output

P = Power

# PIC18F66K80 FAMILY

---

TABLE 1-4: PIC18F2XK80 I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	QFN	SSOP/ SPDIP /SOIC			
RB0/AN10/C1INA/FLT0/INT0	18	21			PORTB is a bidirectional I/O port.
RB0			I/O	ST/ CMOS	Digital I/O.
AN10			I	Analog	Analog Input 10.
C1INA			I	Analog	Comparator 1 Input A.
FLT0			I	ST	Enhanced PWM Fault input for CCP1.
INT0			I	ST	External Interrupt 0.
RB1/AN8/C1INB/P1B/CTDIN/INT1	19	22			
RB1			I/O	ST/ CMOS	Digital I/O.
AN8			I	Analog	Analog Input 8.
C1INB			I	Analog	Comparator 1 Input B.
P1B			O	CMOS	Enhanced PWM1 Output B.
CTDIN			I	ST	CTMU pulse delay input.
INT1			I	ST	External Interrupt 1.
RB2/CANTX/C1OUT/P1C/CTED1/INT2	20	23			
RB2			I/O	ST/ CMOS	Digital I/O.
CANTX			O	CMOS	CAN bus TX.
C1OUT			O	CMOS	Comparator 1 output.
P1C			O	CMOS	Enhanced PWM1 Output C.
CTED1			I	ST	CTMU Edge 1 input.
INT2			I	ST	External Interrupt 2.
RB3/CANRX/C2OUT/P1D/CTED2/INT3	21	24			
RB3			I/O	ST/ CMOS	Digital I/O.
CANRX			I	ST	CAN bus RX.
C2OUT			O	CMOS	Comparator 2 output.
P1D			O	CMOS	Enhanced PWM1 Output D.
CTED2			I	ST	CTMU Edge 2 input.
INT3			I	ST	External Interrupt 3.

**Legend:** CMOS = CMOS compatible input or output      I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer  
 ST = Schmitt Trigger input with CMOS levels      Analog = Analog input  
 I = Input      O = Output  
 P = Power

# PIC18F66K80 FAMILY

TABLE 1-4: PIC18F2XK80 I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	QFN	SSOP/ SPDIP /SOIC			
RB4/AN9/C2INA/ECCP1/ P1A/CTPLS/KBI0	22	25	I/O	ST/ CMOS	Digital I/O.
RB4					
AN9			I	Analog	Analog Input 9.
C2INA			I	Analog	Comparator 2 Input A.
ECCP1			I/O	ST	Capture 1 input/Compare 1 output/PWM1 output.
P1A			O	CMOS	Enhanced PWM1 Output A.
CTPLS			O	ST	CTMU pulse generator output.
KBI0			I	ST	Interrupt-on-change pin.
RB5/T0CKI/T3CKI/CCP5/ KBI1	23	26	I/O	ST/ CMOS	Digital I/O.
RB5					
T0CKI			I	ST	Timer0 external clock input.
T3CKI			I	ST	Timer3 external clock input.
CCP5			I/O	ST/ CMOS	Capture 5 input/Compare 5 output/PWM5 output.
KBI1			I	ST	Interrupt-on-change pin.
RB6/PGC/TX2/CK2/KBI2	24	27	I/O	ST/ CMOS	Digital I/O.
RB6					
PGC			I	ST	In-Circuit Debugger and ICSP™ programming clock input pin.
TX2			O	CMOS	EUSART asynchronous transmit.
CK2			I/O	ST	EUSART synchronous clock. (See related RX2/DT2.)
KBI2			I	ST	Interrupt-on-change pin.
RB7/PGD/T3G/RX2/DT2/ KBI3	25	28	I/O	ST/ CMOS	Digital I/O.
RB7					
PGD			I/O	ST	In-Circuit Debugger and ICSP programming data pin.
T3G			I	ST	Timer3 external clock gate input.
RX2			I	ST	EUSART asynchronous receive.
DT2			I/O	ST	EUSART synchronous data. (See related TX2/CK2.)
KBI3			I	ST	Interrupt-on-change pin.

**Legend:** CMOS = CMOS compatible input or output

$\text{I}^2\text{C}^{\text{TM}}$  =  $\text{I}^2\text{C}/\text{SMBus}$  input buffer

ST = Schmitt Trigger input with CMOS levels

Analog = Analog input

I = Input

O = Output

P = Power

# PIC18F66K80 FAMILY

TABLE 1-4: PIC18F2XK80 I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	QFN	SSOP/ SPDIP /SOIC			
RC0/SOSCO/SCLKI	8	11	I/O	ST/ CMOS	PORTC is a bidirectional I/O port.
RC0					Digital I/O.
SOSCO			I	ST	Timer1 oscillator output.
SCLKI			I	ST	Digital SOSC input.
RC1/SOSCI	9	12	I/O	ST/ CMOS	Digital I/O.
RC1					
SOSCI			I	CMOS	SOSC oscillator input.
RC2/T1G/CCP2	10	13	I/O	ST/ CMOS	Digital I/O.
RC2					
T1G			I	ST	Timer1 external clock gate input.
CCP2			I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
RC3/REFO/SCL/SCK	11	14	I/O	ST/ CMOS	Digital I/O.
RC3					
REFO			O	—	Reference clock out.
SCL			I/O	I <sup>2</sup> C	Synchronous serial clock input/output for I <sup>2</sup> C mode.
SCK			I/O	ST	Synchronous serial clock input/output for SPI mode.
RC4/SDA/SDI	12	15	I/O	ST/ CMOS	Digital I/O.
RC4					
SDA			I/O	I <sup>2</sup> C	I <sup>2</sup> C data input/output.
SDI			I	ST	SPI data in.
RC5/SDO	13	16	I/O	ST/ CMOS	Digital I/O.
RC5					
SDO			O	CMOS	SPI data out.
RC6/CANTX/TX1/CK1/ CCP3	14	17	I/O	ST/ CMOS	Digital I/O.
RC6					
CANTX			O	CMOS	CAN bus TX.
TX1			O	CMOS	EUSART asynchronous transmit.
CK1			I/O	ST	EUSART synchronous clock. (See related RX1/DT1.)
CCP3			I/O	ST/ CMOS	Capture 3 input/Compare 3 output/PWM3 output.

**Legend:** CMOS = CMOS compatible input or output

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---

TABLE 1-4: PIC18F2XK80 I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	QFN	SSOP/ SPDIP /SOIC			
RC7/CANRX/RX1/DT1/ CCP4	15	18	I/O	ST/ CMOS	Digital I/O.
RC7			I	ST	CAN bus RX.
CANRX			I	ST	EUSART asynchronous receive.
RX1			I/O	ST	EUSART synchronous data. (See related TX2/CK2.)
DT1			I/O	ST	Capture 4 input/Compare 4 output/PWM4 output.
CCP4				CMOS	
Vss	5	8	P		Ground reference for logic and I/O pins.
Vss	16	19			Ground reference for logic and I/O pins.
VDDCORE/VCAP	3	6	P		External filter capacitor connection.
VDDCORE					External filter capacitor connection
VCAP					
VDD	17	20	P		Positive supply for logic and I/O pins.
VDD					

**Legend:** CMOS = CMOS compatible input or output      I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer  
 ST = Schmitt Trigger input with CMOS levels      Analog = Analog input  
 I = Input      O = Output  
 P = Power

# PIC18F66K80 FAMILY

---

TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
MCLR/RE3  MCLR  RE3	1	18	I	ST	Master Clear (input) or programming voltage (input). This pin is an active-low Reset to the device.
			I	ST	General purpose, input only pin.
OSC1/CLKIN/RA7  OSC1  CLKIN  RA7	13	30	I	ST	Oscillator crystal input.
			I	CMOS	External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
			I/O	ST/ CMOS	General purpose I/O pin.
OSC2/CLKOUT/RA6  OSC2  CLKOUT  RA6	14	31	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
			O	—	In certain oscillator modes, OSC2 pin outputs CLK0, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
			I/O	ST/ CMOS	General purpose I/O pin.

Legend: I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RA0/CVREF/AN0/ULPWU	2	19	I/O	ST/ CMOS	PORTA is a bidirectional I/O port.
RA0			O	Analog	General purpose I/O pin.
CVREF			I	Analog	Comparator reference voltage output.
AN0			I	Analog	Analog Input 0.
ULPWU					Ultra Low-Power Wake-up input.
RA1/AN1/C1INC	3	20	I/O	ST/ CMOS	Digital I/O.
RA1			I	Analog	Analog Input 1.
AN1			I	Analog	Comparator 1 Input C.
C1INC					
RA2/VREF-/AN2/C2INC	4	21	I/O	ST/ CMOS	Digital I/O.
RA2			I	Analog	A/D reference voltage (low) input.
VREF-			I	Analog	Analog Input 2.
AN2			I	Analog	Comparator 2 Input C.
C2INC					
RA3/VREF+/AN3	5	22	I/O	ST/ CMOS	Digital I/O.
RA3			I	Analog	A/D reference voltage (high) input.
VREF+			I	Analog	Analog Input 3.
AN3					
RA5/AN4/HLVDIN/T1CKI/ SS	7	24	I/O	ST/ CMOS	Digital I/O.
RA5			I	Analog	Analog Input 4.
AN4			I	Analog	High/Low-Voltage Detect input.
HLVDIN			I	ST	Timer1 clock input.
T1CKI			I	ST	SPI slave select input.
SS					

Legend:  $\text{I}^2\text{C}^\text{TM}$  =  $\text{I}^2\text{C}/\text{SMBus}$  input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RB0/AN10/FLT0/INT0 RB0	33	8	I/O	ST/ CMOS	PORTB is a bidirectional I/O port. Digital I/O.
AN10			I	Analog	Analog Input 10.
FLT0			I	ST	Enhanced PWM Fault input for ECCP1.
INT0			I	ST	External Interrupt 0.
RB1/AN8/CTDIN/INT1 RB1	34	9	I/O	ST/ CMOS	Digital I/O.
AN8			I	Analog	Analog Input 8.
CTDIN			I	ST	CTMU pulse delay input.
INT1			I	ST	External Interrupt 1.
RB2/CANTX/CTED1/ INT2 RB2	35	10	I/O	ST/ CMOS	Digital I/O.
CANTX			O	CMOS	CAN bus TX.
CTED1			I	ST	CTMU Edge 1 input.
INT2			I	ST	External Interrupt 2.
RB3/CANRX/CTED2/ INT3 RB3	36	11	I/O	ST/ CMOS	Digital I/O.
CANRX			I	ST	CAN bus RX.
CTED2			I	ST	CTMU Edge 2 input.
INT3			I	ST	External Interrupt 3.
RB4/AN9/CTPLS/KBI0 RB4	37	14	I/O	ST/ CMOS	Digital I/O.
AN9			I	Analog	Analog Input 9.
CTPLS			O	ST	CTMU pulse generator output.
KBI0			I	ST	Interrupt-on-change pin.
RB5/T0CKI/T3CKI/CCP5/ KBI1 RB5	38	15	I/O	ST/ CMOS	Digital I/O.
T0CKI			I	ST	Timer0 external clock input.
T3CKI			I	ST	Timer3 external clock input.
CCP5			I/O	ST	Capture 5 input/Compare 5 output/PWM5 output.
KBI1			I	ST	Interrupt-on-change pin.

Legend: I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---

**TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)**

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RB6/PGC/KBI2  RB6  PGC  KBI2	39	16	I/O	ST/ CMOS	Digital I/O.
			I	ST	In-Circuit Debugger and ICSP™ programming clock input pin.
			I	ST	Interrupt-on-change pin.
RB7/PGD/T3G/KBI3  RB7  PGD  T3G  KBI3	40	17	I/O	ST/ CMOS	Digital I/O.
			I/O	ST	In-Circuit Debugger and ICSP™ programming data pin.
			I	ST	Timer3 external clock gate input.
			I	ST	Interrupt-on-change pin.

**Legend:**  $i^2C$ ™ =  $i^2C$ /SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RC0/SOSCO/SCLKI RC0	15	32	I/O	ST/ CMOS	PORTC is a bidirectional I/O port. Digital I/O.
SOSCO			I	ST	SOSC oscillator output.
SCLKI			I	ST	Digital SOSC input.
RC1/SOSCI RC1	16	35	I/O	ST/ CMOS	Digital I/O.
SOSCI			I	CMOS	SOSC oscillator input.
RC2/T1G/CCP2 RC2	17	36	I/O	ST/ CMOS	Digital I/O.
T1G			I	ST	Timer1 external clock gate input.
CCP2			I/O	ST/ CMOS	Capture 2 input/Compare 2 output/PWM2 output.
RC3/REFO/SCL/SCK RC3	18	37	I/O	ST/ CMOS	Digital I/O.
REFO			O	CMOS	Reference clock out.
SCL			I/O	I <sup>2</sup> C	Synchronous serial clock input/output for I <sup>2</sup> C mode.
SCK			I/O	ST	Synchronous serial clock input/output for SPI mode.
RC4/SDA/SDI RC4	23	42	I/O	ST/ CMOS	Digital I/O.
SDA			I/O	I <sup>2</sup> C	I <sup>2</sup> C data input/output.
SDI			I	ST	SPI data in.
RC5/SDO RC5	24	43	I/O	ST/ CMOS	Digital I/O.
SDO			O	CMOS	SPI data out.
RC6/CANTX/TX1/CK1/ CCP3 RC6	25	44	I/O	ST/ CMOS	Digital I/O.
CANTX			O	CMOS	CAN bus TX.
TX1			O	CMOS	EUSART synchronous transmit.
CK1			I/O	ST	EUSART synchronous clock. (See related RX2/DT2.)
CCP3			I/O	ST	Capture 3 input/Compare 3 output/PWM3 output.

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---

**TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)**

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RC7/CANRX/RX1/DT1/ CCP4	26	1	I/O	ST/ CMOS	Digital I/O.
RC7			I	ST	CAN bus RX.
CANRX			I	ST	EUSART asynchronous receive.
RX1			I/O	ST	EUSART synchronous data. (See related TX2/CK2.)
DT1			I/O	ST	Capture 4 input/Compare 4 output/PWM4 output.
CCP4					

**Legend:**  $I^2C$ ™ =  $I^2C$ /SMBus input buffer

ST = Schmitt Trigger Input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RD0/C1INA/PSP0 RD0	19	38	I/O	ST/ CMOS	PORTD is a bidirectional I/O port. Digital I/O.
C1INA			I	Analog	Comparator 1 Input A.
PSP0			I/O	ST/ CMOS	Parallel Slave Port data.
RD1/C1INB/PSP1 RD1	20	39	I/O	ST/ CMOS	Digital I/O.
C1INB			I	Analog	Comparator 1 Input B.
PSP1			I/O	ST/ CMOS	Parallel Slave Port data.
RD2/C2INA/PSP2 RD2	21	40	I/O	ST/ CMOS	Digital I/O.
C2INA			I	Analog	Comparator 2 Input A.
PSP2			I/O	ST/ CMOS	Parallel Slave Port data.
RD3/C2INB/CTMUI/ PSP3 RD3	22	41	I/O	ST/ CMOS	Digital I/O.
C2INB			I	Analog	Comparator 2 Input B.
CTMUI			I/O	ST/ CMOS	CTMU pulse generator charger for the C2INB.
PSP3			I/O	ST/ CMOS	Parallel Slave Port data.
RD4/ECCP1/P1A/PSP4 RD4	27	2	I/O	ST/ CMOS	Digital I/O.
ECCP1			I/O	ST	Capture 1 input/Compare 1 output/PWM1 output.
P1A			O	CMOS	Enhanced PWM1 Output A.
PSP4			I/O	ST/ CMOS	Parallel Slave Port data.
RD5/P1B/PSP5 RD5	28	3	I/O	ST/ CMOS	Digital I/O.
P1B			O	CMOS	Enhanced PWM1 Output B.
PSP5			I/O	ST/ CMOS	Parallel Slave Port data.

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RD6/TX2/CK2/P1C/PSP6	29	4	I/O	ST/CMOS	Digital I/O.
			I	ST	EUSART asynchronous transmit.
			I/O	ST	EUSART synchronous clock. (See related RX2/DT2.)
			O	CMOS	Enhanced PWM1 Output C.
			I/O	ST/CMOS	Parallel Slave Port data.
RD7/RX2/DT2/P1D/PSP7	30	5	I/O	ST/CMOS	Digital I/O.
			I	ST	EUSART asynchronous receive.
			I/O	ST	EUSART synchronous data. (See related TX2/CK2.)
			O	CMOS	Enhanced PWM1 Output D.
			I/O	ST/CMOS	Parallel Slave Port data.
RE0/AN5/RD	8	25	I/O	ST/CMOS	Digital I/O.
			I	Analog	Analog Input 5.
			I	ST	Parallel Slave Port read strobe.
RE1/AN6/C1OUT/WR	9	26	I/O	ST/CMOS	Digital I/O.
			I	Analog	Analog Input 6.
			O	CMOS	Comparator 1 output.
			I	ST	Parallel Slave Port write strobe.
RE2/AN7/C2OUT/CS	10	27	I/O	ST/CMOS	Digital I/O.
			I	Analog	Analog Input 7.
			O	CMOS	Comparator 2 output.
			I	ST	Parallel Slave Port chip select.
RE3					See the MCLR/RE3 pin.

Legend:  $\text{I}^2\text{C}^{\text{TM}}$  =  $\text{I}^2\text{C}/\text{SMBus}$  input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---



---

TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
Vss Vss	12	29	P		Ground reference for logic and I/O pins.
Vss Vss	31	6			Ground reference for logic and I/O pins.
VDDCORE/VCAP	6	23	P		External filter capacitor connection
VDDCORE VCAP					External filter capacitor connection
VDD VDD	11	28	P		Positive supply for logic and I/O pins.
VDD VDD	32	7	P		Positive supply for logic and I/O pins.

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer      CMOS = CMOS compatible input or output  
 ST = Schmitt Trigger input with CMOS levels      Analog = Analog input  
 I = Input      O = Output  
 P = Power

# PIC18F66K80 FAMILY

---

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Num	Pin Type	Buffer Type	Description
MCLR/RE3 MCLR RE3	28	I	ST	Master Clear (input) or programming voltage (input). This pin is an active-low Reset to the device.
OSC1/CLKIN/RA7 OSC1 CLKIN RA7	46	I I I/O	ST CMOS ST/ CMOS	Oscillator crystal input. External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.) General purpose I/O pin.
OSC2/CLKOUT/RA6 OSC2 CLKOUT RA6	47	O O I/O	— — ST/ CMOS	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In certain oscillator modes, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. General purpose I/O pin.

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RA0/CVREF/AN0/ ULPWU RA0 CVREF AN0 ULPWU	29			PORTA is a bidirectional I/O port. General purpose I/O pin. Comparator reference voltage output. Analog Input 0. Ultra Low-Power Wake-up input.
RA1/AN1/C1INC RA1 AN1 C1INC	30	I/O	ST/ CMOS	Digital I/O. Analog Input 1. Comparator 1 Input C.
RA2/VREF-/AN2/C2INC RA2 VREF- AN2 C2INC	31	I/O	ST/ CMOS	Digital I/O. A/D reference voltage (low) input. Analog Input 2. Comparator 2 Input C.
RA3/VREF+/AN3 RA3 VREF+ AN3	32	I/O	ST/ CMOS	Digital I/O. A/D reference voltage (high) input. Analog Input 3.
RA5/AN4/HLDIN/ T1CKI/SS RA5 AN4 HLDIN T1CKI SS	34	I/O	ST/ CMOS	Digital I/O. Analog Input 4. High/Low-Voltage Detect input. Timer1 clock input. SPI slave select input.

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

| = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RB0/AN10/FLT0/INT0	13			PORTB is a bidirectional I/O port.
RB0		I/O	ST/ CMOS	Digital I/O.
AN10		I	Analog	Analog Input 10.
FLT0		I	ST	Enhanced PWM Fault input for ECCP1.
INT0		I	ST	External Interrupt 0.
RB1/AN8/CTDIN/INT1	14			
RB1		I/O	ST/ CMOS	Digital I/O.
AN8		I	Analog	Analog Input 8.
CTDIN		I	ST	CTMU pulse delay input.
INT1		I	ST	External Interrupt 1.
RB2/CANTX/CTED1/ INT2	15			
RB2		I/O	ST/ CMOS	Digital I/O.
CANTX		O	CMOS	CAN bus TX.
CTED1		I	ST	CTMU Edge 1 input.
INT2		I	ST	External Interrupt 2.
RB3/CANRX/CTED2/ INT3	16			
RB3		I/O	ST/ CMOS	Digital I/O.
CANRX		I	ST	CAN bus RX.
CTED2		I	ST	CTMU Edge 2 input.
INT3		I	ST	External Interrupt 3.
RB4/AN9/CTPLS/KBI0	20			
RB4		I/O	ST/ CMOS	Digital I/O.
AN9		I	Analog	Analog Input 9.
CTPLS		O	ST	CTMU pulse generator output.
KBI0		I	ST	Interrupt-on-change pin.
RB5/T0CKI/T3CKI/CCP5/ KBI1	21			
RB5		I/O	ST/ CMOS	Digital I/O.
T0CKI		I	ST	Timer0 external clock input.
T3CKI		I	ST	Timer3 external clock input.
CCP5		I/O	ST/ CMOS	Capture 5 input/Compare 5 output/PWM5 output.
KBI1		I	ST	Interrupt-on-change pin.

Legend:  $\text{I}^2\text{C}^{\text{TM}}$  =  $\text{I}^2\text{C}/\text{SMBus}$  input buffer

CMOS = CMOS compatible input or output

ST = Schmitt Trigger input with CMOS levels

Analog = Analog input

I = Input

O = Output

P = Power

# PIC18F66K80 FAMILY

---

---

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RB6/PGC/KBI2 RB6	22	I/O	ST/ CMOS	Digital I/O.
PGC		I	ST	In-Circuit Debugger and ICSP™ programming clock input pin.
KBI2		I	ST	Interrupt-on-change pin.
RB7/PGD/T3G/KBI3 RB7	23	I/O	ST/ CMOS	Digital I/O.
PGD		I/O	ST	In-Circuit Debugger and ICSP™ programming data pin.
T3G		I	ST	Timer3 external clock gate input.
KBI3		I	ST	Interrupt-on-change pin.

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer CMOS = CMOS compatible input or output  
ST = Schmitt Trigger input with CMOS levels Analog = Analog input  
I = Input O = Output  
P = Power

# PIC18F66K80 FAMILY

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RC0/SOSCO/SCLKI	48			PORTC is a bidirectional I/O port.
RC0		I/O	ST/ CMOS	Digital I/O.
SOSCO		I	ST	Timer1 oscillator output.
SCLKI		I	ST	Digital SOSC input.
RC1/SOSCI	49			
RC1		I/O	ST/ CMOS	Digital I/O.
SOSCI		I	CMOS	SOSC oscillator input.
RC2/T1G/CCP2	50			
RC2		I/O	ST/ CMOS	Digital I/O.
T1G		I	ST	Timer1 external clock gate input.
CCP2		I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
RC3/REFO/SCL/SCK	51			
RC3		I/O	ST/ CMOS	Digital I/O.
REFO		O	CMOS	Reference clock out.
SCL		I/O	I <sup>2</sup> C	Synchronous serial clock input/output for I <sup>2</sup> C mode.
SCK		I/O	ST	Synchronous serial clock input/output for SPI mode.
RC4/SDA/SDI	62			
RC4		I/O	ST/ CMOS	Digital I/O.
SDA		I/O	I <sup>2</sup> C	I <sup>2</sup> C data input/output.
SDI		I	ST	SPI data in.
RC5/SDO	63			
RC5		I/O	ST/ CMOS	Digital I/O.
SDO		O	CMOS	SPI data out.
RC6/CCP3	64			
RC6		I/O	ST/ CMOS	Digital I/O.
CCP3		I/O	ST/ CMOS	Capture 3 input/Compare 3 output/PWM3 output.
RC7/CCP4	1			
RC7		I/O	ST/ CMOS	Digital I/O.
CCP4		I/O	ST/ CMOS	Capture 4 input/Compare 4 output/PWM4 output.

Legend: I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RD0/C1INA/PSP0 RD0 C1INA PSP0	54	I/O I I/O	ST/ CMOS Analog ST/ CMOS	PORTD is a bidirectional I/O port. Digital I/O. Comparator 1 Input A. Parallel Slave Port data.
RD1/C1INB/PSP1 RD1 C1INB PSP1	55	I/O I I/O	ST/ CMOS Analog ST/ CMOS	Digital I/O. Comparator 1 Input B. Parallel Slave Port data.
RD2/C2INA/PSP2 RD2 C2INA PSP2	58	I/O I I/O	ST/ CMOS Analog ST/ CMOS	Digital I/O. Comparator 2 Input A. Parallel Slave Port data.
RD3/C2INB/CTMUI/ PSP3 RD3 C2INB CTMUI PSP3	59	I/O I O I/O	ST/ CMOS Analog CMOS ST/ CMOS	Digital I/O. Comparator 2 Input B. CTMU pulse generator charger for the C2INB. Parallel Slave Port data.
RD4/ECCP1/P1A/PSP4 RD4 ECCP1 P1A PSP4	2	I/O I/O O I/O	ST/ CMOS ST CMOS ST/ CMOS	Digital I/O. Capture 1 input/Compare 1 output/PWM1 output. Enhanced PWM1 Output A. Parallel Slave Port data.
RD5/P1B/PSP5 RD5 P1B PSP5	3	I/O O I/O	ST/ CMOS CMOS ST/ CMOS	Digital I/O. Enhanced PWM1 Output B. Parallel Slave Port data.

Legend: I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---

**TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)**

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RD6/P1C/PSP6 RD6	4	I/O	ST/ CMOS	Digital I/O.
P1C		O	CMOS	Enhanced PWM1 Output C.
PSP6		I/O	ST/ CMOS	Parallel Slave Port data.
RD7/P1D/PSP7 RD7	5	I/O	ST/ CMOS	Digital I/O.
P1D		O	CMOS	Enhanced PWM1 Output D.
PSP7		I/O	ST/ CMOS	Parallel Slave Port data.

**Legend:**  $I^2C^{\text{TM}}$  =  $I^2C$ /SMBus input buffer      CMOS = CMOS compatible input or output  
 ST = Schmitt Trigger input with CMOS levels      Analog = Analog input  
 I = Input      O = Output  
 P = Power

# PIC18F66K80 FAMILY

---

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RE0/AN5/RD RE0 AN5 RD	37	I/O	ST/ CMOS	PORTE is a bidirectional I/O port. Digital I/O.
RE1/AN6/C1OUT/WR RE1 AN6 C1OUT WR	38	I O	Analog ST	Analog Input 5. Parallel Slave Port read strobe.
RE2/AN7/C2OUT/CS RE2 AN7 C2OUT CS	39	I/O	ST/ CMOS	Digital I/O. Analog Input 6. Comparator 1 output. Parallel Slave Port write strobe.
RE3				See the MCLR/RE3 pin.
RE4/CANRX RE4 CANRX	27	I/O	ST/ CMOS	Digital I/O. CAN bus RX.
RE5/CANTX RE5 CANTX	24	I O	ST/ CMOS	Digital I/O. CAN bus TX.
RE6/RX2/DT2 RE6 RX2 DT2	60	I/O	ST/ CMOS	Digital I/O. EUSART asynchronous receive. EUSART synchronous data. (See related TX2/CK2.)
RE7/TX2/CK2 RE7 TX2 CK2	61	I/O	ST/ CMOS	Digital I/O. EUSART asynchronous transmit. EUSART synchronous clock. (See related RX2/DT2.)

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer CMOS = CMOS compatible input or output  
 ST = Schmitt Trigger input with CMOS levels Analog = Analog input  
 I = Input O = Output  
 P = Power

# PIC18F66K80 FAMILY

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
				PORTF is a bidirectional I/O port.
RF0/MDMIN	17	I/O	ST/CMOS	Digital I/O.
RF0		I	CMOS	Modulator source input.
MDMIN				
RF1	19	I/O	ST/CMOS	Digital I/O.
RF1		I	CMOS	
RF2/MDCIN1	35	I/O	ST/CMOS	Digital I/O.
RF2		I	ST	Modulator Carrier Input 1.
MDCIN1				
RF3	36	I/O	ST/CMOS	Digital I/O.
RF3		I	CMOS	
RF4/MDCIN2	44	I/O	ST/CMOS	Digital I/O.
RF4		I	ST	Modulator Carrier Input 2.
MDCIN2				
RF5	45	I/O	ST/CMOS	Digital I/O.
RF5		I	CMOS	
RF6/MDOUUT	52	I/O	ST/CMOS	Digital I/O.
RF6		I	CMOS	
MDOUUT		O	CMOS	Modulator output.
RF7	53	I/O	ST/CMOS	Digital I/O.
RF7		I	CMOS	

**Legend:**  $\text{I}^2\text{C}^{\text{TM}}$  =  $\text{I}^2\text{C}/\text{SMBus}$  input buffer

ST = Schmitt Trigger input with CMOS levels

I = Input

P = Power

CMOS = CMOS compatible input or output

Analog = Analog input

O = Output

# PIC18F66K80 FAMILY

---

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
RG0/RX1/DT1 RG0 RX1 DT1	6	I/O	ST/ CMOS	PORTG is a bidirectional I/O port. Digital I/O.
RG1/CANTX2 RG1 CANTX2	7	I/O	ST/ CMOS	EUSART asynchronous receive. EUSART synchronous data. (See related TX2/CK2.) Digital I/O.
RG2/T3CKI RG2 T3CKI	11	I/O	ST/ CMOS	CAN bus complimentary transmit output or CAN bus time clock. Digital I/O.
RG3/TX1/CK1 RG3 TX1 CK1	12	I/O	ST/ CMOS	Timer3 clock input. Digital I/O.
RG4/T0CKI RG4 T0CKI	18	I/O	ST/ CMOS	EUSART asynchronous transmit. EUSART synchronous clock. (See related RX2/DT2.) Digital I/O.
		I	ST	Timer0 external clock input.

**Legend:**  $I^2C^{\text{TM}}$  =  $I^2C$ /SMBus input buffer      CMOS = CMOS compatible input or output  
 ST = Schmitt Trigger input with CMOS levels      Analog = Analog input  
 I = Input      O = Output  
 P = Power

# PIC18F66K80 FAMILY

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Num	Pin Type	Buffer Type	Description
Vss	8		P	Ground reference for logic and I/O pins.
Vss	26		P	Ground reference for logic and I/O pins.
AVSS	42		P	Ground reference for analog modules.
AVSS	43		P	Ground reference for logic and I/O pins.
Vss	56		P	Ground reference for logic and I/O pins.
AVDD	9		P	Positive supply for analog modules.
AVDD	10		P	Positive supply for logic and I/O pins.
VDD	25		P	Positive supply for logic and I/O pins.
VDD	33		P	External filter capacitor connection.
VDDCORE/VCAP				External filter capacitor connection.
VDDCORE				
VCAP				
AVDD	40		P	Positive supply for analog modules.
AVDD	41		P	Positive supply for logic and I/O pins.
VDD	57		P	Positive supply for logic and I/O pins.

**Legend:** I<sup>2</sup>C™ = I<sup>2</sup>C/SMBus input buffer

CMOS = CMOS compatible input or output

ST = Schmitt Trigger input with CMOS levels

Analog = Analog input

I = Input

O = Output

P = Power

# PIC18F66K80 FAMILY

---

---

**NOTES:**

## 2.0 GUIDELINES FOR GETTING STARTED WITH PIC18FXXKXX MICROCONTROLLERS

### 2.1 Basic Connection Requirements

Getting started with the PIC18F66K80 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”](#))
- ENVREG (if implemented) and VCAP/VDDCORE pins  
(see [Section 2.4 “Voltage Regulator Pins \(ENVREG and VCAP/VDDCORE\)”](#))

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 2.5 “ICSP Pins”](#))
- OSCI and OSCO pins when an external oscillator source is used  
(see [Section 2.6 “External Oscillator Pins”](#))

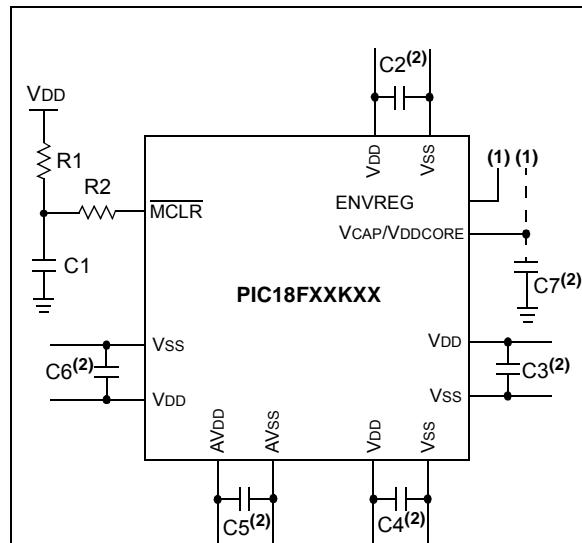
Additionally, the following pins may be required:

- 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**



**Key (all values are recommendations):**

C1 through C6: 0.1 µF, 20V ceramic

R1: 10 kΩ

R2: 100Ω to 470Ω

**Note 1:** See [Section 2.4 “Voltage Regulator Pins \(ENVREG and VCAP/VDDCORE\)”](#) for explanation of ENVREG pin connections.

**2:** The example shown is for a PIC18F device with five VDD/Vss and AVDD/AVss pairs. Other devices may have more or less pairs; adjust the number of decoupling capacitors appropriately.

# PIC18F66K80 FAMILY

## 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  $0.1\ \mu\text{F}$  ( $100\ \text{nF}$ ), 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  $47\ \mu\text{F}$ .

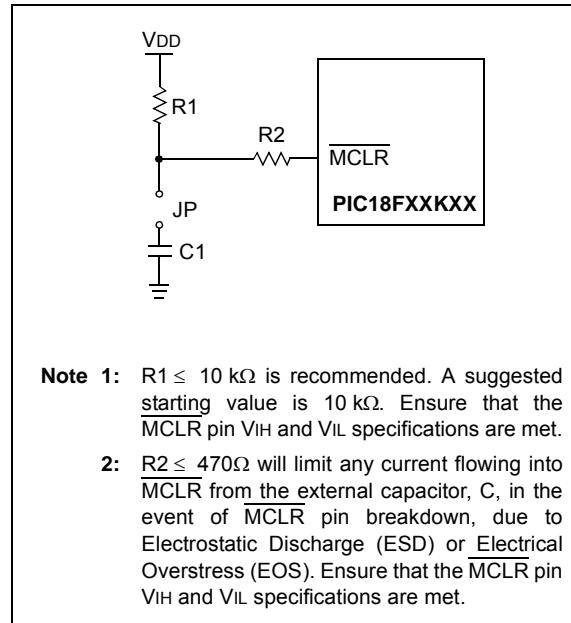
## 2.3 Master Clear (MCLR) Pin

The 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 VDD 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 MCLR pin. Consequently, specific voltage levels ( $\text{VIH}$  and  $\text{VIL}$ ) 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 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 MCLR pin should be placed within 0.25 inch (6 mm) of the pin.

**FIGURE 2-2: EXAMPLE OF MCLR PIN CONNECTIONS**



## 2.4 Voltage Regulator Pins (ENVREG and VCAP/VDDCORE)

The on-chip voltage regulator enable pin, ENVREG, must always be connected directly to either a supply voltage or to ground. Tying ENVREG to VDD enables the regulator, while tying it to ground disables the regulator. Refer to [Section 28.3 “On-Chip Voltage Regulator”](#) for details on connecting and using the on-chip regulator.

When the regulator is enabled, a low-ESR ( $< 5\Omega$ ) capacitor is required on the VCAP/VDDCORE pin to stabilize the voltage regulator output voltage. The VCAP/VDDCORE pin must not be connected to VDD and must use a capacitor of  $10 \mu F$  connected to ground. The type can be ceramic or tantalum. Suitable examples of capacitors are shown in [Table 2-1](#). Capacitors with equivalent specifications can be used.

Designers may use [Figure 2-3](#) to evaluate ESR equivalence of candidate devices.

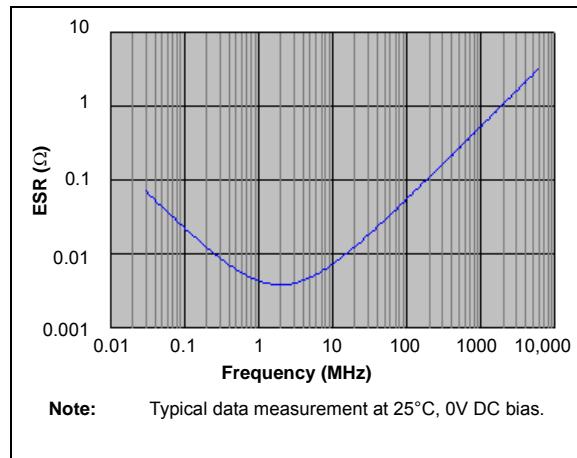
It is recommended that the trace length not exceed 0.25 inch (6 mm). Refer to [Section 31.0 “Electrical Characteristics”](#) for additional information.

When the regulator is disabled, the VCAP/VDDCORE pin must be tied to a voltage supply at the VDDCORE level. Refer to [Section 31.0 “Electrical Characteristics”](#) for information on VDD and VDDCORE.

Some PIC18FXXKXX families, or some devices within a family, do not provide the option of enabling or disabling the on-chip voltage regulator:

- Some devices (with the name, PIC18LFXXKXX) permanently disable the voltage regulator. These devices do not have the ENVREG pin and require a  $0.1 \mu F$  capacitor on the VCAP/VDDCORE pin. The VDD level of these devices must comply with the “voltage regulator disabled” specification for Parameter D001, in [Section 31.0 “Electrical Characteristics”](#).
- Some devices permanently enable the voltage regulator. These devices also do not have the ENVREG pin. The  $10 \mu F$  capacitor is still required on the VCAP/VDDCORE pin.

**FIGURE 2-3: FREQUENCY vs. ESR PERFORMANCE FOR SUGGESTED VCAP**



**TABLE 2-1: SUITABLE CAPACITOR EQUIVALENTS**

Make	Part #	Nominal Capacitance	Base Tolerance	Rated Voltage	Temp. Range
TDK	C3216X7R1C106K	10 $\mu F$	$\pm 10\%$	16V	-55 to 125°C
TDK	C3216X5R1C106K	10 $\mu F$	$\pm 10\%$	16V	-55 to 85°C
Panasonic	ECJ-3YX1C106K	10 $\mu F$	$\pm 10\%$	16V	-55 to 125°C
Panasonic	ECJ-4YB1C106K	10 $\mu F$	$\pm 10\%$	16V	-55 to 85°C
Murata	GRM32DR71C106KA01L	10 $\mu F$	$\pm 10\%$	16V	-55 to 125°C
Murata	GRM31CR61C106KC31L	10 $\mu F$	$\pm 10\%$	16V	-55 to 85°C

# PIC18F66K80 FAMILY

## 2.4.1 CONSIDERATIONS FOR CERAMIC CAPACITORS

In recent years, large value, low-voltage, surface-mount ceramic capacitors have become very cost effective in sizes up to a few tens of microfarad. The low-ESR, small physical size and other properties make ceramic capacitors very attractive in many types of applications.

Ceramic capacitors are suitable for use with the internal voltage regulator of this microcontroller. However, some care is needed in selecting the capacitor to ensure that it maintains sufficient capacitance over the intended operating range of the application.

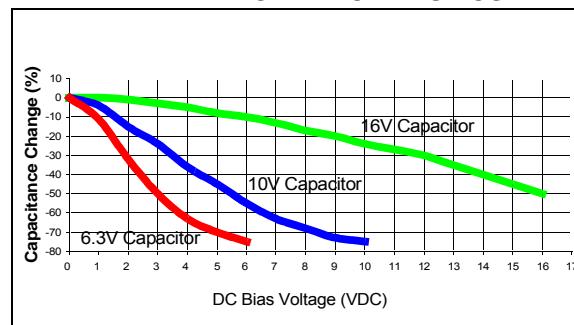
Typical low-cost, 10  $\mu\text{F}$  ceramic capacitors are available in X5R, X7R and Y5V dielectric ratings (other types are also available, but are less common). The initial tolerance specifications for these types of capacitors are often specified as  $\pm 10\%$  to  $\pm 20\%$  (X5R and X7R), or  $-20\%/+80\%$  (Y5V). However, the effective capacitance that these capacitors provide in an application circuit will also vary based on additional factors, such as the applied DC bias voltage and the temperature. The total in-circuit tolerance is, therefore, much wider than the initial tolerance specification.

The X5R and X7R capacitors typically exhibit satisfactory temperature stability (ex:  $\pm 15\%$  over a wide temperature range, but consult the manufacturer's data sheets for exact specifications). However, Y5V capacitors typically have extreme temperature tolerance specifications of  $+22\%/-82\%$ . Due to the extreme temperature tolerance, a 10  $\mu\text{F}$  nominal rated Y5V type capacitor may not deliver enough total capacitance to meet minimum internal voltage regulator stability and transient response requirements. Therefore, Y5V capacitors are not recommended for use with the internal regulator if the application must operate over a wide temperature range.

In addition to temperature tolerance, the effective capacitance of large value ceramic capacitors can vary substantially, based on the amount of DC voltage applied to the capacitor. This effect can be very significant, but is often overlooked or is not always documented.

A typical DC bias voltage vs. capacitance graph for X7R type and Y5V type capacitors is shown in Figure 2-4.

**FIGURE 2-4: DC BIAS VOLTAGE vs. CAPACITANCE CHARACTERISTICS**



When selecting a ceramic capacitor to be used with the internal voltage regulator, it is suggested to select a high-voltage rating, so that the operating voltage is a small percentage of the maximum rated capacitor voltage. For example, choose a ceramic capacitor rated at 16V for the 2.5V core voltage. Suggested capacitors are shown in Table 2-1.

## 2.5 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 100 $\Omega$ .

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 (VIH) and input low (VIL) 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 30.0 "Development Support"](#).

## 2.6 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).

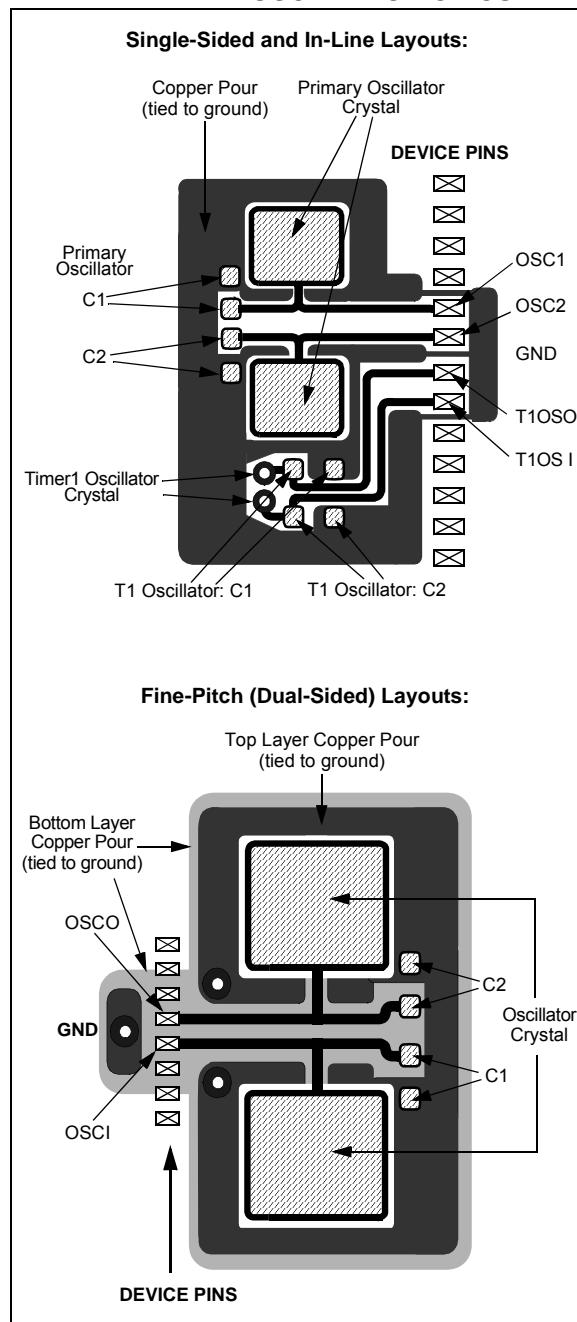
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site ([www.microchip.com](http://www.microchip.com)):

- [AN826, “Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices”](#)
- [AN849, “Basic PICmicro® Oscillator Design”](#)
- [AN943, “Practical PICmicro® Oscillator Analysis and Design”](#)
- [AN949, “Making Your Oscillator Work”](#)

## 2.7 Unused I/Os

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

**FIGURE 2-5: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT**



# **PIC18F66K80 FAMILY**

---

---

## **NOTES:**

## 3.0 OSCILLATOR CONFIGURATIONS

### 3.1 Oscillator Types

The PIC18F66K80 family of devices can be operated in the following oscillator modes:

- EC External Clock, RA6 Available
- ECIO External Clock, Clock Out RA6 (Fosc/4 on RA6)
- HS High-Speed Crystal/Resonator
- XT Crystal/Resonator
- LP Low-Power Crystal
- RC External Resistor/Capacitor, RA6 Available
- RCIO External Resistor/Capacitor, Clock Out RA6 (Fosc/4 on RA6)
- INTIO2 Internal Oscillator with I/O on RA6 and RA7
- INTIO1 Internal Oscillator with Fosc/4 Output on RA6 and I/O on RA7

There is also an option for running the 4xPLL on any of the clock sources in the input frequency range of 4 to 16 MHz.

The PLL is enabled by setting the PLLCFG bit (CONFIG1H<4>) or the PLLEN bit (OSCTUNE<6>).

For the EC and HS modes, the PLLEN (software) or PLLCFG (CONFIG1H<4>) bit can be used to enable the PLL.

For the INTIOx modes (HF-INTOSC):

- Only the PLLEN can enable the PLL (PLLCFG is ignored).
- When the oscillator is configured for the internal oscillator ( $\text{FOSC}<3:0> = 100x$ ), the PLL can be enabled only when the HF-INTOSC frequency is 4, 8 or 16 MHz.

When the RA6 and RA7 pins are not used for an oscillator function or CLKOUT function, they are available as general purpose I/Os.

To optimize power consumption when using EC/HS/XT/LP/RC as the primary oscillator, the frequency input range can be configured to yield an optimized power bias:

- Low-Power Bias – External frequency less than 160 kHz
- Medium Power Bias – External frequency between 160 kHz and 16 MHz
- High-Power Bias – External frequency greater than 16 MHz

All of these modes are selected by the user by programming the FOSC<3:0> Configuration bits (CONFIG1H<3:0>). In addition, PIC18F66K80 family devices can switch between different clock sources, either under software control, or under certain conditions, automatically. This allows for additional power savings by managing device clock speed in real time without resetting the application. The clock sources for the PIC18F66K80 family of devices are shown in [Figure 3-1](#).

For the HS and EC mode, there are additional power modes of operation, depending on the frequency of operation.

HS1 is the Medium Power mode with a frequency range of 4 MHz to 16 MHz. HS2 is the High-Power mode, where the oscillator frequency can go from 16 MHz to 25 MHz. HS1 and HS2 are achieved by setting the CONFIG1H<3:0> bits correctly. (For details, see [Register 28-2 on Page 460](#).)

EC mode has these modes of operation:

- EC1 – For low power with a frequency range up to 160 kHz
- EC2 – Medium power with a frequency range of 160 kHz to 16 MHz
- EC3 – High power with a frequency range of 16 MHz to 64 MHz

EC1, EC2 and EC3 are achieved by setting the CONFIG1H<3:0> correctly. (For details, see [Register 28-2 on Page 460](#).)

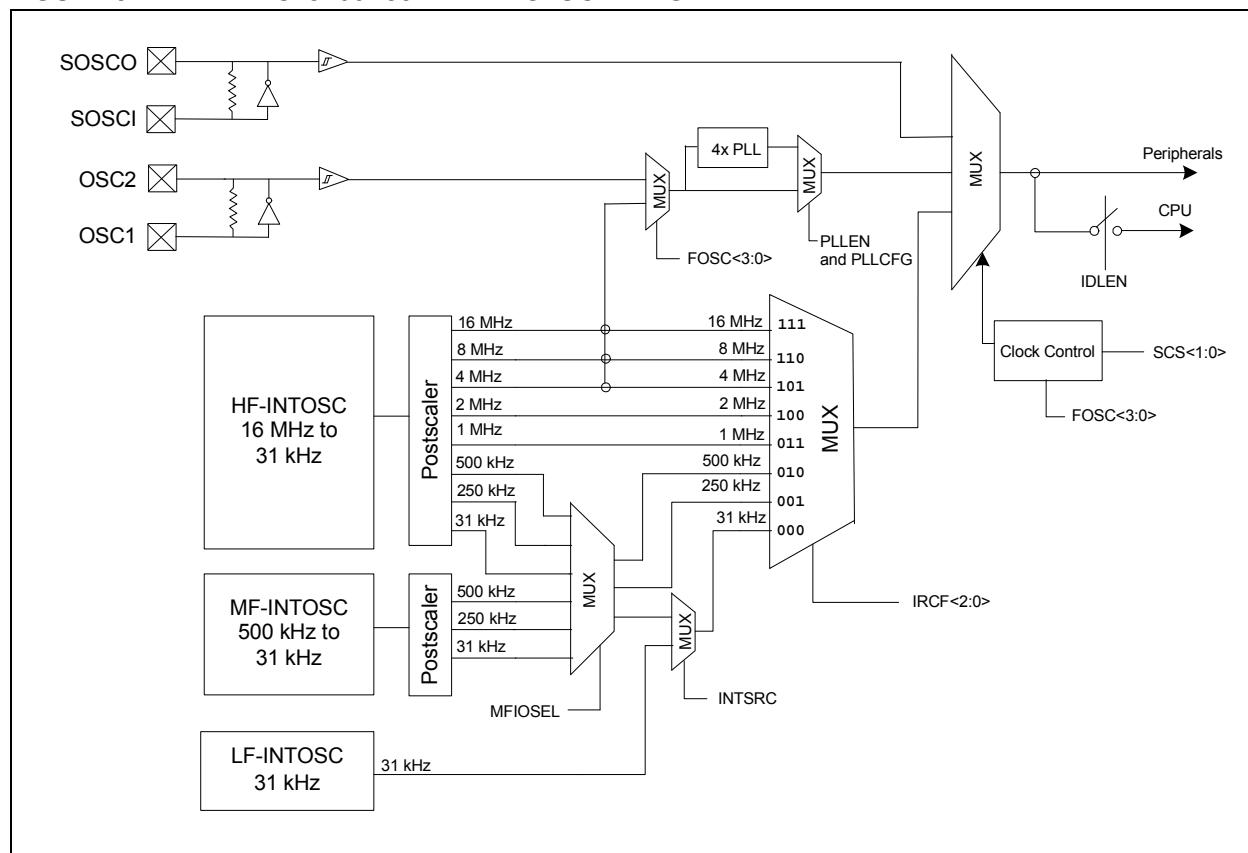
[Table 3-1](#) shows the HS and EC modes' frequency range and FOSC<3:0> settings.

# PIC18F66K80 FAMILY

TABLE 3-1: HS, EC, XT, LP AND RC MODES: RANGES AND SETTINGS

Mode	Frequency Range	FOSC<3:0> Setting
EC1 (low power) (EC1 & EC1IO)	DC-160 kHz	1101
		1100
EC2 (medium power) (EC2 & EC2IO)	160 kHz-16 MHz	1011
		1010
EC3 (high power) (EC3 & EC3IO)	16 MHz-64 MHz	0101
		0100
HS1 (medium power)	4 MHz-16 MHz	0011
HS2 (high power)	16 MHz-25 MHz	0010
XT	100 kHz-4 MHz	0001
LP	31.25 kHz	0000
RC (External)	0-4 MHz	001x
INTIO	32 kHz-16 MHz	100x (and OSCCON, OSCCON2)

FIGURE 3-1: PIC18F66K80 FAMILY CLOCK DIAGRAM



## 3.2 Control Registers

The OSCCON register ([Register 3-1](#)) controls the main aspects of the device clock's operation. It selects the oscillator type to be used, which of the power-managed modes to invoke and the output frequency of the INTOSC source. It also provides status on the oscillators.

The OSCTUNE register ([Register 3-3](#)) controls the tuning and operation of the internal oscillator block. It also implements the PLL bit which controls the operation of the Phase Locked Loop (PLL) (see [Section 3.5.3 "PLL Frequency Multiplier"](#)).

**REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER**

R/W-0	R/W-1	R/W-1	R/W-0	R <sup>(1)</sup>	R-0	R/W-0	R/W-0
IDLEN	IRCF2 <sup>(2)</sup>	IRCF1 <sup>(2)</sup>	IRCF0 <sup>(2)</sup>	OSTS	HFOFS	SCS1 <sup>(4)</sup>	SCS0 <sup>(4)</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	<b>IDLEN:</b> Idle Enable bit 1 = Device enters an Idle mode when a SLEEP instruction is executed 0 = Device enters Sleep mode when a SLEEP instruction is executed
bit 6-4	<b>IRCF&lt;2:0&gt;:</b> Internal Oscillator Frequency Select bits <sup>(2)</sup> 111 = HF-INTOSC output frequency is used (16 MHz) 110 = HF-INTOSC/2 output frequency is used (8 MHz, default) 101 = HF-INTOSC/4 output frequency is used (4 MHz) 100 = HF-INTOSC/8 output frequency is used (2 MHz) 011 = HF-INTOSC/16 output frequency is used (1 MHz) <u>If INTSRC = 0 and MFIOSEL = 0<sup>(3,5)</sup></u> 010 = HF-INTOSC/32 output frequency is used (500 kHz) 001 = HF-INTOSC/64 output frequency is used (250 kHz) 000 = LF-INTOSC output frequency is used (31.25 kHz) <sup>(6)</sup> <u>If INTSRC = 1 and MFIOSEL = 0<sup>(3,5)</sup></u> 010 = HF-INTOSC/32 output frequency is used (500 kHz) 001 = HF-INTOSC/64 output frequency is used (250 kHz) 000 = HF-INTOSC/512 output frequency is used (31.25 kHz) <u>If INTSRC = 0 and MFIOSEL = 1<sup>(3,5)</sup></u> 010 = MF-INTOSC output frequency is used (500 kHz) 001 = MF-INTOSC/2 output frequency is used (250 kHz) 000 = LF-INTOSC output frequency is used (31.25 kHz) <sup>(6)</sup> <u>If INTSRC = 1 and MFIOSEL = 1<sup>(3,5)</sup></u> 010 = MF-INTOSC output frequency is used (500 kHz) 001 = MF-INTOSC/2 output frequency is used (250 kHz) 000 = MF-INTOSC/16 output frequency is used (31.25 kHz)
bit 3	<b>OSTS:</b> Oscillator Start-up Timer Time-out Status bit <sup>(1)</sup> 1 = Oscillator Start-up Timer (OST) time-out has expired; primary oscillator is running, as defined by FOSC<3:0> 0 = Oscillator Start-up Timer (OST) time-out is running; primary oscillator is not ready – device is running from internal oscillator (HF-INTOSC, MF-INTOSC or LF-INTOSC)

- Note 1:** The Reset state depends on the state of the IESO Configuration bit (CONFIG1H<7>).
- 2:** Modifying these bits will cause an immediate clock frequency switch if the internal oscillator is providing the device clocks.
- 3:** The source is selected by the INTSRC bit (OSCTUNE<7>).
- 4:** Modifying these bits will cause an immediate clock source switch.
- 5:** INTSRC = OSCTUNE<7> and MFIOSEL = OSCCON2<0>.
- 6:** This is the lowest power option for an internal source.

# PIC18F66K80 FAMILY

## REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER (CONTINUED)

bit 2	<b>HFIOFS:</b> HF-INTOSC Frequency Stable bit 1 = HF-INTOSC oscillator frequency is stable 0 = HF-INTOSC oscillator frequency is not stable
bit 1-0	<b>SCS&lt;1:0&gt;:</b> System Clock Select bits <sup>(4)</sup> 1x = Internal oscillator block (LF-INTOSC, MF-INTOSC or HF-INTOSC) 01 = SOSC oscillator 00 = Default primary oscillator (OSC1/OSC2 or HF-INTOSC with or without PLL; defined by the FOSC<3:0> Configuration bits, CONFIG1H<3:0>)

- Note 1:** The Reset state depends on the state of the IESO Configuration bit (CONFIG1H<7>).
- 2:** Modifying these bits will cause an immediate clock frequency switch if the internal oscillator is providing the device clocks.
- 3:** The source is selected by the INTSRC bit (OSCTUNE<7>).
- 4:** Modifying these bits will cause an immediate clock source switch.
- 5:** INTSRC = OSCTUNE<7> and MFIOSEL = OSCCON2<0>.
- 6:** This is the lowest power option for an internal source.

## REGISTER 3-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

U-0	R-0	U-0	R/W-1	R/W-0	U-0	R-x	R/W-0
—	SOSCRUN	—	SOSCDRV <sup>(1)</sup>	SOSCGO	—	MFIOFS	MFIOSEL
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	<b>Unimplemented:</b> Read as '0'
bit 6	<b>SOSCRUN:</b> SOSC Run Status bit 1 = System clock comes from a secondary SOSC 0 = System clock comes from an oscillator other than SOSC
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>SOSCDRV:</b> Secondary Oscillator Drive Control bit <sup>(1)</sup> 1 = High-power SOSC circuit is selected 0 = Low/high-power select is done via the SOSCSEL<1:0> Configuration bits
bit 3	<b>SOSCGO:</b> Oscillator Start Control bit 1 = Oscillator is running even if no other sources are requesting it. 0 = Oscillator is shut off if no other sources are requesting it (When the SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.)
bit 2	<b>Unimplemented:</b> Read as '0'
bit 1	<b>MFIOFS:</b> MF-INTOSC Frequency Stable bit 1 = MF-INTOSC is stable 0 = MF-INTOSC is not stable
bit 0	<b>MFIOSEL:</b> MF-INTOSC Select bit 1 = MF-INTOSC is used in place of HF-INTOSC frequencies of 500 kHz, 250 kHz and 31.25 kHz 0 = MF-INTOSC is not used

- Note 1:** When SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.

# PIC18F66K80 FAMILY

## REGISTER 3-3: OSCTUNE: OSCILLATOR TUNING 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
INTSRC	PLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0
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      **INTSRC:** Internal Oscillator Low-Frequency Source Select bit  
1 = 31.25 kHz device clock is derived from 16 MHz INTOSC source (divide-by-512 enabled, HF-INTOSC)  
0 = 31 kHz device clock is derived from INTOSC 31 kHz oscillator (LF-INTOSC)
- bit 6      **PLLEN:** Frequency Multiplier PLL Enable bit  
1 = PLL is enabled  
0 = PLL is disabled
- bit 5-0     **TUN<5:0>:** Fast RC Oscillator (INTOSC) Frequency Tuning bits  
011111 = Maximum frequency  
•            •  
•            •  
000001  
000000 = Center frequency; fast RC oscillator is running at the calibrated frequency  
111111  
•            •  
•            •  
100000 = Minimum frequency

# PIC18F66K80 FAMILY

---

### 3.3 Clock Sources and Oscillator Switching

Essentially, PIC18F66K80 family devices have these independent clock sources:

- Primary oscillators
- Secondary oscillators
- Internal oscillator

The **primary oscillators** can be thought of as the main device oscillators. These are any external oscillators connected to the OSC1 and OSC2 pins, and include the External Crystal and Resonator modes and the External Clock modes. If selected by the FOSC<3:0> Configuration bits (CONFIG1H<3:0>), the internal oscillator block may be considered a primary oscillator. The internal oscillator block can be one of the following:

- 31 kHz LF-INTOSC source
- 31 kHz to 500 kHz MF-INTOSC source
- 31 kHz to 16 MHz HF-INTOSC source

The particular mode is defined by the FOSC<sub>x</sub> Configuration bits. The details of these modes are covered in [Section 3.5 “External Oscillator Modes”](#).

The **secondary oscillators** are external clock sources that are not connected to the OSC1 or OSC2 pin. These sources may continue to operate, even after the controller is placed in a power-managed mode. PIC18F66K80 family devices offer the SOSC (Timer1/3/5/7) oscillator as a secondary oscillator source.

The SOSC can be enabled from any peripheral that requests it. The SOSC can be enabled several ways by doing one of the following:

- The SOSC is selected as the source by either of the odd timers, which is done by each respective SOSCEN bit (TxCON<3>)
- The SOSC is selected as the CPU clock source by the SCS<sub>x</sub> bits (OSCCON<1:0>)
- The SOSCGO bit is set (OSCCON2<3>)

The SOSCGO bit is used to warm up the SOSC so that it is ready before any peripheral requests it.

The secondary oscillator has three Run modes. The SOSCSEL<1:0> bits (CONFIG1L<4:3>) decide the SOSC mode of operation:

- 11 = High-Power SOSC Circuit
- 10 = Digital (SCLKI) mode
- 11 = Low-Power SOSC Circuit

If a secondary oscillator is not desired and digital I/O on port pins, RC0 and RC1, is needed, the SOSCSEL<sub>x</sub> bits must be set to Digital mode.

In addition to being a primary clock source in some circumstances, the **internal oscillator** is available as a power-managed mode clock source. The LF-INTOSC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor. The internal oscillator block is discussed in more detail in [Section 3.6 “Internal Oscillator Block”](#).

The PIC18F66K80 family includes features that allow the device clock source to be switched from the main oscillator, chosen by device configuration, to one of the alternate clock sources. When an alternate clock source is enabled, various power-managed operating modes are available.

#### 3.3.1 OSC1/OSC2 OSCILLATOR

The OSC1/OSC2 oscillator block is used to provide the oscillator modes and frequency ranges:

Mode	Design Operating Frequency
LP	31.25-100 kHz
XT	100 kHz to 4 MHz
HS	4 MHz to 25 MHz
EC	0 to 64 MHz (external clock)
EXTRC	0 to 4 MHz (external RC)

The crystal-based oscillators (XT, HS and LP) have a built-in start-up time. The operation of the EC and EXT RC clocks is immediate.

#### 3.3.2 CLOCK SOURCE SELECTION

The System Clock Select bits, SCS<1:0> (OSCCON<1:0>), select the clock source. The available clock sources are the primary clock defined by the FOSC<3:0> Configuration bits, the secondary clock (SOSC oscillator) and the internal oscillator. The clock source changes after one or more of the bits is written to, following a brief clock transition interval.

The OSTs (OSCCON<3>) and SOSCRUN (OSCCON2<6>) bits indicate which clock source is currently providing the device clock. The OSTs bit indicates that the Oscillator Start-up Timer (OST) has timed out and the primary clock is providing the device clock in primary clock modes. The SOSCRUN bit indicates when the SOSC oscillator (from Timer1/3/5/7) is providing the device clock in secondary clock modes. In power-managed modes, only one of these bits will be set at any time. If neither of these bits is set, the INTOSC is providing the clock or the internal oscillator has just started and is not yet stable.

The IDLEN bit (OSCCON<7>) 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 Timer1/3/5/7 oscillator must be enabled to select the secondary clock source. The Timerx oscillator is enabled by setting the SOSCEN bit in the Timerx Control register (TxCON<3>). If the Timerx oscillator is not enabled, then any attempt to select a secondary clock source when executing a SLEEP instruction will be ignored.
- 2:** It is recommended that the Timerx oscillator be operating and stable before executing the SLEEP instruction or a very long delay may occur while the Timerx oscillator starts.

### 3.3.2.1 System Clock Selection and Device Resets

Since the SCSx bits are cleared on all forms of Reset, this means the primary oscillator defined by the FOSC<3:0> Configuration bits is used as the primary clock source on device Resets. This could either be the internal oscillator block by itself, or one of the other primary clock sources (HS, EC, XT, LP, External RC and PLL-enabled modes).

In those cases when the internal oscillator block, without PLL, is the default clock on Reset, the Fast RC Oscillator (INTOSC) will be used as the device clock source. It will initially start at 8 MHz; the postscaler selection that corresponds to the Reset value of the IRCF<2:0> bits ('110').

Regardless of which primary oscillator is selected, INTOSC will always be enabled on device power-up. It serves as the clock source until the device has loaded its configuration values from memory. It is at this point that the FOSCx Configuration bits are read and the oscillator selection of the operational mode is made.

Note that either the primary clock source or the internal oscillator will have two bit setting options for the possible values of the SCS<1:0> bits, at any given time.

### 3.3.3 OSCILLATOR TRANSITIONS

PIC18F66K80 family 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”](#).

### 3.4 RC Oscillator

For timing-insensitive applications, the RC and RCIO Oscillator modes offer additional cost savings. The actual oscillator frequency is a function of several factors:

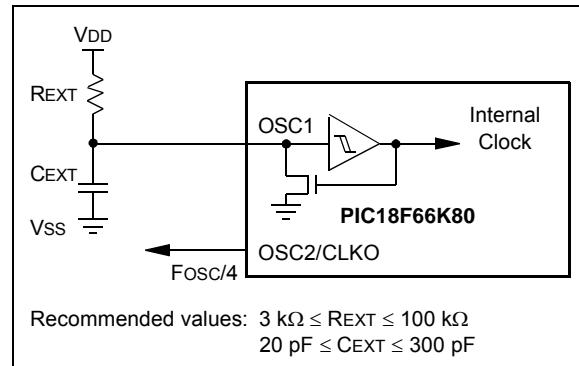
- Supply voltage
- Values of the external resistor (REXT) and capacitor (CEXT)
- Operating temperature

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 CEXT values)
- Variations within the tolerance of the limits of REXT and CEXT

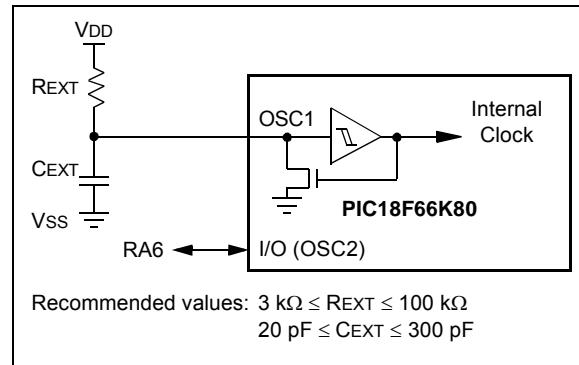
In the RC Oscillator mode, the oscillator frequency, divided by 4, is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. [Figure 3-2](#) shows how the R/C combination is connected.

**FIGURE 3-2: RC OSCILLATOR MODE**



The RCIO Oscillator mode ([Figure 3-3](#)) 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 3-3: RCIO OSCILLATOR MODE**



# PIC18F66K80 FAMILY

## 3.5 External Oscillator Modes

### 3.5.1 CRYSTAL OSCILLATOR/CERAMIC RESONATORS (HS MODES)

In HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. [Figure 3-4](#) shows the pin connections.

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

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

**TABLE 3-2: CAPACITOR SELECTION FOR CERAMIC RESONATORS**

Typical Capacitor Values Used:			
Mode	Freq.	OSC1	OSC2
HS	8.0 MHz	27 pF	27 pF
	16.0 MHz	22 pF	22 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. Refer to the following application notes for oscillator-specific information:

- AN588, "PIC® Microcontroller Oscillator Design Guide"
- AN826, "Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices"
- AN849, "Basic PIC® Oscillator Design"
- AN943, "Practical PIC® Oscillator Analysis and Design"
- AN949, "Making Your Oscillator Work"

See the notes following [Table 3-3](#) for additional information.

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

Osc Type	Crystal Freq.	Typical Capacitor Values Tested:	
		C1	C2
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	20 MHz	15 pF	15 pF

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

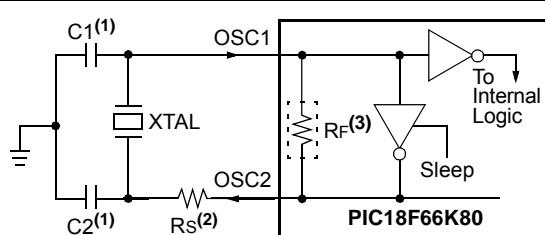
Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

Refer to the Microchip application notes cited in [Table 3-2](#) for oscillator specific information. Also see the notes following this table for additional information.

**Note 1:** Higher capacitance increases the stability of oscillator but also increases the start-up time.

- 2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
- 3: Rs may be required to avoid overdriving crystals with low drive level specification.
- 4: Always verify oscillator performance over the VDD and temperature range that is expected for the application.

**FIGURE 3-4: CRYSTAL/CERAMIC RESONATOR OPERATION (HS OR HSPLL CONFIGURATION)**



**Note 1:** See [Table 3-2](#) and [Table 3-3](#) for initial values of C1 and C2.

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

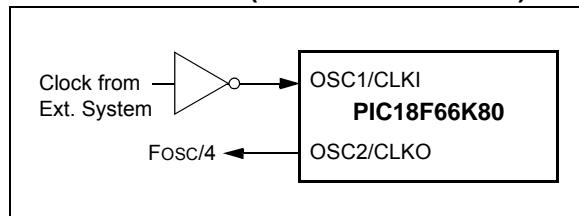
**3:** RF varies with the oscillator mode chosen.

### 3.5.2 EXTERNAL CLOCK INPUT (EC MODES)

The EC and ECPLL 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.

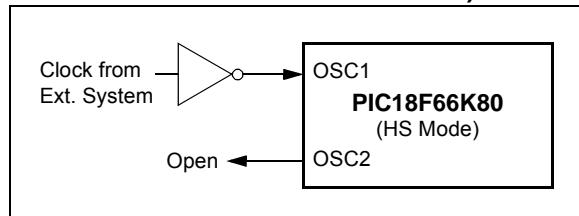
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. [Figure 3-5](#) shows the pin connections for the EC Oscillator mode.

**FIGURE 3-5:** EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)



An external clock source may also be connected to the OSC1 pin in the HS mode, as shown in [Figure 3-6](#). In this configuration, the divide-by-4 output on OSC2 is not available. Current consumption in this configuration will be somewhat higher than EC mode, as the internal oscillator's feedback circuitry will be enabled (in EC mode, the feedback circuit is disabled).

**FIGURE 3-6:** EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)



### 3.5.3 PLL FREQUENCY MULTIPLIER

A Phase Lock Loop (PLL) circuit is provided as an option for users who want 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.

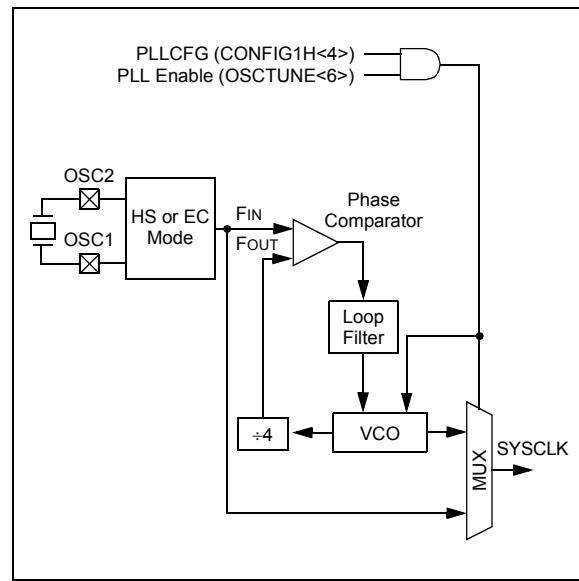
### 3.5.3.1 HSPLL and ECPLL Modes

The HSPLL and ECPLL modes provide the ability to selectively run the device at four times the external oscillating source to produce frequencies up to 64 MHz.

The PLL is enabled by setting the PLLEN bit (OSCTUNE<6>) or the PLLCFG bit (CONFIG1H<4>). For the HF-INTOSC as primary, the PLL must be enabled with the PLLEN. This provides a software control for the PLL, enabling even if PLLCFG is set to '1', so that the PLL is enabled only when the HF-INTOSC frequency is within the 4 MHz to 16 MHz input range.

This also enables additional flexibility for controlling the application's clock speed in software. The PLLEN should be enabled in HF-INTOSC mode only if the input frequency is in the range of 4 MHz-16 MHz.

**FIGURE 3-7:** PLL BLOCK DIAGRAM



### 3.5.3.2 PLL and HF-INTOSC

The PLL is available to the internal oscillator block when the internal oscillator block is configured as the primary clock source. In this configuration, the PLL is enabled in software and generates a clock output of up to 64 MHz.

The operation of INTOSC with the PLL is described in [Section 3.6.2 “INTPLL Modes”](#). Care should be taken that the PLL is enabled only if the HF-INTOSC postscaler is configured for 4 MHz, 8 MHz or 16 MHz.

# PIC18F66K80 FAMILY

## 3.6 Internal Oscillator Block

The PIC18F66K80 family of devices includes an internal oscillator block which generates two different clock signals. Either clock can be used as the microcontroller's clock source, which may eliminate the need for an external oscillator circuit on the OSC1 and/or OSC2 pins.

The Internal oscillator consists of three blocks, depending on the frequency of operation. They are HF-INTOSC, MF-INTOSC and LF-INTOSC.

In HF-INTOSC mode, the internal oscillator can provide a frequency ranging from 31 KHz to 16 MHz, with the postscaler deciding the selected frequency (IRCF<2:0>).

The INTSRC bit (OSCTUNE<7>) and MFIOSEL bit (OSCCON2<0>) also decide which INTOSC provides the lower frequency (500 kHz to 31 KHz). For the HF-INTOSC to provide these frequencies, INTSRC = 1 and MFIOSEL = 0.

In HF-INTOSC, the postscaler (IRCF<2:0>) provides the frequency range of 31 kHz to 16 MHz. If HF-INTOSC is used with the PLL, the input frequency to the PLL should be 4 MHz to 16 MHz (IRCF<2:0> = 111, 110 or 101).

For MF-INTOSC mode to provide a frequency range of 500 kHz to 31 kHz, INTSRC = 1 and MFIOSEL = 1. The postscaler (IRCF<2:0>), in this mode, provides the frequency range of 31 kHz to 500 kHz.

The LF-INTOSC can provide only 31 kHz if INTSRC = 0.

The LF-INTOSC provides 31 kHz and is enabled if it is selected as the device clock source. The mode is enabled automatically when any of the following are enabled:

- Power-up Timer (PWRT)
- Fail-Safe Clock Monitor (FSCM)
- Watchdog Timer (WDT)
- Two-Speed Start-up

These features are discussed in greater detail in [Section 28.0 "Special Features of the CPU"](#).

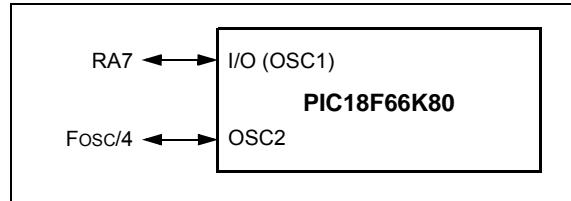
The clock source frequency (HF-INTOSC, MF-INTOSC or LF-INTOSC direct) is selected by configuring the IRCFx bits of the OSCCON register, as well the INTSRC and MFIOSEL bits. The default frequency on device Resets is 8 MHz.

### 3.6.1 INTIO MODES

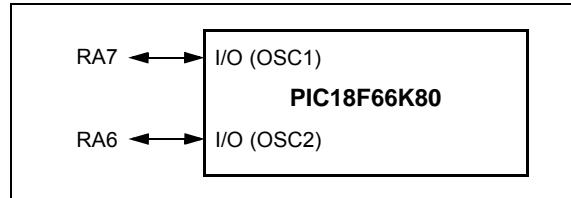
Using the internal oscillator as the clock source eliminates the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct oscillator configurations, which are determined by the FOSCx Configuration bits, are available:

- In INTIO1 mode, the OSC2 pin (RA6) outputs Fosc/4, while OSC1 functions as RA7 for digital input and output. (see [Figure 3-8](#)) for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6 (see [Figure 3-9](#)). Both are available as digital input and output ports.

**FIGURE 3-8: INTIO1 OSCILLATOR MODE**



**FIGURE 3-9: INTIO2 OSCILLATOR MODE**



### 3.6.2 INTPLL MODES

The 4x Phase Lock Loop (PLL) can be used with the HF-INTOSC to produce faster device clock speeds than are normally possible with the internal oscillator sources. When enabled, the PLL produces a clock speed of 16 MHz or 64 MHz.

PLL operation is controlled through software. The control bits, PLLEN (OSCTUNE<6>) and PLLCFG (CONFIG1H<4>), are used to enable or disable its operation. The PLL is available only to HF-INTOSC. The other oscillator is set with HS and EC modes. Additionally, the PLL will only function when the selected output frequency is either 4 MHz or 16 MHz (OSCCON<6:4> = 111, 110 or 101).

Like the INTIO modes, there are two distinct INTPLL modes available:

- In INTPLL1 mode, the OSC2 pin outputs Fosc/4, while OSC1 functions as RA7 for digital input and output. Externally, this is identical in appearance to INTIO1 (see [Figure 3-8](#)).
- In INTPLL2 mode, OSC1 functions as RA7 and OSC2 functions as RA6, both for digital input and output. Externally, this is identical to INTIO2 (see [Figure 3-9](#)).

### 3.6.3 INTERNAL OSCILLATOR OUTPUT FREQUENCY AND TUNING

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 16 MHz. It can be adjusted in the user's application by writing to TUN<5:0> (OSCTUNE<5:0>) in the OSCTUNE register ([Register 3-3](#)).

When the OSCTUNE register is modified, the INTOSC (HF-INTOSC and MF-INTOSC) frequency will begin shifting to the new frequency. The oscillator will require some time to stabilize. Code execution continues during this shift and there is no indication that the shift has occurred.

The LF-INTOSC oscillator operates independently of the HF-INTOSC or the MF-INTOSC source. Any changes in the HF-INTOSC or the MF-INTOSC source, across voltage and temperature, are not necessarily reflected by changes in LF-INTOSC or vice versa. The frequency of LF-INTOSC is not affected by OSCTUNE.

### 3.6.4 INTOSC FREQUENCY DRIFT

The INTOSC frequency may drift as VDD or temperature changes and 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. Depending on the device, this may have no effect on the LF-INTOSC clock source frequency.

Tuning INTOSC requires knowing when to make the adjustment, in which direction it should be made, and in some cases, how large a change is needed. Three compensation techniques are shown here.

#### 3.6.4.1 Compensating with the EUSARTx

An adjustment may be required when the EUSARTx begins to generate framing errors or receives data with errors while in Asynchronous mode. Framing errors indicate that the device clock frequency is too high. To adjust for this, decrement the value in OSCTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low. To compensate, increment OSCTUNE to increase the clock frequency.

#### 3.6.4.2 Compensating with the Timers

This technique compares device clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the SOSC oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is much greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

#### 3.6.4.3 Compensating with the CCP Module in Capture Mode

A CCP module can use free-running Timer1 (or Timer3), clocked by the internal oscillator block and an external event with a known period (i.e., AC power frequency). The time of the first event is captured in the CCPRxH:CCPRxL registers and is recorded for use later. When the second event causes a capture, the time of the first event is subtracted from the time of the second event. Since the period of the external event is known, the time difference between events can be calculated.

If the measured time is much greater than the calculated time, the internal oscillator block is running too fast. To compensate, decrement the OSCTUNE register. If the measured time is much less than the calculated time, the internal oscillator block is running too slow. To compensate, increment the OSCTUNE register.

### 3.7 Reference Clock Output

In addition to the Fosc/4 clock output, in certain oscillator modes, the device clock in the PIC18F66K80 family can also be configured to provide a reference clock output signal to a port pin. This feature is available in all oscillator configurations and allows the user to select a greater range of clock submultiples to drive external devices in the application.

This reference clock output is controlled by the REFOCON register ([Register 3-4](#)). Setting the ROON bit (REFOCON<7>) makes the clock signal available on the REFO (RC3) pin. The RODIV<3:0> bits enable the selection of 16 different clock divider options.

The ROSSLP and ROSEL bits (REFOCON<5:4>) control the availability of the reference output during Sleep mode. The ROSEL bit determines if the oscillator on OSC1 and OSC2, or the current system clock source, is used for the reference clock output. The ROSSLP bit determines if the reference source is available on RE3 when the device is in Sleep mode.

To use the reference clock output in Sleep mode, both the ROSSLP and ROSEL bits must be set. The device clock must also be configured for an EC or HS mode. If not, the oscillator on OSC1 and OSC2 will be powered down when the device enters Sleep mode. Clearing the ROSEL bit allows the reference output frequency to change as the system clock changes during any clock switches.

# PIC18F66K80 FAMILY

## REGISTER 3-4: REFOCON: REFERENCE OSCILLATOR 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
ROON	—	ROSSLP	ROSEL <sup>(1)</sup>	RODIV3	RODIV2	RODIV1	RODIV0
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      **ROON:** Reference Oscillator Output Enable bit  
1 = Reference oscillator output is available on REFO pin  
0 = Reference oscillator output is disabled
- bit 6      **Unimplemented:** Read as '0'
- bit 5      **ROSSLP:** Reference Oscillator Output Stop in Sleep bit  
1 = Reference oscillator continues to run in Sleep  
0 = Reference oscillator is disabled in Sleep
- bit 4      **ROSEL:** Reference Oscillator Source Select bit<sup>(1)</sup>  
1 = Primary oscillator (EC or HS) is used as the base clock  
0 = System clock is used as the base clock; base clock reflects any clock switching of the device
- bit 3-0     **RODIV<3:0>:** Reference Oscillator Divisor Select bits  
1111 = Base clock value divided by 32,768  
1110 = Base clock value divided by 16,384  
1101 = Base clock value divided by 8,192  
1100 = Base clock value divided by 4,096  
1011 = Base clock value divided by 2,048  
1010 = Base clock value divided by 1,024  
1001 = Base clock value divided by 512  
1000 = Base clock value divided by 256  
0111 = Base clock value divided by 128  
0110 = Base clock value divided by 64  
0101 = Base clock value divided by 32  
0100 = Base clock value divided by 16  
0011 = Base clock value divided by 8  
0010 = Base clock value divided by 4  
0001 = Base clock value divided by 2  
0000 = Base clock value

**Note 1:** For ROSEL (REFOCON<4>), the primary oscillator is available only when configured as the default via the FOSCx settings. This is regardless of whether the device is in Sleep mode.

## 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 SOSC oscillator is operating and providing the device clock. The SOSC oscillator may also run in all power-managed modes if required to clock SOSC.

In RC\_RUN and RC\_IDLE modes, the internal oscillator provides the device clock source. The 31 kHz LF-INTOSC 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 28.2 “Watchdog Timer \(WDT\)”](#) through [Section 28.5 “Fail-Safe Clock Monitor”](#) for more information on WDT, Fail-Safe Clock Monitor and Two-Speed Start-up).

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 INTOSC is required to support WDT operation. The SOSC oscillator may be operating to support Timer1 or 3. Other features may be operating that do not require a device clock source (i.e., MSSP slave, INTx pins and others). Peripherals that may add significant current consumption are listed in [Section 31.2 “DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family \(Industrial/Extended\)”](#).

**TABLE 3-4: OSC1 AND OSC2 PIN STATES IN SLEEP MODE**

Oscillator Mode	OSC1 Pin	OSC2 Pin
EC, ECPLL	Floating, pulled by external clock	At logic low (clock/4 output)
HS, HSPLL	Feedback inverter disabled at quiescent voltage level	Feedback inverter disabled at quiescent voltage level
INTOSC, INTPLL1/2	I/O pin, RA6, direction controlled by TRISA<6>	I/O pin, RA6, direction controlled by TRISA<7>

**Note:** See [Section 5.0 “Reset”](#) for time-outs due to Sleep and MCLR Reset.

## 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 5.6.1 “Power-up Timer \(PWRT\)”](#).

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up time of about 64 ms (Parameter 33, [Table 31-11](#)); it is always enabled.

The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (HS, XT or LP modes). The OST does this by counting 1,024 oscillator cycles before allowing the oscillator to clock the device.

There is a delay of interval, TCSD (Parameter 38, [Table 31-11](#)), following POR, while the controller becomes ready to execute instructions.

# **PIC18F66K80 FAMILY**

---

---

**NOTES:**

## 4.0 POWER-MANAGED MODES

The PIC18F66K80 family of devices offers 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 (such as battery-powered devices).

There are three categories of power-managed mode:

- Run modes
- Idle modes
- Sleep mode

There is an Ultra Low-Power Wake-up (ULPWU) for waking from Sleep mode.

These categories define which portions of the device are clocked, and sometimes, at 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 ULPWU mode, on the RA0 pin, enables a slow falling voltage to generate a wake-up, even from Sleep, without excess current consumption. (See [Section 4.7 “Ultra Low-Power Wake-up”](#).)

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. This feature allows the controller to use the SOSC oscillator instead of the primary one. Another power-saving feature is 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:

- Will the CPU be clocked or not
- What will be the clock source

The IDLEN bit (OSCCON<7>) controls CPU clocking, while the SCS<1:0> 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 SCS<1:0> bits select one of three clock sources for power-managed modes. Those sources are:

- The primary clock as defined by the FOSC<3:0> Configuration bits
- The Secondary Clock (the SOSC oscillator)
- The Internal Oscillator block (for LF-INTOSC modes)

#### 4.1.2 ENTERING POWER-MANAGED MODES

Switching from one power-managed mode to another begins by loading the OSCCON register. The SCS<1:0> bits select the clock source and determine which Run or Idle mode is 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 considerations are discussed in [Section 4.1.3 “Clock Transitions and Status Indicators”](#) and subsequent sections.

Entering 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 and impending mode, a change to a power-managed mode does not always require setting all of the previously discussed bits. Many transitions can 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 as desired, 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>	SCS<1:0>	CPU	Peripherals	
Sleep	0	N/A	Off	Off	None – All clocks are disabled
PRI_RUN	N/A	00	Clocked	Clocked	Primary – XT, LP, HS, EC, RC and PLL modes. This is the normal, full-power execution mode.
SEC_RUN	N/A	01	Clocked	Clocked	Secondary – SOSC Oscillator
RC_RUN	N/A	1x	Clocked	Clocked	Internal oscillator block <sup>(2)</sup>
PRI_IDLE	1	00	Off	Clocked	Primary – LP, XT, HS, RC, EC
SEC_IDLE	1	01	Off	Clocked	Secondary – SOSC 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 (HF-INTOSC and MG-INTOSC) and INTOSC postscaler, as well as the LF-INTOSC source.

# PIC18F66K80 FAMILY

## 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. The HF-INTOSC and MF-INTOSC are termed as INTOSC in this chapter.

Three bits indicate the current clock source and its status, as shown in [Table 4-2](#). The three bits are:

- OSTS (OSCCON<3>)
- HFIOFS (OSCCON<2>)
- SOSCRUN (OSCCON2<6>)

**TABLE 4-2: SYSTEM CLOCK INDICATOR**

Main Clock Source	OSTS	HFIOFS or MFIOFS	SOSCRUN
Primary Oscillator	1	0	0
INTOSC (HF-INTOSC or MF-INTOSC)	0	1	0
Secondary Oscillator	0	0	1
MF-INTOSC or HF-INTOSC as Primary Clock Source	1	1	0
LF-INTOSC is Running or INTOSC is Not Yet Stable	0	0	0

When the OSTS bit is set, the primary clock is providing the device clock. When the HFIOFS or MFIOFS bit is set, the INTOSC output is providing a stable clock source to a divider that actually drives the device clock. When the SOSCRUN bit is set, the SOSC oscillator is providing the clock. If none of these bits are set, either the LF-INTOSC 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 FOSC<3:0> Configuration bits (CONFIG1H<3:0>). Then, the OSTS and HFIOFS or MFIOFS bits can be set when in PRI\_RUN or PRI\_IDLE mode. This indicates that the primary clock (INTOSC output) is generating a stable output. Entering another INTOSC power-managed mode at the same frequency would clear the OSTS bit.

- Note 1:** Caution should be used when modifying a single IRCF bit. At a lower VDD, it is possible to select a higher clock speed than is supportable by that 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. (For details, see [Section 28.4 “Two-Speed Start-up”](#).) In this mode, the OSTS bit is set. The HFIOFS or MFIOFS bit may be set if the internal oscillator block is the primary clock source. (See [Section 3.2 “Control Registers”](#).)

### 4.2.2 SEC\_RUN MODE

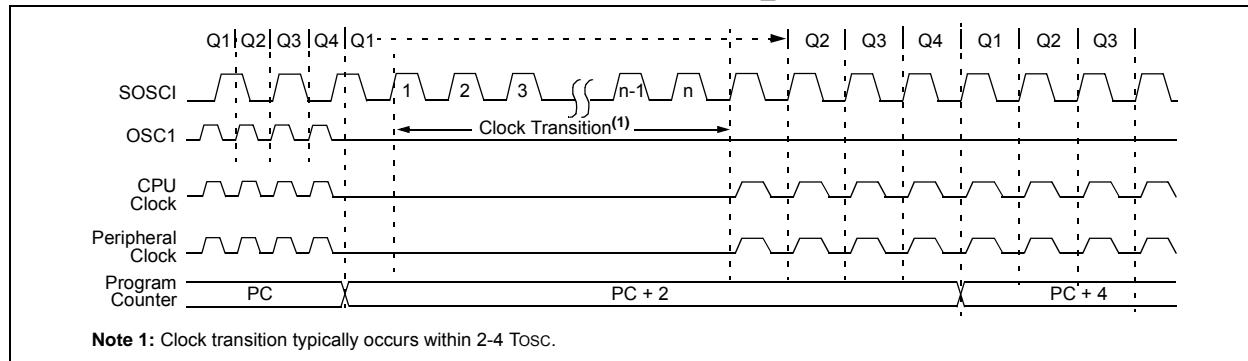
The SEC\_RUN mode is the compatible mode to the “clock-switching” feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the SOSC oscillator. This enables lower power consumption while retaining a high-accuracy clock source.

SEC\_RUN mode is entered by setting the SCS<1:0> bits to ‘01’. The device clock source is switched to the SOSC oscillator (see [Figure 4-1](#)), the primary oscillator is shut down, the SOSCRUN bit (OSCCON2<6>) is set and the OSTS bit is cleared.

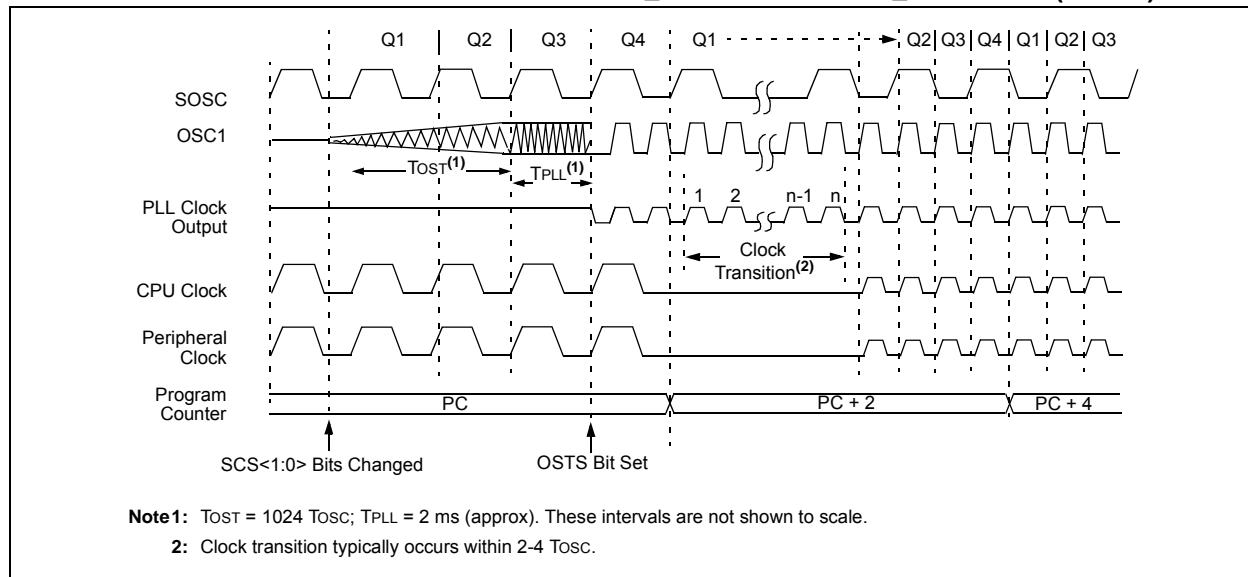
**Note:** The SOSC oscillator can be enabled by setting the SOSCGO bit (OSCCON2<3>). If this bit is set, the clock switch to the SEC\_RUN mode can switch immediately once SCS<1:0> are set to ‘01’.

On transitions from SEC\_RUN mode to PRI\_RUN mode, the peripherals and CPU continue to be clocked from the SOSC 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 SOSCRUN bit is cleared, the OSTS bit is set and the primary clock is providing the clock. The IDLEN and SCSx bits are not affected by the wake-up and the SOSC 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 LF-INTOSC 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 LF-INTOSC or INTOSC (MF-INTOSC or HF-INTOSC) – there are no distinguishable differences between the PRI\_RUN and RC\_RUN modes during execution. Entering or exiting RC\_RUN mode, however, causes a clock switch delay. Therefore, if the primary clock source is the internal oscillator block, using RC\_RUN mode is not recommended.

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

**Note:** Caution should be used when modifying a single IRCF bit. At a lower VDD, it is possible to select a higher clock speed than is supportable by that VDD. Improper device operation may result if the VDD/FOSC specifications are violated.

# PIC18F66K80 FAMILY

---

If the IRCFx bits and the INTSRC bit are all clear, the INTOSC output (HF-INTOSC/MF-INTOSC) is not enabled and the HFIOFS and MFIOFS bits will remain clear. There will be no indication of the current clock source. The LF-INTOSC source is providing the device clocks.

If the IRCFx bits are changed from all clear (thus, enabling the INTOSC output) or if INTSRC or MFIOSEL is set, the HFIOFS or MFIOFS bit is set after the INTOSC output becomes stable. For details, see [Table 4-3](#).

**TABLE 4-3: INTERNAL OSCILLATOR FREQUENCY STABILITY BITS**

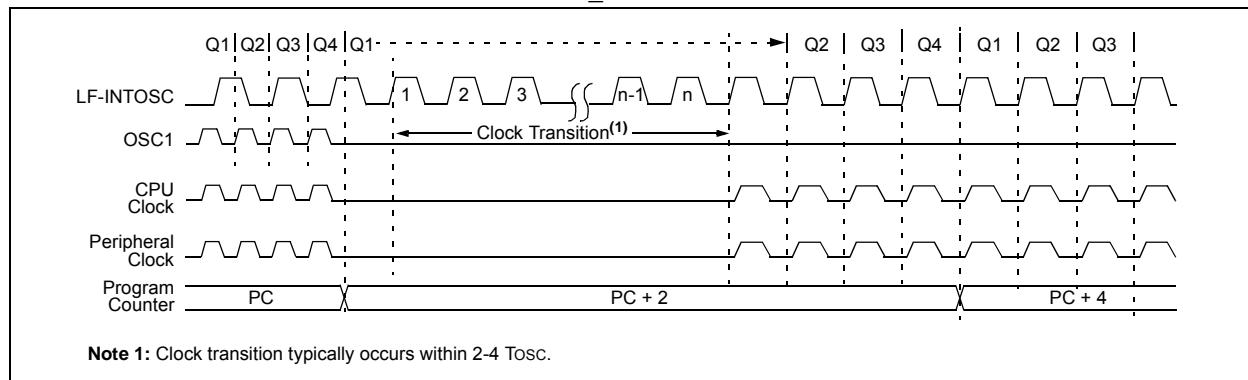
IRCF<2:0>	INTSRC	MFIOSEL	Status of MFIOFS or HFIOFS when INTOSC is Stable
000	0	x	MFIOFS = 0, HFIOFS = 0 and clock source is LF-INTOSC
000	1	0	MFIOFS = 0, HFIOFS = 1 and clock source is HF-INTOSC
000	1	1	MFIOFS = 1, HFIOFS = 0 and clock source is MF-INTOSC
Non-Zero	x	0	MFIOFS = 0, HFIOFS = 1 and clock source is HF-INTOSC
Non-Zero	x	1	MFIOFS = 1, HFIOFS = 0 and clock source is MF-INTOSC

Clocks to the device continue while the INTOSC source stabilizes after an interval of TIOBST (Parameter 39, [Table 31-11](#)).

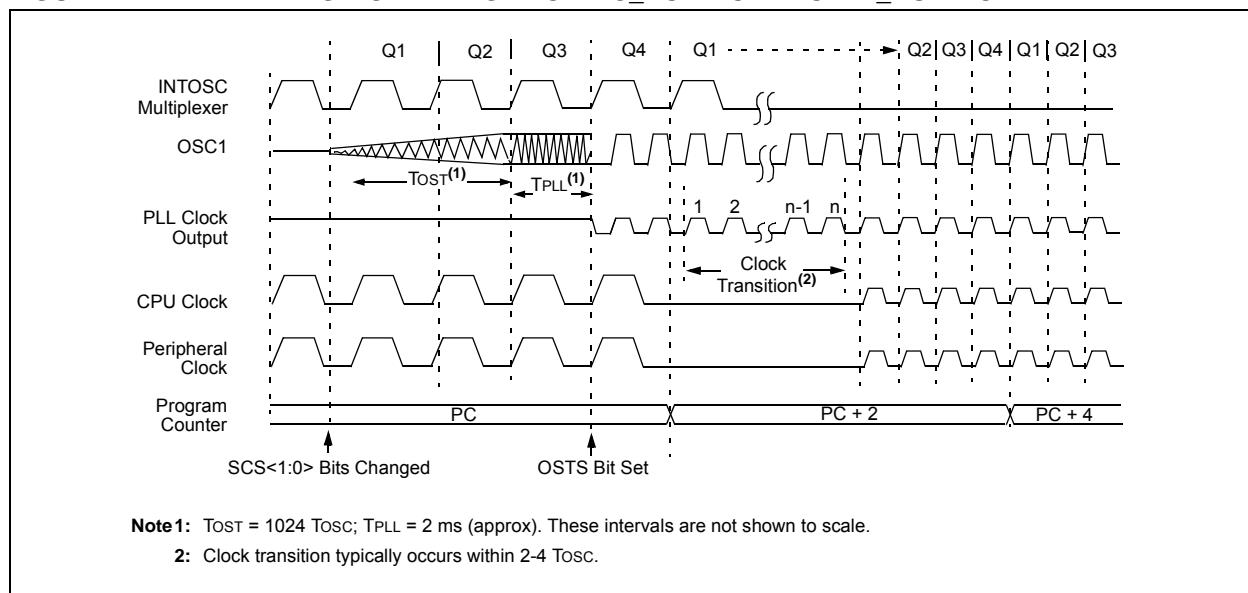
If the IRCFx bits were previously at a non-zero value, or if INTSRC was set before setting SCS1 and the INTOSC source was already stable, the HFIOFS or MFIOFS bit will remain set.

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 HFIOFS or MFIOFS bit is cleared, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCSx bits are not affected by the switch. The LF-INTOSC source will continue to run if either the WDT or the Fail-Safe Clock Monitor (FSCM) is enabled.

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



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



# PIC18F66K80 FAMILY

## 4.3 Sleep Mode

The power-managed Sleep mode in the PIC18F66K80 family of 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 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 LF-INTOSC source will continue to operate. If the SOSC 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 SCS<1:0> bits becomes ready (see Figure 4-6). Alternately, the device will be clocked from the internal oscillator block if either the Two-Speed Start-up or the Fail-Safe Clock Monitor is enabled (see Section 28.0 “Special Features of the CPU”). In either case, the OSTST bit is set when the primary clock is providing the device clocks. The IDLEN and SCSx 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.

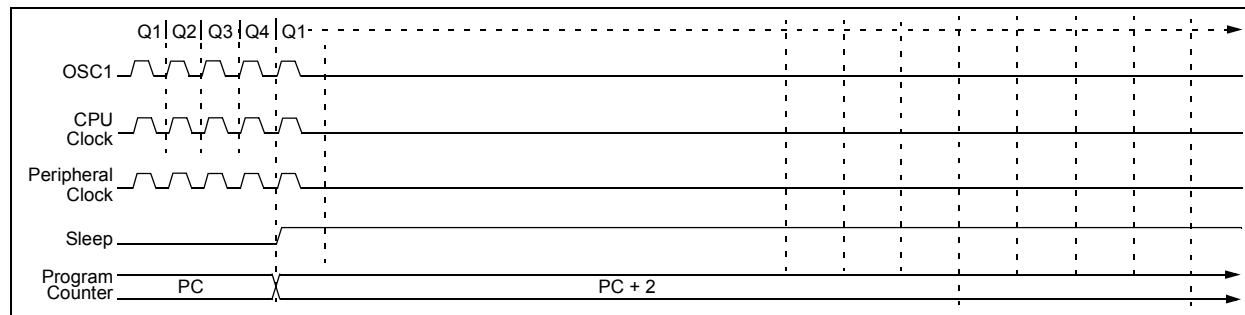
If the IDLEN bit is set to a ‘1’ when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected using the SCS<1:0> bits. The CPU, however, will not be clocked. The clock source status bits are not affected. This approach is a quick method to switch from a given Run mode to its corresponding Idle mode.

If the WDT is selected, the LF-INTOSC source will continue to operate. If the SOSC oscillator is enabled, it will also continue to run.

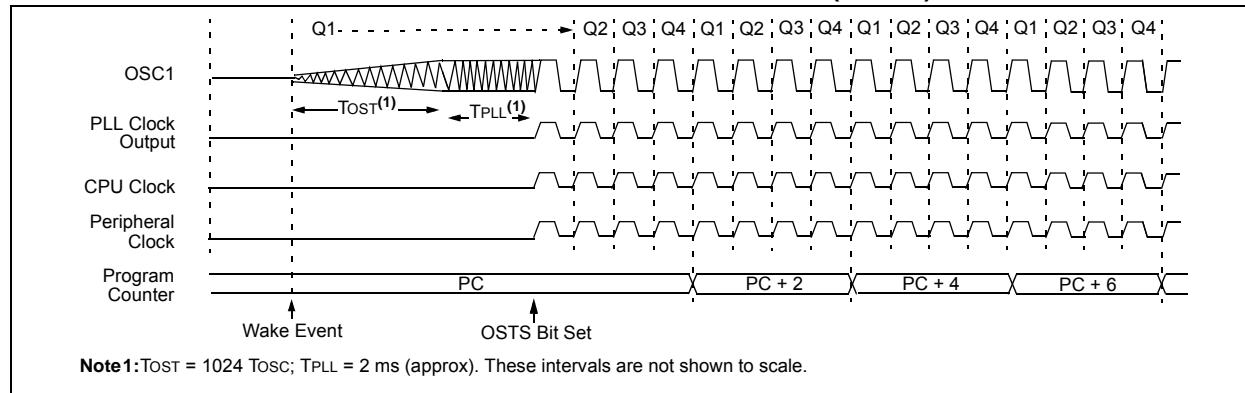
Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset. When a wake event occurs, CPU execution is delayed by an interval of Tcsd (Parameter 38, Table 31-11) while it becomes ready to execute code. When the CPU begins executing 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 SCSx bits are not affected by the wake-up.

While in any Idle mode or Sleep mode, a WDT time-out will result in a WDT wake-up to the Run mode currently specified by the SCS<1:0> bits.

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



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



## 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 SCSx bits and execute SLEEP. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC<3:0> Configuration bits. The OSTS bit remains set (see [Figure 4-7](#)).

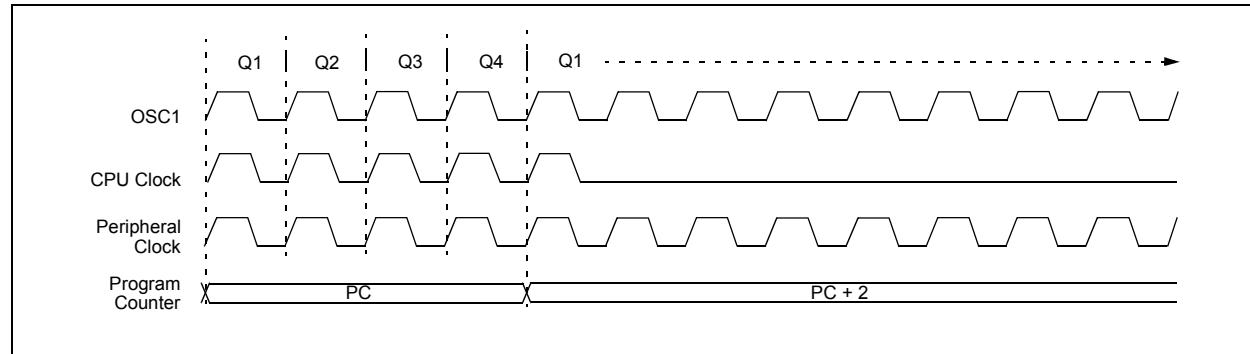
When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval, TcSD (Parameter 39, [Table 31-11](#)), is required between the wake event and the start of code execution. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCSx bits are not affected by the wake-up (see [Figure 4-8](#)).

## 4.4.2 SEC\_IDLE MODE

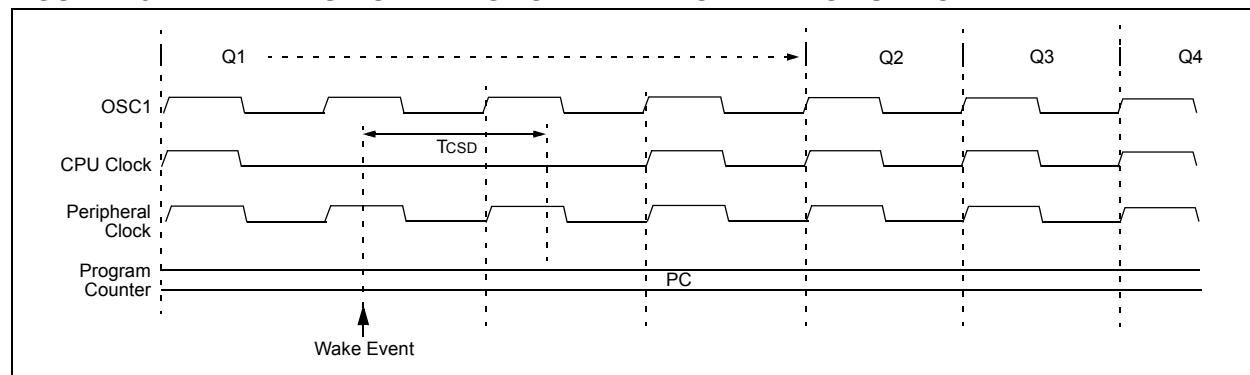
In SEC\_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the SOSC oscillator. This mode is entered from SEC\_RUN by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set the IDLEN bit first, then set the SCS<1:0> bits to ‘01’ and execute SLEEP. When the clock source is switched to the SOSC oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the SOSCRUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the SOSC oscillator. After an interval of TcSD following the wake event, the CPU begins executing code that is being clocked by the SOSC oscillator. The IDLEN and SCSx bits are not affected by the wake-up and the SOSC oscillator continues to run (see [Figure 4-8](#)).

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



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



# PIC18F66K80 FAMILY

---

## 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 provides 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. To maintain software compatibility with future devices, it is recommended that SCS0 also be cleared, though its value is ignored. The INTOSC multiplexer may be used to select a higher clock frequency by modifying the IRCFx 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 IRCFx bits are set to any non-zero value, or the INTSRC/MFIOSEL bit is set, the INTOSC output is enabled. The HFIOFS/MFIOFS bits become set, after the INTOSC output becomes stable, after an interval of TIOBST (Parameter 38, Table 31-11). For information on the HFIOFS/MFIOFS bits, see [Table 4-3](#).

Clocks to the peripherals continue while the INTOSC source stabilizes. The HFIOFS/MFIOFS bits will remain set if the IRCFx bits were previously at a non-zero value or if INTSRC was set before the SLEEP instruction was executed and the INTOSC source was already stable. If the IRCFx bits and INTSRC are all clear, the INTOSC output will not be enabled, the HFIOFS/MFIOFS bits 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 (Parameter 38, Table 31-11) following the wake event, the CPU begins executing code clocked by the INTOSC multiplexer. The IDLEN and SCSx bits are not affected by the wake-up. The INTOSC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

## 4.5 Selective Peripheral Module Control

Idle mode allows users to substantially reduce power consumption by stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume power. There may be cases where the application needs what this mode does not provide: the allocation of power resources to the CPU processing with minimal power consumption from the peripherals.

PIC18F66K80 family devices address this requirement by allowing peripheral modules to be selectively disabled, reducing or eliminating their power consumption. This can be done with two control bits:

- Peripheral Enable bit, generically named XXXEN – Located in the respective module's main control register
- Peripheral Module Disable (PMD) bit, generically named, XXXMD – Located in one of the PMDx Control registers (PMD0, PMD1 or PMD2)

Disabling a module by clearing its XXXEN bit disables the module's functionality, but leaves its registers available to be read and written to. This reduces power consumption, but not by as much as the second approach.

Most peripheral modules have an enable bit.

In contrast, setting the PMD bit for a module disables all clock sources to that module, reducing its power consumption to an absolute minimum. In this state, the control and status registers associated with the peripheral are also disabled, so writes to those registers have no effect and read values are invalid. Many peripheral modules have a corresponding PMD bit.

There are three PMD registers in PIC18F66K80 family devices: PMD0, PMD1 and PMD2. These registers have bits associated with each module for disabling or enabling a particular peripheral.

# PIC18F66K80 FAMILY

## REGISTER 4-1: PMD2: PERIPHERAL MODULE DISABLE REGISTER 2

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	MODMD <sup>(1)</sup>	ECANMD	CMP2MD	CMP1MD
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      **Unimplemented:** Read as '0'
- bit 3      **MODMD:** Modulator Output Module Disable bit<sup>(1)</sup>  
1 = The modulator output module is disabled; all Modulator Output registers are held in Reset and are not writable  
0 = The modulator output module is enabled
- bit 2      **ECANMD:** Enhanced CAN Module Disable bit  
1 = The Enhanced CAN module is disabled; all Enhanced CAN registers are held in Reset and are not writable  
0 = The Enhanced CAN module is enabled
- bit 1      **CMP2MD:** Comparator 2 Module Disable bit  
1 = The Comparator 2 module is disabled; all Comparator 2 registers are held in Reset and are not writable  
0 = The Comparator 2 module is enabled
- bit 0      **CMP1MD:** Comparator 1 Module Disable bit  
1 = The Comparator 1 module is disabled; all Comparator 1 registers are held in Reset and are not writable  
0 = The Comparator 1 module is enabled

**Note 1:** This bit is only implemented on devices with 64 pins (PIC18F6XK80, PIC18LF6XK80).

# PIC18F66K80 FAMILY

## REGISTER 4-2: PMD1: PERIPHERAL MODULE DISABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPMD <sup>(1)</sup>	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
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            **PSPMD:** Peripheral Module Disable bit<sup>(1)</sup>

1 = The PSP module is disabled; all PSP registers are held in Reset and are not writable

0 = The PSP module is enabled

bit 6            **CTMUMD:** PMD CTMU Disable bit

1 = The CTMU module is disabled; all CTMU registers are held in Reset and are not writable

0 = The CTMU module is enabled

bit 5            **ADCMD:** A/D Module Disable bit

1 = The A/D module is disabled; all A/D registers are held in Reset and are not writable

0 = The A/D module is enabled

bit 4            **TMR4MD:** TMR4MD Disable bit

1 = The Timer4 module is disabled; all Timer4 registers are held in Reset and are not writable

0 = The Timer4 module is enabled

bit 3            **TMR3MD:** TMR3MD Disable bit

1 = The Timer3 module is disabled; all Timer3 registers are held in Reset and are not writable

0 = The Timer3 module is enabled

bit 2            **TMR2MD:** TMR2MD Disable bit

1 = The Timer2 module is disabled; all Timer2 registers are held in Reset and are not writable

0 = The Timer2 module is enabled

bit 1            **TMR1MD:** TMR1MD Disable bit

1 = The Timer1 module is disabled; all Timer1 registers are held in Reset and are not writable

0 = The Timer1 module is enabled

bit 0            **TMR0MD:** Timer0 Module Disable bit

1 = The Timer0 module is disabled; all Timer0 registers are held in Reset and are not writable

0 = The Timer0 module is enabled

**Note 1:** This bit is unimplemented on 28-pin devices (PIC18F2XK80, PIC18LF2XK80).

# PIC18F66K80 FAMILY

## REGISTER 4-3: PMD0: PERIPHERAL MODULE DISABLE 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
CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD
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	<b>CCP5MD:</b> CCP5 Module Disable bit 1 = The CCP5 module is disabled; all CCP5 registers are held in Reset and are not writable 0 = The CCP5 module is enabled
bit 6	<b>CCP4MD:</b> CCP4 Module Disable bit 1 = The CCP4 module is disabled; all CCP4 registers are held in Reset and are not writable 0 = The CCP4 module is enabled
bit 5	<b>CCP3MD:</b> CCP3 Module Disable bit 1 = The CCP3 module is disabled; all CCP3 registers are held in Reset and are not writable 0 = The CCP3 module is enabled
bit 4	<b>CCP2MD:</b> CCP2 Module Disable bit 1 = The CCP2 module is disabled; all CCP2 registers are held in Reset and are not writable 0 = The CCP2 module is enabled
bit 3	<b>CCP1MD:</b> ECCP1 Module Disable bit 1 = The ECCP1 module is disabled; all ECCP1 registers are held in Reset and are not writable 0 = The ECCP1 module is enabled
bit 2	<b>UART2MD:</b> EUSART2 Module Disable bit 1 = The USART2 module is disabled; all USART2 registers are held in Reset and are not writable 0 = The USART2 module is enabled
bit 1	<b>UART1MD:</b> EUSART1 Module Disable bit 1 = The USART1 module is disabled; all USART1 registers are held in Reset and are not writable 0 = The USART1 module is enabled
bit 0	<b>SSPMD:</b> MSSP Module Disable bit 1 = The MSSP module is disabled; all SSP registers are held in Reset and are not writable 0 = The MSSP module is enabled

# PIC18F66K80 FAMILY

---

## 4.6 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”](#)).

### 4.6.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle mode or 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 INTCONx 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 10.0 “Interrupts”](#)).

### 4.6.2 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 28.2 “Watchdog Timer \(WDT\)”](#)).

Executing a SLEEP or CLRWDT instruction clears the WDT timer and postscaler, loses the currently selected clock source (if the Fail-Safe Clock Monitor is enabled) and modifies the IRCFx bits in the OSCCON register (if the internal oscillator block is the device clock source).

### 4.6.3 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 HFIOFS/MFIOFS bits are 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 4-4](#).

Code execution can begin before the primary clock becomes ready. If either the Two-Speed Start-up (see [Section 28.4 “Two-Speed Start-up”](#)) or Fail-Safe Clock Monitor (see [Section 28.5 “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.

### 4.6.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

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

- When in PRI\_IDLE mode, where the primary clock source is not stopped
- When 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.

## 4.7 Ultra Low-Power Wake-up

The Ultra Low-Power Wake-up (ULPWU) on pin, RA0, allows a slow falling voltage to generate an interrupt without excess current consumption.

To use this feature:

1. Charge the capacitor on RA0 by configuring the RA0 pin to an output and setting it to '1'.
2. Stop charging the capacitor by configuring RA0 as an input.
3. Discharge the capacitor by setting the ULPEN and ULPSINK bits in the WDTCON register.
4. Configure Sleep mode.
5. Enter Sleep mode.

When the voltage on RA0 drops below VIL, the device wakes up and executes the next instruction.

This feature provides a low-power technique for periodically waking up the device from Sleep mode.

The time-out is dependent on the discharge time of the RC circuit on RA0.

When the ULPWU module wakes the device from Sleep mode, the ULPLVL bit (WDTCON<5>) is set. Software can check this bit upon wake-up to determine the wake-up source.

See [Example 4-1](#) for initializing the ULPWU module.

### EXAMPLE 4-1:    ULTRA LOW-POWER                   WAKE-UP INITIALIZATION

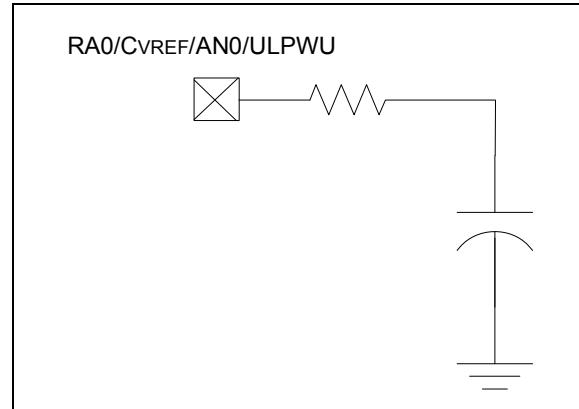
```

//*****
//Charge the capacitor on RA0
//*****
TRISAbits.TRISA0 = 0;
PORTAbits.RA0 = 1;
for(i = 0; i < 10000; i++) Nop();
//*****
//Stop Charging the capacitor
//on RA0
//*****
TRISAbits.TRISA0 = 1;
//*****
//Enable the Ultra Low Power
//Wakeup module and allow
//capacitor discharge
//*****
WDTCONbits.ULPEN = 1;
WDTCONbits.ULPSINK = 1;
//For Sleep
OSCCONbits.IDLEN = 0;
//Enter Sleep Mode
//
Sleep();
//for sleep, execution will
//resume here

```

A series resistor, between RA0 and the external capacitor, provides overcurrent protection for the RA0/CVREF/AN0/ULPWU pin and enables software calibration of the time-out (see [Figure 4-9](#)).

**FIGURE 4-9:    ULTRA LOW-POWER  
                  WAKE-UP INITIALIZATION**



A timer can be used to measure the charge time and discharge time of the capacitor. The charge time can then be adjusted to provide the desired delay in Sleep. This technique compensates for the affects of temperature, voltage and component accuracy. The peripheral can also be configured as a simple programmable Low-Voltage Detect (LVD) or temperature sensor.

**Note:** For more information, see *AN879, "Using the Microchip Ultra Low-Power Wake-up Module"* (DS00879).

# PIC18F66K80 FAMILY

---

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

Power-Managed Mode	Clock Source <sup>(5)</sup>	Exit Delay	Clock Ready Status Bits
PRI_IDLE mode	LP, XT, HS	Tcsd <sup>(1)</sup>	OSTS
	HSPLL		
	EC, RC		
	HF-INTOSC <sup>(2)</sup>		
	MF-INTOSC <sup>(2)</sup>		
	LF-INTOSC		
SEC_IDLE mode	SOSC	Tcsd <sup>(1)</sup>	SOSCRUN
RC_IDLE mode	HF-INTOSC <sup>(2)</sup>	Tcsd <sup>(1)</sup>	HFIOFS MFIOFS None
	MF-INTOSC <sup>(2)</sup>		
	LF-INTOSC		
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>	
	HF-INTOSC <sup>(2)</sup>	TIOBST <sup>(4)</sup>	HFIOFS MFIOFS None
	MF-INTOSC <sup>(2)</sup>		
	LF-INTOSC		

**Note 1:** Tcsd (Parameter 38, Table 31-11) 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"](#)).

**2:** Includes postscaler derived frequencies. On Reset, INTOSC defaults to HF-INTOSC at 8 MHz.

**3:** Tost is the Oscillator Start-up Timer (Parameter 32, Table 31-11). TRC is the PLL Lock-out Timer (Parameter F12, Table 31-7); it is also designated as TPLL.

**4:** Execution continues during TIOBST (Parameter 39, Table 31-11), the INTOSC stabilization period.

**5:** The clock source is dependent upon the settings of the SCSx (OSCCON<1:0>), IRCFx (OSCCON<6:4>) and FOSCx (CONFIG1H<3:0>) bits.

## 5.0 RESET

The PIC18F66K80 family devices differentiate between various kinds of Reset:

- Power-on Reset (POR)
- MCLR Reset during Normal Operation
- MCLR Reset during Power-Managed modes
- Watchdog Timer (WDT) Reset (during execution)
- Configuration Mismatch (CM) Reset
- Programmable Brown-out Reset (BOR)
- RESET Instruction
- Stack Full Reset
- Stack Underflow Reset

This section discusses Resets generated by MCLR, POR and BOR, and covers the operation of the various start-up timers. Stack Reset events are covered in [Section 6.1.3.4 “Stack Full and Underflow Resets”](#). WDT Resets are covered in [Section 28.2 “Watchdog Timer \(WDT\)”](#).

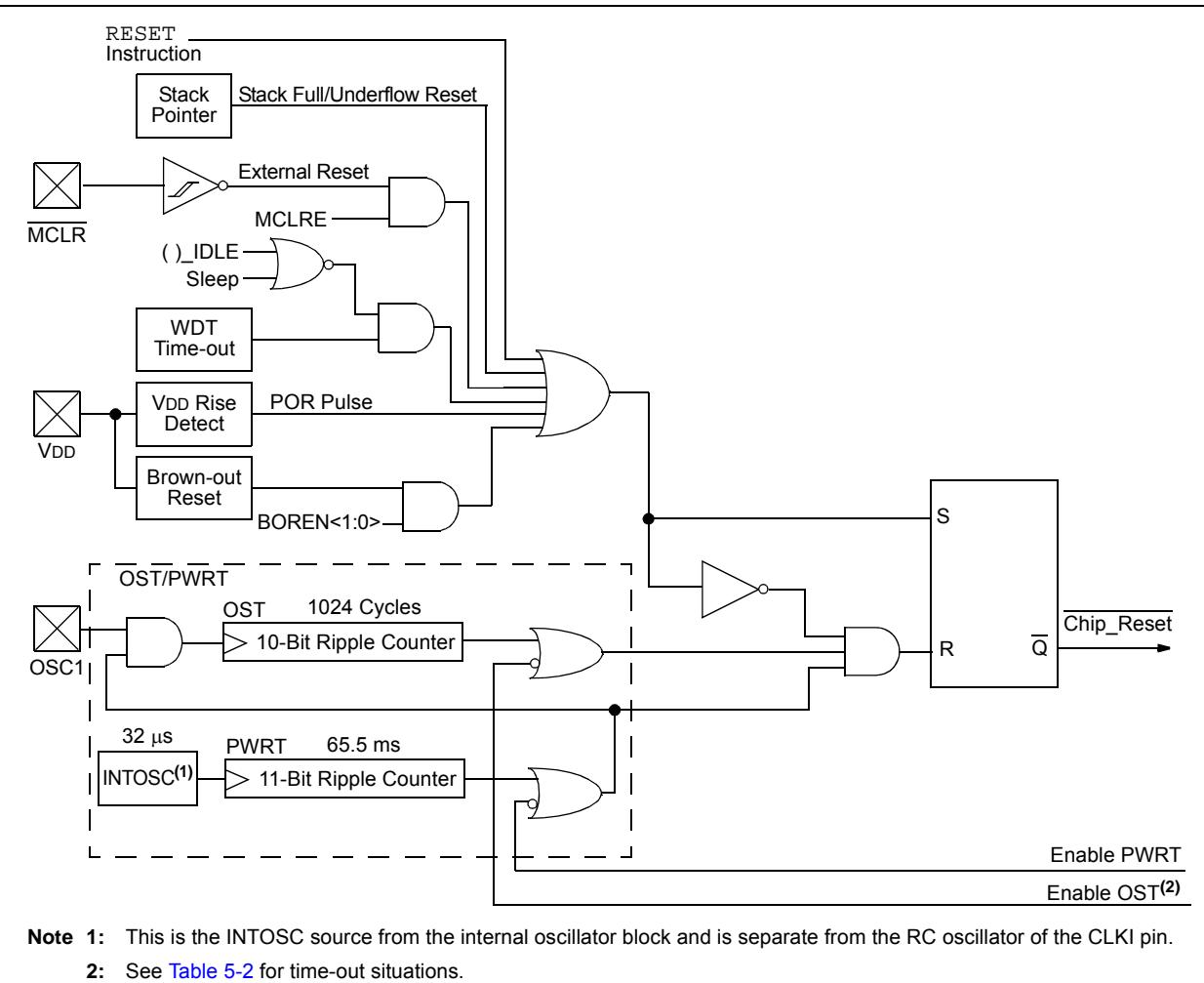
A simplified block diagram of the On-Chip Reset Circuit is shown in [Figure 5-1](#).

## 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 5.7 “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 10.0 “Interrupts”](#). BOR is covered in [Section 5.4 “Brown-out Reset \(BOR\)”](#).

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



# PIC18F66K80 FAMILY

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

R/W-0	R/W-1 <sup>(1)</sup>	R/W-1	R/W-1	R-1	R-1	R/W-0 <sup>(2)</sup>	R/W-0
IPEN	SBOREN	CM	RI	TO	PD	POR	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 = Enables priority levels on interrupts  
0 = Disables priority levels on interrupts (PIC16CXXX Compatibility mode)
- bit 6      **SBOREN:** BOR Software Enable bit<sup>(1)</sup>  
If BOREN<1:0> = 01:  
1 = BOR is enabled  
0 = BOR is disabled  
If BOREN<1:0> = 00, 10 or 11:  
Bit is disabled and reads as '0'.
- bit 5      **CM:** Configuration Mismatch Flag bit  
1 = A Configuration Mismatch Reset has not occurred.  
0 = A Configuration Mismatch Reset has occurred (must be set in software once the Reset occurs)
- bit 4      **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      **TO:** Watchdog Time-out Flag bit  
1 = Set by power-up, CLRWDT instruction or SLEEP instruction  
0 = A WDT time-out has occurred
- bit 2      **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      **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 has occurred (must be set in software after a Power-on Reset occurs)
- bit 0      **BOR:** Brown-out Reset Status bit  
1 = A Brown-out Reset has not occurred (set by firmware only)  
0 = A Brown-out Reset has 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'.  
**2:** The actual Reset value of POR is determined by the type of device Reset. See the notes following this register and [Section 5.7 "Reset State of Registers"](#) for additional information.

- Note 1:** It is recommended that the POR bit be set after a Power-on Reset has been detected so that subsequent Power-on Resets may be detected.  
**2:** Brown-out Reset is said to have occurred when BOR is '0' and POR is '1' (assuming that POR was set to '1' by software immediately after a Power-on Reset).

## 5.2 Master Clear Reset (MCLR)

The 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 MCLR Reset path which detects and ignores small pulses.

The MCLR pin is not driven low by any internal Resets, including the WDT.

In PIC18F66K80 family devices, the MCLR input can be disabled with the MCLRE Configuration bit. When MCLR is disabled, the pin becomes a digital input. See [Section 11.6 "PORTE, TRISE and LATE Registers"](#) for more information.

## 5.3 Power-on Reset (POR)

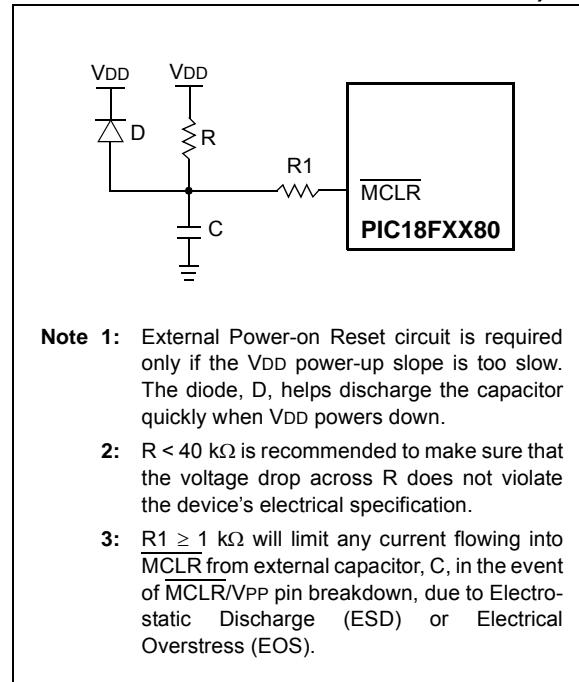
A Power-on Reset pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

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

POR events are captured by the 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. 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 VDD POWER-UP)**



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

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

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

# PIC18F66K80 FAMILY

## 5.4 Brown-out Reset (BOR)

The PIC18F66K80 family has four BOR Power modes:

- High-Power BOR
- Medium Power BOR
- Low-Power BOR
- Zero-Power BOR

Each power mode is selected by the BORPWR<1:0> setting (CONFIG2L<6:5>). For low, medium and high-power BOR, the module monitors the VDD depending on the BORV<1:0> setting (CONFIG1L<3:2>). The typical current draw ( $\Delta I_{BOR}$ ) for zero, low and medium power BOR is 200 nA, 750 nA and 3  $\mu$ A, respectively. A BOR event re-arms the Power-on Reset. It also causes a Reset, depending on which of the trip levels has been set: 1.8V, 2V, 2.7V or 3V.

BOR is enabled by BOREN<1:0> (CONFIG2L<2:1>) and the SBOREN bit (RCON<6>). The four BOR configurations are summarized in [Table 5-1](#).

In Zero-Power BOR (ZPBORMV), the module monitors the VDD voltage and re-arms the POR at about 2V. ZPBORMV does not cause a Reset, but re-arms the POR.

The BOR accuracy varies with its power level. The lower the power setting, the less accurate the BOR trip levels are. Therefore, the high-power BOR has the highest accuracy and the low-power BOR has the lowest accuracy. The trip levels (BVDD, Parameter [D005](#)), current consumption ([Section 31.2 “DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family \(Industrial/Extended\)”](#)) and time required below BVDD (TBOR, Parameter [35](#)) can all be found in [Section 31.0 “Electrical Characteristics”](#).

### 5.4.1 SOFTWARE ENABLED BOR

When BOREN<1:0> = 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’.

**TABLE 5-1: BOR CONFIGURATIONS**

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

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 BORV<1:0> Configuration bits; it cannot be changed in software.

### 5.4.2 DETECTING BOR

When Brown-out Reset is enabled, the  $\overline{BOR}$  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 BOR alone. A more reliable method is to simultaneously check the state of both POR and  $\overline{BOR}$ . This assumes that the POR bit is reset to ‘1’ in software immediately after any Power-on Reset event. If BOR is ‘0’ while 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 BOREN<1:0> = 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.

## 5.5 Configuration Mismatch (CM)

The Configuration Mismatch (CM) Reset is designed to detect, and attempt to recover from, random memory corrupting events. These include Electrostatic Discharge (ESD) events, which can cause widespread, single bit changes throughout the device and result in catastrophic failure.

In PIC18FXXKXX Flash devices, the device Configuration registers (located in the configuration memory space) are continuously monitored during operation by comparing their values to complimentary Shadow registers. If a mismatch is detected between the two sets of registers, a CM Reset automatically occurs. These events are captured by the CM bit (RCON<5>) being set to '0'.

This bit does not change for any other Reset event. A CM Reset behaves similarly to a Master Clear Reset, `RESET` instruction, WDT time-out or Stack Event Resets. As with all hard and power Reset events, the device Configuration Words are reloaded from the Flash Configuration Words in program memory as the device restarts.

## 5.6 Device Reset Timers

PIC18F66K80 family 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

### 5.6.1 POWER-UP TIMER (PWRT)

The Power-up Timer (PWRT) of the PIC18F66K80 family devices is an 11-bit counter which uses the INTOSC 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 INTOSC 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 `PWRTE` Configuration bit.

# PIC18F66K80 FAMILY

## 5.6.2 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.

## 5.6.3 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 (TPLL) is typically 2 ms and follows the oscillator start-up time-out.

## 5.6.4 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, on the other hand, there will be no time-out at all.

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, all time-outs will expire. Bringing 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.

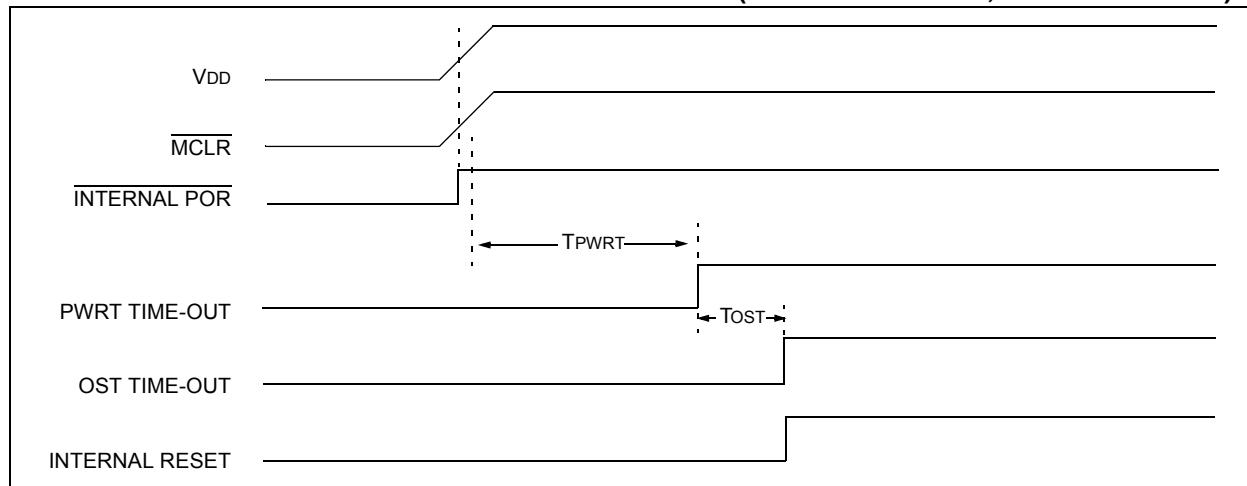
TABLE 5-2: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up and Brown-out		Exit from Power-Managed Mode
	PWRTE = 0	PWRTE = 1	
HSPLL	66 ms <sup>(1)</sup> + 1024 Tosc + 2 ms <sup>(2)</sup>	1024 Tosc + 2 ms <sup>(2)</sup>	1024 Tosc + 2 ms <sup>(2)</sup>
HS, XT, LP	66 ms <sup>(1)</sup> + 1024 Tosc	1024 Tosc	1024 Tosc
EC, ECIO	66 ms <sup>(1)</sup>	—	—
RC, RCIO	66 ms <sup>(1)</sup>	—	—
INTIO1, INTIO2	66 ms <sup>(1)</sup>	—	—

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

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

FIGURE 5-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD, VDD RISE < TPWRT)



# PIC18F66K80 FAMILY

FIGURE 5-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

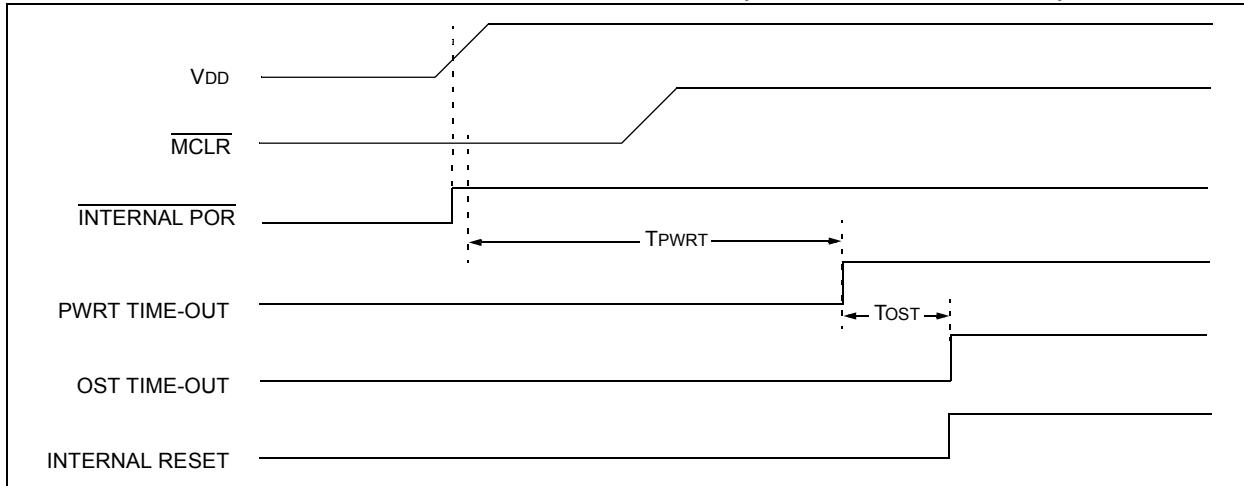


FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

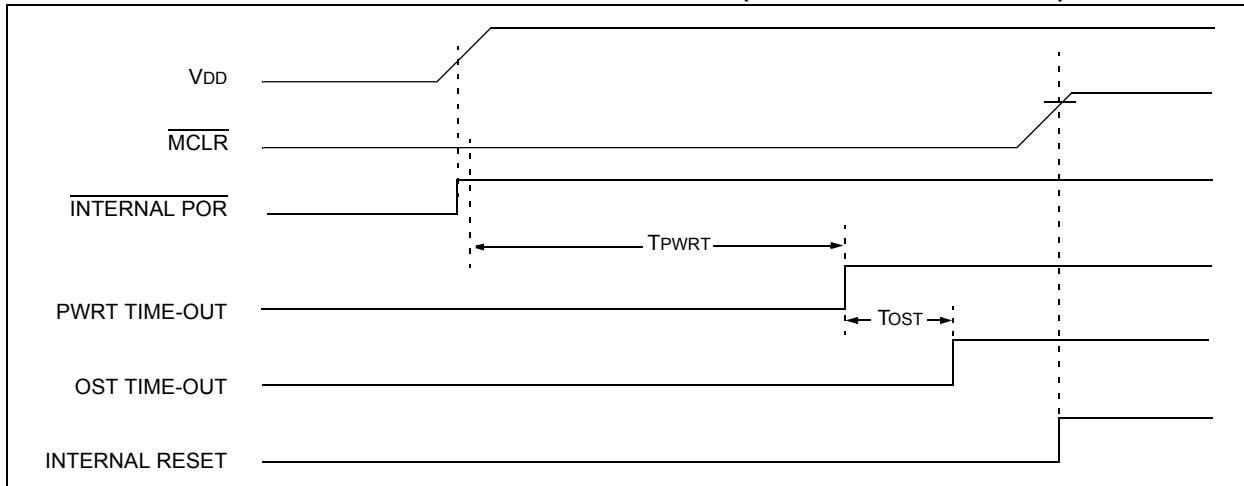
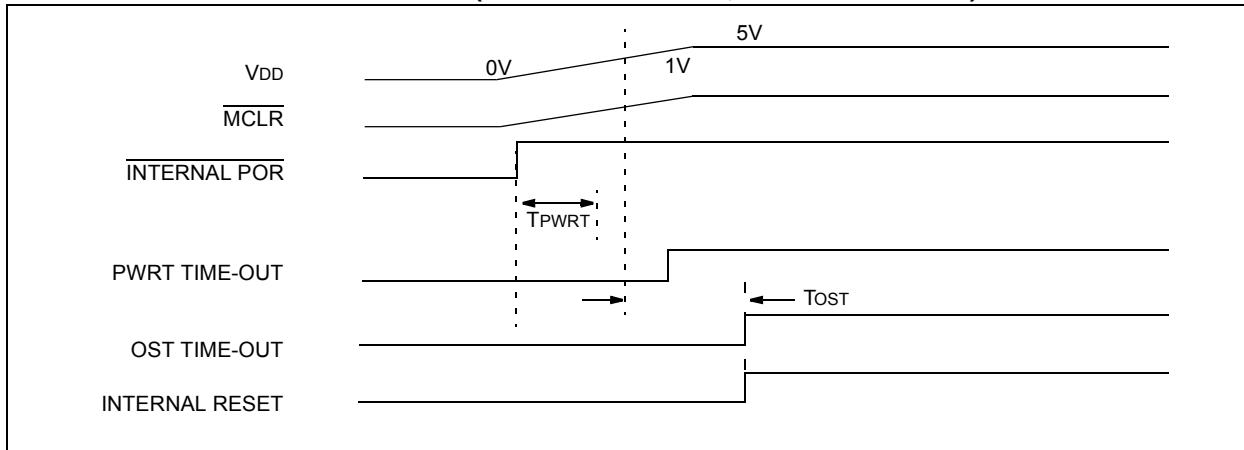
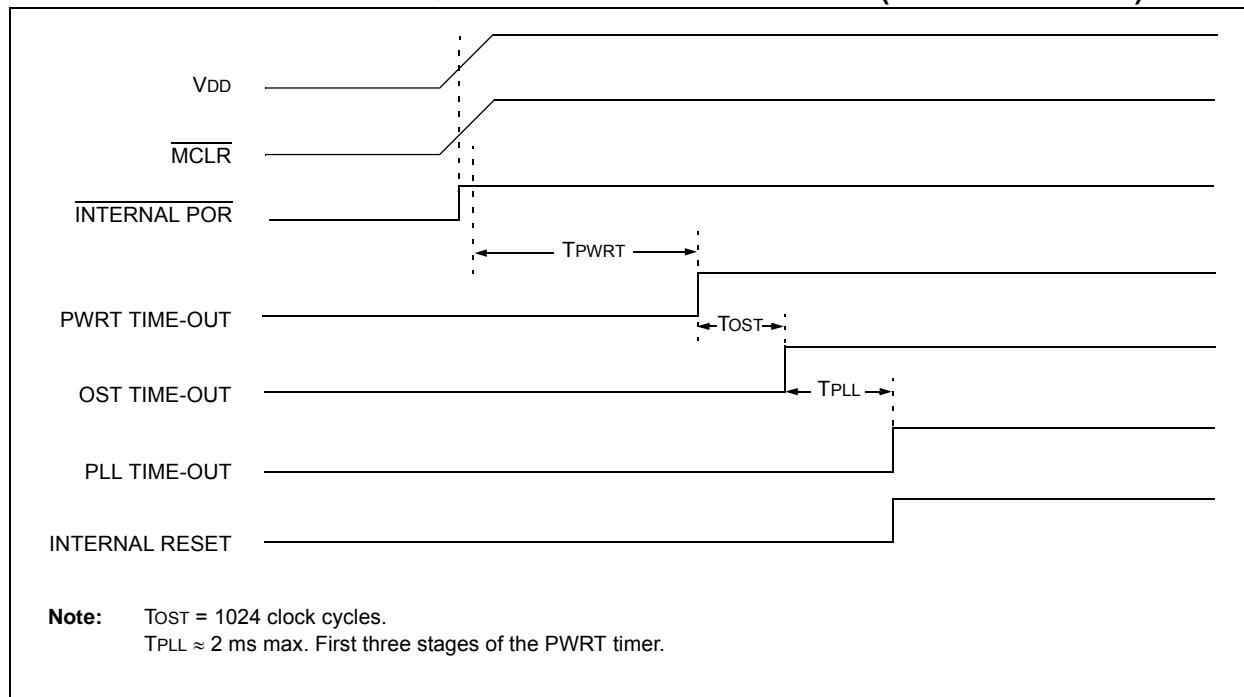


FIGURE 5-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)



# PIC18F66K80 FAMILY

FIGURE 5-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED (MCLR TIED TO VDD)



## 5.7 Reset State of Registers

Most registers are unaffected by a Reset. Their status is unknown on a Power-on Reset 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,  $\overline{RI}$ ,  $\overline{TO}$ ,  $\overline{PD}$ ,  $\overline{CM}$ ,  $\overline{POR}$  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 <sup>(1)</sup>	RCON Register							STKPTR Register	
		SBOREN	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	STKFUL	STKUNF
Power-on Reset	0000h	1	1	1	1	1	0	0	0	0
RESET Instruction	0000h	u <sup>(2)</sup>	u	0	u	u	u	u	u	u
Brown-out Reset	0000h	u <sup>(2)</sup>	1	1	1	1	u	0	u	u
MCLR Reset during Power-Managed Run modes	0000h	u <sup>(2)</sup>	u	u	1	u	u	u	u	u
MCLR Reset during Power-Managed Idle modes and Sleep mode	0000h	u <sup>(2)</sup>	u	u	1	0	u	u	u	u
WDT Time-out during Full Power or Power-Managed Run modes	0000h	u <sup>(2)</sup>	u	u	0	u	u	u	u	u
MCLR Reset during Full-Power execution	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	u	u
Stack Full Reset (STVREN = 1)	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	1	u
Stack Underflow Reset (STVREN = 1)	0000h	u <sup>(2)</sup>	u	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	u	1
WDT Time-out during Power-Managed Idle or Sleep modes	PC + 2	u <sup>(2)</sup>	u	u	0	0	u	u	u	u
Interrupt Exit from Power-Managed modes	PC + 2	u <sup>(2)</sup>	u	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 bits are 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 (BOREN<1:0>, Configuration bits = 01 and SBOREN = 1); otherwise, the Reset state is ‘0’.

# PIC18F66K80 FAMILY

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	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---0 0000	---0 0000	---0 uuuu <sup>(3)</sup>
TOSH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
TOSL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu <sup>(3)</sup>
STKPTR	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	00-0 0000	uu-0 0000	uu-u uuuu <sup>(3)</sup>
PCLATU	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---0 0000	---0 0000	---u uuuu
PCLATH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
PCL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	PC + 2 <sup>(2)</sup>
TBLPTRU	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--00 0000	--00 0000	--uu uuuu
TBLPTRH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
TABLAT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
PRODH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 000x	0000 000u	uuuu uuuu <sup>(1)</sup>
INTCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 1111	1111 -1-1	uuuu -u-u <sup>(1)</sup>
INTCON3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1100 0000	11x0 0x00	uuuu uuuu <sup>(1)</sup>
INDF0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
POSTINC0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
POSTDEC0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
PREINC0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
PLUSW0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
FSR0H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---- xxxx	---- uuuu	---- uuuu
FSR0L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
POSTINC1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
POSTDEC1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
PREINC1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
PLUSW1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
FSR1H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---- xxxx	---- uuuu	---- uuuu
FSR1L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---- 0000	---- 0000	---- uuuu
INDF2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A

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

Shaded cells indicate conditions do not apply for the designated device.

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4:** See Table 5-3 for Reset value for specific conditions.
- 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'.

# PIC18F66K80 FAMILY

**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
POSTINC2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
POSTDEC2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
PREINC2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
PLUSW2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	N/A	N/A	N/A
FSR2H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---- xxxx	---- uuuu	---- uuuu
FSR2L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---x xxxx	---u uuuu	---u uuuu
TMR0H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	uuuu uuuu	uuuu uuuu
TMR0L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 1111	1111 1111	uuuu uuuu
OSCCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0110 q000	0100 00q0	uuuu uuqu
OSCCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-0-1 0-x0	-0-0 0-01	-u-u u-uu
WDTCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0-x0 -xx0	0-x0 -xx0	u-u0 -uu0
RCON <sup>(4)</sup>	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0111 11q0	0111 qquu	uuuu qquu
TMR1H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	u0uu uuuu	uuuu uuuu
TMR2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
PR2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 1111	1111 1111	uuuu uuuu
T2CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-000 0000	-000 0000	-uuu uuuu
SSPBUF	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
SSPCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
SSPCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
ADRESH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-000 0000	-000 0000	-uuu uuuu
ADCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 xxxx	0000 0qqq	uuuu uuuu
ADCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0-00 0000	0-00 0000	u-uu uuuu
ECCP1AS	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	xxxx xxxx
CCPR1H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
TXSTA2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0010	0000 0010	uuuu uuuu

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

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

**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'.

# PIC18F66K80 FAMILY

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
BAUDCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	01x0 0-00	01x0 0-00	uuuu u-uu
IPR4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 -111	1111 -111	uuuu -uuu
PIR4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 -000	0000 -000	uuuuu -uuuu
PIE4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 -000	0000 -000	uuuuu -uuuu
CVRCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
CMSTAT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx-- ----	xx-- ----	uu-- ----
TMR3H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuuu uuuuu	uuuuu uuuuu
TMR3L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuuu uuuuu	uuuuu uuuuu
T3CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
T3GCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0x00	0000 0x00	uuuuu u-uu
SPBRG1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
RCREG1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
TXREG1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
TXSTA1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0010	0000 0010	uuuuu uuuuu
RCSTA1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 000x	0000 000x	uuuuu uuuuu
T1GCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0x00	0000 0x00	uuuuu u-uu
PR4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 1111	1111 1111	uuuuu uuuuu
HLVDCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
BAUDCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	01x0 0-00	01x0 0-00	uuuuu u-uu
RCSTA2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 000x	0000 000x	uuuuu uuuuu
IPR3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--11 111-	--11 111-	--uuu uuuu-
PIR3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--00 000-	--x0 xxx-	--uuu uuuu-
PIE3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--00 000-	0000 0000	uuuuu uuuuu
IPR2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1--- 1111	1--- 111x	u--- uuuuu
PIR2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0--- 0000	0--- 000x	u--- uuuuu <sup>(1)</sup>
PIE2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0--- 0000	0--- 0000	u--- uuuuu
IPR1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 1111	1111 1111	uuuuu uuuuu
	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-111 1111	-111 1111	-uuuu uuuuu
PIR1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu <sup>(1)</sup>
	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-000 0000	-000 0000	-uuuu uuuuu
PIE1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-000 0000	-000 0000	-uuuu uuuuu
PSTR1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	00-0 0001	xx-x xxxx	—
OSCTUNE	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu
REFOCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0-00 0000	0-00 0000	u-uu uuuuu

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

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

**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'.

# PIC18F66K80 FAMILY

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
CCPTMRS	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---	0 0000	---x xxxx	---u uuuu
TRISG	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---	1 1111	---1 1111	---u uuuu
TRISF	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111	1111	1111 1111	uuuu uuuu
TRISE	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111	-111	1111 -111	uuuu -uuu
TRISD	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111	1111	1111 1111	uuuu uuuu
TRISC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111	1111	1111 1111	uuuu uuuu
TRISB	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111	1111	1111 1111	uuuu uuuu
TRISA <sup>(5)</sup>	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	111-	1111 <sup>(5)</sup>	111- 1111 <sup>(5)</sup>	uuu- uuuu <sup>(5)</sup>
ODCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
SLRCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-000	0000	-111 1111	-111 1111
LATG	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---	x xxxx	---	u uuuu
LATF	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx	xxxx	xxxx -xxx	uuuu -uuu
LATE	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	-xxx	xxxxx xxxx	uuuu uuuu
LATD	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	xxxxx	xxxxx xxxx	uuuu uuuu
LATC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	xxxxx	xxxxx xxxx	uuuu uuuu
LATB	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	xxxxx	xxxxx xxxx	uuuu uuuu
LATA <sup>(5)</sup>	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx-	xxxx <sup>(5)</sup>	xxx- xxxx <sup>(5)</sup>	uuu- uuuu <sup>(5)</sup>
T4CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-000	0000	-000 0000	-uuu uuuu
TMR4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
PORTG	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---	x xxxx	---	u uuuu
PORTF	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx	xxxx	xxxxx xxxx	uuuu uuuu
PORTE	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	xxxxx	xxxxx xxxx	uuuu uuuu
PORTD	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	xxxxx	xxxxx xxxx	uuuu uuuu
PORTC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	xxxxx	xxxxx xxxx	uuuu uuuu
PORTB	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx	xxxxx	xxxxx xxxx	uuuu uuuu
PORTA <sup>(5)</sup>	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx-	xxxx <sup>(5)</sup>	xxx- xxxx <sup>(5)</sup>	uuu- uuuu <sup>(5)</sup>
EECON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx-0	x000	uu-0 u000	uu-u uuuu
EECON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
SPBRGH1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
SPBRGH2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
SPBRG2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
RCREG2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
TXREG2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu
IPR5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111	1111	1111 1111	uuuu uuuu
PIR5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000	0000	0000 0000	uuuu uuuu

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

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

**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'.

# PIC18F66K80 FAMILY

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
PIE5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
EEADDRH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---- --00	---- --00	---- --00
EEADR	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
EEDATA	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
ECANCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 0000	0001 0000	uuuu uuuu
COMSTAT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CIOCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 ---0	0000 ---0	uuuu ---u
CANCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
CANSTAT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
RXB0D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu
RXB0SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CM1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 1111	0001 1111	uuuu uuuu
CM2CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 1111	0001 1111	uuuu uuuu
ANCON0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1111 1111	1111 1111	uuuu uuuu
ANCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-111 1111	-111 1111	-uuu uuuu
WPUB	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	1111 1111	uuuu uuuu
IOCB	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 ----	0000 ----	uuuu ----
PMDO	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu

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

Shaded cells indicate conditions do not apply for the designated device.

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
  - 3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
  - 4: See [Table 5-3](#) for Reset value for specific conditions.
  - 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'.

# PIC18F66K80 FAMILY

**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
PMD1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu	
PMD2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---- 0000	---- 0000	---- uuuu	
PADCFG1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 ---0	0000 ---0	uuuu ---u	
CTMUCONH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0-00 0000	0-00 0000	u-uu uuuu	
CTMUCONL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu	
CTMUICON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu	
CCPR2H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCPR2L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCP2CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--00 0000	--00 0000	--uu uuuu	
CCPR3H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCPR3L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCP3CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--00 0000	--00 0000	--uu uuuu	
CCPR4H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCPR4L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCP4CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--00 0000	--00 0000	--uu uuuu	
CCPR5H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCPR5L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	
CCP5CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	--00 0000	--00 0000	--uu uuuu	
PSPCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 ----	0000 ----	uuuu ----	
MDCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0010 0--0	0010 0--0	uuuu u--u	
MDSRC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0--- xxxx	0--- xxxx	u--- uuuu	
MDCARH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0xx- xxxx	0xx- xxxx	uuu- uuuu	
MDCARL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0xx- xxxx	0xx- xxxx	uuu- uuuu	
CANCON_RO0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	
CANSTAT_RO0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	
RXB1D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXB1DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	xxxx xxxx	

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

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
  - 3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
  - 4:** See [Table 5-3](#) for Reset value for specific conditions.
  - 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'.

# PIC18F66K80 FAMILY

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
RXB1EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXB1EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXB1SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu	uuuu u-uu
RXB1SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXB1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu
CANCON_RO1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
CANSTAT_RO1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
TXB0D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB0DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	xxx- x-xx	uuu- u-uu	uuu- u-uu
TXB0SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0-00	0000 0-00	uuuu u-uu	uuuu u-uu
CANCON_RO2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
CANSTAT_RO2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
TXB1D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TXB1SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu	uuu- u-uu

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

Shaded cells indicate conditions do not apply for the designated device.

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

- 2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4: See [Table 5-3](#) for Reset value for specific conditions.
- 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'.

# PIC18F66K80 FAMILY

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
TXB1SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0-00	0000 0-00	uuuu u-uu	
CANCON_RO3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	
CANSTAT_RO3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	
TXB2D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- -x-xx	uuu- u-uu	uuu- u-uu	
TXB2SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
TXB2CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0-00	0000 0-00	uuuu u-uu	
RXM1EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXM1EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXM1SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- -0-xx	uuu- u-uu	uuu- u-uu	
RXM1SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXM0EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXM0EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXM0SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- -0-xx	uuu- u-uu	uuu- u-uu	
RXM0SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXF5EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXF5EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXF5SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- -x-xx	uuu- u-uu	uuu- u-uu	
RXF5SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXF4EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXF4EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXF4SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- -x-xx	uuu- u-uu	uuu- u-uu	
RXF4SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	
RXF3EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

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

**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'.

# PIC18F66K80 FAMILY

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
RXF3EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF3SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu	uuu- u-uu
RXF3SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF2EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF2EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF2SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu	uuu- u-uu
RXF2SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF1EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF1EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF1SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu	uuu- u-uu
RXF1SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF0EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF0EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RXF0SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu	uuu- u-uu
RXF0SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
CANCON_RO4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
CANSTAT_RO4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
B5D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	uuuu uuuu	uuuu uuuu
B5EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu	uuu- u-uu
B5SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B5CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu
CANCON_RO5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
CANSTAT_RO5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu	uuuu uuuu
B4D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
B4D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

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

Shaded cells indicate conditions do not apply for the designated device.

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

- 2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4: See [Table 5-3](#) for Reset value for specific conditions.
- 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'.

# PIC18F66K80 FAMILY

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
B4D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	-uuu uuuu	
B4EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx x-xx	uuuuu u-uu	uuuuu u-uu	
B4SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B4CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
CANCON_RO6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuuu uuuuu	
CANSTAT_RO6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuuu uuuuu	
B3D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	-uuu uuuu	
B3EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx x-xx	uuuuu u-uu	uuuuu u-uu	
B3SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B3CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
CANCON_RO7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuuu uuuuu	
B2D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B2D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B2D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B2D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B2D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B2D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

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

**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'.

# PIC18F66K80 FAMILY

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
B2D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	-uuu uuuu
B2EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu
B2SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CANCON_RO8	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
B1D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	-uuu uuuu
B1EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu
B1SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CANCON_RO9	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
CANSTAT_RO	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
B0D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	-uuu uuuu
B0EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu

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

Shaded cells indicate conditions do not apply for the designated device.

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

- 2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4: See [Table 5-3](#) for Reset value for specific conditions.
- 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'.

# PIC18F66K80 FAMILY

**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
B0EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B0SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx x-xx	uuuuu u-uu	uuuuu u-uu	
B0SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
B0CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
TXBIE	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---0 00--	---u uu--	---u uu--	
BIE0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
BSEL0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 00--	0000 00--	uuuuu uu--	
MSEL3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
MSEL2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
MSEL1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0101	0000 0101	uuuuu uuuuu	
MSEL0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0101 0000	0101 0000	uuuuu uuuuu	
RXFBCON7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
RXFBCON6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
RXFBCON5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
RXFBCON4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
RXFBCON3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
RXFBCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 0001	0001 0001	uuuuu uuuuu	
RXFBCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0001 0001	0001 0001	uuuuu uuuuu	
RXFBCON0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuuu uuuuu	
SDFLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	---0 0000	---0 0000	---u uuuu	
RXF15EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF15EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF15SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu	
RXF15SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF14EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF14EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF14SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu	
RXF14SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF13EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF13EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF13SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu	
RXF13SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF12EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF12EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxxx xxxx	uuuuu uuuuu	uuuuu uuuuu	
RXF12SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxx- x-xx	uuu- u-uu	uuu- u-uu	

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

**4:** See [Table 5-3](#) for Reset value for specific conditions.

**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'.

# PIC18F66K80 FAMILY

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
RXF12SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu
RXF11SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF10EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF10EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF10SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu
RXF10SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF9EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF9EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF9SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu
RXF9SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF8EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF8EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF8SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu
RXF8SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF7EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF7EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF7SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu
RXF7SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF6EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF6EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF6SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xx- x-xx	uuu- u-uu	uuu- u-uu
RXF6SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXFCON0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
RXFCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
BRGCON3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	00-- -000	00-- -000	uu-- -uuu
BRGCON2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
BRGCON1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
TXERRCNT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
RXERRCNT	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

**4:** See [Table 5-3](#) for Reset value for specific conditions.

**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'.

## 6.0 MEMORY ORGANIZATION

PIC18F66K80 family devices have these types of memory:

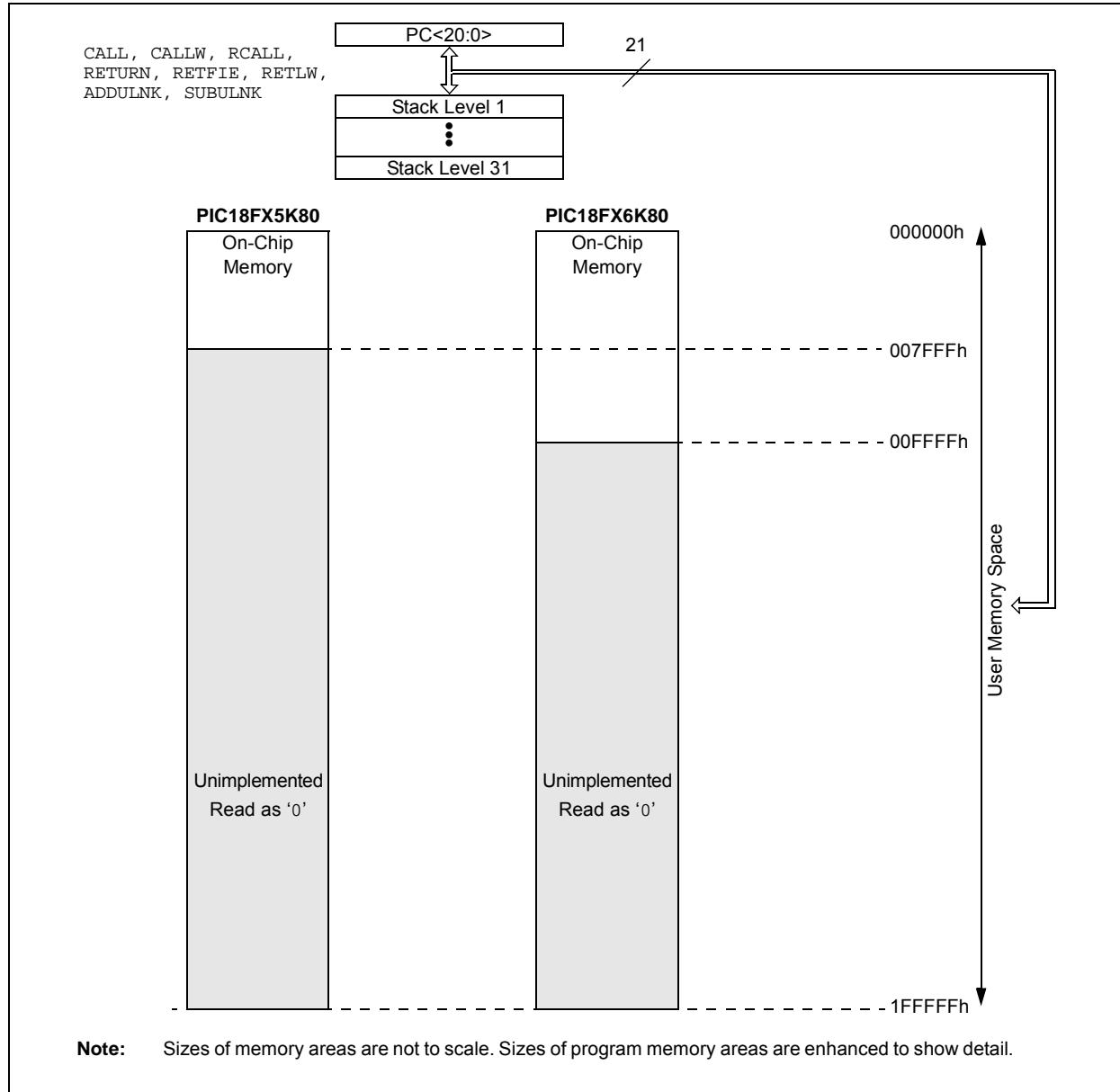
- Program Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate busses. This enables concurrent access of the two memory spaces.

The data EEPROM, for practical purposes, can be regarded as a peripheral device because 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”](#). The data EEPROM is discussed separately in [Section 8.0 “Data EEPROM Memory”](#).

**FIGURE 6-1: MEMORY MAPS FOR PIC18F66K80 FAMILY DEVICES**



# PIC18F66K80 FAMILY

## 6.1 Program Memory Organization

PIC18 microcontrollers implement a 21-bit Program Counter (PC) that 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 entire PIC18F66K80 family offers a range of on-chip Flash program memory sizes, from 32 Kbytes (16,384 single-word instructions) to 64 Kbytes (32,768 single-word instructions).

- PIC18F25K80, PIC18F45K80 and PIC18F65K80 – 32 Kbytes of Flash memory, storing up to 16,384 single-word instructions
- PIC18F26K80, PIC18F46K80 and PIC18F66K80 – 64 Kbytes of Flash memory, storing up to 32,768 single-word instructions

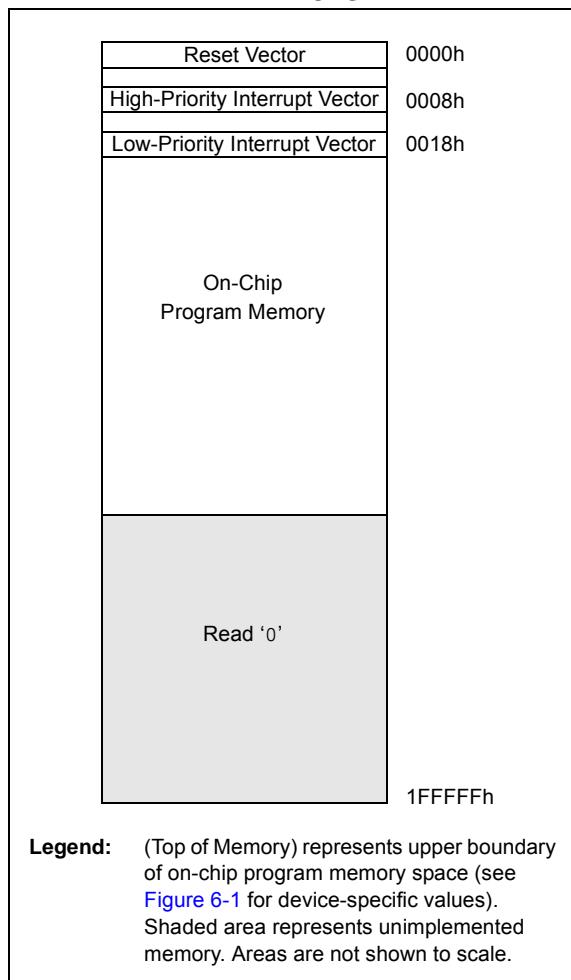
The program memory maps for individual family members are shown in [Figure 6-1](#).

### 6.1.1 HARD MEMORY VECTORS

All PIC18 devices have a total of three hard-coded return vectors in their program memory space. The Reset vector address is the default value to which the Program Counter returns on all device Resets. It is located at 0000h.

PIC18 devices also have two interrupt vector addresses for handling high-priority and low-priority interrupts. The high-priority interrupt vector is located at 0008h and the low-priority interrupt vector is at 0018h. The locations of these vectors are shown, in relation to the program memory map, in [Figure 6-2](#).

**FIGURE 6-2: HARD VECTOR FOR PIC18F66K80 FAMILY DEVICES**



## 6.1.2 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and 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<sub><15:8></sub> bits and 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<sub><20:16></sub> 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 PCL. Similarly, the upper two bytes of the Program Counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see [Section 6.1.5.1 "Computed GOTO"](#)).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit (LSb) of PCL is fixed to a value of '0'. The PC increments by two 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.

## 6.1.3 RETURN ADDRESS STACK

The return address stack enables execution of any combination of up to 31 program calls and interrupts. 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. The value is also pulled off the stack on ADDULNK and SUBULNK instructions if the extended instruction set is enabled. 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 (TOS) 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.

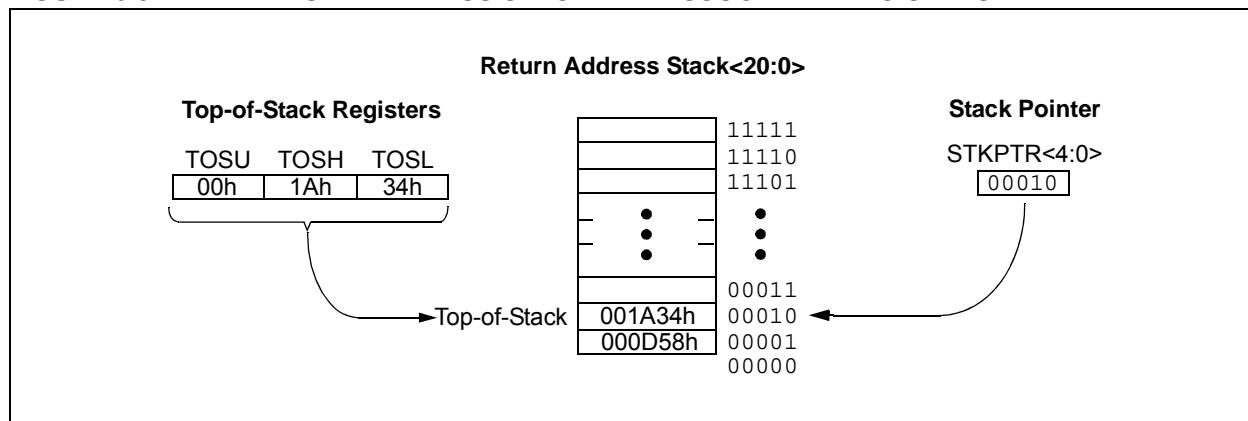
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

### 6.1.3.1 Top-of-Stack Access

Only the top of the return address stack is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register ([Figure 6-3](#)). This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt (or ADDULNK and SUBULNK instructions, if the extended instruction set is enabled), 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.

While accessing the stack, users must disable the Global Interrupt Enable bits to prevent inadvertent stack corruption.

**FIGURE 6-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



# PIC18F66K80 FAMILY

## 6.1.3.2 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 of 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.

What happens when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (For a description of the device Configuration bits, see [Section 28.1 “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 the 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 set 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.

## 6.1.3.3 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 of 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	U = Unimplemented bit, read as ‘0’
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared      x = Bit is unknown

bit 7	<b>STKFUL:</b> Stack Full Flag bit <sup>(1)</sup>
	1 = Stack has become full or overflowed
	0 = Stack has not become full or overflowed
bit 6	<b>STKUNF:</b> Stack Underflow Flag bit <sup>(1)</sup>
	1 = Stack underflow has occurred
	0 = Stack underflow did not occur
bit 5	<b>Unimplemented:</b> Read as ‘0’
bit 4-0	<b>SP&lt;4:0&gt;:</b> Stack Pointer Location bits

**Note 1:** Bit 7 and bit 6 are cleared by user software or by a POR.

### 6.1.3.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit (CONFIG4L<0>). When STVREN is set, a full or underflow condition 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 bits are cleared by the user software or a Power-on Reset.

### 6.1.4 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers to provide a “fast return” option for interrupts. This stack 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 the working 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
```

### 6.1.5 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

#### 6.1.5.1 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 two (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

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

#### 6.1.5.2 Table Reads

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 while programming. The Table Pointer (TBLPTR) specifies the byte address and the Table Latch (TABLAT) contains the data that is read from the program memory. Data is transferred from program memory one byte at a time.

The table read operation is discussed further in **Section 7.1 “Table Reads and Table Writes”**.

# PIC18F66K80 FAMILY

## 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, with the instruction 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 6-4](#).

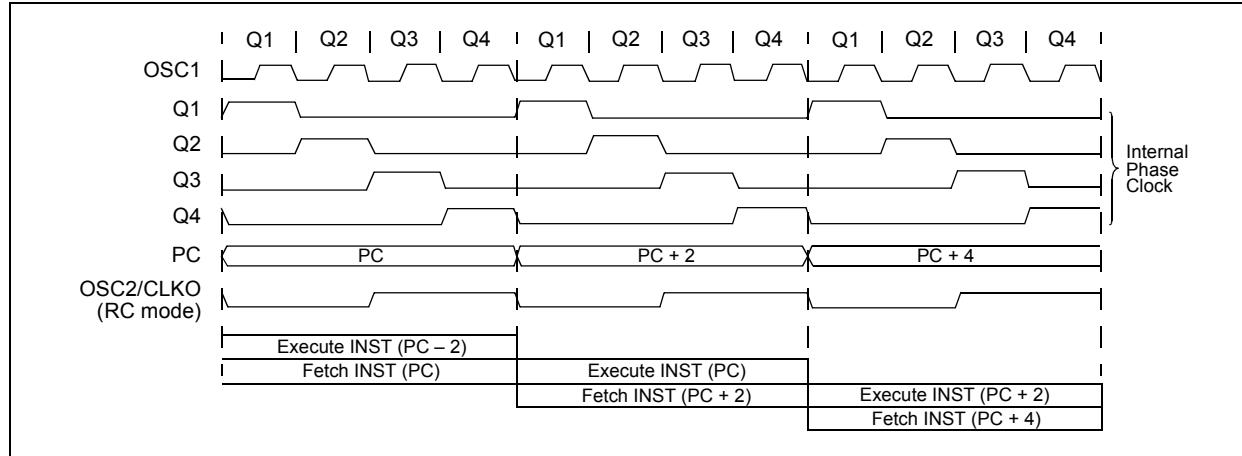
### 6.2.2 INSTRUCTION FLOW/PIPELINING

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 (such as GOTO) causes the Program Counter to change, two cycles are required to complete the instruction. (See [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 6-4:** CLOCK/INSTRUCTION CYCLE



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

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

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

### 6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two or four bytes in program memory. The Least Significant Byte (LSB) 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 two and the LSB will always read '0' (see [Section 6.1.2 "Program Counter"](#)).

[Figure 6-5](#) 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 6-5](#) 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. For more details on the instruction set, see [Section 29.0 "Instruction Set Summary"](#).

**FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →		Word Address ↓	
LSB = 1	LSB = 0		
	000000h		
	000002h		
	000004h		
	000006h		
0Fh	55h	000008h	
EFh	03h	0000Ah	
F0h	00h	0000Ch	
C1h	23h	0000Eh	
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 LSFR. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits (MSbs). 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:** For information on two-word instructions in the extended instruction set, see [Section 6.5 "Program Memory and 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

# PIC18F66K80 FAMILY

---

## 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 6.6 “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 4,096 bytes of data memory. The memory space is divided into 16 banks that contain 256 bytes each.

[Figure 6-6](#) and [Figure 6-7](#) show the data memory organization for the 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 section.

To ensure that commonly used registers (select 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 select SFRs and the lower portion of GPR Bank 0 without using the Bank Select Register. For details on the Access RAM, see [Section 6.3.2 “Access Bank”](#).

### 6.3.1 BANK SELECT REGISTER

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 eight-bit, low-order address and a four-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 ( $\text{BSR}\langle 3:0 \rangle$ ). The upper four bits are unused and always read as '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 eight 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 6-7](#).

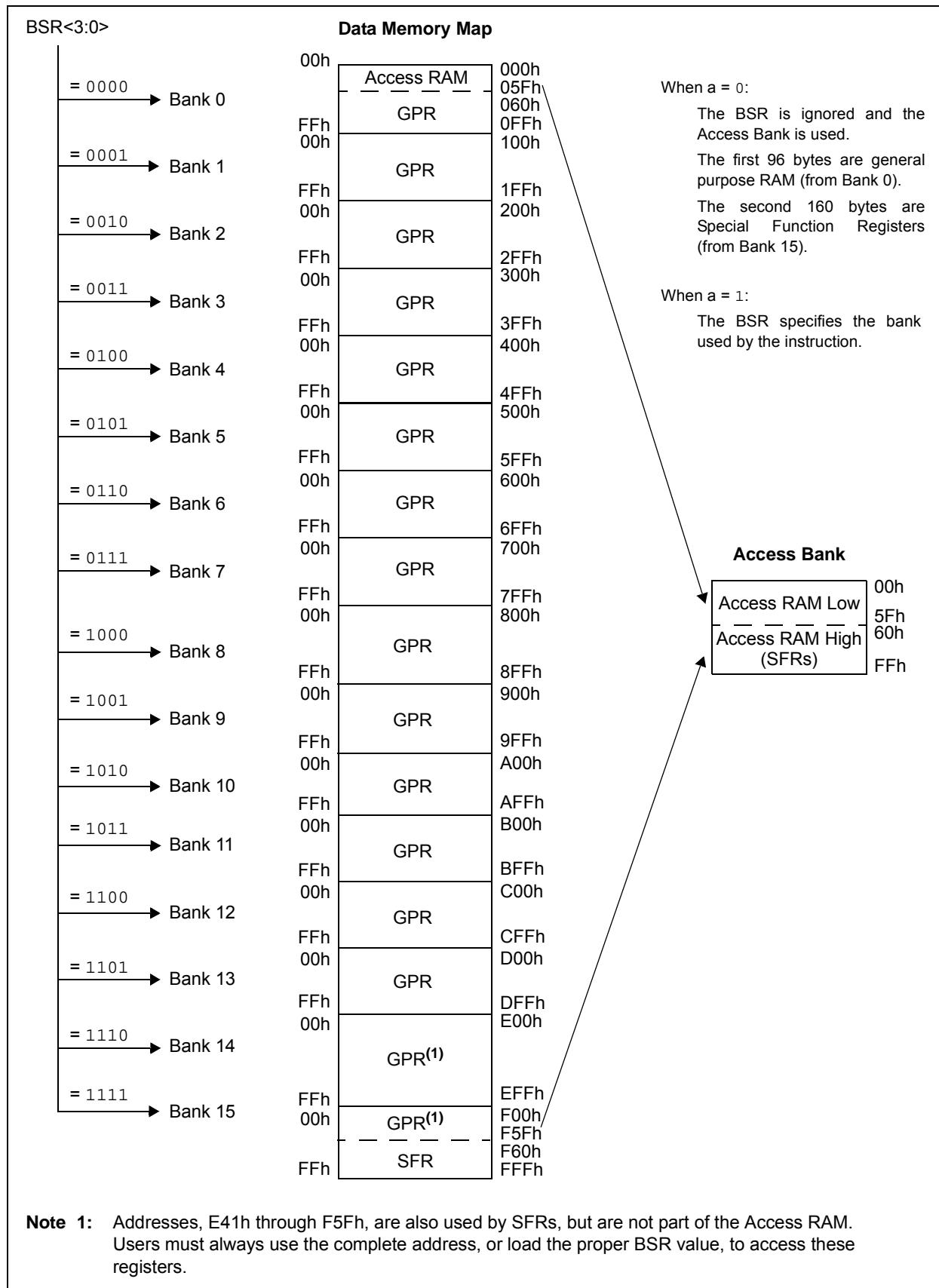
Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an eight-bit address of F9h while the BSR is 0Fh, will end up resetting 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 6-6](#) 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. When this instruction executes, it ignores the BSR completely. 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.

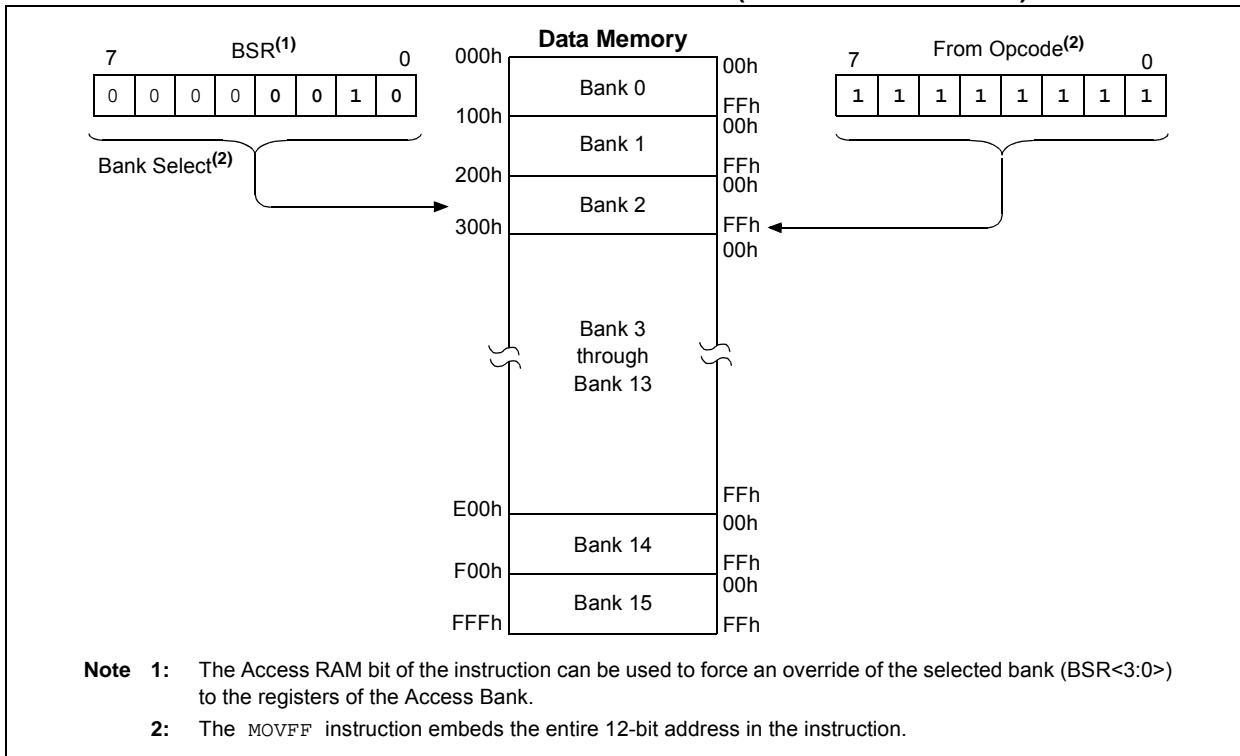
# PIC18F66K80 FAMILY

**FIGURE 6-6: DATA MEMORY MAP FOR PIC18FX5K80 AND PIC18FX6K80 DEVICES**



# PIC18F66K80 FAMILY

FIGURE 6-7: 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 ensure that the correct bank is selected. If not, 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 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Bank 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 eight-bit address (Figure 6-6).

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. In that case, 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 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h 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 6.6.3 "Mapping the Access Bank in Indexed Literal Offset 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.

### 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 all of Bank 15 (F00h to FFFh) and the top part of Bank 14 (EF4h to EFFh).

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 the peripheral features 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 PIC18F66K80 FAMILY**

Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name
FFFh	TOSU	FDFh	INDF2 <sup>(1)</sup>	FBFh	ECCP1AS	F9Fh	IPR1	F7Fh	EECON1	F5Fh	CM1CON <sup>(5)</sup>
FFEh	TOSH	FDEh	POSTINC2 <sup>(1)</sup>	FBEh	ECCP1DEL	F9Eh	PIR1	F7Eh	EECON2	F5Eh	CM2CON <sup>(5)</sup>
FFDh	TOSL	FDDh	POSTDEC2 <sup>(1)</sup>	FBDh	CCPR1H	F9Dh	PIE1	F7Dh	SPBRGH1	F5Dh	ANCON0 <sup>(5)</sup>
FFCh	STKPTR	FDCh	PREINC2 <sup>(1)</sup>	FBCh	CCPR1L	F9Ch	PSTR1CON	F7Ch	SPBRGH2	F5Ch	ANCON1 <sup>(5)</sup>
FFBh	PCLATU	FDBh	PLUSW2 <sup>(1)</sup>	FBBh	CCP1CON	F9Bh	OSCTUNE	F7Bh	SPBREG2	F5Bh	WPUB <sup>(5)</sup>
FFAh	PCLATH	FDAh	FSR2H	FBAh	TXSTA2	F9Ah	REFOCON	F7Ah	RCREG2	F5Ah	IOCB <sup>(5)</sup>
FF9h	PCL	FD9h	FSR2L	FB9h	BAUDCON2	F99h	CCPTMRS	F79h	TXREG2	F59h	PMD0 <sup>(5)</sup>
FF8h	TBLPTRU	FD8h	STATUS	FB8h	IPR4	F98h	TRISG <sup>(3)</sup>	F78h	IPR5	F58h	PMD1 <sup>(5)</sup>
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PIR4	F97h	TRISE <sup>(3)</sup>	F77h	PIR5	F57h	PMD2 <sup>(5)</sup>
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	PIE4	F96h	TRISE <sup>(4)</sup>	F76h	PIE5	F56h	PADCFG1 <sup>(5)</sup>
FF5h	TABLAT	FD5h	TOCON	FB5h	CVRCON	F95h	TRISD <sup>(4)</sup>	F75h	EEADDRH	F55h	CTMUCONH <sup>(5)</sup>
FF4h	PRODH	FD4h	— <sup>(2)</sup>	FB4h	CMSTAT	F94h	TRISC	F74h	EEADR	F54h	CTMUCONL <sup>(5)</sup>
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	EEDATA	F53h	CTMUICONH <sup>(5)</sup>
FF2h	INTCON	FD2h	OSCCON2	FB2h	TMR3L	F92h	TRISA	F72h	ECANCON	F52h	CCPR2H <sup>(5)</sup>
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	ODCON	F71h	COMSTAT	F51h	CCPR2L <sup>(5)</sup>
FF0h	INTCON3	FD0h	RCON	FB0h	T3GCON	F90h	SLRCON	F70h	CIOCON	F50h	CCP2CON <sup>(4,5)</sup>
FEFh	INDF0 <sup>(1)</sup>	FCFh	TMR1H	FAFh	SPBRG1	F8Fh	LATG <sup>(3)</sup>	F6Fh	CANCON	F4Fh	CCPR3H <sup>(4,5)</sup>
FEEh	POSTINC0 <sup>(1)</sup>	FCEh	TMR1L	FAEh	RCREG1	F8Eh	LATF <sup>(3)</sup>	F6Eh	CANSTAT	F4Eh	CCPR3L <sup>(4,5)</sup>
FEDh	POSTDEC0 <sup>(1)</sup>	FCDh	T1CON	FADh	TXREG1	F8Dh	LATE <sup>(4)</sup>	F6Dh	RXB0D7	F4Dh	CCP3CON <sup>(5)</sup>
FECh	PREINC0 <sup>(1)</sup>	FCCh	TMR2	FACH	TXSTA1	F8Ch	LATD <sup>(4)</sup>	F6Ch	RXB0D6	F4Ch	CCPR4H <sup>(5)</sup>
FEBh	PLUSW0 <sup>(1)</sup>	FCBh	PR2	FABh	RCSTA1	F8Bh	LATC	F6Bh	RXB0D5	F4Bh	CCPR4L <sup>(5)</sup>
FEAh	FSR0H	FCAh	T2CON	FAAh	T1GCON	F8Ah	LATB	F6Ah	RXB0D4	F4Ah	CCP4CON <sup>(5)</sup>
FE9h	FSR0L	FC9h	SSPBUF	FA9h	PR4	F89h	LATA	F69h	RXB0D3	F49h	CCPR5H <sup>(5)</sup>
FE8h	WREG	FC8h	SSPADD	FA8h	HLVDCON	F88h	T4CON	F68h	RXB0D2	F48h	CCPR5L <sup>(5)</sup>
FE7h	INDF1 <sup>(1)</sup>	FC8h	SSPMISK	FA7h	BAUDCON1	F87h	TMR4	F67h	RXB0D1	F47h	CCP5CON <sup>(5)</sup>
FE6h	POSTINC1 <sup>(1)</sup>	FC7h	SSPSTAT	FA6h	RCSTA2	F86h	PORTG <sup>(3)</sup>	F66h	RXB0D0	F46h	PSPCON <sup>(4,5)</sup>
FE5h	POSTDEC1 <sup>(1)</sup>	FC6h	SSPCON1	FA5h	IPR3	F85h	PORTF <sup>(3)</sup>	F65h	RXB0DLC	F45h	MDCON <sup>(3,5)</sup>
FE4h	PREINC1 <sup>(1)</sup>	FC5h	SSPCON2	FA4h	PIR3	F84h	PORTE	F64h	RXB0EIDL	F44h	MDSRC <sup>(3,5)</sup>
FE3h	PLUSW1 <sup>(1)</sup>	FC4h	ADRESH	FA3h	PIE3	F83h	PORTD <sup>(4)</sup>	F63h	RXB0EIDH	F43h	MDCARH <sup>(3,5)</sup>
FE2h	FSR1H	FC3h	ADRESL	FA2h	IPR2	F82h	PORTC	F62h	RXB0SIDL	F42h	MDCARL <sup>(3,5)</sup>
FE1h	FSR1L	FC2h	ADCON0	FA1h	PIR2	F81h	PORTB	F61h	RXB0SIDH	F41h	— <sup>(2)</sup>
FE0h	BSR	FC1h	ADCON1	FA0h	PIE2	F80h	PORTA	F60h	RXB0CON	F40h	— <sup>(2)</sup>
		FC0h	ADCON2								

**Note 1:** This is not a physical register.

**2:** Unimplemented registers are read as ‘0’.

**3:** This register is only available on devices with 64 pins.

**4:** This register is not available on devices with 28 pins.

**5:** Addresses, E41h through F5Fh, are also used by the SFRs, but are not part of the Access RAM. To access these registers, users must always load the proper BSR value.

# PIC18F66K80 FAMILY

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F66K80 FAMILY (CONTINUED)**

Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name
F3Fh	CANCON_RO0 <sup>(5)</sup>	F0Fh	CANCON_RO3 <sup>(5)</sup>	EDFh	CANCON_RO4 <sup>(5)</sup>	EAFh	CANCON_RO7 <sup>(5)</sup>	E7Fh	TXBIE <sup>(5)</sup>	E4Fh	RXF7EIDL <sup>(5)</sup>
F3Eh	CANSTAT_RO0 <sup>(5)</sup>	F0Eh	CANSTAT_RO3 <sup>(5)</sup>	EDEh	CANSTAT_RO4 <sup>(5)</sup>	EAEh	CANSTAT_RO7 <sup>(5)</sup>	E7Eh	BIE0 <sup>(5)</sup>	E4Eh	RXF7EIDL <sup>(5)</sup>
F3Dh	RXB1D7 <sup>(5)</sup>	F0Dh	TXB2D7 <sup>(5)</sup>	EDDh	B5D7 <sup>(5)</sup>	EADh	B2D7 <sup>(5)</sup>	E7Dh	BSELO <sup>(5)</sup>	E4Dh	RXF7SIDL <sup>(5)</sup>
F3Ch	RXB1D6 <sup>(5)</sup>	F0Ch	TXB2D6 <sup>(5)</sup>	EDCh	B5D6 <sup>(5)</sup>	EACh	B2D6 <sup>(5)</sup>	E7Ch	MSEL3 <sup>(5)</sup>	E4Ch	RXF7SIDH <sup>(5)</sup>
F3Bh	RXB1D5 <sup>(5)</sup>	F0Bh	TXB2D5 <sup>(5)</sup>	EDBh	B5D5 <sup>(5)</sup>	EABh	B2D5 <sup>(5)</sup>	E7Bh	MSEL2 <sup>(5)</sup>	E4Bh	RXF6EIDL <sup>(5)</sup>
F3Ah	RXB1D4 <sup>(5)</sup>	F0Ah	TXB2D4 <sup>(5)</sup>	EDAh	B5D4 <sup>(5)</sup>	EAAh	B2D4 <sup>(5)</sup>	E7Ah	MSEL1 <sup>(5)</sup>	E4Ah	RXF6EIDL <sup>(5)</sup>
F39h	RXB1D3 <sup>(5)</sup>	F09h	TXB2D3 <sup>(5)</sup>	ED9h	B5D3 <sup>(5)</sup>	EA9h	B2D3 <sup>(5)</sup>	E79h	MSEL0 <sup>(5)</sup>	E49h	RXF6SIDL <sup>(5)</sup>
F38h	RXB1D2 <sup>(5)</sup>	F08h	TXB2D2 <sup>(5)</sup>	ED8h	B5D2 <sup>(5)</sup>	EA8h	B2D2 <sup>(5)</sup>	E78h	RXFBCON7 <sup>(5)</sup>	E48h	RXF6SIDH <sup>(5)</sup>
F37h	RXB1D1 <sup>(5)</sup>	F07h	TXB2D1 <sup>(5)</sup>	ED7h	B5D1 <sup>(5)</sup>	EA7h	B2D1 <sup>(5)</sup>	E77h	RXFBCON6 <sup>(5)</sup>	E47h	RXFCON1 <sup>(5)</sup>
F36h	RXB1D0 <sup>(5)</sup>	F06h	TXB2D0 <sup>(5)</sup>	ED6h	B5D0 <sup>(5)</sup>	EA6h	B2D0 <sup>(5)</sup>	E76h	RXFBCON5 <sup>(5)</sup>	E46h	RXFCON0 <sup>(5)</sup>
F35h	RXB1DLC <sup>(5)</sup>	F05h	TXB2DLC <sup>(5)</sup>	ED5h	B5DLC <sup>(5)</sup>	EA5h	B2DLC <sup>(5)</sup>	E75h	RXFBCON4 <sup>(5)</sup>	E45h	BRGCON3 <sup>(5)</sup>
F34h	RXB1EIDL <sup>(5)</sup>	F04h	TXB2EIDL <sup>(5)</sup>	ED4h	B5EIDL <sup>(5)</sup>	EA4h	B2EIDL <sup>(5)</sup>	E74h	RXFBCON3 <sup>(5)</sup>	E44h	BRGCON2 <sup>(5)</sup>
F33h	RXB1EIDH <sup>(5)</sup>	F03h	TXB2EIDH <sup>(5)</sup>	ED3h	B5EIDH <sup>(5)</sup>	EA3h	B2EIDH <sup>(5)</sup>	E73h	RXFBCON2 <sup>(5)</sup>	E43h	BRGCON1 <sup>(5)</sup>
F32h	RXB1SIDI <sup>(5)</sup>	F02h	TXB2SIDL <sup>(5)</sup>	ED2h	B5SIDL <sup>(5)</sup>	EA2h	B2SIDL <sup>(5)</sup>	E72h	RXFBCON1 <sup>(5)</sup>	E42h	TXERRCNT <sup>(5)</sup>
F31h	RXB1SIDH <sup>(5)</sup>	F01h	TXB2SIDH <sup>(5)</sup>	ED1h	B5SIDH <sup>(5)</sup>	EA1h	B2SIDH <sup>(5)</sup>	E71h	RXFBCON0 <sup>(5)</sup>	E41h	RXERRCNT <sup>(5)</sup>
F30h	RXB1CON <sup>(5)</sup>	F00h	TXB2CON <sup>(5)</sup>	ED0h	B5CON <sup>(5)</sup>	EA0h	B2CON <sup>(5)</sup>	E70h	SDFLC <sup>(5)</sup>		
F30h	RXB1CON <sup>(5)</sup>	EFFh	RXM1EIDL <sup>(5)</sup>	ECEh	CANCON_RO5 <sup>(5)</sup>	E9Fh	CANCON_RO8 <sup>(5)</sup>	E6Fh	RXF15EIDL <sup>(5)</sup>		
F2Fh	CANCON_RO1 <sup>(5)</sup>	EEFh	RXM1EIDH <sup>(5)</sup>	ECDh	B4D7 <sup>(5)</sup>	E9Eh	CANSTAT_RO8 <sup>(5)</sup>	E6Eh	RXF15EIDH <sup>(5)</sup>		
F2Eh	CANSTAT_RO1 <sup>(5)</sup>	EFDh	RXM1SIDL <sup>(5)</sup>	ECCh	B4D6 <sup>(5)</sup>	E9Ch	B1D6 <sup>(5)</sup>	E6Dh	RXF15SIDL <sup>(5)</sup>		
F2Dh	TXB0D7 <sup>(5)</sup>	EFCh	RXM1SIDH <sup>(5)</sup>	ECBh	B4D5 <sup>(5)</sup>	E9Bh	B1D5 <sup>(5)</sup>	E6Ch	RXF15SIDH <sup>(5)</sup>		
F2Ch	TXB0D6 <sup>(5)</sup>	EFBh	RXM0EIDL <sup>(5)</sup>	ECAh	B4D4 <sup>(5)</sup>	E9Ah	B1D4 <sup>(5)</sup>	E6Bh	RXF14EIDL <sup>(5)</sup>		
F2Bh	TXB0D5 <sup>(5)</sup>	EFAh	RXM0EIDH <sup>(5)</sup>	EC9h	B4D3 <sup>(5)</sup>	E99h	B1D3 <sup>(5)</sup>	E6Ah	RXF14EIDH <sup>(5)</sup>		
F2Ah	TXB0D4 <sup>(5)</sup>	EF9h	RXM0SIDL <sup>(5)</sup>	EC8h	B4D2 <sup>(5)</sup>	E98h	B1D2 <sup>(5)</sup>	E69h	RXF14SIDL <sup>(5)</sup>		
F29h	TXB0D3 <sup>(5)</sup>	EF8h	RXM0SIDH <sup>(5)</sup>	EC7h	B4D1 <sup>(5)</sup>	E97h	B1D1 <sup>(5)</sup>	E68h	RXF14SIDH <sup>(5)</sup>		
F28h	TXB0D2 <sup>(5)</sup>	EF7h	RXF5EIDL <sup>(5)</sup>	EC6h	B4D0 <sup>(5)</sup>	E96h	B1D0 <sup>(5)</sup>	E67h	RXF13EIDL <sup>(5)</sup>		
F27h	TXB0D1 <sup>(5)</sup>	EF6h	RXF5EIDH <sup>(5)</sup>	EC5h	B4DLC <sup>(5)</sup>	E95h	B1DLC <sup>(5)</sup>	E66h	RXF13EIDH <sup>(5)</sup>		
F26h	TXB0D0 <sup>(5)</sup>	EF5h	RXF5SIDL <sup>(5)</sup>	EC4h	B4EIDL <sup>(5)</sup>	E94h	B1EIDL <sup>(5)</sup>	E65h	RXF13SIDL <sup>(5)</sup>		
F25h	TXB0DLC <sup>(5)</sup>	EF4h	RXF5SIDH <sup>(5)</sup>	EC3h	B4EIDH <sup>(5)</sup>	E93h	B1EIDH <sup>(5)</sup>	E64h	RXF13SIDH <sup>(5)</sup>		
F24h	TXB0EIDL <sup>(5)</sup>	EF3h	RXF4EIDL <sup>(5)</sup>	EC2h	B4SIDL <sup>(5)</sup>	E92h	B1SIDL <sup>(5)</sup>	E63h	RXF12EIDL <sup>(5)</sup>		
F23h	TXB0EIDH <sup>(5)</sup>	EF2h	RXF4EIDH <sup>(5)</sup>	EC1h	B4SIDH <sup>(5)</sup>	E91h	B1SIDH <sup>(5)</sup>	E62h	RXF12EIDH <sup>(5)</sup>		
F22h	TXB0SIDL <sup>(5)</sup>	EF1h	RXF4SIDL <sup>(5)</sup>	EC0h	B4CON <sup>(5)</sup>	E90h	B1CON <sup>(5)</sup>	E61h	RXF12SIDL <sup>(5)</sup>		
F21h	TXB0SIDH <sup>(5)</sup>	EF0h	RXF4SIDH <sup>(5)</sup>	EBFh	CANCON_RO6 <sup>(5)</sup>	E90h	B1CON <sup>(5)</sup>	E60h	RXF12SIDH <sup>(5)</sup>		
F20h	TXB0CON <sup>(5)</sup>	EEFh	RXF3EIDL <sup>(5)</sup>	EBEh	CANSTAT_RO6 <sup>(5)</sup>	E8Fh	CANCON_RO9 <sup>(5)</sup>	E5Fh	RXF11EIDL <sup>(5)</sup>		
F1Fh	CANCON_RO2 <sup>(5)</sup>	EEEh	RXF3EIDH <sup>(5)</sup>	EBDh	B3D7 <sup>(5)</sup>	E8Eh	CANSTAT_RO9 <sup>(5)</sup>	E5Eh	RXF11EIDH <sup>(5)</sup>		
F1Eh	CANSTAT_RO2 <sup>(5)</sup>	EEDh	RXF3SIDL <sup>(5)</sup>	EBCh	B3D6 <sup>(5)</sup>	E8Dh	B0D7 <sup>(5)</sup>	E5Dh	RXF11SIDL <sup>(5)</sup>		
F1Dh	TXB1D7 <sup>(5)</sup>	EECh	RXF3SIDH <sup>(5)</sup>	EBBh	B3D5 <sup>(5)</sup>	E8Ch	B0D6 <sup>(5)</sup>	E5Ch	RXF11SIDH <sup>(5)</sup>		
F1Ch	TXB1D6 <sup>(5)</sup>	EEBh	RXF2EIDL <sup>(5)</sup>	EBAh	B3D4 <sup>(5)</sup>	E8Bh	B0D5 <sup>(5)</sup>	E5Bh	RXF10EIDL <sup>(5)</sup>		
F1Bh	TXB1D5 <sup>(5)</sup>	EEAh	RXF2EIDH <sup>(5)</sup>	EB9h	B3D3 <sup>(5)</sup>	E8Ah	B0D4 <sup>(5)</sup>	E5Ah	RXF10EIDH <sup>(5)</sup>		
F1Ah	TXB1D4 <sup>(5)</sup>	EE9h	RXF2SIDL <sup>(5)</sup>	EB8h	B3D2 <sup>(5)</sup>	E89h	B0D3 <sup>(5)</sup>	E59h	RXF10SIDL <sup>(5)</sup>		
F19h	TXB1D3 <sup>(5)</sup>	EE8h	RXF2SIDH <sup>(5)</sup>	EB7h	B3D1 <sup>(5)</sup>	E88h	B0D2 <sup>(5)</sup>	E58h	RXF10SIDH <sup>(5)</sup>		
F18h	TXB1D2 <sup>(5)</sup>	EE7h	RXF1EIDL <sup>(5)</sup>	EB6h	B3D0 <sup>(5)</sup>	E87h	B0D1 <sup>(5)</sup>	E57h	RXF9EIDL <sup>(5)</sup>		
F17h	TXB1D1 <sup>(5)</sup>	EE6h	RXF1EIDH <sup>(5)</sup>	EB5h	B3DLC <sup>(5)</sup>	E86h	B0D0 <sup>(5)</sup>	E56h	RXF9EIDH <sup>(5)</sup>		
F16h	TXB1D0 <sup>(5)</sup>	EE5h	RXF1SIDL <sup>(5)</sup>	EB4h	B3EIDL <sup>(5)</sup>	E85h	B0DLC <sup>(5)</sup>	E55h	RXF9SIDL <sup>(5)</sup>		
F15h	TXB1DLC <sup>(5)</sup>	EE4h	RXF1SIDH <sup>(5)</sup>	EB3h	B3EIDH <sup>(5)</sup>	E84h	B0EIDL <sup>(5)</sup>	E54h	RXF9SIDH <sup>(5)</sup>		
F14h	TXB1EIDL <sup>(5)</sup>	EE3h	RXF0EIDL <sup>(5)</sup>	EB2h	B3SIDL <sup>(5)</sup>	E83h	B0EIDH <sup>(5)</sup>	E53h	RXF8EIDL <sup>(5)</sup>		
F13h	TXB1EIDH <sup>(5)</sup>	EE2h	RXF0EIDH <sup>(5)</sup>	EB1h	B3SIDH <sup>(5)</sup>	E82h	B0SIDL <sup>(5)</sup>	E52h	RXF8EIDH <sup>(5)</sup>		
F12h	TXB1SIDL <sup>(5)</sup>	EE1h	RXF0SIDL <sup>(5)</sup>	EE0h	B3CON <sup>(5)</sup>	E81h	B0SIDH <sup>(5)</sup>	E51h	RXF8SIDL <sup>(5)</sup>		
F11h	TXB1SIDH <sup>(5)</sup>					E80h	B0CON <sup>(5)</sup>	E50h	RXF8SIDH <sup>(5)</sup>		
F10h	TXB1CON <sup>(5)</sup>										

- Note**
- 1: This is not a physical register.
  - 2: Unimplemented registers are read as '0'.
  - 3: This register is only available on devices with 64 pins.
  - 4: This register is not available on devices with 28 pins.
  - 5: Addresses, E41h through F5Fh, are also used by the SFRs, but are not part of the Access RAM. To access these registers, users must always load the proper BSR value.

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page		
FFFh	TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					88		
FFEh	TOSH	Top-of-Stack High Byte (TOS<15:8>)										88
FFDh	Tosl	Top-of-Stack Low Byte (TOS<7:0>)										88
FFCh	STKPTR	STKFUL	STKUNF	—	SP4	SP3	SP2	SP1	SP0	88		
FFBh	PCLATU	—	—	Bit 21	Holding Register for PC<20:16>					88		
FFAh	PCLATH	Holding Register for PC<15:8>										88
FF9h	PCL	PC Low Byte (PC<7:0>)										88
FF8h	TBLPTRU	—	—	Bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					88		
FF7h	TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)										88
FF6h	TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)										88
FF5h	TABLAT	Program Memory Table Latch										88
FF4h	PRODH	Product Register High Byte										88
FF3h	PRODL	Product Register Low Byte										88
FF2h	INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	88		
FF1h	INTCON2	RBU	INTEGD0	INTEGD1	INTEGD2	INTEGD3	TMR0IP	INT3IP	RBIP	88		
FF0h	INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	88		
FEFh	INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)										88
FEEh	POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)										88
FEDh	POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)										88
FECh	PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)										88
FEBh	PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W										88
FEAh	FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte					88	
FE9h	FSR0L	Indirect Data Memory Address Pointer 0 Low Byte										88
FE8h	WREG	Working Register										88
FE7h	INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)										88
FE6h	POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)										88
FE5h	POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)										88
FE4h	PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)										88
FE3h	PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W										88
FE2h	FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte					88	
FE1h	FSR1L	Indirect Data Memory Address Pointer 1 Low Byte										88
FE0h	BSR	—	—	—	—	Bank Select Register					88	
FDFh	INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)										88
FDEh	POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)										89
FDDh	POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)										89
FDCh	PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)										89
FDBh	PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W										89
FDAh	FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte					89	
FD9h	FSR2L	Indirect Data Memory Address Pointer 2 Low Byte										89
FD8h	STATUS	—	—	—	N	OV	Z	DC	C	89		
FD7h	TMR0H	Timer0 Register High Byte										89
FD6h	TMR0L	Timer0 Register Low Byte										89
FD5h	T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	89		
FD4h	Unimplemented											—
FD3h	OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	HFIofs	SCS1	SCS0	89		
FD2h	OSCCON2	—	SOSCRUN	—	SOSCDRV	SOSCIGO	—	MFIOFS	MFIOSEL	89		
FD1h	WDTCON	REGSLP	—	ULPLVL	SRETEN	—	ULPEN	ULPSINK	SWDTEN	89		
FD0h	RCON	IPEN	SBOREN	CM	RI	TO	PD	POR	BOR	89		

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page
FCFh	TMR1H	Timer1 Register High Byte								89
FCEh	TMR1L	Timer1 Register Low Bytes								89
FCDh	T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	SOSCEN	T1SYNC	RD16	TMR1ON	89
FCCh	TMR2	Timer2 Register								89
FCBh	PR2	Timer2 Period Register								89
FCAh	T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	89
FC9h	SSPBUF	MSSP Receive Buffer/Transmit Register								89
FC8h	SSPADD	MSSP Address Register (I <sup>2</sup> C™ Slave Mode), MSSP Baud Rate Reload Register (I <sup>2</sup> C Master Mode)								89
FC8h	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	89
FC7h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	89
FC6h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	89
FC5h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	89
FC4h	ADRESH	A/D Result Register High Byte								89
FC3h	ADRESL	A/D Result Register Low Byte								89
FC2h	ADCON0	—	CHS4	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	89
FC1h	ADCON1	TRIGSEL1	TRIGSEL0	VCFG1	VCFG0	VNCFG	CHSN2	CHSN1	CHSN0	89
FC0h	ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	89
FBFh	ECCP1AS	ECCP1ASE	ECCP1AS2	ECCP1AS1	ECCP1AS0	PSS1AC1	PSS1AC0	PSS1BD1	PSS1BD0	89
FBEh	ECCP1DEL	P1RSEN	P1DC6	P1DC5	P1DC4	P1DC3	P1DC2	P1DC1	P1DC0	89
FBDh	CCPR1H	Capture/Compare/PWM Register 1 High Byte								89
FBCh	CCPR1L	Capture/Compare/PWM Register 1 Low Byte								89
FBBh	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	89
FBAh	TXSTA2	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D	89
FB9h	BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	89
FB8h	IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP	89
FB7h	PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF	89
FB6h	PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE	89
FB5h	CVRCON	CVREN	CVROE	CVRSS	CVR4	CVR3	CVR2	CVR1	CVR0	89
FB4h	CMSTAT	CMP2OUT	CMP1OUT	—	—	—	—	—	—	89
FB3h	TMR3H	Timer3 Register High Byte								89
FB2h	TMR3L	Timer3 Register Low Bytes								89
FB1h	T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	SOSCEN	T3SYNC	RD16	TMR3ON	89
FB0h	T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/ T3DONE	T3GVAL	T3GSS1	T3GSS0	89
FAFh	SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								89
FAEh	RCREG1	EUSART1 Receive Register								89
FADh	TXREG1	EUSART1 Transmit Register								89
FACH	TXSTA1	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D	89
FABh	RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	89
FAAh	T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/ T1DONE	T1GVAL	T1GSS1	T1GSS0	89
FA9h	PR4	Timer4 Period Register								89
FA8h	HLVDCON	VDIRIMAG	BGVST	IRVST	HLVDEN	HLVLD3	HLVLD2	HLVLD1	HLVLD0	89
FA7h	BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	89
FA6h	RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	89
FA5h	IPR3	—	—	RC2IP	TX2IP	CTMU1P	CCP2IP	CCP1IP	—	89
FA4h	PIR3	—	—	RC2IF	TX2IF	CTMU1F	CCP2IF	CCP1IF	—	89
FA3h	PIE3	—	—	RC2IE	TX2IE	CTMU1E	CCP2IE	CCP1IE	—	89
FA2h	IPR2	OSCFIP	—	—	—	BCLIP	HLVDIP	TMR3IP	TMR3GIP	89
FA1h	PIR2	OSCFIF	—	—	—	BCLIF	HLVDIF	TMR3IF	TMR3GIF	89
FA0h	PIE2	OSCFIE	—	—	—	BCLIE	HLVDIE	TMR3IE	TMR3GIE	89

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page	
F9Fh	IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP	90	
F9Eh	PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF	89	
F9Dh	PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE	89	
F9Ch	PSTR1CON	CMPL1	CMPL0	—	STRSYNC	STRD	STRC	STRB	STRA	89	
F9Bh	OSCTUNE	INTSRC	PLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	90	
F9Ah	REFOCON	ROON	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0	90	
F99h	CCPTMRS	—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL	90	
F98h	TRISG	—	—	—	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	90	
F97h	TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	90	
F96h	TRISE	TRISE7	TRISE6	TRISE5	TRISE4	—	TRISE2	TRISE1	TRISE0	90	
F95h	TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	90	
F94h	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	90	
F93h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	90	
F92h	TRISA	TRISA7	TRISA6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0	90	
F91h	ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD	90	
F90h	SLRCON	—	SLRG	SLRF	SLRE	SLRD	SLRC	SLRB	SLRA	90	
F8Fh	LATG	—	—	—	LATG4	LATG3	LATG2	LATG1	LATG0	90	
F8Eh	LATF	LATF7	LATF6	LATF5	LATF4	—	LATF2	LATF1	LATF0	90	
F8Dh	LATE	LATE7	LATE6	LATE5	LATE4	—	LATE2	LATE1	LATE0	90	
F8Ch	LATD	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	90	
F8Bh	LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	90	
F8Ah	LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	90	
F89h	LATA	LATA7	LATA6	LATA5	—	LATA3	LATA2	LATA1	LATA0	90	
F88h	T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0	90	
F87h	TMR4	Timer4 Register									90
F86h	PORTG	—	—	—	RG4	RG3	RG2	RG1	RG0	90	
F85h	PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	90	
F84h	PORTE	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0	90	
F83h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	90	
F82h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	90	
F81h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	90	
F80h	PORTA	RA7	RA6	RA5	—	RA3	RA2	RA1	RA0	90	
F7fh	EECON1	EEPGD	CFG5	—	FREE	WRERR	WREN	WR	RD	90	
F7Eh	EECON2	Flash Self-Program Control Register (not a physical register)									90
F7Dh	SPBRGH1	EUSART1 Baud Rate Generator Register High Byte									90
F7Ch	SPBRGH2	EUSART2 Baud Rate Generator Register High Byte									90
F7Bh	SPBRG2	EUSART2 Baud Rate Generator Register Low Byte									90
F7Ah	RCREG2	EUSART2 Receive Register									90
F79h	TXREG2	EUSART2 Transmit Register									91
F78h	IPR5	IRXIP	WAKIP	ERRIP	TX2BIP	TXB1IP	TXB0IP	RXB1IP	RXB0IP	91	
F77h	PIR5	IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF	TXB0IF	RXB1IF	RXB0IF	91	
F76H	PIE5	IRXIE	WAKIE	ERRIE	TX2BIE	TXB1IE	TXB0IE	RXB1IE	RXB0IE	91	
F75h	EEADRH	Data EE Address Register High Byte									91
F74h	EEADR	Data EE Address Register Low Byte									91
F73h	EEDATA	Data EE Data Register									91
F72h	ECANCON	MDSEL1	MDSEL0	FIFOWM	EWIN4	EWIN3	EWIN2	EWIN1	EWIN0	91	
F71h	COMSTAT	RXB0OVFL	RXB1OVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN	91	
F70h	CIOCON	TX2SRC	TX2EN	ENDRHI	CANCAP	—	—	—	CLKSEL	91	
F6Fh	CANCON	REQOP2	REQOP1	REQOP0	ABAT	WIN2/FP3	WIN1/FP2	WIN0/FP1	FP0	91	
F6Eh	CANSTAT	OPMODE2	OPMODE1	OPMODE0	—/ EICOD4	ICODE2/ EICODE3	ICODE1/ EICODE2	ICODE0/ EICODE1	—/ EICODE0	91	

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page
F6Dh	RXB0D7	RXB0D77	RXB0D76	RXB0D75	RXB0D74	RXB0D73	RXB0D72	RXB0D71	RXB0D70	91
F6Ch	RXB0D6	RXB0D67	RXB0D66	RXB0D65	RXB0D64	RXB0D63	RXB0D62	RXB0D61	RXB0D60	91
F6Bh	RXB0D5	RXB0D57	RXB0D56	RXB0D55	RXB0D54	RXB0D53	RXB0D52	RXB0D51	RXB0D50	91
F6Ah	RXB0D4	RXB0D47	RXB0D46	RXB0D45	RXB0D44	RXB0D43	RXB0D42	RXB0D41	RXB0D40	91
F69h	RXB0D3	RXB0D37	RXB0D36	RXB0D35	RXB0D34	RXB0D33	RXB0D32	RXB0D31	RXB0D30	91
F68h	RXB0D2	RXB0D27	RXB0D26	RXB0D25	RXB0D24	RXB0D23	RXB0D22	RXB0D21	RXB0D20	91
F67h	RXB0D1	RXB0D17	RXB0D16	RXB0D15	RXB0D14	RXB0D13	RXB0D12	RXB0D11	RXB0D10	91
F66h	RXB0D0	RXB0D07	RXB0D06	RXB0D05	RXB0D04	RXB0D03	RXB0D02	RXB0D01	RXB0D00	91
F65h	RXB0DLC	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	91
F64h	RXB0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	91
F63h	RXB0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	91
F62h	RXB0SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	91
F61h	RXB0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	91
F60h	RXB0CON	RXFUL	RXM1	RXM0	—	RXRTRRO	RXB0DBEN	JTOFF	FILHITO	91
F60h	RXB0CON	RXFUL	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHITO	91
F5Fh	CM1CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0	91
F5Eh	CM2CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0	91
F5Dh	ANCON0	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0	91
F5Ch	ANCON1	—	ANSEL14	ANSEL13	ANSEL12	ANSEL11	ANSEL10	ANSEL9	ANSEL8	91
F5Bh	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	91
F5Ah	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—	91
F59h	PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD	91
F58h	PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD	93
F57h	PMD2	—	—	—	MODMD	ECANMD	CMP2MD	CMP1MD	93	
F56h	PADCFG1	RDPU	REPU	RFPU	RGPU	—	—	—	CTMUDS	93
F55h	CTMUCONH	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	93
F54h	CTMUCONL	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	93
F53h	CTMUIICON	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	93
F52h	CCPR2H	Capture/Compare/PWM Register 2 High Byte								93
F51h	CCPR2L	Capture/Compare/PWM Register 2 Low Byte								93
F50h	CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	93
F4Fh	CCPR3H	Capture/Compare/PWM Register 3 High Byte								93
F4Eh	CCPR3L	Capture/Compare/PWM Register 3 Low Byte								93
F4Dh	CCP3CON	—	—	DC3B1	D32B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	93
F4Ch	CCPR4H	Capture/Compare/PWM Register 4 High Byte								93
F4Bh	CCPR4L	Capture/Compare/PWM Register 4 Low Byte								93
F4Ah	CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0	93
F49h	CCPR5H	Capture/Compare/PWM Register 5 High Byte								93
F48h	CCPR5L	Capture/Compare/PWM Register 5 Low Byte								93
F47h	CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0	93
F46h	PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	93
F45h	MDCON	MDEN	MDOE	MDSLR	MDOPOL	MDO	—	—	MDBIT	93
F44h	MDSRC	MDSODIS	—	—	—	MDSRC3	MDSRC2	MDSRC1	MDSRC0	93
F43h	MDCARH	MDCHODIS	MDCHPOL	MDCHSYNC	—	MDCH3	MDCH2	MDCH1	MDCH0	93
F42h	MDCARL	MDCLODIS	MDCLPOL	MDCLSYNC	—	MDCL3	MDCL2	MDCL1	MDCL0	93
F41h	Unimplemented								—	
F40h	Unimplemented								—	
F3Fh	CANCON_RO0	CANCON_RO0								93
F3Eh	CANSTAT_RO0	CANSTAT_RO0								93
F3Dh	RXB1D7	RXB1D77	RXB1D76	RXB1D75	RXB1D74	RXB1D73	RXB1D72	RXB1D71	RXB1D70	93
F3Ch	RXB1D6	RXB1D67	RXB1D66	RXB1D65	RXB1D64	RXB1D63	RXB1D62	RXB1D61	RXB1D60	93

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page
F3Bh	RXB1D5	RXB1D57	RXB1D56	RXB1D55	RXB1D54	RXB1D53	RXB1D52	RXB1D51	RXB1D50	93
F3Ah	RXB1D4	RXB1D47	RXB1D46	RXB1D45	RXB1D44	RXB1D43	RXB1D42	RXB1D41	RXB1D40	93
F39h	RXB1D3	RXB1D37	RXB1D36	RXB1D35	RXB1D34	RXB1D33	RXB1D32	RXB1D31	RXB1D30	93
F38h	RXB1D2	RXB1D27	RXB1D26	RXB1D25	RXB1D24	RXB1D23	RXB1D22	RXB1D21	RXB1D20	93
F37h	RXB1D1	RXB1D17	RXB1D16	RXB1D15	RXB1D14	RXB1D13	RXB1D12	RXB1D11	RXB1D10	93
F36h	RXB1D0	RXB1D07	RXB1D06	RXB1D05	RXB1D04	RXB1D03	RXB1D02	RXB1D01	RXB1D00	93
F35h	RXB1DLC	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	93
F34h	RXB1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	94
F33h	RXB1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	94
F32h	RXB1SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	94
F31h	RXB1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	94
F30h	RXB1CON	RXFUL	RXM1	RXM0	—	RXRTRRO	RXBODBEN	JTOFF	FILHIT0	94
F30h	RXB1CON	RXFUL	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0	94
F2Fh	CANCON_RO1	CANCON_RO1								94
F2Eh	CANSTAT_RO1	CANSTAT_RO1								94
F2Dh	TXB0D7	TXB0D77	TXB0D76	TXB0D75	TXB0D74	TXB0D73	TXB0D72	TXB0D71	TXB0D70	94
F2Ch	TXB0D6	TXB0D67	TXB0D66	TXB0D65	TXB0D64	TXB0D63	TXB0D62	TXB0D61	TXB0D60	94
F2Bh	TXB0D5	TXB0D57	TXB0D56	TXB0D55	TXB0D54	TXB0D53	TXB0D52	TXB0D51	TXB0D50	94
F2Ah	TXB0D4	TXB0D47	TXB0D46	TXB0D45	TXB0D44	TXB0D43	TXB0D42	TXB0D41	TXB0D40	94
F29h	TXB0D3	TXB0D37	TXB0D36	TXB0D35	TXB0D34	TXB0D33	TXB0D32	TXB0D31	TXB0D30	94
F28h	TXB0D2	TXB0D27	TXB0D26	TXB0D25	TXB0D24	TXB0D23	TXB0D22	TXB0D21	TXB0D20	94
F27h	TXB0D1	TXB0D17	TXB0D16	TXB0D15	TXB0D14	TXB0D13	TXB0D12	TXB0D11	TXB0D10	94
F26h	TXB0D0	TXB0D07	TXB0D06	TXB0D05	TXB0D04	TXB0D03	TXB0D02	TXB0D01	TXB0D00	94
F25h	TXB0DLC	—	RXRTR	—	—	DLC3	DLC2	DLC1	DLC0	94
F24h	TXB0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	94
F23h	TXB0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	94
F22h	TXB0SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	94
F21h	TXB0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	94
F20h	TXB0CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	94
F1Fh	CANCON_RO2	CANCON_RO2								94
F1Eh	CANSTAT_RO2	CANSTAT_RO2								94
F1Dh	TXB1D7	TXB1D77	TXB1D76	TXB1D75	TXB1D74	TXB1D73	TXB1D72	TXB1D71	TXB1D70	94
F1Ch	TXB1D6	TXB1D67	TXB1D66	TXB1D65	TXB1D64	TXB1D63	TXB1D62	TXB1D61	TXB1D60	94
F1Bh	TXB1D5	TXB1D57	TXB1D56	TXB1D55	TXB1D54	TXB1D53	TXB1D52	TXB1D51	TXB1D50	94
F1Ah	TXB1D4	TXB1D47	TXB1D46	TXB1D45	TXB1D44	TXB1D43	TXB1D42	TXB1D41	TXB1D40	94
F19h	TXB1D3	TXB1D37	TXB1D36	TXB1D35	TXB1D34	TXB1D33	TXB1D32	TXB1D31	TXB1D30	94
F18h	TXB1D2	TXB1D27	TXB1D26	TXB1D25	TXB1D24	TXB1D23	TXB1D22	TXB1D21	TXB1D20	94
F17h	TXB1D1	TXB1D17	TXB1D16	TXB1D15	TXB1D14	TXB1D13	TXB1D12	TXB1D11	TXB1D10	94
F16h	TXB1D0	TXB1D07	TXB1D06	TXB1D05	TXB1D04	TXB1D03	TXB1D02	TXB1D01	TXB1D00	94
F15h	TXB1DLC	—	RXRTR	—	—	DLC3	DLC2	DLC1	DLC0	94
F14h	TXB1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	94
F13h	TXB1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	94
F12h	TXB1SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	94
F11h	TXB1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	94
F10h	TXB1CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	94
F0Fh	CANCON_RO3	CANCON_RO3								94
F0Eh	CANSTAT_RO3	CANSTAT_RO3								94
F0Dh	TXB2D7	TXB2D77	TXB2D76	TXB2D75	TXB2D74	TXB2D73	TXB2D72	TXB2D71	TXB2D70	94
F0Ch	TXB2D6	TXB2D67	TXB2D66	TXB2D65	TXB2D64	TXB2D63	TXB2D62	TXB2D61	TXB2D60	95
F0Bh	TXB2D5	TXB2D57	TXB2D56	TXB2D55	TXB2D54	TXB2D53	TXB2D52	TXB2D51	TXB2D50	95
F0Ah	TXB2D4	TXB2D47	TXB2D46	TXB2D45	TXB2D44	TXB2D43	TXB2D42	TXB2D41	TXB2D40	—

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page
F09h	TXB2D3	TXB2D37	TXB2D36	TXB2D35	TXB2D34	TXB2D33	TXB2D32	TXB2D31	TXB2D30	95
F08h	TXB2D2	TXB2D27	TXB2D26	TXB2D25	TXB2D24	TXB2D23	TXB2D22	TXB2D21	TXB2D20	95
F07h	TXB2D1	TXB2D17	TXB2D16	TXB2D15	TXB2D14	TXB2D13	TXB2D12	TXB2D11	TXB2D10	95
F06h	TXB2D0	TXB2D07	TXB2D06	TXB2D05	TXB2D04	TXB2D03	TXB2D02	TXB2D01	TXB2D00	95
F05h	TXB2DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	95
F04h	TXB2EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
F03h	TXB2EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
F02h	TXB2SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	95
F01h	TXB2SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
F00h	TXB2CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	95
EFFh	RXM1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EEFh	RXM1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EFDh	RXM1SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EFCh	RXM1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EFBh	RXM0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EFAh	RXM0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EF9h	RXM0SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EF8h	RXM0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EF7h	RXF5EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EF6h	RXF5EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EF5h	RXF5SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EF4h	RXF5SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EF3h	RXF4EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EF2h	RXF4EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EF1h	RXF4SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EF0h	RXF4SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EEFh	RXF3EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EEEh	RXF3EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EEDh	RXF3SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EECh	RXF3SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EEBh	RXF2EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EEAh	RXF2EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EE9h	RXF2SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EE8h	RXF2SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EE7h	RXF1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EE6h	RXF1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EE5h	RXF1SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EE4h	RXF1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EE3h	RXF0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
EE2h	RXF0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
EE1h	RXF0SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	95
EE0h	RXF0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
EDFh	CANCON_RO4	CANCON_RO4								95
EDEh	CANSTAT_RO4	CANSTAT_RO4								95
EDDh	B5D7	B5D77	B5D76	B5D75	B5D74	B5D73	B5D72	B5D71	B5D70	95
EDCh	B5D6	B5D67	B5D66	B5D65	B5D64	B5D63	B5D62	B5D61	B5D60	95
EDBh	B5D5	B5D57	B5D56	B5D55	B5D54	B5D53	B5D52	B5D51	B5D50	95
EDAh	B5D4	B5D47	B5D46	B5D45	B5D44	B5D43	B5D42	B5D41	B5D40	95
ED9h	B5D3	B5D37	B5D36	B5D35	B5D34	B5D33	B5D32	B5D31	B5D30	95
ED8h	B5D2	B5D27	B5D26	B5D25	B5D24	B5D23	B5D22	B5D21	B5D20	95
ED7h	B5D1	B5D17	B5D16	B5D15	B5D14	B5D13	B5D12	B5D11	B5D10	95

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page
ED6h	B5D0	B5D07	B5D06	B5D05	B5D04	B5D03	B5D02	B5D01	B5D00	95
ED5h	B5DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	95
ED4h	B5EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	95
ED3h	B5EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	95
ED2h	B5SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	95
ED1h	B5SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	95
ED0h	B5CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	95
ECFh	CANCON_RO5	CANCON_RO5								95
ECEh	CANSTAT_RO5	CANSTAT_RO5								96
ECDh	B4D7	B4D77	B4D76	B4D75	B4D74	B4D73	B4D72	B4D71	B4D70	96
ECCh	B4D6	B4D67	B4D66	B4D65	B4D64	B4D63	B4D62	B4D61	B4D60	96
ECBh	B4D5	B4D57	B4D56	B4D55	B4D54	B4D53	B4D52	B4D51	B4D50	96
ECAh	B4D4	B4D47	B4D46	B4D45	B4D44	B4D43	B4D42	B4D41	B4D40	96
EC9h	B4D3	B4D37	B4D36	B4D35	B4D34	B4D33	B4D32	B4D31	B4D30	96
EC8h	B4D2	B4D27	B4D26	B4D25	B4D24	B4D23	B4D22	B4D21	B4D20	96
EC7h	B4D1	B4D17	B4D16	B4D15	B4D14	B4D13	B4D12	B4D11	B4D10	96
EC6h	B4D0	B4D07	B4D06	B4D05	B4D04	B4D03	B4D02	B4D01	B4D00	96
EC5h	B4DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	96
EC4h	B4EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	96
EC3h	B4EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	96
EC2h	B4SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	96
EC1h	B4SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	96
EC0h	B4CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	96
EBFh	CANCON_RO6	CANCON_RO6								96
EBEh	CANSTAT_RO6	CANSTAT_RO6								96
EBDh	B3D7	B3D77	B3D76	B3D75	B3D73	B3D73	B3D72	B3D71	B3D70	96
EBCh	B3D6	B3D67	B3D66	B3D65	B3D63	B3D63	B3D62	B3D61	B3D60	96
EBBh	B3D5	B3D57	B3D56	B3D55	B3D53	B3D53	B3D52	B3D51	B3D50	96
EBAh	B3D4	B3D47	B3D46	B3D45	B3D43	B3D43	B3D42	B3D41	B3D40	96
EB9h	B3D3	B3D37	B3D36	B3D35	B3D33	B3D33	B3D32	B3D31	B3D30	96
EB8h	B3D2	B3D27	B3D26	B3D25	B3D23	B3D23	B3D22	B3D21	B3D20	96
EB7h	B3D1	B3D17	B3D16	B3D15	B3D13	B3D13	B3D12	B3D11	B3D10	96
EB6h	B3D0	B3D07	B3D06	B3D05	B3D03	B3D03	B3D02	B3D01	B3D00	96
EB5h	B3DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	96
EB4h	B3EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	96
EB3h	B3EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	96
EB2h	B3SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	96
EB1h	B3SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	96
EB0h	B3CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	96
EAFh	CANCON_RO7	CANCON_RO7								96
EEAh	CANSTAT_RO7	CANSTAT_RO7								96
EADh	B2D7	B2D77	B2D76	B2D75	B2D72	B2D73	B2D72	B2D71	B2D70	96
EACH	B2D6	B2D67	B2D66	B2D65	B2D62	B2D63	B2D62	B2D61	B2D60	96
EABh	B2D5	B2D57	B2D56	B2D55	B2D52	B2D53	B2D52	B2D51	B2D50	97
EAAh	B2D4	B2D47	B2D46	B2D45	B2D42	B2D43	B2D42	B2D41	B2D40	97
EA9h	B2D3	B2D37	B2D36	B2D35	B2D32	B2D33	B2D32	B2D31	B2D30	97
EA8h	B2D2	B2D27	B2D26	B2D25	B2D22	B2D23	B2D22	B2D21	B2D20	97
EA7h	B2D1	B2D17	B2D16	B2D15	B2D12	B2D13	B2D12	B2D11	B2D10	97
EA6h	B2D0	B2D07	B2D06	B2D05	B2D02	B2D03	B2D02	B2D01	B2D00	97
EA5h	B2DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	97
EA4h	B2EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	97

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page
EA3h	B2EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	97
EA2h	B2SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	97
EA1h	B2SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	97
EA0h	B2CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	97
E9Fh	CANCON_RO8	CANCON_RO8								97
E9Eh	CANSTAT_RO8	CANSTAT_RO8								97
E9Dh	B1D7	B1D77	B1D76	B1D75	B1D71	B1D73	B1D72	B1D71	B1D70	97
E9Ch	B1D6	B1D67	B1D66	B1D65	B1D61	B1D63	B1D62	B1D61	B1D60	97
E9Bh	B1D5	B1D57	B1D56	B1D55	B1D51	B1D53	B1D52	B1D51	B1D50	97
E9Ah	B1D4	B1D47	B1D46	B1D45	B1D41	B1D43	B1D42	B1D41	B1D40	97
E99h	B1D3	B1D37	B1D36	B1D35	B1D31	B1D33	B1D32	B1D31	B1D30	97
E98h	B1D2	B1D27	B1D26	B1D25	B1D21	B1D23	B1D22	B1D21	B1D20	97
E97h	B1D1	B1D17	B1D16	B1D15	B1D11	B1D13	B1D12	B1D11	B1D10	97
E96h	B1D0	B1D07	B1D06	B1D05	B1D01	B1D03	B1D02	B1D01	B1D00	97
E95h	B1DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	97
E94h	B1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	97
E93h	B1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	97
E92h	B1SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	97
E91h	B1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	97
E90h	B1CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	RTREN	TXPRI1	TXPRI0	97
E90h	B1CON	RXFUL	RXM1	RXRTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0	97
E8Fh	CANCON_RO9	CANCON_RO9								97
E8Eh	CANSTAT_RO9	CANSTAT_RO9								97
E8Dh	B0D7	B0D77	B0D76	B0D75	B0D70	B0D73	B0D72	B0D71	B0D70	97
E8Ch	B0D6	B0D67	B0D66	B0D65	B0D60	B0D63	B0D62	B0D61	B0D60	97
E8Bh	B0D5	B0D57	B0D56	B0D55	B0D50	B0D53	B0D52	B0D51	B0D50	97
E8Ah	B0D4	B0D47	B0D46	B0D45	B0D40	B0D43	B0D42	B0D41	B0D40	97
E89h	B0D3	B0D37	B0D36	B0D35	B0D30	B0D33	B0D32	B0D31	B0D30	97
E88h	B0D2	B0D27	B0D26	B0D25	B0D20	B0D23	B0D22	B0D21	B0D20	98
E87h	B0D1	B0D17	B0D16	B0D15	B0D10	B0D13	B0D12	B0D11	B0D10	98
E86h	B0D0	B0D07	B0D06	B0D05	B0D00	B0D03	B0D02	B0D01	B0D00	98
E85h	B0DLC	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	98
E84h	B0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	98
E83h	B0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	98
E82h	B0SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	98
E81h	B0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	98
E80h	B0CON	TXBIF	TXABT	TXLARB	TXERR	TXREQ	RTREN	TXPRI1	TXPRI0	98
E80h	B0CON	RXFUL	RXM1	RXRTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0	98
E7Fh	TXBIE	—	—	—	CAN TX Buffer Interrupt Enable	—	—	—	—	98
E7Eh	BIE0	CAN Buffer Interrupt Enable								98
E7Dh	BSEL0	Mode Select Register 0					—	—	—	98
E7Ch	MSEL3	CAN Mask Select Register 3								98
E7Bh	MSEL2	CAN Mask Select Register 2								98
E7Ah	MSEL1	CAN Mask Select Register 1								98
E79h	MSEL0	CAN Mask Select Register 0								98
E78h	RXFBCON7	CAN Buffer 15/14 Pointer Register								98
E77h	RXFBCON6	CAN Buffer 13/12 Pointer Register								98
E76h	RXFBCON5	CAN Buffer 11/10 Pointer Register								98
E75h	RXFBCON4	CAN Buffer 9/8 Pointer Register								98
E74h	RXFBCON3	CAN Buffer 7/6 Pointer Register								98
E73h	RXFBCON2	CAN Buffer 5/4 Pointer Register								98

# PIC18F66K80 FAMILY

**TABLE 6-2: PIC18F66K80 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR on page
E72h	RXFBCON1	CAN Buffer 3/2 Pointer Register								98
E71h	RXFBCON0	CAN Buffer 1/0 Pointer Register								98
E70h	SDFLC	—	—	—	CAN Device Net Count Register					98
E6Fh	RXF15EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	98
E6Eh	RXF15EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	98
E6Dh	RXF15SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	98
E6Ch	RXF15SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	98
E6Bh	RXF14EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	98
E6Ah	RXF14EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	98
E69h	RXF14SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	98
E68h	RXF14SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	98
E67h	RXF13EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	98
E66h	RXF13EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E65h	RXF13SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E64h	RXF13SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E63h	RXF12EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	99
E62h	RXF12EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E61h	RXF12SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E60h	RXF12SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E5Fh	RXF11EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	99
E5Eh	RXF11EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E5Dh	RXF11SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E5Ch	RXF11SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E5Bh	RXF10EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	99
E5Ah	RXF10EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E59h	RXF10SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E58h	RXF10SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E57h	RXF9EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	99
E56h	RXF9EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E55h	RXF9SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E54h	RXF9SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E53h	RXF8EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	99
E52h	RXF8EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E51h	RXF8SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E50h	RXF8SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E4Fh	RXF7EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	99
E4Eh	RXF7EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E4Dh	RXF7SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E4Ch	RXF7SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E4Bh	RXF6EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	99
E4Ah	RXF6EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	99
E49h	RXF6SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	99
E48h	RXF6SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	99
E47h	RXFCON1	CAN Receive Filter Control Register 1								99
E46h	RXFCON0	CAN Receive Filter Control Register 0								99
E45h	BRGCON3	WAKDIS	WAKFIL	—	—	—	SEG2PH2	SEG2PH1	SEG2PH0	99
E44h	BRGCON2	SEG2PHTS	SAM	SEG1PH2	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0	100
E43h	BRGCON1	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	100
E42h	TXERRCNT	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	100
E41h	RXERRCNT	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	100

# PIC18F66K80 FAMILY

## 6.3.5 STATUS REGISTER

The STATUS register, shown in [Register 6-2](#), contains the arithmetic status of the ALU. The STATUS register can be the operand for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the write to these five bits is disabled.

These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the STATUS register as destination may be different than intended. For example, `CLRF STATUS` will set the Z bit but leave the other bits unchanged. The STATUS register then reads back as '000u uluu'.

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions be 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 not affecting any Status bits, see the instruction set summaries in [Table 29-2](#) and [Table 29-3](#).

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

## 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  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'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 seven-bit magnitude which causes the sign bit (bit 7) to change state.  
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)  
0 = No overflow occurred
- bit 2      **Z:** Zero bit  
1 = The result of an arithmetic or logic operation is zero  
0 = The result of an arithmetic or logic operation is not zero
- bit 1      **DC:** Digit Carry/Borrow bit<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 occurred
- 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.
- 2:** For borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand.

## 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. For more information, see [Section 6.6 “Data Memory and the Extended Instruction Set”](#).

While the program memory can be addressed in only one way, through the Program Counter, information in 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). For details on this mode's operation, see [Section 6.6.1 “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 of this mode include SLEEP, RESET and DAW.

Other instructions work in a similar way, but require an additional explicit argument in the opcode. This method is known as the Literal Addressing mode because the instructions require some literal value as an argument. Examples of this 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 the instruction's data source as either a register address in one of the banks

of data RAM (see [Section 6.3.3 “General Purpose Register File”](#)) or a location in the Access Bank (see [Section 6.3.2 “Access Bank”](#)).

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”](#)) 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, 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](#). It also enables users to perform Indexed Addressing and other Stack Pointer operations for program memory in data memory.

### EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```

LFSR   FSR0, 100h ; 
NEXT   CLRF  POSTINC0 ; Clear INDF
          ; register then
          ; inc pointer
          ; 
          ; BTFSS FSR0H, 1 ; All done with
          ; 
          ; BRA    NEXT    ; NO, clear next
CONTINUE                      ; YES, continue

```

# PIC18F66K80 FAMILY

## 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.

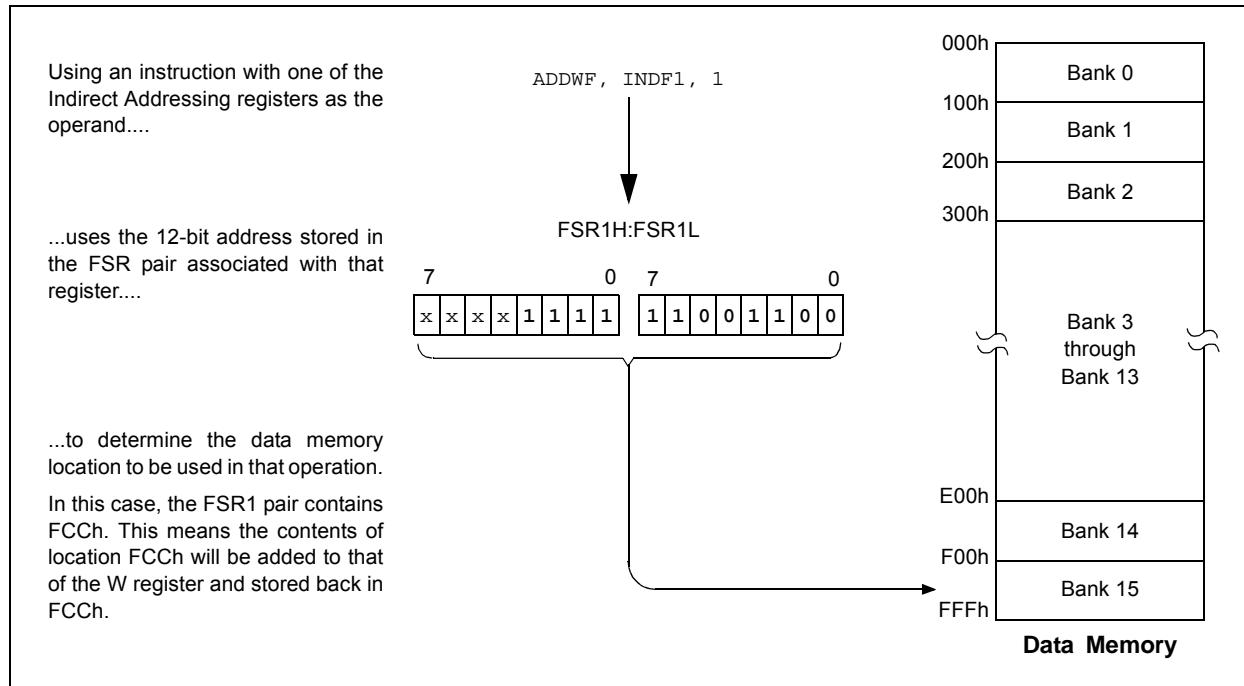
Indirect Addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as "virtual" registers. The operands are

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.

**FIGURE 6-8: INDIRECT ADDRESSING**



### 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.

These operands 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 the value in the W register, with neither value 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. Rollovers of the FSRRnL register, from FFh to 00h, carry over to the FSRRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (for example, Z, N and OV bits).

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 the FSR0H:FSR0L registers contain 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, however, 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, so that they do not inadvertently change settings that might affect the operation of the device.

## 6.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds five additional two-word commands to the existing PIC18 instruction set: ADDFSR, CALLW, MOVSF, MOVSS and SUBFSR. These instructions are executed as described in [Section 6.2.4 “Two-Word Instructions”](#).

# PIC18F66K80 FAMILY

---

## 6.6 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. Using the Access Bank for many of the core PIC18 instructions introduces a new addressing mode for the data memory space. This mode also alters the behavior of Indirect Addressing using FSR2 and its associated operands.

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.

### 6.6.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair and its associated file operands. 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 the Indexed Literal Offset mode.

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

- Use of the Access Bank ('a' = 0)
- A file address argument that 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.

### 6.6.2 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 = 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 6-9](#).

Those who desire to use byte-oriented or bit-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 29.2.1 “Extended Instruction Syntax”](#).

# PIC18F66K80 FAMILY

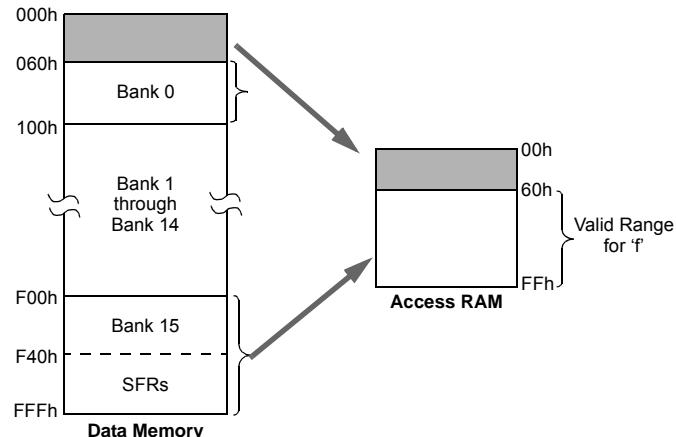
**FIGURE 6-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

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

**When a = 0 and f  $\geq$  60h:**

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and FFFh. This is the same as locations, F60h to FFFh, (Bank 15) of data memory.

Locations below 060h are not available in this addressing mode.

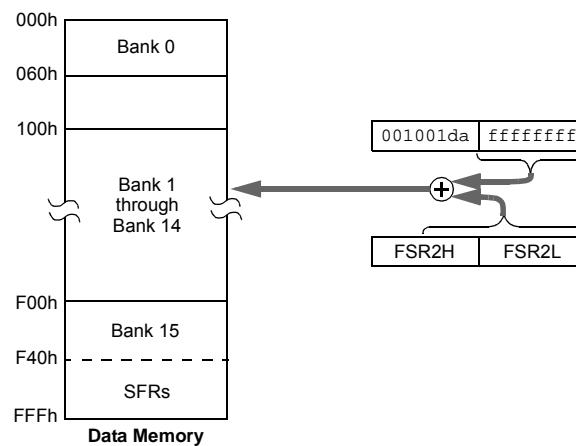


**When a = 0 and f  $\leq$  5Fh:**

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.

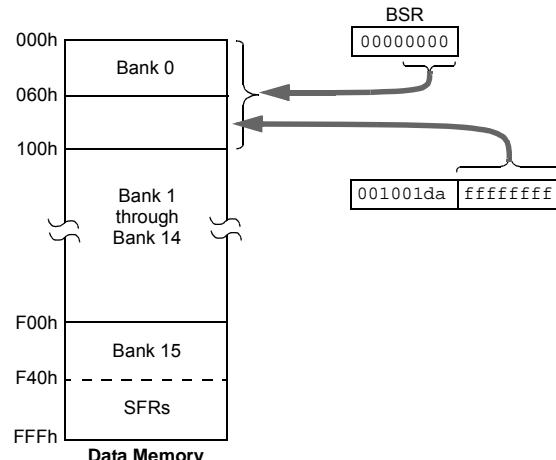
Note that in this mode, the correct syntax is now:

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.



# PIC18F66K80 FAMILY

## 6.6.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

The use of Indexed Literal Offset Addressing mode effectively changes how the lower part of Access RAM (00h to 5Fh) is mapped. Rather than containing just the contents of the bottom part 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.

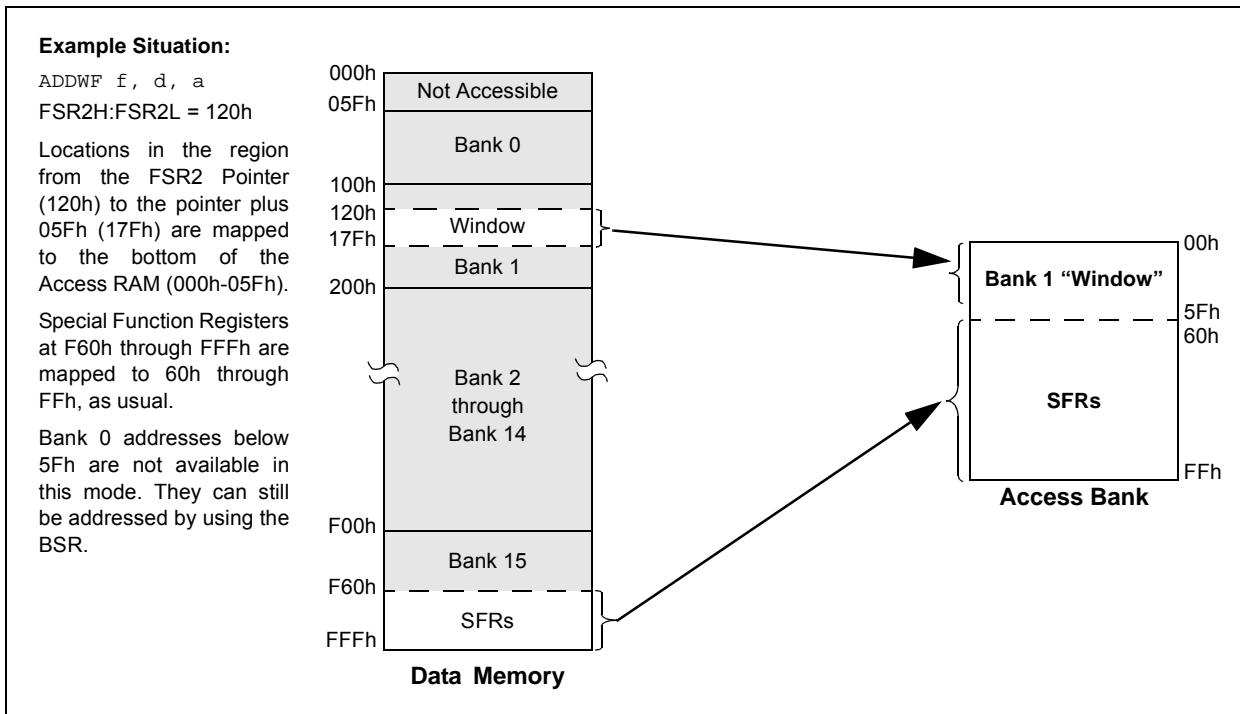
The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described. (See [Section 6.3.2 “Access Bank”](#).) An example of Access Bank remapping in this addressing mode is shown in [Figure 6-10](#).

Remapping the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit = 1) will continue to use Direct Addressing as before. Any Indirect or Indexed Addressing operation that explicitly uses any of the indirect file operands (including FSR2) will continue to operate as standard Indirect Addressing. Any instruction that uses the Access Bank, but includes a register address of greater than 05Fh, will use Direct Addressing and the normal Access Bank map.

## 6.6.4 BSR IN INDEXED LITERAL OFFSET MODE

Although the Access Bank is remapped when the extended instruction set is enabled, the operation of the BSR remains unchanged. Direct Addressing, using the BSR to select the data memory bank, operates in the same manner as previously described.

**FIGURE 6-10: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING**



## 7.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire V<sub>DD</sub> range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 64 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)

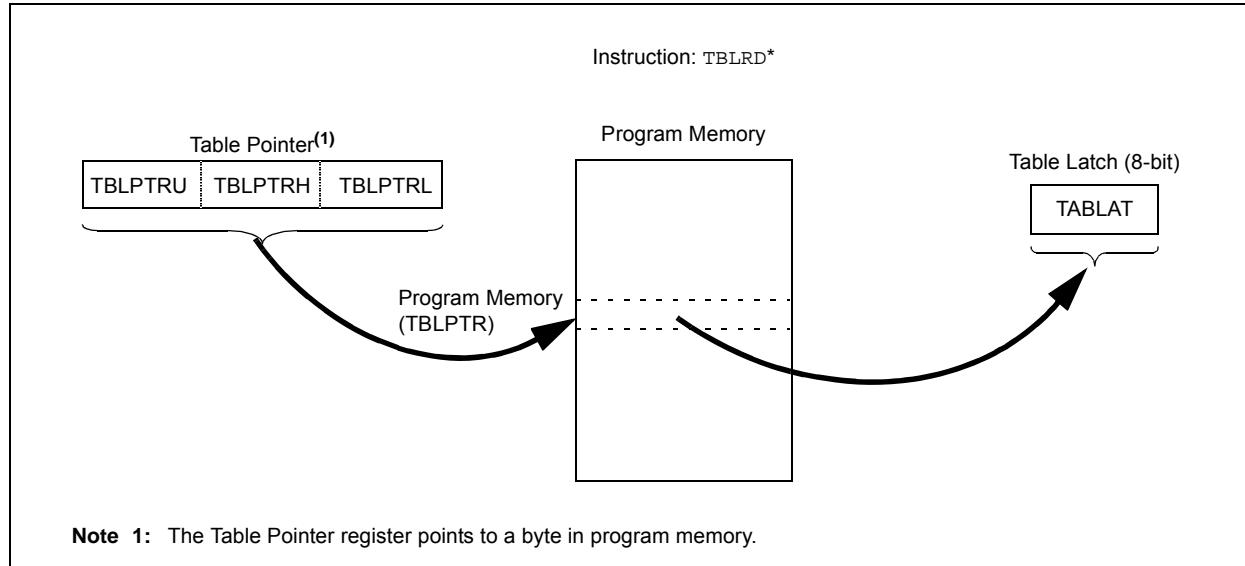
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

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.

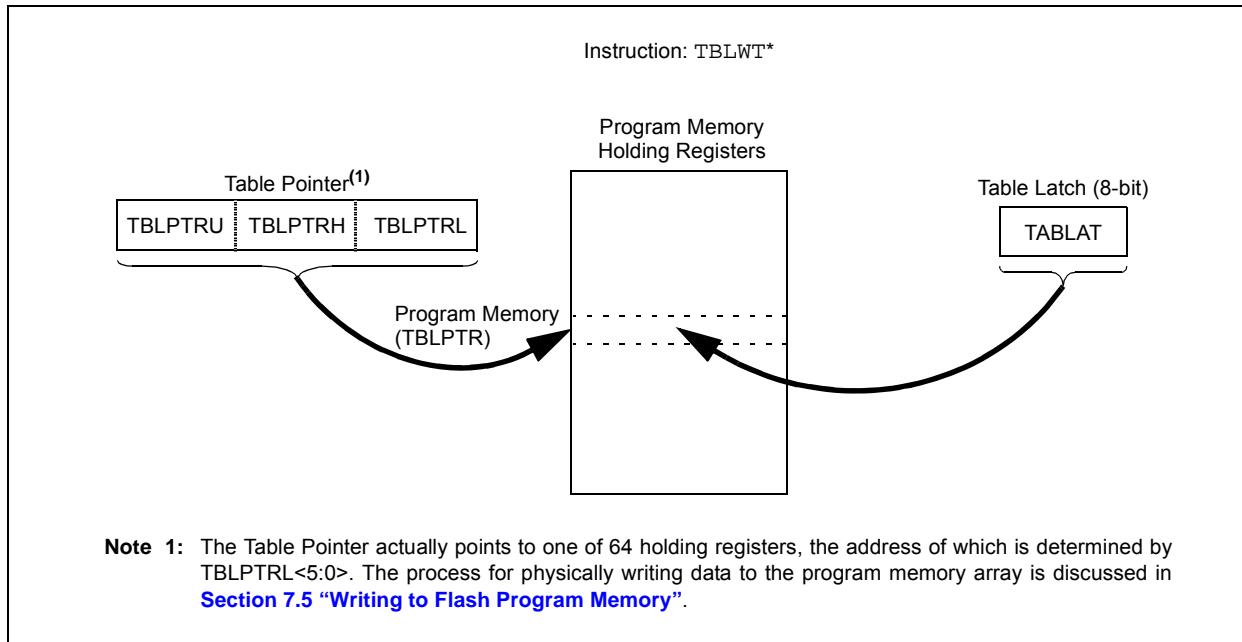
Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions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

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



# PIC18F66K80 FAMILY

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, not a physical register, 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 is a program or data EEPROM memory access. When clear, any subsequent operations operate on the data EEPROM memory. When set, any subsequent operations operate on the program memory.

The CFGS control bit determines if the access is to the Configuration registers or to program memory/data EEPROM memory. When set, subsequent operations operate on Configuration registers regardless of EEPGD (see [Section 28.0 “Special Features of the CPU”](#)). When clear, memory selection access is determined by EEPGD.

The FREE bit, when set, allows 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, allows 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 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.

**Note:** The EEIF interrupt flag bit (PIR4<6>) is set when the write is complete. It must be cleared in software.

# PIC18F66K80 FAMILY

## 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	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7	<b>EEPGD:</b> Flash Program or Data EEPROM Memory Select bit 1 = Accesses Flash program memory 0 = Accesses data EEPROM memory
bit 6	<b>CFGs:</b> Flash Program/Data EEPROM or Configuration Select bit 1 = Accesses Configuration registers 0 = Accesses Flash program or data EEPROM memory
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>FREE:</b> Flash Row Erase Enable bit 1 = Erases the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation) 0 = Performs write-only
bit 3	<b>WRERR:</b> 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	<b>WREN:</b> 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	<b>WR:</b> 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 the write is complete.) The WR bit can only be set (not cleared) in software. 0 = Write cycle to the EEPROM is complete
bit 0	<b>RD:</b> 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 EEPGD = 1 or CFGs = 1.) 0 = Does not initiate an EEPROM read

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

# PIC18F66K80 FAMILY

## 7.2.2 TABLAT – TABLE LATCH REGISTER

The Table Latch (TABLAT) is an eight-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](#) and 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 a TBLWT is executed, the six LSbs of the Table Pointer register (TBLPTR<5:0>) determine which of the 64 program memory holding registers is written to. When the timed write to program memory begins (via the WR bit), the 16 MSbs of the TBLPTR (TBLPTR<21:6>) determine which program memory block of 64 bytes is written to. For more detail, see [Section 7.5 “Writing to Flash Program Memory”](#).

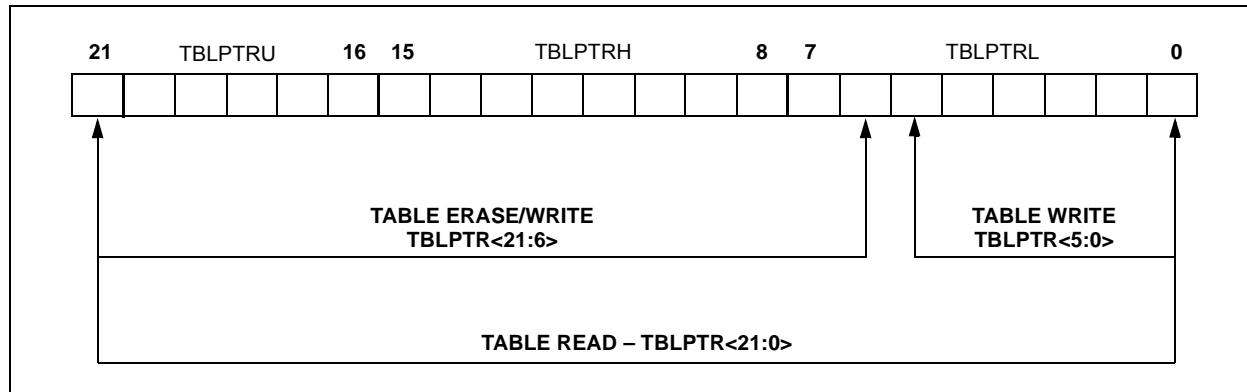
When an erase of program memory is executed, the 16 MSbs of the Table Pointer register (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

[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**



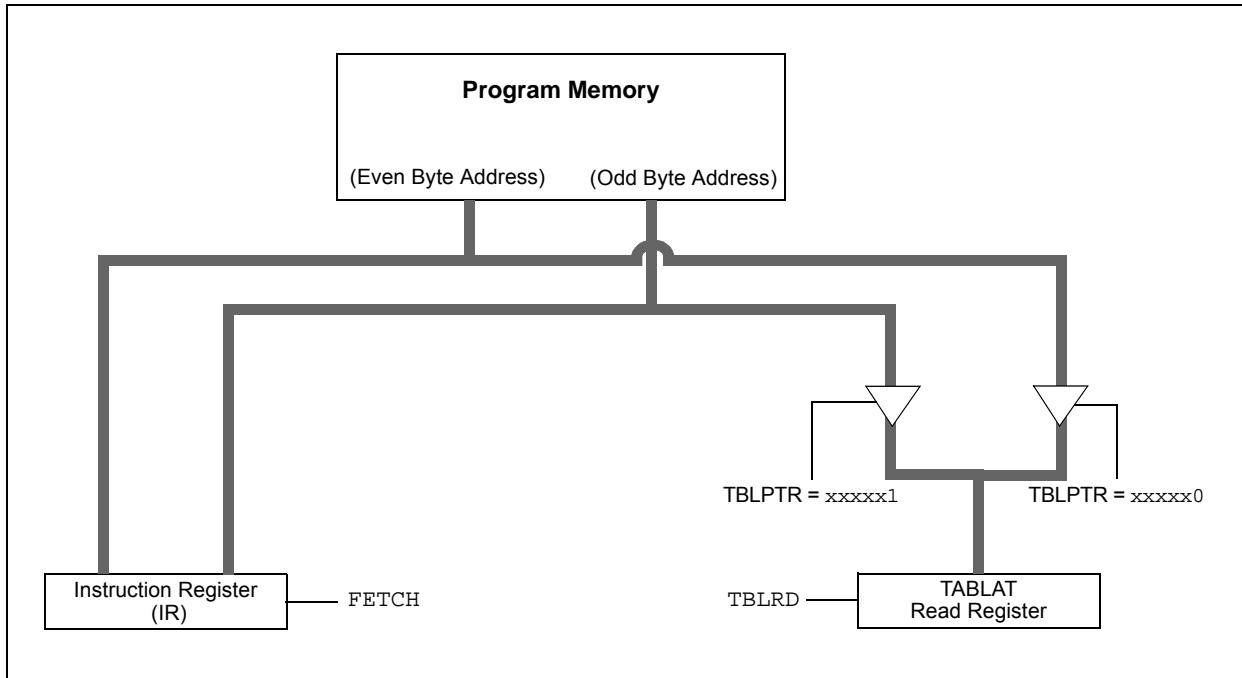
## 7.3 Reading the Flash Program Memory

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

```

MOVlw  CODE_ADDR_UPPER      ; Load TBLPTR with the base
MOVwf  TBLPTRU             ; address of the word
MOVlw  CODE_ADDR_HIGH
MOVwf  TBLPTRH
MOVlw  CODE_ADDR_LOW
MOVwf  TBLPTRL

READ_WORD
    TBLRD*+                  ; read into TABLAT and increment
    MOVF   TABLAT, W          ; get data
    MOVwf WORD_EVEN
    TBLRD*+                  ; read into TABLAT and increment
    MOVF   TABLAT, W          ; get data
    MOVF   WORD_ODD

```

# PIC18F66K80 FAMILY

## 7.4 Erasing Flash Program Memory

The erase blocks are 32 words or 64 bytes.

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. The TBLPTR<5:0> bits 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 the Table Pointer register with the address of row to be erased.
2. Set the EECON1 register for the erase operation:
  - Set the EEPGD bit to point to program memory
  - Clear the CFGS bit to access program memory
  - Set the WREN bit to enable writes
  - Set the FREE bit to enable the erase
3. Disable the interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit.

This begins the row erase cycle.

The CPU will stall for the duration of the erase for TiW. (See Parameter [D133A](#).)

7. 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
Required Sequence	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 programming blocks are 32 words or 64 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 64 holding registers for programming by the table writes.

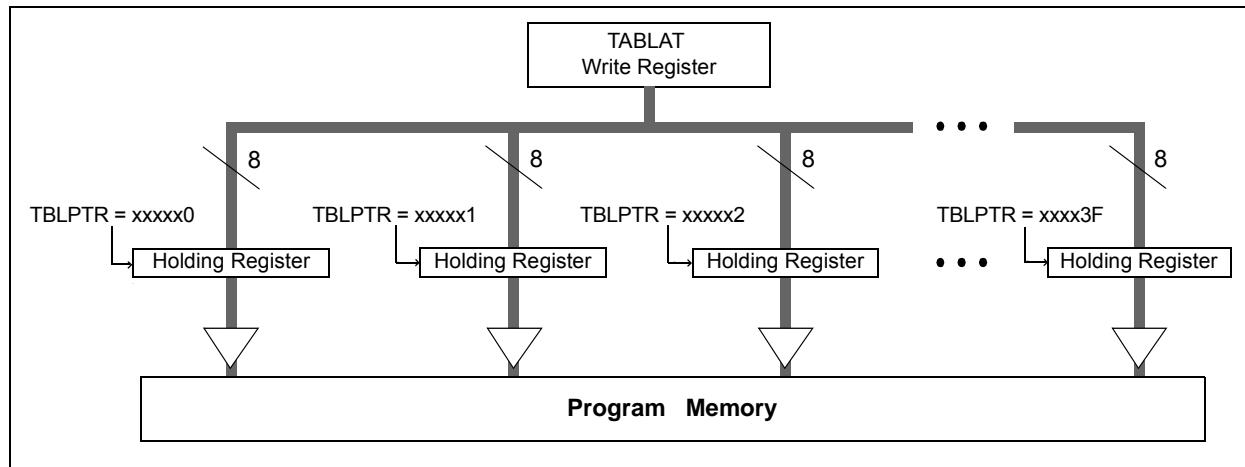
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 64 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the 64 or 128 holding 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 is 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 64 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 the 64 bytes into RAM.
2. Update the data values in RAM as necessary.
3. Load the Table Pointer register with the address being erased.
4. Execute the row erase procedure.
5. Load the Table Pointer register with the address of the first byte being written.
6. Write the 64 bytes into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
  - Set the EEPGD bit to point to program memory
  - Clear the CFGS bit to access program memory
  - Set the WREN to enable byte writes
8. Disable the interrupts.

9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the WR bit. This will begin the write cycle. The CPU will stall for the duration of the write for TIW (see Parameter D133A).
12. Re-enable the interrupts.
13. Verify the memory (table read).

An example of the required code is shown in [Example 7-3](#) on the following page.

**Note:** Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the 64 bytes in the holding register.

# PIC18F66K80 FAMILY

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

```
        MOVLW  SIZE_OF_BLOCK          ; number of bytes in erase block
        MOVWF  COUNTER
        MOVLW  BUFFER_ADDR_HIGH      ; point to buffer
        MOVWF  FSROH
        MOVLW  BUFFER_ADDR_LOW
        MOVWF  FSROL
        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*+
        MOVF   TABLAT, W            ; read into TABLAT, and inc
        MOVWF  POSTINCO             ; get data
        DECFSZ COUNTER              ; store data
        BRA    READ_BLOCK            ; done?
        BRA    READ_BLOCK            ; repeat

MODIFY_WORD
        MOVLW  DATA_ADDR_HIGH       ; point to buffer
        MOVWF  FSROH
        MOVLW  DATA_ADDR_LOW
        MOVWF  FSROL
        MOVLW  NEW_DATA_LOW         ; update buffer word
        MOVWF  POSTINCO
        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
        MOVWF  EECON2                ; write 55h
        MOVLW  0AAh
        MOVWF  EECON2                ; write 0AAh
        BSF    EECON1, WR              ; start erase (CPU stall)
        BSF    INTCON, GIE             ; re-enable interrupts
        TBLRD*-
        MOVLW  BUFFER_ADDR_HIGH      ; dummy read decrement
        MOVWF  FSROH
        MOVLW  BUFFER_ADDR_LOW
        MOVWF  FSROL

WRITE_BUFFER_BACK
        MOVLW  SIZE_OF_BLOCK          ; number of bytes in holding register
        MOVWF  COUNTER

WRITE_BYTE_TO_HREGS
        MOVFF  POSTINCO, 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_BYTE_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
<b>Required Sequence</b>	MOVLW 55h
	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 28.0 “Special Features of the CPU”](#) for more detail.

#### 7.6 Flash Program Operation During Code Protection

See [Section 28.6 “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
TBLPTRU	—	—	bit 21 <sup>(1)</sup>	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)				
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)							
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)							
TABLAT	Program Memory Table Latch							
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
EECON2	EEPROM Control Register 2 (not a physical register)							
EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE

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

**Note 1:** Bit 21 of the TBLPTRU allows access to the device Configuration bits.

# PIC18F66K80 FAMILY

---

---

**NOTES:**

## 8.0 DATA EEPROM MEMORY

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, that is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space, but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Five SFRs are used to read and write to the data EEPROM, as well as the program memory. They are:

- EECON1
- EECON2
- EEDATA
- EEADR
- EEADRH

The data EEPROM allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and the EEADRH:EEADR register pair holds the address of the EEPROM location being accessed.

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; it will vary with voltage and temperature, as well as from chip-to-chip. Please refer to Parameter D122 (Table 31-1 in [Section 31.0 “Electrical Characteristics”](#)) for exact limits.

## 8.1 EEADR and EEADRH Registers

The EEADRH:EEADR register pair is used to address the data EEPROM for read and write operations. EEADRH holds the two MSbs of the address; the upper 6 bits are ignored. The 10-bit range of the pair can address a memory range of 1024 bytes (00h to 3FFh).

## 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 8-1](#)) is the control register for data and program memory access. Control bit, EEPGD, determines if the access will be to program memory 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:** During normal operation, the WRERR 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.

**Note:** The EEIF interrupt flag bit (PIR4<6>) is set when the write is complete. It must be cleared in 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.

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.

# PIC18F66K80 FAMILY

## REGISTER 8-1: EECON1: DATA 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	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

- bit 7      **EEPGD:** Flash Program or Data EEPROM Memory Select bit  
1 = Accesses Flash program memory  
0 = Accesses data EEPROM memory
- bit 6      **CFGs:** Flash Program/Data EEPROM or Configuration Select bit  
1 = Accesses Configuration registers  
0 = Accesses Flash program or data EEPROM memory
- bit 5      **Unimplemented:** Read as '0'
- bit 4      **FREE:** Flash Row Erase Enable bit  
1 = Erases the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)  
0 = Performs 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 the 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 EEPGD = 1 or CFGs = 1.)  
0 = Does not initiate an EEPROM read

**Note 1:** When a WRERR occurs, the EEPGD and CFGs bits are 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 EEADRH:EEADR register pair, clear the EEPGD control bit (EECON1<7>) and then set control bit, RD (EECON1<0>). The data is available after one instruction cycle, in the EEDATA register. It can be read after one NOP instruction. EEDATA will hold this value until another read operation or until it is written to by the user (during a write operation).

The basic process is shown in [Example 8-1](#).

## 8.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADRH:EEADR register pair 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, EEADRH: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.

**Note:** Self-write execution to Flash and EEPROM memory cannot be done while running in LP Oscillator (low-power) mode. Executing a self-write will put the device into High-Power mode.

# PIC18F66K80 FAMILY

---

---

## EXAMPLE 8-1: DATA EEPROM READ

```
MOVlw  DATA_EE_ADDRH      ;  
MOVwf  EEADRH           ; Upper bits of Data Memory Address to read  
MOVlw  DATA_EE_ADDR      ;  
MOVwf  EEADR             ; Lower bits of Data Memory Address to read  
BCF    EECON1, EEPGD     ; Point to DATA memory  
BCF    EECON1, CFGS      ; Access EEPROM  
BSF    EECON1, RD        ; EEPROM Read  
NOP  
MOVf   EEDATA, W         ; W = EEDATA
```

## EXAMPLE 8-2: DATA EEPROM WRITE

```
MOVLW  DATA_EE_ADDRH      ;  
MOVWF  EEADRH           ; Upper bits of Data Memory Address to write  
MOVLW  DATA_EE_ADDR      ;  
MOVWF  EEADR             ; Lower bits of Data Memory Address to write  
MOVLW  DATA_EE_DATA      ;  
MOVWF  EEDATA            ; Data Memory Value to write  
BCF    EECON1, EEPGD     ; Point to DATA memory  
BCF    EECON1, CFGS      ; Access EEPROM  
BSF    EECON1, WREN      ; Enable writes  
  
BCF    INTCON, GIE       ; Disable Interrupts  
MOVLW  55h               ;  
Required Sequence MOVWF  EECON2          ; Write 55h  
MOVLW  0AAh               ;  
MOVWF  EECON2          ; Write 0AAh  
BSF    EECON1, WR        ; Set WR bit to begin write  
BTFS  EECON1, WR        ; Wait for write to complete GOTO $-2  
BSF    INTCON, GIE       ; Enable Interrupts  
  
; User code execution  
BCF    EECON1, WREN      ; Disable writes on write complete (EEIF set)
```

## 8.6 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is 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 28.0 “Special Features of the CPU”](#) for additional information.

## 8.7 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 implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT, Parameter 33).

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

## 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 Parameter 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 often, an array refresh is likely not required. See Parameter D124.

### EXAMPLE 8-3: DATA EEPROM REFRESH ROUTINE

```
CLRF EEADR ; Start at address 0
CLRF EEADRH ;
BCF EECON1, CFGS ; Set for memory
BCF EECON1, EEPGD ; Set for Data EEPROM
BCF INTCON, GIE ; Disable interrupts
BSF EECON1, WREN ; Enable writes
LOOP
    BSF EECON1, RD ; Read current address
    MOVLW 55h ;
    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
    INCFSZ EEADR, F ; Increment address
    BRA LOOP ; Not zero, do it again
    INCFSZ EEADRH, F ; Increment the high address
    BRA LOOP ; Not zero, do it again
    BCF EECON1, WREN ; Disable writes
    BSF INTCON, GIE ; Enable interrupts
```

# PIC18F66K80 FAMILY

---

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
EEADDRH	EEPROM Address Register High Byte							
EEADR	EEPROM Address Register Low Byte							
EEDATA	EEPROM Data Register							
EECON2	EEPROM Control Register 2 (not a physical register)							
EECON1	EEPGD	CFG5	—	FREE	WRERR	WREN	WR	RD
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE

**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 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
BTFS C ARG2, SB   ; Test Sign Bit
SUBWF PRODH, F    ; PRODH = PRODH
                  ; - ARG1
MOVF ARG2, W      ;
BTFS C 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			
				@ 64 MHz	@ 48 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	4.3 µs	5.7 µs	27.6 µs	69 µs
	Hardware multiply	1	1	62.5 ns	83.3 ns	400 ns	1 µs
8 x 8 signed	Without hardware multiply	33	91	5.6 µs	7.5 µs	36.4 µs	91 µs
	Hardware multiply	6	6	375 ns	500 ns	2.4 µs	6 µs
16 x 16 unsigned	Without hardware multiply	21	242	15.1 µs	20.1 µs	96.8 µs	242 µs
	Hardware multiply	28	28	1.7 µs	2.3 µs	11.2 µs	28 µs
16 x 16 signed	Without hardware multiply	52	254	15.8 µs	21.2 µs	101.6 µs	254 µs
	Hardware multiply	35	40	2.5 µs	3.3 µs	16.0 µs	40 µs

# PIC18F66K80 FAMILY

[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

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

## 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

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

## 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   ;
;

BTFS ARG2H, 7     ; ARG2H:ARG2L neg?
BRA SIGN_ARG1    ; no, check ARG1
MOVF ARG1L, W    ;
SUBWF RES2        ;
MOVF ARG1H, W    ;
SUBWFB RES3        ;
;

SIGN_ARG1
BTFS ARG1H, 7     ; ARG1H:ARG1L neg?
BRA CONT_CODE    ; no, done
MOVF ARG2L, W    ;
SUBWF RES2        ;
MOVF ARG2H, W    ;
SUBWFB RES3        ;
;

CONT_CODE
:
```

## 10.0 INTERRUPTS

Members of the PIC18F66K80 family of 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.

The registers for controlling interrupt operation are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3, PIR4 and PIR5
- PIE1, PIE2, PIE3, PIE4 and PIE5
- IPR1, IPR2, IPR3, IPR4 and IPR5

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** – Indicating that an interrupt event occurred
- **Enable bit** – Enabling program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** – Specifying 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 that 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 that enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit that 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 (ISR), 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.

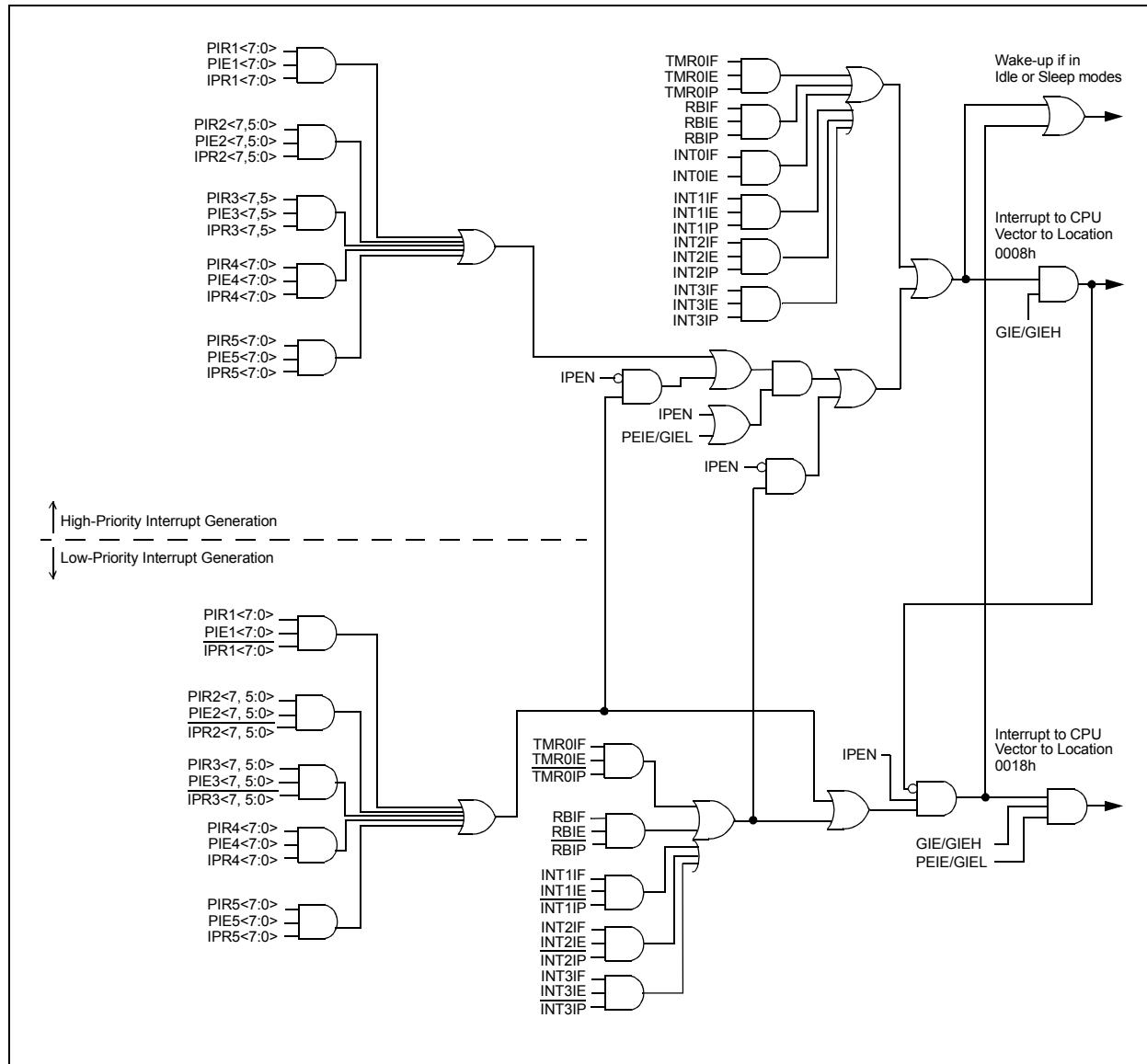
The “return from interrupt” instruction, RETFIE, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used) that re-enables interrupts.

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

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

# PIC18F66K80 FAMILY

**FIGURE 10-1: PIC18F66K80 FAMILY INTERRUPT LOGIC**



## 10.1 INTCON Registers

The INTCON registers are readable and writable registers that 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 10-1: 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 <sup>(2)</sup>	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	<b>GIE/GIEH:</b> Global Interrupt Enable bit <u>When IPEN = 0:</u> 1 = Enables all unmasked interrupts 0 = Disables all interrupts <u>When IPEN = 1:</u> 1 = Enables all high-priority interrupts 0 = Disables all interrupts
bit 6	<b>PEIE/GIEL:</b> Peripheral Interrupt Enable bit <u>When IPEN = 0:</u> 1 = Enables all unmasked peripheral interrupts 0 = Disables all peripheral interrupts <u>When IPEN = 1:</u> 1 = Enables all low-priority peripheral interrupts 0 = Disables all low-priority peripheral interrupts
bit 5	<b>TMR0IE:</b> TMR0 Overflow Interrupt Enable bit 1 = Enables the TMR0 overflow interrupt 0 = Disables the TMR0 overflow interrupt
bit 4	<b>INT0IE:</b> INT0 External Interrupt Enable bit 1 = Enables the INT0 external interrupt 0 = Disables the INT0 external interrupt
bit 3	<b>RBIE:</b> RB Port Change Interrupt Enable bit <sup>(2)</sup> 1 = Enables the RB port change interrupt 0 = Disables the RB port change interrupt
bit 2	<b>TMR0IF:</b> TMR0 Overflow Interrupt Flag bit 1 = TMR0 register has overflowed (must be cleared in software) 0 = TMR0 register has not overflowed
bit 1	<b>INT0IF:</b> 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	<b>RBIF:</b> RB Port Change Interrupt Flag bit <sup>(1)</sup> 1 = At least one of the RB<7:4> pins changed state (must be cleared in software) 0 = None of the RB<7:4> pins have changed state

**Note 1:** A mismatch condition will continue to set this bit. To end the mismatch condition and allow the bit to be cleared, read PORTB and wait one additional instruction cycle.

**2:** Each pin on PORTB for interrupt-on-change is individually enabled and disabled in the IOCB register. By default, all pins are disabled.

# PIC18F66K80 FAMILY

## REGISTER 10-2: 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
RBPU	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      **RBPU:** PORTB Pull-up Enable bit  
1 = All PORTB pull-ups are disabled  
0 = PORTB pull-ups are enabled by individual port TRIS 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.

# PIC18F66K80 FAMILY

## REGISTER 10-3: 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	<b>INT2IP:</b> INT2 External Interrupt Priority bit 1 = High priority 0 = Low priority
bit 6	<b>INT1IP:</b> INT1 External Interrupt Priority bit 1 = High priority 0 = Low priority
bit 5	<b>INT3IE:</b> INT3 External Interrupt Enable bit 1 = Enables the INT3 external interrupt 0 = Disables the INT3 external interrupt
bit 4	<b>INT2IE:</b> INT2 External Interrupt Enable bit 1 = Enables the INT2 external interrupt 0 = Disables the INT2 external interrupt
bit 3	<b>INT1IE:</b> INT1 External Interrupt Enable bit 1 = Enables the INT1 external interrupt 0 = Disables the INT1 external interrupt
bit 2	<b>INT3IF:</b> 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	<b>INT2IF:</b> 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	<b>INT1IF:</b> 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.

# PIC18F66K80 FAMILY

## 10.2 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are six Peripheral Interrupt Request (Flag) registers (PIR1 through PIR5).

**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 10-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	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      **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit  
1 = A read or write operation has taken place (must be cleared in software)  
0 = No read or write operation has occurred
- bit 6      **ADIF:** A/D Converter Interrupt Flag bit  
1 = An A/D conversion completed (must be cleared in software)  
0 = The A/D conversion is not complete
- bit 5      **RC1IF:** EUSARTx Receive Interrupt Flag bit  
1 = The EUSARTx receive buffer, RCREG1, is full (cleared when RCREG1 is read)  
0 = The EUSARTx receive buffer is empty
- bit 4      **TX1IF:** EUSARTx Transmit Interrupt Flag bit  
1 = The EUSARTx transmit buffer, TXREG1, is empty (cleared when TXREG1 is written)  
0 = The EUSARTx transmit buffer is full
- bit 3      **SSPIF:** Master Synchronous Serial Port Interrupt Flag bit  
1 = The transmission/reception is complete (must be cleared in software)  
0 = Waiting to transmit/receive
- bit 2      **TMR1GIF:** Timer1 Gate Interrupt Flag bit  
1 = Timer gate interrupt occurred (must be cleared in software)  
0 = No timer gate interrupt occurred
- bit 1      **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit  
1 = TMR2 to PR2 match occurred (must be cleared in software)  
0 = No TMR2 to PR2 match occurred
- bit 0      **TMR1IF:** TMR1 Overflow Interrupt Flag bit  
1 = TMR1 register overflowed (must be cleared in software)  
0 = TMR1 register did not overflow

# PIC18F66K80 FAMILY

## REGISTER 10-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIF	—	—	—	BCLIF	HLVDIF	TMR3IF	TMR3GIF
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 (bit must be cleared in software)  
0 = Device clock is operating
- bit 6-4     **Unimplemented:** Read as '0'
- bit 3       **BCLIF:** Bus Collision Interrupt Flag bit  
1 = A bus collision occurred (bit must be cleared in software)  
0 = No bus collision occurred
- bit 2       **HLVDIF:** High/Low-Voltage Detect Interrupt Flag bit  
1 = A low-voltage condition occurred (bit must be cleared in software)  
0 = The device voltage is above the regulator's low-voltage trip point
- bit 1       **TMR3IF:** TMR3 Overflow Interrupt Flag bit  
1 = TMR3 register overflowed (bit must be cleared in software)  
0 = TMR3 register did not overflow
- bit 0       **TMR3GIF:** TMR3 Gate Interrupt Flag bit  
1 = Timer gate interrupt occurred (bit must be cleared in software)  
0 = No timer gate interrupt occurred

# PIC18F66K80 FAMILY

## REGISTER 10-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3

U-0	U-0	R-0	R-0	R/W-0	R/W-0	R/W-0	U-0
—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
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      **RC2IF:** EUSARTx Receive Interrupt Flag bit  
1 = The EUSARTx receive buffer, RCREG2, is full (cleared when RCREG2 is read)  
0 = The EUSARTx receive buffer is empty
- bit 4      **TX2IF:** EUSARTx Transmit Interrupt Flag bit  
1 = The EUSARTx transmit buffer, TXREG2, is empty (cleared when TXREG2 is written)  
0 = The EUSARTx transmit buffer is full
- bit 3      **CTMUIF:** CTMU Interrupt Flag bit  
1 = CTMU interrupt occurred (must be cleared in software)  
0 = No CTMU interrupt occurred
- bit 2      **CCP2IF:** CCP2 Interrupt Flag bit  
Capture mode:  
1 = A TMR1/TMR3 register capture occurred (must be cleared in software)  
0 = No TMR1/TMR3 register capture occurred  
Compare mode:  
1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)  
0 = No TMR1/TMR3 register compare match occurred  
PWM mode:  
Unused in this mode.
- bit 1      **CCP1IF:** ECCP1 Interrupt Flag bit  
Capture mode:  
1 = A TMR1/TMR3 register capture occurred (must be cleared in software)  
0 = No TMR1/TMR3 register capture occurred  
Compare mode:  
1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)  
0 = No TMR1/TMR3 register compare match occurred  
PWM mode:  
Unused in this mode.
- bit 0      **Unimplemented:** Read as '0'

# PIC18F66K80 FAMILY

## REGISTER 10-7: PIR4: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 4

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
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           **TMR4IF:** TMR4 Overflow Interrupt Flag bit  
                 1 = TMR4 register overflowed (must be cleared in software)  
                 0 = TMR4 register did not overflow
- bit 6           **EEIF:** Data EEDATA/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 5           **CMP2IF:** CMP2 Interrupt Flag bit  
                 1 = CMP2 interrupt occurred (must be cleared in software)  
                 0 = CMP2 interrupt did not occur
- bit 4           **CMP1IF:** CMP1 Interrupt Flag bit  
                 1 = CMP1 interrupt occurred (must be cleared in software)  
                 0 = CMP1 interrupt did not occur
- bit 3           **Unimplemented:** Read as '0'
- bit 2           **CCP5IF:** CCP5 Interrupt Flag bit  
                 Capture Mode  
                 1 = A TMR register capture occurred (bit must be cleared in software)  
                 0 = No TMR register capture occurred  
                 Compare Mode  
                 1 = A TMR register compare match occurred (must be cleared in software)  
                 0 = No TMR register compare match occurred  
                 PWM Mode  
                 Not used in PWM mode.
- bit 1           **CCP4IF:** CCP4 Interrupt Flag bit  
                 Capture Mode  
                 1 = A TMR register capture occurred (bit must be cleared in software)  
                 0 = No TMR register capture occurred  
                 Compare Mode  
                 1 = A TMR register compare match occurred (must be cleared in software)  
                 0 = No TMR register compare match occurred  
                 PWM Mode  
                 Not used in PWM mode.
- bit 0           **CCP3IF:** CCP3 Interrupt Flag bit  
                 Capture Mode  
                 1 = A TMR register capture occurred (bit must be cleared in software)  
                 0 = No TMR register capture occurred  
                 Compare Mode  
                 1 = A TMR register compare match occurred (must be cleared in software)  
                 0 = No TMR register compare match occurred  
                 PWM Mode  
                 Not used in PWM mode.

# PIC18F66K80 FAMILY

## REGISTER 10-8: PIR5: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 5

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

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

- bit 7      **IRXIF:** Invalid Message Received Interrupt Flag bits  
1 = An invalid message occurred on the CAN bus  
0 = No invalid message occurred on the CAN bus
- bit 6      **WAKIF:** Bus Wake-up Activity Interrupt Flag bit  
1 = Activity on the CAN bus has occurred  
0 = No activity on the CAN bus
- bit 5      **ERRIF:** Error Interrupt Flag bit (Multiple sources in COMSTAT register)  
1 = An error has occurred in the CAN module (multiple sources)  
0 = No CAN module errors have occurred
- bit 4      **TXB2IF:** Transmit Buffer 2 Interrupt Flag bit  
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded  
0 = Transmit Buffer 2 has not completed transmission of a message
- bit 3      **TXB1IF:** Transmit Buffer 1 Interrupt Flag bit  
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded  
0 = Transmit Buffer 1 has not completed transmission of a message
- bit 2      **TXB0IF:** Transmit Buffer 0 Interrupt Flag bit  
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded  
0 = Transmit Buffer 0 has not completed transmission of a message
- bit 1      **RXB1IF:** Receive Buffer 1 Interrupt Flag bit  
Mode 0:  
1 = CAN Receive Buffer 1 has received a new message  
0 = CAN Receive Buffer 1 has not received a new message  
Modes 1 and 2:  
1 = A CAN Receive Buffer/FIFO has received a new message  
0 = A CAN Receive Buffer/FIFO has not received a new message
- bit 0      Bit operation is dependent on the selected mode:  
Mode 0:  
**RXB0IF:** Receive Buffer 0 Interrupt Flag bit  
1 = CAN Receive Buffer 0 has received a new message  
0 = CAN Receive Buffer 0 has not received a new message  
Mode 1:  
**Unimplemented:** Read as '0'  
Mode 2:  
**FIFOIF:** FIFO Full Interrupt Flag bit  
1 = FIFO has reached full status as defined by the FIFO\_HF bit  
0 = FIFO has not reached full status as defined by the FIFO\_HF bit

## 10.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are six Peripheral Interrupt Enable registers (PIE1 through PIE6). When IPEN (RCON<7>) = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

**REGISTER 10-9: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
bit 7	bit 0						

**Legend:**

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

- |       |  |
|-------|--|
| bit 7 | <b>PSPIE:</b> Parallel Slave Port Read/Write Interrupt Enable bit<br>1 = Enables the PSP read/write interrupt<br>0 = Disables the PSP read/write interrupt |
| bit 6 | <b>ADIE:</b> A/D Converter Interrupt Enable bit<br>1 = Enables the A/D interrupt<br>0 = Disables the A/D interrupt   |
| bit 5 | <b>RC1IE:</b> EUSARTx Receive Interrupt Enable bit<br>1 = Enables the EUSARTx receive interrupt<br>0 = Disables the EUSARTx receive interrupt              |
| bit 4 | <b>TX1IE:</b> EUSARTx Transmit Interrupt Enable bit<br>1 = Enables the EUSARTx transmit interrupt<br>0 = Disables the EUSARTx transmit interrupt           |
| bit 3 | <b>SSPIE:</b> Master Synchronous Serial Port Interrupt Enable bit<br>1 = Enables the MSSP interrupt<br>0 = Disables the MSSP interrupt                     |
| bit 2 | <b>TMR1GIE:</b> TMR1 Gate Interrupt Enable bit<br>1 = Enables the gate<br>0 = Disabled the gate  |
| bit 1 | <b>TMR2IE:</b> TMR2 to PR2 Match Interrupt Enable bit<br>1 = Enables the TMR2 to PR2 match interrupt<br>0 = Disables the TMR2 to PR2 match interrupt       |
| bit 0 | <b>TMR1IE:</b> TMR1 Overflow Interrupt Enable bit<br>1 = Enables the TMR1 overflow interrupt<br>0 = Disables the TMR1 overflow interrupt                   |

# PIC18F66K80 FAMILY

## REGISTER 10-10: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIE	—	—	—	BCLIE	HLVDIE	TMR3IE	TMR3GIE
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 6-4      **Unimplemented:** Read as '0'

bit 3      **BCLIE:** Bus Collision Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 2      **HLVDIE:** High/Low-Voltage Detect Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 1      **TMR3IE:** TMR3 Overflow Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 0      **TMR3GIE:** Timer3 Gate Interrupt Enable bit

1 = Enabled

0 = Disabled

# PIC18F66K80 FAMILY

## REGISTER 10-11: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

U-0	U-0	R-0	R-0	R/W-0	R/W-0	R/W-0	U-0
—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
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      **RC2IE:** EUSARTx Receive Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 4      **TX2IE:** EUSARTx Transmit Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 3      **CTMUIE:** CTMU Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 2      **CCP2IE:** CCP2 Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 1      **CCP1IE:** ECCP1 Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 0      **Unimplemented:** Read as '0'

# PIC18F66K80 FAMILY

## REGISTER 10-12: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
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           **TMR4IE:** TMR4 Overflow Interrupt Flag bit

1 = Interrupt is enabled

0 = Interrupt is disabled

bit 6           **EEIE:** Data EEDATA/Flash Write Operation Interrupt Flag bit

1 = Interrupt is enabled

0 = Interrupt is disabled

bit 5           **CMP2IE:** CMP2 Interrupt Flag bit

1 = Interrupt is enabled

0 = Interrupt is disabled

bit 4           **CMP1IE:** CMP1 Interrupt Flag bit

1 = Interrupt is enabled

0 = Interrupt is disabled

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

bit 2           **CCP5IE:** CCP5 Interrupt Flag bit

1 = Interrupt is enabled

0 = Interrupt is disabled

bit 1           **CCP4IE:** CCP4 Interrupt Flag bit

1 = Interrupt is enabled

0 = Interrupt is disabled

bit 0           **CCP3IE:** CCP3 Interrupt Flag bits

1 = Interrupt is enabled

0 = Interrupt is disabled

# PIC18F66K80 FAMILY

## REGISTER 10-13: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIE	WAKIE	ERRIE	TXB2IE	TXB1IE	TXB0IE	RXB1IE	RXB0IE/ FIFOIE
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      **IRXIE:** Invalid Message Received Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled
- bit 6      **WAKIE:** Bus Wake-up Activity Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled
- bit 5      **ERRIE:** Error Interrupt Flag bit (multiple sources in the COMSTAT register)  
1 = Interrupt is enabled  
0 = Interrupt is disabled
- bit 4      **TXB2IE:** Transmit Buffer 2 Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled
- bit 3      **TXB1IE:** Transmit Buffer 1 Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled
- bit 2      **TXB0IE:** Transmit Buffer 0 Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled
- bit 1      **RXB1IE:** Receive Buffer 1 Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled
- bit 0      Bit operation is dependent on the selected mode:  
Mode 0:  
**RXB0IE:** Receive Buffer 0 Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled  
Mode 1:  
**Unimplemented:** Read as '0'  
Mode 2:  
**FIFOIE:** FIFO Full Interrupt Flag bit  
1 = Interrupt is enabled  
0 = Interrupt is disabled

# PIC18F66K80 FAMILY

---

---

## 10.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are six Peripheral Interrupt Priority registers (IPR1 through IPR6). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit (RCON<7>) be set.

**REGISTER 10-14: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1**

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
bit 7							

**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      **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 6      **ADIP:** A/D Converter Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 5      **RC1IP:** EUSARTx Receive Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 4      **TX1IP:** EUSARTx Transmit Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 3      **SSPIP:** Master Synchronous Serial Port Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 2      **TMR1GIP:** Timer1 Gate Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 1      **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 0      **TMR1IP:** TMR1 Overflow Interrupt Priority bit  
1 = High priority  
0 = Low priority

# PIC18F66K80 FAMILY

## REGISTER 10-15: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

R/W-1	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	—	—	—	BCLIP	HLVDIP	TMR3IP	TMR3GIP
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 6-4     **Unimplemented:** Read as '0'
- bit 3       **BCLIP:** Bus Collision Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 2       **HLVDIP:** High/Low-Voltage Detect Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 1       **TMR3IP:** TMR3 Overflow Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 0       **TMR3GIP:** TMR3 Gate Interrupt Priority bit  
1 = High priority  
0 = Low priority

# PIC18F66K80 FAMILY

## REGISTER 10-16: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
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      **RC2IP:** EUSARTx Receive Priority Flag bit

1 = High priority  
0 = Low priority

bit 4      **TX2IP:** EUSARTx Transmit Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 3      **CTMUIP:** CTMU Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 2      **CCP2IP:** CCP2 Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 1      **CCP1IP:** ECCP1 Interrupt Priority bit

1 = High priority  
0 = Low priority

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

# PIC18F66K80 FAMILY

## REGISTER 10-17: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	R/W-1	R/W-1
TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
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           **TMR4IP:** TMR4 Overflow Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 6           **EEIP:** EE Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 5           **CMP2IP:** CMP2 Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 4           **CMP1IP:** CMP1 Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 3           **Unimplemented:** Read as '0'
- bit 2           **CCP5IP:** CCP5 Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 1           **CCP4IP:** CCP4 Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit             **CCP3IP:** CCP3 Interrupt Priority bits  
1 = High priority  
0 = Low priority

# PIC18F66K80 FAMILY

## REGISTER 10-18: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

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

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

bit 7      **IRXIP:** Invalid Message Received Interrupt Priority bits

1 = High priority  
0 = Low priority

bit 6      **WAKIP:** Bus Wake-up Activity Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 5      **ERRIP:** CAN Bus Error Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 4      **TXB2IP:** Transmit Buffer 2 Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 3      **TXB1IP:** Transmit Buffer 1 Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 2      **TXB0IP:** Transmit Buffer 0 Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 1      **RXB1IP:** Receive Buffer 1 Interrupt Priority bit

**Mode 0:**

1 = High priority for Receive Buffer 1  
0 = Low priority for Receive Buffer 1

**Modes 1 and 2:**

1 = High priority for received messages  
0 = Low priority for received messages

bit 0      **RXB0IP/FIFOIE:** Receive Buffer 0 Interrupt Priority bit

**Mode 0:**

1 = High priority for Receive Buffer 0  
0 = Low priority for Receive Buffer 0

**Mode 1:**

**Unimplemented:** Read as '0'

**Mode 2:**

**FIFOIE:** FIFO Full Interrupt Flag bit

1 = High priority  
0 = Low priority

## 10.5 RCON Register

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

### REGISTER 10-19: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1	R/W-1	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	SBOREN	<u>CM</u>	<u>RI</u>	<u>TO</u>	<u>PD</u>	<u>POR</u>	<u>BOR</u>
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 | <b>IPEN:</b> Interrupt Priority Enable bit<br>1 = Enables priority levels on interrupts<br>0 = Disables priority levels on interrupts (PIC16CXXX Compatibility mode)                        |
| bit 6 | <b>SBOREN:</b> Software BOR Enable bit<br>For details of bit operation, see <a href="#">Register 5-1</a> .  |
| bit 5 | <b>CM:</b> Configuration Mismatch Flag bit<br>1 = A Configuration Mismatch Reset has not occurred<br>0 = A Configuration Mismatch Reset has occurred (must be subsequently set in software) |
| bit 4 | <b>RI:</b> RESET Instruction Flag bit<br>For details of bit operation, see <a href="#">Register 5-1</a> .   |
| bit 3 | <b>TO:</b> Watchdog Timer Time-out Flag bit<br>For details of bit operation, see <a href="#">Register 5-1</a> .   |
| bit 2 | <b>PD:</b> Power-Down Detection Flag bit<br>For details of bit operation, see <a href="#">Register 5-1</a> .  |
| bit 1 | <b>POR:</b> Power-on Reset Status bit<br>For details of bit operation, see <a href="#">Register 5-1</a> .   |
| bit 0 | <b>BOR:</b> Brown-out Reset Status bit<br>For details of bit operation, see <a href="#">Register 5-1</a> .  |

# PIC18F66K80 FAMILY

## 10.6 INTx Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1, RB2/INT2 and RB3/INT3 pins are edge-triggered. If the corresponding INTEDG<sub>x</sub> bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge. If that bit is clear, the trigger is on the falling edge.

When a valid edge appears on the RB<sub>x</sub>/INT<sub>x</sub> pin, the corresponding flag bit, INT<sub>x</sub>IF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INT<sub>x</sub>IE. Before re-enabling the interrupt, the flag bit (INT<sub>x</sub>IF) must be cleared in software in the Interrupt Service Routine.

All external interrupts (INT0, INT1, INT2 and INT3) can wake up the processor from the power-managed modes, if bit, INT<sub>x</sub>IE, was set prior to going into the power-managed modes. If the Global Interrupt Enable bit (GIE) is set, the processor will branch to the interrupt vector following wake-up.

The 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.

## REGISTER 10-20: IOCB: INTERRUPT-ON-CHANGE PORTB CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
IOCB7 <sup>(1)</sup>	IOCB6 <sup>(1)</sup>	IOCB5 <sup>(1)</sup>	IOCB4 <sup>(1)</sup>	—	—	—	—
bit 7	bit 0						

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

bit 7-4      **IOCB<7:4>**: Interrupt-on-Change PORTB Control bits<sup>(1)</sup>

1 = Interrupt-on-change is enabled  
0 = Interrupt-on-change is disabled

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

**Note 1:** Interrupt-on-change also requires that the RBIE bit of the INTCON register be set.

## 10.9 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 (ISR). Depending on the user's application, other registers also may need to be saved.

[Example 10-1](#) saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

### EXAMPLE 10-1: 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_TEMP located anywhere
;
; USER ISR CODE
;
MOVFF  BSR_TEMP, BSR      ; Restore BSR
MOVF   W_TEMP, W          ; Restore WREG
MOVFF  STATUS_TEMP, STATUS ; Restore STATUS

```

TABLE 10-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
PIR1	PSPIP	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIR2	OSCFIF	—	—	—	BCLIF	HLVDIF	TMR3IF	TMR3GIF
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIR5	IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF	TXB0IF	RXB1IF	RXB0IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
PIE2	OSCFIE	—	—	—	BCLIE	HLVDIE	TMR3IE	TMR3GIE
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
PIE4	TMR4IE	EEIE	CCP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
PIE5	IRXIE	WAKIE	ERRIE	TXB2IE	TXB1IE	TXB0IE	RXB1IE	RXB0IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
IPR2	OSCFIP	—	—	—	BCLIP	HLVDIP	TMR3IP	TMR3GIP
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
IPR5	IRXIP	WAKIP	ERRIP	TXB2IP	TXB1IP	TXB0IP	RXB1IP	RXB0IP
RCON	IPEN	SBOREN	CM	RI	TO	PD	POR	BOR

**Legend:** Shaded cells are not used by the interrupts.

# **PIC18F66K80 FAMILY**

---

---

## **NOTES:**

## 11.0 I/O PORTS

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

Each port has three memory mapped registers for its operation:

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

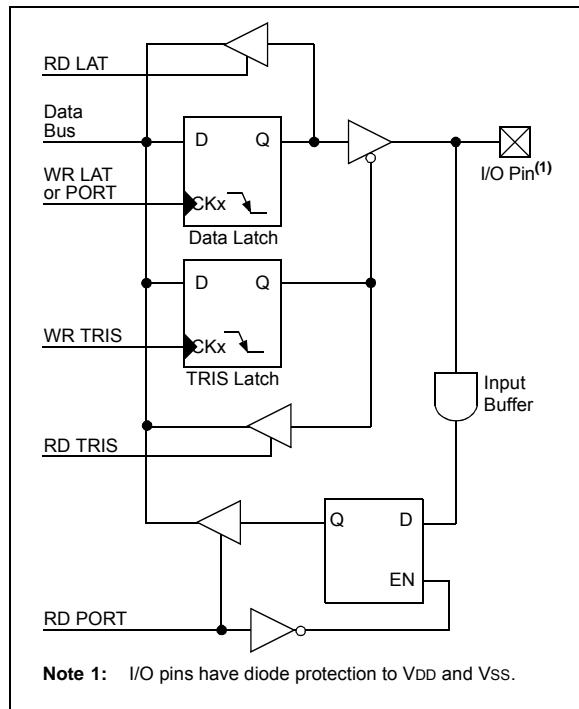
Reading the PORT register reads the current status of the pins, whereas writing to the PORT register, writes to the Output Latch (LAT) register.

Setting a TRIS bit (= 1) makes the corresponding port pin an input (putting the corresponding output driver in a High-Impedance mode). Clearing a TRIS bit (= 0) makes the corresponding port pin an output (i.e., put the contents of the corresponding LAT bit on the selected pin).

The Output Latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving. Read-modify-write operations on the LAT register read and write the latched output value for the PORT register.

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in [Figure 11-1](#).

**FIGURE 11-1: GENERIC I/O PORT OPERATION**



## 11.1 I/O Port Pin Capabilities

When developing an application, the capabilities of the port pins must be considered. Outputs on some pins have higher output drive strength than others. Similarly, some pins can tolerate higher than VDD input levels.

All of the digital ports are 5.5V input tolerant. The analog ports have the same tolerance, having clamping diodes implemented internally.

### 11.1.1 PIN OUTPUT DRIVE

When used as digital I/O, the output pin drive strengths vary, according to the pins' grouping to meet the needs for a variety of applications. In general, there are two classes of output pins, in terms of drive capability:

- Outputs that are designed to drive higher current loads, such as LEDs:
  - PORTA
  - PORTB
  - PORTC
- Outputs with lower drive levels, but capable of driving normal digital circuit loads with a high input impedance. Able to drive LEDs, but only those with smaller current requirements:
  - PORTD<sup>(1)</sup>
  - PORTE<sup>(1)</sup>
  - PORTF<sup>(2)</sup>
  - PORTG<sup>(2)</sup>

**Note 1:** These ports are not available on 28-pin devices.

**2:** These ports are not available on 28-pin or 40/44-pin devices

For more details, see “Absolute Maximum Ratings” in [Section 31.0 “Electrical Characteristics”](#).

### 11.1.2 PULL-UP CONFIGURATION

Five of the I/O ports (PORTB, PORTD, PORTE, PORTF and PORTG) implement configurable weak pull-ups on all pins. These are internal pull-ups that allow floating digital input signals to be pulled to a consistent level without the use of external resistors.

The pull-ups are enabled with a single bit for each of the ports: RBPU (INTCON2<7>) for PORTB, and RDPU, REPU, RFPU and RGPU (PADCFG1<7:4>) for the other ports.

Additionally, the PORTB pull-up resistors can be enabled individually using the WPUB register. Each bit in the register corresponds to a bit on PORTB.

# PIC18F66K80 FAMILY

## REGISTER 11-1: PADCFG1: PAD CONFIGURATION REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0
RDPU <sup>(1)</sup>	REPU <sup>(1)</sup>	RFP <sup>(2)</sup>	RGPU <sup>(2)</sup>	—	—	—	CTMUDS
bit 7	bit 0						

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

- bit 7      **RDPU:** PORTD Pull-up Enable bit<sup>(1)</sup>  
1 = PORTD pull-up resistors are enabled by individual port latch values  
0 = All PORTD pull-up resistors are disabled
- bit 6      **REPU:** PORTE Pull-up Enable bit<sup>(1)</sup>  
1 = PORTE pull-up resistors are enabled by individual port latch values  
0 = All PORTE pull-up resistors are disabled
- bit 5      **RFP<sup>(2)</sup>:** PORTF Pull-up Enable bit<sup>(2)</sup>  
1 = PORTF pull-up resistors are enabled by individual port latch values  
0 = All PORTF pull-up resistors are disabled
- bit 4      **RGPU:** PORTG Pull-up Enable bit<sup>(2)</sup>  
1 = PORTG pull-up resistors are enabled by individual port latch values  
0 = All PORTG pull-up resistors are disabled
- bit 3-1     **Unimplemented:** Read as '0'
- bit 0      **CTMUDS:** CTMU Comparator Data Select bit  
1 = External comparator (with output on pin CTDIN) is used for CTMU compares  
0 = Internal comparator (CMP2) is used for CTMU compares

**Note 1:** These bits are unimplemented on 28-pin devices.

**2:** These bits are unimplemented on 40-pin devices.

## REGISTER 11-2: WPUB: WEAK PULL-UP PORTB ENABLE REGISTER

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 |
| bit 7 | bit 0 |       |       |       |       |       |       |

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

- bit 7-0     **WPUB<7:0>:** Weak Pull-Up Enable Register bits  
1 = Pull-up is enabled on corresponding PORTB pin when RBPU = 0 and the pin is an input  
0 = Pull-up is disabled on corresponding PORTB pin

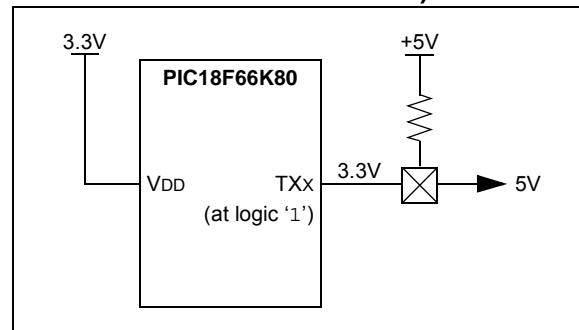
### 11.1.3 OPEN-DRAIN OUTPUTS

The output pins for several peripherals are also equipped with a configurable, open-drain output option. This allows the peripherals to communicate with external digital logic, operating at a higher voltage level, without the use of level translators.

The open-drain option is implemented on port pins specifically associated with the data and clock outputs of the USARTs, the MSSP module (in SPI mode) and the CCP modules. This option is selectively enabled by setting the open-drain control bits in the ODCON register.

When the open-drain option is required, the output pin must also be tied through an external pull-up resistor provided by the user to a higher voltage level, up to 5V (Figure 11-2). When a digital logic high signal is output, it is pulled up to the higher voltage level.

**FIGURE 11-2: USING THE OPEN-DRAIN OUTPUT (USARTx SHOWN AS EXAMPLE)**



### REGISTER 11-3: ODCON: PERIPHERAL OPEN-DRAIN 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
SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD
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	<b>SSPOD:</b> SPI Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled
bit 6	<b>CCP5OD:</b> CCP5 Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled
bit 5	<b>CCP4OD:</b> CCP4 Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled
bit 4	<b>CCP3OD:</b> CCP3 Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled
bit 3	<b>CCP2OD:</b> CCP2 Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled
bit 2	<b>CCP1OD:</b> CCP1 Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled
bit 1	<b>U2OD:</b> UART2 Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled
bit 0	<b>U1OD:</b> UART1 Open-Drain Output Enable bit 1 = Open-drain capability is enabled 0 = Open-drain capability is disabled

# PIC18F66K80 FAMILY

## 11.1.4 ANALOG AND DIGITAL PORTS

Many of the ports multiplex analog and digital functionality, providing a lot of flexibility for hardware designers. PIC18F66K80 family devices can make any analog pin analog or digital, depending on an application's needs. The ports' analog/digital functionality is controlled by the registers: ANCON0 and ANCON1.

Setting these registers makes the corresponding pins analog and clearing the registers makes the ports digital. For details on these registers, see [Section 23.0 “12-Bit Analog-to-Digital Converter \(A/D\) Module”](#)

## 11.1.5 PORT SLEW RATE

The output slew rate of each port is programmable to select either the standard transition rate, or a reduced transition rate of ten percent of the standard transition time, to minimize EMI. The reduced transition time is the default slew rate for all ports.

### REGISTER 11-4: SLRCON: SLEW RATE CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SLRG <sup>(1)</sup>	SLRF <sup>(1)</sup>	SLRE <sup>(2)</sup>	SLRD <sup>(2)</sup>	SLRC <sup>(2)</sup>	SLRB	SLRA
bit 7	bit 0						

#### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

- bit 7      **Unimplemented:** Read as '0'
- bit 6      **SLRG:** PORTG Slew Rate Control bit<sup>(1)</sup>  
1 = All output pins on PORTG slew at 0.1 the standard rate  
0 = All output pins on PORTG slew at standard rate
- bit 5      **SLRF:** PORTF Slew Rate Control bit<sup>(1)</sup>  
1 = All output pins on PORTF slew at 0.1 the standard rate  
0 = RAII output pins on PORTF slew at standard rate
- bit 4      **SLRE:** PORTE Slew Rate Control bit<sup>(2)</sup>  
1 = All output pins on PORTE slew at 0.1 the standard rate  
0 = All output pins on PORTE slew at standard rate
- bit 3      **SLRD:** PORTD Slew Rate Control bit<sup>(2)</sup>  
1 = All output pins on PORTD slew at 0.1 the standard rate  
0 = All output pins on PORTD slew at standard rate
- bit 2      **SLRC:** PORTC Slew Rate Control bit<sup>(2)</sup>  
1 = All output pins on PORTC slew at 0.1 the standard rate  
0 = All output pins on PORTC slew at standard rate
- bit 1      **SLRB:** PORTB Slew Rate Control bit  
1 = All output pins on PORTB slew at 0.1 the standard rate  
0 = All output pins on PORTB slew at standard rate
- bit 0      **SLRA:** PORTA Slew Rate Control bit  
1 = All output pins on PORTA slew at 0.1 the standard rate  
0 = All output pins on PORTA slew at standard rate

**Note 1:** These bits are unimplemented and read back as '0' on 28-pin and 40/44-pin devices.

**2:** These bits are unimplemented and read back as '0' on 28-pin devices.

## 11.2 PORTA, TRISA and LATA Registers

PORTA is a seven-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISA and LATA.

RA5 and RA<3:0> are multiplexed with analog inputs for the A/D Converter.

The operation of the analog inputs as A/D Converter inputs is selected by clearing or setting the ANSELx control bits in the ANCON1 register. The corresponding TRISA bits control the direction of these pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

**Note:** RA5 and RA<3:0> are configured as analog inputs on any Reset and are read as '0'.

OSC2/CLKO/RA6 and OSC1/CLKI/RA7 normally serve as the external circuit connections for the external (primary) oscillator circuit (HS Oscillator modes) or the external clock input and output (EC Oscillator modes). In these cases, RA6 and RA7 are not available as digital I/O and their corresponding TRIS and LAT bits are read as '0'. When the device is configured to use HF-INTOSC, MF-INTOSC or LF-INTOSC as the default oscillator mode, RA6 and RA7 are automatically configured as digital I/O; the oscillator and clock in/clock out functions are disabled.

RA5 has additional functionality for Timer1 and Timer3. It can be configured as the Timer1 clock input or the Timer3 external clock gate input.

### EXAMPLE 11-1: INITIALIZING PORTA

```
CLRF    PORTA    ; Initialize PORTA by
                  ; clearing output latches
CLRF    LATA     ; Alternate method to
                  ; clear output data latches
MOVLW  00h      ; Configure A/D
MOVWF  ANCON1   ; for digital inputs
MOVLW  0BFh    ; Value used to initialize
                  ; data direction
MOVWF  TRISA    ; Set RA<7, 5:0> as inputs,
                  ; RA<6> as output
```

# PIC18F66K80 FAMILY

---

**TABLE 11-1: PORTA FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RA0/CVREF/AN0/ ULPWU	RA0	0	O	DIG	LATA<0> data output; not affected by analog input.
		1	I	ST	PORTA<0> data input; disabled when analog input is enabled.
	CVREF	x	O	ANA	Comparator voltage reference output. Enabling this feature disables digital I/O.
	AN0	1	I	ANA	A/D Input Channel 0. Default input configuration on POR; does not affect digital output.
	ULPWU	1	O	DIG	Ultra Low-Power Wake-up input.
RA1/AN1/C1INC	RA1	0	O	DIG	LATA<1> data output; not affected by analog input.
		1	I	ST	PORTA<1> data input; disabled when analog input is enabled.
	AN1	1	I	ANA	A/D Input Channel 1. Default input configuration on POR; does not affect digital output.
	C1INC <sup>(1)</sup>	x	I	ANA	Comparator 1 Input C.
RA2/VREF-/AN2/ C2INC	RA2	0	O	DIG	LATA<2> data output; not affected by analog input.
		1	I	ST	PORTA<2> data input; disabled when analog functions are enabled.
	VREF-	1	I	ANA	A/D and comparator low reference voltage input.
	AN2	1	I	ANA	A/D Input Channel 2. Default input configuration on POR.
	C2INC <sup>(1)</sup>	x	I	ANA	Comparator 2 Input C.
RA3/VREF+/AN3	RA3	0	O	DIG	LATA<3> data output; not affected by analog input.
		1	I	ST	PORTA<3> data input; disabled when analog input is enabled.
	VREF+	1	I	ANA	A/D Input Channel 3. Default input configuration on POR.
	AN3	1	I	ANA	A/D and comparator high reference voltage input.
RA5/AN4/C2INB/ HLVDIN/T1CKI/ SS/CTMUI	RA5	0	O	DIG	LATA<5> data output; not affected by analog input.
		1	I	ST	PORTA<5> data input; disabled when analog input is enabled.
	AN4	1	I	ANA	A/D Input Channel 4. Default configuration on POR.
	C2INB <sup>(2)</sup>	1	I	ANA	Comparator 2 Input B.
	HLVDIN	1	I	ANA	High/Low-Voltage Detect external trip point input.
	T1CKI	x	I	ST	Timer1 clock input.
	SS	1	I	ST	Slave select input for MSSP module.
RA6/OSC2/ CLKOUT	RA6	0	O	DIG	LATA<6> data output; disabled when FOSC2 Configuration bit is set.
		1	I	ST	PORTA<6> data input; disabled when FOSC2 Configuration bit is set.
	OSC2	x	O	ANA	Main oscillator feedback output connection (HS, XT and LP modes).
	CLKOUT	x	O	DIG	System cycle clock output (Fosc/4) (EC and INTOSC modes).
RA7/OSC1/CLKIN	RA7	0	O	DIG	LATA<7> data output; disabled when FOSC2 Configuration bit is set.
		1	I	ST	PORTA<7> data input; disabled when FOSC2 Configuration bit is set.
	OSC1	x	I	ANA	Main oscillator input connection (HS, XT, and LP modes).
	CLKIN	x	I	ANA	Main external clock source input (EC modes).

**Legend:** O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** This pin assignment is unavailable for 28-pin devices (PIC18F2XK80).

**2:** This pin assignment is only available for 28-pin devices (PIC18F2XK80).

**TABLE 11-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	—	RA3	RA2	RA1	RA0
LATA	LATA7 <sup>(1)</sup>	LATA6 <sup>(1)</sup>	LATA5	—	LATA3	LATA2	LATA1	LATA0
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0
ANCON0	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0

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

**Note 1:** These bits are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as 'x'.

### 11.3 PORTB, TRISB and LATB Registers

PORTB is an eight-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISB and LATB. All pins on PORTB are digital only.

#### EXAMPLE 11-2: INITIALIZING PORTB

```
CLRF    PORTB    ; Initialize PORTB by
                  ; clearing output
                  ; data latches
CLRF    LATB     ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW   0CFh    ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISB    ; Set 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, 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.

Four of the PORTB pins (RB<7:4>) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur. Any RB<7:4> pins that are configured as outputs are excluded from the interrupt-on-change comparison.

Comparisons with the input pins (of RB<7:4>) are made with the old value latched on the last read of PORTB. The “mismatch” outputs of RB<7:4> are ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from power-managed modes. To clear the interrupt in the Interrupt Service Routine:

1. Perform any read or write of PORTB (except with the MOVFF (ANY), PORTB instruction).
  2. Wait one instruction cycle (such as executing a NOP instruction).
- This ends the mismatch condition.
3. Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit, RBIF, to be cleared after a one Tcy delay.

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

The RB<3:2> pins are multiplexed as CTMU edge inputs. RB5 has an additional function for Timer3 and Timer1. It can be configured for Timer3 clock input or Timer1 external clock gate input.

# PIC18F66K80 FAMILY

---

**TABLE 11-3: PORTB FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RB0/AN10/C1INA FLT0/INT0	RB0	0	O	DIG	LATB<0> data output.
		1	I	ST	PORTB<0> data input; weak pull-up when RBPU bit is cleared.
	AN10	1	I	ANA	A/D Input Channel 10 and Comparator C1+ input. Default input configuration on POR.
	C1INA <sup>(1)</sup>	1	I	ANA	Comparator 1 Input A.
	FLT0	x	I	ST	Enhanced PWM Fault input for ECCPx.
	INT0	1	I	ST	External Interrupt 0 input.
RB1/AN8/C1INB/ P1B/CTDIN/INT1	RB1	0	O	DIG	LATB<1> data output.
		1	I	ST	PORTB<1> data input; weak pull-up when RBPU bit is cleared.
	AN8	1	I	ANA	A/D Input Channel 8 and Comparator C2+ input. Default input configuration on POR; not affected by analog output.
	C1INB <sup>(1)</sup>	1	I	ANA	Comparator 1 Input B.
	P1B <sup>(1)</sup>	0	O	DIG	ECCP1 PWM Output B. May be configured for tri-state during Enhanced PWM shutdown events.
	CTDIN	1	I	ST	CTMU pulse delay input.
RB2/CANTX/C1OUT/ P1C/CTED1/INT2	RB2	0	O	DIG	LATB<2> data output.
		1	I	ST	PORTB<2> data input; weak pull-up when RBPU bit is cleared.
	CANTX <sup>(2)</sup>	0	O	DIG	CAN bus TX.
	C1OUT <sup>(1)</sup>	0	O	DIG	Comparator 1 output; takes priority over port data.
	P1C <sup>(1)</sup>	0	O	DIG	ECCP1 PWM Output C. May be configured for tri-state during Enhanced PWM.
	CTED1	x	I	ST	CTMU Edge 1 input.
RB3/CANRX/ C2OUT/P1D/ CTED2/INT3	RB3	0	O	DIG	LATB<3> data output.
		1	I	ST	PORTB<3> data input; weak pull-up when RBPU bit is cleared.
	CANRX <sup>(2)</sup>	1	I	ST	CAN bus RX.
	C2OUT <sup>(1)</sup>	x	I	ST	CTMU Edge 2 input.
	P1D <sup>(1)</sup>	0	O	DIG	ECCP1 PWM Output D. May be configured for tri-state during Enhanced PWM.
	CTED2	x	I	ST	CTMU Edge 2 input.
	INT3	1	I	ST	External Interrupt 3 input.

**Legend:** O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

- Note 1:** This pin assignment is only available for 28-pin devices (PIC18F2XK80).  
**2:** This is the default pin assignment for CANRX and CANTX when the CANMX Configuration bit is set.  
**3:** This is the default pin assignment for T0CKI when the T0CKMX Configuration bit is set.  
**4:** This is the default pin assignment for T3CKI for 28, 40 and 44-pin devices. This is the alternate pin assignment for T3CKI for 64-pin devices when T3CKMX is cleared.

# PIC18F66K80 FAMILY

**TABLE 11-3: PORTB FUNCTIONS (CONTINUED)**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RB4/AN9/C2INA/ ECCP1/P1A/CTPLS/ KBI0	RB4	0	O	DIG	LATB<4> data output.
		1	I	ST	PORTB<4> data input; weak pull-up when RBPU bit is cleared.
	AN9	1	I	ANA	A/D Input Channel 9 and Comparator C2+ input. Default input configuration on POR; not affected by analog output.
	C2INA <sup>(1)</sup>	2	I	ANA	Comparator 2 Input A.
	ECCP1 <sup>(1)</sup>	0	O	DIG	ECCP1 compare output and ECCP1 PWM output. Takes priority over port data.
		1	I	ST	ECCP1 capture input.
	P1A <sup>(1)</sup>	0	O	DIG	ECCP1 Enhanced PWM output, Channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.
	CTPLS	x	O	DIG	CTMU pulse generator output.
	KBI0	1	I	ST	Interrupt-on-pin change.
RB5/T0CKI/T3CKI/ CCP5/KBI1	RB5	0	O	DIG	LATB<5> data output.
		1	I	ST	PORTB<5> data input; weak pull-up when RBPU bit is cleared.
	T0CKI <sup>(3)</sup>	x	I	ST	Timer0 clock input.
	T3CKI <sup>(4)</sup>	x	I	ST	Timer3 clock input.
	CCP5	0	O	DIG	CCP5 compare/PWM output. Takes priority over port data.
		1	I	ST	CCP5 capture input.
	KBI1	1	I	ST	Interrupt-on-pin change.
RB6/PGC/TX2/CK2/ KBI2	RB6	0	O	DIG	LATB<6> data output.
		1	I	ST	PORTB<6> data input; weak pull-up when RBPU bit is cleared.
	PGC	x	I	ST	Serial execution (ICSP™) clock input for ICSP and ICD operation.
	TX2 <sup>(1)</sup>	0	O	DIG	Asynchronous serial data output (EUSARTx module); takes priority over port data.
		0	O	DIG	Synchronous serial clock output (EUSARTx module); user must configure as an input.
		1	I	ST	Synchronous serial clock input (EUSARTx module); user must configure as an input.
	KBI2	1	I	ST	Interrupt-on-pin change.
RB7/PGD/T3G/RX2/ DT2/KBI3	RB7	0	O	DIG	LATB<7> data output.
		1	I	ST	PORTB<7> data input; weak pull-up when RBPU bit is cleared.
	PGD	x	O	DIG	Serial execution data output for ICSP and ICD operation.
		x	I	ST	Serial execution data input for ICSP and ICD operation.
	T3G	x	I	ST	Timer3 external clock gate input.
	RX2 <sup>(1)</sup>	1	I	ST	Asynchronous serial receive data input (EUSARTx module).
		1	O	DIG	Synchronous serial data output (AUSART module); takes priority over port data.
		1	I	ST	Synchronous serial data input (AUSART module); user must configure as an input.
	KBI3	1	I	ST	Interrupt-on-pin change.

**Legend:** O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

- Note 1:** This pin assignment is only available for 28-pin devices (PIC18F2XK80).  
**2:** This is the default pin assignment for CANRX and CANTX when the CANMX Configuration bit is set.  
**3:** This is the default pin assignment for T0CKI when the T0CKMX Configuration bit is set.  
**4:** This is the default pin assignment for T3CKI for 28, 40 and 44-pin devices. This is the alternate pin assignment for T3CKI for 64-pin devices when T3CKMX is cleared.

# PIC18F66K80 FAMILY

---

TABLE 11-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD
ANCON1	—	ANSEL14	ANSEL13	ANSEL12	ANSEL11	ANSEL10	ANSEL9	ANSEL8

**Legend:** Shaded cells are not used by PORTB.

## 11.4 PORTC, TRISC and LATC Registers

PORTC is an eight-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISC and LATC. Only PORTC pins, RC2 through RC7, are digital only pins.

PORTC is multiplexed with CCP, MSSP and EUSARTx peripheral functions ([Table 11-5](#)). The pins have Schmitt Trigger input buffers. The pins for CCP, SPI and EUSARTx are also configurable for open-drain output whenever these functions are active. Open-drain configuration is selected by setting the SSPOD, CCPxOD and U1OD control bits in the ODCON register.

RC1 is configurable for open-drain output when CCP2 is active on this pin. Open-drain configuration is selected by setting the CCP2OD control bit (ODCON<3>).

When enabling peripheral functions, use care in defining TRIS bits for each PORTC pin. Some peripherals can override the TRIS bit to make a pin an output or input. Consult the corresponding peripheral section for the correct TRIS bit settings.

**Note:** These pins are configured as digital inputs on any device Reset.

The contents of the TRISC register are affected by peripheral overrides. Reading TRISC always returns the current contents, even though a peripheral device may be overriding one or more of the pins.

### EXAMPLE 11-3: INITIALIZING PORTC

```
CLRF    PORTC    ; Initialize PORTC by
                ; clearing output
                ; data latches
CLRF    LATC     ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISC    ; Set RC<3:0> as inputs
                ; RC<5:4> as outputs
                ; RC<7:6> as inputs
```

# PIC18F66K80 FAMILY

---

**TABLE 11-5: PORTC FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RC0/SOSCO/ SCLKI	RC0	0	O	DIG	LATC<0> data output.
		1	I	ST	PORTC<0> data input.
	SOSCO	1	I	ST	SOSC oscillator output.
	SCLKI	1	I	ST	Digital clock input; enabled when SOSC oscillator is disabled.
RC1/SOSCI	RC1	0	O	DIG	LATC<1> data output.
		1	I	ST	PORTC<1> data input.
	SOSCI	x	I	ANA	SOSC oscillator input.
RC2/T1G/ CCP2	RC2	0	O	DIG	LATC<2> data output.
		1	I	ST	PORTC<2> data input.
	T1G	x	I	ST	Timer1 external clock gate input.
	CCP2	0	O	DIG	CCP2 compare/PWM output; takes priority over port data.
		1	I	ST	CCP2 capture input.
RC3/REFO/ SCL/SCK	RC3	0	O	DIG	LATC<3> data output.
		1	I	ST	PORTC<3> data input.
	REFO	x	O	DIG	Reference output clock.
	SCL	0	O	DIG	I <sup>2</sup> C™ clock output (MSSP module); takes priority over port data.
		1	I	I <sup>2</sup> C	I <sup>2</sup> C clock input (MSSP module); input type depends on module setting.
	SCK	0	O	DIG	SPI clock output (MSSP module); takes priority over port data.
		1	I	ST	SPI clock input (MSSP module).
RC4/SDA/SDI	RC4	0	O	DIG	LATC<4> data output.
		1	I	ST	PORTC<4> data input.
	SDA	1	O	DIG	I <sup>2</sup> C data output (MSSP module); takes priority over port data.
		1	I	I <sup>2</sup> C	I <sup>2</sup> C data input (MSSP module); input type depends on module setting.
	SDI	1	I	ST	SPI data input (MSSP module).
RC5/SDO	RC5	0	O	DIG	LATC<5> data output.
		1	I	ST	PORTC<5> data input.
	SDO	0	O	DIG	SPI data output (MSSP module).
RC6/CANTX/ TX1/CK1/ CCP3	RC6	0	O	DIG	LATC<6> data output.
		1	I	ST	PORTC<6> data input.
	CANTX <sup>(2)</sup>	0	O	DIG	CAN bus TX.
	TX1 <sup>(1)</sup>	0	O	DIG	Asynchronous serial data output (EUSARTx module); takes priority over port data.
	CK1 <sup>(1)</sup>	0	O	DIG	Synchronous serial clock output (EUSARTx module); user must configure as an input.
		1	I	ST	Synchronous serial clock input (EUSARTx module); user must configure as an input.
	CCP3	0	O	DIG	CCP3 compare/PWM output. Takes priority over port data.
		1	I	ST	CCP3 capture input.

**Legend:** O = Output; I = Input; I<sup>2</sup>C = I<sup>2</sup>C/SMBus; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input; x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** The pin assignment for 28, 40 and 44-pin devices (PIC18F2XK80 and PIC18F4XK80).

**2:** The alternate pin assignment for CANRX and CANTX on 28, 40 and 44-pin devices (PIC18F4XK80) when the CANMX Configuration bit is set.

# PIC18F66K80 FAMILY

---

**TABLE 11-5: PORTC FUNCTIONS (CONTINUED)**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RC7/CANRX/ RX1/DT1/ CCP4	RC7	0	O	DIG	LATC<7> data output.
		1	I	ST	PORTC<7> data input.
	CANRX <sup>(2)</sup>	1	I	ST	CAN bus RX.
	RX1 <sup>(1)</sup>	1	I	ST	Asynchronous serial receive data input (EUSARTx module).
	DT1 <sup>(1)</sup>	1	O	DIG	Synchronous serial data output (EUSARTx module); takes priority over port data.
		1	I	ST	Synchronous serial data input (EUSARTx module); user must configure as an input.
	CCP4	0	O	DIG	CCP4 compare/PWM output; takes priority over port data.
		1	I	ST	CCP4 capture input.

**Legend:** O = Output; I = Input; I<sup>2</sup>C = I<sup>2</sup>C/SMBus; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input; x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** The pin assignment for 28, 40 and 44-pin devices (PIC18F2XK80 and PIC18F4XK80).

**2:** The alternate pin assignment for CANRX and CANTX on 28, 40 and 44-pin devices (PIC18F4XK80) when the CANMX Configuration bit is set.

**TABLE 11-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
LATC	LATC7	LATBC6	LATC5	LATCB4	LATC3	LATC2	LATC1	LATC0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

**Legend:** Shaded cells are not used by PORTC.

# PIC18F66K80 FAMILY

## 11.5 PORTD, TRISD and LATD Registers

PORTD is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISD and LATD.

**Note:** PORTD is unavailable on 28-pin devices.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**Note:** These pins are configured as digital inputs on any device Reset.

Each of the PORTD pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by setting bit, RDPU (PADC<sub>1</sub>CFG1<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on all device Resets.

PORTD can also be configured as an 8-bit wide micro-processor port (Parallel Slave Port) by setting control bit, PSPMODE (PSPCON<4>). In this mode, the input buffers are ST. For additional information, see [Section 11.9 “Parallel Slave Port”](#).

RD3 has a CTMU functionality.

### EXAMPLE 11-4: INITIALIZING PORTD

```
CLRF    PORTD    ; Initialize PORTD by  
              ; clearing output  
              ; data latches  
CLRF    LATD     ; Alternate method  
              ; to clear output  
              ; data latches  
MOVLW   0CFh    ; Value used to  
              ; initialize data  
              ; direction  
MOVWF   TRISD    ; Set RD<3:0> as inputs  
              ; RD<5:4> as outputs  
              ; RD<7:6> as inputs
```

# PIC18F66K80 FAMILY

**TABLE 11-7: PORTD FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RD0/C1INA/ PSP0	RD0	0	O	DIG	LATD<0> data output.
		1	I	ST	PORTD<0> data input.
	C1INA	1	I	ANA	Comparator 1 Input A.
	PSP0	x	I/O	ST	Parallel Slave Port data.
RD1/C1INB/ PSP1	RD1 <sup>(1)</sup>	0	O	DIG	LATD<1> data output.
		1	I	ST	PORTD<1> data input.
	C1INB <sup>(1)</sup>	1	I	ANA	Comparator 1 Input B.
	PSP1 <sup>(1)</sup>	x	I/O	ST	Parallel Slave Port data.
RD2/C2INA/ PSP2	RD2	0	O	DIG	LATD<2> data output.
		1	I	ST	PORTD<2> data input.
	C2INA	1	I	ANA	Comparator 2 Input A.
	PSP2	x	I/O	ST	Parallel Slave Port data.
RD3/C2INB/ CTMUI/PSP3	RD3	0	O	DIG	LATD<3> data output.
		1	I	ST	PORTD<3> data input.
	C2INB	1	I	ANA	Comparator 2 Input B.
	CTMUI	x	I	—	CTMU pulse generator charger for the C2INB comparator input.
	PSP3	x	I/O	ST	Parallel Slave Port data.
RD4/ECCP1/ P1A/PSP4	RD4	0	O	DIG	LATD<4> data output.
		1	I	ST	PORTD<4> data input.
	ECCP1	0	O	DIG	ECCP1 compare output and ECCP1 PWM output; takes priority over port data.
		1	I	ST	ECCP1 capture input.
	P1A	0	O	DIG	ECCP1 Enhanced PWM output, Channel A. May be configured for tri-state during Enhanced PWM shutdown events; takes priority over port data.
	PSP4	x	I/O	ST	Parallel Slave Port data.
RD5/P1B/PSP5	RD5	0	O	DIG	LATD<5> data output.
		1	I	ST	PORTD<5> data input.
	P1B	0	O	DIG	ECCP1 Enhanced PWM output, Channel B. May be configured for tri-state during Enhanced PWM shutdown events; takes priority over port data.
	PSP5	x	I/O	ST	Parallel Slave Port data.
RD6/TX2/CK2 P1C/PSP6	RD6	0	O	DIG	LATD<6> data output.
		1	I	ST	PORTD<6> data input.
	TX2 <sup>(1)</sup>	0	O	DIG	Asynchronous serial data output (EUSARTx module); takes priority over port data.
	CK2 <sup>(1)</sup>	0	O	DIG	Synchronous serial clock output (EUSARTx module); user must configure as an input.
		1	I	ST	Synchronous serial clock input (EUSARTx module); user must configure as an input.
	P1C	0	O	DIG	ECCP1 Enhanced PWM output, Channel C. May be configured for tri-state during Enhanced PWM.
	PSP6	x	I/O	ST	Parallel Slave Port data.

**Legend:** O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** This is the pin assignment for 40 and 44-pin devices (PIC18F4XK80).

# PIC18F66K80 FAMILY

---

**TABLE 11-7: PORTD FUNCTIONS (CONTINUED)**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description	
RD7/RX2/DT2/ P1D/PSP7	RD7	0	O	DIG	LATD<7> data output.	
		1	I	ST	PORTD<7> data input.	
	RX2 <sup>(1)</sup>	1	I	ST	Asynchronous serial receive data input (EUSARTx module).	
		1	O	DIG	Synchronous serial data output (EUSARTx module); takes priority over port data.	
	DT2 <sup>(1)</sup>		I	ST	Synchronous serial data input (EUSARTx module); user must configure as an input.	
			O	DIG	ECCP1 Enhanced PWM output, Channel D. May be configured for tri-state during Enhanced PWM.	
	PSP7	x	I/O	ST	Parallel Slave Port data.	

**Legend:** O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input;

x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** This is the pin assignment for 40 and 44-pin devices (PIC18F4XK80).

**TABLE 11-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
LATD	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
PADCFG1	RDPU <sup>(1)</sup>	REPU <sup>(1)</sup>	RFP <sup>(2)</sup>	RGP <sup>(2)</sup>	—	—	—	CTMUDS
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD
ANCON1	—	ANSEL14	ANSEL13	ANSEL12	ANSEL11	ANSEL10	ANSEL9	ANSEL8

**Legend:** Shaded cells are not used by PORTD.

**Note 1:** These bits are unimplemented on 28-pin devices, read as '0'.

**2:** These bits are unimplemented on 28/40/44-pin devices, read as '0'.

## 11.6 PORTE, TRISE and LATE Registers

PORTE is a seven-bit-wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISE and LATE.

**Note:** PORTE is unavailable on 28-pin devices.

All pins on PORTE are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**Note:** These pins are configured as digital inputs on any device Reset.

Each of the PORTE pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by clearing bit, REPU (PADC<sub>1</sub>CFG1<6>). The

weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

PORTE is also multiplexed with the Parallel Slave Port address lines. RE1 and RE0 are multiplexed with the Parallel Slave Port (PSP) control signals, WR and RD.

### EXAMPLE 11-5: INITIALIZING PORTE

```

CLRF  PORTE    ; Initialize PORTE by
                  ; clearing output
                  ; data latches
CLRF  LATE     ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW  03h    ; Value used to
                  ; initialize data
                  ; direction
MOVWF  TRISE   ; Set RE<1:0> as inputs
                  ; RE<7:2> as outputs

```

TABLE 11-9: PORTE FUNCTIONS

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RE0/AN5/RD	RE0	0	O	DIG	LATE<0> data output.
		1	I	ST	PORTE<0> data input.
	AN5	1	I	ANA	A/D Input Channel 5. Default input configuration on POR; does not affect digital output.
		x	O	DIG	Parallel Slave Port read strobe pin.
		x	I	ST	Parallel Slave Port read pin.
RE1/AN6/C1OUT/WR	RE1	0	O	DIG	LATE<1> data output.
		1	I	ST	PORTE<1> data input.
	AN6	1	I	ANA	A/D Input Channel 6. Default input configuration on POR; does not affect digital output.
		0	O	DIG	Comparator 1 output; takes priority over port data.
	C1OUT	x	O	DIG	Parallel Slave Port write strobe pin.
		x	I	ST	Parallel Slave Port write pin.
RE2/AN7/C2OUT/CS	RE2	0	O	DIG	LATE<2> data output.
		1	I	ST	PORTE<2> data input.
	AN7	1	I	ANA	A/D Input Channel 7. Default input configuration on POR; does not affect digital output.
		0	O	DIG	Comparator 2 output; takes priority over port data.
	CS	x	I	ST	Parallel Slave Port chip select.
RE3	RE3	1	I	ST	PORT<3> data input.
RE4/CANRX	RE4 <sup>(1)</sup>	0	O	DIG	LATE<4> data output.
		1	I	ST	PORTE<4> data input.
	CANRX <sup>(1,2)</sup>	1	I	ST	CAN bus RX.

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = CMOS Output, ST = Schmitt Trigger Buffer Input,

x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** These bits are unavailable for 40 and 44-pin devices (PIC18F4XK0).

**2:** This is the alternate pin assignment for CANRX and CANTX on 64-pin devices (PIC18F6XK80) when the CANMX Configuration bit is cleared.

# PIC18F66K80 FAMILY

---

**TABLE 11-9: PORTE FUNCTIONS (CONTINUED)**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RE5/CANTX	RE5 <sup>(1)</sup>	0	O	DIG	LATE<5> data output.
		1	I	ST	PORTE<5> data input.
	CANTX <sup>(1,2)</sup>	0	O	DIG	CAN bus TX.
RE6/RX2/DT2	RE6 <sup>(1)</sup>	0	O	DIG	LATE<6> data output.
		1	I	ST	PORTE<6> data input.
	RX2 <sup>(1)</sup>	1	I	ST	Asynchronous serial receive data input (EUSARTx module).
	DT2 <sup>(1)</sup>	1	O	DIG	Synchronous serial data output (EUSARTx module); takes priority over port data.
		1	I	ST	Synchronous serial data input (EUSARTx module); user must configure as an input.
RE7/TX2/CK2	RE7 <sup>(1)</sup>	0	O	DIG	LATE<7> data output.
		1	I	ST	PORTE<7> data input.
	TX2 <sup>(1)</sup>	0	O	DIG	Asynchronous serial data output (EUSARTx module); takes priority over port data.
	CK2 <sup>(1)</sup>	0	O	DIG	Synchronous serial clock output (EUSARTx module); user must configure as an input.
		1	I	ST	Synchronous serial clock input (EUSARTx module); user must configure as an input.

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = CMOS Output, ST = Schmitt Trigger Buffer Input,  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** These bits are unavailable for 40 and 44-pin devices (PIC18F4XK0).

**2:** This is the alternate pin assignment for CANRX and CANTX on 64-pin devices (PIC18F6XK80) when the CANMX Configuration bit is cleared.

**TABLE 11-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTE	RE7 <sup>(1)</sup>	RE6 <sup>(1)</sup>	RE5 <sup>(1)</sup>	RE4 <sup>(1)</sup>	RE3	RE2	RE1	RE0
LATE	LATE7	LATE6	LATE5	LATE4	—	LATE2	LATE1	LATE0
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	—	TRISE2	TRISE1	TRISE0
PADCFG1	RDPU	REPU	RFP <sup>(1)</sup>	RGPU <sup>(1)</sup>	—	—	—	CTMUDS
ANCON0	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0

**Legend:** Shaded cells are not used by PORTE.

**Note 1:** These bits are unimplemented on 44-pin devices, read as '0'.

## 11.7 PORTF, LATF and TRISF Registers

PORTF is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISF and LATF. All pins on PORTF are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**Note:** PORTF is only available on 64-pin devices.

Each of the PORTF pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is done by clearing bit, RPPU (PADC<sub>1<5></sub>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

**Note:** On device Resets, pins, RF<7:1>, are configured as analog inputs and are read as '0'.

### EXAMPLE 11-6: INITIALIZING PORTF

```
CLRF    PORTF    ; Initialize PORTF by
                ; clearing output
                ; data latches
CLRF    LATF     ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CEh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISF    ; Set RF3:RF1 as inputs
                ; RF5:RF4 as outputs
                ; RF7:RF6 as inputs
```

TABLE 11-11: PORTF FUNCTIONS

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RF0/MDMIN	RF0	0	O	DIG	LATF<0> data output.
		1	I	ST	PORTF<0> data input.
	MDMIN	1	I	ST	Modulator source input.
RF1	RF1	0	O	DIG	LATF<1> data output.
		1	I	ST	PORTF<1> data input.
RF2/MDCIN1	RF2	0	O	DIG	LATF<2> data output.
		1	I	ST	PORTF<2> data input.
	MDCIN1	1	I	ST	Modulator Carrier Input 1.
RF3	RF3	0	O	DIG	LATF<3> data output.
		1	I	ST	PORTF<3> data input.
RF4/MDCIN2	RF4	0	O	DIG	LATF<4> data output.
		1	I	ST	PORTF<4> data input.
	MDCIN2	1	I	ST	Modulator Carrier Input 2.
RF5	RF5	0	O	DIG	LATF<5> data output.
		1	I	ST	PORTF<5> data input.
RF6/MDOUT	RF6	0	O	DIG	LATF<6> data output.
		1	I	ST	PORTF<6> data input.
	MDOUT	0	O	DIG	Modulator output.
RF7	RF7	0	O	DIG	LATF<7> data output.
		1	I	ST	PORTF<7> data input.

**Legend:** O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

TABLE 11-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0
PADCFG1	RDP <u>U</u>	REPU	RFFPU <sup>(1)</sup>	RGPU <sup>(1)</sup>	—	—	—	CTMUDS

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

**Note 1:** These bits are unimplemented on 28-pin devices, read as '0'.

# PIC18F66K80 FAMILY

## 11.8 PORTG, TRISG and LATG Registers

PORTG is a 5-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISG and LATG.

**Note:** PORTG is only available on 64-pin devices.

PORTG is multiplexed with EUSARTx and CCP, ECCP, Analog, Comparator and Timer input functions (Table 11-13). When operating as I/O, all PORTG pins have Schmitt Trigger input buffers. The open-drain functionality for the EUARTx can be configured using ODCON.

Each of the PORTG pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by clearing bit, RGPU (PADC<sub>1</sub>CFG1<4>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTG pin. Some peripherals override the TRIS bit to make a pin an out-

put, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings. The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register without concern due to peripheral overrides.

### EXAMPLE 11-7: INITIALIZING PORTG

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

TABLE 11-13: PORTG FUNCTIONS

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RG0/RX1/DT1	RG0	0	O	DIG	LATG<0> data output.
		1	I	ST	PORTG<0> data input.
	RX1	1	I	ST	Asynchronous serial receive data input (EUSARTx module).
	DT1	0	O	DIG	Synchronous serial data output (EUSARTx module); takes priority over port data.
		1	I	ST	Synchronous serial data input (EUSARTx module); user must configure as an input.
RG1/CANTX	RG1	0	O	DIG	LATG<1> data output.
		1	I	ST	PORTG<1> data input.
	CANTX	0	O	DIG	CAN bus TX.
RG2/T3CKI	RG2	0	O	DIG	LATG<2> data output.
		1	I	ST	PORTG<2> data input.
	T3CKI <sup>(2)</sup>	x	I	ST	Timer3 clock input.
RG3/TX1/CK1	RG3	0	O	DIG	LATG<3> data output.
		1	I	ST	PORTG<3> data input.
	TX1	0	O	DIG	Asynchronous serial data output (EUSARTx module); takes priority over port data.
	CK1	0	O	DIG	Synchronous serial clock output (EUSARTx module); user must configure as an input.
		1	I	ST	Synchronous serial clock input (EUSARTx module); user must configure as an input.
RG4/T0CKI	RG4	0	O	DIG	LATG<4> data output.
		1	I	ST	PORTG<4> data input.
	T0CKI <sup>(1)</sup>	x	I	ST	Timer0 clock input.

**Legend:** O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

**Note 1:** This is the alternate pin assignment for T0CKI on 64-pin devices when the TOCKMX Configuration bit is cleared.

**2:** This is the default pin assignment for T3CKI on 64-pin devices when the T3CKMX Configuration bit is set.

# PIC18F66K80 FAMILY

---

TABLE 11-14: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTG	—	—	—	RG4	RG3	RG2	RG1	RG0
TRISG	—	—	—	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0
PADCFG1	RDPU	REPU	RFP <u>U</u> <sup>(1)</sup>	RGPU <sup>(1)</sup>	—	—	—	CTMUDS

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

**Note 1:** These bits are unimplemented on 28-pin devices; read as '0'.

# PIC18F66K80 FAMILY

## 11.9 Parallel Slave Port

PORTD can function as an 8-bit-wide Parallel Slave Port (PSP), or microprocessor port, when control bit, PSPMODE (PSPCON<4>), is set. The port is asynchronously readable and writable by the external world through the RD control input pin (RE0/AN5/RD) and WR control input pin (RE1/AN6/C1OUT/WR).

**Note:** The Parallel Slave Port is available only on 40/44-pin and 64-pin devices.

The PSP can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an eight-bit latch.

Setting bit, PSPMODE, enables port pin, RE0/AN5/RD, to be the RD input, RE1/AN6/C1OUT/WR to be the WR input and RE2/AN7/C2OUT/CS to be the CS (Chip Select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (= 111).

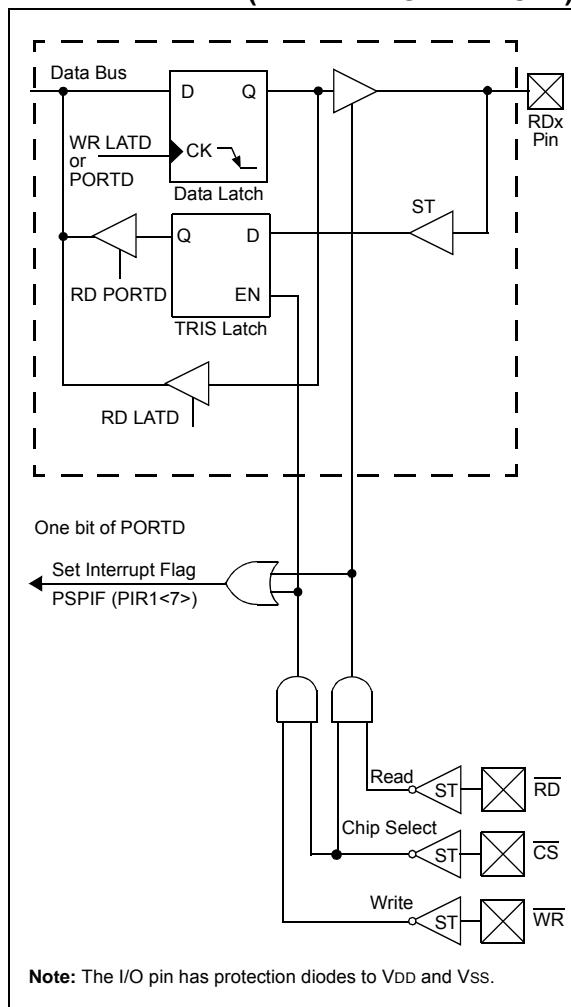
A write to the PSP occurs when both the CS and WR lines are first detected low and ends when either are detected high. The PSPIF and IBF flag bits (PIR1<7> and PSPCON<7>, respectively) are set when the write ends.

A read from the PSP occurs when both the CS and RD lines are first detected low. The data in PORTD is read out and the OBF bit (PSPCON<6>) is set. If the user writes new data to PORTD to set OBF, the data is immediately read out, but the OBF bit is not set.

When either the CS or RD line is detected high, the PORTD pins return to the input state and the PSPIF bit is set. User applications should wait for PSPIF to be set before servicing the PSP. When this happens, the IBF and OBF bits can be polled and the appropriate action taken.

The timing for the control signals in Write and Read modes is shown in [Figure 11-4](#) and [Figure 11-5](#), respectively.

**FIGURE 11-3: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)**



## REGISTER 11-5: PSPCON: PARALLEL SLAVE PORT CONTROL REGISTER

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

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

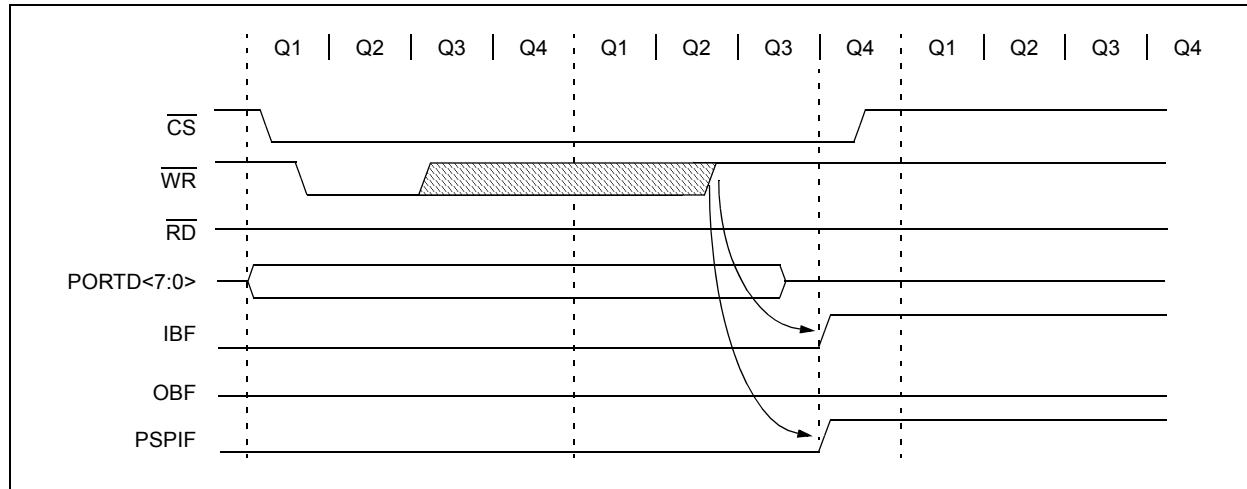
'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

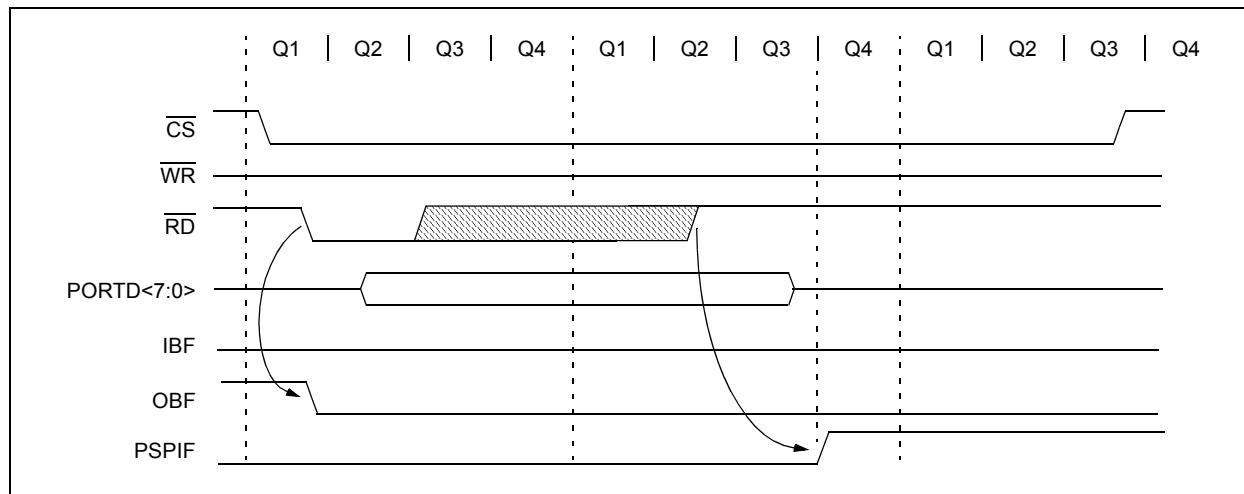
- |         |  |
|---------|--|
| bit 7   | <b>IBF:</b> Input Buffer Full Status bit<br>1 = A word has been received and is waiting to be read by the CPU<br>0 = No word has been received                                 |
| bit 6   | <b>OBF:</b> Output Buffer Full Status bit<br>1 = The output buffer still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

**FIGURE 11-4: PARALLEL SLAVE PORT WRITE WAVEFORMS**



# PIC18F66K80 FAMILY

**FIGURE 11-5: PARALLEL SLAVE PORT READ WAVEFORMS**



**TABLE 11-15: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
LATD	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
PORTE	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0
LATE	LATE7	LATE6	LATE5	LATE4	—	LATE2	LATE1	LATE0
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	—	TRISE2	TRISE1	TRISE0
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.

## 12.0 DATA SIGNAL MODULATOR

**Note:** The Data Signal Modulator is only available on 64-pin devices (PIC18F6XK80).

The Data Signal Modulator (DSM) is a peripheral which allows the user to mix a data stream, also known as a modulator signal, with a carrier signal to produce a modulated output.

Both the carrier and the modulator signals are supplied to the DSM module, either internally from the output of a peripheral, or externally through an input pin.

The modulated output signal is generated by performing a logical “AND” operation of both the carrier and modulator signals and then it is provided to the MDOUT pin.

The carrier signal is comprised of two distinct and separate signals: a carrier high (CARH) signal and a carrier low (CARL) signal. During the time in which the modulator (MOD) signal is in a logic high state, the DSM mixes the carrier high signal with the modulator signal. When the modulator signal is in a logic low state, the DSM mixes the carrier low signal with the modulator signal.

Using this method, the DSM can generate the following types of key modulation schemes:

- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

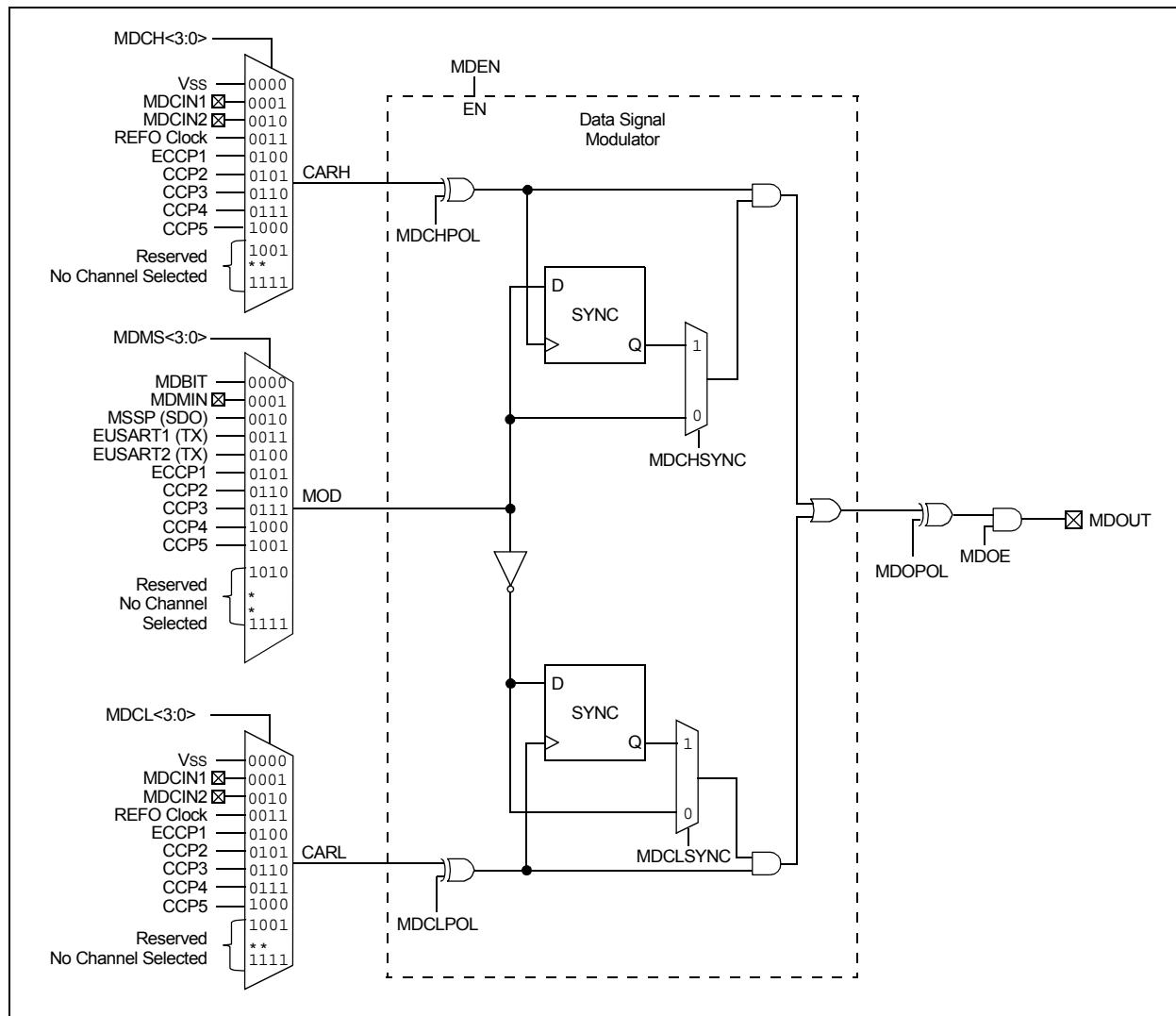
Additionally, the following features are provided within the DSM module:

- Carrier Synchronization
- Carrier Source Polarity Select
- Carrier Source Pin Disable
- Programmable Modulator Data
- Modulator Source Pin Disable
- Modulated Output Polarity Select
- Slew Rate Control

[Figure 12-1](#) shows a simplified block diagram of the Data Signal Modulator peripheral.

# PIC18F66K80 FAMILY

**FIGURE 12-1: SIMPLIFIED BLOCK DIAGRAM OF THE DATA SIGNAL MODULATOR**



## 12.1 DSM Operation

The DSM module can be enabled by setting the MDEN bit in the MDCON register. Clearing the MDEN bit in the MDCON register, disables the DSM module by automatically switching the carrier high and carrier low signals to the Vss signal source. The modulator signal source is also switched to the MDBIT in the MDCON register. This not only assures that the DSM module is inactive, but that it is also consuming the least amount of current.

The values used to select the carrier high, carrier low and modulator sources held by the Modulation Source, Modulation High Carrier and Modulation Low Carrier Control registers are not affected when the MDEN bit is cleared, and the DSM module is disabled. The values inside these registers remain unchanged while the DSM is inactive. The sources for the carrier high, carrier low and modulator signals will once again be selected when the MDEN bit is set and the DSM module is again enabled and active.

The modulated output signal can be disabled without shutting down the DSM module. The DSM module will remain active and continue to mix signals, but the output value will not be sent to the MDOUT pin. During the time that the output is disabled, the MDOUT pin will remain low. The modulated output can be disabled by clearing the MDOE bit in the MDCON register.

## 12.2 Modulator Signal Sources

The modulator signal can be supplied from the following sources:

- ECCP1 Signal
- CCP2 Signal
- CCP3 Signal
- CCP4 Signal
- CCP5 Signal
- MSSP SDO Signal (SPI mode only)
- EUSART1 TX1 Signal
- EUSART2 TX2 Signal
- External Signal on MDMIN Pin (RF0/MDMIN)
- MDBIT bit in the MDCON Register

The modulator signal is selected by configuring the MDSRC<3:0> bits in the MDSRC register.

## 12.3 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from the following sources:

- CCP1 Signal
- CCP2 Signal
- CCP3 Signal
- CCP4 Signal
- Reference Clock Module Signal
- External Signal on MDCIN1 Pin (RF2/MDCIN1)
- External Signal on MDCIN2 Pin (RF4/MDCIN2)
- Vss

The carrier high signal is selected by configuring the MDCH<3:0> bits in the MDCARH register. The carrier low signal is selected by configuring the MDCL<3:0> bits in the MDCARL register.

## 12.4 Carrier Synchronization

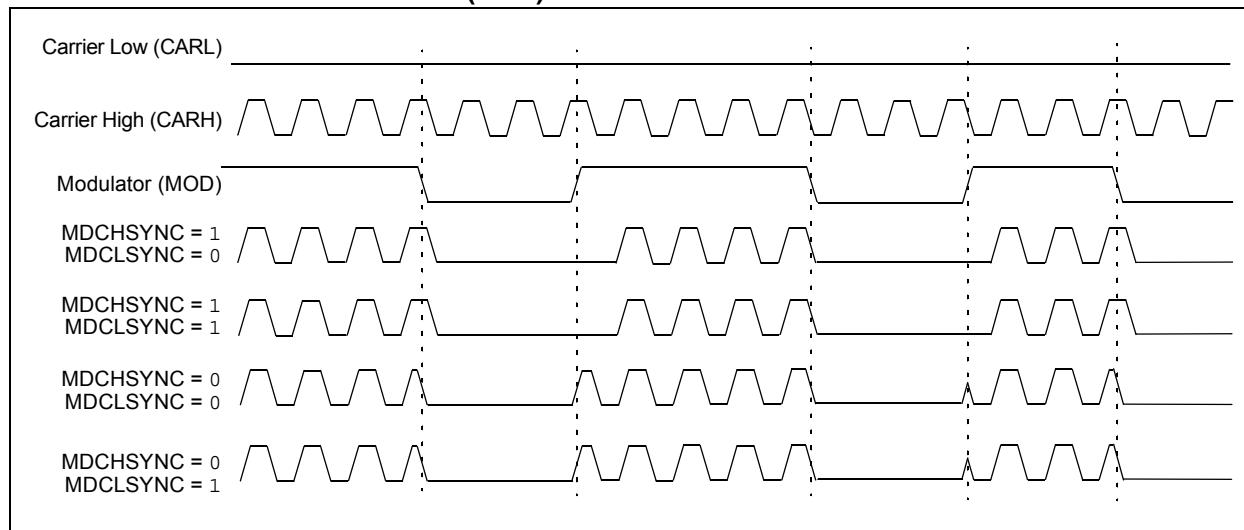
During the time when the DSM switches between carrier high and carrier low signal sources, the carrier data in the modulated output signal can become truncated. To prevent this, the carrier signal can be synchronized to the modulator signal. When synchronization is enabled, the carrier pulse that is being mixed at the time of the transition is allowed to transition low before the DSM switches over to the next carrier source.

Synchronization is enabled separately for the carrier high and carrier low signal sources. Synchronization for the carrier high signal can be enabled by setting the MDCHSYNC bit in the MDCARH register. Synchronization for the carrier low signal can be enabled by setting the MDCLSYNC bit in the MDCARL register.

[Figure 12-1](#) through [Figure 12-5](#) show timing diagrams of using various synchronization methods.

# PIC18F66K80 FAMILY

FIGURE 12-2: ON/OFF KEYING (OOK) SYNCHRONIZATION



EXAMPLE 12-1: NO SYNCHRONIZATION (MDCHSYNC = 0, MDCLSYNC = 0)

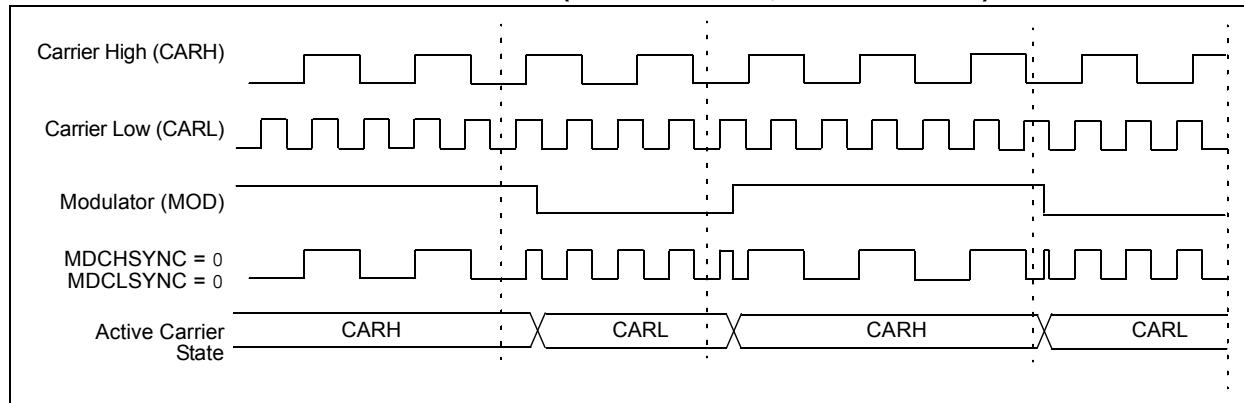
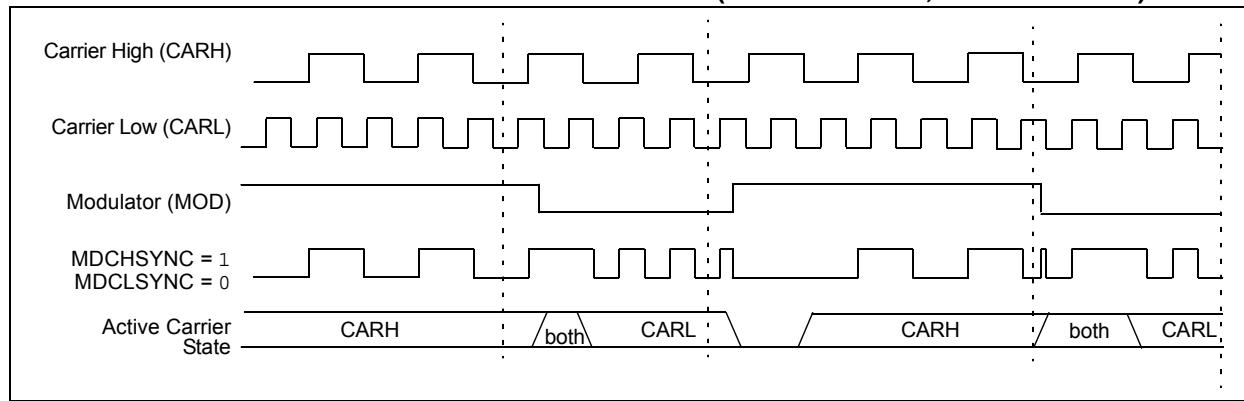
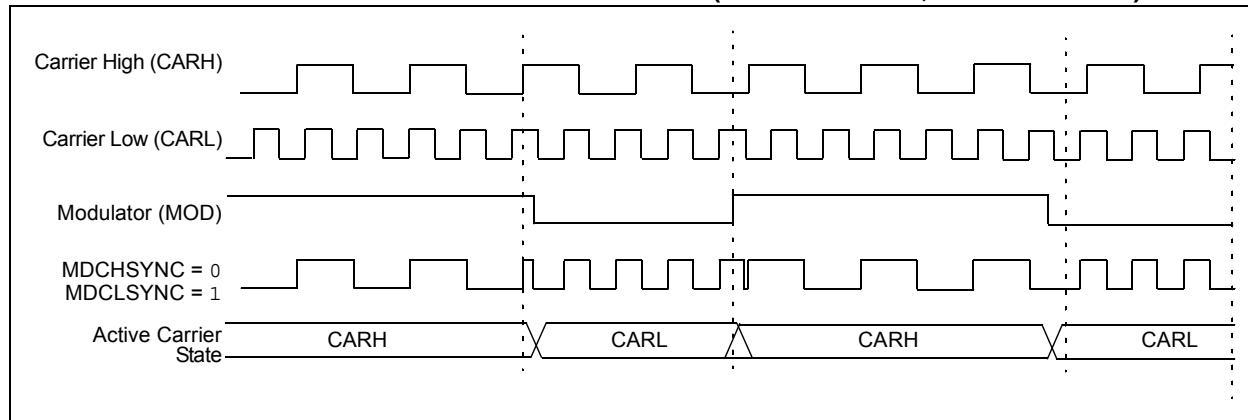


FIGURE 12-3: CARRIER HIGH SYNCHRONIZATION (MDCHSYNC = 1, MDCLSYNC = 0)

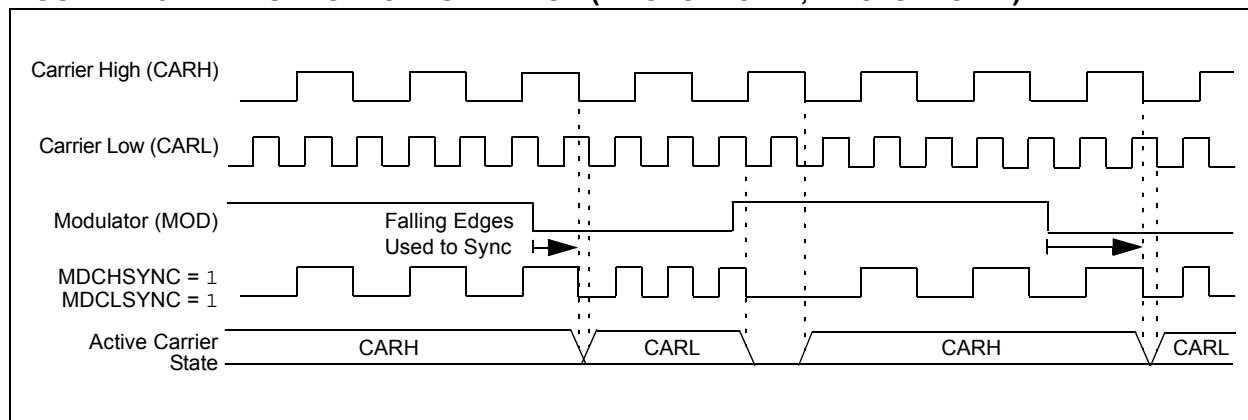


# PIC18F66K80 FAMILY

**FIGURE 12-4: CARRIER LOW SYNCHRONIZATION (MDCHSYNC = 0, MDCLSYNC = 1)**



**FIGURE 12-5: FULL SYNCHRONIZATION (MDCHSYNC = 1, MDCLSYNC = 1)**



# PIC18F66K80 FAMILY

---

## 12.5 Carrier Source Polarity Select

The signal provided from any selected input source for the carrier high and carrier low signals can be inverted. Inverting the signal for the carrier high source is enabled by setting the MDCHPOL bit of the MDCARH register. Inverting the signal for the carrier low source is enabled by setting the MDCLPOL bit of the MDCARL register.

## 12.6 Carrier Source Pin Disable

Some peripherals assert control over their corresponding output pin when they are enabled. For example, when the CCP1 module is enabled, the output of CCP1 is connected to the CCP1 pin.

This default connection to a pin can be disabled by setting the MDCHODIS bit in the MDCARH register for the carrier high source and the MDCLODIS bit in the MDCARL register for the carrier low source.

## 12.7 Programmable Modulator Data

The MDBIT of the MDCON register can be selected as the source for the modulator signal. This gives the user the ability to program the value used for modulation.

## 12.8 Modulator Source Pin Disable

The modulator source default connection to a pin can be disabled by setting the MDSODIS bit in the MDSRC register.

## 12.9 Modulated Output Polarity

The modulated output signal provided on the MDOUT pin can also be inverted. Inverting the modulated output signal is enabled by setting the MDOPOL bit of the MDCON register.

## 12.10 Slew Rate Control

When modulated data streams of 20 MHz or greater are required, the slew rate limitation on the output port pin can be disabled. The slew rate limitation can be removed by clearing the MDSLRL bit in the MDCON register.

## 12.11 Operation In Sleep Mode

The DSM module is not affected by Sleep mode. The DSM can still operate during Sleep if the Carrier and Modulator input sources are also still operable during Sleep.

## 12.12 Effects of a Reset

Upon any device Reset, the Data Signal Modulator module is disabled. The user's firmware is responsible for initializing the module before enabling the output. The registers are reset to their default values.

# PIC18F66K80 FAMILY

## REGISTER 12-1: MDCON: MODULATION CONTROL REGISTER

R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	U-0	U-0	R/W-0
MDEN	MDOE	MDSLR	MDOPOL	MDO	—	—	MDBIT
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           **MDEN:** Modulator Module Enable bit  
1 = Modulator module is enabled and mixing input signals  
0 = Modulator module is disabled and has no output
- bit 6           **MDOE:** Modulator Module Pin Output Enable bit  
1 = Modulator pin output is enabled  
0 = Modulator pin output is disabled
- bit 5           **MDSLR:** MDOUT Pin Slew Rate Limiting bit  
1 = MDOUT pin slew rate limiting is enabled  
0 = MDOUT pin slew rate limiting is disabled
- bit 4           **MDOPOL:** Modulator Output Polarity Select bit  
1 = Modulator output signal is inverted  
0 = Modulator output signal is not inverted
- bit 3           **MDO:** Modulator Output bit  
Displays the current output value of the modulator module.<sup>(2)</sup>
- bit 2-1       **Unimplemented:** Read as '0'
- bit 0           **MDBIT:** Modulator Source Input bit  
Allows software to manually set modulation source input to module.<sup>(1)</sup>

- Note 1:** The MDBIT must be selected as the modulation source in the MDSRC register for this operation.
- 2:** The modulated output frequency can be greater and asynchronous from the clock that updates this register bit. The bit value may not be valid for higher speed modulator or carrier signals.

# PIC18F66K80 FAMILY

## REGISTER 12-2: MDSRC: MODULATION SOURCE CONTROL REGISTER

R/W-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
MDSODIS	—	—	—	MDSRC3	MDSRC2	MDSRC1	MDSRC0
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      **MDSODIS:** Modulation Source Output Disable bit

1 = Output signal driving the peripheral output pin (selected by MDMS<3:0>) is disabled

0 = Output signal driving the peripheral output pin (selected by MDMS<3:0>) is enabled

bit 6-4      **Unimplemented:** Read as '0'

bit 3-0      **MDSRC<3:0>** Modulation Source Selection bits

1111-1010 = Reserved; no channel connected

1001 = CCP5 output (PWM Output mode only)

1000 = CCP4 output (PWM Output mode only)

0111 = CCP3 output (PWM Output mode only)

0110 = CCP2 output (PWM Output mode only)

0101 = ECCP1 output (PWM Output mode only)

0100 = EUSART2 TX output

0011 = EUSART1 TX output

0010 = MSSP SDO output

0001 = MDMIN port pin

0000 = MDBIT bit of the MDCON register is the modulation source

# PIC18F66K80 FAMILY

## REGISTER 12-3: MDCARH: MODULATION HIGH CARRIER CONTROL REGISTER

R/W-0	R/W-x	R/W-x	U-0	R/W-x	R/W-x	R/W-x	R/W-x
MDCHODIS	MDCHPOL	MDCHSYNC	—	MDCH3 <sup>(1)</sup>	MDCH2 <sup>(1)</sup>	MDCH1 <sup>(1)</sup>	MDCH0 <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	<b>MDCHODIS:</b> Modulator High Carrier Output Disable bit 1 = Output signal driving the peripheral output pin (selected by MDCH<3:0>) is disabled 0 = Output signal driving the peripheral output pin (selected by MDCH<3:0>) is enabled
bit 6	<b>MDCHPOL:</b> Modulator High Carrier Polarity Select bit 1 = Selected high carrier signal is inverted 0 = Selected high carrier signal is not inverted
bit 5	<b>MDCHSYNC:</b> Modulator High Carrier Synchronization Enable bit 1 = Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier 0 = Modulator output is not synchronized to the high time carrier signal <sup>(1)</sup>
bit 4	<b>Unimplemented:</b> Read as '0'
bit 3-0	<b>MDCH&lt;3:0&gt;</b> Modulator Data High Carrier Selection bits <sup>(1)</sup> 1111-1001 = Reserved 1000 = CCP5 output (PWM Output mode only) 0111 = CCP4 output (PWM Output mode only) 0110 = CCP3 output (PWM Output mode only) 0101 = CCP2 output (PWM Output mode only) 0100 = ECCP1 output (PWM Output mode only) 0011 = Reference clock module signal 0010 = MDCIN2 port pin 0001 = MDCIN1 port pin 0000 = Vss

**Note 1:** Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

# PIC18F66K80 FAMILY

## REGISTER 12-4: MDCARL: MODULATION LOW CARRIER CONTROL REGISTER

R/W-0	R/W-x	R/W-x	U-0	R/W-x	R/W-x	R/W-x	R/W-x
MDCLODIS	MDCLPOL	MDCLSYNC	—	MDCL3 <sup>(1)</sup>	MDCL2 <sup>(1)</sup>	MDCL1 <sup>(1)</sup>	MDCL0 <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      **MDCLODIS:** Modulator Low Carrier Output Disable bit  
 1 = Output signal driving the peripheral output pin (selected by MDCL<3:0> of the MDCARL register) is disabled  
 0 = Output signal driving the peripheral output pin (selected by MDCL<3:0> of the MDCARL register) is enabled
- bit 6      **MDCLPOL:** Modulator Low Carrier Polarity Select bit  
 1 = Selected low carrier signal is inverted  
 0 = Selected low carrier signal is not inverted
- bit 5      **MDCLSYNC:** Modulator Low Carrier Synchronization Enable bit  
 1 = Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier  
 0 = Modulator output is not synchronized to the low time carrier signal<sup>(1)</sup>
- bit 4      **Unimplemented:** Read as '0'
- bit 3-0     **MDCL<3:0>** Modulator Data High Carrier Selection bits<sup>(1)</sup>  
 1111-1001 = Reserved  
 1000 = CCP5 output (PWM Output mode only)  
 0111 = CCP4 output (PWM Output mode only)  
 0110 = CCP3 output (PWM Output mode only)  
 0101 = CCP2 output (PWM Output mode only)  
 0100 = ECCP1 output (PWM Output mode only)  
 0011 = Reference clock module signal  
 0010 = MDCIN2 port pin  
 0001 = MDCIN1 port pin  
 0000 = Vss

**Note 1:** Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

TABLE 12-1: SUMMARY OF REGISTERS ASSOCIATED WITH DATA SIGNAL MODULATOR MODE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MDCARH	MDCHODIS	MDCHPOL	MDCHSYNC	—	MDCH3	MDCH2	MDCH1	MDCH0
MDCARL	MDCLODIS	MDCLPOL	MDCLSYNC	—	MDCL3	MDCL2	MDCL1	MDCL0
MDCON	MDEN	MDOE	MDSLR	MDOPOL	MDO	—	—	MDBIT
MDSRC	MDSODIS	—	—	—	MDSRC3	MDSRC2	MDSRC1	MDSRC0
PMD2	—	—	—	—	MODMD	ECANMD	CMP2MD	CMP1MD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used in the Data Signal Modulator mode.

## 13.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register ([Register 13-1](#)) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

[Figure 13-1](#) provides a simplified block diagram of the Timer0 module in 8-bit mode. [Figure 13-2](#) provides a simplified block diagram of the Timer0 module in 16-bit mode.

### REGISTER 13-1: T0CON: TIMER0 CONTROL REGISTER

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

#### 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	<b>TMR0ON:</b> Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0
bit 6	<b>T08BIT:</b> Timer0 8-Bit/16-Bit Control bit 1 = Timer0 is configured as an 8-bit timer/counter 0 = Timer0 is configured as a 16-bit timer/counter
bit 5	<b>T0CS:</b> Timer0 Clock Source Select bit 1 = Transitions on T0CKI pin 0 = Internal instruction cycle clock (CLKO)
bit 4	<b>T0SE:</b> Timer0 Source Edge Select bit 1 = Increments on high-to-low transition on T0CKI pin 0 = Increments on low-to-high transition on T0CKI pin
bit 3	<b>PSA:</b> 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	<b>T0PS&lt;2:0:</b> 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

# PIC18F66K80 FAMILY

## 13.1 Timer0 Operation

Timer0 can operate as either a timer or a counter. The mode is selected with the T0CS bit (T0CON<5>). In Timer mode (T0CS = 0), the module increments on every clock by default unless a different prescaler value is selected (see [Section 13.3 “Prescaler”](#)). If the TMRO register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMRO register.

The Counter mode is selected by setting the T0CS bit (= 1). In this mode, Timer0 increments either on every rising edge or falling edge of the T0CKI pin. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the

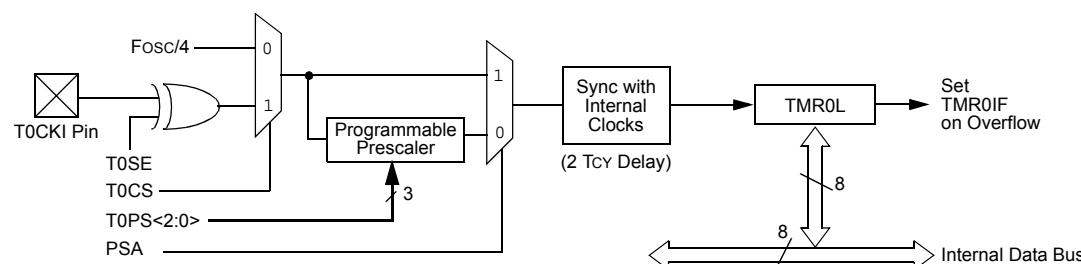
internal phase clock (Tosc). There is a delay between synchronization and the onset of incrementing the timer/counter.

## 13.2 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode. It is actually a buffered version of the real high byte of Timer0, which is not directly readable nor writable. (See [Figure 13-2](#).) TMR0H is updated with the contents of the high byte of Timer0 during a read of TMROL. 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.

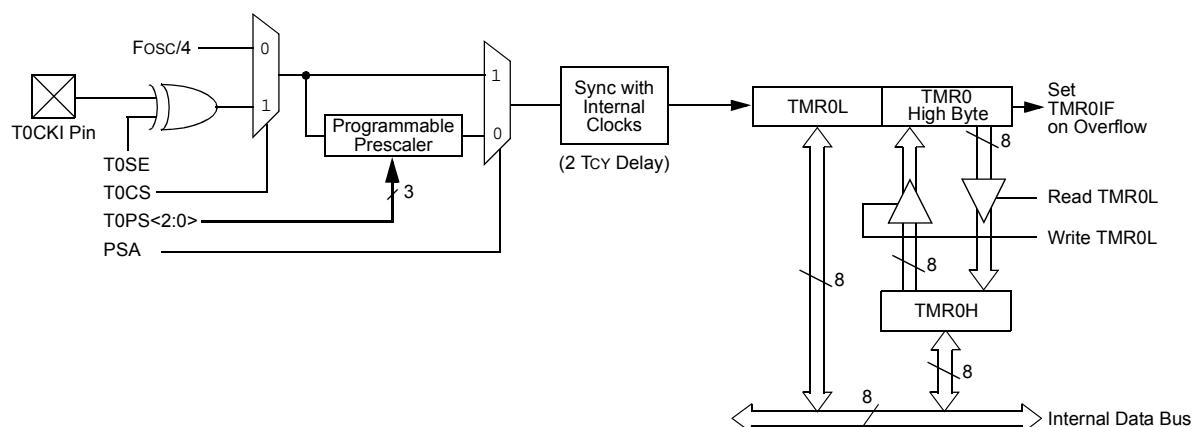
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMROL. This allows all 16 bits of Timer0 to be updated at once.

**FIGURE 13-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)**



**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

**FIGURE 13-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)**



**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

### 13.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable. Its value is set by the PSA and T0PS<2:0> bits (T0CON<3:0>), which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When it is assigned, prescale values from 1:2 through 1:256, in power-of-two increments, are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (for example, CLRF TMR0, MOVWF TMR0, BSF TMR0) 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.

### 13.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed "on-the-fly" during program execution.

### 13.4 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMR0IE bit (INTCON<5>). Before re-enabling the interrupt, the TMR0IF bit must be cleared in software by the Interrupt Service Routine (ISR).

Since Timer0 is shutdown in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

**TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER0**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TMR0L	Timer0 Register Low Byte							
TMR0H	Timer0 Register High Byte							
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by Timer0.

# **PIC18F66K80 FAMILY**

---

---

## **NOTES:**

## 14.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or SOSC oscillator internal options
- Interrupt-on-overflow
- Reset on ECCP Special Event Trigger
- Timer with gated control

[Figure 14-1](#) displays a simplified block diagram of the Timer1 module.

### REGISTER 14-1: T1CON: TIMER1 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
TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	SOSCEN	T1SYNC	RD16	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 7-6	<b>TMR1CS&lt;1:0&gt;</b> : Timer1 Clock Source Select bits 10 = Timer1 clock source is either from pin or oscillator, depending on the SOSCEN bit: <u>SOSCEN = 0:</u> External clock is from the T1CKI pin (on the rising edge). <u>SOSCEN = 1:</u> Depending on the SOSCSELx Configuration bit, the clock source is either a crystal oscillator on SOSC/SOSCO or an internal digital clock from the SCLKI pin. 01 = Timer1 clock source is the system clock (Fosc) <sup>(1)</sup> 00 = Timer1 clock source is the instruction clock (Fosc/4)
bit 5-4	<b>T1CKPS&lt;1:0&gt;</b> : 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	<b>SOSCEN</b> : SOSC Oscillator Enable bit 1 = SOSC is enabled and available for Timer1 0 = SOSC is disabled for Timer1 The oscillator inverter and feedback resistor are turned off to eliminate power drain.
bit 2	<b>T1SYNC</b> : Timer1 External Clock Input Synchronization Select bit <u>TMR1CS&lt;1:0&gt; = 10:</u> 1 = Do not synchronize external clock input 0 = Synchronizes external clock input <u>TMR1CS&lt;1:0&gt; = 0x:</u> This bit is ignored. Timer1 uses the internal clock when TMR1CS<1:0> = 1x.

**Note 1:** The Fosc clock source should not be selected if the timer will be used with the ECCP capture/compare features.

The module derives its clocking source from either the secondary oscillator or from an external digital source. If using the secondary oscillator, there are the additional options for low-power, high-power and external digital clock source.

Timer1 is controlled through the T1CON Control register ([Register 14-1](#)). It also contains the Timer1 Oscillator Enable bit (SOSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

The Fosc clock source should not be used with the ECCP capture/compare features. If the timer will be used with the capture or compare features, always select one of the other timer clocking options.

# PIC18F66K80 FAMILY

---

---

## REGISTER 14-1: T1CON: TIMER1 CONTROL REGISTER (CONTINUED)

bit 1           **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 0           **TMR1ON:** Timer1 On bit

1 = Enables Timer1

0 = Stops Timer1

**Note 1:** The Fosc clock source should not be selected if the timer will be used with the ECCP capture/compare features.

## 14.1 Timer1 Gate Control Register

The Timer1 Gate Control register (T1GCON), displayed in [Register 14-2](#), is used to control the Timer1 gate.

### REGISTER 14-2: T1GCON: TIMER1 GATE CONTROL REGISTER<sup>(1)</sup>

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-x	R/W-0	R/W-0
TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/T1DONE	T1GVAL	T1GSS1	T1GSS0
bit 7							

#### 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	<b>TMR1GE:</b> Timer1 Gate Enable bit  If <u>TMR1ON = 0</u> : This bit is ignored. If <u>TMR1ON = 1</u> : 1 = Timer1 counting is controlled by the Timer1 gate function 0 = Timer1 counts regardless of Timer1 gate function
bit 6	<b>T1GPOL:</b> Timer1 Gate Polarity bit 1 = Timer1 gate is active-high (Timer1 counts when gate is high) 0 = Timer1 gate is active-low (Timer1 counts when gate is low)
bit 5	<b>T1GTM:</b> Timer1 Gate Toggle Mode bit 1 = Timer1 Gate Toggle mode is enabled 0 = Timer1 Gate Toggle mode is disabled and toggle flip-flop is cleared Timer1 gate flip-flop toggles on every rising edge.
bit 4	<b>T1GSPM:</b> Timer1 Gate Single Pulse Mode bit 1 = Timer1 Gate Single Pulse mode is enabled and is controlling Timer1 gate 0 = Timer1 Gate Single Pulse mode is disabled
bit 3	<b>T1GGO/T1DONE:</b> Timer1 Gate Single Pulse Acquisition Status bit 1 = Timer1 gate single pulse acquisition is ready, waiting for an edge 0 = Timer1 gate single pulse acquisition has completed or has not been started This bit is automatically cleared when T1GSPM is cleared.
bit 2	<b>T1GVAL:</b> Timer1 Gate Current State bit Indicates the current state of the Timer1 gate that could be provided to TMR1H:TMR1L; unaffected by Timer1 Gate Enable (TMR1GE) bit.
bit 1-0	<b>T1GSS&lt;1:0&gt;:</b> Timer1 Gate Source Select bits 11 = Comparator 2 output 10 = Comparator 1 output 01 = TMR2 to match PR2 output 00 = Timer1 gate pin

**Note 1:** Programming the T1GCON prior to T1CON is recommended.

# PIC18F66K80 FAMILY

## 14.2 Timer1 Operation

The Timer1 module is an 8 or 16-bit incrementing counter that is accessed through the TMR1H:TMR1L register pair.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter. It increments on every selected edge of the external source.

Timer1 is enabled by configuring the TMR1ON and TMR1GE bits in the T1CON and T1GCON registers, respectively.

When SOSC is selected as Crystal mode (by the SOSCSELx bits), the RC1/SOSCI and RC0/SOSCO/SCLKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

## 14.3 Clock Source Selection

The TMR1CS<1:0> and SOSCEN bits of the T1CON register are used to select the clock source for Timer1. [Table 14-1](#) displays the clock source selections.

### 14.3.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected, the TMR1H:TMR1L register pair will increment on multiples of Fosc as determined by the Timer1 prescaler.

### 14.3.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1 module may work as a timer or a counter.

When enabled to count, Timer1 is incremented on the rising edge of the external clock input, T1CKI. Either of these external clock sources can be synchronized to the microcontroller system clock or they can run asynchronously.

When used as a timer with a clock oscillator, an external, 32.768 kHz crystal can be used in conjunction with the dedicated internal oscillator circuit.

**Note:** In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1 is enabled after POR Reset
- Write to TMR1H or TMR1L
- Timer1 is disabled
- Timer1 is disabled (TMR1ON = 0)

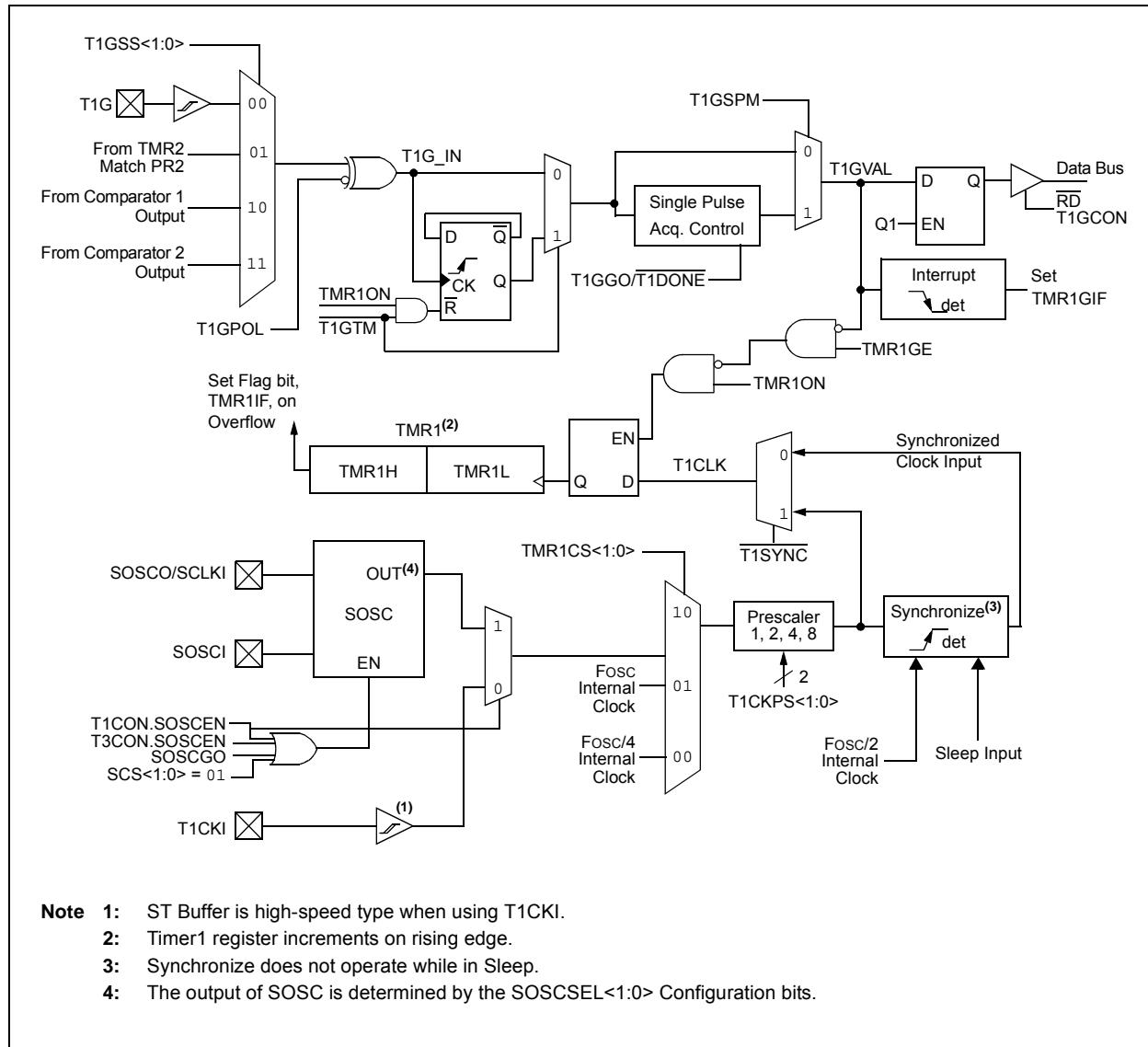
When T1CKI is high, Timer1 is enabled (TMR1ON = 1) when T1CKI is low.

**TABLE 14-1: TIMER1 CLOCK SOURCE SELECTION**

TMR1CS1	TMR1CS0	SOSCEN	Clock Source
0	1	x	Clock Source (Fosc)
0	0	x	Instruction Clock (Fosc/4)
1	0	0	External Clock on T1CKI Pin
1	0	1	Oscillator Circuit on SOSCI/SOSCO Pins

# PIC18F66K80 FAMILY

**FIGURE 14-1: TIMER1 BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

## 14.4 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes. When the RD16 control bit (T1CON<1>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L loads 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, has become invalid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. The 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 at once to both the high and low bytes of Timer1.

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.

## 14.5 SOSC Oscillator

An on-chip crystal oscillator circuit is incorporated between pins, SOSCI (input) and SOSCO (amplifier output). It can be enabled one of these ways:

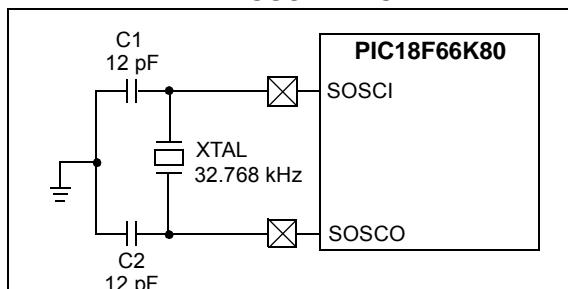
- Setting the SOSCEN bit in either the T1CON or T3CON register (TxCON<3>)
- Setting the SOSCGO bit in the OSCCON2 register (OSCCON2<3>)
- Setting the SCSx bits to secondary clock source in the OSCCON register (OSCCON<1:0> = 01)

The SOSCGO bit is used to warm up the SOSC so that it is ready before any peripheral requests it.

The oscillator is a low-power circuit rated for 32 kHz crystals. It will continue to run during all power-managed modes. The circuit for a typical low-power oscillator is depicted in [Figure 14-2](#). [Table 14-2](#) provides the capacitor selection for the SOSC oscillator.

The user must provide a software time delay to ensure proper start-up of the SOSC oscillator.

**FIGURE 14-2: EXTERNAL COMPONENTS FOR THE SOSC OSCILLATOR**



**Note:** See the Notes with [Table 14-2](#) for additional information about capacitor selection.

**TABLE 14-2: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR<sup>(2,3,4,5)</sup>**

Oscillator Type	Freq.	C1	C2
LP	32 kHz	12 pF <sup>(1)</sup>	12 pF <sup>(1)</sup>

**Note 1:** Microchip suggests these values 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. Values listed would be typical of a CL = 10 pF rated crystal, when SOSCSEL<1:0> = 11.

**5:** Incorrect capacitance value may result in a frequency not meeting the crystal manufacturer's tolerance specification.

The SOSC crystal oscillator drive level is determined based on the SOSCSELx (CONFIG1L<4:3>) Configuration bits. The Higher Drive Level mode, SOSCSEL<1:0> = 11, is intended to drive a wide variety of 32.768 kHz crystals with a variety of Load Capacitance (CL) ratings.

The Lower Drive Level mode is highly optimized for extremely low-power consumption. It is not intended to drive all types of 32.768 kHz crystals. In the Low Drive Level mode, the crystal oscillator circuit may not work correctly if excessively large discrete capacitors are placed on the SOSCO and SOSCI pins. This mode is designed to work only with discrete capacitances of approximately 3 pF-10 pF on each pin.

Crystal manufacturers usually specify a CL (Load Capacitance) rating for their crystals. This value is related to, but not necessarily the same as, the values that should be used for C1 and C2 in [Figure 14-2](#).

For more details on selecting the optimum C1 and C2 for a given crystal, see the crystal manufacture's applications information. The optimum value depends in part on the amount of parasitic capacitance in the circuit, which is often unknown. For that reason, it is highly recommended that thorough testing and validation of the oscillator be performed after values have been selected.

### 14.5.1 USING SOSC AS A CLOCK SOURCE

The SOSC oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS<1:0> (OSCCON<1:0>), to '01', the device switches to SEC\_RUN mode and both the CPU and peripherals are clocked from the SOSC 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 SOSC oscillator is providing the clock source, the SOSC System Clock Status flag, SOSCRUN (OSCCON2<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source currently being used by the Fail-Safe Clock Monitor.

If the Clock Monitor is enabled and the SOSC oscillator fails while providing the clock, polling the SOCSRUN bit will indicate whether the clock is being provided by the SOSC oscillator or another source.

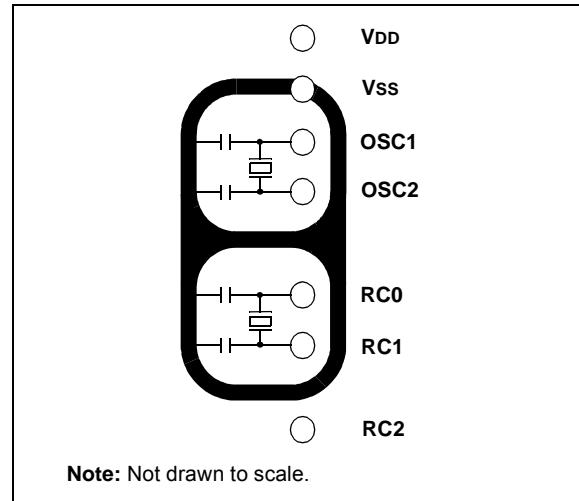
### 14.5.2 SOSC OSCILLATOR LAYOUT CONSIDERATIONS

The SOSC oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity. This is especially true when the oscillator is configured for extremely Low-Power mode, SOSCSEL<1:0> (CONFIG1L<4:3>) = 01.

The oscillator circuit, displayed in [Figure 14-2](#), 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, it may help to have a grounded guard ring around the oscillator circuit. The guard, as displayed in [Figure 14-3](#), could be used on a single-sided PCB or in addition to a ground plane. (Examples of a high-speed circuit include the ECCP1 pin, in Output Compare or PWM mode, or the primary oscillator, using the OSC2 pin.)

**FIGURE 14-3: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**



In the Low Drive Level mode, SOSCSEL<1:0> = 01, it is critical that RC2 I/O pin signals be kept away from the oscillator circuit. Configuring RC2 as a digital output, and toggling it, can potentially disturb the oscillator circuit, even with a relatively good PCB layout. If possible, either leave RC2 unused or use it as an input pin with a slew rate limited signal source. If RC2 must be used as a digital output, it may be necessary to use the Higher Drive Level Oscillator mode (SOSCSEL<1:0> = 11) with many PCB layouts.

Even in the Higher Drive Level mode, careful layout procedures should still be followed when designing the oscillator circuit.

In addition to dV/dt induced noise considerations, it is important to ensure that the circuit board is clean. Even a very small amount of conductive, soldering flux residue can cause PCB leakage currents that can overwhelm the oscillator circuit.

### 14.6 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 or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

# PIC18F66K80 FAMILY

## 14.7 Resetting Timer1 Using the ECCP Special Event Trigger

If ECCP modules are configured to use Timer1 and to generate a Special Event Trigger in Compare mode ( $\text{CCP1M}\langle 3:0 \rangle = 1011$ ), this signal will reset Timer1. The trigger from ECCP will also start an A/D conversion if the A/D module is enabled. (For more information, see [Section 20.3.4 “Special Event Trigger”](#).)

To take advantage of this feature, the module must be configured as either a timer or a synchronous counter. When used this way, the CCPR1H:CCPR1L register pair effectively becomes a Period register for Timer1.

If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a Special Event Trigger, the write operation will take precedence.

**Note:** The Special Event Trigger from the ECCP module will only clear the TMR1 register's content, but not set the TMR1IF interrupt flag bit ( $\text{PIR1}\langle 0 \rangle$ ).

## 14.8 Timer1 Gate

Timer1 can be configured to count freely or the count can be enabled and disabled using the Timer1 gate circuitry. This is also referred to as Timer1 gate count enable.

Timer1 gate can also be driven by multiple selectable sources.

### 14.8.1 TIMER1 GATE COUNT ENABLE

The Timer1 Gate Enable mode is enabled by setting the TMR1GE bit of the T1GCON register. The polarity of the Timer1 Gate Enable mode is configured using the T1GPOL bit (T1GCON $\langle 6 \rangle$ ).

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See [Figure 14-4](#) for timing details.

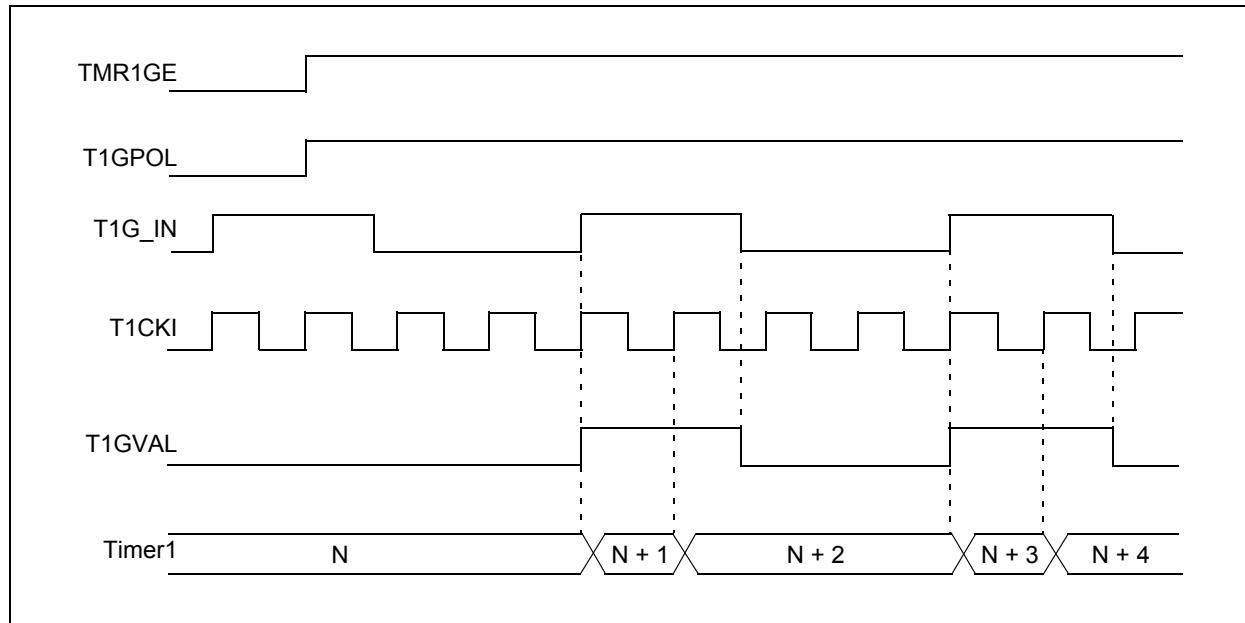
**TABLE 14-3: TIMER1 GATE ENABLE SELECTIONS**

T1CLK <sup>(†)</sup>	T1GPOL (T1GCON $\langle 6 \rangle$ )	T1G Pin	Timer1 Operation
↑	0	0	Counts
↑	0	1	Holds Count
↑	1	0	Holds Count
↑	1	1	Counts

† The clock on which TMR1 is running. For more information, see [Figure 14-1](#).

**Note:** The CCP and ECCP modules use Timers, 1 through 4, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRS register. For more details, see [Register 20-2](#) and [Register 19-2](#).

**FIGURE 14-4: TIMER1 GATE COUNT ENABLE MODE**



## 14.8.2 TIMER1 GATE SOURCE SELECTION

The Timer1 gate source can be selected from one of four sources. Source selection is controlled by the T1GSSx (T1GCON<1:0>) bits (see [Table 14-4](#)).

**TABLE 14-4: TIMER1 GATE SOURCES**

T1GSS<1:0>	Timer1 Gate Source
00	Timer1 Gate Pin
01	TMR2 to Match PR2 (TMR2 increments to match PR2)
10	Comparator 1 Output (comparator logic high output)
11	Comparator 2 Output (comparator logic high output)

The polarity for each available source is also selectable, controlled by the T1GPOL bit (T1GCON<6>).

### 14.8.2.1 T1G Pin Gate Operation

The T1G pin is one source for Timer1 gate control. It can be used to supply an external source to the Timer1 gate circuitry.

### 14.8.2.2 Timer2 Match Gate Operation

The TMR2 register will increment until it matches the value in the PR2 register. On the very next increment cycle, TMR2 will be reset to 00h. When this Reset occurs, a low-to-high pulse will automatically be generated and internally supplied to the Timer1 gate circuitry. The pulse will remain high for one instruction cycle and will return back to a low state until the next match.

Depending on T1GPOL, Timer1 increments differently when TMR2 matches PR2. When T1GPOL = 1, Timer1 increments for a single instruction cycle following a TMR2 match with PR2. When T1GPOL = 0, Timer1 increments continuously except for the cycle following the match when the gate signal goes from low-to-high.

### 14.8.2.3 Comparator 1 Output Gate Operation

The output of Comparator 1 can be internally supplied to the Timer1 gate circuitry. After setting up Comparator 1 with the CM1CON register, Timer1 will increment depending on the transitions of the CMP1OUT (CMSTAT<6>) bit.

### 14.8.2.4 Comparator 2 Output Gate Operation

The output of Comparator 2 can be internally supplied to the Timer1 gate circuitry. After setting up Comparator 2 with the CM2CON register, Timer1 will increment depending on the transitions of the CMP2OUT (CMSTAT<7>) bit.

# PIC18F66K80 FAMILY

## 14.8.3 TIMER1 GATE TOGGLE MODE

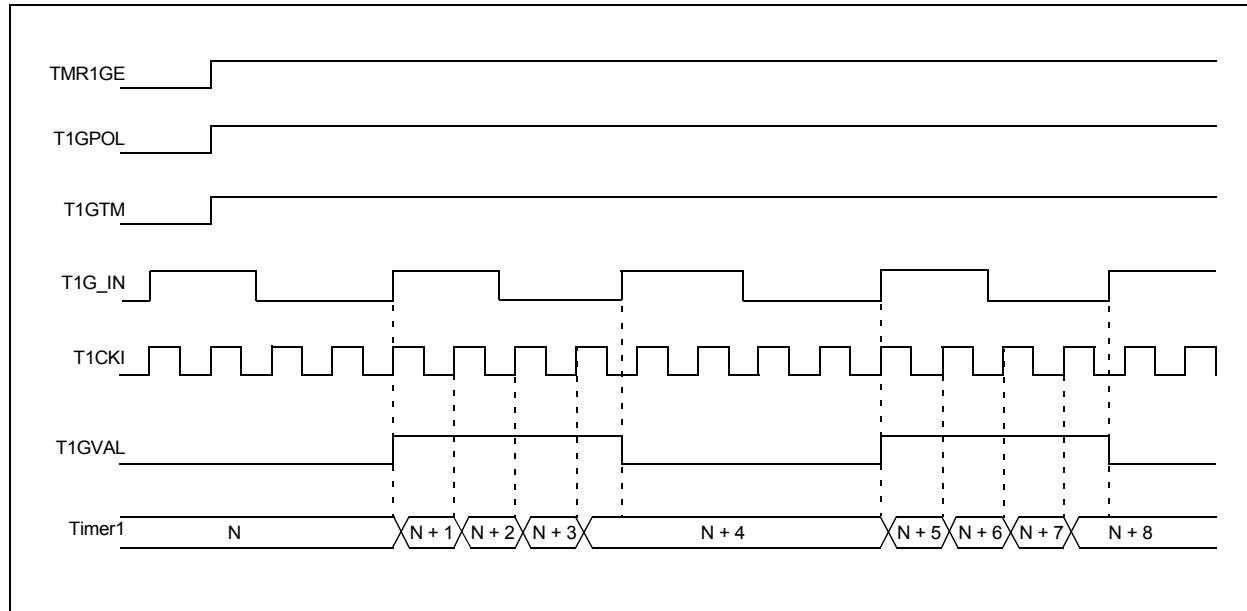
When Timer1 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. (For timing details, see [Figure 14-5](#).)

The T1GVAL bit (T1GCON<2>) indicates when the Toggled mode is active and the timer is counting.

The Timer1 Gate Toggle mode is enabled by setting the T1GTM bit (T1GCON<5>). When T1GTM is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

**FIGURE 14-5: TIMER1 GATE TOGGLE MODE**



## 14.8.4 TIMER1 GATE SINGLE PULSE MODE

When Timer1 Gate Single Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer1 Gate Single Pulse mode is enabled by setting the T1GSPM bit (T1GCON<4>) and the T1GGO/T1DONE bit (T1GCON<3>). The Timer1 will be fully enabled on the next incrementing edge.

On the next trailing edge of the pulse, the T1GGO/T1DONE bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the T1GGO/T1DONE bit is once again set in software.

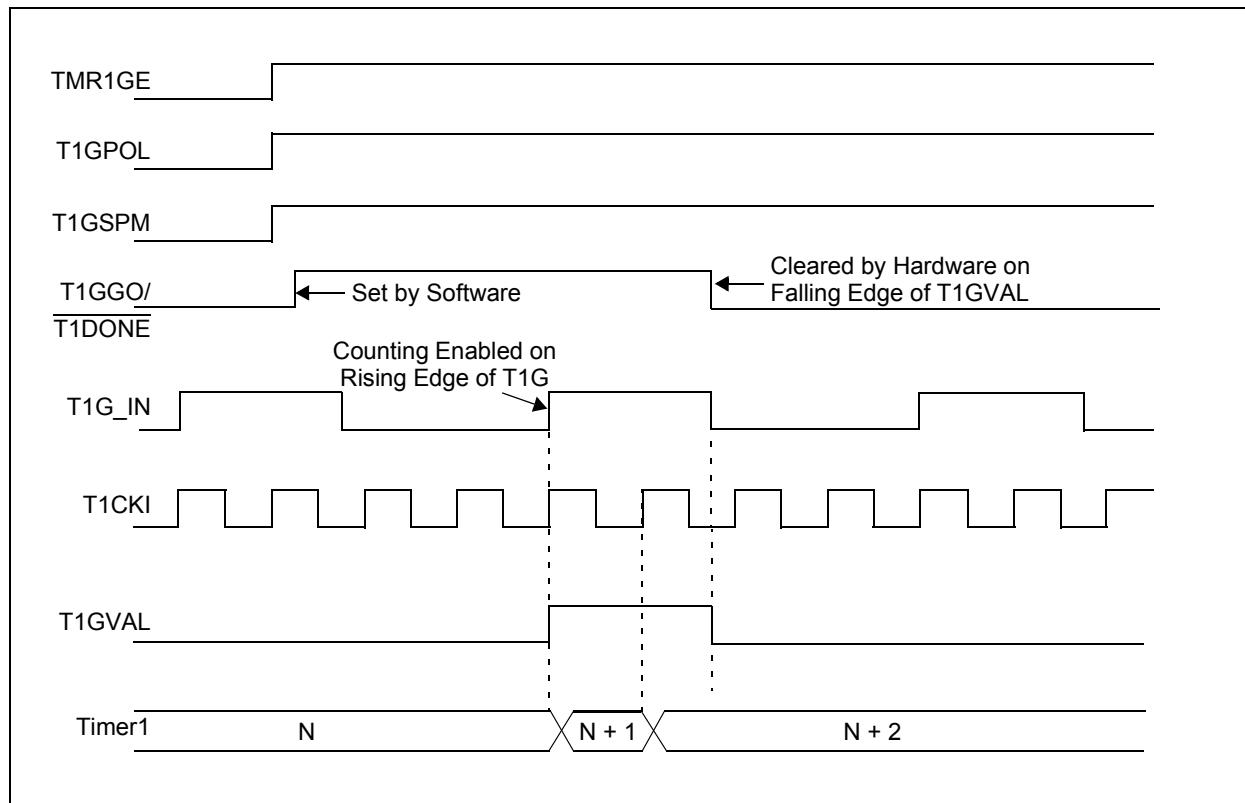
Clearing the T1GSPM bit of the T1GCON register will also clear the T1GGO/T1DONE bit. (For timing details, see [Figure 14-6](#).)

Simultaneously enabling the Toggle and Single Pulse modes will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. (For timing details, see [Figure 14-7](#).)

## 14.8.5 TIMER1 GATE VALUE STATUS

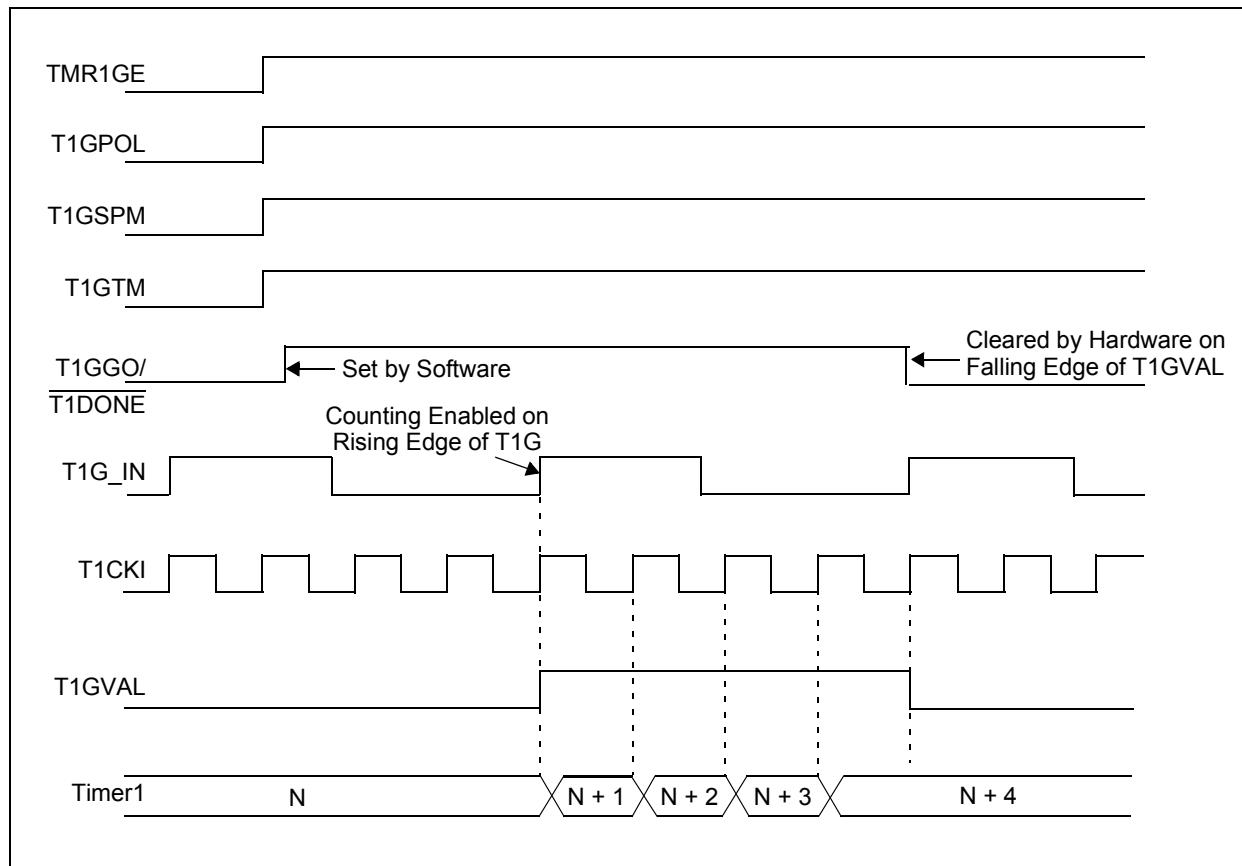
When the Timer1 gate value status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the T1GVAL bit (T1GCON<2>). This bit is valid even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

**FIGURE 14-6: TIMER1 GATE SINGLE PULSE MODE**



# PIC18F66K80 FAMILY

**FIGURE 14-7: TIMER1 GATE SINGLE PULSE AND TOGGLE COMBINED MODE**



**TABLE 14-5: 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
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
TMR1L	Timer1 Register Low Byte							
TMR1H	Timer1 Register High Byte							
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	SOSCEN	T1SYNC	RD16	TMR1ON
T1GCON	TMR1GE	T1GPO	T1GTM	T1GSPM	T1GGO/ T1DONE	T1GVAL	T1GSS1	T1GSS0
OSCCON2	—	SOSCRUN	—	SOSCDRV	SOSCIGO	—	MFIOFS	MFIOSEL
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

**Legend:** Shaded cells are not used by the Timer1 module.

## 15.0 TIMER2 MODULE

The Timer2 module incorporates the following features:

- Eight-bit Timer and Period registers (TMR2 and PR2, respectively)
- Both registers are readable and writable
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscale (1:1 through 1:16)
- Interrupt on TMR2 to PR2 match
- Optional use as the shift clock for the MSSP module

The module is controlled through the T2CON register ([Register 15-1](#)) that enables or disables the timer, and configures the prescaler and postscale. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in [Figure 15-1](#).

### 15.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (Fosc/4). A four-bit counter/prescaler on the clock input gives the prescale options of direct input, divide-by-4 or divide-by-16. These are selected by the prescaler control bits, T2CKPS<1:0> (T2CON<1:0>).

The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscaler. (See [Section 15.2 “Timer2 Interrupt”](#).)

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. Both the prescaler and postscale counters are cleared on the following events:

- A write to the TMR2 register
- A write to the T2CON register
- Any device Reset – Power-on Reset (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or Brown-out Reset (BOR)

TMR2 is not cleared when T2CON is written.

**Note:** The CCP and ECCP modules use Timers, 1 through 4, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRS register. For more details, see [Register 20-2](#) and [Register 19-2](#).

## REGISTER 15-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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	<b>Unimplemented:</b> Read as ‘0’
bit 6-3	<b>T2OUTPS&lt;3:0&gt;:</b> Timer2 Output Postscale Select bits 0000 = 1:1 Postscale 0001 = 1:2 Postscale • • • 1111 = 1:16 Postscale
bit 2	<b>TMR2ON:</b> Timer2 On bit 1 = Timer2 is on 0 = Timer2 is off
bit 1-0	<b>T2CKPS&lt;1:0&gt;:</b> Timer2 Clock Prescale Select bits 00 = Prescaler is 1 01 = Prescaler is 4 1x = Prescaler is 16

# PIC18F66K80 FAMILY

## 15.2 Timer2 Interrupt

Timer2 can also generate an optional device interrupt. The Timer2 output signal (TMR2 to PR2 match) provides the input for the four-bit output counter/postscaler. This counter generates the TMR2 match interrupt flag, which is latched in TMR2IF (PIR1<1>). The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE (PIE1<1>).

A range of 16 postscaler options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS<3:0> (T2CON<6:3>).

## 15.3 Timer2 Output

The unscaled output of TMR2 is available primarily to the ECCP modules, where it is used as a time base for operations in PWM mode.

Timer2 can optionally be used as the shift clock source for the MSSP module operating in SPI mode. Additional information is provided in [Section 21.0 "Master Synchronous Serial Port \(MSSP\) Module"](#).

FIGURE 15-1: TIMER2 BLOCK DIAGRAM

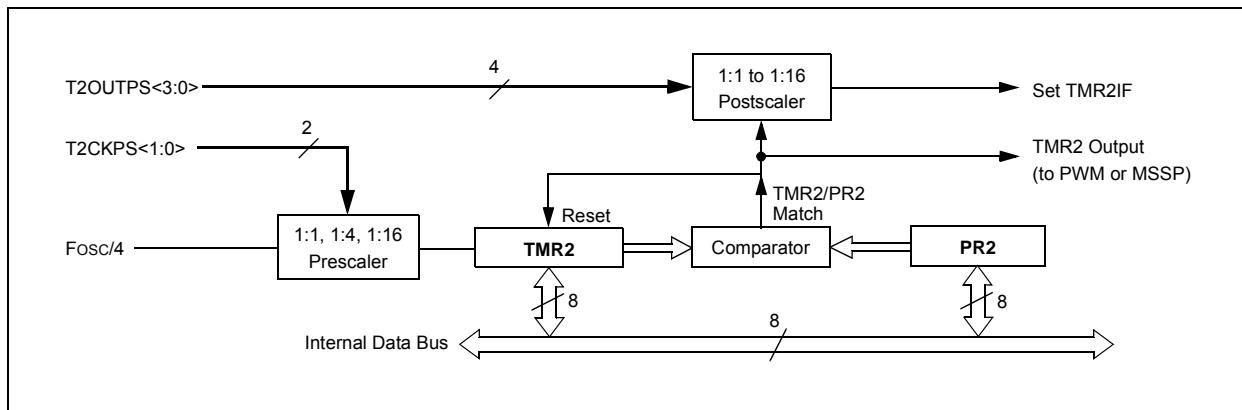


TABLE 15-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
TMR2	Timer2 Register							
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
PR2	Timer2 Period Register							
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

## 16.0 TIMER3 MODULE

The Timer3 timer/counter modules incorporate these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable eight-bit registers (TMR3H and TMR3L)
- Selectable clock source (internal or external) with device clock or SOSC oscillator internal options
- Interrupt-on-overflow
- Module Reset on ECCP Special Event Trigger

A simplified block diagram of the Timer3 module is shown in [Figure 16-1](#).

The Timer3 module is controlled through the T3CON register ([Register 16-1](#)). It also selects the clock source options for the ECCP modules. (For more information, see [Section 20.1.1 “ECCP Module and Timer Resources”](#).)

The Fosc clock source should not be used with the ECCP capture/compare features. If the timer will be used with the capture or compare features, always select one of the other timer clocking options.

### REGISTER 16-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	SOSCEN	T3SYNC	RD16	TMR3ON
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      **TMR3CS<1:0>**: Timer3 Clock Source Select bits  
 10 = Timer3 clock source is either from pin or oscillator, depending on the SOSCEN bit:  
**SOSCEN = 0:**  
 External clock is from T3CKI pin (on the rising edge).  
**SOSCEN = 1:**  
 Depending on the SOSCSELx Configuration bit, the clock source is either a crystal oscillator on SOSCI/SOSCO or an internal digital clock from the SCLKI pin.  
 01 = Timerx clock source is system clock (Fosc)<sup>(1)</sup>  
 00 = Timerx clock source is instruction clock (Fosc/4)
- bit 5-4      **T3CKPS<1:0>**: Timer3 Input Clock Prescale Select bits  
 11 = 1:8 Prescale value  
 10 = 1:4 Prescale value  
 01 = 1:2 Prescale value  
 00 = 1:1 Prescale value
- bit 3      **SOSCEN**: SOSC Oscillator Enable bit  
 1 = SOSC is enabled and available for Timer3  
 0 = SOSC is disabled and available for Timer3
- bit 2      **T3SYNC**: Timer3 External Clock Input Synchronization Control bit  
 (Not usable if the device clock comes from Timer1/Timer3.)  
**When TMR3CS<1:0> = 10:**  
 1 = Does not synchronize external clock input  
 0 = Synchronizes external clock input  
**When TMR3CS<1:0> = 0x:**  
 This bit is ignored; Timer3 uses the internal clock.
- bit 1      **RD16**: 16-Bit Read/Write Mode Enable bit  
 1 = Enables register read/write of Timer3 in one 16-bit operation  
 0 = Enables register read/write of Timer3 in two eight-bit operations
- bit 0      **TMR3ON**: Timer3 On bit  
 1 = Enables Timer3  
 0 = Stops Timer3

**Note 1:** The Fosc clock source should not be selected if the timer will be used with the ECCP capture/compare features.

# PIC18F66K80 FAMILY

## 16.1 Timer3 Gate Control Register

The Timer3 Gate Control register (T3GCON), provided in Register 14-2, is used to control the Timer3 gate.

### REGISTER 16-2: T3GCON: TIMER3 GATE CONTROL REGISTER<sup>(1)</sup>

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-x	R/W-0	R/W-0
TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/T3DONE	T3GVAL	T3GSS1	T3GSS0
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	<b>TMR3GE:</b> Timer3 Gate Enable bit  If TMR3ON = 0: This bit is ignored.  If TMR3ON = 1: 1 = Timer3 counting is controlled by the Timer3 gate function 0 = Timer3 counts regardless of Timer3 gate function
bit 6	<b>T3GPOL:</b> Timer3 Gate Polarity bit 1 = Timer3 gate is active-high (Timer3 counts when gate is high) 0 = Timer3 gate is active-low (Timer3 counts when gate is low)
bit 5	<b>T3GTM:</b> Timer3 Gate Toggle Mode bit 1 = Timer3 Gate Toggle mode is enabled. 0 = Timer3 Gate Toggle mode is disabled and toggle flip-flop is cleared Timer3 gate flip-flop toggles on every rising edge.
bit 4	<b>T3GSPM:</b> Timerx Gate Single Pulse Mode bit 1 = Timer3 Gate Single Pulse mode is enabled and is controlling Timer3 gate 0 = Timer3 Gate Single Pulse mode is disabled
bit 3	<b>T3GGO/T3DONE:</b> Timer3 Gate Single Pulse Acquisition Status bit 1 = Timer3 Gate Single Pulse mode acquisition is ready, waiting for an edge 0 = Timer3 Gate Single Pulse mode acquisition has completed or has not been started This bit is automatically cleared when T3GSPM is cleared.
bit 2	<b>T3GVAL:</b> Timer3 Gate Current State bit Indicates the current state of the Timerx gate that could be provided to TMR3H:TMR3L. Unaffected by Timerx Gate Enable (TMR3GE) bit.
bit 1-0	<b>T3GSS&lt;1:0&gt;:</b> Timer3 Gate Source Select bits 11 = Comparator 2 output 10 = Comparator 1 output 01 = TMR4 to match PR4 output 00 = Timer3 gate pin Watchdog Timer oscillator is turned on if TMR3GE = 1, regardless of the state of TMR3ON.

**Note 1:** Programming the T3GCON prior to T3CON is recommended.

# PIC18F66K80 FAMILY

## REGISTER 16-3: OSCCON2: OSCILLATOR CONTROL REGISTER 2

U-0	R-0	U-0	RW-1	R/W-0	U-0	R-x	R/W-0
—	SOSCRUN	—	SOSCDRV <sup>(1)</sup>	SOSCGO	—	MFIOFS	MFIOSEL
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      **SOSCRUN:** SOSC Run Status bit  
1 = System clock comes from a secondary SOSC  
0 = System clock comes from an oscillator other than SOSC
- bit 5      **Unimplemented:** Read as '0'
- bit 4      **SOSCDRV:** Secondary Oscillator Drive Control bit<sup>(1)</sup>  
1 = High-power SOSC circuit selected  
0 = Low/high-power select is done via the SOSCSEL<1:0> Configuration bits
- bit 3      **SOSCGO:** Oscillator Start Control bit  
1 = Oscillator is running even if no other sources are requesting it  
0 = Oscillator is shut off if no other sources are requesting it (When the SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.)
- bit 2      **Unimplemented:** Read as '0'
- bit 1      **MFIOFS:** MF-INTOSC Frequency Stable bit  
1 = MF-INTOSC is stable  
0 = MF-INTOSC is not stable
- bit 0      **MFIOSEL:** MF-INTOSC Select bit  
1 = MF-INTOSC is used in place of HF-INTOSC frequencies of 500 kHz, 250 kHz and 31.25 kHz  
0 = MF-INTOSC is not used

**Note 1:** When SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.

# PIC18F66K80 FAMILY

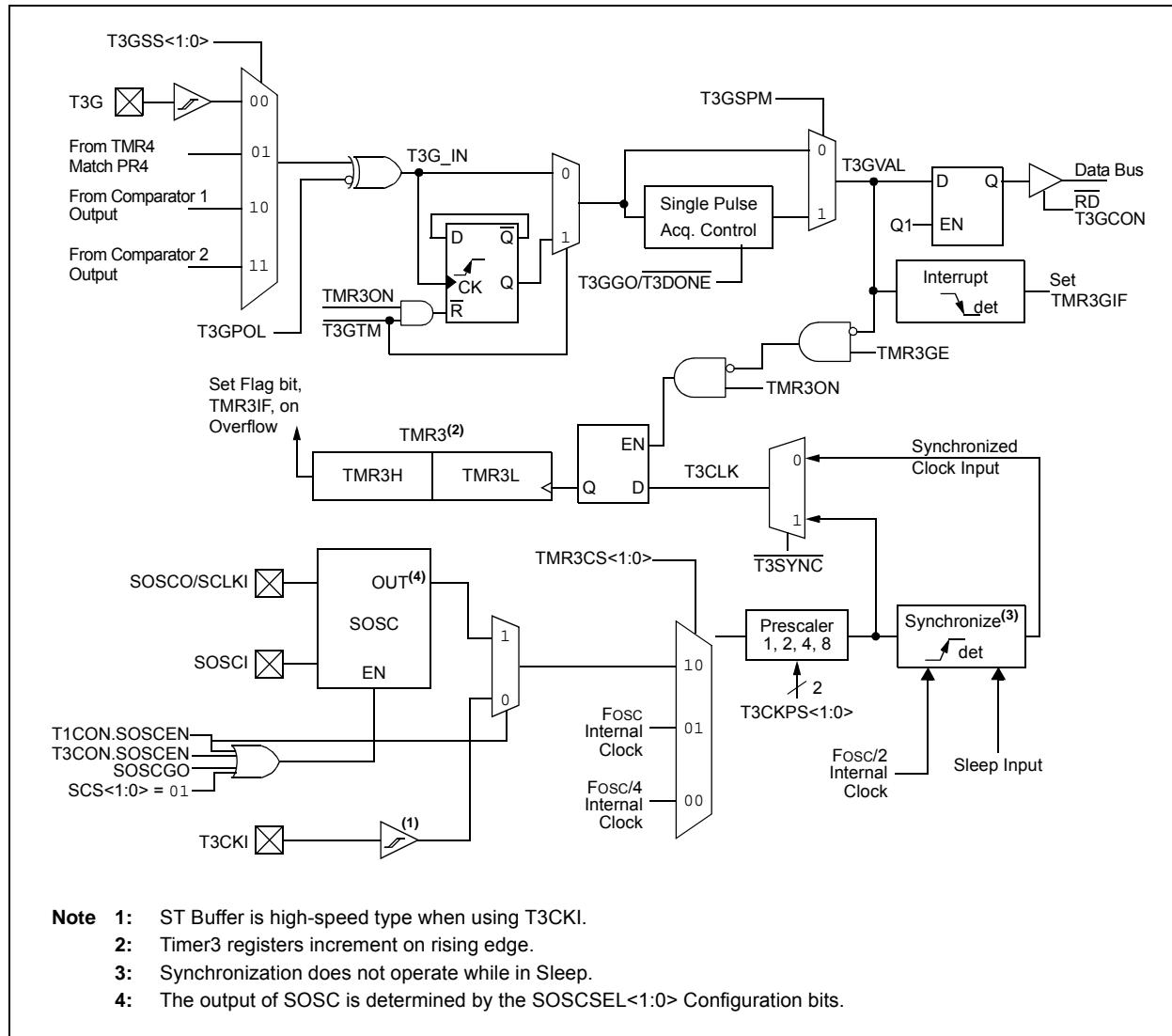
## 16.2 Timer3 Operation

Timer3 can operate in these modes:

- Timer
- Synchronous Counter
- Asynchronous Counter
- Timer with Gated Control

The operating mode is determined by the clock select bits, TMR3CSx (T3CON<7:6>). When the TMR3CSx bits are cleared (= 00), Timer3 increments on every internal instruction cycle (Fosc/4). When TMR3CSx = 01, the Timer3 clock source is the system clock (Fosc), and when it is '10', Timer3 works as a counter from the external clock from the T3CKI pin (on the rising edge after the first falling edge) or the SOSC oscillator.

FIGURE 16-1: TIMER3 BLOCK DIAGRAM



## 16.3 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see [Figure 16.3](#)). When the RD16 control bit (T3CON<1>) is set, the address for TMR3H is mapped to a buffer register for the high byte of Timer3. A read from TMR3L will load the contents of the high byte of Timer3 into the Timer3 High Byte Buffer register. This provides users with the ability to accurately read all 16 bits of Timer3 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

A write to the high byte of Timer3 must also take place through the TMR3H Buffer register. The Timer3 high byte is updated with the contents of TMR3H when a write occurs to TMR3L. This allows users to write all 16 bits to both the high and low bytes of Timer3 at once.

The high byte of Timer3 is not directly readable or writable in this mode. All reads and writes must take place through the Timer3 High Byte Buffer register.

Writes to TMR3H do not clear the Timer3 prescaler. The prescaler is only cleared on writes to TMR3L.

## 16.4 Using the SOSC Oscillator as the Timer3 Clock Source

The SOSC internal oscillator may be used as the clock source for Timer3. It can be enabled in one of these ways:

- Setting the SOSCEN bit in either the T1CON or T3CON register (TxCON<3>)
- Setting the SOSCGO bit in the OSCCON2 register (OSCCON2<3>)
- Setting the SCSx bits to secondary clock source in the OSCCON register (OSCCON<1:0> = 01)

The SOSCGO bit is used to warm up the SOSC so that it is ready before any peripheral requests it.

To use it as the Timer3 clock source, the TMR3CSx bits must also be set. As previously noted, this also configures Timer3 to increment on every rising edge of the oscillator source.

The SOSC oscillator is described in [Section 14.5 “SOSC Oscillator”](#).

# PIC18F66K80 FAMILY

## 16.5 Timer3 Gates

Timer3 can be configured to count freely or the count can be enabled and disabled using the Timer3 gate circuitry. This is also referred to as the Timer3 gate count enable.

The Timer3 gate can also be driven by multiple selectable sources.

### 16.5.1 TIMER3 GATE COUNT ENABLE

The Timer3 Gate Enable mode is enabled by setting the TMR3GE bit (TxGCON<7>). The polarity of the Timer3 Gate Enable mode is configured using the T3GPOL bit (T3GCON<6>).

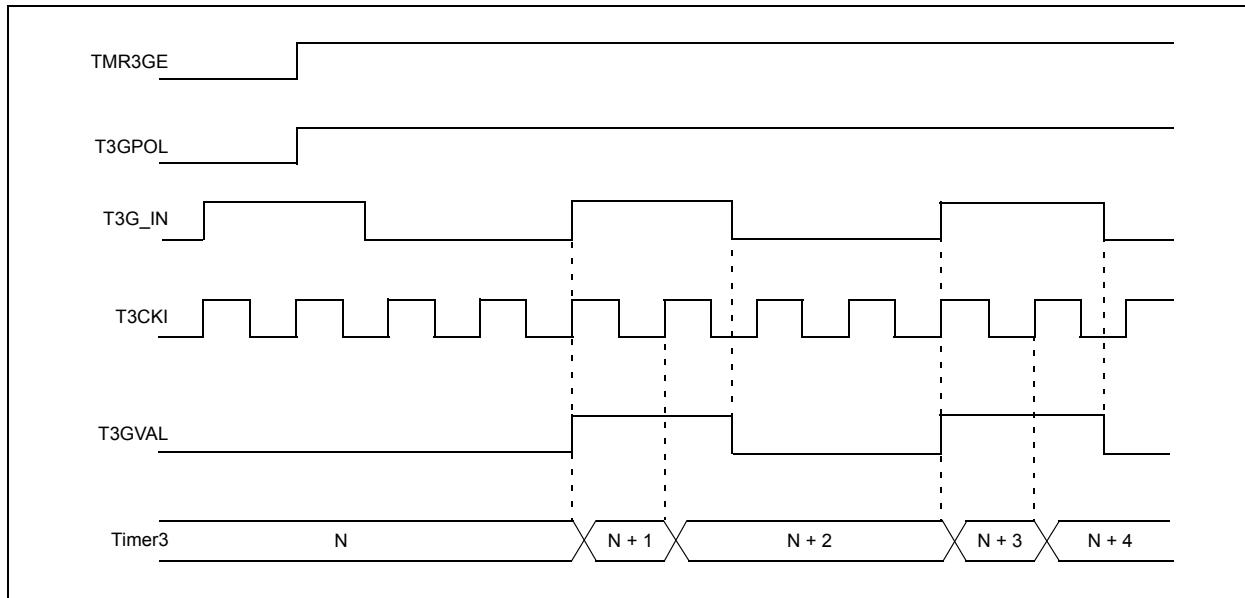
When Timer3 Gate Enable mode is enabled, Timer3 will increment on the rising edge of the Timer3 clock source. When Timer3 Gate Enable mode is disabled, no incrementing will occur and Timer3 will hold the current count. See [Figure 16-2](#) for timing details.

TABLE 16-1: TIMER3 GATE ENABLE SELECTIONS

T3CLK <sup>(†)</sup>	T3GPOL (T3GCON<6>)	T3G Pin	Timer3 Operation
↑	0	0	Counts
↑	0	1	Holds Count
↑	1	0	Holds Count
↑	1	1	Counts

† The clock on which TMR3 is running. For more information, see T3CLK in [Figure 16-1](#).

FIGURE 16-2: TIMER3 GATE COUNT ENABLE MODE



## 16.5.2 TIMER3 GATE SOURCE SELECTION

The Timer3 gate source can be selected from one of four different sources. Source selection is controlled by the T3GSS<1:0> bits (T3GCON<1:0>). The polarity for each available source is also selectable and is controlled by the T3GPOL bit (T3GCON<6>).

**TABLE 16-2: TIMER3 GATE SOURCES**

T3GSS<1:0>	Timer3 Gate Source
00	Timerx Gate Pin
01	TMR4 to Match PR4 (TMR4 increments to match PR4)
10	Comparator 1 Output (comparator logic high output)
11	Comparator 2 Output (comparator logic high output)

### 16.5.2.1 T3G Pin Gate Operation

The T3G pin is one source for Timer3 gate control. It can be used to supply an external source to the Timerx gate circuitry.

### 16.5.2.2 Timer4 Match Gate Operation

The TMR4 register will increment until it matches the value in the PR4 register. On the very next increment cycle, TMR4 will be reset to 00h. When this Reset occurs, a low-to-high pulse will automatically be generated and internally supplied to the Timerx gate circuitry. The pulse will remain high for one instruction cycle and will return back to a low state until the next match.

Depending on T3GPOL, Timerx increments differently when TMR4 matches PR4. When T3GPOL = 1, Timer3 increments for a single instruction cycle following a

TMR4 match with PR4. When T3GPOL = 0, Timer3 increments continuously, except for the cycle following the match, when the gate signal goes from low-to-high.

### 16.5.2.3 Comparator 1 Output Gate Operation

The output of Comparator 1 can be internally supplied to the Timer3 gate circuitry. After setting up Comparator 1 with the CM1CON register, Timer3 will increment depending on the transitions of the CMP1OUT (CMSTAT<6>) bit.

### 16.5.2.4 Comparator 2 Output Gate Operation

The output of Comparator 2 can be internally supplied to the Timer3 gate circuitry. After setting up Comparator 2 with the CM2CON register, Timer3 will increment depending on the transitions of the CMP2OUT (CMSTAT<7>) bit.

### 16.5.3 TIMER3 GATE TOGGLE MODE

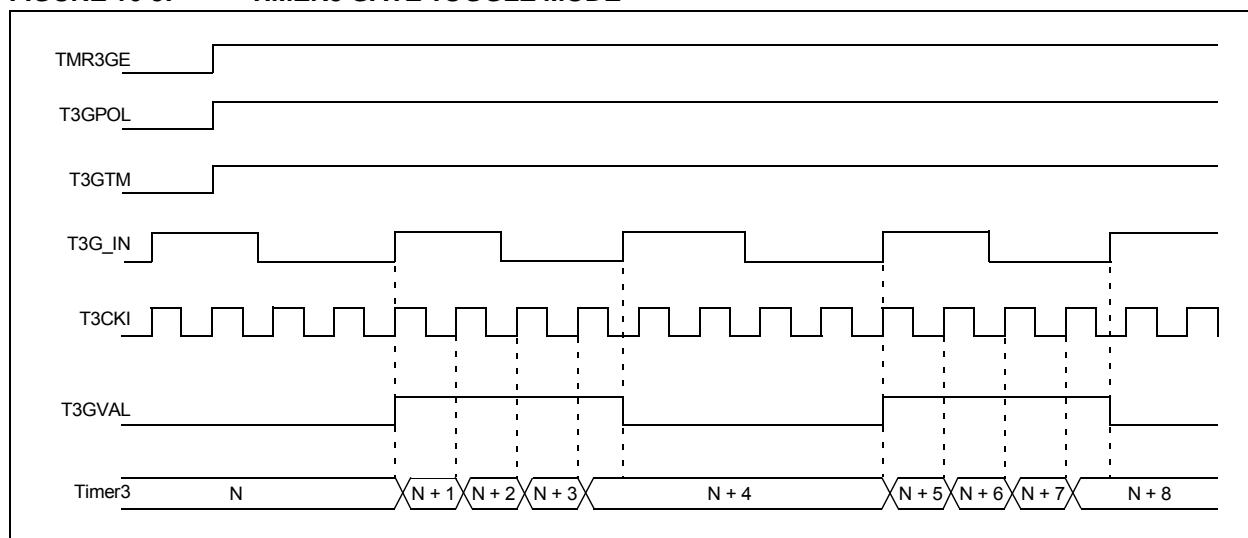
When Timer3 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer3 gate signal, as opposed to the duration of a single level pulse.

The Timer3 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. (For timing details, see [Figure 16-3](#).)

The T3GVAL bit will indicate when the Toggled mode is active and the timer is counting.

Timer3 Gate Toggle mode is enabled by setting the T3GTM bit (T3GCON<5>). When the T3GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

**FIGURE 16-3: TIMER3 GATE TOGGLE MODE**



# PIC18F66K80 FAMILY

## 16.5.4 TIMER3 GATE SINGLE PULSE MODE

When Timer3 Gate Single Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer3 Gate Single Pulse mode is first enabled by setting the T3GSPM bit (T3GCON<4>). Next, the T3GGO/T3DONE bit (T3GCON<3>) must be set.

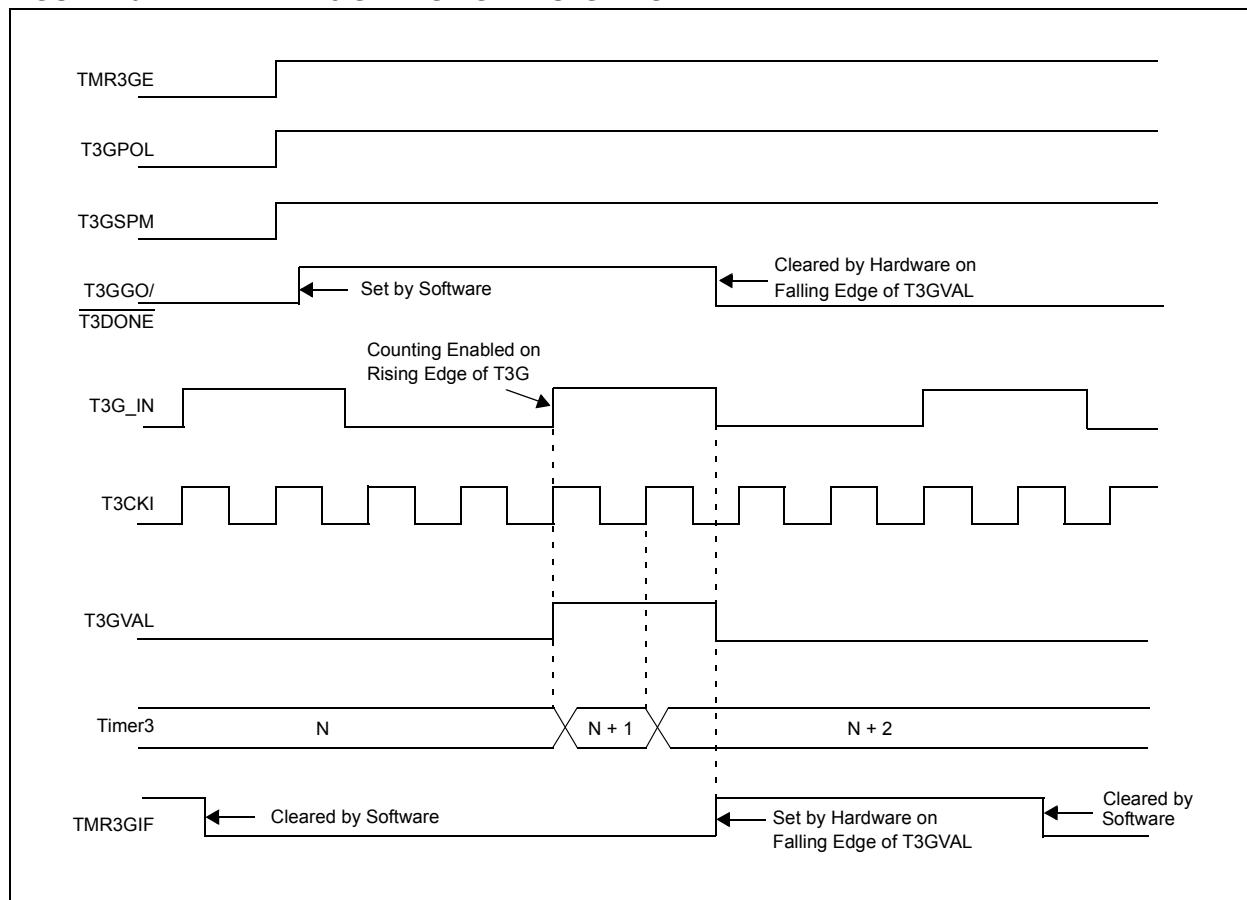
The Timer3 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the T3GGO/T3DONE bit will automatically be cleared. No

other gate events will be allowed to increment Timer3 until the T3GGO/T3DONE bit is once again set in software.

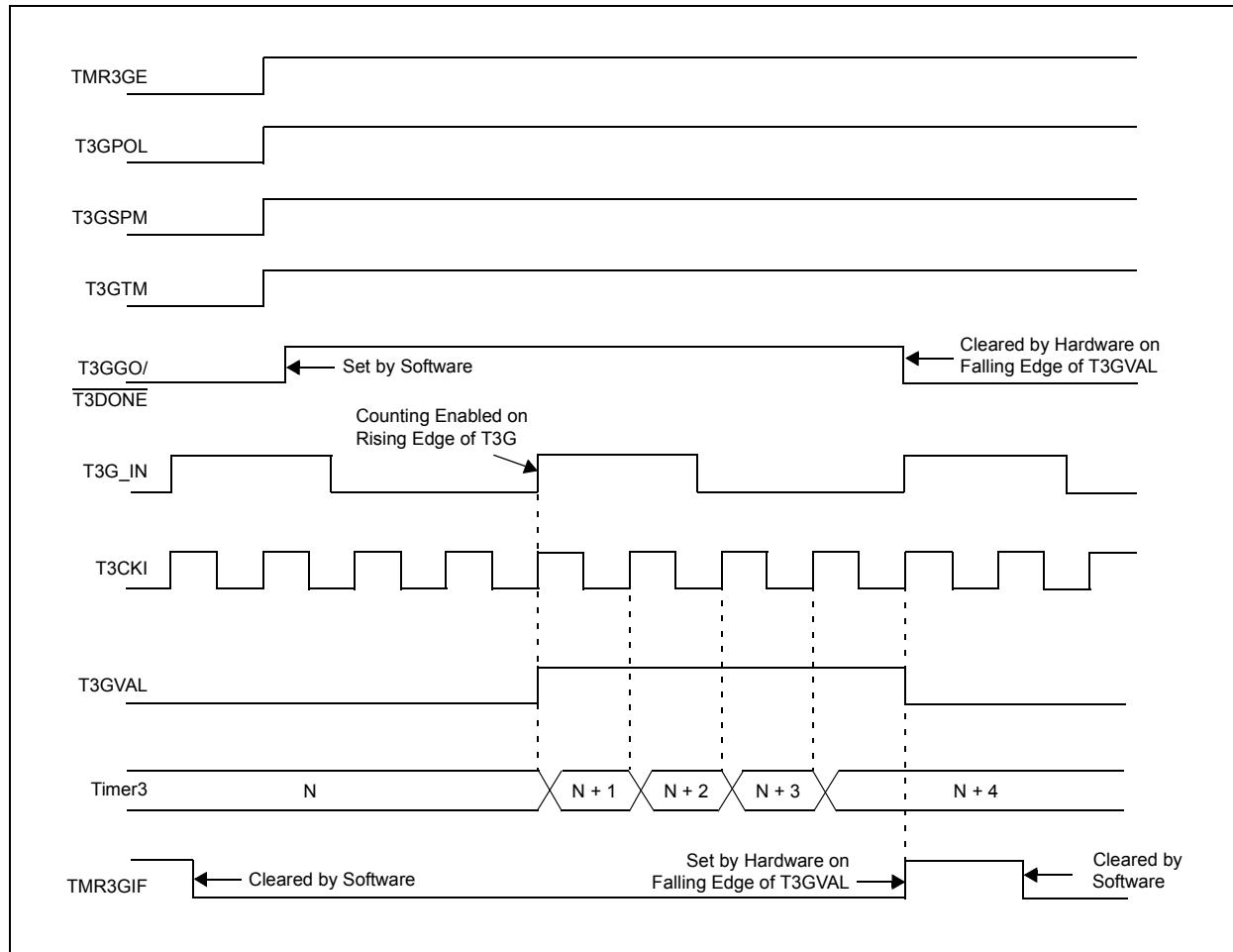
Clearing the T3GSPM bit will also clear the T3GGO/T3DONE bit. (For timing details, see [Figure 16-4](#).)

Simultaneously enabling the Toggle mode and the Single Pulse mode will permit both sections to work together. This allows the cycle times on the Timer3 gate source to be measured. (For timing details, see [Figure 16-5](#).)

**FIGURE 16-4: TIMER3 GATE SINGLE PULSE MODE**



**FIGURE 16-5: TIMER3 GATE SINGLE PULSE AND TOGGLE COMBINED MODE**



### 16.5.5 TIMER3 GATE VALUE STATUS

When Timer3 gate value status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the T3GVAL bit (T3GCON<2>). The T3GVAL bit is valid even when the Timer3 gate is not enabled (TMR3GE bit is cleared).

### 16.5.6 TIMER3 GATE EVENT INTERRUPT

When the Timer3 gate event interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of T3GVAL occurs, the TMR3GIF flag bit in the PIR2 register will be set. If the TMR3GIE bit in the PIE2 register is set, then an interrupt will be recognized.

The TMR3GIF flag bit operates even when the Timer3 gate is not enabled (TMR3GE bit is cleared).

# PIC18F66K80 FAMILY

## 16.6 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and overflows to 0000h. The Timer3 interrupt, if enabled, is generated on overflow and is latched in the interrupt flag bit, TMR3IF. [Table 16-3](#) gives each module's flag bit.

This interrupt can be enabled or disabled by setting or clearing the TMR3IE bit. [Table 16-3](#) displays each module's enable bit.

## 16.7 Resetting Timer3 Using the ECCP Special Event Trigger

If the ECCP modules are configured to use Timer3 and to generate a Special Event Trigger in Compare mode (CCP3M<3:0> = 1011), this signal will reset Timer3. The trigger from ECCP will also start an A/D conversion if the A/D module is enabled (For more information, see [Section 20.3.4 "Special Event Trigger"](#)).

The module must be configured as either a timer or synchronous counter to take advantage of this feature. When used this way, the CCP3H:CCP3L register pair effectively becomes a Period register for Timer3.

If Timer3 is running in Asynchronous Counter mode, the Reset operation may not work.

In the event that a write to Timer3 coincides with a Special Event Trigger from an ECCP module, the write will take precedence.

**Note:** The Special Event Triggers from the ECCPx module will only clear the TMR3 register's content, but not set the TMR3IF interrupt flag bit (PIR2<1>).

**Note:** The CCP and ECCP modules use Timers, 1 through 4, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRS register. For more details, see [Register 20-2](#) and [Register 19-2](#).

**TABLE 16-3: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR5	IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF	TXB0IF	RXB1IF	RXB0IF
PIE5	IRXIE	WAKIE	ERRIE	TX2BIE	TXB1IE	TXB0IE	RXB1IE	RXB0IE
PIR2	OSCFIF	—	—	—	BCLIF	HLVDIF	TMR3IF	TMR3GIF
PIE2	OSCFIE	—	—	—	BCLIE	HLVDIE	TMR3IE	TMR3GIE
TMR3H	Timer3 Register High Byte							
TMR3L	Timer3 Register Low Byte							
T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/ T3DONE	T3GVAL	T3GSS1	T3GSS0
T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	SOSCEN	T3SYNC	RD16	TMR3ON
OSCCON2	—	SOSCRUN	—	SOSCDRV	SOSCIGO	—	MFIOFS	MFIOSEL
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer3 module.

## 17.0 TIMER4 MODULES

The Timer4 timer modules have the following features:

- Eight-bit Timer register (TMR4)
- Eight-bit Period register (PR4)
- Readable and writable (all registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscale (1:1 to 1:16)
- Interrupt on TMR4 match of PR4

The Timer4 modules have a control register shown in [Register 17-1](#). Timer4 can be shut off by clearing control bit, TMR4ON (T4CON<2>), to minimize power consumption. The prescaler and postscale selection of Timer4 also are controlled by this register. [Figure 17-1](#) is a simplified block diagram of the Timer4 modules.

### 17.1 Timer4 Operation

Timer4 can be used as the PWM time base for the PWM mode of the ECCP modules. The TMR4 registers are readable and writable, and are cleared on any device Reset. The input clock (Fosc/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits, T4CKPS<1:0> (T4CON<1:0>). The match output of

TMR4 goes through a four-bit postscale (that gives a 1:1 to 1:16 inclusive scaling) to generate a TMR4 interrupt, latched in the flag bit, TMR4IF. [Table 17-1](#) gives each module's flag bit.

The interrupt can be enabled or disabled by setting or clearing the Timer4 Interrupt Enable bit (TMR4IE), shown in [Table 17-1](#).

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

- A write to the TMR4 register
- A write to the T4CON register
- Any device Reset – Power-on Reset (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or Brown-out Reset (BOR)

A TMR4 is not cleared when a T4CON is written.

**Note:** The CCP and ECCP modules use Timers, 1 through 4, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRS register. For more details, see [Register 20-2](#) and [Register 19-2](#).

## REGISTER 17-1: T4CON: TIMER4 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
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	<b>Unimplemented:</b> Read as '0'
bit 6-3	<b>T4OUTPS&lt;3:0&gt;:</b> Timer4 Output Postscale Select bits 0000 = 1:1 Postscale 0001 = 1:2 Postscale • • • 1111 = 1:16 Postscale
bit 2	<b>TMR4ON:</b> Timer4 On bit 1 = Timer4 is on 0 = Timer4 is off
bit 1-0	<b>T4CKPS&lt;1:0&gt;:</b> Timer4 Clock Prescale Select bits 00 = Prescaler is 1 01 = Prescaler is 4 1x = Prescaler is 16

# PIC18F66K80 FAMILY

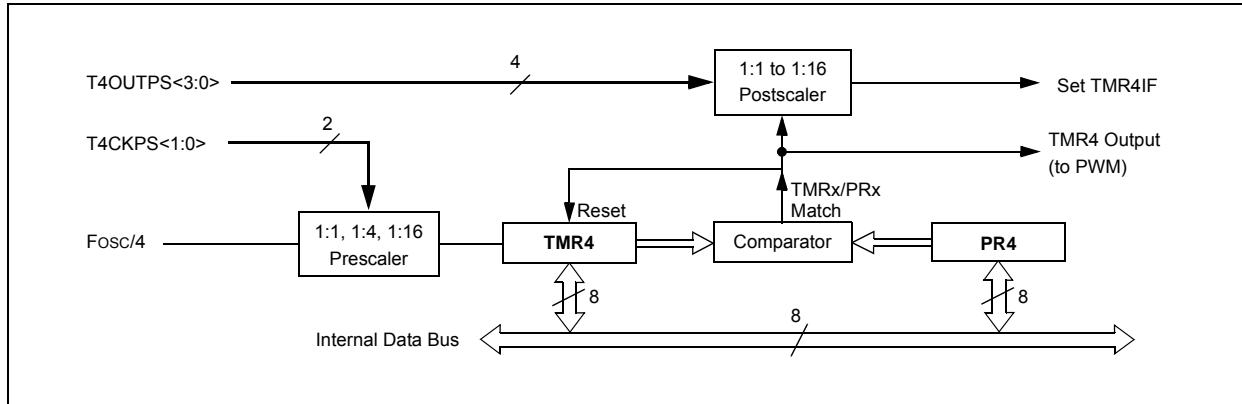
## 17.2 Timer4 Interrupt

The Timer4 module has an eight-bit Period register, PR4, that is both readable and writable. Timer4 increments from 00h until it matches PR4 and then resets to 00h on the next increment cycle. The PR4 register is initialized to FFh upon Reset.

## 17.3 Output of TMR4

The outputs of TMR4 (before the postscaler) are used only as a PWM time base for the CCP modules. They are not used as baud rate clocks for the MSSP module as is the Timer2 output.

**FIGURE 17-1: TIMER4 BLOCK DIAGRAM**



**TABLE 17-1: REGISTERS ASSOCIATED WITH TIMER4 AS A TIMER/COUNTER**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
TMR4	Timer4 Register							
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
PR4	Timer4 Period Register							
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer4 module.

## 18.0 CHARGE TIME MEASUREMENT UNIT (CTMU)

The Charge Time Measurement Unit (CTMU) is a flexible analog module that provides accurate differential time measurement between pulse sources, as well as asynchronous pulse generation. By working with other on-chip analog modules, the CTMU can precisely measure time, capacitance and relative changes in capacitance or generate output pulses with a specific time delay. The CTMU is ideal for interfacing with capacitive-based sensors.

The module includes these key features:

- Up to 11 channels available for capacitive or time measurement input
- Low-cost temperature measurement using on-chip diode channel
- On-chip precision current source
- Four-edge input trigger sources
- Polarity control for each edge source

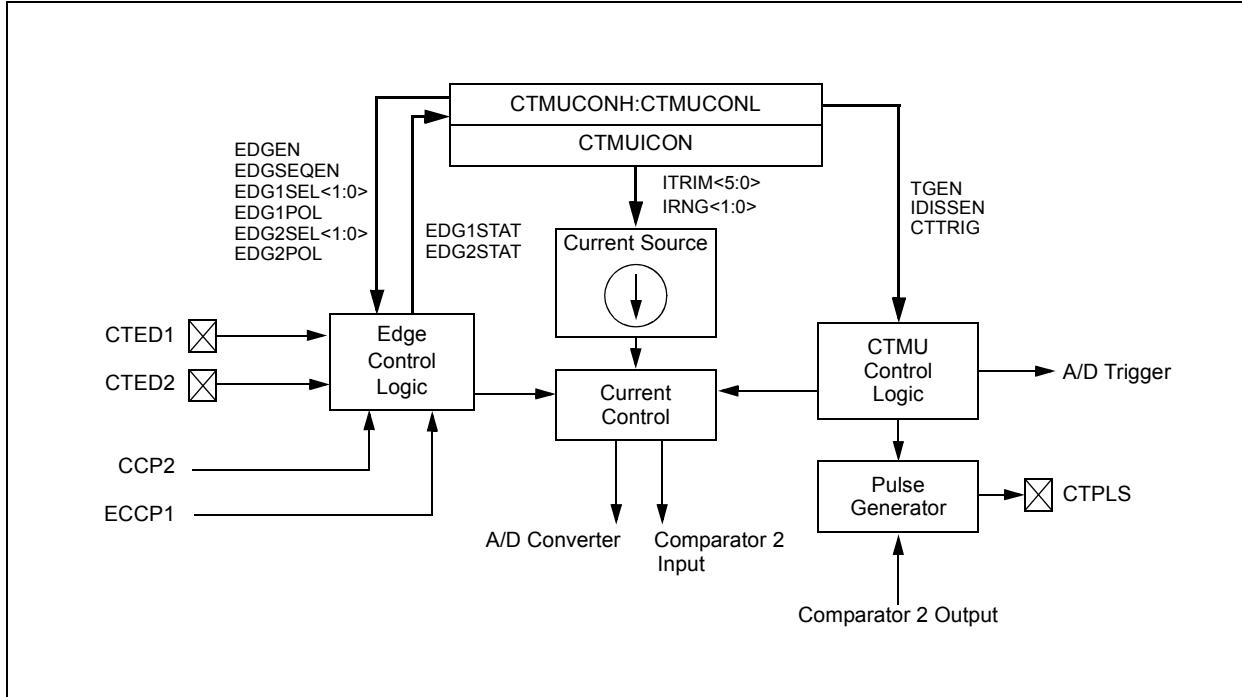
- Control of edge sequence
- Control of response to edges
- Time measurement resolution of 1 nanosecond
- High-precision time measurement
- Time delay of external or internal signal asynchronous to system clock
- Accurate current source suitable for capacitive measurement

The CTMU works in conjunction with the A/D Converter to provide up to 11 channels for time or charge measurement, depending on the specific device and the number of A/D channels available. When configured for time delay, the CTMU is connected to one of the analog comparators. The level-sensitive input edge sources can be selected from four sources: two external inputs or the CCP1/CCP2 Special Event Triggers.

The CTMU special event can trigger the Analog-to-Digital Converter module.

[Figure 18-1](#) provides a block diagram of the CTMU.

**FIGURE 18-1: CTMU BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

## 18.1 CTMU Registers

The control registers for the CTMU are:

- CTMUCONH
- CTMUCONL
- CTMUICON

The CTMUCONH and CTMUCONL registers ([Register 18-1](#) and [Register 18-2](#)) contain control bits for configuring the CTMU module edge source selection, edge source polarity selection, edge sequencing, A/D trigger, analog circuit capacitor discharge and enables. The CTMUICON register ([Register 18-3](#)) has bits for selecting the current source range and current source trim.

### REGISTER 18-1: CTMUCONH: CTMU CONTROL HIGH REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CTMUEEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG
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      **CTMUEEN:** CTMU Enable bit  
1 = Module is enabled  
0 = Module is disabled
- bit 6      **Unimplemented:** Read as '0'
- bit 5      **CTMUSIDL:** Stop in Idle Mode bit  
1 = Discontinues module operation when device enters Idle mode  
0 = Continues module operation in Idle mode
- bit 4      **TGEN:** Time Generation Enable bit  
1 = Enables edge delay generation  
0 = Disables edge delay generation
- bit 3      **EDGEN:** Edge Enable bit  
1 = Edges are not blocked  
0 = Edges are blocked
- bit 2      **ESGSEQEN:** Edge Sequence Enable bit  
1 = Edge 1 event must occur before Edge 2 event can occur  
0 = No edge sequence is needed
- bit 1      **IDISSEN:** Analog Current Source Control bit  
1 = Analog current source output is grounded  
0 = Analog current source output is not grounded
- bit 0      **CTTRIG:** CTMU Special Event Trigger bit  
1 = CTMU Special Event Trigger is enabled  
0 = CTMU Special Event Trigger is disabled

# PIC18F66K80 FAMILY

## REGISTER 18-2: CTMUCONL: CTMU CONTROL LOW 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
EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT
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      **EDG2POL:** Edge 2 Polarity Select bit  
1 = Edge 2 is programmed for a positive edge response  
0 = Edge 2 is programmed for a negative edge response
- bit 6-5     **EDG2SEL<1:0>:** Edge 2 Source Select bits  
11 = CTED1 pin  
10 = CTED2 pin  
01 = ECCP1 Special Event Trigger  
00 = CCP2 Special Event Trigger
- bit 4       **EDG1POL:** Edge 1 Polarity Select bit  
1 = Edge 1 is programmed for a positive edge response  
0 = Edge 1 is programmed for a negative edge response
- bit 3-2     **EDG1SEL<1:0>:** Edge 1 Source Select bits  
11 = CTED1 pin  
10 = CTED2 pin  
01 = ECCP1 Special Event Trigger  
00 = CCP2 Special Event Trigger
- bit 1       **EDG2STAT:** Edge 2 Status bit  
1 = Edge 2 event has occurred  
0 = Edge 2 event has not occurred
- bit 0       **EDG1STAT:** Edge 1 Status bit  
1 = Edge 1 event has occurred  
0 = Edge 1 event has not occurred

# PIC18F66K80 FAMILY

## REGISTER 18-3: CTMUICON: CTMU CURRENT 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
ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0
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 7-2      **ITRIM<5:0>**: Current Source Trim bits

011111 = Maximum positive change (+62% typ.) from nominal current

011110

.

.

000001 = Minimum positive change (+2% typ.) from nominal current

000000 = Nominal current output specified by IRNG<1:0>

111111 = Minimum negative change (-2% typ.) from nominal current

.

.

100010

100001 = Maximum negative change (-62% typ.) from nominal current

bit 1-0      **IRNG<1:0>**: Current Source Range Select bits

11 = 100 x Base Current

10 = 10 x Base Current

01 = Base Current level (0.55  $\mu$ A nominal)

00 = Current source is disabled

## 18.2 CTMU Operation

The CTMU works by using a fixed current source to charge a circuit. The type of circuit depends on the type of measurement being made.

In the case of charge measurement, the current is fixed and the amount of time the current is applied to the circuit is fixed. The amount of voltage read by the A/D becomes a measurement of the circuit's capacitance.

In the case of time measurement, the current, as well as the capacitance of the circuit, is fixed. In this case, the voltage read by the A/D is representative of the amount of time elapsed from the time the current source starts and stops charging the circuit.

If the CTMU is being used as a time delay, both capacitance and current source are fixed, as well as the voltage supplied to the comparator circuit. The delay of a signal is determined by the amount of time it takes the voltage to charge to the comparator threshold voltage.

### 18.2.1 THEORY OF OPERATION

The operation of the CTMU is based on the equation for charge:

$$I = C \cdot \frac{dV}{dT}$$

More simply, the amount of charge measured in coulombs in a circuit is defined as current in amperes (I) multiplied by the amount of time in seconds that the current flows (t). Charge is also defined as the capacitance in farads (C) multiplied by the voltage of the circuit (V). It follows that:

$$I \cdot t = C \cdot V$$

The CTMU module provides a constant, known current source. The A/D Converter is used to measure (V) in the equation, leaving two unknowns: capacitance (C) and time (t). The above equation can be used to calculate capacitance or time, by either the relationship using the known fixed capacitance of the circuit:

$$t = (C \cdot V)/I$$

or by:

$$C = (I \cdot t)/V$$

using a fixed time that the current source is applied to the circuit.

### 18.2.2 CURRENT SOURCE

At the heart of the CTMU is a precision current source, designed to provide a constant reference for measurements. The level of current is user-selectable across three ranges, or a total of two orders of magnitude, with the ability to trim the output in  $\pm 2\%$  increments (nominal). The current range is selected by the IRNG<1:0> bits (CTMUICON<1:0>), with a value of '01' representing the lowest range.

Current trim is provided by the ITRIM<5:0> bits (CTMUICON<7:2>). These six bits allow trimming of the current source in steps of approximately 2% per step. Half of the range adjusts the current source positively and the other half reduces the current source. A value of '000000' is the neutral position (no change). A value of '100001' is the maximum negative adjustment (approximately -62%) and '011111' is the maximum positive adjustment (approximately +62%).

### 18.2.3 EDGE SELECTION AND CONTROL

CTMU measurements are controlled by edge events occurring on the module's two input channels. Each channel, referred to as Edge 1 and Edge 2, can be configured to receive input pulses from one of the edge input pins (CTED1 and CTED2) or CCPx Special Event Triggers (ECCP1 and CCP2). The input channels are level-sensitive, responding to the instantaneous level on the channel rather than a transition between levels. The inputs are selected using the EDG1SEL and EDG2SEL bit pairs (CTMUCONL<3:2>, 6:5>).

In addition to source, each channel can be configured for event polarity using the EDGE2POL and EDGE1POL bits (CTMUCONL<7,4>). The input channels can also be filtered for an edge event sequence (Edge 1 occurring before Edge 2) by setting the EDGSEQEN bit (CTMUCONH<2>).

### 18.2.4 EDGE STATUS

The CTMUCONL register also contains two status bits, EDG2STAT and EDG1STAT (CTMUCONL<1:0>). Their primary function is to show if an edge response has occurred on the corresponding channel. The CTMU automatically sets a particular bit when an edge response is detected on its channel. The level-sensitive nature of the input channels also means that the status bits become set immediately if the channel's configuration is changed and matches the channel's current state.

The module uses the edge status bits to control the current source output to external analog modules (such as the A/D Converter). Current is only supplied to external modules when only one (not both) of the status bits is set. Current is shut off when both bits are either set or cleared. This allows the CTMU to measure current only during the interval between edges. After both status bits are set, it is necessary to clear them before another measurement is taken. Both bits should be cleared simultaneously, if possible, to avoid re-enabling the CTMU current source.

In addition to being set by the CTMU hardware, the edge status bits can also be set by software. This permits a user application to manually enable or disable the current source. Setting either (but not both) of the bits enables the current source. Setting or clearing both bits at once disables the source.

# PIC18F66K80 FAMILY

---

## 18.2.5 INTERRUPTS

The CTMU sets its interrupt flag (PIR3<3>) whenever the current source is enabled, then disabled. An interrupt is generated only if the corresponding interrupt enable bit (PIE3<3>) is also set. If edge sequencing is not enabled (i.e., Edge 1 must occur before Edge 2), it is necessary to monitor the edge status bits and determine which edge occurred last and caused the interrupt.

## 18.3 CTMU Module Initialization

The following sequence is a general guideline used to initialize the CTMU module:

1. Select the current source range using the IRNGx bits (CTMUICON<1:0>).
2. Adjust the current source trim using the ITRIMx bits (CTMUICON<7:2>).
3. Configure the edge input sources for Edge 1 and Edge 2 by setting the EDG1SEL and EDG2SEL bits (CTMUCONL<3:2> and <6:5>, respectively).
4. Configure the input polarities for the edge inputs using the EDG1POL and EDG2POL bits (CTMUCONL<4,7>).

The default configuration is for negative edge polarity (high-to-low transitions).

5. Enable edge sequencing using the EDGSEQEN bit (CTMUCONH<2>).

By default, edge sequencing is disabled.

6. Select the operating mode (Measurement or Time Delay) with the TGEN bit (CTMUCONH<4>).

The default mode is Time/Capacitance Measurement.

7. Configure the module to automatically trigger an A/D conversion when the second edge event has occurred using the CTTRIG bit (CTMUCONH<0>).

The conversion trigger is disabled by default.

8. Discharge the connected circuit by setting the IDISSEN bit (CTMUCONH<1>).

9. After waiting a sufficient time for the circuit to discharge, clear the IDISSEN bit.

10. Disable the module by clearing the CTMUEN bit (CTMUCONH<7>).

11. Clear the Edge Status bits, EDG2STAT and EDG1STAT (CTMUCONL<1:0>).

Both bits should be cleared simultaneously, if possible, to avoid re-enabling the CTMU current source.

12. Enable both edge inputs by setting the EDGEN bit (CTMUCONH<3>).

13. Enable the module by setting the CTMUEN bit.

Depending on the type of measurement or pulse generation being performed, one or more additional modules may also need to be initialized and configured with the CTMU module:

- Edge Source Generation: In addition to the external edge input pins, CCP1/CCP2 Special Event Triggers can be used as edge sources for the CTMU.
- Capacitance or Time Measurement: The CTMU module uses the A/D Converter to measure the voltage across a capacitor that is connected to one of the analog input channels.
- Pulse Generation: When generating system clock independent, output pulses, the CTMU module uses Comparator 2 and the associated comparator voltage reference.

## 18.4 Calibrating the CTMU Module

The CTMU requires calibration for precise measurements of capacitance and time, as well as for accurate time delay. If the application only requires measurement of a relative change in capacitance or time, calibration is usually not necessary. An example of a less precise application is a capacitive touch switch, in which the touch circuit has a baseline capacitance and the added capacitance of the human body changes the overall capacitance of a circuit.

If actual capacitance or time measurement is required, two hardware calibrations must take place:

- The current source needs calibration to set it to a precise current.
- The circuit being measured needs calibration to measure or nullify any capacitance other than that to be measured.

### 18.4.1 CURRENT SOURCE CALIBRATION

The current source on board the CTMU module has a range of  $\pm 62\%$  nominal for each of three current ranges. For precise measurements, it is possible to measure and adjust this current source by placing a high-precision resistor,  $R_{CAL}$ , onto an unused analog channel. An example circuit is shown in [Figure 18-2](#).

To measure the current source:

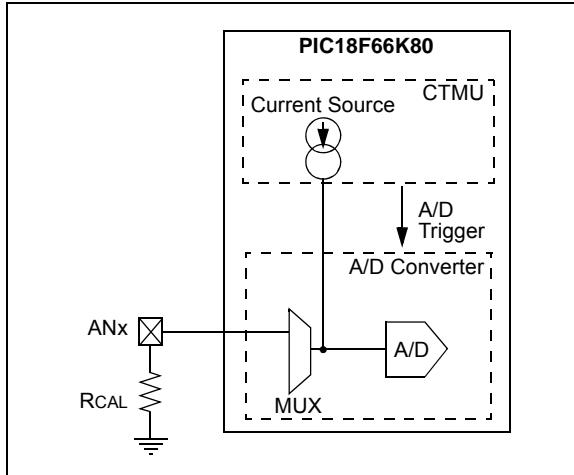
1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Enable the current source by setting EDG1STAT (CTMUCONL<0>).
4. Issue time delay for voltage across  $R_{CAL}$  to stabilize and A/D sample/hold capacitor to charge.
5. Perform the A/D conversion.
6. Calculate the current source current using  $I = V/R_{CAL}$ , where  $R_{CAL}$  is a high-precision resistance and  $V$  is measured by performing an A/D conversion.

The CTMU current source may be trimmed with the trim bits in CTMUICON, using an iterative process to get the exact current desired. Alternatively, the nominal value without adjustment may be used. That value may be stored by software, for use in all subsequent capacitive or time measurements.

To calculate the optimal value for RCAL, the nominal current must be chosen.

For example, if the A/D Converter reference voltage is 3.3V, use 70% of full scale (or 2.31V) as the desired approximate voltage to be read by the A/D Converter. If the range of the CTMU current source is selected to be 0.55  $\mu$ A, the resistor value needed is calculated as  $RCAL = 2.31V / 0.55 \mu$ A, for a value of 4.2 M $\Omega$ . Similarly, if the current source is chosen to be 5.5  $\mu$ A, RCAL would be 420,000 $\Omega$ , and 42,000 $\Omega$  if the current source is set to 55  $\mu$ A.

**FIGURE 18-2:** **CTMU CURRENT SOURCE CALIBRATION CIRCUIT**



A value of 70% of full-scale voltage is chosen to make sure that the A/D Converter is in a range that is well above the noise floor. If an exact current is chosen to incorporate the trimming bits from CTMUICON, the resistor value of RCAL may need to be adjusted accordingly. RCAL also may be adjusted to allow for available resistor values. RCAL should be of the highest precision available, in light of the precision needed for the circuit that the CTMU will be measuring. A recommended minimum would be 0.1% tolerance.

The following examples show a typical method for performing a CTMU current calibration.

- [Example 18-1](#) demonstrates how to initialize the A/D Converter and the CTMU.

This routine is typical for applications using both modules.

- [Example 18-2](#) demonstrates one method for the actual calibration routine.

This method manually triggers the A/D Converter to demonstrate the entire step-wise process. It is also possible to automatically trigger the conversion by setting the CTMU's CTTRIG bit (CTMUCONH<0>).

# PIC18F66K80 FAMILY

## EXAMPLE 18-1: SETUP FOR CTMU CALIBRATION ROUTINES

```
#include "p18cxx.h"
/*********************************************************************
/*Setup CTMU ****
/*********************************************************************
void setup(void)

{ //CTMUCON - CTMU Control register

    CTMUCONH = 0x00;           //make sure CTMU is disabled
    CTMUCONL = 0x90;
    //CTMU continues to run when emulator is stopped,CTMU continues
    //to run in idle mode,Time Generation mode disabled, Edges are blocked
    //No edge sequence order, Analog current source not grounded, trigger
    //output disabled, Edge2 polarity = positive level, Edge2 source =
    //source 0, Edgel polarity = positive level, Edgel source = source 0,
    // Set Edge status bits to zero

    //CTMUICON - CTMU Current Control Register
    CTMUICON = 0x01;          //0.55uA, Nominal - No Adjustment

/*********************************************************************
//Setup AD converter;
/********************************************************************

    TRISA=0x04;                //set channel 2 as an input

    // Configured AN2 as an analog channel
    // ANCON1
    ANCON1 = 0x04;

    // ADCON1
    ADCON2bits.ADFM=1;         // Result format 1= Right justified
    ADCON2bits.ACQT=1;          // Acquisition time 7 = 20TAD 2 = 4TAD 1=2TAD
    ADCON2bits.ADCS=2;          // Clock conversion bits 6= FOSC/64 2=FOSC/32

    // ADCON1
    ADCON1bits.VCFG0 =0;        // Vref+ = AVdd
    ADCON1bits.VCFG1 =0;        // Vref+ = AVdd
    ADCON1bits.VNCFG = 0;       // Vref- = AVss
    ADCON1bits.CHS=2;           // Select ADC channel

    ADCON0bits.ADON=1;          // Turn on ADC

}
```

## EXAMPLE 18-2: CURRENT CALIBRATION ROUTINE

```
#include "p18cxx.h"

#define COUNT 500                      // @ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define RCAL .027
#define ADSCALE 1023
#define ADREF 3.3

int main(void)
{
    int i;
    int j = 0; //index for loop
    unsigned int Vread = 0;
    double VTot = 0;
    float Vavg=0, Vcal=0, CTMUISrc = 0; //float values stored for calcs

    //assume CTMU and A/D have been setup correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONHbits.CTMUEN = 1;           //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;      //drain charge on the circuit
        DELAY;                        //wait 125us
        CTMUCONHbits.IDISSEN = 0;      //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;     //Begin charging the circuit
        //using CTMU current source
        DELAY;                        //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;     //Stop charging circuit

        PIR1bits.ADIF = 0;             //make sure A/D Int not set
        ADCON0bits.GO=1;              //and begin A/D conv.
        while(!PIR1bits.ADIF);        //Wait for A/D convert complete

        Vread = ADRES;                //Get the value from the A/D
        PIR1bits.ADIF = 0;             //Clear A/D Interrupt Flag
        VTot += Vread;                //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);       //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL;            //CTMUISrc is in 1/100ths of uA
}
```

# PIC18F66K80 FAMILY

---

---

## 18.4.2 CAPACITANCE CALIBRATION

There is a small amount of capacitance from the internal A/D Converter sample capacitor as well as stray capacitance from the circuit board traces and pads that affect the precision of capacitance measurements. A measurement of the stray capacitance can be taken by making sure the desired capacitance to be measured has been removed.

After removing the capacitance to be measured:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT (= 1).
3. Wait for a fixed delay of time, t.
4. Clear EDG1STAT.
5. Perform an A/D conversion.
6. Calculate the stray and A/D sample capacitances:

$$COFFSET = CSTRAY + CAD = (I \cdot t) / V$$

Where:

- I is known from the current source measurement step
- t is a fixed delay
- V is measured by performing an A/D conversion

This measured value is then stored and used for calculations of time measurement or subtracted for capacitance measurement. For calibration, it is expected that the capacitance of CSTRAY + CAD is approximately known; CAD is approximately 4 pF.

An iterative process may be required to adjust the time, t, that the circuit is charged to obtain a reasonable voltage reading from the A/D Converter. The value of t may be determined by setting COFFSET to a theoretical value and solving for t. For example, if CSTRAY is theoretically calculated to be 11 pF, and V is expected to be 70% of VDD or 2.31V, t would be:

$$(4 \text{ pF} + 11 \text{ pF}) \cdot 2.31\text{V} / 0.55 \mu\text{A}$$

or 63  $\mu$ s.

See [Example 18-3](#) for a typical routine for CTMU capacitance calibration.

## EXAMPLE 18-3: CAPACITANCE CALIBRATION ROUTINE

```
#include "p18cxx.h"

#define COUNT 25                      // @ 8MHz INTFRC = 62.5 us.
#define ETIME COUNT*2.5                // time in us
#define DELAY for(i=0;i<COUNT;i++)    //for unsigned conversion 10 sig bits
#define ADSCALE 1023                  //Vdd connected to A/D Vr+
#define ADREF 3.3                     //R value is 4200000 (4.2M)
#define RCAL .027                    //scaled so that result is in
                                    //1/100th of uA

int main(void)
{
    int i;
    int j = 0;                      //index for loop
    unsigned int Vread = 0;
    float CTMUISrc, CTMUCap, Vavg, VTot, Vcal;

    //assume CTMU and A/D have been setup correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONHbits.CTMUEN = 1;          //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;      //drain charge on the circuit
        DELAY;                        //wait 125us
        CTMUCONHbits.IDISSEN = 0;      //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;      //Begin charging the circuit
        //using CTMU current source
        DELAY;                        //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;      //Stop charging circuit

        PIR1bits.ADIF = 0;             //make sure A/D Int not set
        ADCON0bits.GO=1;              //and begin A/D conv.
        while(!PIR1bits.ADIF);         //Wait for A/D convert complete

        Vread = ADRES;                //Get the value from the A/D
        PIR1bits.ADIF = 0;             //Clear A/D Interrupt Flag
        VTot += Vread;                //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);       //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF); //CTMUISrc is in 1/100ths of uA
    CTMUCap = (CTMUISrc*ETIME/Vcal)/100;
}
```

# PIC18F66K80 FAMILY

---

## 18.5 Measuring Capacitance with the CTMU

There are two ways to measure capacitance with the CTMU. The absolute method measures the actual capacitance value. The relative method only measures for any change in the capacitance.

### 18.5.1 ABSOLUTE CAPACITANCE MEASUREMENT

For absolute capacitance measurements, both the current and capacitance calibration steps found in [Section 18.4 “Calibrating the CTMU Module”](#) should be followed.

To perform these measurements:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Set EDG1STAT.
4. Wait for a fixed delay, T.
5. Clear EDG1STAT.
6. Perform an A/D conversion.
7. Calculate the total capacitance,  $C_{TOTAL} = (I * T)/V$ , where:
  - I is known from the current source measurement step ([Section 18.4.1 “Current Source Calibration”](#))
  - T is a fixed delay
  - V is measured by performing an A/D conversion
8. Subtract the stray and A/D capacitance (COFFSET from [Section 18.4.2 “Capacitance Calibration”](#)) from CTOTAL to determine the measured capacitance.

### 18.5.2 CAPACITIVE TOUCH SENSE USING RELATIVE CHARGE MEASUREMENT

Not all applications require precise capacitance measurements. When detecting a valid press of a capacitance-based switch, only a relative change of capacitance needs to be detected.

In such an application, when the switch is open (or not touched), the total capacitance is the capacitance of the combination of the board traces, the A/D Converter and other elements. A larger voltage will be measured by the A/D Converter. When the switch is closed (or touched), the total capacitance is larger due to the addition of the capacitance of the human body to the above listed capacitances and a smaller voltage will be measured by the A/D Converter.

To detect capacitance changes simply:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT.
3. Wait for a fixed delay.
4. Clear EDG1STAT.
5. Perform an A/D conversion.

The voltage measured by performing the A/D conversion is an indication of the relative capacitance. In this case, no calibration of the current source or circuit capacitance measurement is needed. (For a sample software routine for a capacitive touch switch, see [Example 18-4](#).)

## EXAMPLE 18-4: ROUTINE FOR CAPACITIVE TOUCH SWITCH

```
#include "p18cxx.h"

#define COUNT 500                      // @ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define OPENSW 1000                     // Un-pressed switch value
#define TRIP 300                        // Difference between pressed
                                         // and un-pressed switch
#define HYST 65                         // amount to change
                                         // from pressed to un-pressed
#define PRESSED 1
#define UNPRESSED 0

int main(void)
{
    unsigned int Vread;                // storage for reading
    unsigned int switchState;
    int i;

    //assume CTMU and A/D have been setup correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONHbits.CTMUEN = 1;          // Enable the CTMU

    CTMUCONHbits.IDISSEN = 1;          // drain charge on the circuit
    DELAY;
    CTMUCONHbits.IDISSEN = 0;          // end drain of circuit

    CTMUCONLbits.EDG1STAT = 1;         // Begin charging the circuit
                                         // using CTMU current source
    DELAY;
    CTMUCONLbits.EDG1STAT = 0;          // Stop charging circuit

    PIR1bits.ADIF = 0;                // make sure A/D Int not set
    ADCON0bits.GO=1;                  // and begin A/D conv.
    while(!PIR1bits.ADIF);            // Wait for A/D convert complete

    Vread = ADRES;                   // Get the value from the A/D

    if(Vread < OPENSW - TRIP)
    {
        switchState = PRESSED;
    }
    else if(Vread > OPENSW - TRIP + HYST)
    {
        switchState = UNPRESSED;
    }
}
```

# PIC18F66K80 FAMILY

## 18.6 Measuring Time with the CTMU Module

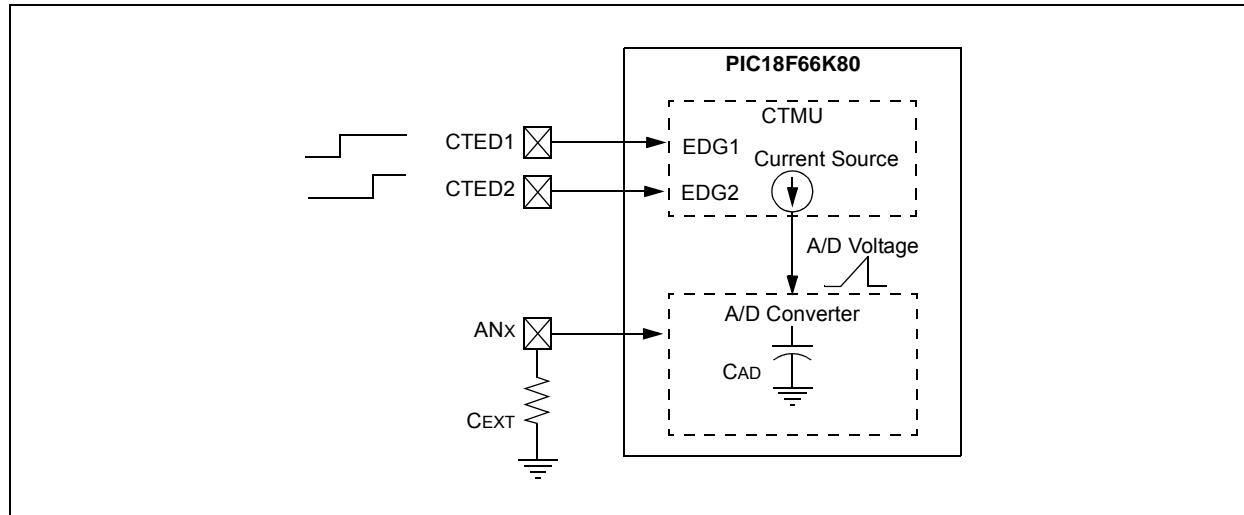
Time can be precisely measured after the ratio (C/I) is measured from the current and capacitance calibration step. To do that:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT.
3. Set EDG2STAT.
4. Perform an A/D conversion.
5. Calculate the time between edges as  $T = (C/I) * V$ , where:
  - I is calculated in the current calibration step ([Section 18.4.1 “Current Source Calibration”](#))
  - C is calculated in the capacitance calibration step ([Section 18.4.2 “Capacitance Calibration”](#))
  - V is measured by performing the A/D conversion

It is assumed that the time measured is small enough that the capacitance, CAD + CEXT, provides a valid voltage to the A/D Converter. For the smallest time measurement, always set the A/D Channel Select bits CHS<4:0> (ADCON0<6:2>) to an unused A/D channel, the corresponding pin for which is not connected to any circuit board trace. This minimizes added stray capacitance, keeping the total circuit capacitance close to that of the A/D Converter itself (25 pF).

To measure longer time intervals, an external capacitor may be connected to an A/D channel and that channel selected whenever making a time measurement.

**FIGURE 18-3: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR TIME MEASUREMENT**



## 18.7 Measuring Temperature with the CTMU

The constant current source provided by the CTMU module can be used for low-cost temperature measurement by exploiting a basic property of common and inexpensive diodes. An on-chip temperature sense diode is provided on A/D Channel 29 to further simplify design and cost.

### 18.7.1 BASIC PRINCIPAL

We can show that the forward voltage ( $V_F$ ) of a P-N junction, such as a diode, is an extension of the equation for the junction's thermal voltage:

$$V_F = \frac{kT}{q} \ln \left( 1 - \frac{I_F}{I_S} \right)$$

where  $k$  is the Boltzmann constant ( $1.38 \times 10^{-23} \text{ J K}^{-1}$ ),  $T$  is the absolute junction temperature in kelvin,  $q$  is the electron charge ( $1.6 \times 10^{-19} \text{ C}$ ),  $I_F$  is the forward current applied to the diode and  $I_S$  is the diode's characteristic saturation current, which varies between devices.

Since  $k$  and  $q$  are physical constants, and  $I_S$  is a constant for the device, this only leaves  $T$  and  $I_F$  as independent variables. If  $I_F$  is held constant, it follows from the equation that  $V_F$  will vary as a function of  $T$ . As the natural log term of the equation will always be negative, the temperature will be negatively proportional to  $V_F$ . In other words, as temperature increases,  $V_F$  decreases.

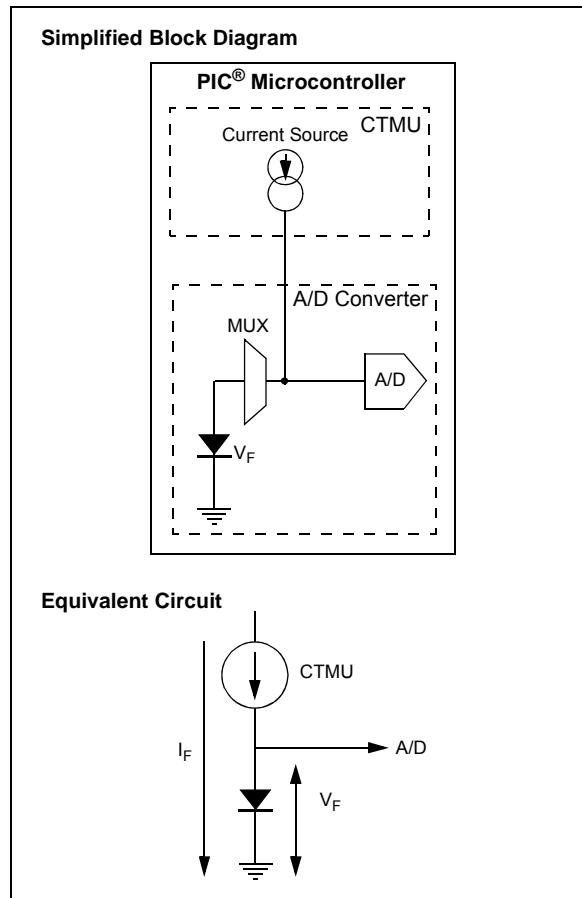
By using the CTMU's current source to provide a constant  $I_F$ , it becomes possible to calculate the temperature by measuring the  $V_F$  across the diode.

### 18.7.2 IMPLEMENTATION

To implement this theory, all that is needed is to connect a regular junction diode to one of the microcontroller's A/D pins (Figure 18-2). The A/D channel multiplexer is shared by the CTMU and the A/D.

To perform a measurement, the multiplexer is configured to select the pin connected to the diode. The CTMU current source is then turned on and an A/D conversion is performed on the channel. As shown in the equivalent circuit diagram, the diode is driven by the CTMU at  $I_F$ . The resulting  $V_F$  across the diode is measured by the A/D. A code snippet is shown in Example 18-5.

**FIGURE 18-4: CTMU TEMPERATURE MEASUREMENT CIRCUIT**



### EXAMPLE 18-5: ROUTINE FOR TEMPERATURE MEASUREMENT USING INTERNAL DIODE

```

// Initialize CTMU
CTMUICON = 0x03;
CTMUCONHbits.CTMUEN = 1;
CTMUCONLbits.EDG1STAT = 1;

// Initialize ADC
ADCON0 = 0x75; // Enable ADC and connect to Internal diode
ADCON1 = 0x00;
ADCON2 = 0xBE; // Right Justified

ADCON0bits.GO = 1; // Start conversion
while(ADCON0bits.GO); // Read ADC results (inversely proportional to temperature)
Temp = ADRES;

```

**Note:** The temperature diode is not calibrated or standardized; the user must calibrate the diode to their application.

# PIC18F66K80 FAMILY

## 18.8 Creating a Delay with the CTMU Module

A unique feature on board the CTMU module is its ability to generate system clock independent output pulses based on either an external voltage or an external capacitor value. When using an external voltage, this is accomplished using the CTDIN input pin as a trigger for the pulse delay. When using an external capacitor value, this is accomplished using the internal comparator voltage reference module and Comparator 2 input pin. The pulse is output onto the CTPLS pin. To enable this mode, set the TGEN bit.

See [Figure 18-5](#) for an example circuit. When CTMUDS (PADC<sub>FG1<0></sub>) is cleared, the pulse delay is determined by the output of Comparator 2, and when it is set, the pulse delay is determined by the input of CTDIN. CDELAY is chosen by the user to determine the output pulse width on CTPLS. The pulse width is calculated by  $T = (CDELAY/I) \cdot V$ , where I is known from the current source measurement step ([Section 18.4.1 "Current Source Calibration"](#)) and V is the internal reference voltage (CVREF).

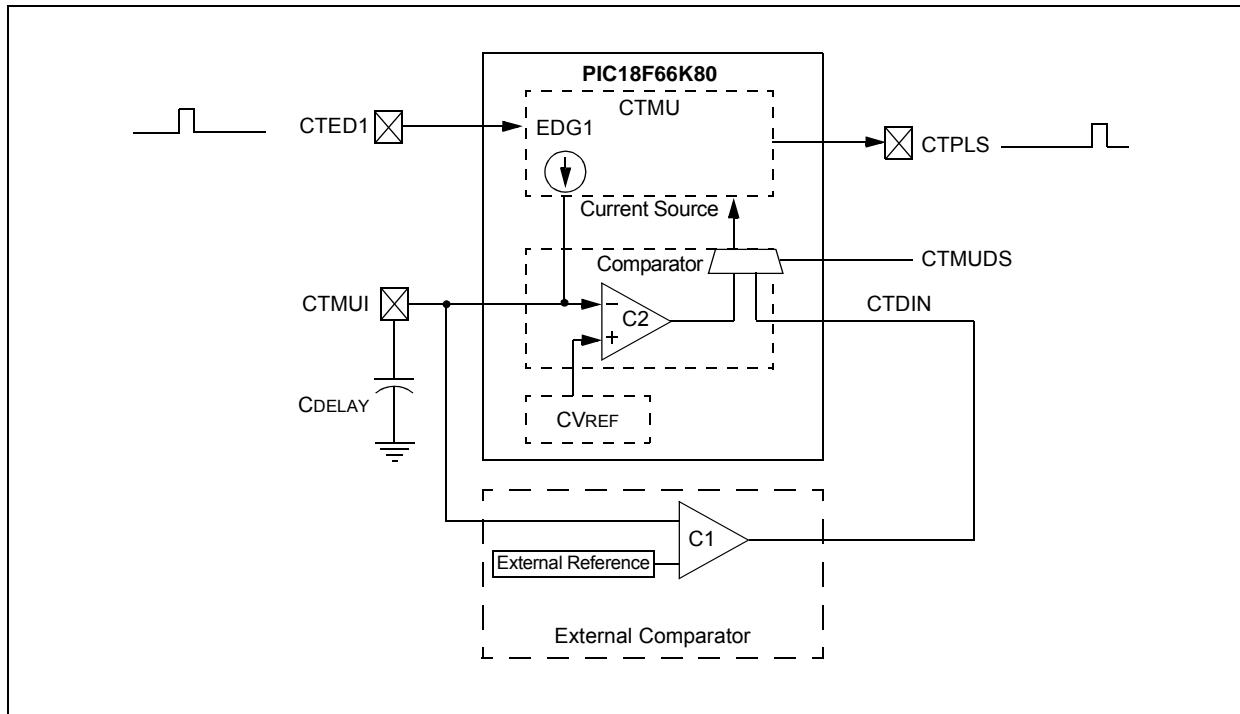
An example use of the external capacitor feature is interfacing with variable capacitive-based sensors, such as a humidity sensor. As the humidity varies, the pulse-width output on CTPLS will vary. An example use of the CTDIN feature is interfacing with a digital sensor. The CTPLS output pin can be connected to an input capture pin and the varying pulse width measured to determine the sensor's output in the application.

To use this feature:

1. If CTMUDS is cleared, initialize Comparator 2.
2. If CTMUDS is cleared, initialize the comparator voltage reference.
3. Initialize the CTMU and enable time delay generation by setting the TGEN bit.
4. Set EDG1STAT.

When CTMUDS is cleared, as soon as CDELAY charges to the value of the voltage reference trip point, an output pulse is generated on CTPLS. When CTMUDS is set, as soon as CTDIN is set, an output pulse is generated on CTPLS.

**FIGURE 18-5: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR PULSE DELAY GENERATION**



## 18.9 Measuring Temperature with the CTMU Module

The CTMU, along with an internal diode, can be used to measure the temperature. The A/D can be connected to the internal diode and the CTMU module can

source the current to the diode. The A/D reading will reflect the temperature. With the increase, the A/D readings will go low. This can be used for low-cost temperature measurement applications.

### EXAMPLE 18-6: ROUTINE FOR TEMPERATURE MEASUREMENT USING INTERNAL DIODE

```
// Initialize CTMU
CTMUICON = 0x03;
CTMUCONHbits.CTMUEN = 1;
CTMUCONLbits.EDG1STAT = 1;

// Initialize ADC
ADCON0 = 0xE5;                                // Enable ADC and connect to Internal diode
ADCON1 = 0x00;
ADCON2 = 0xBE;                                  //Right Justified

ADCON0bits.GO = 1;                             // Start conversion
while(ADCON0bits.GO);
Temp = ADRES;                                 // Read ADC results (inversely proportional to temperature)
```

**Note:** The temperature diode is not calibrated or standardized; the user must calibrate the diode to their application.

# PIC18F66K80 FAMILY

---

## 18.10 Operation During Sleep/Idle Modes

### 18.10.1 SLEEP MODE

When the device enters any Sleep mode, the CTMU module current source is always disabled. If the CTMU is performing an operation that depends on the current source when Sleep mode is invoked, the operation may not terminate correctly. Capacitance and time measurements may return erroneous values.

### 18.10.2 IDLE MODE

The behavior of the CTMU in Idle mode is determined by the CTMUSIDL bit (CTMUCONH<5>). If CTMUSIDL is cleared, the module will continue to operate in Idle mode. If CTMUSIDL is set, the module's current source is disabled when the device enters Idle mode. In this

case, if the module is performing an operation when Idle mode is invoked, the results will be similar to those with Sleep mode.

## 18.11 Effects of a Reset on CTMU

Upon Reset, all registers of the CTMU are cleared. This disables the CTMU module, turns off its current source and returns all configuration options to their default settings. The module needs to be re-initialized following any Reset.

If the CTMU is in the process of taking a measurement at the time of Reset, the measurement will be lost. A partial charge may exist on the circuit that was being measured, which should be properly discharged before the CTMU makes subsequent attempts to make a measurement. The circuit is discharged by setting and clearing the IDISSEN bit (CTMUCONH<1>) while the A/D Converter is connected to the appropriate channel.

**TABLE 18-1: REGISTERS ASSOCIATED WITH CTMU MODULE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CTMUCONH	CTMUEEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG
CTMUCONL	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT
CTMUICON	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
PADCFG1	RDP <u>U</u>	REPU	RFP <u>U</u>	RGPU	—	—	—	CTMUDS

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during ECCP operation.

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

PIC18F66K80 family devices have four CCP (Capture/Compare/PWM) modules, designated CCP2 through CCP5. All the modules implement standard Capture, Compare and Pulse-Width Modulation (PWM) modes.

**Note:** Throughout this section, generic references are used for register and bit names that are the same, except for an 'x' variable that indicates the item's association with the specific CCP module. For example, the control register is named CCPxCON and refers to CCP2CON through CCP5CON.

Each CCP module contains a 16-bit register that can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register. For the sake of clarity, all CCP module operation in the following sections is described with respect to CCP2, but is equally applicable to CCP3 through CCP5.

### REGISTER 19-1: CCPxCON: CCPx CONTROL REGISTER (CCP2-CCP5 MODULES)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3 <sup>(1)</sup>	CCPxM2 <sup>(1)</sup>	CCPxM1 <sup>(1)</sup>	CCPxM0 <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	<b>Unimplemented:</b> Read as '0'
bit 5-4	<b>DCxB&lt;1:0&gt;:</b> PWM Duty Cycle bit 1 and bit 0 for CCPx Module bits <u>Capture mode:</u> Unused. <u>Compare mode:</u> Unused. <u>PWM mode:</u> These bits are the two Least Significant bits (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight Most Significant bits (DCx<9:2>) of the duty cycle are found in CCPRxL.
bit 3-0	<b>CCPxM&lt;3:0&gt;:</b> CCPx Module Mode Select bits <sup>(1)</sup> 0000 = Capture/Compare/PWM disabled (resets CCPx module) 0001 = Reserved 0010 = Compare mode: toggle output on match (CCPxIF bit is set) 0011 = Reserved 0100 = Capture mode: every falling edge or CAN message received (time-stamp) <sup>(2)</sup> 0101 = Capture mode: every rising edge or CAN message received (time-stamp) <sup>(2)</sup> 0110 = Capture mode: every 4th rising edge or on every fourth CAN message received (time-stamp) <sup>(2)</sup> 0111 = Capture mode: every 16th rising edge or on every 16th CAN message received (time-stamp) <sup>(2)</sup> 1000 = Compare mode: initialize CCPx pin low; on compare match, force CCPx pin high (CCPxIF bit is set) 1001 = Compare mode: initialize CCPx pin high; on compare match, force CCPx pin low (CCPxIF bit is set) 1010 = Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state) 1011 = Compare mode: Special Event Trigger; reset timer on CCPx match (CCPxIF bit is set) 11xx = PWM mode

**Note 1:** CCPxM<3:0> = 1011 will only reset the timer and not start an A/D conversion on CCPx match.

**2:** Available only on CCP2. Selected by the CANCAP (CIOCON<4>) bit. Overrides the CCP2 input pin source.

# PIC18F66K80 FAMILY

## REGISTER 19-2: CCPTMRS: CCP TIMER SELECT REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL
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      **C5TSEL:** CCP5 Timer Selection bit  
0 = CCP5 is based off of TMR1/TMR2  
1 = CCP5 is based off of TMR3/TMR4
- bit 3      **C4TSEL:** CCP4 Timer Selection bit  
0 = CCP4 is based off of TMR1/TMR2  
1 = CCP4 is based off of TMR3/TMR4
- bit 2      **C3TSEL:** CCP3 Timer Selection bit  
0 = CCP3 is based off of TMR1/TMR2  
1 = CCP3 is based off of TMR3/TMR4
- bit 1      **C2TSEL:** CCP2 Timer Selection bit  
0 = CCP2 is based off of TMR1/TMR2  
1 = CCP2 is based off of TMR3/TMR4
- bit 0      **C1TSEL:** CCP1 Timer Selection bit  
0 = ECCP1 is based off of TMR1/TMR2  
1 = ECCP1 is based off of TMR3/TMR4

# PIC18F66K80 FAMILY

## REGISTER 19-3: CCPRxL: CCPx PERIOD LOW BYTE REGISTER

| R/W-x   |
|---------|---------|---------|---------|---------|---------|---------|---------|
| CCPRxL7 | CCPRxL6 | CCPRxL5 | CCPRxL4 | CCPRxL3 | CCPRxL2 | CCPRxL1 | CCPRxL0 |
| 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 7-0      **CCPRxL<7:0>**: CCPx Period Register Low Byte bits

Capture Mode: Capture register low byte

Compare Mode: Compare register low byte

PWM Mode: Duty Cycle Buffer register

## REGISTER 19-4: CCPRxH: CCPx PERIOD HIGH BYTE REGISTER

| R/W-x   |
|---------|---------|---------|---------|---------|---------|---------|---------|
| CCPRxH7 | CCPRxH6 | CCPRxH5 | CCPRxH4 | CCPRxH3 | CCPRxH2 | CCPRxH1 | CCPRxH0 |
| 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 7-0      **CCPRxH<7:0>**: CCPx Period Register High Byte bits

Capture Mode: Capture register high byte

Compare Mode: Compare register high byte

PWM Mode: Duty Cycle Buffer register

# PIC18F66K80 FAMILY

## 19.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (generically, CCP<sub>x</sub>CON) and a data register (CCPR<sub>x</sub>). The data register, in turn, is comprised of two 8-bit registers: CCPR<sub>xL</sub> (low byte) and CCPR<sub>xH</sub> (high byte). All registers are both readable and writable.

### 19.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers, 1 through 4, varying with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in [Table 19-1](#).

**TABLE 19-1: CCP MODE – TIMER RESOURCE**

CCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	
PWM	Timer2 or Timer4

The assignment of a particular timer to a module is determined by the Timer to CCP enable bits in the CCPTMRS register (see [Register 19-2](#)). All of the modules may be active at once and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time.

The CCPTMRS register selects the timers for CCP modules, 2, 3, 4 and 5. The possible configurations are shown in [Table 19-2](#).

**TABLE 19-2: TIMER ASSIGNMENTS FOR CCP MODULES 2, 3, 4 AND 5**

CCPTMRS Register												
CCP2			CCP3			CCP4			CCP5			
C2TSEL	Capture/ Compare Mode	PWM Mode	C3TSEL	Capture/ Compare Mode	PWM Mode	C4TSEL	Capture/ Compare Mode	PWM Mode	C5TSEL	Capture/ Compare Mode	PWM Mode	
0	TMR1	TMR2	0	TMR1	TMR2	0	TMR1	TMR2	0 0	TMR1	TMR2	
1	TMR3	TMR4	1	TMR3	TMR4	1	TMR3	TMR4	0 1	TMR3	TMR4	

### 19.1.2 OPEN-DRAIN OUTPUT OPTION

When operating in Output mode (the Compare or PWM modes), the drivers for the CCP<sub>x</sub> pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor and allows the output to communicate with external circuits without the need for additional level shifters.

The open-drain output option is controlled by the CCP<sub>x</sub>OD bits (ODCON<6:2>). Setting the appropriate bit configures the pin for the corresponding module for open-drain operation.

## 19.2 Capture Mode

In Capture mode, the CCPRxH:CCPRxL register pair captures the 16-bit value of the Timer register selected in the CCPTMRS when an event occurs on the CCPx pin. An event is defined as one of the following:

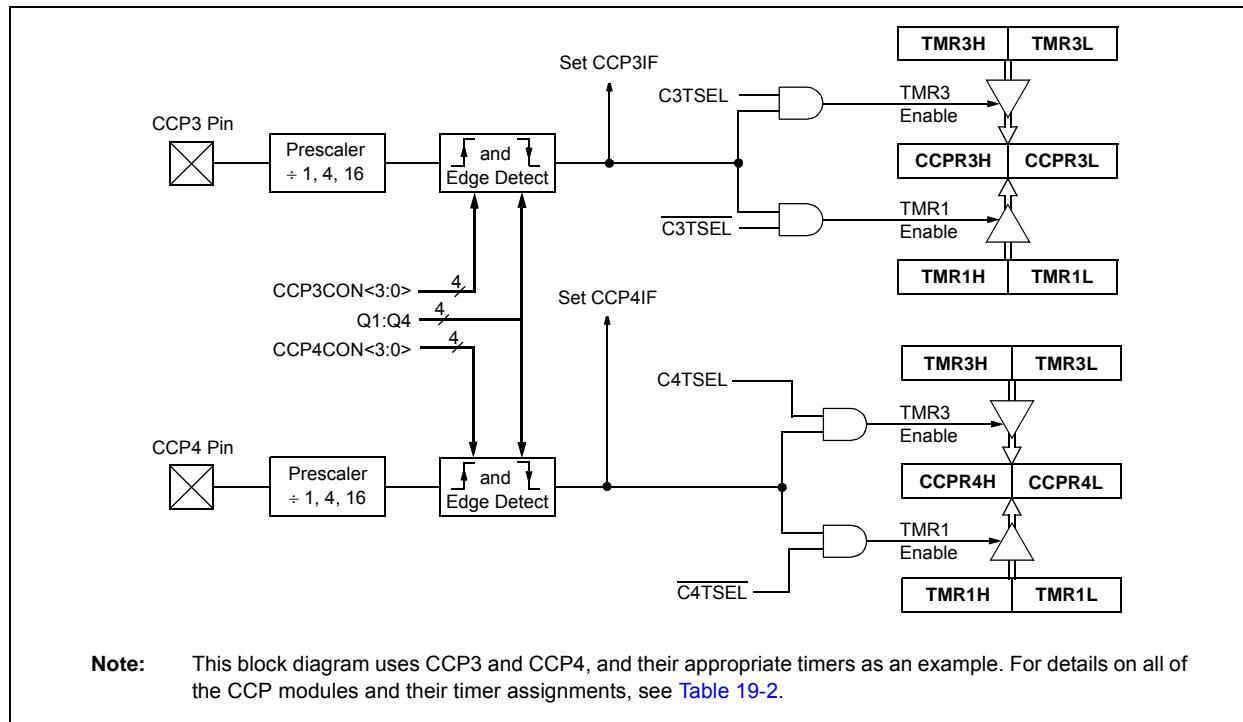
- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

**Note:** For CCP2 only, the Capture mode can use the CCP2 input pin as the capture trigger for CCP2 or the input can function as a time-stamp through the CAN module. The CAN module provides the necessary control and trigger signals.

The event is selected by the mode select bits, CCPxM<3:0> (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF (PIR4<x>), is set; it must be cleared in software. If another capture occurs before the value in CCPRx is read, the old captured value is overwritten by the new captured value.

Figure 19-1 shows the Capture mode block diagram.

**FIGURE 19-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

---

---

## 19.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE bit (PIE4<x>) clear to avoid false interrupts and should clear the flag bit, CCPxIF, following any such change in operating mode.

## 19.2.4 CCP PRESCALER

There are four prescaler settings in Capture mode. They are specified as part of the operating mode selected by the mode select bits (CCPxM<3:0>). Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Doing that also will not clear the prescaler counter – meaning the first capture may be from a non-zero prescaler.

**Example 19-1** shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

### EXAMPLE 19-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF  CCPxCON    ; Turn CCP module off
MOVLW NEW_CAPT_PS ; Load WREG with the
                   ; new prescaler mode
                   ; value and CCP ON
MOVWF CCPxCON     ; Load CCPxCON with
                   ; this value
```

## 19.2.5 CAN MESSAGE TIME-STAMP (CCP2 ONLY)

For CCP2, only the CAN capture event occurs when a message is received in any of the receive buffers. When configured, the CAN module provides the trigger to the CCP2 module to cause a capture event. This feature is provided to “time-stamp” the received CAN messages.

This feature is enabled by setting the CANCAP bit of the CAN I/O Control register (CIOCON<4>). The message receive signal from the CAN module then takes the place of the events on RC2/CCP2.

If this feature is selected, then four different capture options for CCP2M<3:0> are available:

- 0100 – Every time a CAN message is received
- 0101 – Every time a CAN message is received
- 0110 – Every 4th time a CAN message is received
- 0111 – Capture mode, every 16th time a CAN message is received

## 19.3 Compare Mode

In Compare mode, the 16-bit CCPRx register value is constantly compared against the Timer register pair value selected in the CCPTMR register. When a match occurs, the CCPx pin can be:

- Driven high
- Driven low
- Toggled (high-to-low or low-to-high)
- Unchanged (that is, reflecting the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCPxM<3:0>). At the same time, the interrupt flag bit, CCPxIF, is set.

Figure 19-2 gives the Compare mode block diagram

### 19.3.1 CCP PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the appropriate TRIS bit.

**Note:** Clearing the CCPxCON register will force the corresponding CCPx compare output latch (depending on device configuration) to the default low level. This is not the PORTx data latch.

### 19.3.2 TIMER1/3 MODE SELECTION

If the CCPx module is using the compare feature in conjunction with any of the Timer1/3 timers, the timers must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the compare operation may not work.

**Note:** Details of the timer assignments for the CCPx modules are given in [Table 19-2](#).

### 19.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCPxM<3:0> = 1010), the CCPx pin is not affected. Only a CCP interrupt is generated, if enabled, and the CCPxIE bit is set.

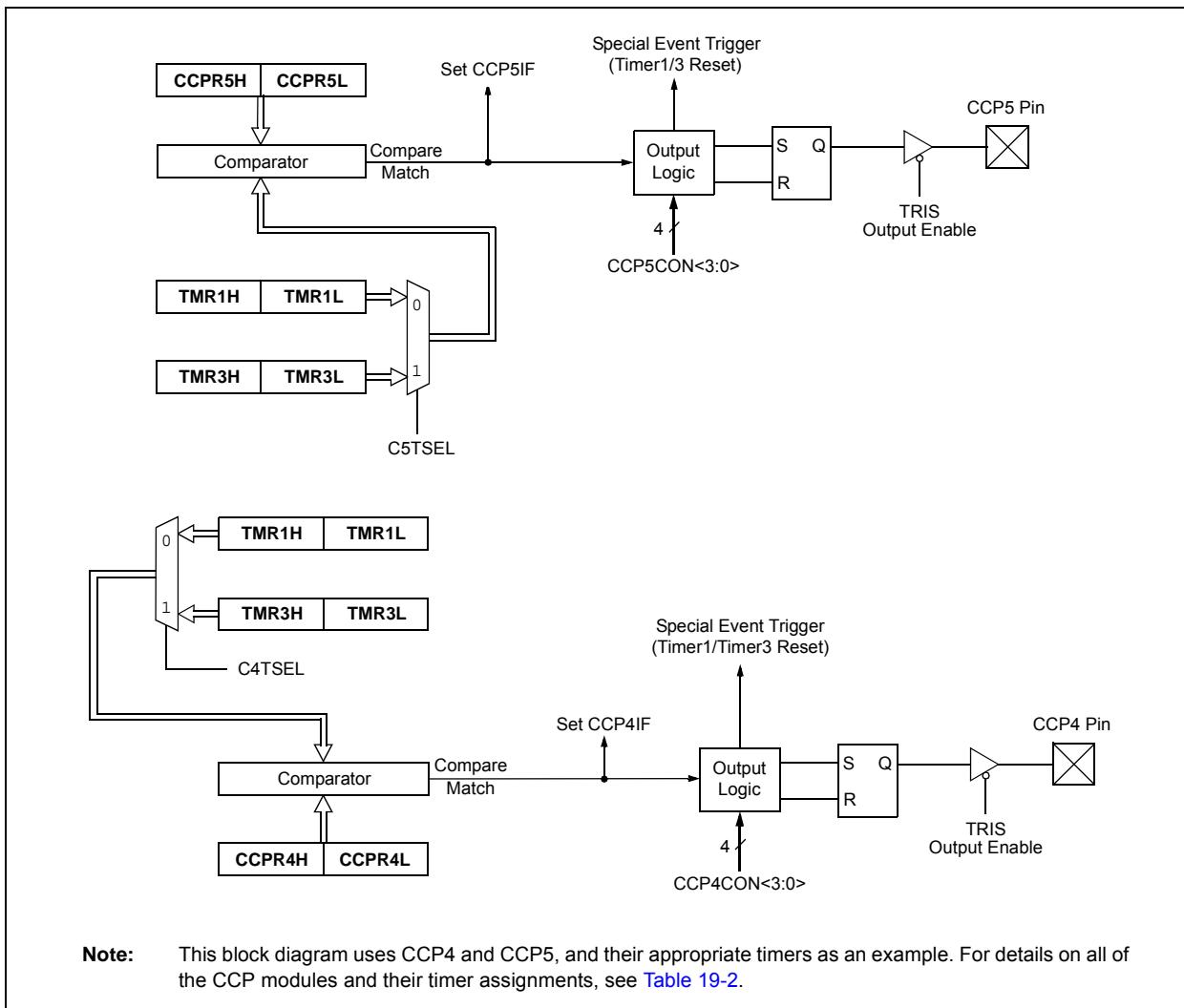
### 19.3.4 SPECIAL EVENT TRIGGER

All CCP modules are equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode bits (CCPxM<3:0> = 1011).

For either CCPx module, the Special Event Trigger resets the Timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCPRx registers to serve as a programmable Period register for either timer.

# PIC18F66K80 FAMILY

**FIGURE 19-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

**TABLE 19-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1/3**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
RCON	IPEN	SBOREN	CM	RI	TO	PD	POR	BOR
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
TMR1L	Timer1 Register Low Byte							
TMR1H	Timer1 Register High Byte							
TMR3L	Timer3 Register Low Byte							
TMR3H	Timer3 Register High Byte							
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	SOSCEN	T1SYNC	RD16	TMR1ON
T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	SOSCEN	T3SYNC	RD16	TMR3ON
CCPR2L	Capture/Compare/PWM Register 2 Low Byte							
CCPR2H	Capture/Compare/PWM Register 2 High Byte							
CCPR3L	Capture/Compare/PWM Register 3 Low Byte							
CCPR3H	Capture/Compare/PWM Register 3 High Byte							
CCPR4L	Capture/Compare/PWM Register 4 Low Byte							
CCPR4H	Capture/Compare/PWM Register 4 High Byte							
CCPR5L	Capture/Compare/PWM Register 5 Low Byte							
CCPR5H	Capture/Compare/PWM Register 5 High Byte							
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0
CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0
CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0
CCPTMRS	—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by Capture/Compare or Timer1/3.

# PIC18F66K80 FAMILY

## 19.4 PWM Mode

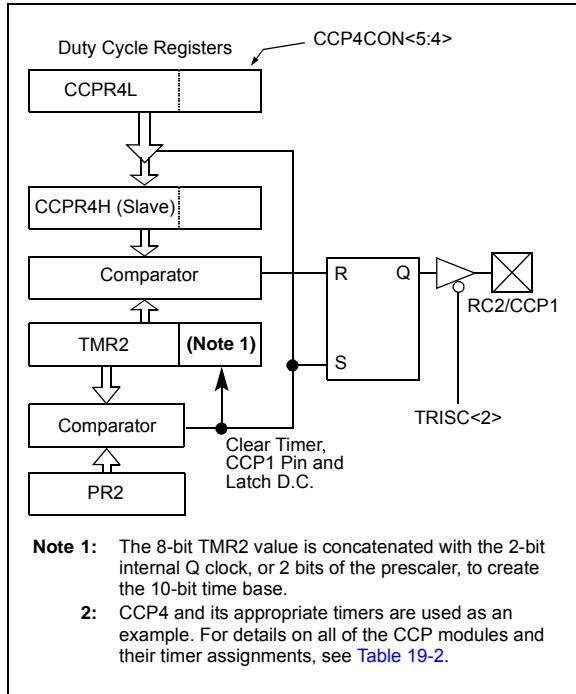
In Pulse-Width Modulation (PWM) mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCPx pin is multiplexed with a PORTC or PORTB data latch, the appropriate TRIS bit must be cleared to make the CCPx pin an output.

**Note:** Clearing the CCPxCON register will force the corresponding CCPx output latch (depending on device configuration) to the default low level. This is not the PORTx I/O data latch.

Figure 19-3 shows a simplified block diagram of the CCPx module in PWM mode.

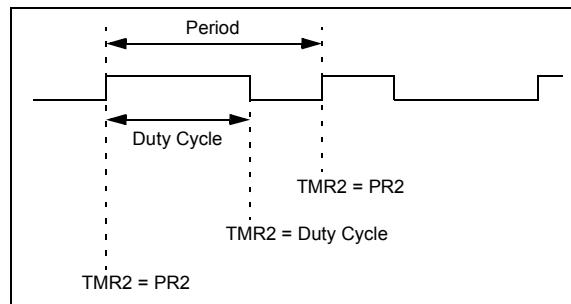
For a step-by-step procedure on how to set up the CCP module for PWM operation, see [Section 19.4.3 “Setup for PWM Operation”](#).

**FIGURE 19-3: SIMPLIFIED PWM BLOCK DIAGRAM**



A PWM output (Figure 19-4) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

**FIGURE 19-4: PWM OUTPUT**



### 19.4.1 PWM PERIOD

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

**EQUATION 19-1:**

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

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

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

- TMR2 is cleared
- The CCP4 pin is set  
(An exception: If PWM duty cycle = 0%, the CCP4 pin will not be set)
- The PWM duty cycle is latched from CCPR4L into CCPR4H

**Note:** The Timer2 postscalers (see [Section 15.0 “Timer2 Module”](#)) are not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

## 19.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified, to use CCP4 as an example, by writing to the CCP4L register and to the CCP4CON<5:4> bits. Up to 10-bit resolution is available. The CCP4L contains the eight MSbs and the CCP4CON<5:4> contains the two LSbs. This 10-bit value is represented by CCP4L:CCP4CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

### EQUATION 19-2:

$$\text{PWM Duty Cycle} = (\text{CCP4L:CCP4CON<5:4>} \cdot \text{TOSC} \cdot (\text{TMR2 Prescale Value}))$$

CCP4L and CCP4CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCP4H until after a match between PR2 and TMR2 occurs (that is, the period is complete). In PWM mode, CCP4H is a read-only register.

The CCP4H register and a two-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation.

When the CCP4H and two-bit latch match TMR2, concatenated with an internal two-bit Q clock or two bits of the TMR2 prescaler, the CCP4 pin is cleared.

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

### EQUATION 19-3:

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

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP4 pin will not be cleared.

**TABLE 19-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz**

PWM Frequency	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	6.58

## 19.4.3 SETUP FOR PWM OPERATION

To configure the CCP module for PWM operation, using CCP4 as an example:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCP4L register and CCP4CON<5:4> bits.

3. Make the CCP4 pin an output by clearing the appropriate TRIS bit.
4. Set the TMR2 prescale value, then enable Timer2 by writing to T2CON.
5. Configure the CCP4 module for PWM operation.

# PIC18F66K80 FAMILY

---

TABLE 19-5: REGISTERS ASSOCIATED WITH PWM AND TIMERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
RCON	IPEN	SBOREN	CM	RI	TO	PD	POR	BOR
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISCO
TMR2	Timer2 Register							
TMR4	Timer4 Register							
PR2	Timer2 Period Register							
PR4	Timer4 Period Register							
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
CCPR2L	Capture/Compare/PWM Register 2 Low Byte							
CCPR2H	Capture/Compare/PWM Register 2 High Byte							
CCPR3L	Capture/Compare/PWM Register 3 Low Byte							
CCPR3H	Capture/Compare/PWM Register 3 High Byte							
CCPR4L	Capture/Compare/PWM Register 4 Low Byte							
CCPR4H	Capture/Compare/PWM Register 4 High Byte							
CCPR5L	Capture/Compare/PWM Register 5 Low Byte							
CCPR5H	Capture/Compare/PWM Register 5 High Byte							
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0
CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0
CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0
CCPTMRS	—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PWM or Timer2/4.

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

PIC18F66K80 family devices have one Enhanced Capture/Compare/PWM (ECCP) module: ECCP1. These modules contain a 16-bit register, which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register. These ECCP modules are upward compatible with CCP. ECCP1 is implemented as standard CCP modules with enhanced PWM capabilities. These include:

- Provision for two or four output channels
- Output Steering modes
- Programmable polarity
- Programmable dead-band control
- Automatic shutdown and restart

The enhanced features are discussed in detail in [Section 20.4 “PWM \(Enhanced Mode\)”](#).

The ECCP1 module uses the control register, CCP1CON. The control registers, CCP2CON through CCP5CON, are for the modules, CCP2 through CCP5.

# PIC18F66K80 FAMILY

## REGISTER 20-1: CCP1CON: ENHANCED CAPTURE/COMPARE/PWM1 CONTROL

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
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      **P1M<1:0>**: Enhanced PWM Output Configuration bits  
If CCP1M<3:2> = 00, 01, 10:  
xx = P1A assigned as capture/compare input/output; P1B, P1C and P1D assigned as port pins  
If CCP1M<3:2> = 11:  
00 = Single output: P1A, P1B, P1C and P1D are controlled by steering (see [Section 20.4.7 "Pulse Steering Mode"](#))  
01 = Full-bridge output forward: P1D is modulated; P1A is active; P1B, P1C is inactive  
10 = Half-bridge output: P1A, P1B are modulated with dead-band control; P1C and P1D are assigned as port pins  
11 = Full-bridge output reverse: P1B is modulated; P1C is active; P1A and P1D are inactive
- bit 5-4      **DC1B<1:0>**: PWM Duty Cycle bit 1 and bit 0  
Capture mode:  
Unused.  
Compare mode:  
Unused.  
PWM mode:  
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.
- bit 3-0      **CCP1M<3:0>**: ECCP1 Mode Select bits  
0000 = Capture/Compare/PWM off (resets ECCP1 module)  
0001 = Reserved  
0010 = Compare mode: Toggle output on match  
0011 = Capture mode  
0100 = Capture mode: Every falling edge  
0101 = Capture mode: Every rising edge  
0110 = Capture mode: Every fourth rising edge  
0111 = Capture mode: Every 16<sup>th</sup> rising edge  
1000 = Compare mode: Initialize ECCP1 pin low, set output on compare match (set CCP1IF)  
1001 = Compare mode: Initialize ECCP1 pin high, clear output on compare match (set CCP1IF)  
1010 = Compare mode: Generate software interrupt only, ECCP1 pin reverts to I/O state  
1011 = Compare mode: Trigger special event (ECCP1 resets TMR1 or TMR3, starts A/D conversion, sets CCP1IF bit)  
1100 = PWM mode: P1A and P1C are active-high; P1B and P1D are active-high  
1101 = PWM mode: P1A and P1C are active-high; P1B and P1D are active-low  
1110 = PWM mode: P1A and P1C are active-low; P1B and P1D are active-high  
1111 = PWM mode: P1A and P1C are active-low; P1B and P1D are active-low

# PIC18F66K80 FAMILY

## REGISTER 20-2: CCPTMRS: CCP TIMER SELECT REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL
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	<b>Unimplemented:</b> Read as '0'
bit 4	<b>C5TSEL:</b> CCP5 Timer Selection bit 0 = CCP5 is based off of TMR1/TMR2 1 = CCP5 is based off of TMR3/TMR4
bit 3	<b>C4TSEL:</b> CCP4 Timer Selection bit 0 = CCP4 is based off of TMR1/TMR2 1 = CCP4 is based off of TMR3/TMR4
bit 2	<b>C3TSEL:</b> CCP3 Timer Selection bit 0 = CCP3 is based off of TMR1/TMR2 1 = CCP3 is based off of TMR3/TMR4
bit 1	<b>C2TSEL:</b> CCP2 Timer Selection bit 0 = CCP2 is based off of TMR1/TMR2 1 = CCP2 is based off of TMR3/TMR4
bit 0	<b>C1TSEL:</b> CCP1 Timer Selection bit 0 = ECCP1 is based off of TMR1/TMR2 1 = ECCP1 is based off of TMR3/TMR4

# PIC18F66K80 FAMILY

---

In addition to the expanded range of modes available through the CCP1CON and ECCP1AS registers, the ECCP module has two additional registers associated with Enhanced PWM operation and auto-shutdown features. They are:

- ECCP1DEL – Enhanced PWM Control
- PSTR1CON – Pulse Steering Control

## 20.1 ECCP Outputs and Configuration

The Enhanced CCP module may have up to four PWM outputs, depending on the selected operating mode. The CCP1CON register is modified to allow control over four PWM outputs: ECCP1/P1A, P1B, P1C and P1D. Applications can use one, two or four of these outputs.

The outputs that are active depend on the ECCP operating mode selected. The pin assignments are summarized in [Table 20-2](#).

To configure the I/O pins as PWM outputs, the proper PWM mode must be selected by setting the P1M<1:0> and CCP1M<3:0> bits. The appropriate TRIS direction bits for the port pins must also be set as outputs.

### 20.1.1 ECCP MODULE AND TIMER RESOURCES

The ECCP modules use Timers, 1, 2, 3 and 4, depending on the mode selected. These timers are available to CCP modules in Capture, Compare or PWM modes, as shown in [Table 20-1](#).

**TABLE 20-1: ECCP MODE – TIMER RESOURCE**

ECCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2 or Timer4

The assignment of a particular timer to a module is determined by the Timer to ECCP enable bits in the CCPTMRS register ([Register 20-2](#)). The interactions between the two modules are depicted in [Figure 20-1](#). Capture operations are designed to be used when the timer is configured for Synchronous Counter mode. Capture operations may not work as expected if the associated timer is configured for Asynchronous Counter mode.

## 20.2 Capture Mode

In Capture mode, the CCP1H:CCP1L register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the corresponding ECCP1 pin. An event is defined as one of the following:

- Every falling edge
- Every rising edge
- Every fourth rising edge
- Every 16<sup>th</sup> rising edge

The event is selected by the mode select bits, CCP1M<3:0> (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit, CCP1IF, is set (PIR3<1>). The flag must be cleared by software. If another capture occurs before the value in the CCP1H/L register is read, the old captured value is overwritten by the new captured value.

### 20.2.1 ECCP PIN CONFIGURATION

In Capture mode, the appropriate ECCP1 pin should be configured as an input by setting the corresponding TRIS direction bit.

**Note:** If the ECCP1 pin is configured as an output, a write to the port can cause a capture condition.

## 20.2.2 TIMER1/2/3/4 MODE SELECTION

The timers that are to be used with the capture feature (Timer1 2, 3 or 4) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer to be used with each CCP module is selected in the CCPTMRS register ([Register 20-2](#)).

## 20.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCP1IE interrupt enable bit clear to avoid false interrupts. The interrupt flag bit, CCP1IF, should also be cleared following any such change in operating mode.

## 20.2.4 ECCP PRESCALER

There are four prescaler settings in Capture mode; they are specified as part of the operating mode selected by the mode select bits (CCP1M<3:0>). Whenever the CCP module is turned off, or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. [Example 20-1](#) provides the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

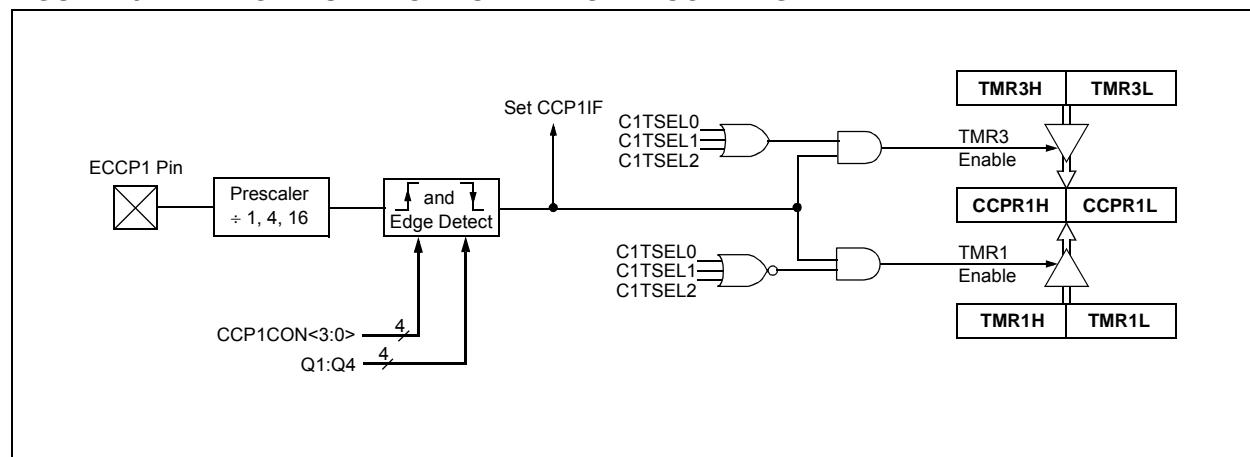
### EXAMPLE 20-1: CHANGING BETWEEN CAPTURE PRESCALERS

```

CLRF  CCP1CON      ; Turn CCP module off
MOVLW NEW_CAPT_PS ; Load WREG with the
; new prescaler mode
; value and CCP ON
MOVWF CCP1CON     ; Load CCP1CON with
; this value

```

**FIGURE 20-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

## 20.3 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against the Timer register pair value selected in the CCPTMR1 register. When a match occurs, the ECCP1 pin can be:

- Driven high
- Driven low
- Toggled (high-to-low or low-to-high)
- Unchanged (that is, reflecting the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCP1M<3:0>). At the same time, the interrupt flag bit, CCP1IF, is set.

### 20.3.1 ECCP PIN CONFIGURATION

Users must configure the ECCP1 pin as an output by clearing the appropriate TRIS bit.

**Note:** Clearing the CCP1CON register will force the ECCP1 compare output latch (depending on device configuration) to the default low level. This is not the port I/O data latch.

### 20.3.2 TIMER1/2/3/4 MODE SELECTION

Timer1, 2, 3 or 4 must be running in Timer mode or Synchronized Counter mode if the ECCP module is using the compare feature. In Asynchronous Counter mode, the compare operation will not work reliably.

### 20.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCP1M<3:0> = 1010), the ECCP1 pin is not affected; only the CCP1IF interrupt flag is affected.

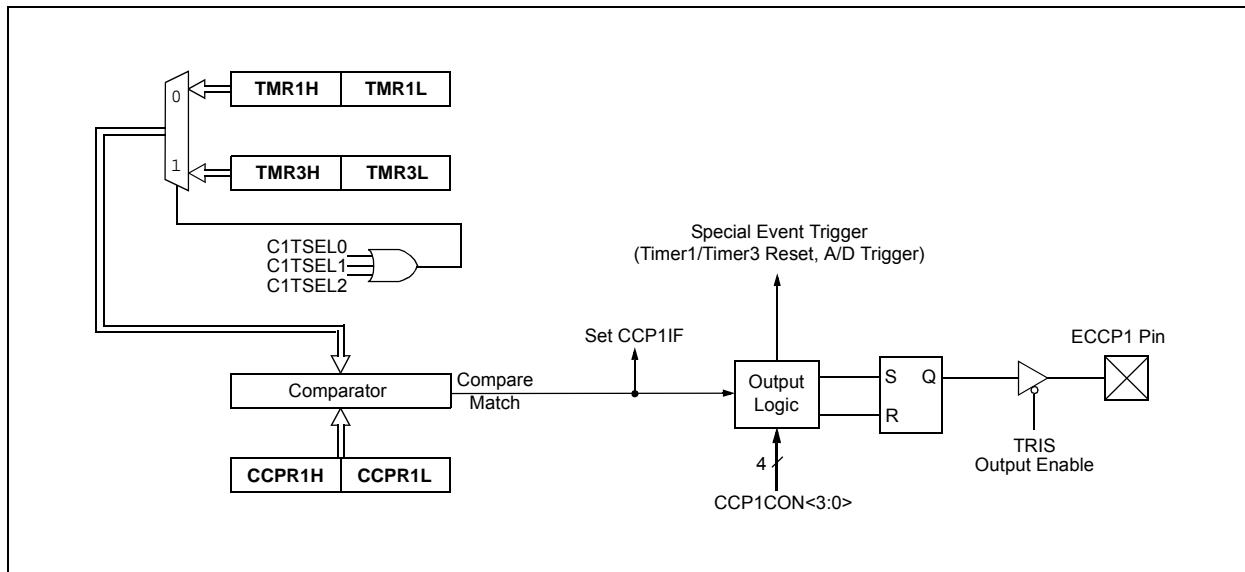
### 20.3.4 SPECIAL EVENT TRIGGER

The ECCP module is equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCP1M<3:0> = 1011).

The Special Event Trigger resets the Timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCPR1 registers to serve as a programmable Period register for either timer.

The Special Event Trigger can also start an A/D conversion. In order to do this, the A/D Converter must already be enabled.

FIGURE 20-2: COMPARE MODE OPERATION BLOCK DIAGRAM



## 20.4 PWM (Enhanced Mode)

The Enhanced PWM mode can generate a PWM signal on up to four different output pins with up to 10 bits of resolution. It can do this through four different PWM Output modes:

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

To select an Enhanced PWM mode, the P1M bits of the CCP1CON register must be set appropriately.

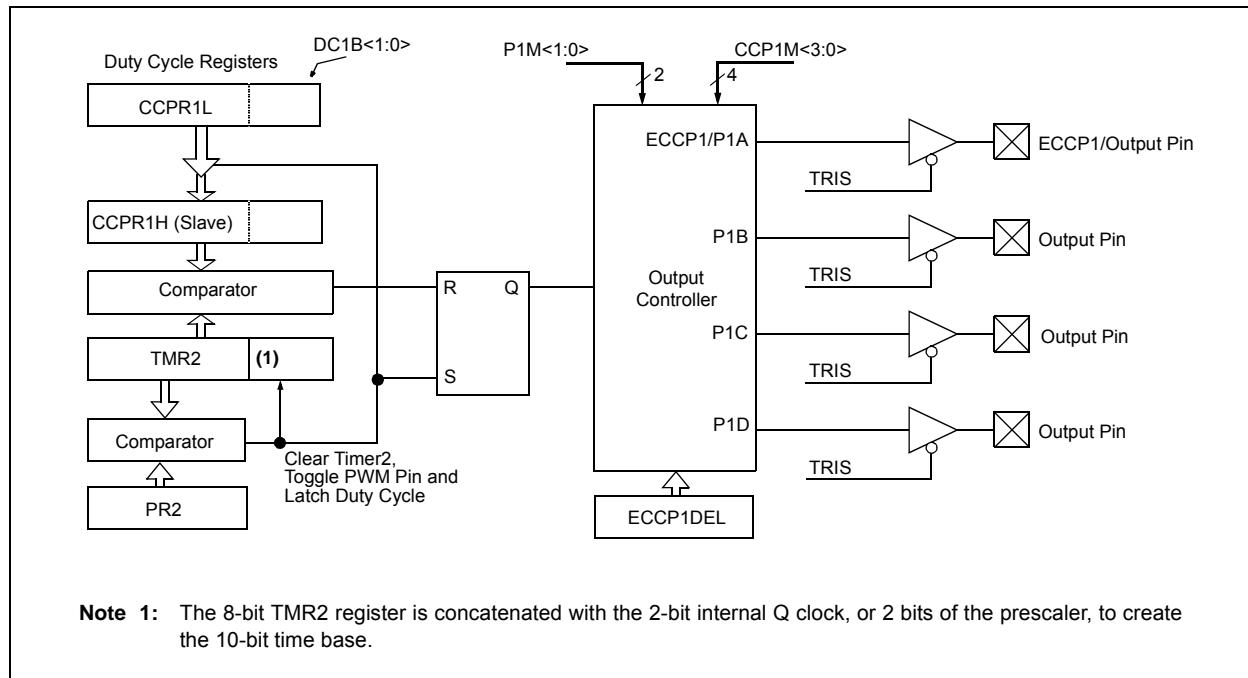
The PWM outputs are multiplexed with I/O pins and are designated: P1A, P1B, P1C and P1D. The polarity of the PWM pins is configurable and is selected by setting the CCP1M bits in the CCP1CON register appropriately.

Table 20-1 provides the pin assignments for each Enhanced PWM mode.

Figure 20-3 provides an example of a simplified block diagram of the Enhanced PWM module.

**Note:** To prevent the generation of an incomplete waveform when the PWM is first enabled, the ECCP module waits until the start of a new PWM period before generating a PWM signal.

**FIGURE 20-3: EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE**



**Note 1:** The TRIS register value for each PWM output must be configured appropriately.

**2:** Any pin not used by an Enhanced PWM mode is available for alternate pin functions.

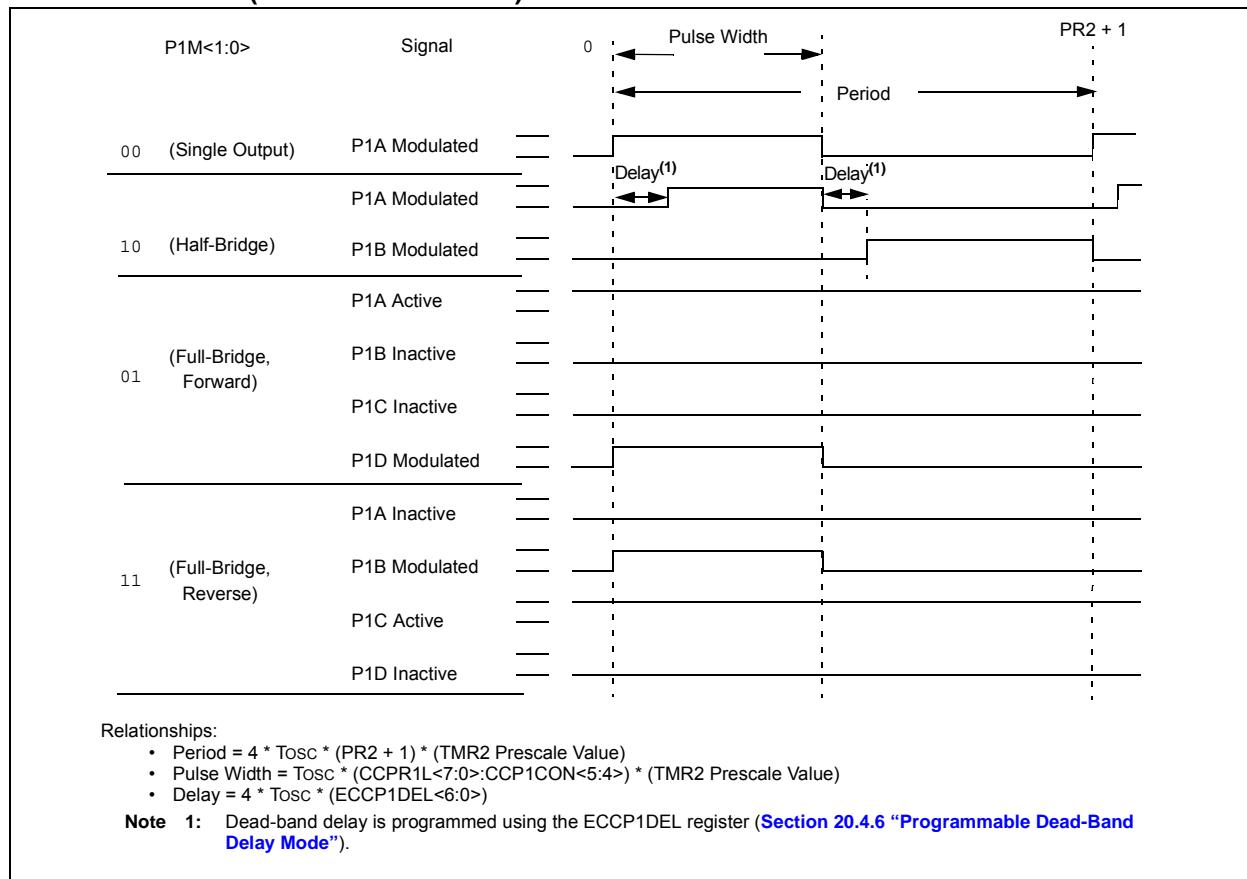
# PIC18F66K80 FAMILY

TABLE 20-2: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES

ECCP Mode	P1M<1:0>	P1A	P1B	P1C	P1D
Single	00	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
Half-Bridge	10	Yes	Yes	No	No
Full-Bridge, Forward	01	Yes	Yes	Yes	Yes
Full-Bridge, Reverse	11	Yes	Yes	Yes	Yes

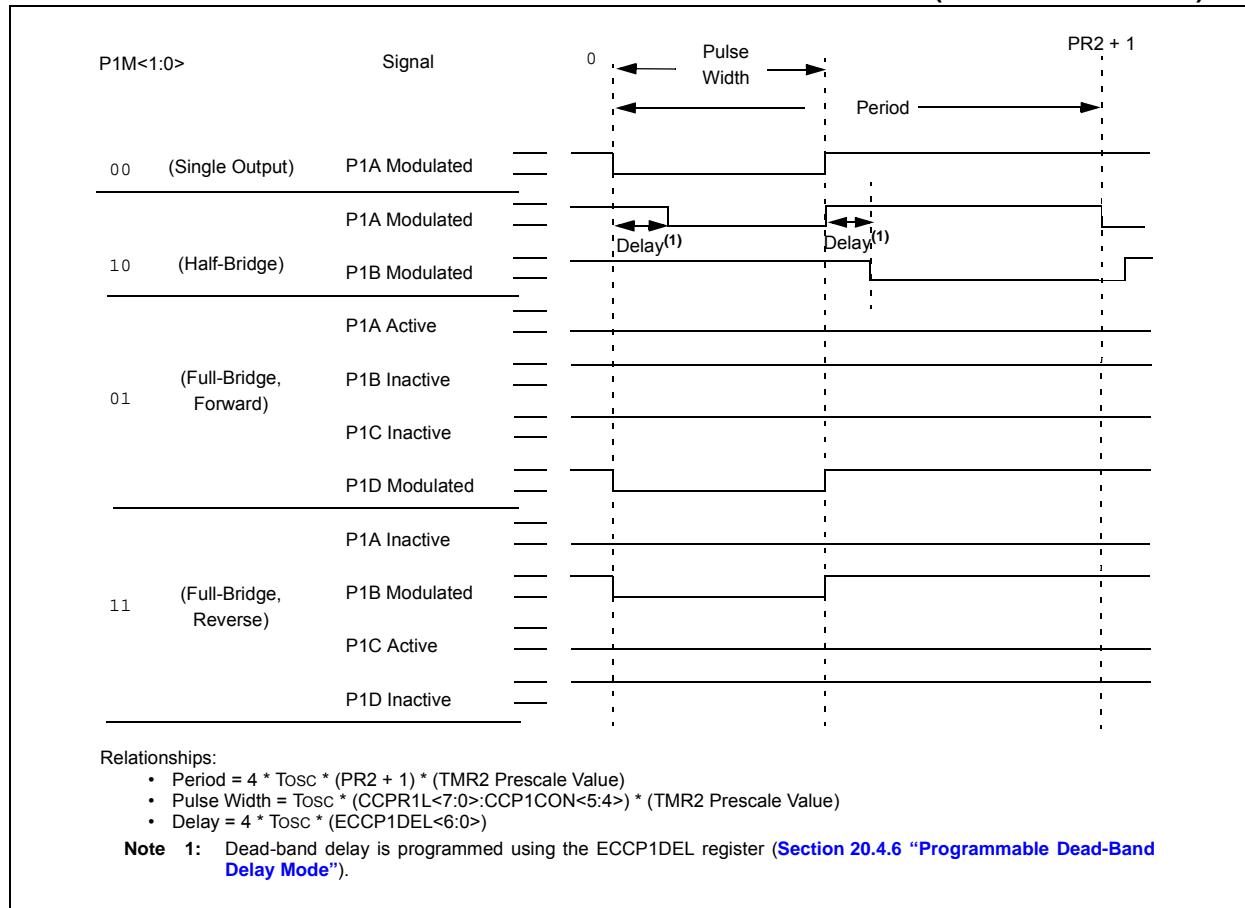
Note 1: Outputs are enabled by pulse steering in Single mode (see [Register 20-5](#)).

FIGURE 20-4: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)



# PIC18F66K80 FAMILY

**FIGURE 20-5: EXAMPLE ENHANCED PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**



# PIC18F66K80 FAMILY

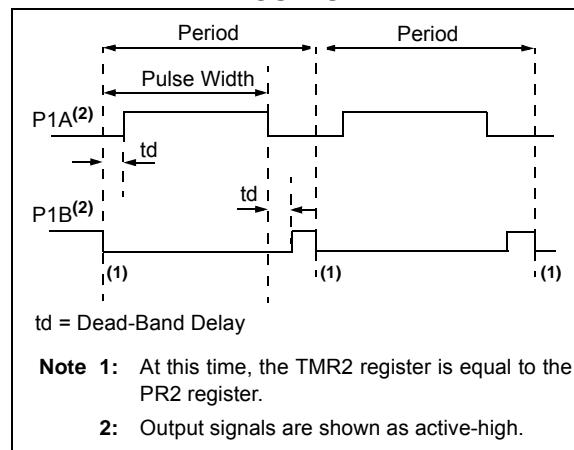
## 20.4.1 HALF-BRIDGE MODE

In Half-Bridge mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the P1A pin, while the complementary PWM output signal is output on the P1B pin (see [Figure 20-6](#)). This mode can be used for half-bridge applications, as shown in [Figure 20-7](#), or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of the P1DC<sub><6:0</sub> bits of the ECCP1DEL register sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. For more details on the dead-band delay operations, see [Section 20.4.6 “Programmable Dead-Band Delay Mode”](#).

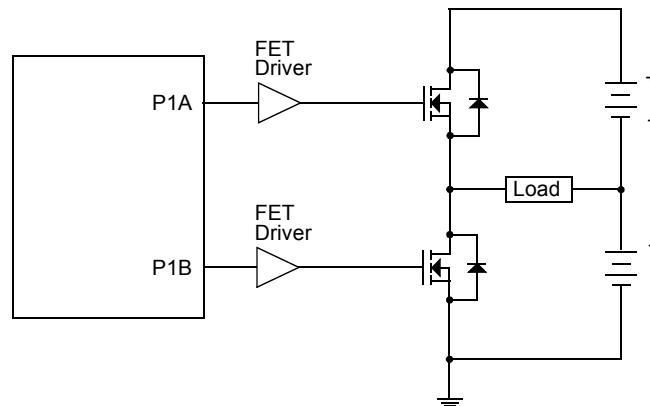
Since the P1A and P1B outputs are multiplexed with the port data latches, the associated TRIS bits must be cleared to configure P1A and P1B as outputs.

**FIGURE 20-6: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**

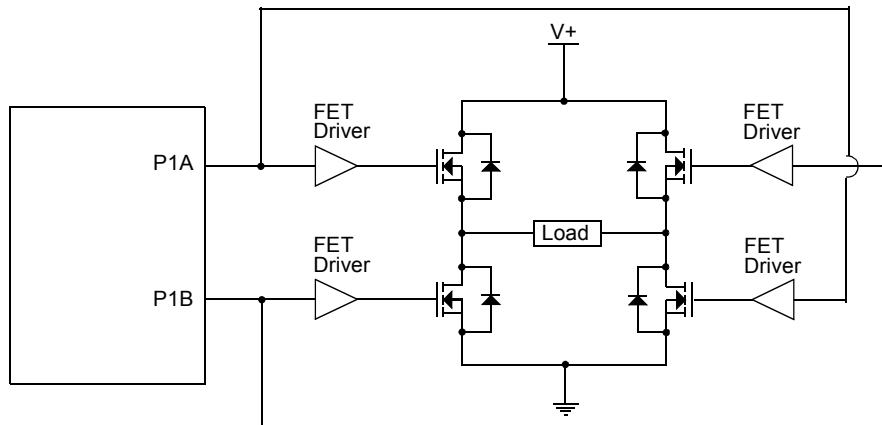


**FIGURE 20-7: EXAMPLE OF HALF-BRIDGE APPLICATIONS**

Standard Half-Bridge Circuit (“Push-Pull”)



Half-Bridge Output Driving a Full-Bridge Circuit



## 20.4.2 FULL-BRIDGE MODE

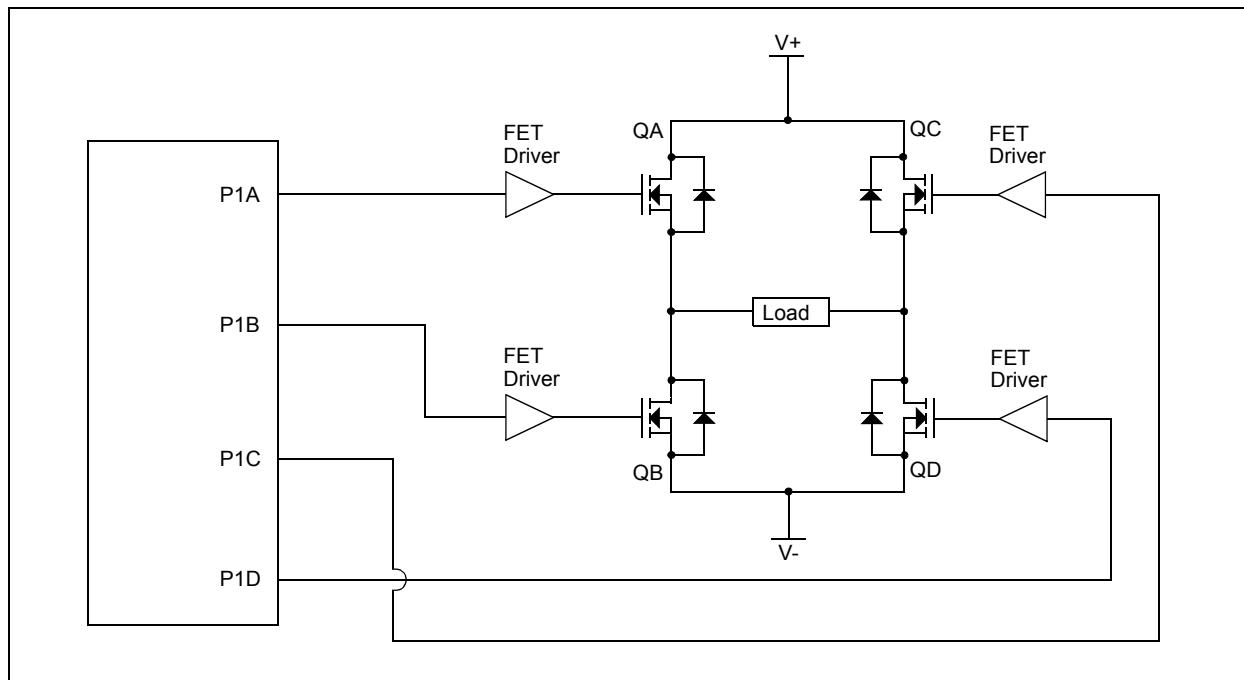
In Full-Bridge mode, all four pins are used as outputs. An example of a full-bridge application is provided in [Figure 20-8](#).

In the Forward mode, the P1A pin is driven to its active state and the P1D pin is modulated, while the P1B and P1C pins are driven to their inactive state, as provided in [Figure 20-9](#).

In the Reverse mode, the P1C pin is driven to its active state and the P1B pin is modulated, while the P1A and P1D pins are driven to their inactive state, as provided in [Figure 20-9](#).

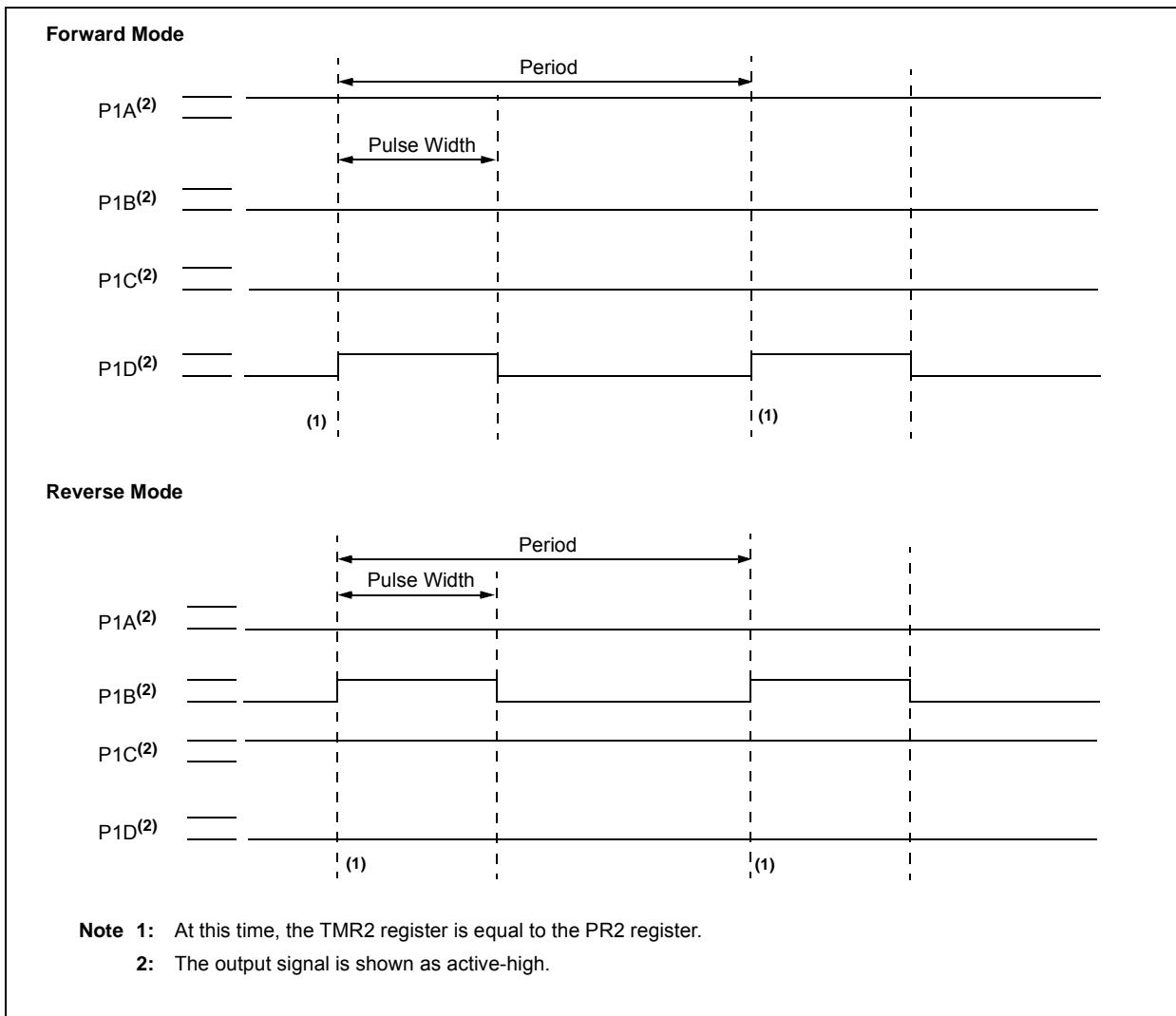
The P1A, P1B, P1C and P1D outputs are multiplexed with the port data latches. The associated TRIS bits must be cleared to configure the P1A, P1B, P1C and P1D pins as outputs.

**FIGURE 20-8: EXAMPLE OF FULL-BRIDGE APPLICATION**



# PIC18F66K80 FAMILY

FIGURE 20-9: EXAMPLE OF FULL-BRIDGE PWM OUTPUT



## 20.4.2.1 Direction Change in Full-Bridge Mode

In the Full-Bridge mode, the P1M1 bit in the CCP1CON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will change to the new direction on the next PWM cycle.

A direction change is initiated in software by changing the P1M1 bit of the CCP1CON register. The following sequence occurs prior to the end of the current PWM period:

- The modulated outputs (P1B and P1D) are placed in their inactive state.
- The associated unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction.
- PWM modulation resumes at the beginning of the next period.

For an illustration of this sequence, see [Figure 20-10](#).

The Full-Bridge mode does not provide a dead-band delay. As one output is modulated at a time, a dead-band delay is generally not required. There is a situation where a dead-band delay is required. This situation occurs when both of the following conditions are true:

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

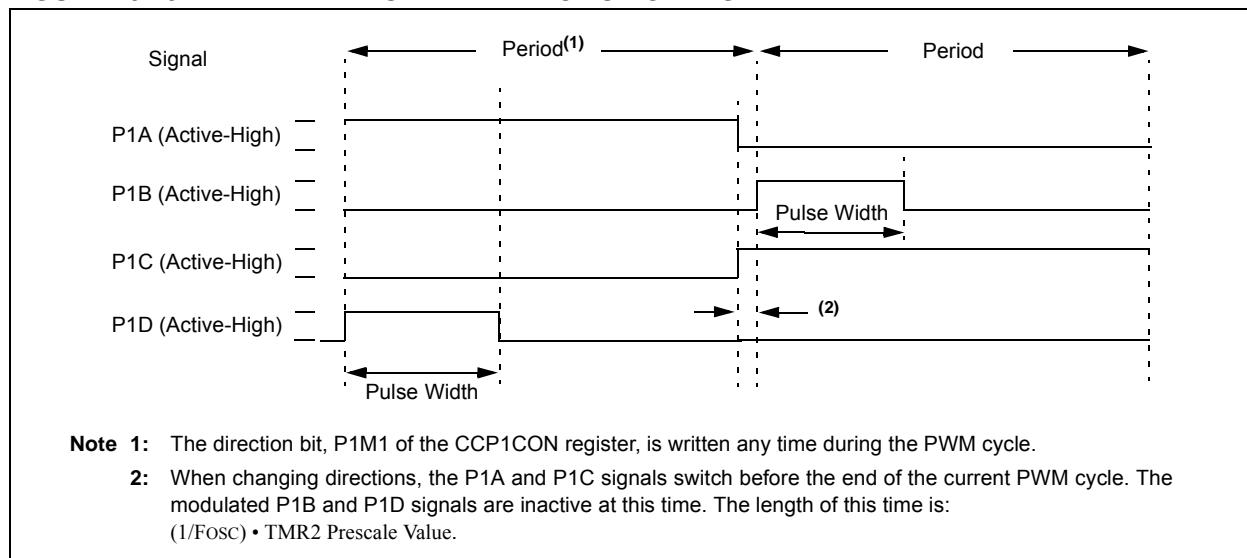
[Figure 20-11](#) shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time, t1, the P1A and P1D outputs become inactive, while the P1C output becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices, QC and QD (see [Figure 20-8](#)), for the duration of 't'. The same phenomenon will occur to power devices, QA and QB, for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

- Reduce PWM duty cycle for one PWM period before changing directions.
- Use switch drivers that can drive the switches off faster than they can drive them on.

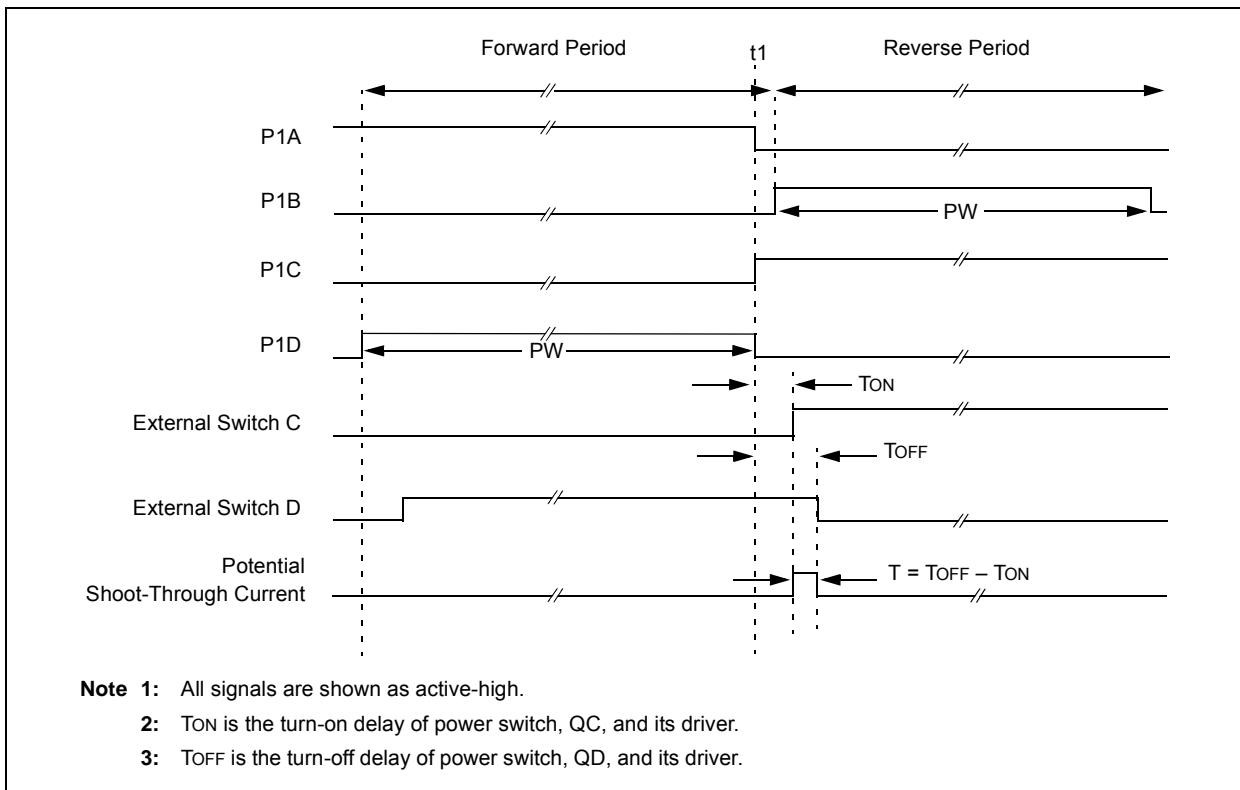
Other options to prevent shoot-through current may exist.

**FIGURE 20-10: EXAMPLE OF PWM DIRECTION CHANGE**



# PIC18F66K80 FAMILY

FIGURE 20-11: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



#### 20.4.3 START-UP CONSIDERATIONS

When any PWM mode is used, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins.

**Note:** When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the OFF state until the microcontroller drives the I/O pins with the proper signal levels or activates the PWM output(s).

The CCP1M<1:0> bits of the CCP1CON register allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (P1A/P1C and P1B/P1D). The PWM output polarities must be selected before the PWM pin output drivers are enabled. Changing the polarity configuration while the PWM pin output drivers are enabled is not recommended since it may result in damage to the application circuits.

The P1A, P1B, P1C and P1D output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pin output drivers at the same time as the Enhanced PWM modes may cause

damage to the application circuit. The Enhanced PWM modes must be enabled in the proper Output mode and complete a full PWM cycle before enabling the PWM pin output drivers. The completion of a full PWM cycle is indicated by the TMR2IF or TMR4IF bit of the PIR1 or PIR4 register being set as the second PWM period begins.

#### 20.4.4 ENHANCED PWM AUTO-SHUTDOWN MODE

The PWM mode supports an Auto-Shutdown mode that will disable the PWM outputs when an external shutdown event occurs. Auto-Shutdown mode places the PWM output pins into a predetermined state. This mode is used to help prevent the PWM from damaging the application.

The auto-shutdown sources are selected using the ECCP1AS<2:0> bits (ECCP1AS<6:4>). A shutdown event may be generated by:

- A logic '0' on the pin that is assigned the FLT0 input function
- Comparator C1
- Comparator C2
- Setting the ECCP1ASE bit in firmware

A shutdown condition is indicated by the ECCP1ASE (Auto-Shutdown Event Status) bit (ECCP1AS<7>). If the bit is a '0', the PWM pins are operating normally. If the bit is a '1', the PWM outputs are in the shutdown state.

When a shutdown event occurs, two things happen:

- The ECCP1ASE bit is set to '1'. The ECCP1ASE will remain set until cleared in firmware or an auto-restart occurs. (See [Section 20.4.5 "Auto-Restart Mode"](#).)
- The enabled PWM pins are asynchronously placed in their shutdown states. The PWM output pins are grouped into pairs (P1A/P1C) and (P1B/P1D). The state of each pin pair is determined by the PSS1ACx and PSS1BDx bits (ECCP1AS<3:2> and <1:0>, respectively).

Each pin pair may be placed into one of three states:

- Drive logic '1'
- Drive logic '0'
- Tri-state (high-impedance)

## REGISTER 20-3: ECCP1AS: ECCP1 AUTO-SHUTDOWN CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ECCP1ASE	ECCP1AS2	ECCP1AS1	ECCP1AS0	PSS1AC1	PSS1AC0	PSS1BD1	PSS1BD0
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

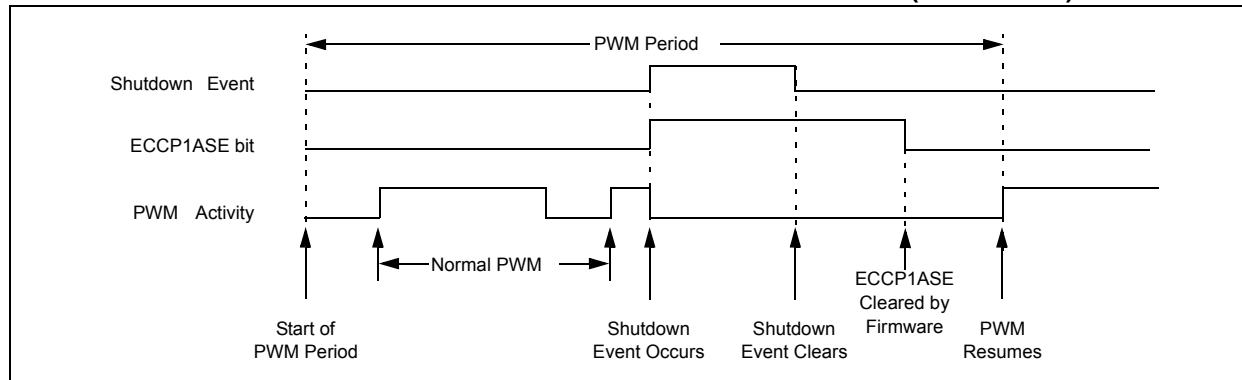
x = Bit is unknown

bit 7	<b>ECCP1ASE:</b> ECCP Auto-Shutdown Event Status bit 1 = A shutdown event has occurred; ECCP outputs are in a shutdown state 0 = ECCP outputs are operating
bit 6-4	<b>ECCP1AS&lt;2:0&gt;:</b> ECCP Auto-Shutdown Source Select bits 000 = Auto-shutdown is disabled 001 = Comparator C1OUT output is high 010 = Comparator C2OUT output is high 011 = Either Comparator C1OUT or C2OUT is high 100 = VIL on FLT0 pin 101 = VIL on FLT0 pin or Comparator C1OUT output is high 110 = VIL on FLT0 pin or Comparator C2OUT output is high 111 = VIL on FLT0 pin or Comparator C1OUT or Comparator C2OUT is high
bit 3-2	<b>PSS1AC&lt;1:0&gt;:</b> P1A and P1C Pins Shutdown State Control bits 00 = Drive pins, P1A and P1C, to '0' 01 = Drive pins, P1A and P1C, to '1' 1x = Pins, P1A and P1C, tri-state
bit 1-0	<b>PSS1BD&lt;1:0&gt;:</b> P1B and P1D Pins Shutdown State Control bits 00 = Drive pins, P1B and P1D, to '0' 01 = Drive pins, P1B and P1D, to '1' 1x = Pins, P1B and P1D, tri-state

- Note 1:** The auto-shutdown condition is a level-based signal, not an edge-based signal. As long as the level is present, the auto-shutdown will persist.
- 2:** Writing to the ECCP1ASE bit is disabled while an auto-shutdown condition persists.
- 3:** Once the auto-shutdown condition has been removed and the PWM restarted (either through firmware or auto-restart), the PWM signal will always restart at the beginning of the next PWM period.

# PIC18F66K80 FAMILY

**FIGURE 20-12: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (P1RSEN = 0)**



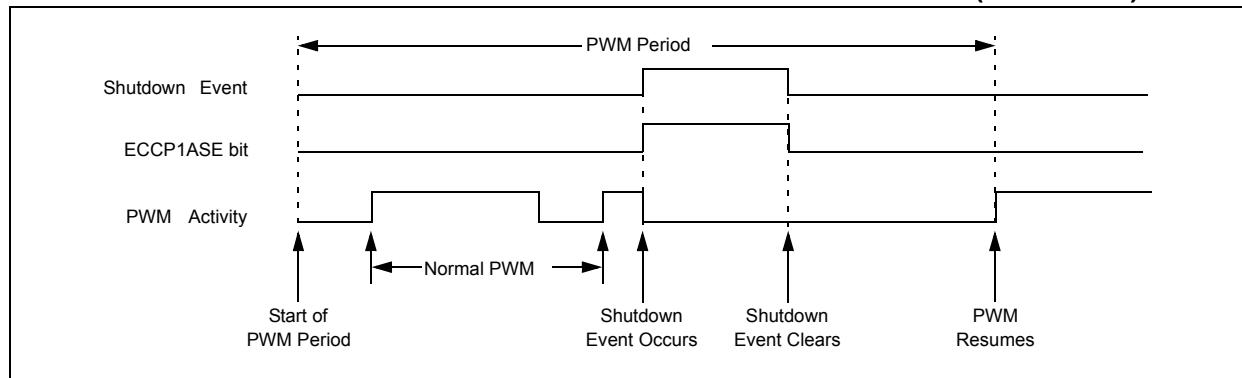
## 20.4.5 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the P1RSEN bit (ECCP1DEL<7>).

If auto-restart is enabled, the ECCP1ASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the ECCP1ASE bit will be cleared via hardware and normal operation will resume.

The module will wait until the next PWM period begins, however, before re-enabling the output pin. This behavior allows the auto-shutdown with auto-restart features to be used in applications based on current mode of PWM control.

**FIGURE 20-13: PWM AUTO-SHUTDOWN WITH AUTO-RESTART ENABLED (P1RSEN = 1)**

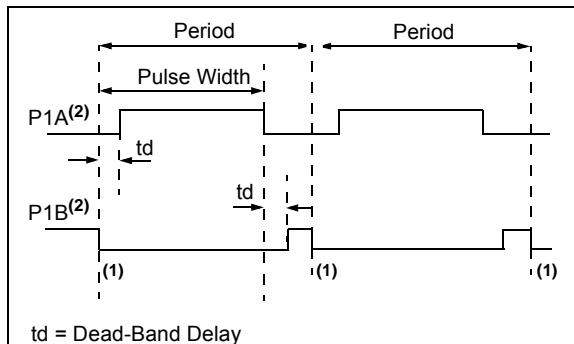


## 20.4.6 PROGRAMMABLE DEAD-BAND DELAY MODE

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

In Half-Bridge mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. For an illustration, see [Figure 20-14](#). The lower seven bits of the associated ECCP1DEL register ([Register 20-4](#)) set the delay period in terms of microcontroller instruction cycles (TCY or 4 Tosc).

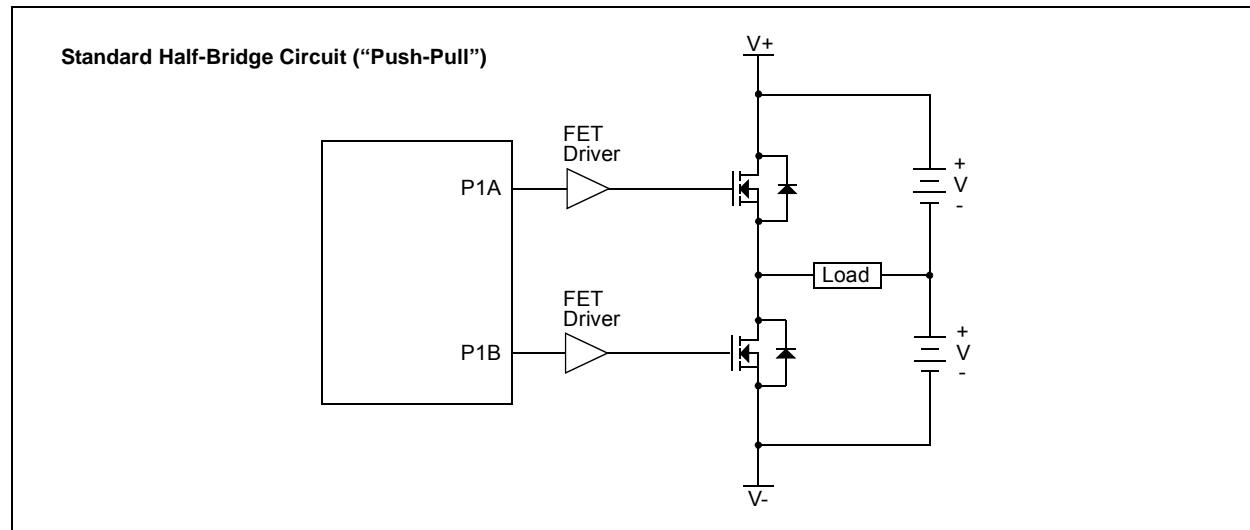
**FIGURE 20-14: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**



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

2: Output signals are shown as active-high.

**FIGURE 20-15: EXAMPLE OF HALF-BRIDGE APPLICATIONS**



# PIC18F66K80 FAMILY

## REGISTER 20-4: ECCP1DEL: ENHANCED PWM 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
P1RSEN	P1DC6	P1DC5	P1DC4	P1DC3	P1DC2	P1DC1	P1DC0
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

### P1RSEN: PWM Restart Enable bit

1 = Upon auto-shutdown, the ECCP1ASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically  
0 = Upon auto-shutdown, ECCP1ASE must be cleared by software to restart the PWM

bit 6-0

### P1DC<6:0>: PWM Delay Count bits

P1DCn = Number of Fosc/4 (4 \* Tosc) cycles between the scheduled time when a PWM signal **should** transition active and the **actual** time it does transition active.

## 20.4.7 PULSE STEERING MODE

In Single Output mode, pulse steering allows any of the PWM pins to be the modulated signal. Additionally, the same PWM signal can simultaneously be available on multiple pins.

Once the Single Output mode is selected (CCP1M<3:2> = 11 and P1M<1:0> = 00 of the CCP1CON register), the user firmware can bring out the same PWM signal to one, two, three or four output pins by setting the appropriate STR<D:A> bits (PSTR1CON<3:0>), as provided in [Table 20-2](#).

**Note:** The associated TRIS bits must be set to output ('0') to enable the pin output driver in order to see the PWM signal on the pin.

While the PWM Steering mode is active, the CCP1M<1:0> bits (CCP1CON<1:0>) select the PWM output polarity for the P1<D:A> pins.

The PWM auto-shutdown operation also applies to the PWM Steering mode, as described in [Section 20.4.4 "Enhanced PWM Auto-shutdown mode"](#). An auto-shutdown event will only affect pins that have PWM outputs enabled.

# PIC18F66K80 FAMILY

## REGISTER 20-5: PSTR1CON: PULSE STEERING CONTROL<sup>(1)</sup>

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
CMPL1	CMPL0	—	STRSYNC	STRD	STRC	STRB	STRA
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

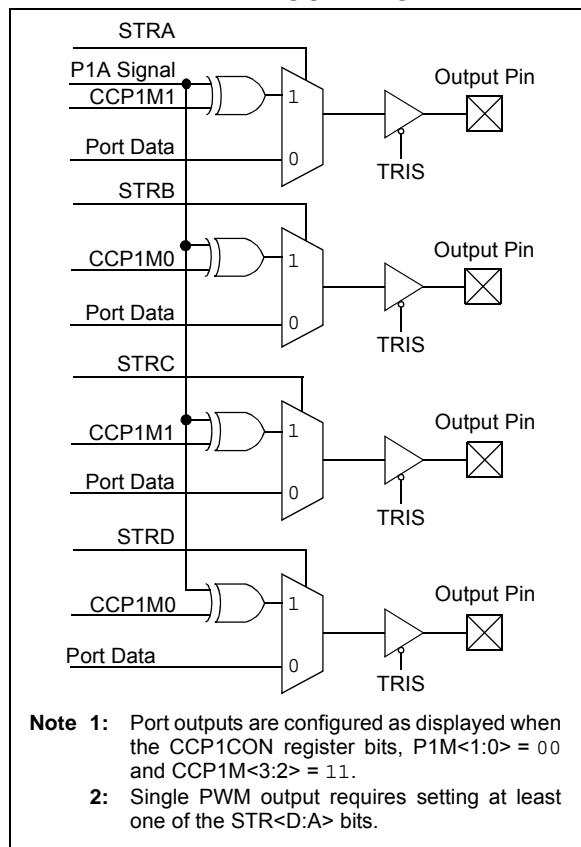
x = Bit is unknown

bit 7-6	<b>CMPL&lt;1:0&gt;</b> : Complementary Mode Output Assignment Steering Sync bits 00 = See STR<D:A>. 01 = PA and PB are selected as the complementary output pair 10 = PA and PC are selected as the complementary output pair 11 = PA and PD are selected as the complementary output pair
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>STRSYNC:</b> Steering Sync bit 1 = Output steering update occurs on the next PWM period 0 = Output steering update occurs at the beginning of the instruction cycle boundary
bit 3	<b>STRD:</b> Steering Enable bit D 1 = P1D pin has the PWM waveform with polarity control from CCP1M<1:0> 0 = P1D pin is assigned to port pin
bit 2	<b>STRC:</b> Steering Enable bit C 1 = P1C pin has the PWM waveform with polarity control from CCP1M<1:0> 0 = P1C pin is assigned to port pin
bit 1	<b>STRB:</b> Steering Enable bit B 1 = P1B pin has the PWM waveform with polarity control from CCP1M<1:0> 0 = P1B pin is assigned to port pin
bit 0	<b>STRA:</b> Steering Enable bit A 1 = P1A pin has the PWM waveform with polarity control from CCP1M<1:0> 0 = P1A pin is assigned to port pin

**Note 1:** The PWM Steering mode is available only when the CCP1CON register bits, CCP1M<3:2> = 11 and P1M<1:0> = 00.

# PIC18F66K80 FAMILY

**FIGURE 20-16:** SIMPLIFIED STEERING BLOCK DIAGRAM<sup>(1,2)</sup>



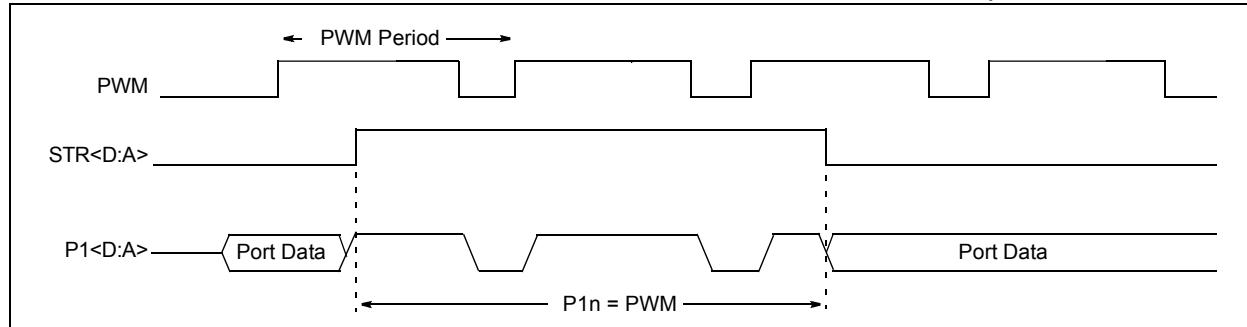
## 20.4.7.1 Steering Synchronization

The STRSYNC bit of the PSTR1CON register gives the user two choices for when the steering event will happen. When the STRSYNC bit is '0', the steering event will happen at the end of the instruction that writes to the PSTR1CON register. In this case, the output signal at the P1<D:A> pins may be an incomplete PWM waveform. This operation is useful when the user firmware needs to immediately remove a PWM signal from the pin.

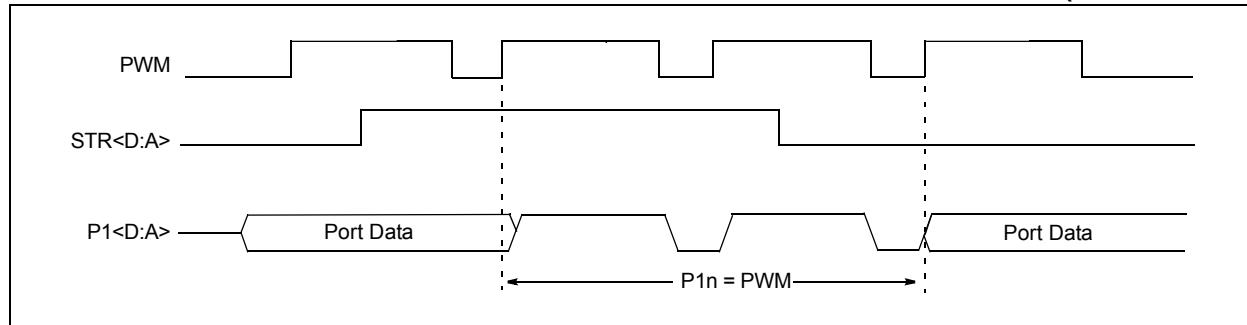
When the STRSYNC bit is '1', the effective steering update will happen at the beginning of the next PWM period. In this case, steering on/off the PWM output will always produce a complete PWM waveform.

Figures 20-17 and 20-18 illustrate the timing diagrams of the PWM steering depending on the STRSYNC setting.

**FIGURE 20-17:** EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRSYNC = 0 )



**FIGURE 20-18:** EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRSYNC = 1 )



## 20.4.8 OPERATION IN POWER-MANAGED MODES

In Sleep mode, all clock sources are disabled. Timer2/4 will not increment and the state of the module will not change. If the ECCP1 pin is driving a value, it will continue to drive that value. When the device wakes up, it will continue from this state. If Two-Speed Start-ups are enabled, the initial start-up frequency from HF-INTOSC and the postscaler may not be stable immediately.

In PRI\_IDLE mode, the primary clock will continue to clock the ECCP1 module without change.

## 20.4.8.1 Operation with Fail-Safe Clock Monitor (FSCM)

If the Fail-Safe Clock Monitor (FSCM) is enabled, a clock failure will force the device into the power-managed RC\_RUN mode and the OSCFIF bit of the PIR2 register will be set. The ECCP1 will then be clocked from the internal oscillator clock source, which may have a different clock frequency than the primary clock.

## 20.4.9 EFFECTS OF A RESET

Both Power-on Reset and subsequent Resets will force all ports to Input mode and the CCP registers to their Reset states.

This forces the ECCP module to reset to a state compatible with previous, non-enhanced CCP modules used on other PIC18 and PIC16 devices.

# PIC18F66K80 FAMILY

---

**TABLE 20-3: REGISTERS ASSOCIATED WITH ECCP1 MODULE AND TIMER1/2/3/4**

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
RCON	IPEN	SBOREN	CM	RI	TO	PD	POR	BOR
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
TRISE <sup>(1)</sup>	TRISE7	TRISE6	TRISE5	TRISE4	—	TRISE2	TRISE1	TRISE0
TMR1H	Timer1 Register High Byte							
TMR1L	Timer1 Register Low Byte							
TMR2	Timer2 Register							
TMR3H	Timer3 Register High Byte							
TMR3L	Timer3 Register Low Byte							
TMR4	Timer4 Register							
PR2	Timer2 Period Register							
PR4	Timer4 Period Register							
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	SOSCEN	T1SYNC	RD16	TMR1ON
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	SOSCEN	T3SYNC	RD16	TMR3ON
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
CCPR1H	Capture/Compare/PWM Register 1 High Byte							
CCPR1L	Capture/Compare/PWM Register 1 Low Byte							
CCPR2H	Capture/Compare/PWM Register 2 High Byte							
CCPR2L	Capture/Compare/PWM Register 2 Low Byte							
CCPR3H	Capture/Compare/PWM Register 3 High Byte							
CCPR3L	Capture/Compare/PWM Register 3 Low Byte							
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0
CCPTMRS	—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL
ECCP1AS	ECCP1ASE	ECCP1AS2	ECCP1AS1	ECCP1AS0	PSS1AC1	PSS1AC0	PSS1BD1	PSS1BD0
ECCP1DEL	P1RSEN	P1DC6	P1DC5	P1DC4	P1DC3	P1DC2	P1DC1	P1DC0
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18F25K80 and PIC18F46K80).

## 21.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

### 21.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be devices such as serial EEPROMs, shift registers, display drivers and A/D Converters. The MSSP module can operate in either of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit ( $I^2C$ )<sup>TM</sup>
  - Full Master mode
  - Slave mode (with general address call)

The  $I^2C$  interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode with 5-bit and 7-bit address masking (with address masking for both 10-bit and 7-bit addressing)

### 21.2 Control Registers

The MSSP module has three associated control registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual configuration bits differ significantly depending on whether the MSSP module is operated in SPI or  $I^2C$  mode.

Additional details are provided under the individual sections.

### 21.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

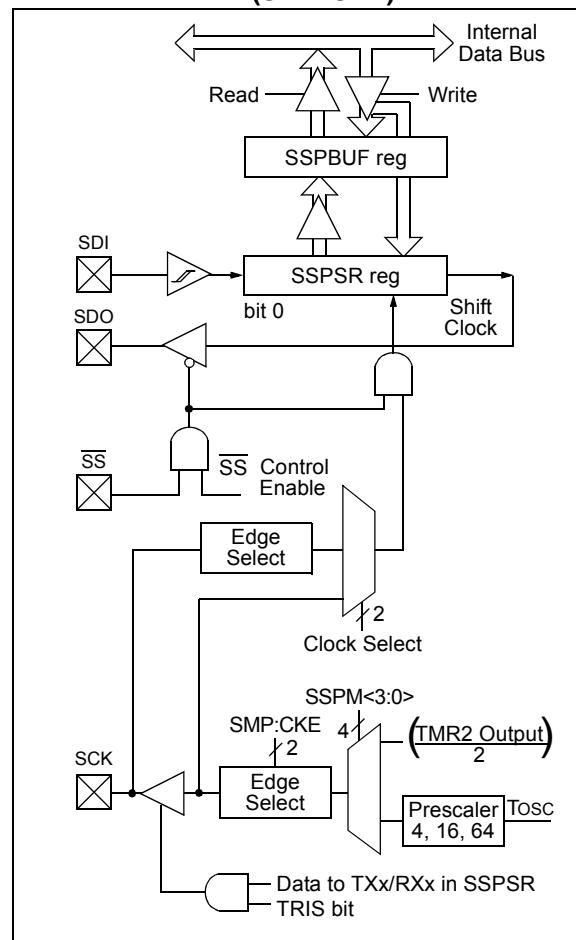
- Serial Data Out (SDO) – RC5/SDO
- Serial Data In (SDI) – RC4/SDA/SDI
- Serial Clock (SCK) – RC3/REF0/SCL/SCK

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ( $\overline{SS}$ ) – RA5/AN4/C2INB/HLDIN/T1CKI/SS/CTMU1

Figure 21-1 shows the block diagram of the MSSP module when operating in SPI mode.

**FIGURE 21-1: MSSP BLOCK DIAGRAM (SPI MODE)**



**Note:** Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

# PIC18F66K80 FAMILY

## 21.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

## REGISTER 21-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE <sup>(1)</sup>	D/A	P	S	R/W	UA	BF
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

**SMP:** Sample bit

#### SPI Master mode:

1 = Input data is sampled at the end of data output time

0 = Input data is sampled at the middle of data output time

#### SPI Slave mode:

SMP must be cleared when SPI is used in Slave mode.

bit 6

**CKE:** SPI Clock Select bit<sup>(1)</sup>

1 = Transmit occurs on transition from active to Idle clock state

0 = Transmit occurs on transition from Idle to active clock state

bit 5

**D/A:** Data/Address bit

Used in I<sup>2</sup>C™ mode only.

bit 4

**P:** Stop bit

Used in I<sup>2</sup>C mode only. This bit is cleared when the MSSP module is disabled; SSPEN is cleared.

bit 3

**S:** Start bit

Used in I<sup>2</sup>C mode only.

bit 2

**R/W:** Read/Write Information bit

Used in I<sup>2</sup>C mode only.

bit 1

**UA:** Update Address bit

Used in I<sup>2</sup>C mode only.

bit 0

**BF:** Buffer Full Status bit (Receive mode only)

1 = Receive is complete, SSPBUF is full

0 = Receive is not complete, SSPBUF is empty

**Note 1:** Polarity of clock state is set by the CKP bit (SSPCON1<4>).

## REGISTER 21-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV <sup>(1)</sup>	SSPEN <sup>(2)</sup>	CKP	SSPM3 <sup>(3)</sup>	SSPM2 <sup>(3)</sup>	SSPM1 <sup>(3)</sup>	SSPM0 <sup>(3)</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	<b>WCOL:</b> Write Collision Detect bit 1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software) 0 = No collision
bit 6	<b>SSPOV:</b> Receive Overflow Indicator bit <sup>(1)</sup> <b>SPI Slave mode:</b> 1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software). 0 = No overflow
bit 5	<b>SSPEN:</b> Master Synchronous Serial Port Enable bit <sup>(2)</sup> 1 = Enables the serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins 0 = Disables the serial port and configures these pins as I/O port pins
bit 4	<b>CKP:</b> Clock Polarity Select bit 1 = Idle state for clock is a high level 0 = Idle state for clock is a low level
bit 3-0	<b>SSPM&lt;3:0&gt;:</b> Master Synchronous Serial Port Mode Select bits <sup>(3)</sup> 1010 = SPI Master mode: clock = Fosc/8 0101 = SPI Slave mode: clock = SCK pin; $\overline{SS}$ pin control disabled; $\overline{SS}$ can be used as I/O pin 0100 = SPI Slave mode: clock = SCK pin; SS pin control enabled 0011 = SPI Master mode: clock = TMR2 output/2 0010 = SPI Master mode: clock = Fosc/64 0001 = SPI Master mode: clock = Fosc/16 0000 = SPI Master mode: clock = Fosc/4

**Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

**2:** When enabled, these pins must be properly configured as inputs or outputs.

**3:** Bit combinations not specifically listed here are either reserved or implemented in I<sup>2</sup>C mode only.

# PIC18F66K80 FAMILY

---

## 21.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

The MSSP module consists of a Transmit/Receive Shift register (SSPSR) and a Buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full detect bit, BF (SSPSTAT<0>), and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the Write Collision Detect bit, WCOL (SSPCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. [Example 21-1](#) shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the SSPSTAT register indicates the various status conditions.

## 21.3.3 OPEN-DRAIN OUTPUT OPTION

The drivers for the SDO output and SCK clock pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor, and allows the output to communicate with external circuits without the need for additional level shifters. For more information, see [Section 11.1.3 “Open-Drain Outputs”](#).

The open-drain output option is controlled by the SSPOD bit (ODCON<7>). Setting the SSPOD bit configures the SDO and SCK pins for open-drain operation.

## EXAMPLE 21-1: LOADING THE SSPBUF (SSPSR) REGISTER

LOOP	BTFSS	SSPSTAT, BF	;Has data been received (transmit complete)?
	BRA	LOOP	;No
	MOVF	SSPBUF, W	;WREG reg = contents of SSPBUF
	MOVWF	RXDATA	;Save in user RAM, if data is meaningful
	MOVF	TXDATA, W	;W reg = contents of TXDATA
	MOVWF	SSPBUF	;New data to xmit

### 21.3.4 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have the TRISC<5> bit cleared
- SCK (Master mode) must have the TRISC<3> bit cleared
- SCK (Slave mode) must have the TRISC<3> bit set
- SS must have the TRISA<5> bit set

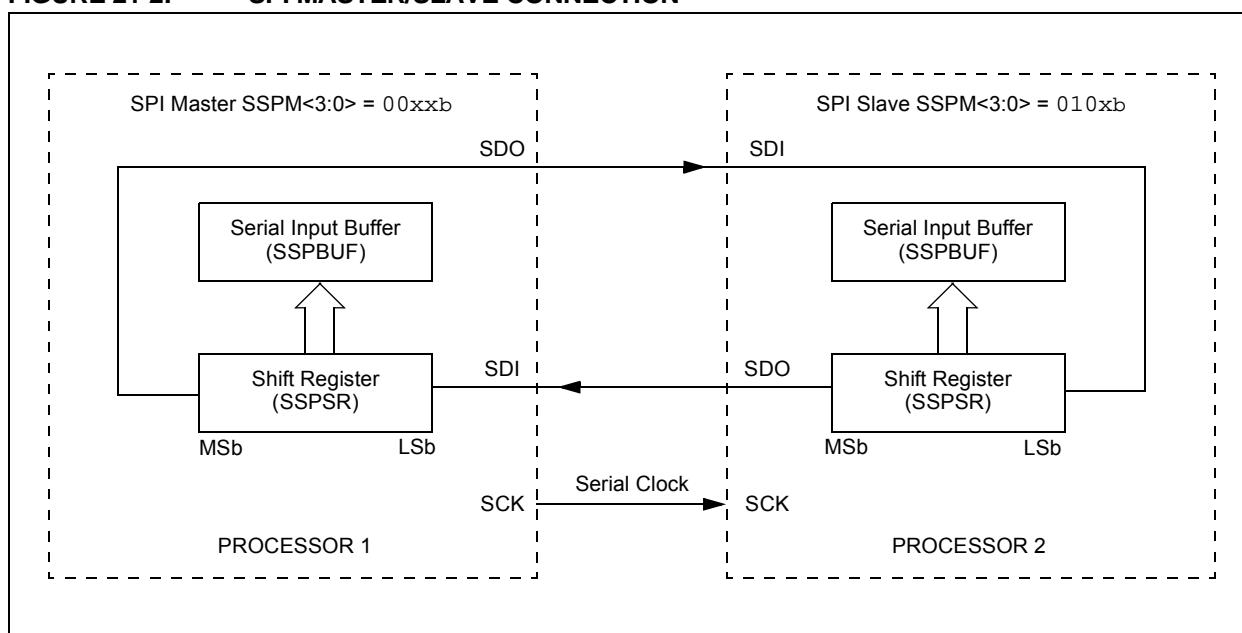
Any serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value.

### 21.3.5 TYPICAL CONNECTION

**Figure 21-2** shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

**FIGURE 21-2: SPI MASTER/SLAVE CONNECTION**



# PIC18F66K80 FAMILY

## 21.3.6 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 1, [Figure 21-2](#)) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a “Line Activity Monitor” mode.

The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication as

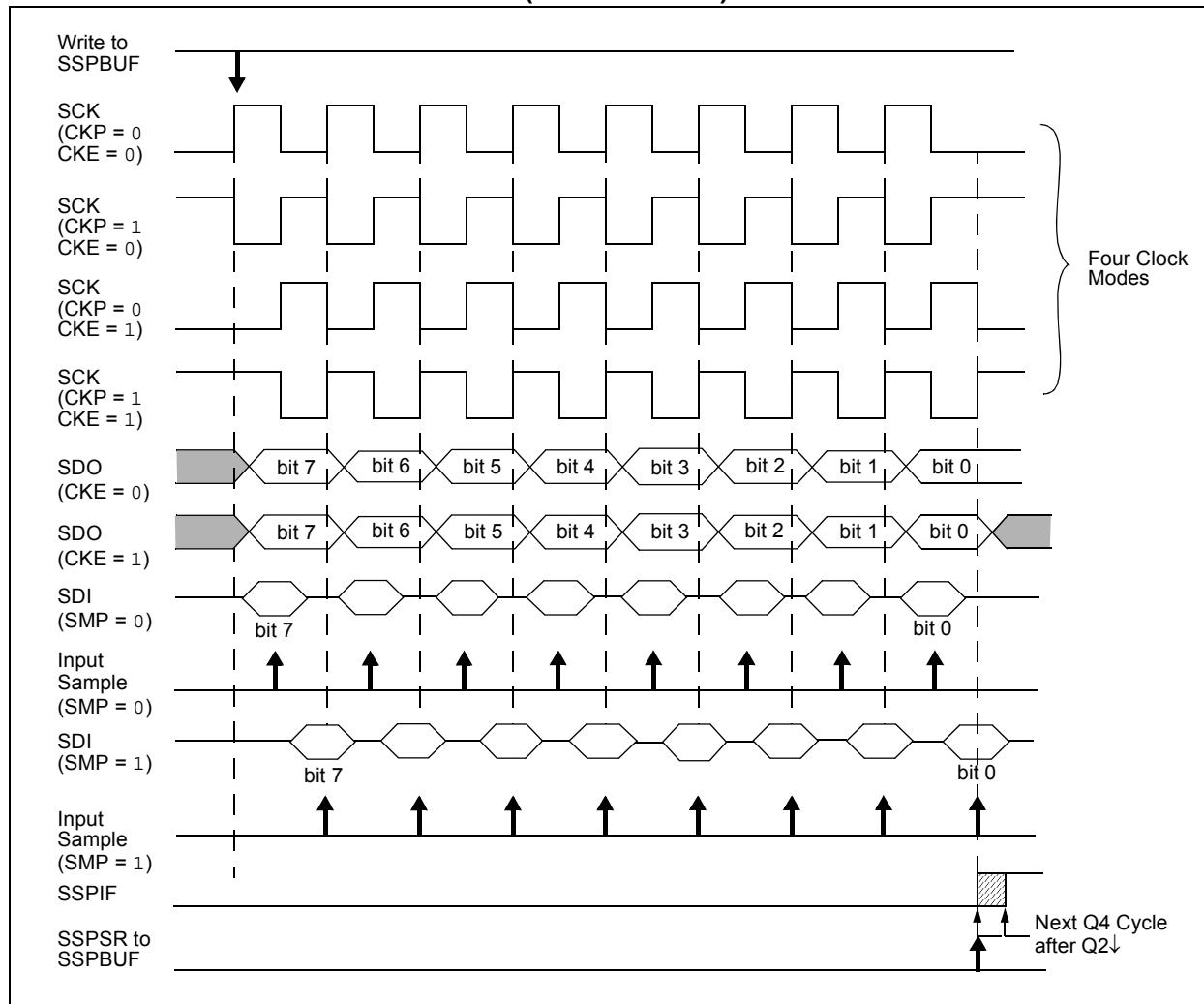
shown in [Figure 21-3](#), [Figure 21-5](#) and [Figure 21-6](#), where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- Fosc/4 (or TCY)
- Fosc/16 (or 4 • TCY)
- Fosc/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum data rate (at 64 MHz) of 16 Mbps.

[Figure 21-3](#) shows the waveforms for Master mode. When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

**FIGURE 21-3: SPI MODE WAVEFORM (MASTER MODE)**



### 21.3.7 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device can be configured to wake-up from Sleep.

### 21.3.8 SLAVE SELECT SYNCHRONIZATION

The SS pin allows a Synchronous Slave mode. The SPI must be in Slave mode with the SS pin control enabled (SSPCON1<3:0> = 04h). When the SS pin is low, transmission and reception are enabled and the SDO pin is driven. When the SS pin goes high, the SDO pin is no longer driven, even if in the middle of a

transmitted byte, and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

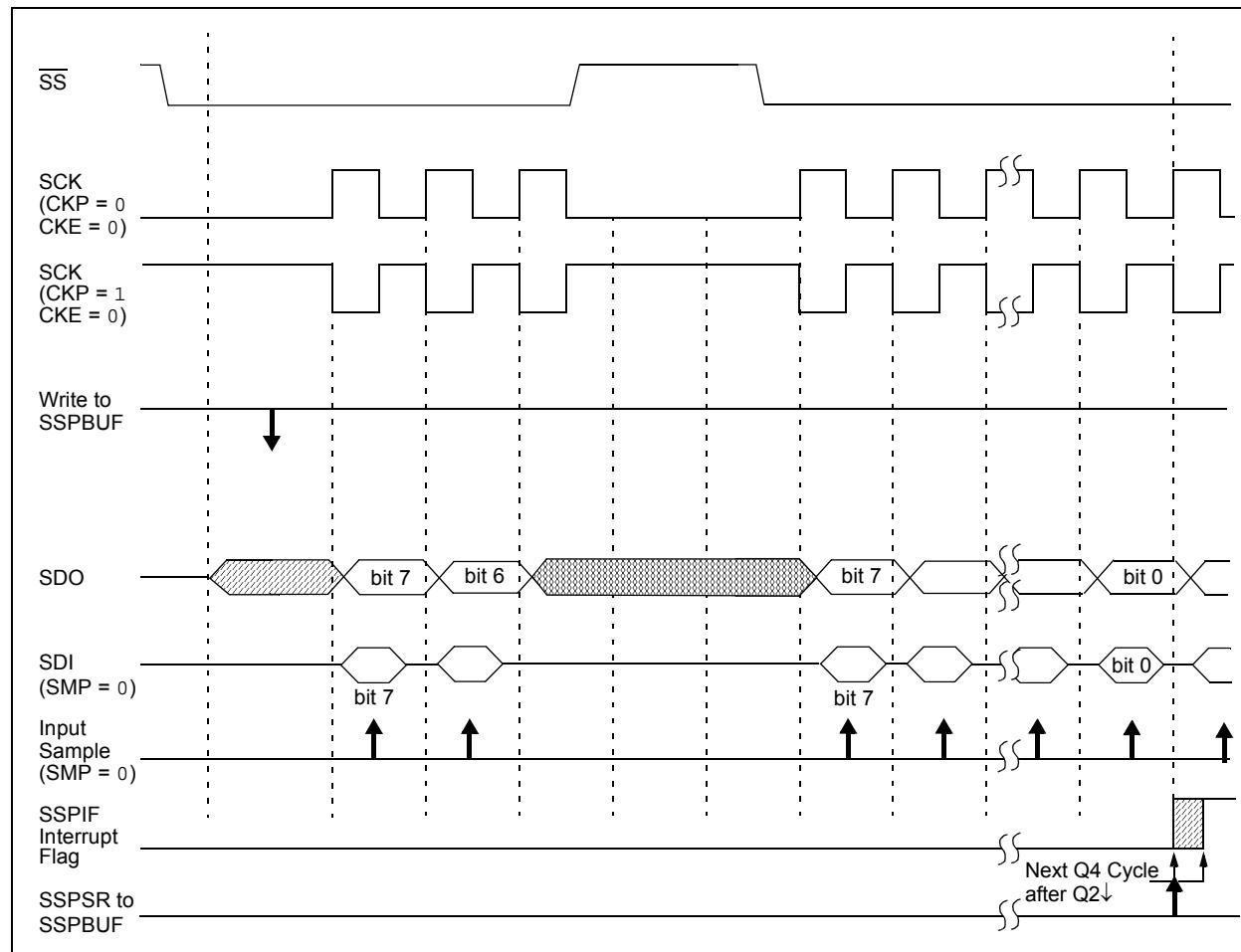
**Note 1:** When the SPI is in Slave mode, with SS pin control enabled (SSPCON1<3:0> = 0100), the SPI module will reset if the SS pin is set to VDD.

**2:** If the SPI is used in Slave mode, with CKE set, then the SS pin control must be enabled.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the SS pin to a high level or clearing the SSPEN bit.

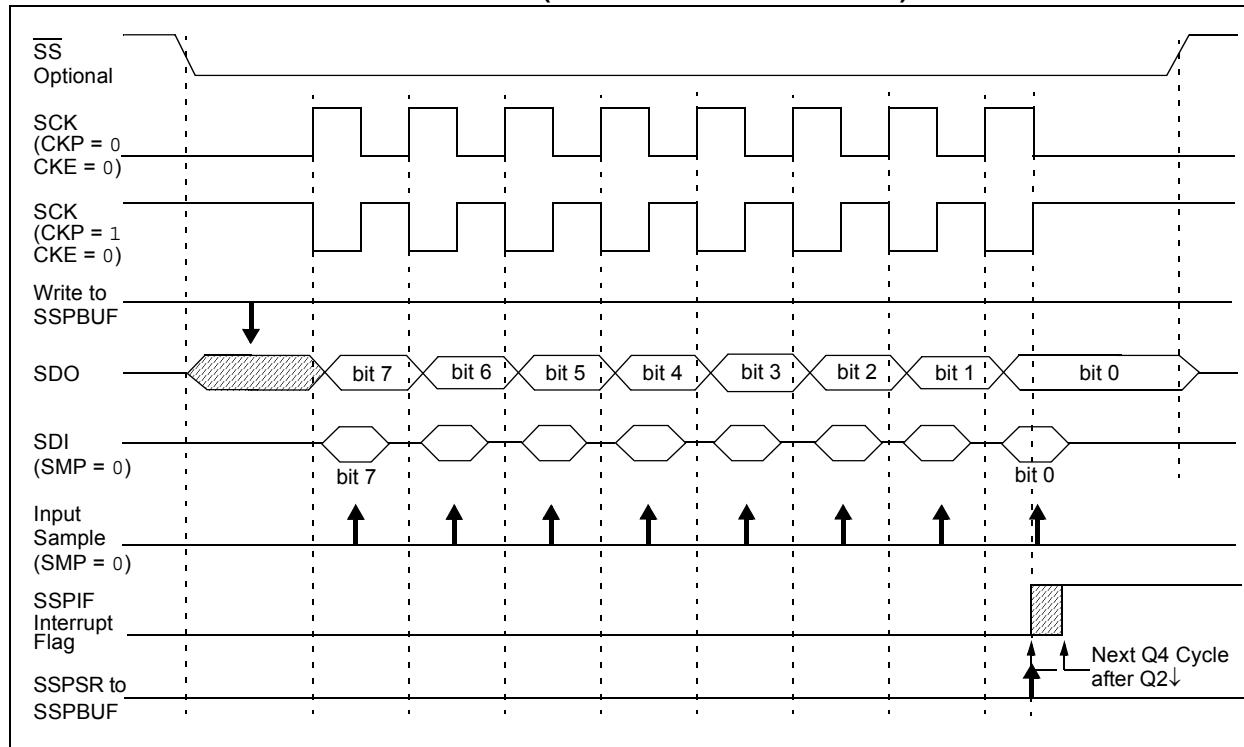
To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

**FIGURE 21-4: SLAVE SYNCHRONIZATION WAVEFORM**

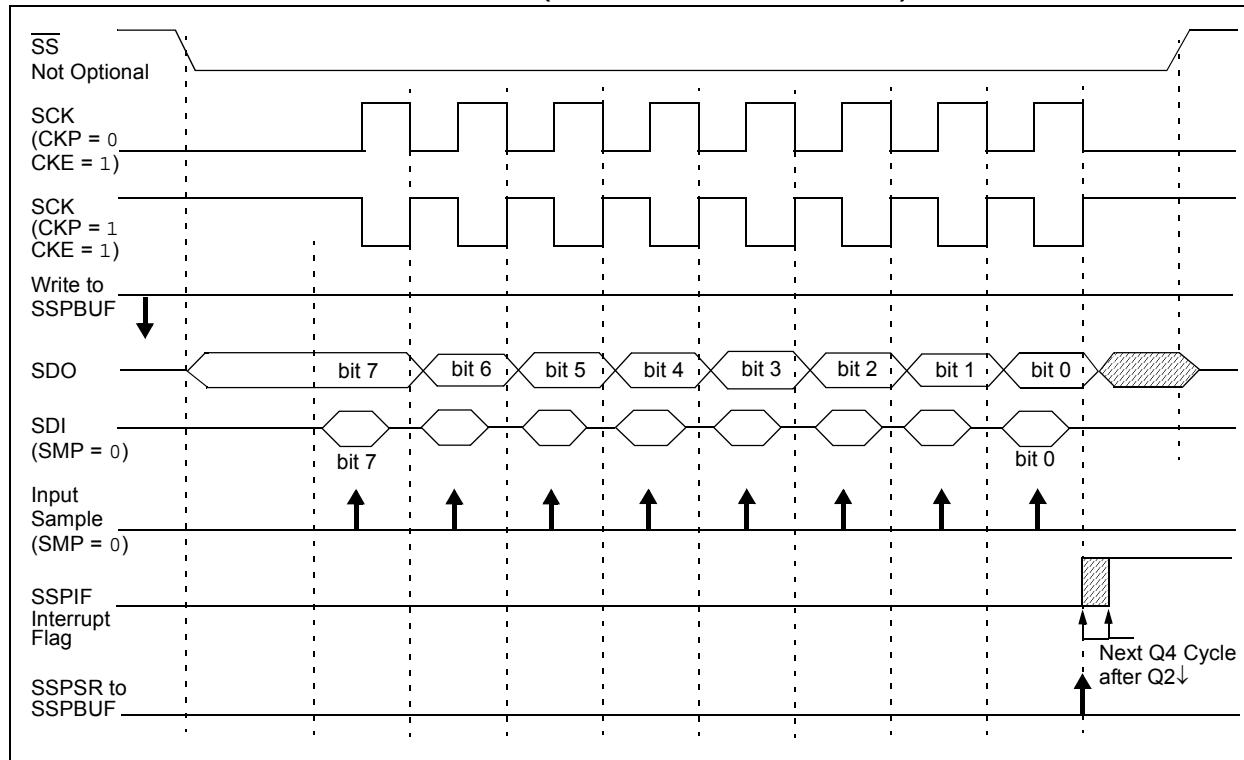


# PIC18F66K80 FAMILY

**FIGURE 21-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**



**FIGURE 21-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**



## 21.3.9 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in full-power mode; in the case of the Sleep mode, all clocks are halted.

In Idle modes, a clock is provided to the peripherals. That clock can be from the primary clock source, the secondary clock (SOSC oscillator) or the INTOSC source. See [Section 3.3 “Clock Sources and Oscillator Switching”](#) for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

If MSSP interrupt is enabled, it can wake the controller from Sleep mode, or one of the Idle modes, when the master completes sending data. If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

If the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in any power-managed mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set, and if enabled, will wake the device.

## 21.3.10 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

## 21.3.11 BUS MODE COMPATIBILITY

[Table 21-1](#) shows the compatibility between the standard SPI modes, and the states of the CKP and CKE control bits.

**TABLE 21-1: SPI BUS MODES**

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

There is also an SMP bit which controls when the data is sampled.

**TABLE 21-2: REGISTERS ASSOCIATED WITH SPI OPERATION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
TRISA	TRISA7	TRISA6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
SSPBUF	MSSP Receive Buffer/Transmit Register							
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
SSPSTAT	SMP	CKE	D/Ā	P	S	R/W	UA	BF
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD

**Legend:** Shaded cells are not used by the MSSP module in SPI mode.

# PIC18F66K80 FAMILY

## 21.4 I<sup>2</sup>C Mode

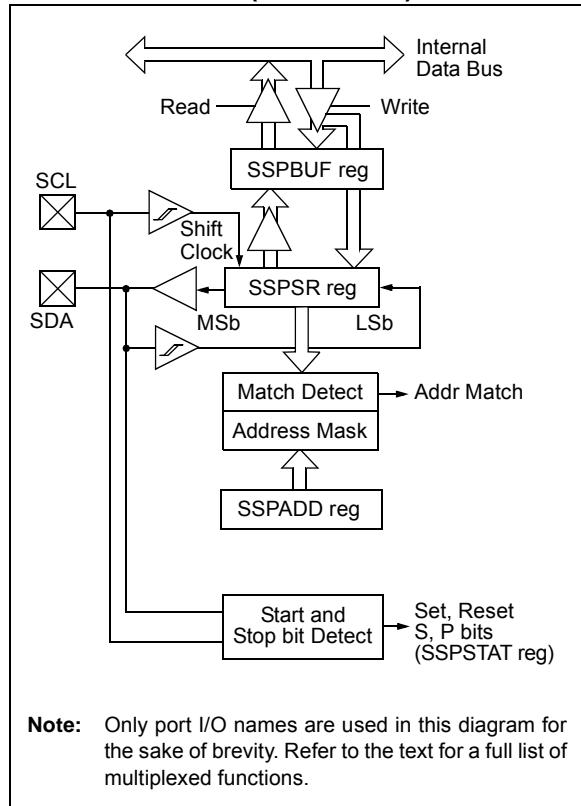
The MSSP module in I<sup>2</sup>C mode fully implements all master and slave functions (including general call support), and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial Clock (SCL) – RC3/REF0/SCL/SCK
- Serial Data (SDA) – RC4/SDA/SDI

The user must configure these pins as inputs by setting the associated TRIS bits.

**FIGURE 21-7: MSSP BLOCK DIAGRAM (I<sup>2</sup>C™ MODE)**



### 21.4.1 REGISTERS

The MSSP module has seven registers for I<sup>2</sup>C operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)
- I<sup>2</sup>C Slave Address Mask Register (SSPMSK)

SSPCON1, SSPCON2 and SSPSTAT are the control and status registers in I<sup>2</sup>C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

SSPADD contains the slave device address when the MSSP is configured in I<sup>2</sup>C Slave mode. When the MSSP is configured in Master mode, all eight bits of SSPADD act as the Baud Rate Generator reload value.

SSPMSK holds the slave address mask value when the module is configured for 7-Bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPADD; it is only accessible when the SSPM<3:0> bits are specifically set to permit access. Additional details are provided in [Section 21.4.3.4 “7-Bit Address Masking Mode”](#).

In receive operations, SSPSR and SSPBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

# PIC18F66K80 FAMILY

## REGISTER 21-3: SSPSTAT: MSSP STATUS REGISTER ( $I^2C$ <sup>TM</sup> MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P <sup>(1)</sup>	S <sup>(1)</sup>	R/W <sup>(2,3)</sup>	UA	BF
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	<b>SMP:</b> Slew Rate Control bit  <u>In Master or Slave mode:</u> 1 = Slew rate control is disabled for Standard Speed mode (100 kHz and 1 MHz) 0 = Slew rate control is enabled for High-Speed mode (400 kHz)
bit 6	<b>CKE:</b> SMBus Select bit  <u>In Master or Slave mode:</u> 1 = Enables SMBus specific inputs 0 = Disables SMBus specific inputs
bit 5	<b>D/A:</b> Data/Address bit  <u>In Master mode:</u> Reserved.  <u>In Slave mode:</u> 1 = Indicates that the last byte received or transmitted was data 0 = Indicates that the last byte received or transmitted was address
bit 4	<b>P:</b> Stop bit <sup>(1)</sup> 1 = Indicates that a Stop bit has been detected last 0 = Stop bit was not detected last
bit 3	<b>S:</b> Start bit <sup>(1)</sup> 1 = Indicates that a Start bit has been detected last 0 = Start bit was not detected last
bit 2	<b>R/W:</b> Read/Write Information bit <sup>(2,3)</sup>  <u>In Slave mode:</u> 1 = Read 0 = Write  <u>In Master mode:</u> 1 = Transmit is in progress 0 = Transmit is not in progress
bit 1	<b>UA:</b> Update Address bit (10-Bit Slave mode only) 1 = Indicates that the user needs to update the address in the SSPADD register 0 = Address does not need to be updated
bit 0	<b>BF:</b> Buffer Full Status bit  <u>In Transmit mode:</u> 1 = SSPBUF is full 0 = SSPBUF is empty  <u>In Receive mode:</u> 1 = SSPBUF is full (does not include the ACK and Stop bits) 0 = SSPBUF is empty (does not include the ACK and Stop bits)

**Note 1:** This bit is cleared on Reset and when SSPEN is cleared.

**2:** This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not ACK bit.

**3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Active mode.

# PIC18F66K80 FAMILY

## REGISTER 21-4: SSPCON1: MSSP CONTROL REGISTER 1 (I<sup>2</sup>C<sup>TM</sup> MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN <sup>(1)</sup>	CKP	SSPM3 <sup>(2)</sup>	SSPM2 <sup>(2)</sup>	SSPM1 <sup>(2)</sup>	SSPM0 <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      **WCOL:** Write Collision Detect bit

#### In Master Transmit mode:

1 = A write to the SSPBUF register was attempted while the I<sup>2</sup>C conditions were not valid for a transmission to be started (must be cleared in software)  
0 = No collision

#### In Slave Transmit mode:

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)  
0 = No collision

#### In Receive mode (Master or Slave modes):

This is a "don't care" bit.

bit 6      **SSPOV:** Receive Overflow Indicator bit

#### In Receive mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)  
0 = No overflow

#### In Transmit mode:

This is a "don't care" bit in Transmit mode.

bit 5      **SSPEN:** Master Synchronous Serial Port Enable bit<sup>(1)</sup>

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins  
0 = Disables serial port and configures these pins as I/O port pins

bit 4      **CKP:** SCK Release Control bit

#### In Slave mode:

1 = Releases clock  
0 = Holds clock low (clock stretch), used to ensure data setup time

#### In Master mode:

Unused in this mode.

bit 3-0      **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits<sup>(2)</sup>

1111 = I<sup>2</sup>C Slave mode, 10-bit address with Start and Stop bit interrupts enabled  
1110 = I<sup>2</sup>C Slave mode, 7-bit address with Start and Stop bit interrupts enabled  
1011 = I<sup>2</sup>C Firmware Controlled Master mode (slave Idle)  
1001 = Load SSPMSK register at SSPADD SFR address<sup>(3,4)</sup>  
1000 = I<sup>2</sup>C Master mode, clock = Fosc/(4 \* (SSPADD + 1))  
0111 = I<sup>2</sup>C Slave mode, 10-bit address  
0110 = I<sup>2</sup>C Slave mode, 7-bit address

**Note 1:** When enabled, the SDA and SCL pins must be configured as inputs.

**2:** Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

**3:** When SSPM<3:0> = 1001, any reads or writes to the SSPADD SFR address actually access the SSPMSK register.

**4:** This mode is only available when 7-Bit Address Masking mode is selected (MSSPMSK Configuration bit is '1').

# PIC18F66K80 FAMILY

## REGISTER 21-5: SSPCON2: MSSP CONTROL REGISTER 2 (I<sup>2</sup>C™ MASTER MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT <sup>(1)</sup>	ACKEN <sup>(2)</sup>	RCEN <sup>(2)</sup>	PEN <sup>(2)</sup>	RSEN <sup>(2)</sup>	SEN <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 | <b>GCEN:</b> General Call Enable bit<br>Unused in Master mode.  |
| bit 6 | <b>ACKSTAT:</b> Acknowledge Status bit (Master Transmit mode only)<br>1 = Acknowledge was not received from slave<br>0 = Acknowledge was received from slave  |
| bit 5 | <b>ACKDT:</b> Acknowledge Data bit (Master Receive mode only) <sup>(1)</sup><br>1 = Not Acknowledged<br>0 = Acknowledged  |
| bit 4 | <b>ACKEN:</b> Acknowledge Sequence Enable bit <sup>(2)</sup><br>1 = Initiates Acknowledge sequence on SDA and SCL pins and transmits ACKDT data bit;<br>automatically cleared by hardware<br>0 = Acknowledge sequence is Idle |
| bit 3 | <b>RCEN:</b> Receive Enable bit (Master Receive mode only) <sup>(2)</sup><br>1 = Enables Receive mode for I <sup>2</sup> C™<br>0 = Receive is Idle  |
| bit 2 | <b>PEN:</b> Stop Condition Enable bit <sup>(2)</sup><br>1 = Initiates Stop condition on SDA and SCL pins; automatically cleared by hardware<br>0 = Stop condition is Idle   |
| bit 1 | <b>RSEN:</b> Repeated Start Condition Enable bit <sup>(2)</sup><br>1 = Initiates Repeated Start condition on SDA and SCL pins; automatically cleared by hardware<br>0 = Repeated Start condition Idle                         |
| bit 0 | <b>SEN:</b> Start Condition Enable bit <sup>(2)</sup><br>1 = Initiates Start condition on SDA and SCL pins; automatically cleared by hardware<br>0 = Start condition Idle   |

- Note 1:** The value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.  
**2:** If the I<sup>2</sup>C module is active, these bits may not be set (no spooling) and the SSPBUF may not be written to (or writes to the SSPBUF are disabled).

# PIC18F66K80 FAMILY

## REGISTER 21-6: SSPCON2: MSSP CONTROL REGISTER 2 (I<sup>2</sup>C™ SLAVE MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT <sup>(1)</sup>	ACKEN <sup>(1)</sup>	RCEN <sup>(1)</sup>	PEN <sup>(1)</sup>	RSEN <sup>(1)</sup>	SEN <sup>(1)</sup>
bit 7							bit 0

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

bit 7	<b>GCEN:</b> General Call Enable bit 1 = Enables interrupt when a general call address (0000h) is received in the SSPSR 0 = General call address is disabled
bit 6	<b>ACKSTAT:</b> Acknowledge Status bit Unused in Slave mode.
bit 5	<b>ACKDT:</b> Acknowledge Data bit (Master Receive mode only) <sup>(1)</sup> 1 = Not Acknowledged 0 = Acknowledged
bit 4	<b>ACKEN:</b> Acknowledge Sequence Enable bit <sup>(1)</sup> 1 = Initiates Acknowledge sequence on SDA and SCL pins and transmits ACKDT data bit; automatically cleared by hardware 0 = Acknowledge sequence is Idle
bit 3	<b>RCEN:</b> Receive Enable bit (Master Receive mode only) <sup>(1)</sup> 1 = Enables Receive mode for I <sup>2</sup> C™ 0 = Receive is Idle
bit 2	<b>PEN:</b> Stop Condition Enable bit <sup>(1)</sup> 1 = Initiates Stop condition on SDA and SCL pins; automatically cleared by hardware 0 = Stop condition is Idle
bit 1	<b>RSEN:</b> Repeated Start Condition Enable bit <sup>(1)</sup> 1 = Initiates Repeated Start condition on SDA and SCL pins; automatically cleared by hardware 0 = Repeated Start condition is Idle
bit 0	<b>SEN:</b> Stretch Enable bit <sup>(1)</sup> 1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled) 0 = Clock stretching is disabled

**Note 1:** If the I<sup>2</sup>C module is active, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

## REGISTER 21-7: SSPMSK: I<sup>2</sup>C™ SLAVE ADDRESS MASK REGISTER (7-BIT MASKING MODE)<sup>(1)</sup>

| R/W-1               |
|-------|-------|-------|-------|-------|-------|-------|---------------------|
| MSK7  | MSK6  | MSK5  | MSK4  | MSK3  | MSK2  | MSK1  | MSK0 <sup>(2)</sup> |
| bit 7 |       |       |       |       |       |       | bit 0               |

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

bit 7-0	<b>MSK&lt;7:0&gt;:</b> Slave Address Mask Select bit <sup>(2)</sup> 1 = Masking of corresponding bit of SSPADD is enabled 0 = Masking of corresponding bit of SSPADD is disabled
---------	--

**Note 1:** This register shares the same SFR address as SSPADD and is only addressable in select MSSP operating modes. See [Section 21.4.3.4 “7-Bit Address Masking Mode”](#) for more details.  
**2:** MSK0 is not used as a mask bit in 7-bit addressing.

## 21.4.2 OPERATION

The MSSP module functions are enabled by setting the MSSP Enable bit, SSPEN (SSPCON1<5>).

The SSPCON1 register allows control of the I<sup>2</sup>C operation. Four mode selection bits (SSPCON1<3:0>) allow one of the following I<sup>2</sup>C modes to be selected:

- I<sup>2</sup>C Master mode, clock
- I<sup>2</sup>C Slave mode (7-bit address)
- I<sup>2</sup>C Slave mode (10-bit address)
- I<sup>2</sup>C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I<sup>2</sup>C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I<sup>2</sup>C Firmware Controlled Master mode, slave is Idle

Selection of any I<sup>2</sup>C mode with the SSPEN bit set forces the SCL and SDA pins to be open-drain, provided these pins are programmed as inputs by setting the appropriate TRISC bit. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

## 21.4.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The I<sup>2</sup>C Slave mode hardware will always generate an interrupt on an address match. Address masking will allow the hardware to generate an interrupt for more than one address (up to 31 in 7-bit addressing and up to 63 in 10-bit addressing). Through the mode select bits, the user can also choose to interrupt on Start and Stop bits.

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge (ACK) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this ACK pulse:

- The Buffer Full bit, BF (SSPSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPCON1<6>), was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit, SSPIF, is set. The BF bit is cleared by reading the SSPBUF register, while bit, SSPOV, is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I<sup>2</sup>C specification, as well as the requirement of the MSSP module, are shown in timing Parameter 100 and Parameter 101.

## 21.4.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8 bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register, SSPSR<7:1>, is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit, BF, is set.
3. An ACK pulse is generated.
4. The MSSP Interrupt Flag bit, SSPIF, is set (and interrupt is generated, if enabled) on the falling edge of the ninth SCL pulse.

In 10-Bit Addressing mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. The R/W (SSPSTAT<2>) bit must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '11110 A9 A8 0', where 'A9' and 'A8' are the two MSbs of the address. The sequence of events for 10-bit addressing is as follows, with Steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits, SSPIF, BF and UA, are set on address match).
2. Update the SSPADD register with second (low) byte of address (clears bit, UA, and releases the SCL line).
3. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.
4. Receive second (low) byte of address (bits, SSPIF, BF and UA, are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit, UA.
6. Read the SSPBUF register (clears bit, BF) and clear flag bit SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits, SSPIF and BF, are set).
9. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.

# PIC18F66K80 FAMILY

## 21.4.3.2 Address Masking Modes

Masking an address bit causes that bit to become a “don’t care”. When one address bit is masked, two addresses will be Acknowledged and cause an interrupt. It is possible to mask more than one address bit at a time, which greatly expands the number of addresses Acknowledged.

The I<sup>2</sup>C slave behaves the same way, whether address masking is used or not. However, when address masking is used, the I<sup>2</sup>C slave can Acknowledge multiple addresses and cause interrupts. When this occurs, it is necessary to determine which address caused the interrupt by checking the SSPBUF.

The PIC18F66K80 family of devices is capable of using two different Address Masking modes in I<sup>2</sup>C slave operation: 5-Bit Address Masking and 7-Bit Address Masking. The Masking mode is selected at device configuration using the MSSPMSK Configuration bit. The default device configuration is 7-Bit Address Masking.

Both Masking modes, in turn, support address masking of 7-bit and 10-bit addresses. The combination of Masking modes and addresses provide different ranges of Acknowledgable addresses for each combination.

While both Masking modes function in roughly the same manner, the way they use address masks are different.

## 21.4.3.3 5-Bit Address Masking Mode

As the name implies, 5-Bit Address Masking mode uses an address mask of up to 5 bits to create a range of addresses to be Acknowledged, using bits, 5 through 1,

of the incoming address. This allows the module to Acknowledge up to 31 addresses when using 7-bit addressing, or 63 addresses with 10-bit addressing (see [Example 21-2](#)). This Masking mode is selected when the MSSPMSK Configuration bit is programmed ('0').

The address mask in this mode is stored in the SSPCON2 register, which stops functioning as a control register in I<sup>2</sup>C Slave mode ([Register 21-6](#)). In 7-Bit Addressing mode, address mask bits, ADMSK<5:1> (SSPCON2<5:1>), mask the corresponding address bits in the SSPADD register. For any ADMSKx bits that are set (ADMSK<n> = 1), the corresponding address bit is ignored (SSPADD<n> = x). For the module to issue an address Acknowledge, it is sufficient to match only on addresses that do not have an active address mask.

In 10-Bit Addressing mode, bits, ADMSK<5:2>, mask the corresponding address bits in the SSPADD register. In addition, ADMSK1 simultaneously masks the two LSbs of the address (SSPADD<1:0>). For any ADMSKx bits that are active (ADMSK<n> = 1), the corresponding address bit is ignored (SPxADD<n> = x). Also note that although in 10-Bit Addressing mode, the upper address bits reuse part of the SSPADD register bits. The address mask bits do not interact with those bits; they only affect the lower address bits.

- Note 1:** ADMSK1 masks the two Least Significant bits of the address.
- 2:** The two Most Significant bits of the address are not affected by address masking.

## EXAMPLE 21-2: ADDRESS MASKING EXAMPLES IN 5-BIT MASKING MODE

### 7-Bit Addressing:

SSPADD<7:1> = A0h (1010000) (SSPADD<0> is assumed to be '0')

ADMSK<5:1> = 00111

Addresses Acknowledged: A0h, A2h, A4h, A6h, A8h, AAh, ACh, AEh

### 10-Bit Addressing:

SSPADD<7:0> = A0h (10100000) (The two MSb of the address are ignored in this example, since they are not affected by masking)

ADMSK<5:1> = 00111

Addresses Acknowledged: A0h, A1h, A2h, A3h, A4h, A5h, A6h, A7h, A8h, A9h, AAh, ABh, ACh, ADh, AEh, AFh

#### 21.4.3.4 7-Bit Address Masking Mode

Unlike 5-bit masking, 7-Bit Address Masking mode uses a mask of up to 8 bits (in 10-bit addressing) to define a range of addresses that can be Acknowledged, using the lowest bits of the incoming address. This allows the module to Acknowledge up to 127 different addresses with 7-bit addressing, or 255 with 10-bit addressing (see [Example 21-3](#)). This mode is the default configuration of the module, which is selected when MSSPMSK is unprogrammed ('1').

The address mask for 7-Bit Address Masking mode is stored in the SSPMSK register, instead of the SSPCON2 register. SSPMSK is a separate hardware register within the module, but it is not directly addressable. Instead, it shares an address in the SFR space with the SSPADD register. To access the SSPMSK register, it is necessary to select MSSP mode, '1001' (SSPCON1<3:0> = 1001) and then read or write to the location of SSPADD.

To use 7-Bit Address Masking mode, it is necessary to initialize SSPMSK with a value before selecting the I<sup>2</sup>C Slave Addressing mode. Thus, the required sequence of events is:

1. Select SSPMSK Access mode (SSPCON2<3:0> = 1001).
2. Write the mask value to the appropriate SSPADD register address (FC8h).
3. Set the appropriate I<sup>2</sup>C Slave mode (SSPCON2<3:0> = 0111 for 10-bit addressing, 0110 for 7-bit addressing).

#### EXAMPLE 21-3: ADDRESS MASKING EXAMPLES IN 7-BIT MASKING MODE

##### 7-Bit Addressing:

SSPADD<7:1> = 1010 000

SSPMSK<7:1> = 1111 001

Addresses Acknowledged = ACh, A8h, A4h, A0h

##### 10-Bit Addressing:

SSPADD<7:0> = 1010 0000 (The two MSb are ignored in this example since they are not affected)

SSPMSK<5:1> = 1111 0011

Addresses Acknowledged = ACh, A8h, A4h, A0h

Setting or clearing mask bits in SSPMSK behaves in the opposite manner of the ADMSKx bits in 5-Bit Address Masking mode. That is, clearing a bit in SSPMSK causes the corresponding address bit to be masked; setting the bit requires a match in that position. SSPMSK resets to all '1's upon any Reset condition and, therefore, has no effect on the standard MSSP operation until written with a mask value.

With 7-bit addressing, SSPMSK<7:1> bits mask the corresponding address bits in the SSPADD register. For any SSPMSK bits that are active (SSPMSK<n> = 0), the corresponding SSPADD address bit is ignored (SSPADD<n> = x). For the module to issue an address Acknowledge, it is sufficient to match only on addresses that do not have an active address mask.

With 10-bit addressing, SSPMSK<7:0> bits mask the corresponding address bits in the SSPADD register. For any SSPMSK bits that are active (= 0), the corresponding SSPADD address bit is ignored (SSPADD<n> = x).

**Note:** The two Most Significant bits of the address are not affected by address masking.

# PIC18F66K80 FAMILY

---

## 21.4.3.5 Reception

When the R/W bit of the address byte is clear and an address match occurs, the R/W bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low (ACK).

When the address byte overflow condition exists, then the no Acknowledge (ACK) pulse is given. An overflow condition is defined as either bit, BF (SSPSTAT<0>), is set or bit, SSPOV (SSPCON1<6>), is set.

An MSSP interrupt is generated for each data transfer byte. The interrupt flag bit, SSPIF, must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON2<0> = 1), SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit, CKP (SSPCON1<4>). See [Section 21.4.4 “Clock Stretching”](#) for more details.

## 21.4.3.6 Transmission

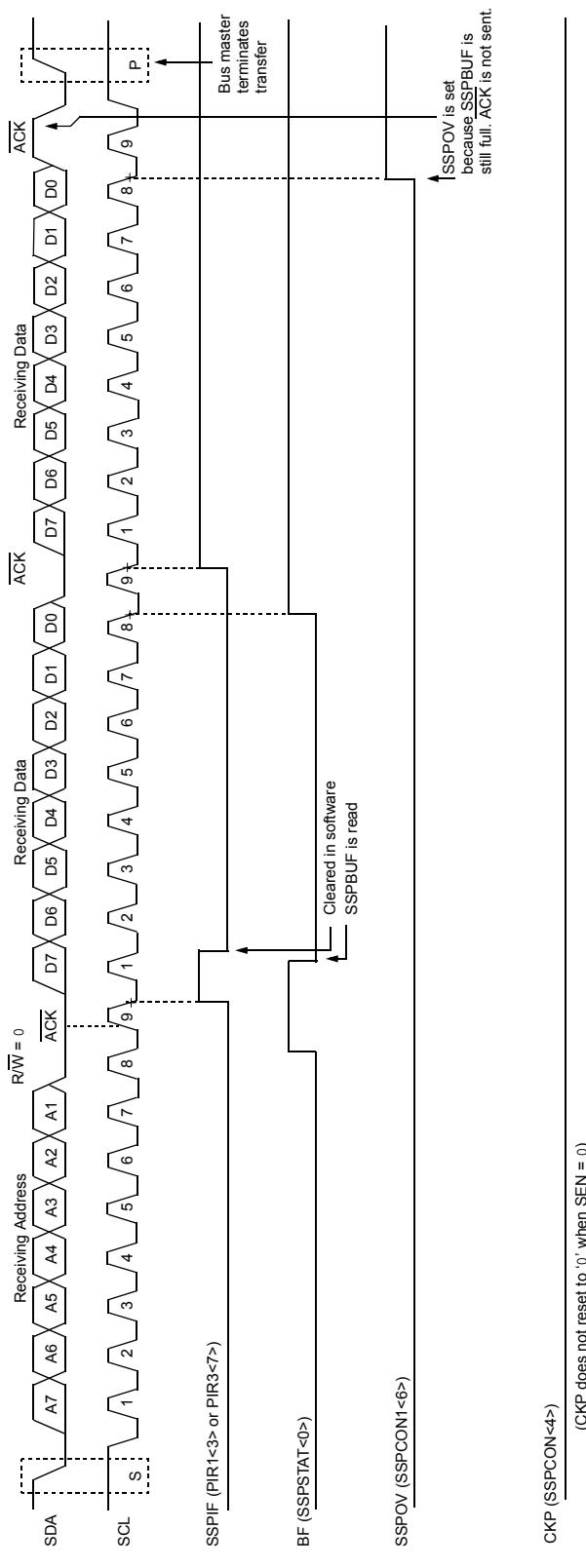
When the R/W bit of the incoming address byte is set and an address match occurs, the R/W bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The ACK pulse will be sent on the ninth bit and pin SCL is held low regardless of SEN (see [Section 21.4.4 “Clock Stretching”](#) for more details). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then, the SCL pin should be enabled by setting bit, CKP (SSPCON1<4>). The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time ([Figure 21-10](#)).

The ACK pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not ACK), then the data transfer is complete. In this case, when the ACK is latched by the slave, the slave logic is reset and the slave monitors for another occurrence of the Start bit. If the SDA line was low (ACK), the next transmit data must be loaded into the SSPBUF register. Again, the SCL pin must be enabled by setting bit, CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

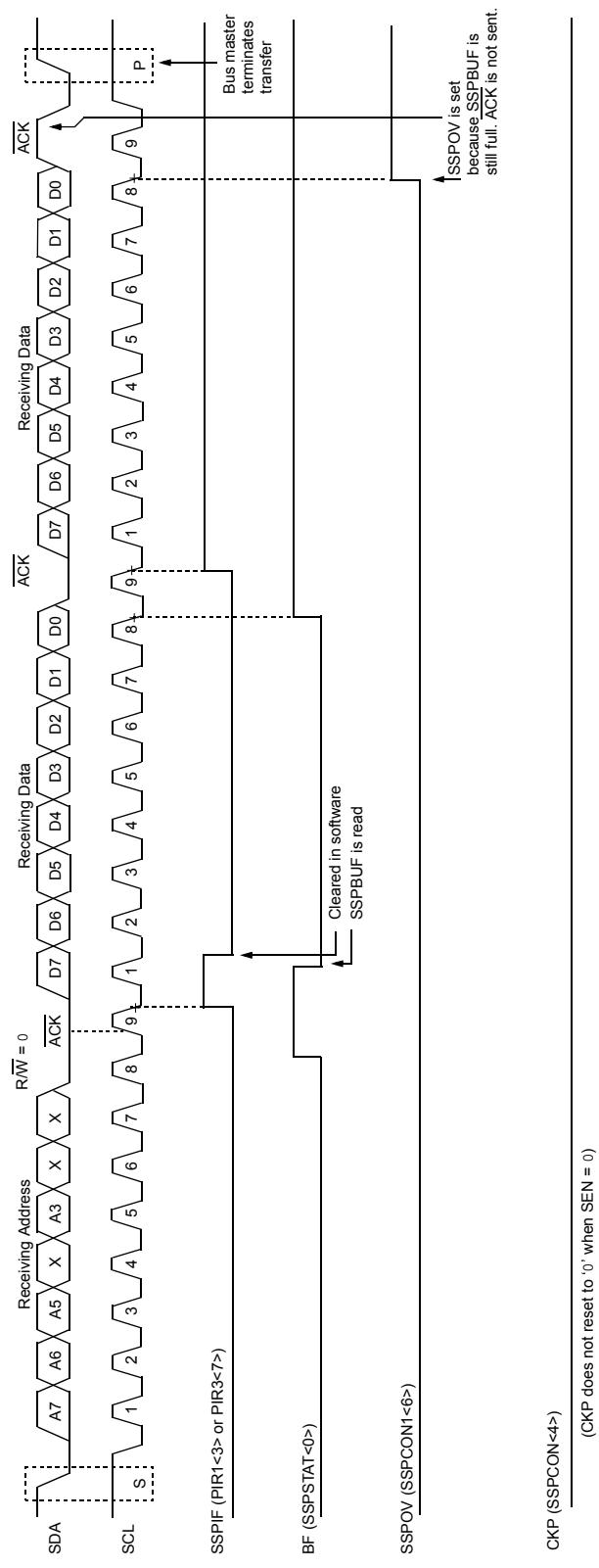
# PIC18F66K80 FAMILY

FIGURE 21-8:  $\text{I}^2\text{C}$ <sup>TM</sup> SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)



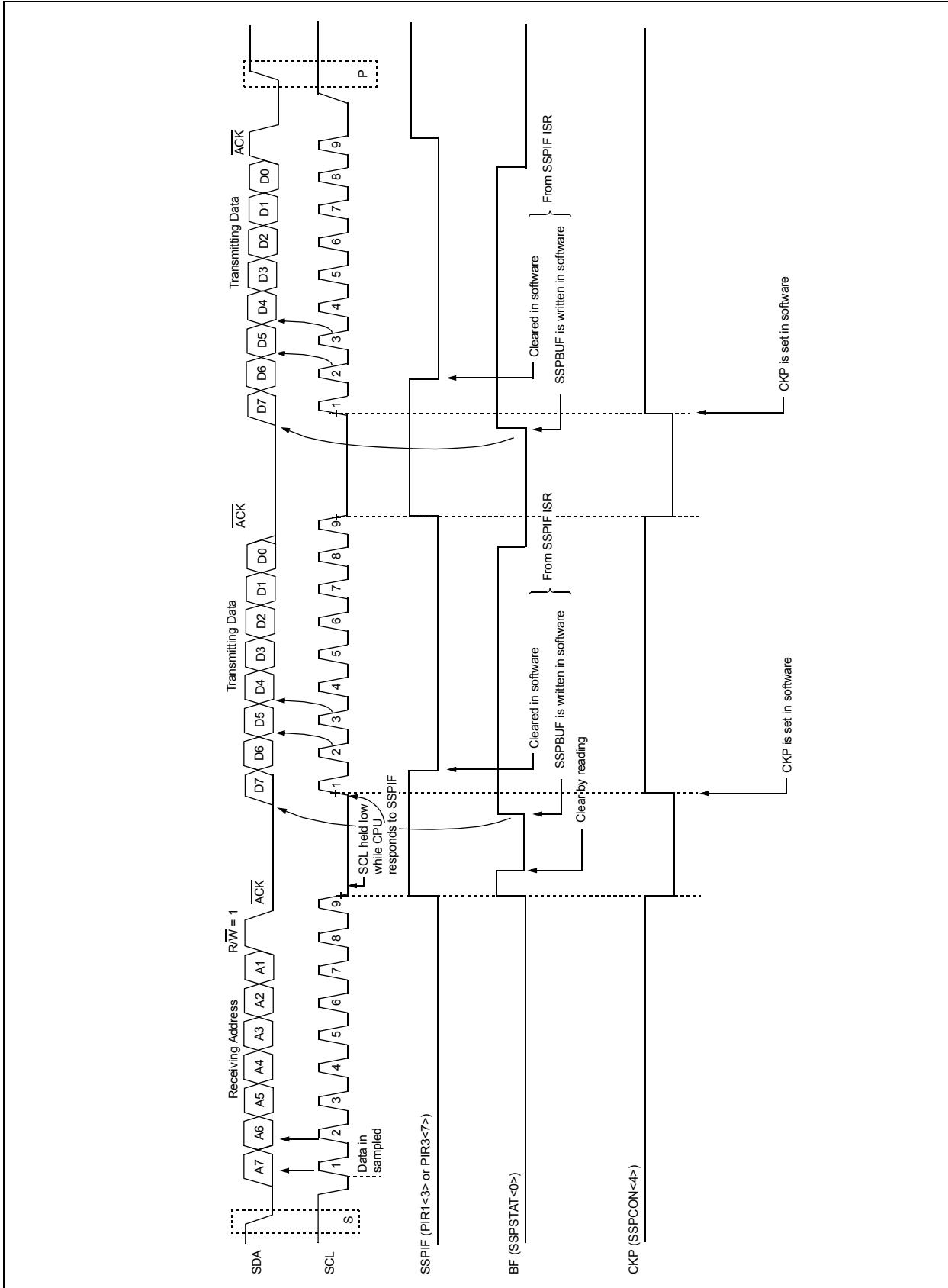
# PIC18F66K80 FAMILY

**FIGURE 21-9: I<sup>2</sup>C™ SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01011  
(RECEPTION, 7-BIT ADDRESS)**



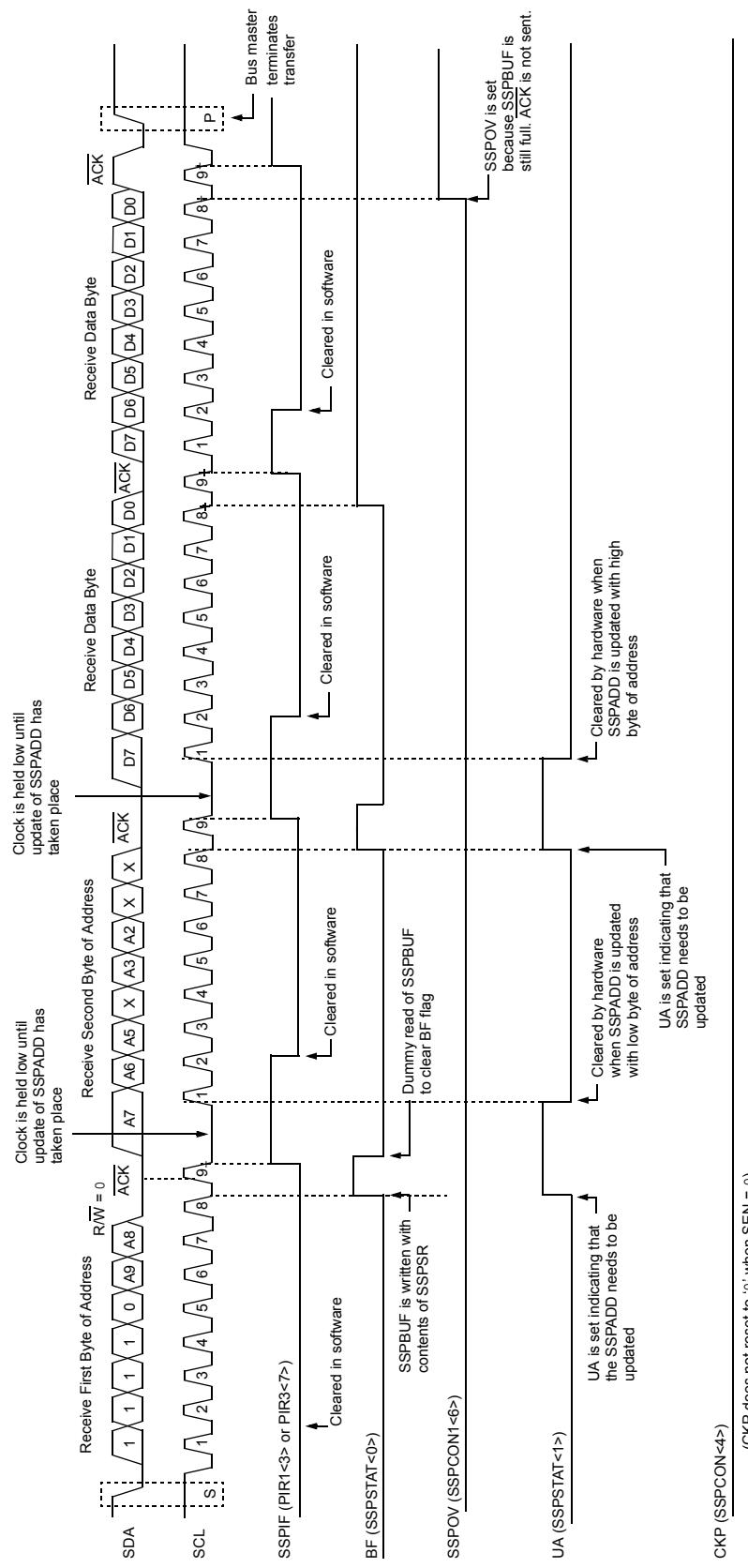
- Note 1:** x = Don't care (i.e., address bit can either be a '1' or a '0').  
**Note 2:** In this example, an address equal to A7.A6.A5.X.A3.X will be Acknowledged and cause an interrupt.

**FIGURE 21-10: I<sup>2</sup>C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**



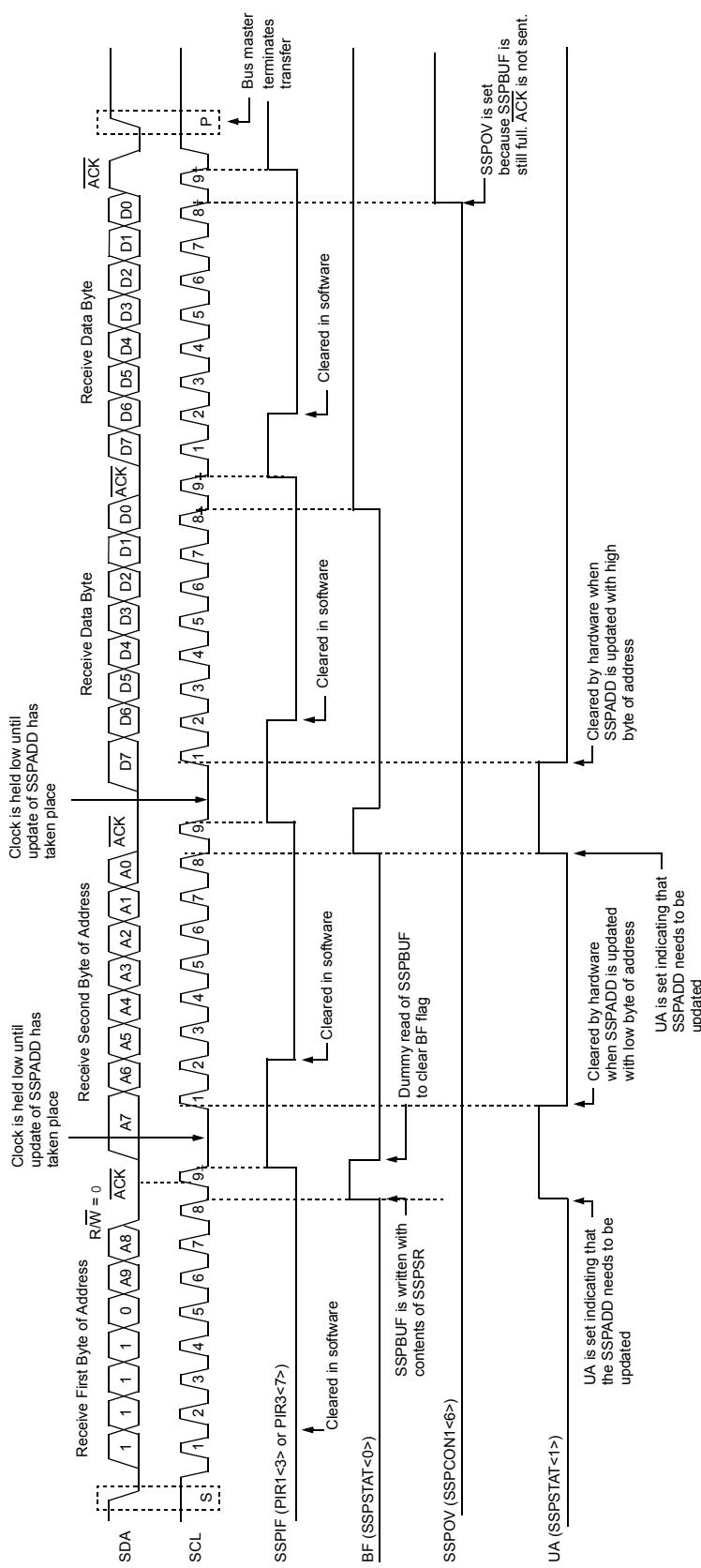
# PIC18F66K80 FAMILY

**FIGURE 21-11: I<sup>2</sup>C™ SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01001  
(RECEPTION, 10-BIT ADDRESS)**



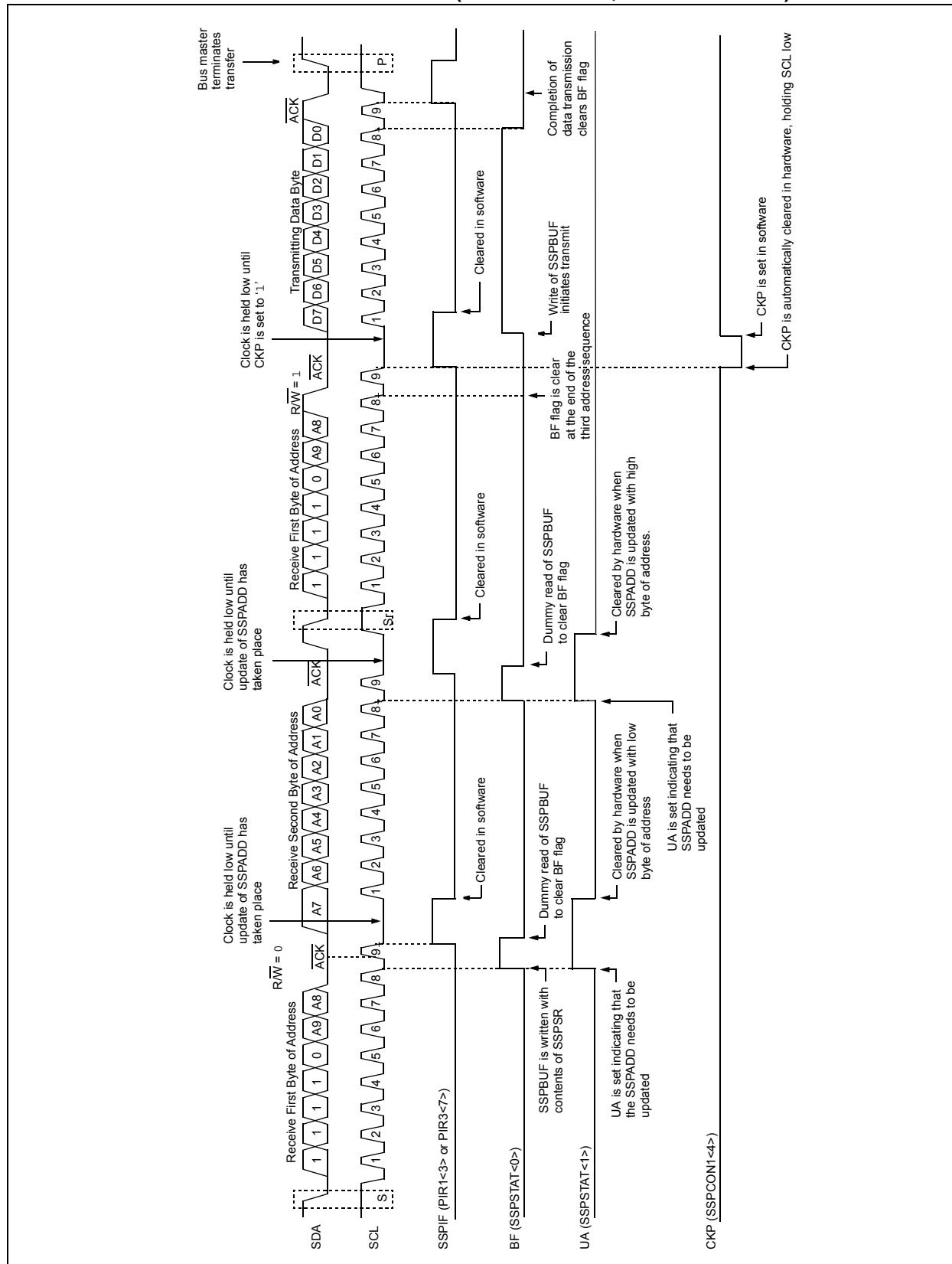
- Note 1:**  $x$  = Don't care (i.e., address bit can either be a '1' or a '0').  
**2:** In this example, an address equal to A9.A8.A7.A6.A5.XA3.A2.XX will be Acknowledged and cause an interrupt.  
**3:** Note that the Most Significant bits of the address are not affected by the bit masking.

**FIGURE 21-12: I<sup>2</sup>C<sup>TM</sup> SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)**



# PIC18F66K80 FAMILY

**FIGURE 21-13: I<sup>2</sup>C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**



## 21.4.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

### 21.4.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the ACK sequence, if the BF bit is set, the CKP bit in the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP bit being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 21-15).

- Note 1:** If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.
- 2:** The CKP bit can be set in software, regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

### 21.4.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = 1)

In 10-Bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/W bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

**Note:** If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs, and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

### 21.4.4.3 Clock Stretching for 7-Bit Slave Transmit Mode

The 7-Bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 21-10).

**Note 1:** If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

**2:** The CKP bit can be set in software regardless of the state of the BF bit.

### 21.4.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high-order bits of the 10-bit address and the R/W bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 21-13).

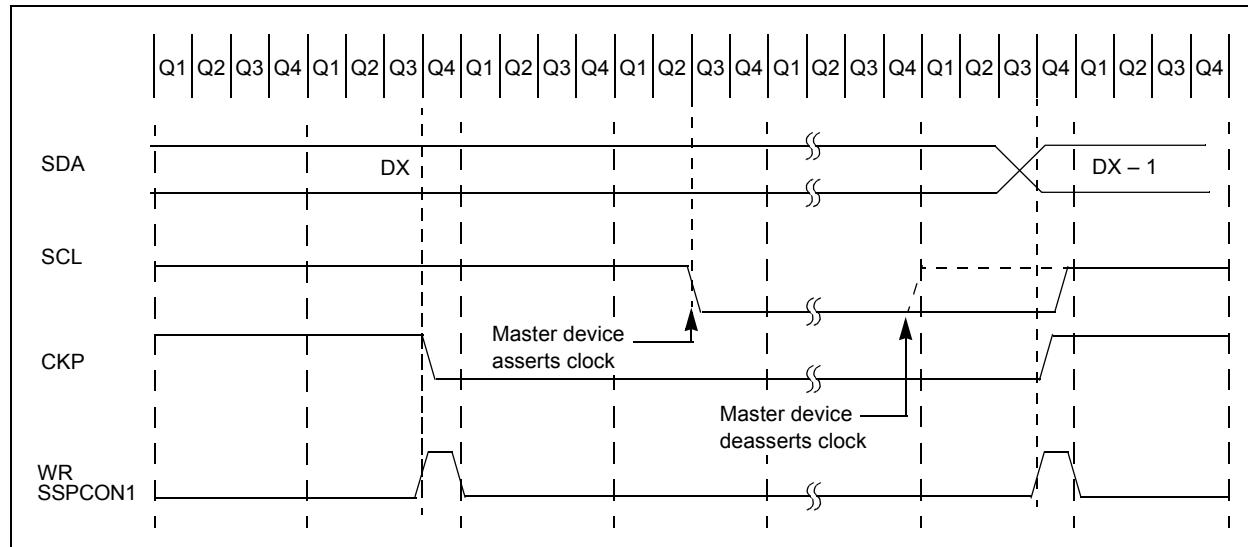
# PIC18F66K80 FAMILY

## 21.4.4.5 Clock Synchronization and the CKP bit

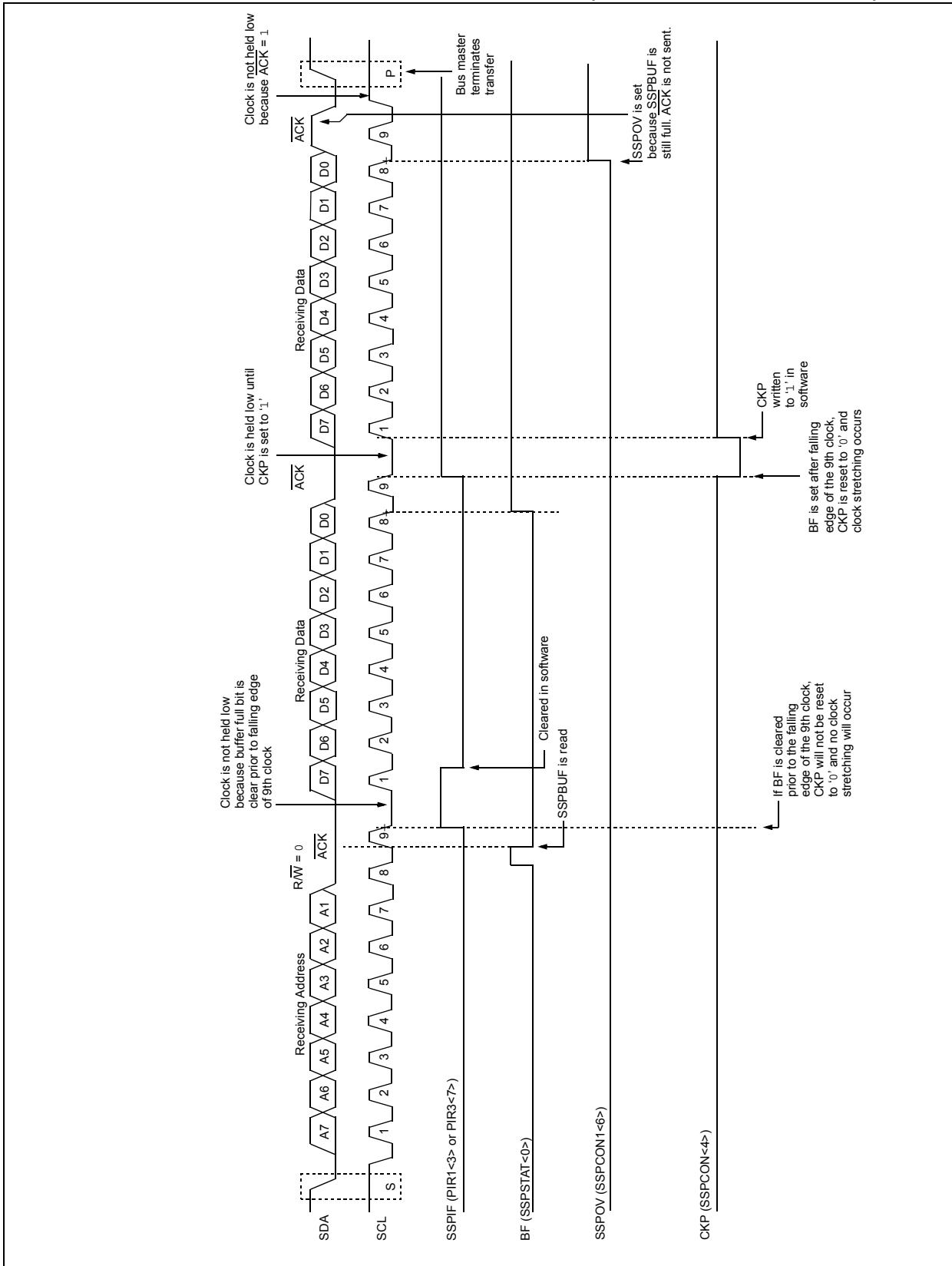
When the CKP bit is cleared, the SCL output is forced to '0'. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I<sup>2</sup>C master device has

already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I<sup>2</sup>C bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 21-14).

**FIGURE 21-14: CLOCK SYNCHRONIZATION TIMING**

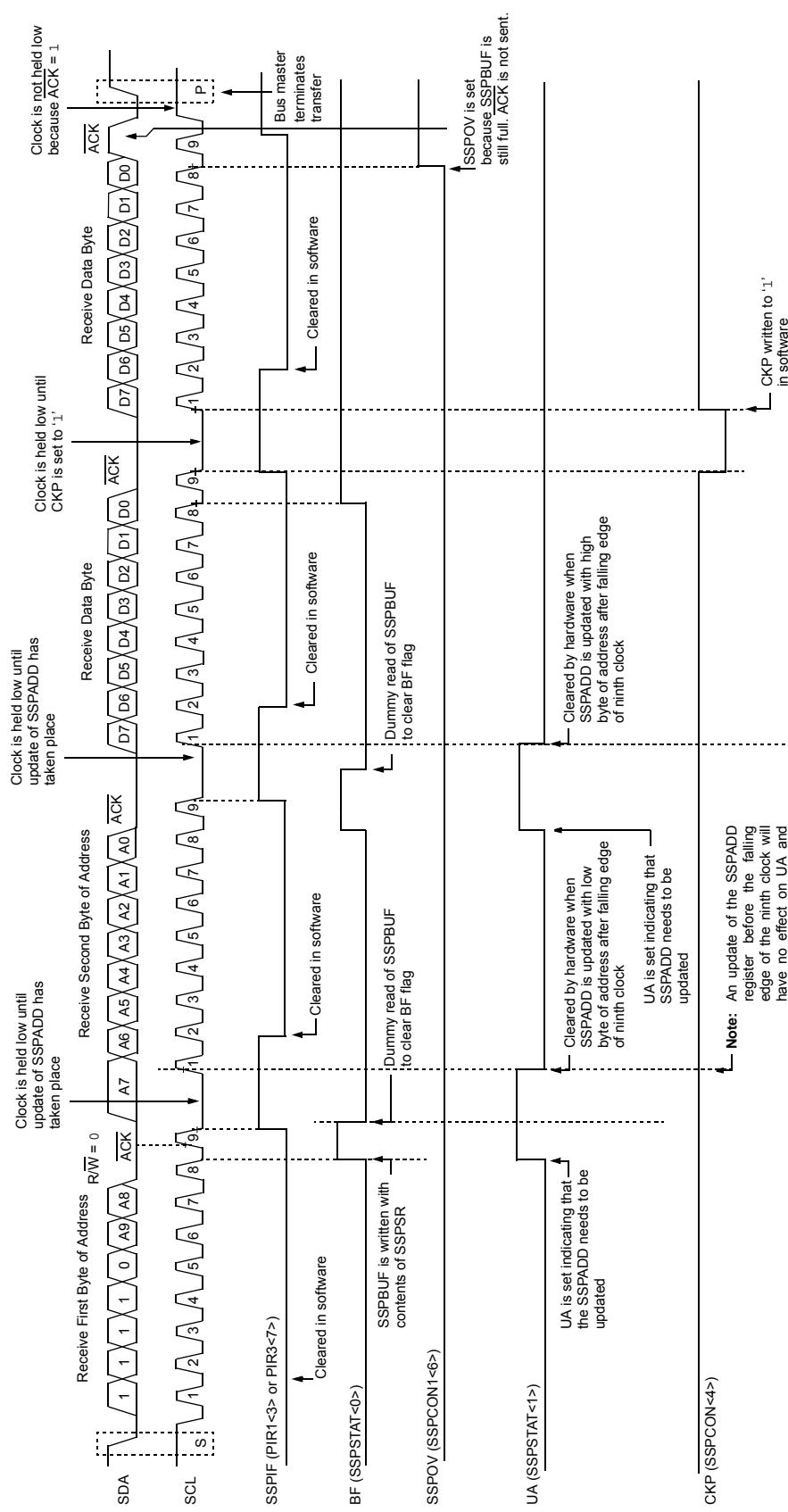


**FIGURE 21-15: I<sup>2</sup>C<sup>TM</sup> SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)**



# PIC18F66K80 FAMILY

**FIGURE 21-16: I<sup>2</sup>C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)**



## 21.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I<sup>2</sup>C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an Acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the I<sup>2</sup>C protocol. It consists of all '0's with R/W = 0.

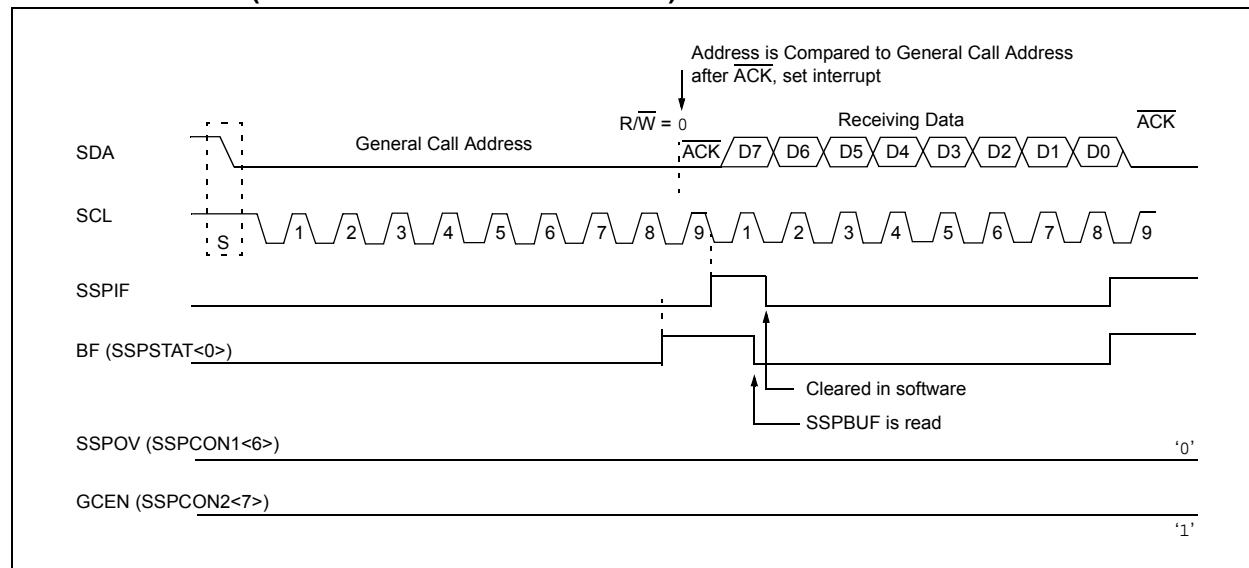
The general call address is recognized when the General Call Enable bit, GCEN, is enabled (SSPCON2<7> set). Following a Start bit detect, eight bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eighth bit), and on the falling edge of the ninth bit (ACK bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device-specific or a general call address.

In 10-Bit Addressing mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when the GCEN bit is set, while the slave is configured in 10-Bit Addressing mode, then the second half of the address is not necessary, the UA bit will not be set and the slave will begin receiving data after the Acknowledge (Figure 21-17).

**FIGURE 21-17: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE  
(7 OR 10-BIT ADDRESSING MODE)**



# PIC18F66K80 FAMILY

## 21.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPMx bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware if the TRIS bits are set.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I<sup>2</sup>C bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I<sup>2</sup>C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

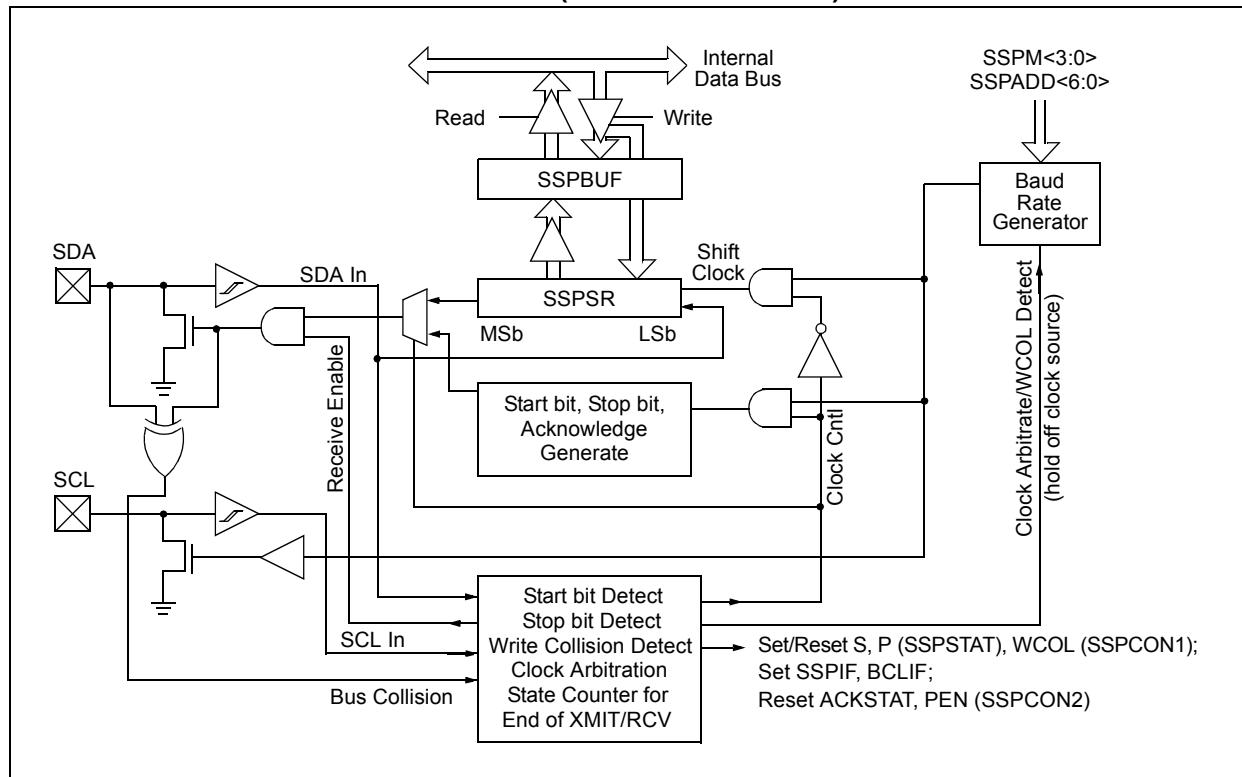
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I<sup>2</sup>C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

**Note:** The MSSP module, when configured in I<sup>2</sup>C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause the MSSP Interrupt Flag bit, SSPIF, to be set (and MSSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmitted
- Repeated Start

**FIGURE 21-18: MSSP BLOCK DIAGRAM (I<sup>2</sup>C™ MASTER MODE)**



## 21.4.6.1 I<sup>2</sup>C™ Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I<sup>2</sup>C bus will not be released.

In Master Transmitter mode, serial data is output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In this case, the R/W bit will be logic '0'. Serial data is transmitted, 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address, followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received, 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

The Baud Rate Generator, used for the SPI mode operation, is used to set the SCL clock frequency for either 100 kHz, 400 kHz or 1 MHz I<sup>2</sup>C operation. See [Section 21.4.7 "Baud Rate"](#) for more details.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start Enable bit, SEN (SSPCON2<0>).
2. SSPIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPBUF with the slave address to transmit.
4. Address is shifted out the SDA pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a Stop condition by setting the Stop Enable bit, PEN (SSPCON2<2>).
12. Interrupt is generated once the Stop condition is complete.

# PIC18F66K80 FAMILY

## 21.4.7 BAUD RATE

In I<sup>2</sup>C Master mode, the Baud Rate Generator (BRG) reload value is placed in the 8 bits of the SSPADD register (Figure 21-19). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (Tcy) on the Q2 and Q4 clocks. In I<sup>2</sup>C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 21-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD. The SSPADD BRG value of 00h is not supported.

FIGURE 21-19: BAUD RATE GENERATOR BLOCK DIAGRAM

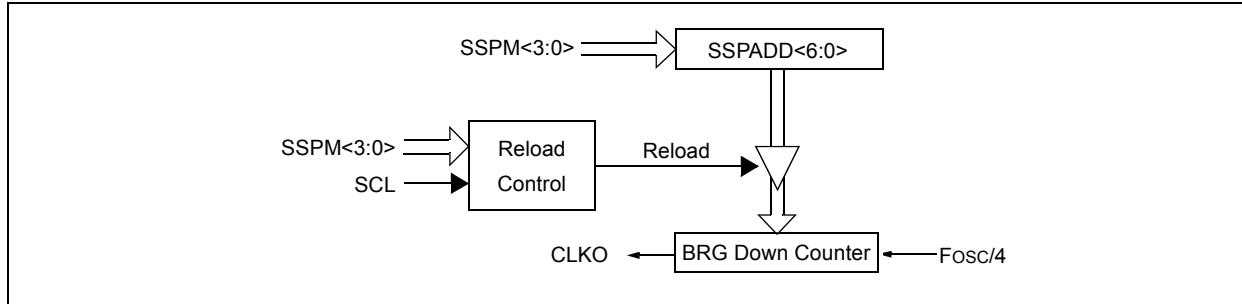


TABLE 21-3: I<sup>2</sup>C™ CLOCK RATE w/BRG

Fosc	Fcy	Fcy * 2	BRG Value	Fscl (2 Rollovers of BRG)
40 MHz	10 MHz	20 MHz	18h	400 kHz <sup>(1)</sup>
40 MHz	10 MHz	20 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	20 MHz	63h	100 kHz
16 MHz	4 MHz	8 MHz	09h	400 kHz <sup>(1)</sup>
16 MHz	4 MHz	8 MHz	0Ch	308 kHz
16 MHz	4 MHz	8 MHz	27h	100 kHz
4 MHz	1 MHz	2 MHz	02h	333 kHz <sup>(1)</sup>
4 MHz	1 MHz	2 MHz	09h	100 kHz
16 MHz <sup>(2)</sup>	4 MHz	8 MHz	03h	1 MHz <sup>(1)</sup>

Note 1: The I<sup>2</sup>C interface does not conform to the 400 kHz I<sup>2</sup>C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

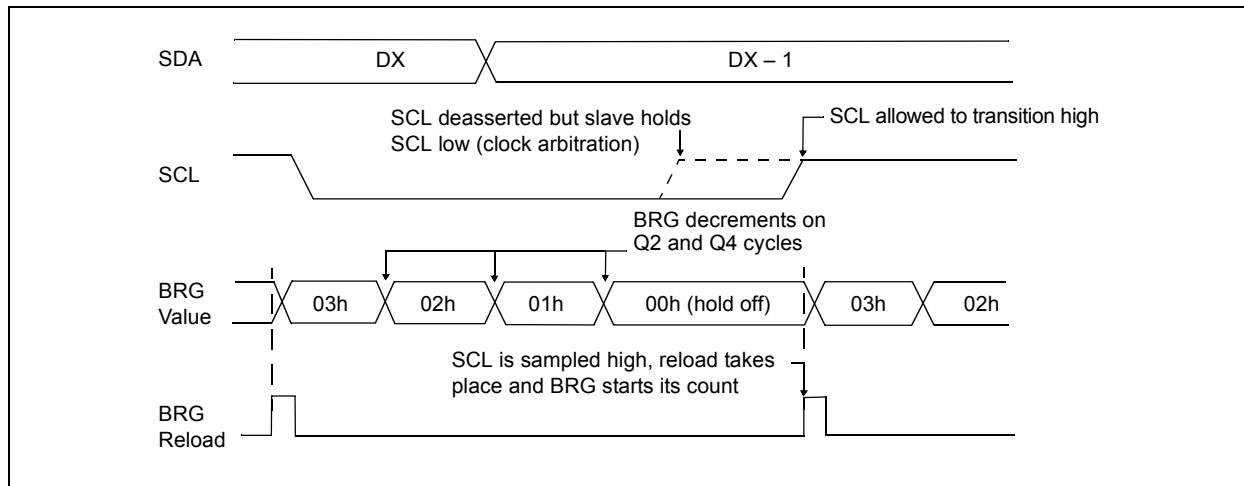
2: A minimum 16-MHz Fosc is required for 1 MHz I<sup>2</sup>C.

## 21.4.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the

SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 21-20).

**FIGURE 21-20: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**



# PIC18F66K80 FAMILY

## 21.4.8 I<sup>2</sup>C™ MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit (SSPSTAT<3>) to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit (SSPCON2<0>) will be automatically cleared by hardware. The Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

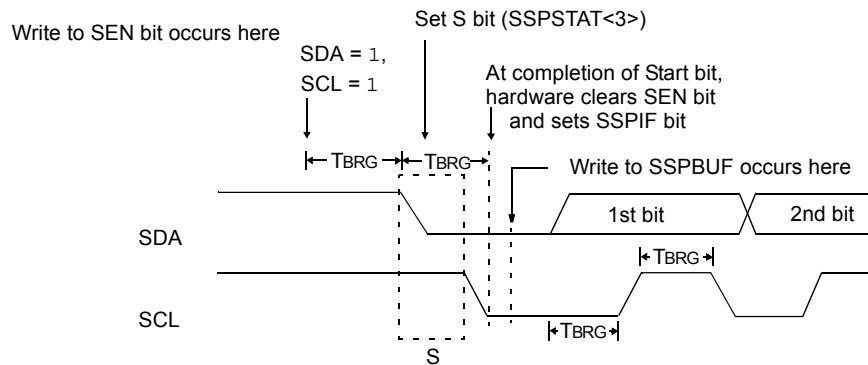
**Note:** If, at the beginning of the Start condition, the SDA and SCL pins are already sampled low or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I<sup>2</sup>C module is reset into its Idle state.

### 21.4.8.1 WCOL Status Flag

If the user writes the SSPBUF when a Start sequence is in progress, the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

**Note:** Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the Start condition is complete.

**FIGURE 21-21: FIRST START BIT TIMING**



## 21.4.9 I<sup>2</sup>C™ MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I<sup>2</sup>C logic module is in the Idle state. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded with the contents of SSPADD<5:0> and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, and if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. Following this, the RSEN bit (SSPCON2<1>) will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit (SSPSTAT<3>) will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

**Note 1:** If RSEN is programmed while any other event is in progress, it will not take effect.

**2:** A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low-to-high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

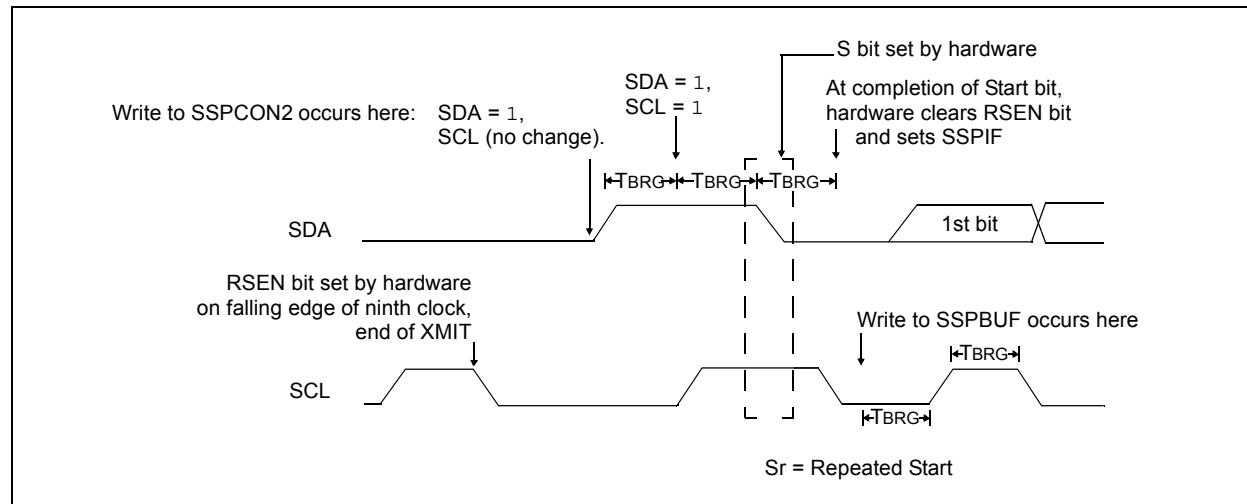
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

### 21.4.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**Note:** Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

**FIGURE 21-22: REPEATED START CONDITION WAVEFORM**



# PIC18F66K80 FAMILY

---

## 21.4.10 I<sup>2</sup>C<sup>TM</sup> MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address, is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification Parameter 106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification Parameter 107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an ACK bit during the ninth bit time if an address match occurred, or if data was received properly. The status of ACK is written into the ACKDT bit on the falling edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared; if not, the bit is set. After the ninth clock, the SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 21-23).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF flag is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

### 21.4.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT<0>) is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

### 21.4.10.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur) after 2 Tcy after the SSPBUF write. If SSPBUF is rewritten within 2 Tcy, the WCOL bit is set and SSPBUF is updated. This may result in a corrupted transfer.

The user should verify that the WCOL bit is clear after each write to SSPBUF to ensure the transfer is correct. In all cases, WCOL must be cleared in software.

### 21.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge (ACK = 0) and is set when the slave does not Acknowledge (ACK = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

## 21.4.11 I<sup>2</sup>C<sup>TM</sup> MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPCON2<3>).

**Note:** The MSSP module must be in an inactive state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting, and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>).

### 21.4.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

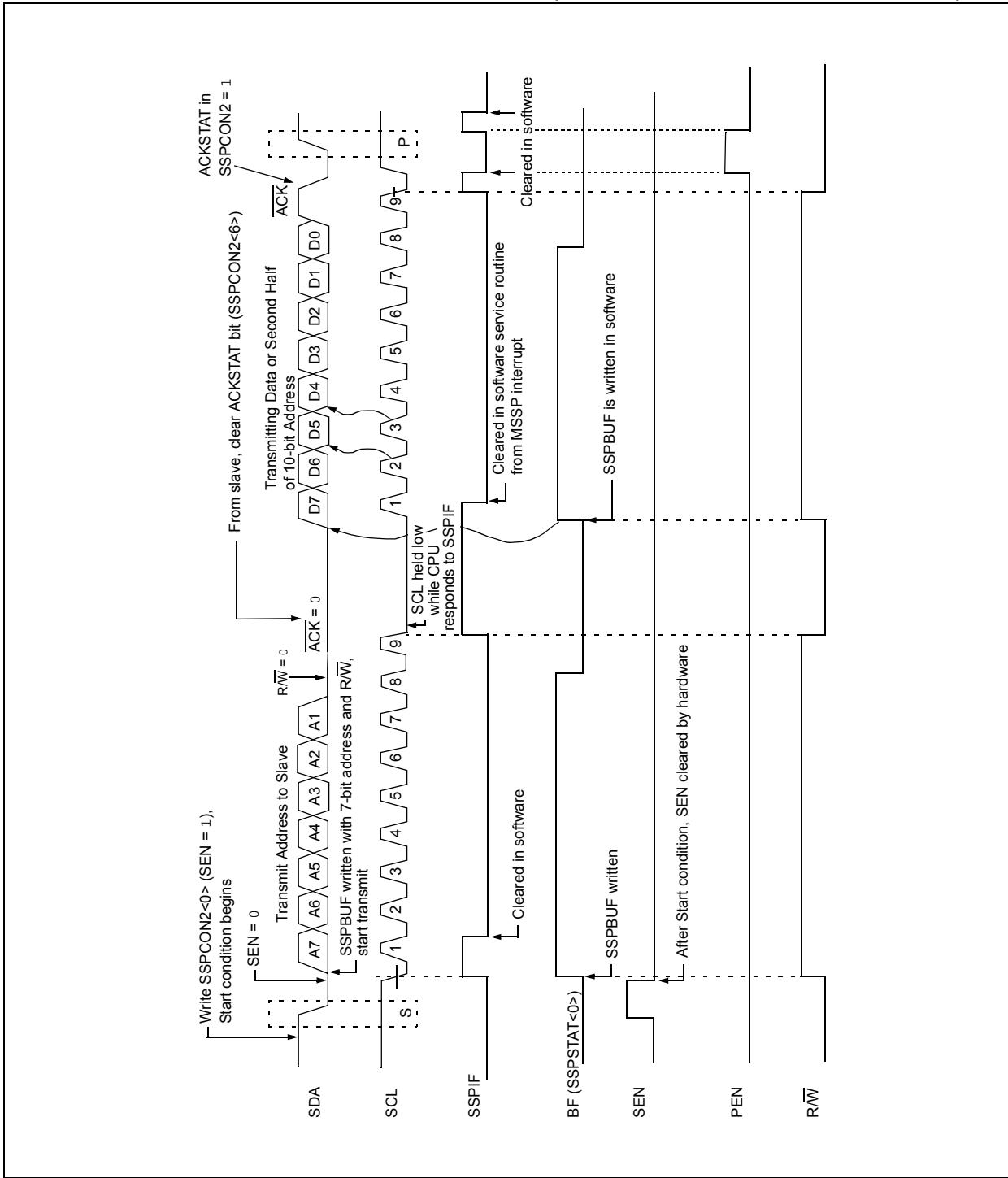
### 21.4.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

### 21.4.11.3 WCOL Status Flag

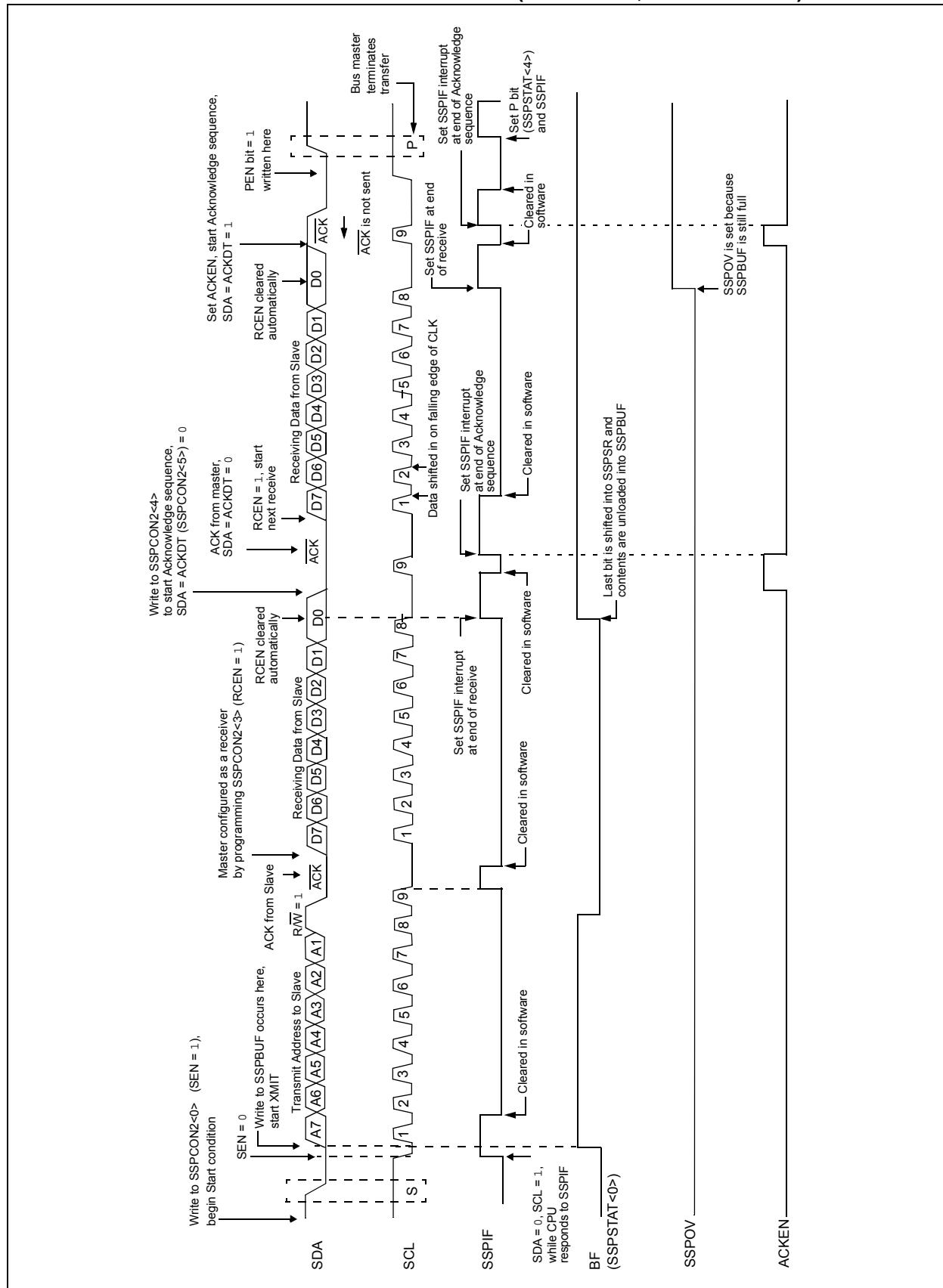
If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 21-23: I<sup>2</sup>C™ MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**



# PIC18F66K80 FAMILY

**FIGURE 21-24: I<sup>2</sup>C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)**



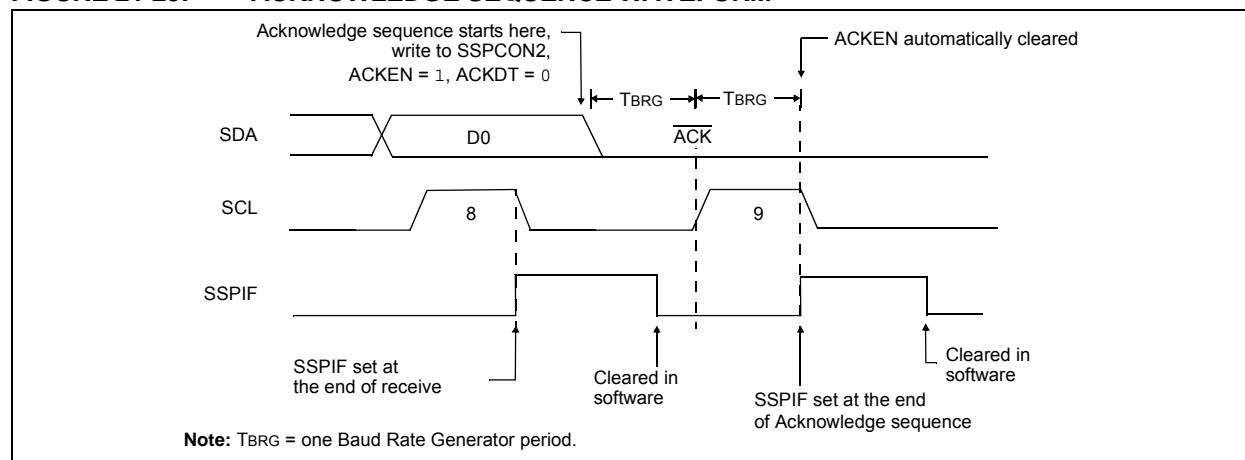
## 21.4.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>). When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG; the SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into an inactive state (Figure 21-25).

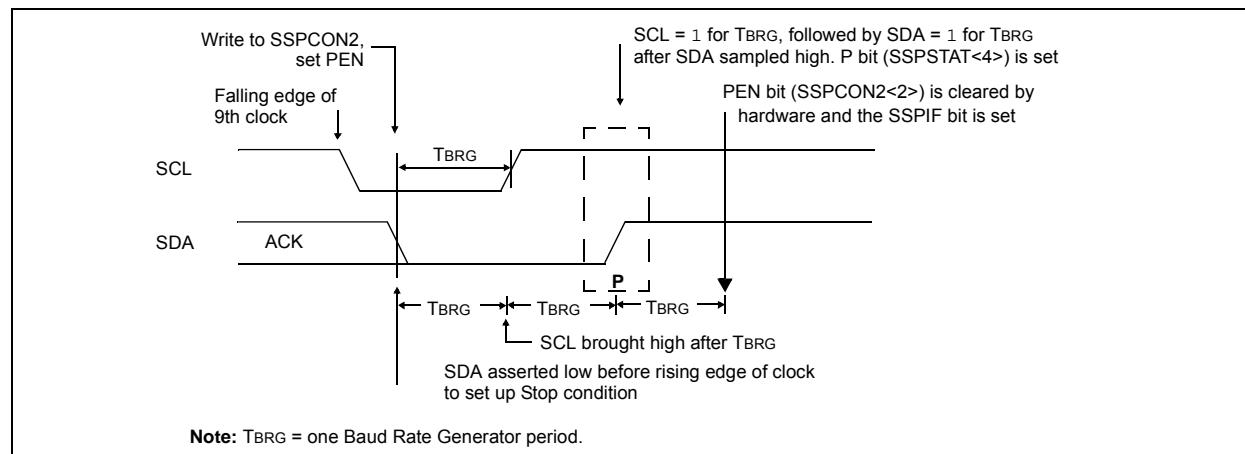
### 21.4.12.1 WCOL Status Flag

If the user writes the SSPBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 21-25: ACKNOWLEDGE SEQUENCE WAVEFORM**



**FIGURE 21-26: STOP CONDITION RECEIVE OR TRANSMIT MODE**



# PIC18F66K80 FAMILY

## 21.4.14 SLEEP OPERATION

While in Sleep mode, the I<sup>2</sup>C module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

## 21.4.15 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

## 21.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I<sup>2</sup>C bus may be taken when the P bit (SSPSTAT<4>) is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the MSSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed in hardware with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

## 21.4.17 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high, and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = 0, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF, and reset the I<sup>2</sup>C port to its Idle state ([Figure 21-27](#)).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I<sup>2</sup>C bus is free, the user can resume communication by asserting a Start condition.

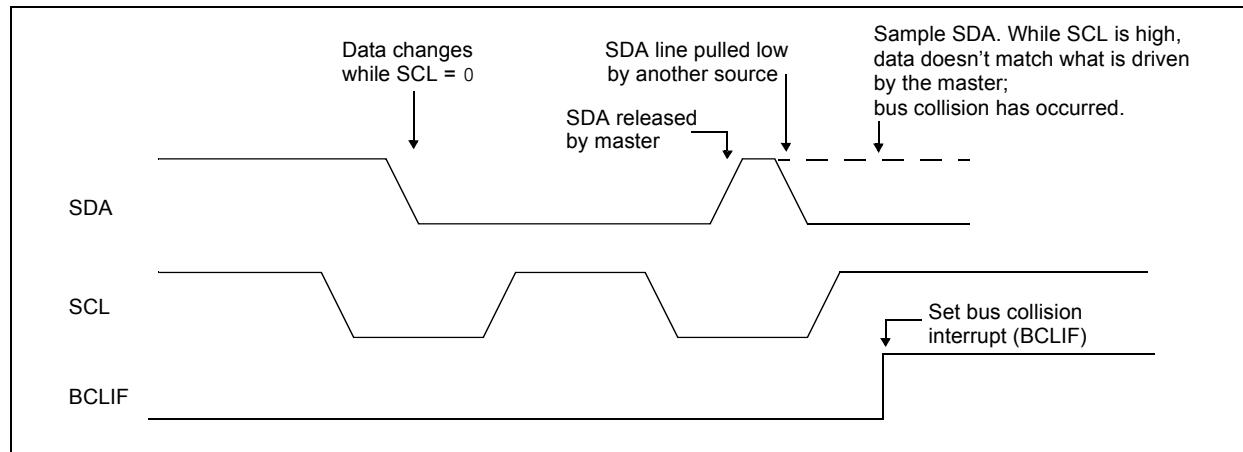
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the I<sup>2</sup>C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I<sup>2</sup>C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is Idle and the S and P bits are cleared.

**FIGURE 21-27: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



## 21.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

- SDA or SCL is sampled low at the beginning of the Start condition ([Figure 21-28](#)).
- SCL is sampled low before SDA is asserted low ([Figure 21-29](#)).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

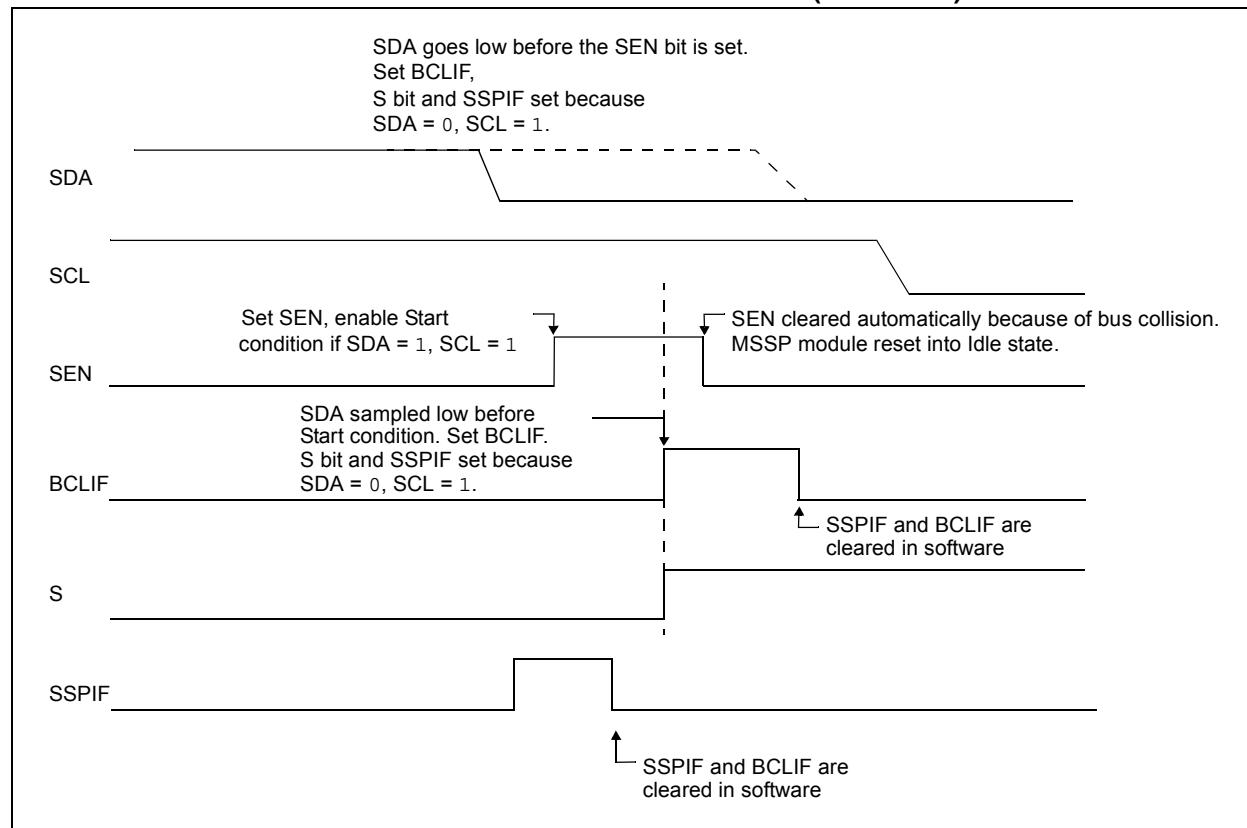
- the Start condition is aborted,
- the BCLIF flag is set and
- the MSSP module is reset to its inactive state ([Figure 21-28](#))

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded from SSPADD<6:0> and counts down to 0. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early ([Figure 21-30](#)). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to 0. If the SCL pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

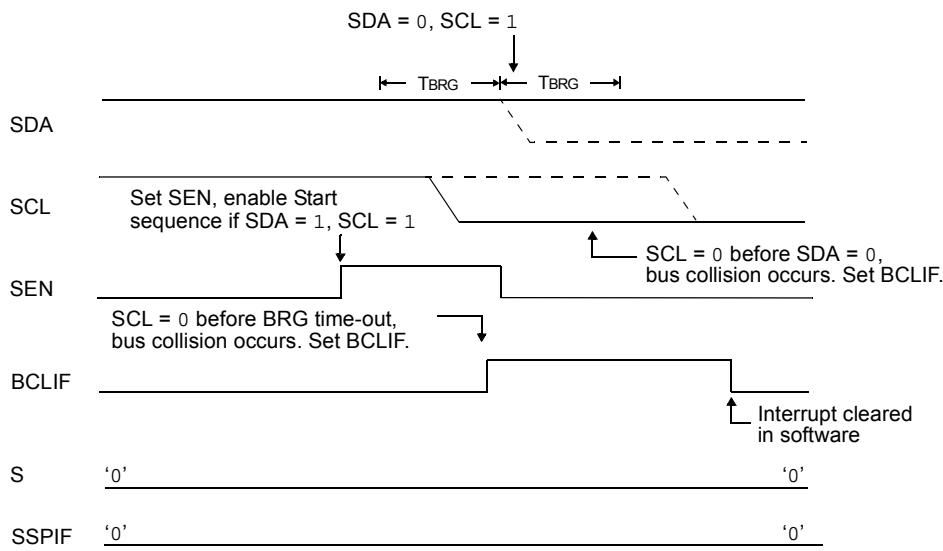
**Note:** The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

**FIGURE 21-28: BUS COLLISION DURING START CONDITION (SDA ONLY)**

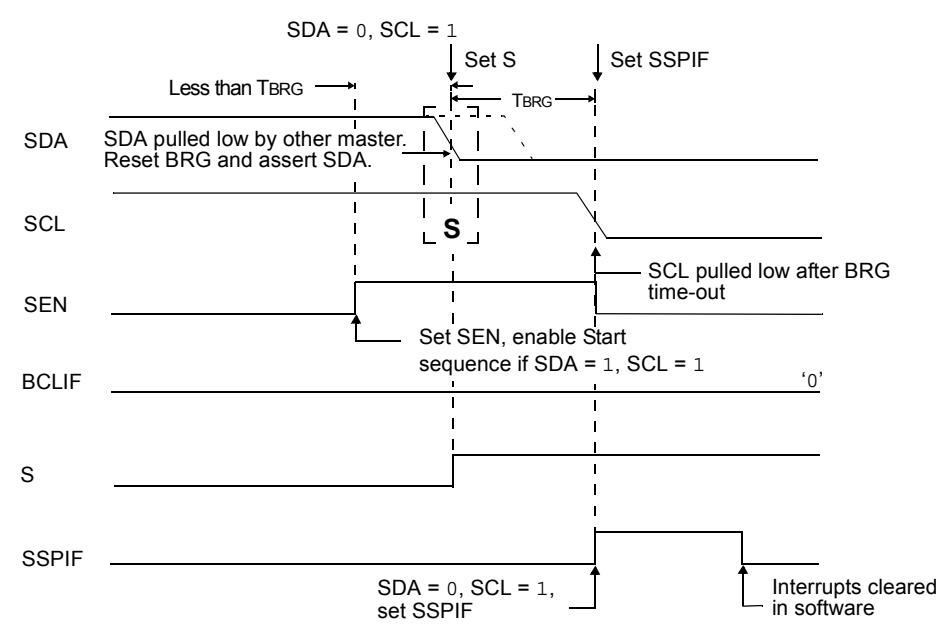


# PIC18F66K80 FAMILY

**FIGURE 21-29: BUS COLLISION DURING START CONDITION (SCL = 0)**



**FIGURE 21-30: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**



### 21.4.17.2 Bus Collision During a Repeated Start Condition

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

- A low level is sampled on SDA when SCL goes from a low level to a high level.
- SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

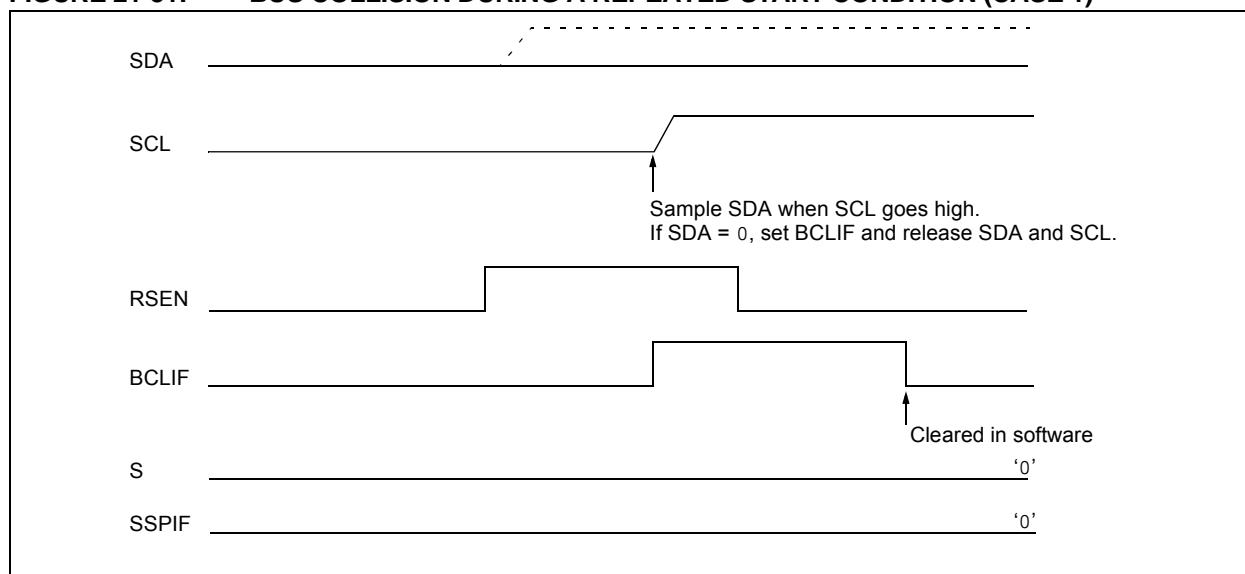
When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', [Figure 21-31](#)). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

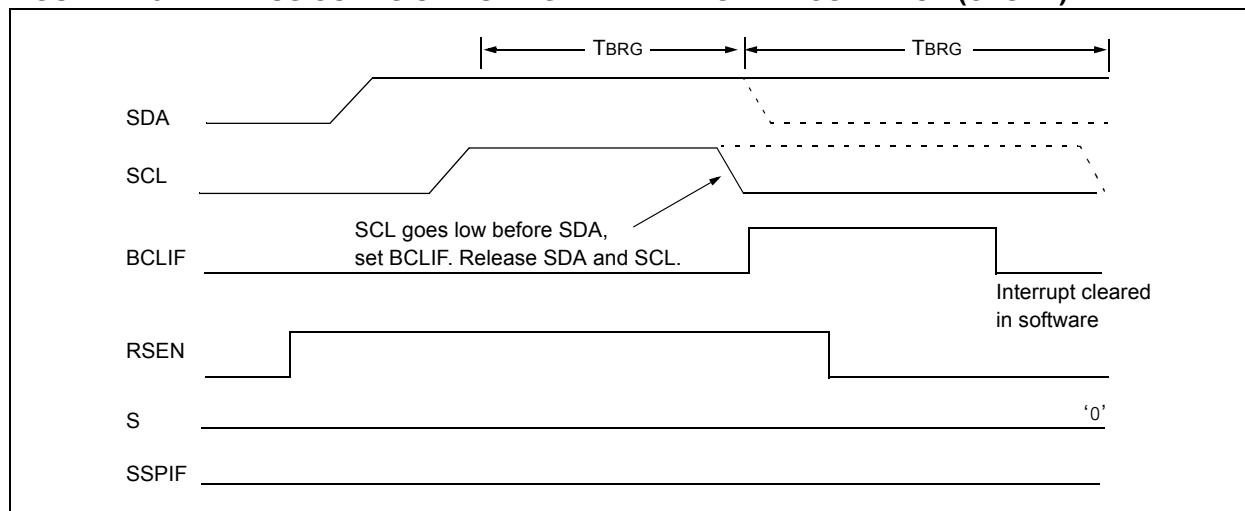
If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (see [Figure 21-32](#)).

If, at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

**FIGURE 21-31: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**



**FIGURE 21-32: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**



# PIC18F66K80 FAMILY

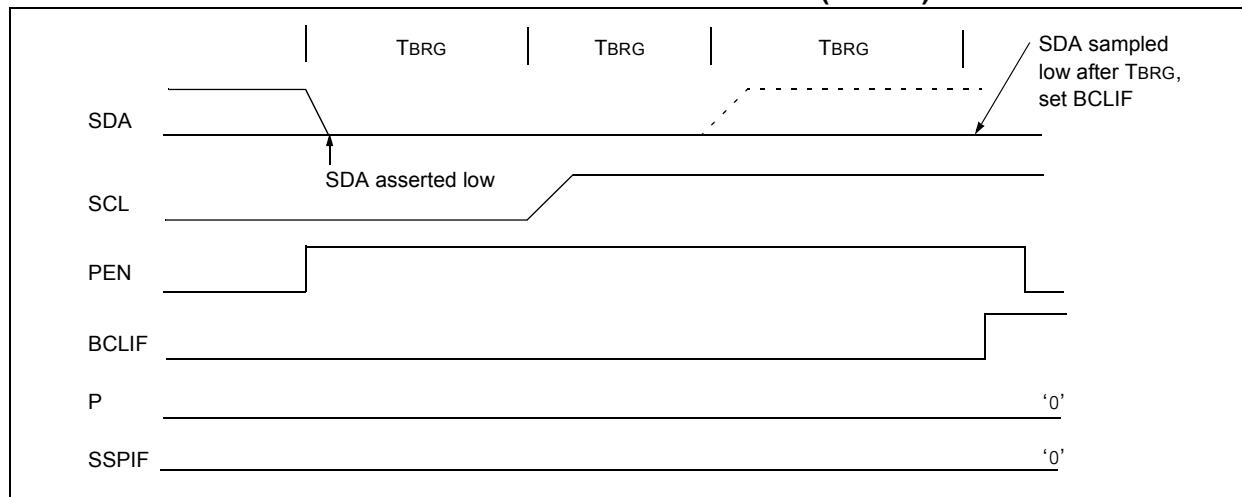
## 21.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

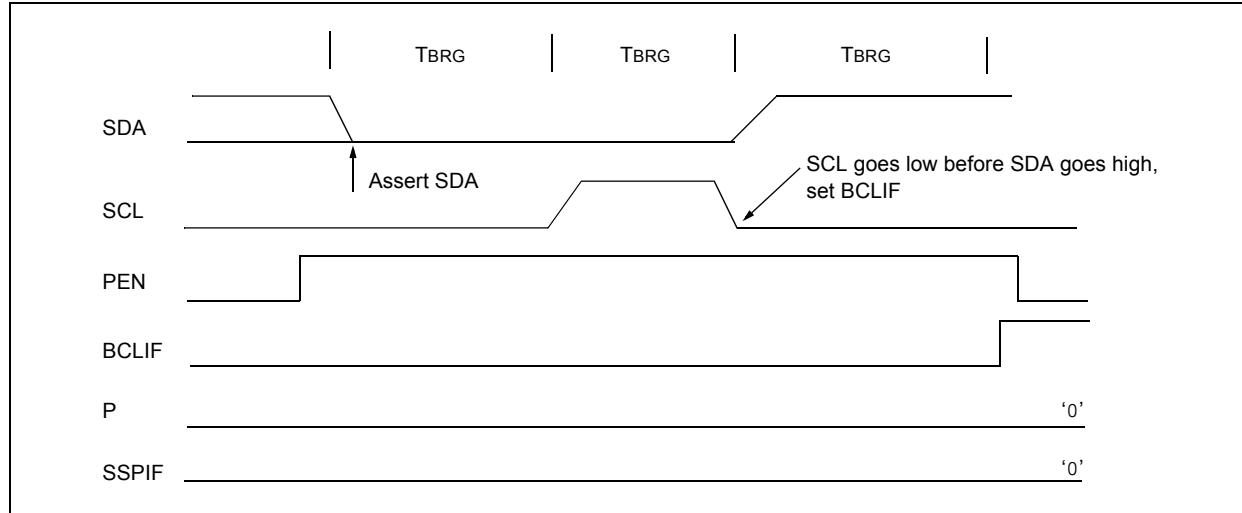
- After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.
- After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD<6:0> and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 21-33). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 21-34).

**FIGURE 21-33: BUS COLLISION DURING A STOP CONDITION (CASE 1)**



**FIGURE 21-34: BUS COLLISION DURING A STOP CONDITION (CASE 2)**



# PIC18F66K80 FAMILY

**TABLE 21-4: REGISTERS ASSOCIATED WITH I<sup>2</sup>C™ OPERATION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR2	OSCFIF	—	—	—	BCLIF	HLVDIF	TMR3IF	TMR3GIF
PIE2	OSCFIE	—	—	—	BCLIE	HLVDIE	TMR3IE	TMR3GIE
IPR2	OSCFIP	—	—	—	BCLIP	HLVDIP	TMR3IP	TMR3GIP
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISCO
SSPBUF	MSSP Receive Buffer/Transmit Register							
SSPADD	MSSP Address Register (I <sup>2</sup> C™ Slave mode), MSSP Baud Rate Reload Register (I <sup>2</sup> C Master mode)							
SSPMSK <sup>(1)</sup>	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
	GCEN	ACKSTAT	ADMSK5 <sup>(2)</sup>	ADMSK4 <sup>(2)</sup>	ADMSK3 <sup>(2)</sup>	ADMSK2 <sup>(2)</sup>	ADMSK1 <sup>(2)</sup>	SEN
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the MSSP module in I<sup>2</sup>C™ mode.

**Note 1:** SSPMSK shares the same address in SFR space as SSPADD, but is only accessible in certain I<sup>2</sup>C™ Slave operating modes in 7-Bit Masking mode. See [Section 21.4.3.4 “7-Bit Address Masking Mode”](#) for more details.

**2:** Alternate bit definitions for use in I<sup>2</sup>C Slave mode operations only.

# PIC18F66K80 FAMILY

---

---

**NOTES:**

## 22.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of two serial I/O modules. (Generically, the EUSART 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 USARTx modules implement additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN/J2602 bus) systems.

**TABLE 22-1: CONFIGURING USARTx PINS<sup>(1)</sup>**

Pin Count	USART1			USART2		
	Port	Pins	Port	Pins		
28-pin	PORTC	RC6/TX1/CK1 and RC7/RX1/DT1	PORTB	RB6/PGC/TX2/CK2/KBI2 and RB7/PGD/T3G/RX2/DT2/KBI3		
40/44-pin	PORTC	RC6/TX1/CK1 and RC7/RX1/DT1	PORTD	RD6/TX2/CK2/P1C/PSP6 and RD7/RX2/DT2/P1D/PSP7		
64-pin	PORTG	RG3/TX1/CK1 and RG0/RX1/DT1	PORTE	RE7/TX2/CK2 and RE6/RX2/DT2		

**Note 1:** The USARTx control will automatically reconfigure the pin from input to output as needed.

In order to configure the pins as an USARTx:

- For USART1:
  - SPEN (RCSTA1<7>) must be set (= 1)
  - TRISx<x> must be set (= 1)
  - For Asynchronous and Synchronous Master modes, TRISx<x> must be cleared (= 0)
  - For Synchronous Slave mode, TRISx<x> must be set (= 1)
- For USART2:
  - SPEN (RCSTA2<7>) must be set (= 1)
  - TRISx<x> must be set (= 1)
  - For Asynchronous and Synchronous Master modes, TRISx<x> must be cleared (= 0)
  - For Synchronous Slave mode, TRISx<x> must be set (= 1)

# PIC18F66K80 FAMILY

## 22.1 EUSARTx Control Registers

The operation of each Enhanced USARTx module is controlled through three registers:

- Transmit Status and Control (TXSTAx)
- Receive Status and Control (RCSTAx)
- Baud Rate Control (BAUDCONx)

These are detailed on the following pages in [Register 22-1](#), [Register 22-2](#) and [Register 22-3](#), respectively.

**Note:** Throughout this section, references to register and bit names that may be associated with a specific USART module are referred to generically by the use of 'x' in place of the specific module number. Thus, "RCSTAx" might refer to the Receive Status register for either EUSART1 or EUSART2.

### REGISTER 22-1: TXSTAx: 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	SENDDB	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 is enabled

0 = Transmit is disabled

bit 4      **SYNC:** EUSARTx Mode Select bit

1 = Synchronous mode

0 = Asynchronous mode

bit 3      **SENDDB:** Send Break Character bit

#### Asynchronous mode:

1 = Sends Sync Break on next transmission (cleared by hardware upon completion)

0 = Sync Break transmission is 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 is empty

0 = TSR is 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.

# PIC18F66K80 FAMILY

## REGISTER 22-2: RCSTAx: 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	<b>SPEN:</b> Serial Port Enable bit 1 = Serial port is enabled (configures RXx/DTx and TXx/CKx pins as serial port pins) 0 = Serial port is disabled (held in Reset)
bit 6	<b>RX9:</b> 9-Bit Receive Enable bit 1 = Selects 9-bit reception 0 = Selects 8-bit reception
bit 5	<b>SREN:</b> Single Receive Enable bit <u>Asynchronous mode:</u> Don't care. <u>Synchronous mode – Master:</u> 1 = Enables single receive 0 = Disables single receive This bit is cleared after reception is complete. <u>Synchronous mode – Slave:</u> Don't care.
bit 4	<b>CREN:</b> Continuous Receive Enable bit <u>Asynchronous mode:</u> 1 = Enables receiver 0 = Disables receiver <u>Synchronous mode:</u> 1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN) 0 = Disables continuous receive
bit 3	<b>ADDEN:</b> Address Detect Enable bit <u>Asynchronous mode 9-Bit (RX9 = 1):</u> 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 the ninth bit can be used as a parity bit <u>Asynchronous mode 9-Bit (RX9 = 0):</u> Don't care.
bit 2	<b>FERR:</b> Framing Error bit 1 = Framing error (can be cleared by reading the RCREGx register and receiving next valid byte) 0 = No framing error
bit 1	<b>OERR:</b> Overrun Error bit 1 = Overrun error (can be cleared by clearing bit, CREN) 0 = No overrun error
bit 0	<b>RX9D:</b> 9th bit of Received Data This can be address/data bit or a parity bit and must be calculated by user firmware.

# PIC18F66K80 FAMILY

## REGISTER 22-3: BAUDCONx: BAUD RATE CONTROL REGISTER

R/W-0	R-1	R/W-x	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  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

bit 7	<b>ABDOVF:</b> 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	<b>RCIDL:</b> Receive Operation Idle Status bit 1 = Receive operation is Idle 0 = Receive operation is active
bit 5	<b>RXDTP:</b> Received Data Polarity Select bit (Asynchronous mode only) <u>Asynchronous mode:</u> 1 = Receive data (RXx) is inverted 0 = Receive data (RXx) is not inverted
bit 4	<b>TXCKP:</b> Clock and Data Polarity Select bit <u>Asynchronous mode:</u> 1 = Idle state for transmit (TXx) is a low level 0 = Idle state for transmit (TXx) is a high level <u>Synchronous mode:</u> 1 = Idle state for clock (CKx) is a high level 0 = Idle state for clock (CKx) is a low level
bit 3	<b>BRG16:</b> 16-Bit Baud Rate Register Enable bit 1 = 16-bit Baud Rate Generator – SPBRGHx and SPBRGx 0 = 8-bit Baud Rate Generator – SPBRGx only (Compatible mode), SPBRGHx value is ignored
bit 2	<b>Unimplemented:</b> Read as '0'
bit 1	<b>WUE:</b> Wake-up Enable bit <u>Asynchronous mode:</u> 1 = EUSARTx will continue to sample the RXx pin: interrupt generated on falling edge; bit cleared in hardware on following rising edge 0 = RXx pin is not monitored or rising edge is detected <u>Synchronous mode:</u> Unused in this mode.
bit 0	<b>ABDEN:</b> Auto-Baud Detect Enable bit <u>Asynchronous mode:</u> 1 = Enables baud rate measurement on the next character: requires reception of a Sync field (55h); cleared in hardware upon completion 0 = Baud rate measurement is disabled or completed <u>Synchronous mode:</u> Unused in this mode.

## 22.2 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 EUSARTTx. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCONx<3>) selects 16-bit mode.

The SPBRGHx:SPBRGx register pair controls the period of a free-running timer. In Asynchronous mode, bits, BRGH (TXSTAx<2>) and BRG16 (BAUDCONx<3>), also control the baud rate. In Synchronous mode, BRGH is ignored. [Table 22-2](#) shows the formula for computation of the baud rate for different EUSARTTx modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRGHx:SPBRGx registers can be calculated using the formulas in [Table 22-2](#). From this, the error in baud rate can be determined. An example calculation is shown in [Example 22-1](#). Typical baud rates and error values for the various Asynchronous modes are shown in [Table 22-3](#). 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 SPBRGHx:SPBRGx 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.

### 22.2.1 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 SPBRGHx:SPBRGx register pair.

### 22.2.2 SAMPLING

The data on the RXx pin (either RC7/CANRX/RX1/DT1 or RB7/PGD/T3G/RX2/DT2/KBI3) is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RXx pin.

**TABLE 22-2: BAUD RATE FORMULAS**

Configuration Bits			BRG/EUSARTTx Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

**Legend:** x = Don't care, n = value of SPBRGHx:SPBRGx register pair

# PIC18F66K80 FAMILY

## EXAMPLE 22-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, and 8-bit BRG:

$$\text{Desired Baud Rate} = \text{FOSC}/(64 ([\text{SPBRGHx:SPBRGx}] + 1))$$

Solving for SPBRGHx:SPBRGx:

$$X = ((\text{FOSC}/\text{Desired Baud Rate})/64) - 1$$

$$= ((16000000/9600)/64) - 1$$

$$= [25.042] = 25$$

$$\text{Calculated Baud Rate} = 16000000/(64 (25 + 1))$$

$$= 9615$$

$$\text{Error} = (\text{Calculated Baud Rate} - \text{Desired Baud Rate})/\text{Desired Baud Rate}$$

$$= (9615 - 9600)/9600 = 0.16\%$$

TABLE 22-3: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TXSTA1	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPM <sub>D</sub>

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

# PIC18F66K80 FAMILY

TABLE 22-4: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 64.000 MHz			Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 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
2.4	—	—	—	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64
9.6	9.615	0.16	103	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15
19.2	19.231	0.16	51	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7
57.6	58.824	2.13	16	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2
115.2	111.111	-3.55	8	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 = 8.000 MHz			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)	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	1.201	-0.16	103	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.403	-0.16	51	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	9.615	-0.16	12	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 = 64.000 MHz			Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 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
9.6	—	—	—	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64
19.2	19.417	1.13	207	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31
57.6	59.701	3.65	68	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10
115.2	121.212	5.22	34	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 = 8.000 MHz			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)	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	2.403	-0.16	207	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	-0.16	51	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.230	-0.16	25	19.231	0.16	12	—	—	—	—	—	—
57.6	55.555	3.55	8	62.500	8.51	3	—	—	—	—	—	—
115.2	—	—	—	125.000	8.51	1	—	—	—	—	—	—

# PIC18F66K80 FAMILY

TABLE 22-4: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	Fosc = 64.000 MHz			Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 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	13332	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082
1.2	1.200	0.00	3332	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520
2.4	2.400	0.00	1666	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259
9.6	9.592	-0.08	416	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64
19.2	19.417	1.13	207	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31
57.6	59.701	3.65	68	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10
115.2	121.212	5.22	34	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 = 8.000 MHz			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)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	-0.04	1665	0.300	0.04	832	0.300	-0.16	415	0.300	-0.16	207
1.2	1.201	-0.16	415	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.403	-0.16	207	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	-0.16	51	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.230	-0.16	25	19.231	0.16	12	—	—	—	—	—	—
57.6	55.555	3.55	8	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 = 64.000 MHz			Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 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	53332	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332
1.2	1.200	0.00	13332	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082
2.4	2.400	0.00	6666	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040
9.6	9.598	-0.02	1666	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259
19.2	19.208	0.04	832	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129
57.6	57.348	-0.44	278	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42
115.2	115.108	-0.08	138	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	Fosc = 8.000 MHz			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)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	-0.01	6665	0.300	0.01	3332	0.300	-0.04	1665	0.300	-0.04	832
1.2	1.200	-0.04	1665	1.200	0.04	832	1.201	-0.16	415	1.201	-0.16	207
2.4	2.400	-0.04	832	2.404	0.16	415	2.403	-0.16	207	2.402	-0.16	103
9.6	9.615	-0.16	207	9.615	0.16	103	9.615	-0.16	51	9.615	-0.16	25
19.2	19.230	-0.16	103	19.231	0.16	51	19.230	-0.16	25	19.230	-0.16	12
57.6	57.142	0.79	34	58.824	2.12	16	55.555	3.55	8	—	—	—
115.2	117.647	-2.12	16	111.111	-3.55	8	—	—	—	—	—	—

### 22.2.3 AUTO-BAUD RATE DETECT

The Enhanced USARTx modules support 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 22-1](#)) 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 RXx signal, the RXx 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 SPBRGx begins counting up, using the preselected clock source on the first rising edge of RXx. After eight bits on the RXx pin or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGHx:SPBRGx 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 (BAUDCONx<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 22-2](#)).

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. The BRG clock will 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 SPBRGHx register. Refer to [Table 22-5](#) for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSARTx state machine is held in Idle. The RCxIF interrupt is set once the fifth rising edge on RXx is detected. The value in the RCREGx needs to be read to clear the RCxIF interrupt. The contents of RCREGx 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 EUSARTx 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, if that feature is used it is recommended that the BRG16 bit (BAUDCONx<3>) be set.

**TABLE 22-5: 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

#### 22.2.3.1 ABD and EUSARTx Transmission

Since the BRG clock is reversed during ABD acquisition, the EUSARTx transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREGx 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 EUSARTx operation.

# PIC18F66K80 FAMILY

FIGURE 22-1: AUTOMATIC BAUD RATE CALCULATION

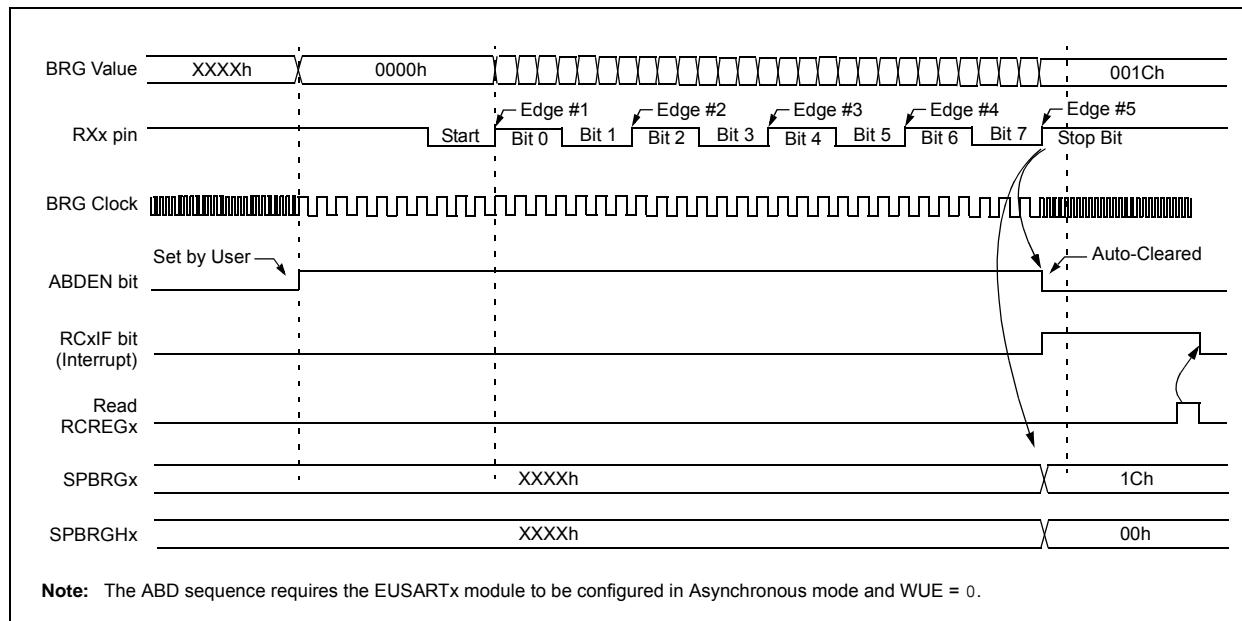
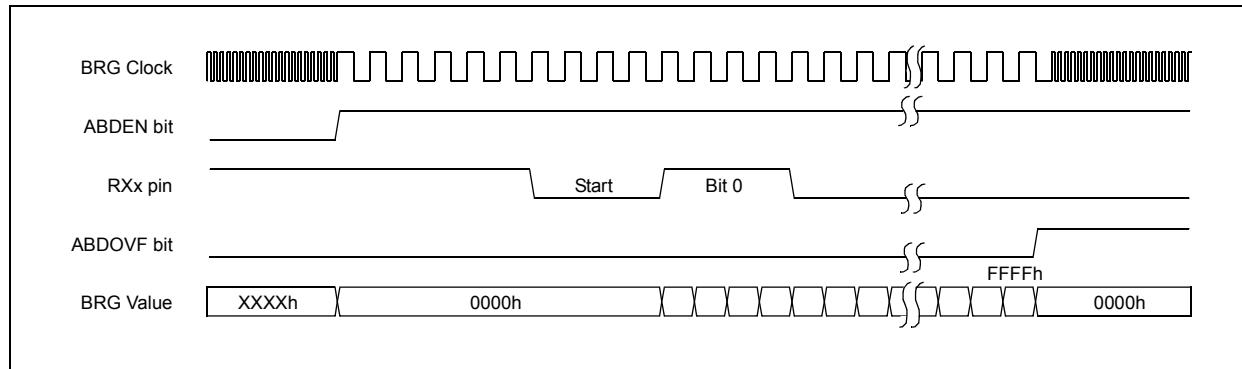


FIGURE 22-2: BRG OVERFLOW SEQUENCE



## 22.3 EUSARTx Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTAx<4>). In this mode, the EUSARTx 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 EUSARTx transmits and receives the LSb first. The EUSARTx'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 (TXSTAx<2> and BAUDCONx<3>). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.

When operating in Asynchronous mode, the EUSARTx 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

### 22.3.1 EUSARTx ASYNCHRONOUS TRANSMITTER

The EUSARTx transmitter block diagram is shown in [Figure 22-3](#). 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, TXREGx. The TXREGx 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 TXREGx register (if available).

Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx register is empty and the TXxIF flag bit is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxE. TXxIF will be set regardless of the state of TXxE; it cannot be cleared in software. TXxIF is also not cleared immediately upon loading TXREGx, but becomes valid in the second instruction cycle following the load instruction. Polling TXxIF immediately following a load of TXREGx will return invalid results.

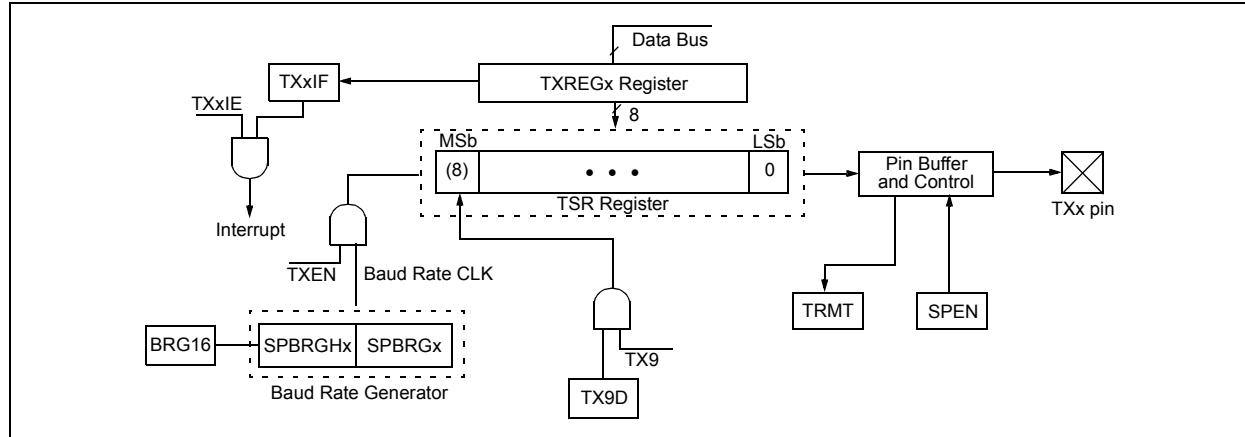
While TXxIF indicates the status of the TXREGx register; another bit, TRMT (TXSTAx<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty.

- Note 1:** The TSR register is not mapped in data memory, so it is not available to the user.  
**2:** Flag bit, TXxIF, is set when enable bit, TXEN, is set.

To set up an Asynchronous Transmission:

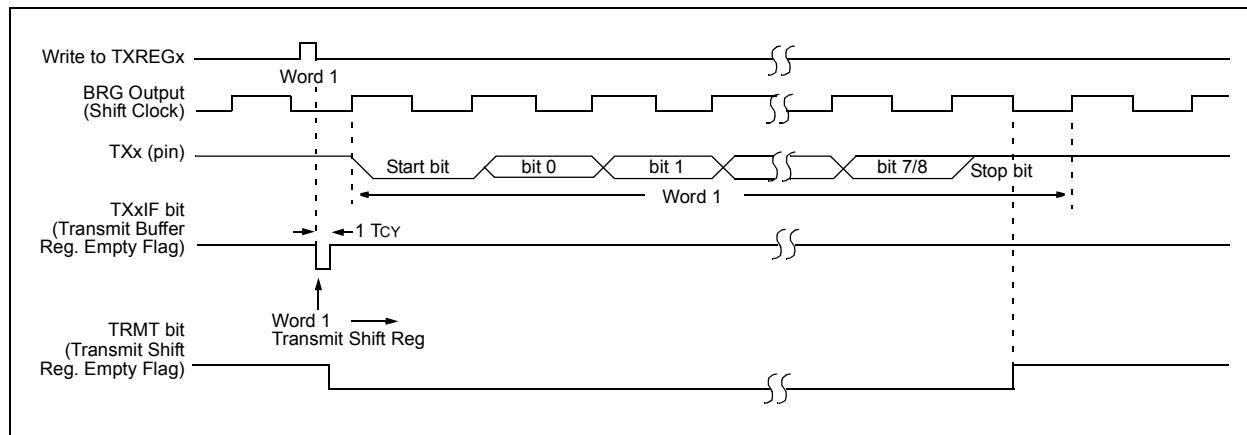
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, TXxE.
4. If 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, TXxIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Load data to the TXREGx register (starts transmission).
8. If using interrupts, ensure that the GIE and PEIE bits (INTCON<7:6>) are set.

**FIGURE 22-3: EUSARTx TRANSMIT BLOCK DIAGRAM**

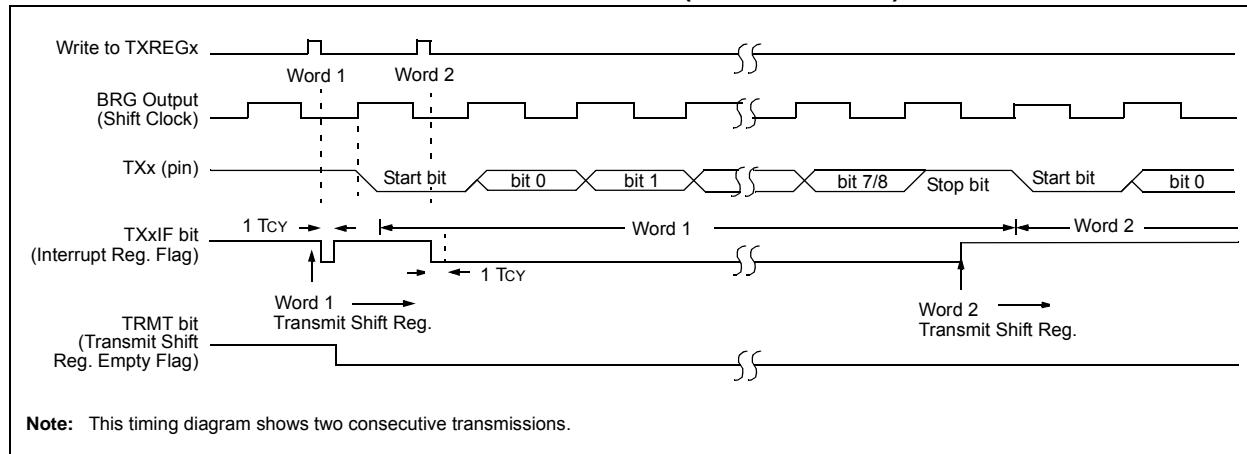


# PIC18F66K80 FAMILY

**FIGURE 22-4: ASYNCHRONOUS TRANSMISSION**



**FIGURE 22-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**



# PIC18F66K80 FAMILY

**TABLE 22-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG1	EUSART1 Transmit Register							
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte							
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG2	EUSART2 Transmit Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

# PIC18F66K80 FAMILY

## 22.3.2 EUSARTx ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in [Figure 22-6](#). The data is received on the RXx 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:

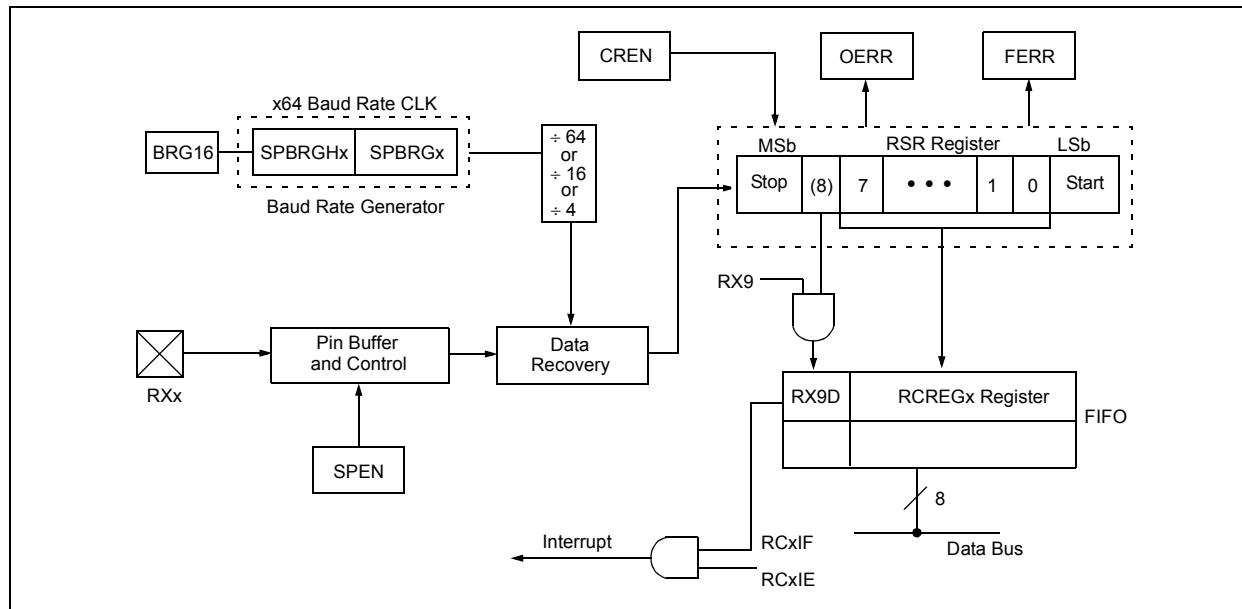
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCxIE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RCxIF, will be set when reception is complete and an interrupt will be generated if enable bit, RCxIE, was set.
7. Read the RCSTAx 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 RCREGx 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 (INTCON<7:6>) are set.

## 22.3.3 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:

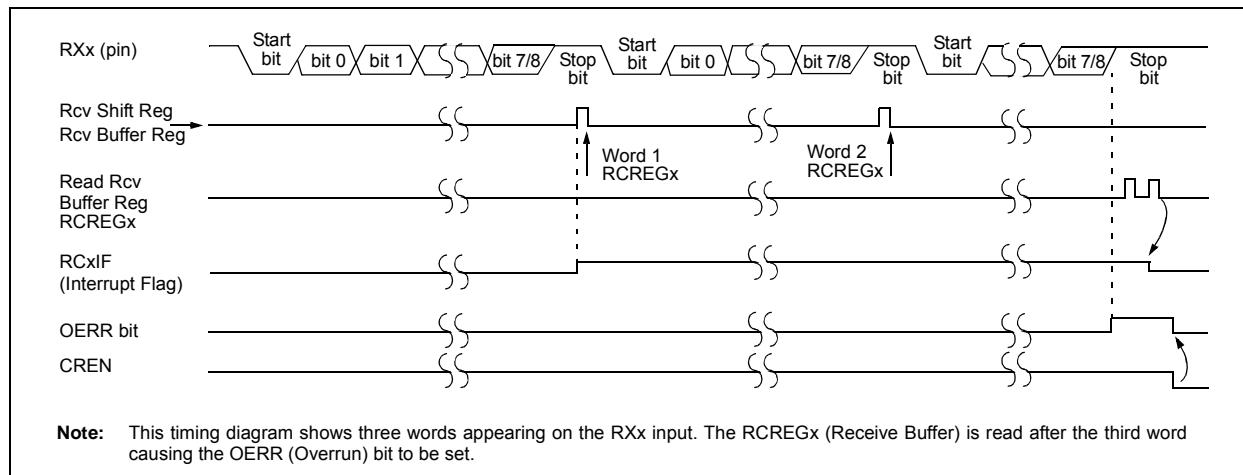
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCxIP 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 RCxIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCxIE and GIE bits are set.
8. Read the RCSTAx register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREGx 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 22-6: EUSARTx RECEIVE BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

**FIGURE 22-7: ASYNCHRONOUS RECEPTION**



**TABLE 22-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG1	EUSART1 Receive Register							
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register							
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG2	EUSART2 Receive Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

# PIC18F66K80 FAMILY

---

## 22.3.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSARTx 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 RXx/DTx line while the EUSARTx is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCONx<1>). Once set, the typical receive sequence on RXx/DTx is disabled and the EUSARTx 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 RXx/DTx 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 RCxIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes ([Figure 22-8](#)) and asynchronously if the device is in Sleep mode ([Figure 22-9](#)). The interrupt condition is cleared by reading the RCREGx register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RXx line following the wake-up event. At this point, the EUSARTx module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

## 22.3.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RXx/DTx, information with any state changes before the Stop bit may signal a false End-of-Character (EOC) and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bytes) 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., HS or HSPLL 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 EUSARTx.

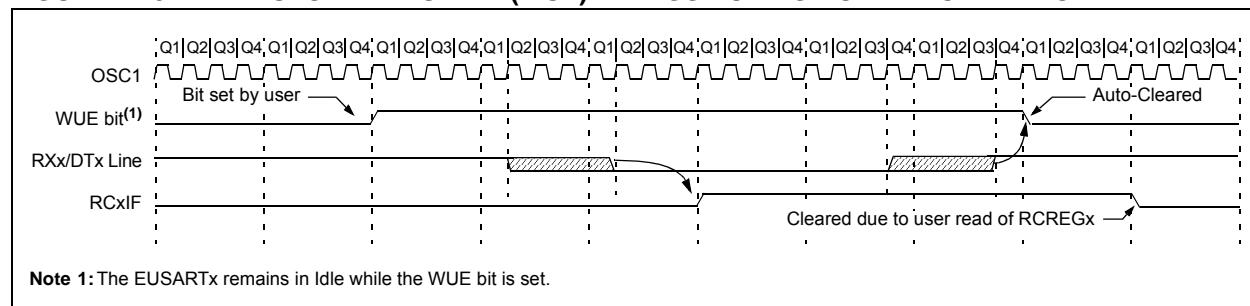
### 22.3.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCxIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSARTx in an Idle mode. The wake-up event causes a receive interrupt by setting the RCxIF bit. The WUE bit is cleared after this when a rising edge is seen on RXx/DTx. The interrupt condition is then cleared by reading the RCREGx register. Ordinarily, the data in RCREGx will be dummy data and should be discarded.

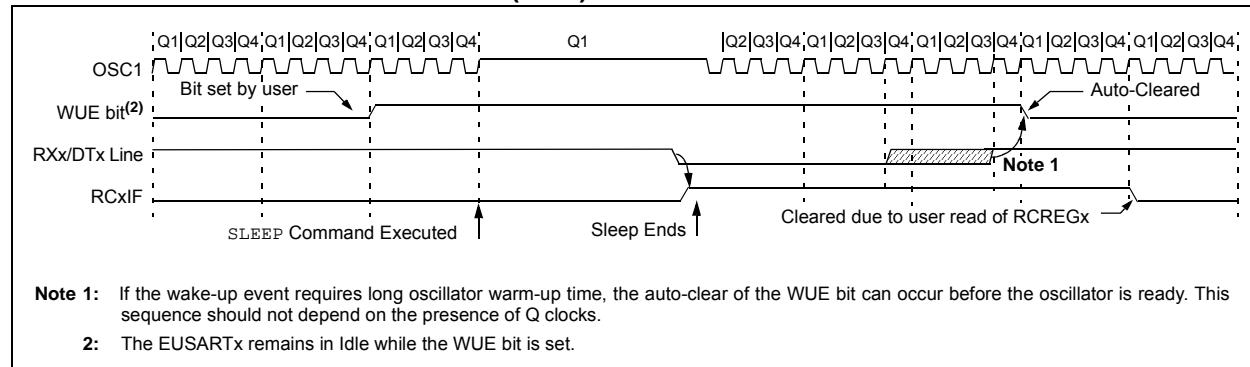
The fact that the WUE bit has been cleared (or is still set) and the RCxIF flag is set should not be used as an indicator of the integrity of the data in RCREGx. Users should consider implementing a parallel method in firmware to verify received data integrity.

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 22-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**



**FIGURE 22-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**



# PIC18F66K80 FAMILY

## 22.3.5 BREAK CHARACTER SEQUENCE

The EUSARTx 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 (TXSTAx<3> and TXSTAx<5>, respectively) are set while the Transmit Shift Register is loaded with data. Note that the value of data written to TXREGx 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 TXREGx 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 22-10](#) for the timing of the Break character sequence.

### 22.3.5.1 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 EUSARTx for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREGx with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREGx 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 TXREGx becomes empty, as indicated by the TXxIF, the next data byte can be written to TXREGx.

## 22.3.6 RECEIVING A BREAK CHARACTER

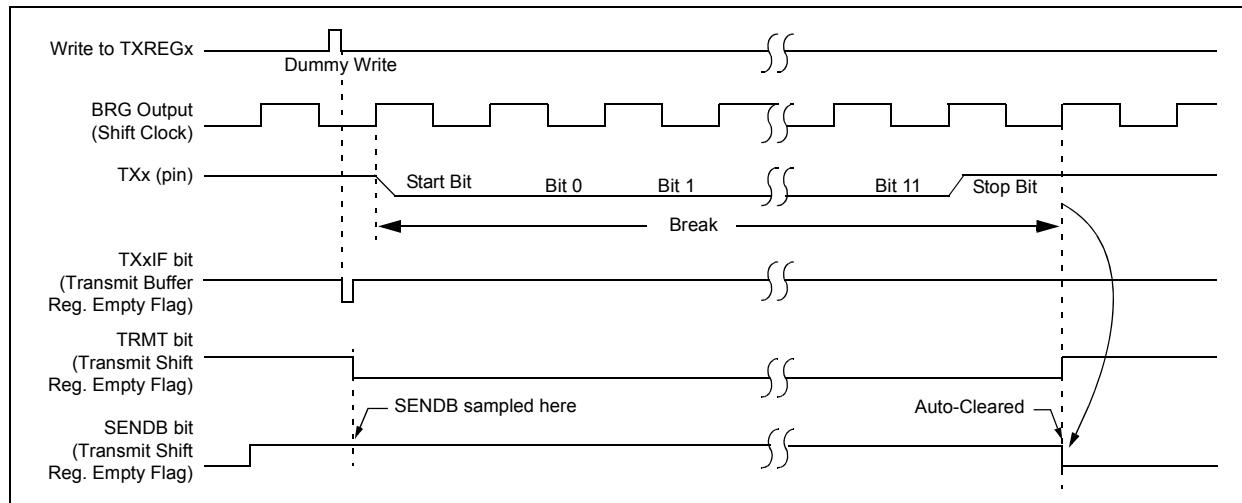
The Enhanced USART modules can receive a Break character in two ways.

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 22.3.4 "Auto-Wake-up on Sync Break Character"](#). By enabling this feature, the EUSARTx will sample the next two transitions on RXx/DTx, cause an RCxIF 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 TXxIF interrupt is observed.

**FIGURE 22-10: SEND BREAK CHARACTER SEQUENCE**



## 22.4 EUSARTx Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTAx<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 (TXSTAx<4>). In addition, enable bit, SPEN (RCSTAx<7>), is set in order to configure the TXx and RXx pins to CKx (clock) and DTx (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CKx line. Clock polarity is selected with the TXCKP bit (BAUDCONx<4>). Setting TXCKP sets the Idle state on CKx as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

### 22.4.1 EUSARTx SYNCHRONOUS MASTER TRANSMISSION

The EUSARTx transmitter block diagram is shown in Figure 22-3. 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, TXREGx. The TXREGx 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 TXREGx (if available).

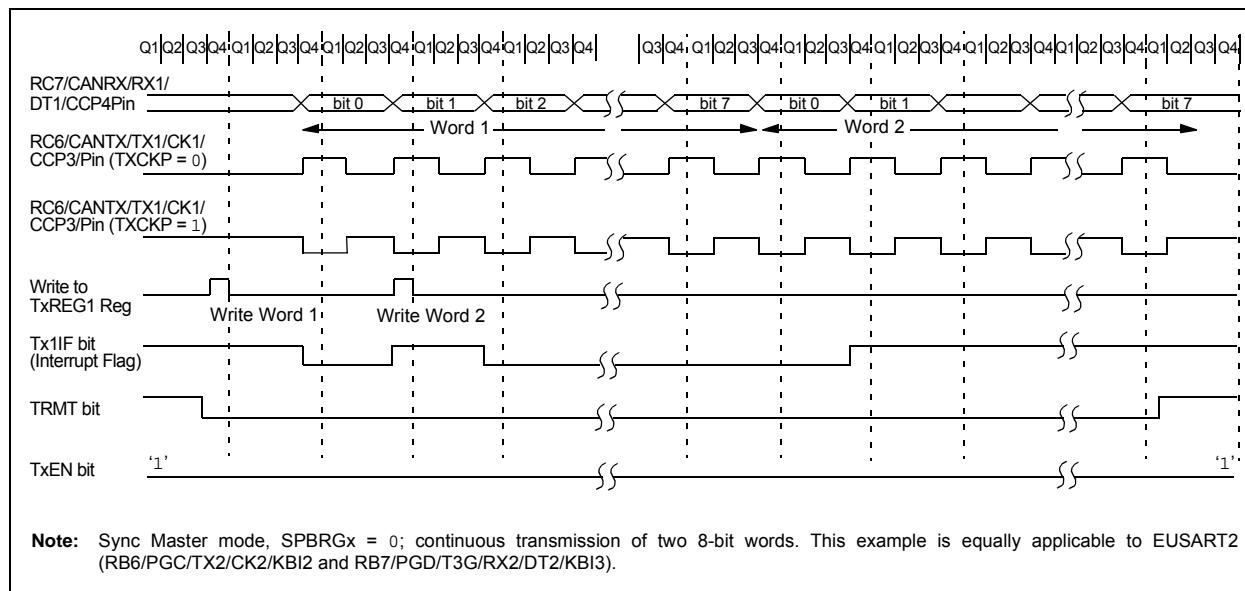
Once the TXREGx register transfers the data to the TSR register (occurs in one Tcy), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxE. TXxIF is set regardless of the state of enable bit, TXxE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREGx register.

While flag bit, TXxIF, indicates the status of the TXREGx register, another bit, TRMT (TXSTAx<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 must 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:

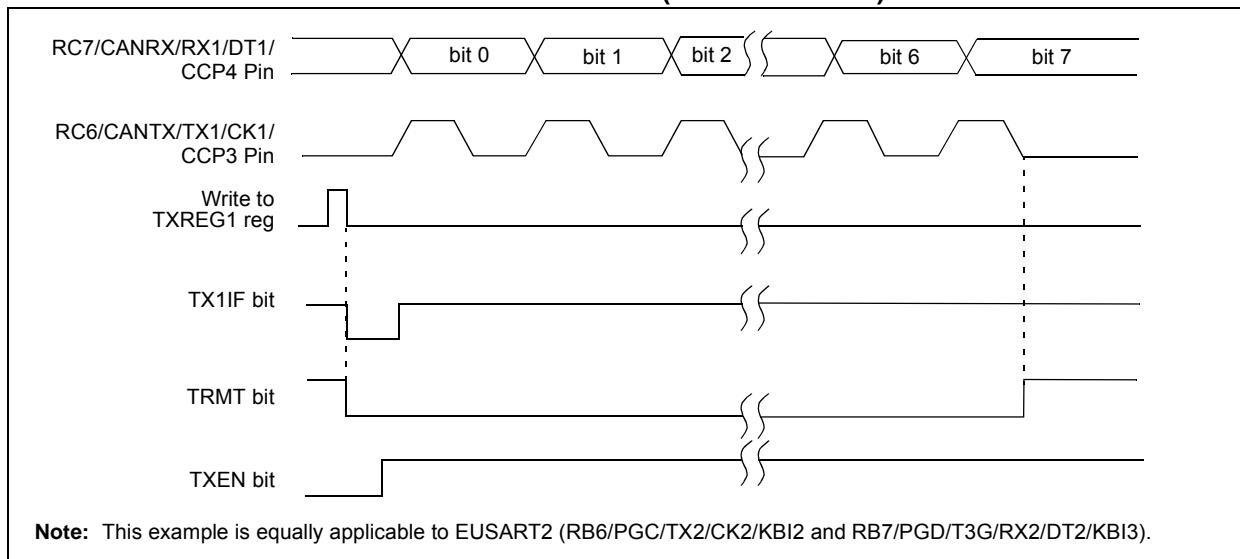
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TXxE.
4. If 9-bit transmission 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 TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits (INTCON<7:6>) are set.

**FIGURE 22-11: SYNCHRONOUS TRANSMISSION**



# PIC18F66K80 FAMILY

**FIGURE 22-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**



**TABLE 22-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG1	EUSART1 Transmit Register							
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte							
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG2	EUSART2 Transmit Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMOD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

## 22.4.2 EUSARTx SYNCHRONOUS MASTER RECEPTION

Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTAx<5>) or the Continuous Receive Enable bit, CREN (RCSTAx<4>). Data is sampled on the RXx pin on the falling edge of the clock.

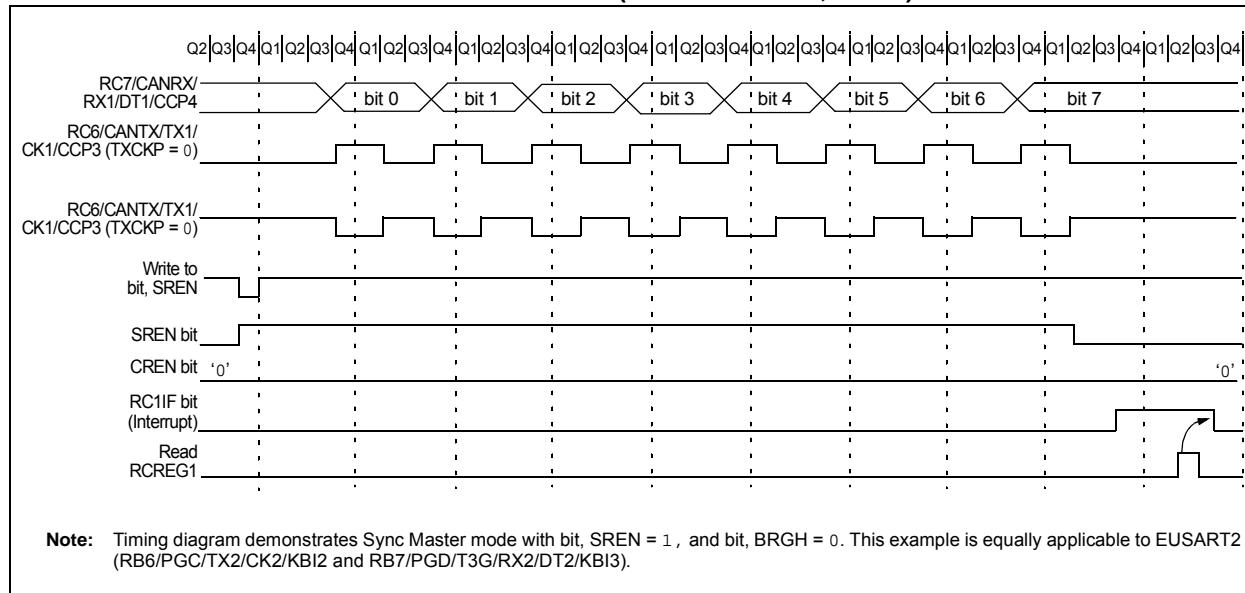
If enable bit, SREN, is set, only a single word is received. If enable bit, CREN, is set, the reception is continuous until CREN is cleared. If both bits are set, then CREN takes precedence.

To set up a Synchronous Master Reception:

1. Initialize the SPBRGHx:SPBRGx 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. Ensure bits, CREN and SREN, are clear.
4. If interrupts are desired, set enable bit, RCxIE.
5. If 9-bit reception is desired, set bit, RX9.
6. If a single reception is required, set bit, SREN. For continuous reception, set bit, CREN.
7. Interrupt flag bit, RCxIF, will be set when reception is complete and an interrupt will be generated if the enable bit, RCxIE, was set.
8. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREGx register.
10. If any error occurred, clear the error by clearing bit, CREN.
11. If using interrupts, ensure that the GIE and PEIE bits (INTCON<7:6>) are set.

**FIGURE 22-13: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**



# PIC18F66K80 FAMILY

---

TABLE 22-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG1	EUSART1 Receive Register							
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte							
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG2	EUSART2 Receive Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

## 22.5 EUSART<sub>x</sub> Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit, CSRC (TXSTA<sub>x</sub><7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the CK<sub>x</sub> pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

### 22.5.1 EUSART<sub>x</sub> SYNCHRONOUS SLAVE TRANSMISSION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep mode.

If two words are written to the TXREG<sub>x</sub> 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<sub>x</sub> register.
- c) Flag bit, TX<sub>x</sub>IF, will not be set.
- d) When the first word has been shifted out of TSR, the TXREG<sub>x</sub> register will transfer the second word to the TSR and flag bit, TX<sub>x</sub>IF, will now be set.

- e) If enable bit, TX<sub>x</sub>IE, 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, TX<sub>x</sub>IE.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting enable bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREG<sub>x</sub> register.
8. If using interrupts, ensure that the GIE and PEIE bits (INTCON<7:6>) are set.

**TABLE 22-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG1	EUSART1 Transmit Register							
TXSTA1	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte							
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG2	EUSART2 Transmit Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

# PIC18F66K80 FAMILY

## 22.5.2 EUSARTx 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 RCREGx register. If the RCxIE 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, RCxIE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RCxF, will be set when reception is complete. An interrupt will be generated if enable bit, RCxIE, was set.
6. Read the RCSTA $x$  register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG $x$  register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits (INTCON<7:6>) are set.

TABLE 22-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIFF
PIR1	PSP1IF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSP1IE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSP1IP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG1	EUSART1 Receive Register							
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte							
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG2	EUSART2 Receive Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMOD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

## 23.0 12-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module in the PIC18F66K80 family of devices. It is a 13-bit differential A/D with 12-bit single-ended compatibility. It has inputs eight inputs for the 28-pin devices, 11 inputs for the 40/44-pin and 64-pin devices. This module allows conversion of an analog input signal to a corresponding 12-bit digital number.

The module has these registers:

- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)
- A/D Port Configuration Register 1 (ANCON0)
- A/D Port Configuration Register 2 (ANCON1)
- ADRESH (the upper, A/D Results register)
- ADRESL (the lower, A/D Results register)

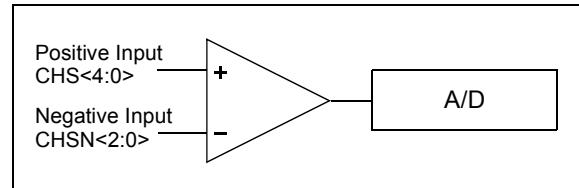
The ADCON0 register, shown in [Register 23-1](#), controls the operation of the A/D module. The ADCON1 register, shown in [Register 23-2](#), configures the voltage reference and special trigger selection. The ADCON2 register, shown in [Register 23-3](#), configures the A/D clock source and programmed acquisition time and justification.

### 23.1 Differential A/D Converter

The converter in PIC18F66K80 family devices is implemented as a differential A/D where the differential voltage between two channels is measured and converted to digital values (see [Figure 23-1](#)).

The converter also can be configured to measure a voltage from a single input by clearing the CHSNx bits (ADCON1<2:0>). With this configuration, the negative channel input is connected internally to AVss (see [Figure 23-2](#)).

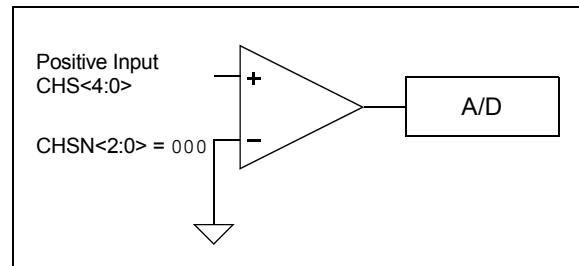
**FIGURE 23-1: DIFFERENTIAL CHANNEL MEASUREMENT**



Differential conversion feeds the two input channels to a unity gain differential amplifier. The positive channel input is selected using the CHSx bits (ADCON0<6:2>) and the negative channel input is selected using the CHSNx bits (ADCON1<2:0>).

The output from the amplifier is fed to the A/D Converter, as shown in [Figure 23-1](#). The 12-bit result is available on the ADRESH and ADRESL registers. An additional bit indicates if the 12-bit result is a positive or negative value.

**FIGURE 23-2: SINGLE CHANNEL MEASUREMENT**



In the Single Channel Measurement mode, the negative input is connected to Avss by clearing the CHSNx bits (ADCON1<2:0>).

# PIC18F66K80 FAMILY

## 23.2 A/D Registers

### 23.2.1 A/D CONTROL REGISTERS

#### REGISTER 23-1: ADCON0: A/D CONTROL REGISTER 0

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CHS4	CHS3	CHS2	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      **Unimplemented:** Read as '0'

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

00000 = Channel 00 (AN0)	10000 = (Reserved) <sup>(2)</sup>
00001 = Channel 01 (AN1)	10001 = (Reserved) <sup>(2)</sup>
00010 = Channel 02 (AN2)	10010 = (Reserved) <sup>(2)</sup>
00011 = Channel 03 (AN3)	10011 = (Reserved) <sup>(2)</sup>
00100 = Channel 04 (AN4)	10100 = (Reserved) <sup>(2)</sup>
00101 = Channel 05 (AN5) <sup>(1,2)</sup>	10101 = (Reserved) <sup>(2)</sup>
00110 = Channel 06 (AN6) <sup>(1,2)</sup>	10110 = (Reserved) <sup>(2)</sup>
00111 = Channel 07 (AN7) <sup>(1,2)</sup>	10111 = (Reserved) <sup>(2)</sup>
01000 = Channel 08 (AN8)	11000 = (Reserved) <sup>(2)</sup>
01001 = Channel 09 (AN9)	11001 = (Reserved) <sup>(2)</sup>
01010 = Channel 10 (AN10)	11010 = (Reserved) <sup>(2)</sup>
01011 = (Reserved) <sup>(2)</sup>	11011 = (Reserved) <sup>(2)</sup>
01100 = (Reserved) <sup>(2)</sup>	11100 = (MUX disconnect) <sup>(3)</sup>
01101 = (Reserved) <sup>(2)</sup>	11101 = Channel 29 (temperature diode)
01110 = (Reserved) <sup>(2)</sup>	11110 = Channel 30 (VDDCORE)
01111 = (Reserved) <sup>(2)</sup>	11111 = Channel 31 (1.024V band gap)

bit 1      **GO/DONE:** A/D Conversion Status bit

1 = A/D cycle is in progress. Setting this bit starts an A/D conversion cycle. The bit is cleared automatically by hardware when the A/D conversion is completed.

0 = A/D conversion has completed or is not in progress

bit 0      **ADON:** A/D On bit

1 = A/D Converter is operating

0 = A/D conversion module is shut off and consuming no operating current

**Note 1:** These channels are not implemented on 28-pin devices.

**2:** Performing a conversion on unimplemented channels will return random values.

**3:** Channel 28 turns off analog MUX switches to allow for minimum capacitive loading of the A/D input, for finer resolution CTMU time measurements.

# PIC18F66K80 FAMILY

## REGISTER 23-2: ADCON1: A/D CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-x	R/W-x	R/W-x	R/W-x
TRIGSEL1	TRIGSEL0	VCFG1	VCFG0	VNCFG	CHSN2	CHSN1	CHSN0
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      **TRIGSEL<1:0>**: Special Trigger Select bits

- 11 = Selects the special trigger from the CCP2
- 10 = Selects the special trigger from the Timer1
- 01 = Selects the special trigger from the CTMU
- 00 = Selects the special trigger from the ECCP1

bit 5-4      **VCFG<1:0>**: A/D VREF+ Configuration bits

- 11 = Internal VREF+ (4.1V)
- 10 = Internal VREF+ (2.0V)
- 01 = External VREF+
- 00 = AVDD

bit 3      **VNCFG**: A/D VREF- Configuration bit

- 1 = External VREF
- 0 = AVss

bit 2-0      **CHSN<2:0>**: Analog Negative Channel Select bits

- 111 = Channel 07 (AN6)
- 110 = Channel 06 (AN5)
- 101 = Channel 05 (AN4)
- 100 = Channel 04 (AN3)
- 011 = Channel 03 (AN2)
- 010 = Channel 02 (AN1)
- 001 = Channel 01 (AN0)
- 000 = Channel 00 (AVss)

# PIC18F66K80 FAMILY

## REGISTER 23-3: 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      **ACQT<2:0>:** 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      **ADCS<2:0>:** 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 TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

## 23.2.2 A/D RESULT REGISTERS

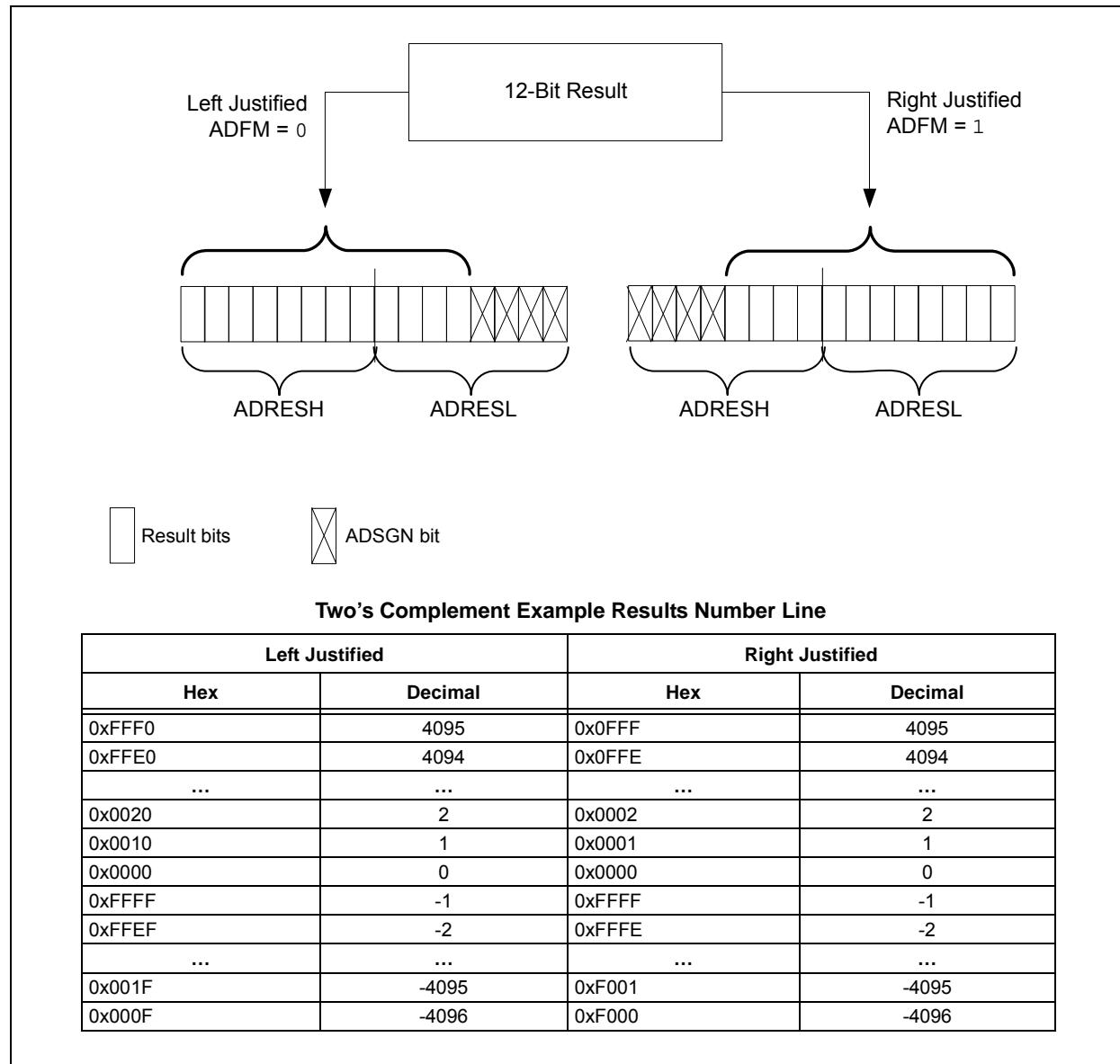
The ADRESH:ADRESL register pair is where the 12-bit A/D result and extended sign bits (ADSGNx) are loaded at the completion of a conversion. This register pair is 16 bits wide. The A/D module gives the flexibility of left or right justifying the 12-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification.

Figure 23-3 shows the operation of the A/D result justification and the location of the sign bit (ADSGNx). The extended sign bits allow for easier 16-bit math to

be performed on the result. The results are represented as a two's compliment binary value. This means that when sign bits and magnitude bits are considered together in right justification, the ADRESH and ADRESL registers can be read as a single signed integer value.

When the A/D Converter is disabled, these 8-bit registers can be used as two general purpose registers.

**FIGURE 23-3: A/D RESULT JUSTIFICATION**



# PIC18F66K80 FAMILY

## REGISTER 23-4: ADRESH: A/D RESULT HIGH BYTE REGISTER, LEFT JUSTIFIED (ADFM = 0)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
bit 7							

**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 7-0      **ADRES<11:4>**: A/D Result High Byte bits

## REGISTER 23-5: ADRESL: A/D RESULT LOW BYTE REGISTER, LEFT JUSTIFIED (ADFM = 0)

R/W-x	R/W-x	R/W-x	R/W-x	U-x	U-x	U-x	U-x
ADRES3	ADRES2	ADRES1	ADRES0	ADSGN3	ADSGN2	ADSGN1	ADSGN0
bit 7							

**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      **ADRES<3:0>**: A/D Result Low Byte bitsbit 3-0      **ADSGN<3:0>**: A/D Result Sign bits

1 = A/D result is negative

0 = A/D result is positive

## REGISTER 23-6: ADRESH: A/D RESULT HIGH BYTE REGISTER, RIGHT JUSTIFIED (ADFM = 1)

U-x	U-x	U-x	U-x	R/W-x	R/W-x	R/W-x	R/W-x
ADSGN7	ADSGN6	ADSGN5	ADSGN4	ADRES11	ADRES10	ADRES9	ADRES8
bit 7							

**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      **ADSGN<7:4>**: A/D Result Sign bits

1 = A/D result is negative

0 = A/D result is positive

bit 3-0      **ADRES<11:8>**: A/D Result High Byte bits

# PIC18F66K80 FAMILY

## REGISTER 23-7: ADRESL: A/D RESULT LOW BYTE REGISTER, RIGHT JUSTIFIED (ADFM = 1)

| R/W-x  |
|--------|--------|--------|--------|--------|--------|--------|--------|
| ADRES7 | ADRES6 | ADRES5 | ADRES4 | ADRES3 | ADRES2 | ADRES1 | ADRES0 |
| bit 7  |        |        |        |        |        |        |        |

### 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 7-0

### ADRES<7:0>: A/D Result Low Byte bits

The ANCONx registers are used to configure the operation of the I/O pin associated with each analog channel. Clearing an ANSELx bit configures the corresponding pin (ANx) to operate as a digital only I/O. Setting a bit configures the pin to operate as an analog input for either the A/D Converter or the comparator

module, with all digital peripherals disabled and digital inputs read as '0'.

As a rule, I/O pins that are multiplexed with analog inputs default to analog operation on any device Reset.

## REGISTER 23-8: ANCON0: A/D PORT CONFIGURATION REGISTER 0

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANSEL7 <sup>(1)</sup>	ANSEL6 <sup>(1)</sup>	ANSEL5 <sup>(1)</sup>	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0
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 7-0

### ANSEL<7:0>: Analog Port Configuration bits (AN7 and AN0)<sup>(1)</sup>

1 = Pin configured as an analog channel: digital input disabled and any inputs read as '0'

0 = Pin configured as a digital port

**Note 1:** AN14 through AN11 and AN7 to AN5 are implemented only on 40/44-pin and 64-pin devices. For 28-pin devices, the corresponding ANSELx bits are still implemented for these channels, but have no effect.

# PIC18F66K80 FAMILY

## REGISTER 23-9: ANCON1: A/D PORT CONFIGURATION REGISTER 1

U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ANSEL14 <sup>(1)</sup>	ANSEL13 <sup>(1)</sup>	ANSEL12 <sup>(1)</sup>	ANSEL11 <sup>(1)</sup>	ANSEL10	ANSEL9	ANSEL8
bit 7							bit 0

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

- bit 7      **Unimplemented:** Read as '0'
- bit 6      **ANSEL14:** RD3/C2INB Pin Analog Enable bit<sup>(1)</sup>  
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'  
0 = Pin is configured as a digital port
- bit 5      **ANSEL13:** RD2/C2INA Pin Analog Enable bit<sup>(1)</sup>  
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'  
0 = Pin is configured as a digital port
- bit 4      **ANSEL12:** RD1/C1INB Pin Analog Enable bit<sup>(1)</sup>  
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'  
0 = Pin is configured as a digital port
- bit 3      **ANSEL11:** RD0/C1INA Pin Analog Enable bit<sup>(1)</sup>  
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'  
0 = Pin is configured as a digital port
- bit 2-0     **ANSEL11<10:8>:** Analog Port Configuration bits (AN10 through AN8)  
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'  
0 = Pin configured as a digital port

**Note 1:** AN14 through AN11 and AN7 to AN5 are implemented only on 40/44-pin and 64-pin devices. For 28-pin devices, the corresponding ANSELx bits are still implemented for these channels, but have no effect.

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (AVDD and AVss) or the voltage level on the RA3/VREF+/AN3 and RA2/VREF-/AN2 pins. VREF+ has two additional internal voltage reference selections: 2.0V and 4.1V.

The A/D Converter can uniquely operate while the device is in Sleep mode. To operate in **Sleep**, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

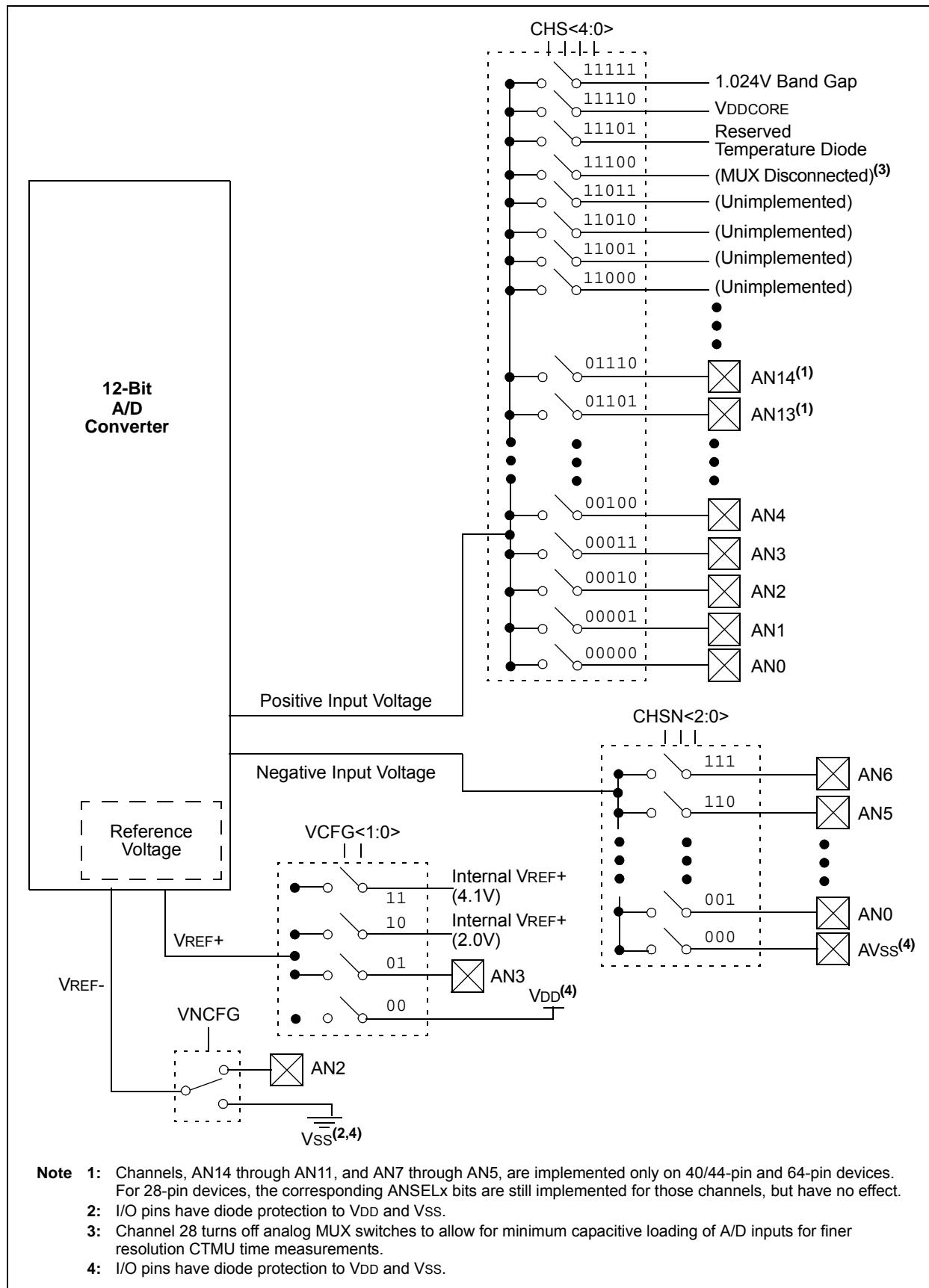
The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

Each port pin associated with the A/D Converter can be configured as an analog input or 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<1>) is cleared and the A/D Interrupt Flag bit, ADIF (PIR1<6>), is set.

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

The block diagram of the A/D module is shown in [Figure 23-4](#).

**FIGURE 23-4: A/D BLOCK DIAGRAM**



# PIC18F66K80 FAMILY

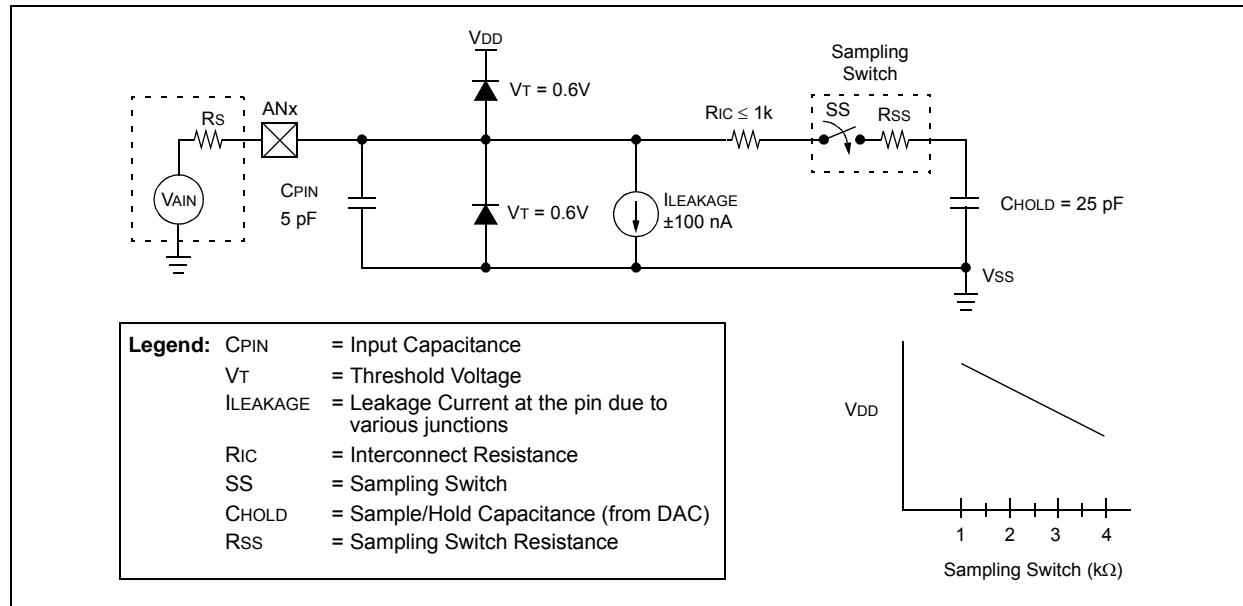
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion can start. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see [Section 23.3 "A/D Acquisition Requirements"](#). After 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.

To do an A/D conversion, follow these steps:

1. Configure the A/D module:
  - Configure the required A/D pins as analog pins (ADCON0 and ANCON1)
  - Set the voltage reference (ADCON1)
  - Select the A/D positive and negative input channels (ADCON0 and ADCON1)
  - Select the A/D acquisition time (ADCON2)
  - Select the A/D conversion clock (ADCON2)
  - Turn on the A/D module (ADCON0)
2. Configure the A/D interrupt (if desired):
  - Clear the ADIF bit (PIR1<6>)
  - Set the ADIE bit (PIE1<6>)
  - Set the GIE bit (INTCON<7>)
3. Wait the required acquisition time (if required).
4. Start the conversion:
  - Set the GO/DONE bit (ADCON0<1>)
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) and, if required, clear bit, ADIF.
7. For the next conversion, begin with Step 1 or 2, as required.

The A/D conversion time per bit is defined as TAD. Before the next acquisition starts, a minimum wait of 2 TAD is required.

**FIGURE 23-5: ANALOG INPUT MODEL**



### 23.3 A/D Acquisition Requirements

For the A/D Converter to meet its specified accuracy, the Charge Holding (CHOLD) capacitor must be allowed to fully charge to the input channel voltage level. The analog input model is shown in [Figure 23-5](#). The source impedance ( $R_s$ ) and the internal sampling switch ( $R_{ss}$ ) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch ( $R_{ss}$ ) impedance varies over the device voltage ( $V_{DD}$ ).

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 or 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 23-1](#) can be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

[Equation 23-3](#) shows the calculation of the minimum required acquisition time, TACQ. This calculation is based on the following application system assumptions:

CHOLD	= 25 pF
$R_s$	= 2.5 kΩ
Conversion Error	≤ 1/2 LSb
$V_{DD}$	= 3V → $R_{ss} = 2\text{ k}\Omega$
Temperature	= 85°C (system max.)

### EQUATION 23-1: ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

### EQUATION 23-2: A/D MINIMUM CHARGING TIME

$$\begin{aligned} V_{HOLD} &= (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/CHOLD(RIC + RSS + RS))}) \\ \text{or} \\ T_C &= -(CHOLD)(RIC + RSS + RS) \ln(1/2048) \end{aligned}$$

### EQUATION 23-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= T_{AMP} + T_C + T_{COFF} \\ T_{AMP} &= 0.2 \mu s \\ T_{COFF} &= (Temp - 25^\circ C)(0.02 \mu s/\text{ }^\circ C) \\ &\quad (85^\circ C - 25^\circ C)(0.02 \mu s/\text{ }^\circ C) \\ &\quad 1.2 \mu s \\ \text{Temperature coefficient is only required for temperatures } > 25^\circ C. \text{ Below } 25^\circ C, T_{COFF} = 0 \text{ ms.} \\ T_C &= -(CHOLD)(RIC + RSS + RS) \ln(1/2048) \mu s \\ &\quad -(25 \text{ pF})(1 \text{ k}\Omega + 2 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(0.0004883) \mu s \\ &\quad 1.05 \mu s \\ T_{ACQ} &= 0.2 \mu s + 1.05 \mu s + 1.2 \mu s \\ &\quad 2.45 \mu s \end{aligned}$$

# PIC18F66K80 FAMILY

## 23.4 Selecting and Configuring Automatic Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/DONE bit is set.

When the 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 GO/DONE bit.

This occurs when the ACQT<sub><2:0></sub> bits (ADCON2<5:3>) remain in their Reset state ('000'), which is compatible with devices that do not offer programmable acquisition times.

If desired, the ACQTx bits can be set to select a programmable acquisition time for the A/D module. When the 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 GO/DONE bit.

In either case, when the conversion is completed, the 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.

## 23.5 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 14 TAD per 12-bit conversion. The source of the A/D conversion clock is software selectable.

The possible options for TAD are:

- 2 Tosc
- 4 Tosc
- 8 Tosc
- 16 Tosc
- 32 Tosc
- 64 Tosc
- Using the 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. (For more information, see Parameter 130 in Table 31-26.)

Table 23-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

**TABLE 23-1: TAD vs. DEVICE OPERATING FREQUENCIES**

AD Clock Source (TAD)		Maximum Device Frequency
Operation	ADCS<2:0>	
2 Tosc	000	2.50 MHz
4 Tosc	100	5.00 MHz
8 Tosc	001	10.00 MHz
16 Tosc	101	20.00 MHz
32 Tosc	010	40.00 MHz
64 Tosc	110	64.00 MHz
RC <sup>(2)</sup>	x11	1.00 MHz <sup>(1)</sup>

**Note 1:** The RC source has a typical TAD time of 4  $\mu$ s.

**2:** For device frequencies above 1 MHz, the device must be in Sleep mode for the entire conversion or the A/D accuracy may be out of specification.

## 23.6 Configuring Analog Port Pins

The ANCON0, ANCON1, TRISA, TRISB, TRISC and TRISD registers control the operation of the A/D port pins. The port pins needed as analog inputs must have their corresponding TRISx bits set (input). If the TRISx 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 CHS<3:0> bits and the TRISx bits.

**Note:** When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will be accurately converted.

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.

## 23.7 A/D Conversions

Figure 23-6 shows the operation of the A/D Converter after the GO/DONE bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 23-7 shows the operation of the A/D Converter after the GO/DONE bit has been set, the ACQT<2:0> bits set to '010' and a 4 TAD acquisition time selected.

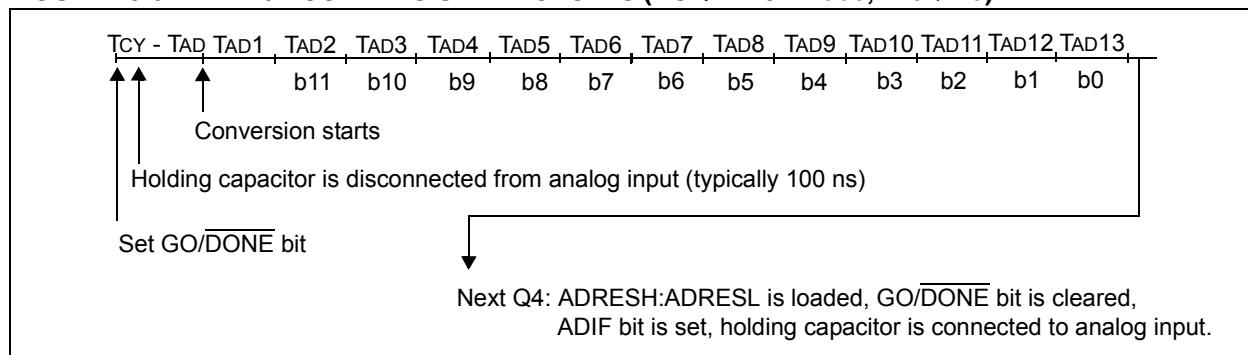
Clearing the GO/DONE bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. This means the

ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

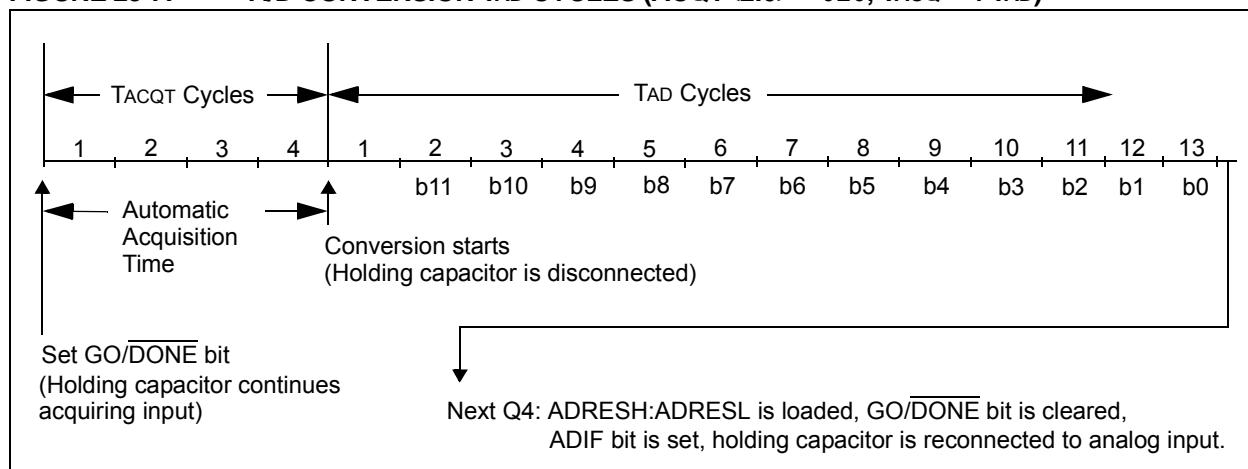
After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

**Note:** The GO/DONE bit should **NOT** be set in the same instruction that turns on the A/D.

**FIGURE 23-6: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**



**FIGURE 23-7: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



# PIC18F66K80 FAMILY

---

## 23.8 Use of the Special Event Triggers

A/D conversion can be started by the Special Event Trigger of any of these modules:

- CCP2 – Requires CCP2M<3:0> bits (CCP2CON<3:0>) set at '1011'†
- ECCP1
- CTMU – Requires the setting of the CTTRIG bit (CTMUCONH<0>)
- Timer1

To start an A/D conversion:

- The A/D module must be enabled (ADON = 1)
- The appropriate analog input channel selected
- The minimum acquisition period set one of these ways:
  - Timing provided by the user
  - Selection made of an appropriate TACQ time

With these conditions met, the trigger sets the GO/DONE bit and the A/D acquisition starts.

If the A/D module is not enabled (ADON = 0), the module ignores the Special Event Trigger.

**Note:** With an ECCP1 or CCP2 trigger, Timer1 or Timer3 is cleared. The timers reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH:ADRESL to the desired location). If the A/D module is not enabled, the Special Event Trigger is ignored by the module, but the timer's counter resets.

## 23.9 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 ACQT<2:0> and ADCS<2:0> bits in ADCON2 should be updated in accordance with the power-managed mode clock that will be used.

After the power-managed mode is entered (either of the power-managed Run modes), an A/D acquisition or conversion may be started. Once an acquisition or conversion is started, the device should continue to be clocked by the same power-managed mode clock source until the conversion has been completed. If desired, the device may be placed into the corresponding power-managed Idle mode during the conversion.

If the power-managed mode clock frequency is less than 1 MHz, the A/D RC clock source should be selected.

Operation in Sleep mode requires that the A/D RC clock be selected. If bits, ACQT<2:0>, 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 into Sleep mode. The IDLEN and SCS<1:0> bits in the OSCCON register must have already been cleared prior to starting the conversion.

# PIC18F66K80 FAMILY

---

**TABLE 23-2: REGISTERS ASSOCIATED WITH THE A/D MODULE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
ADRESH	A/D Result Register High Byte							
ADRESL	A/D Result Register Low Byte							
ADC0N0	—	CHS4	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
ADC0N1	TRIGSEL1	TRIGSEL0	VCFG1	VCFG0	VNCFG	CHSN2	CHSN1	CHSN0
ADC0N2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
ANCON0	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0
ANCON1	—	ANSEL14	ANSEL13	ANSEL12	ANSEL11	ANSEL10	ANSEL9	ANSEL8
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	—	RA3	RA2	RA1	RA0
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
PORTE	RE7	RE6	RE5	RE4	RE3	—	RE1	RE0
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	—	TRISE2	TRISE1	TRISE0
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** These bits are available only in certain oscillator modes when the FOSC2 Configuration bit = 0. If that Configuration bit is cleared, this signal is not implemented.

# **PIC18F66K80 FAMILY**

---

---

## **NOTES:**

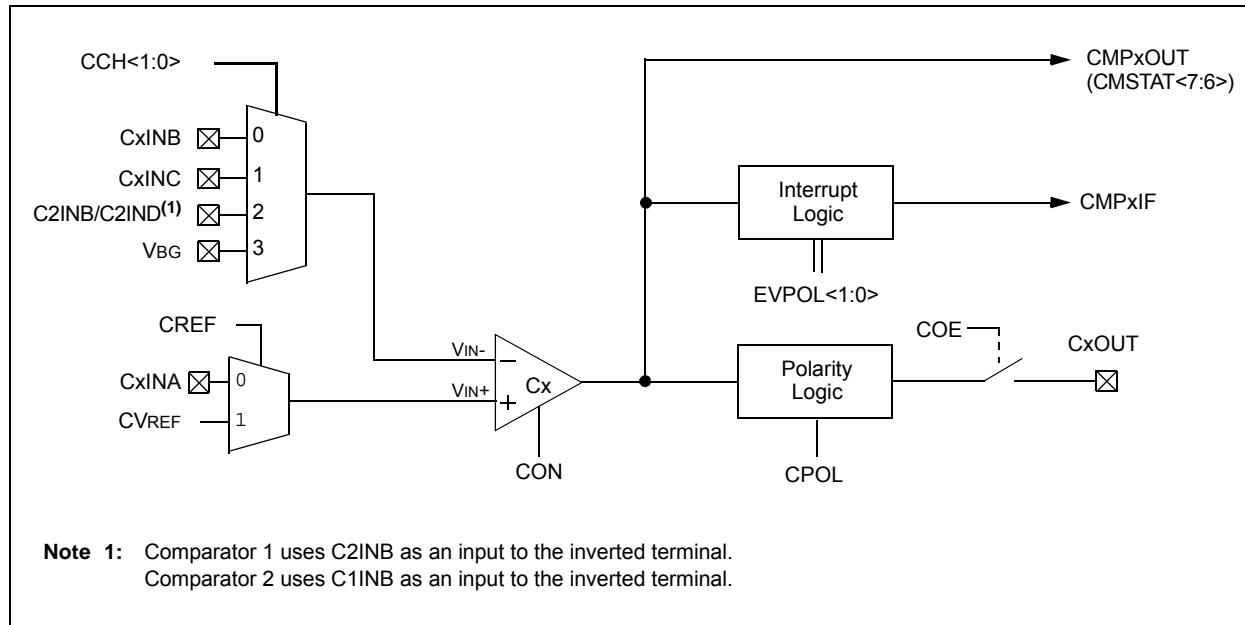
## 24.0 COMPARATOR MODULE

The analog comparator module contains two comparators that can be independently configured in a variety of ways. The inputs can be selected from the analog inputs and two internal voltage references. The digital outputs are available at the pin level and can also be read through the control register. Multiple output and interrupt event generation are also available. A generic single comparator from the module is shown in Figure 24-1.

Key features of the module includes:

- Independent comparator control
- Programmable input configuration
- Output to both pin and register levels
- Programmable output polarity
- Independent interrupt generation for each comparator with configurable interrupt-on-change

**FIGURE 24-1: COMPARATOR SIMPLIFIED BLOCK DIAGRAM**



## 24.1 Registers

The CMxCON registers (CM1CON and CM2CON) select the input and output configuration for each comparator, as well as the settings for interrupt generation (see [Register 24-1](#)).

The CMSTAT register ([Register 24-2](#)) provides the output results of the comparators. The bits in this register are read-only.

# PIC18F66K80 FAMILY

## REGISTER 24-1: CMxCON: COMPARATOR CONTROL x REGISTER

R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CON	COE	CPOL	EVPOL1	EVPOLO	CREF	CCH1	CCH0
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	<b>CON:</b> Comparator Enable bit 1 = Comparator is enabled 0 = Comparator is disabled
bit 6	<b>COE:</b> Comparator Output Enable bit 1 = Comparator output is present on the CxOUT pin 0 = Comparator output is internal only
bit 5	<b>CPOL:</b> Comparator Output Polarity Select bit 1 = Comparator output is inverted 0 = Comparator output is not inverted
bit 4-3	<b>EVPOL&lt;1:0&gt;:</b> Interrupt Polarity Select bits 11 = Interrupt generation on any change of the output <sup>(1)</sup> 10 = Interrupt generation only on high-to-low transition of the output 01 = Interrupt generation only on low-to-high transition of the output 00 = Interrupt generation is disabled
bit 2	<b>CREF:</b> Comparator Reference Select bit (non-inverting input) 1 = Non-inverting input connects to internal CVREF voltage 0 = Non-inverting input connects to CxINA pin
bit 1-0	<b>CCH&lt;1:0&gt;:</b> Comparator Channel Select bits 11 = Inverting input of comparator connects to VBG 10 = Inverting input of comparator connects to C2INB pin <sup>(2)</sup> 01 = Inverting input of comparator connects to CxINC pin 00 = Inverting input of comparator connects to C1INB pin <sup>(2)</sup>

**Note 1:** The CMPxIF is automatically set any time this mode is selected and must be cleared by the application after the initial configuration.

**2:** Comparator 1 uses C2INB as an input to the inverting terminal. Comparator 2 uses C1INB as an input to the inverted terminal.

# PIC18F66K80 FAMILY

## REGISTER 24-2: CMSTAT: COMPARATOR STATUS REGISTER

R-x	R-x	U-0	U-0	U-0	U-0	U-0	U-0
CMP2OUT	CMP1OUT	—	—	—	—	—	—
bit 7							

### 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      **CMP2OUT:CMP1OUT:** Comparator x Status bits

If CPOL (CMxCON<5>) = 0 (non-inverted polarity):

1 = Comparator x's VIN+ > VIN-

0 = Comparator x's VIN+ < VIN-

If CPOL = 1 (inverted polarity):

1 = Comparator x's VIN+ < VIN-

0 = Comparator x's VIN+ > VIN-

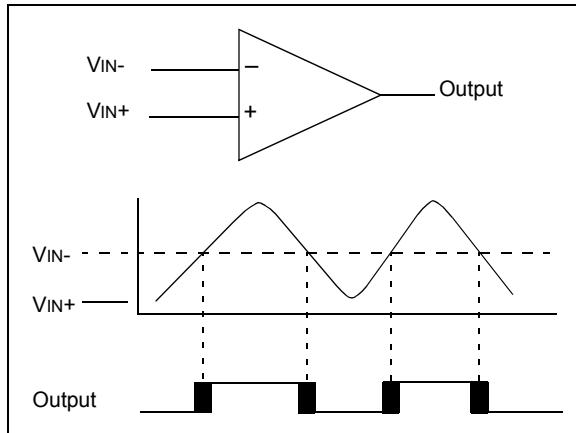
bit 4-0      **Unimplemented:** Read as '0'

# PIC18F66K80 FAMILY

## 24.2 Comparator Operation

A single comparator is shown in Figure 24-2, along with the relationship between the analog input levels and the digital output. When the analog input at  $V_{IN+}$  is less than the analog input,  $V_{IN-}$ , the output of the comparator is a digital low level. When the analog input at  $V_{IN+}$  is greater than the analog input,  $V_{IN-}$ , the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 24-2 represent the uncertainty due to input offsets and response time.

FIGURE 24-2: SINGLE COMPARATOR



## 24.3 Comparator Response Time

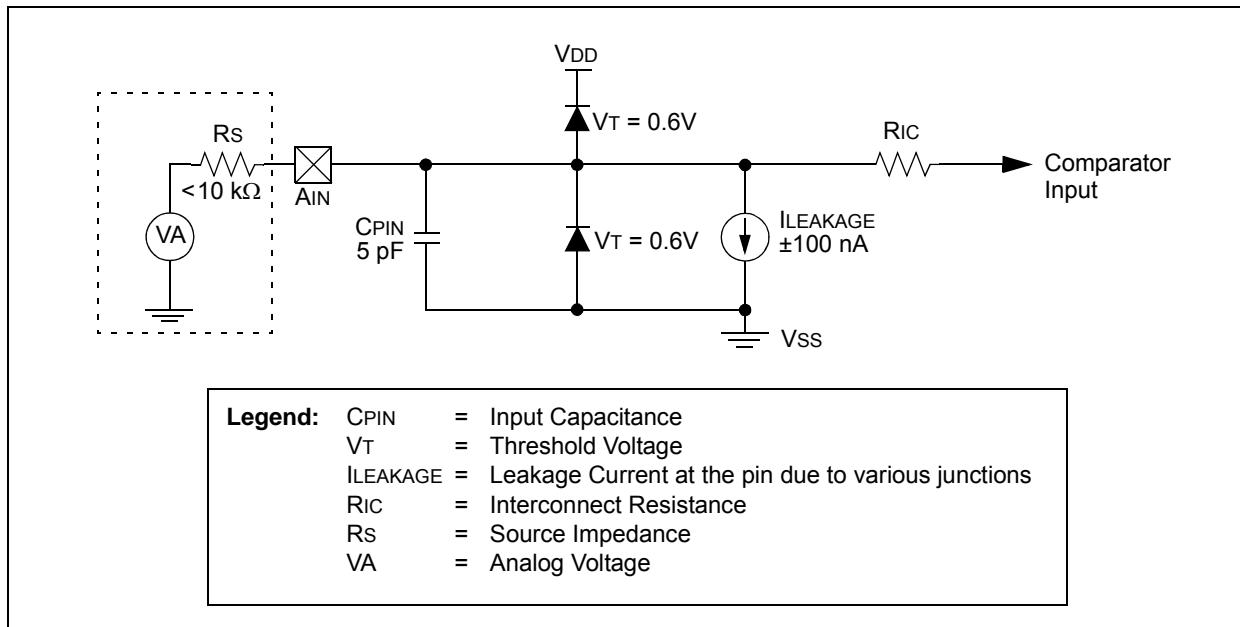
Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response to a comparator input change. Otherwise, the maximum delay of the comparators should be used (see Section 31.0 “Electrical Characteristics”).

## 24.4 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 24-3. Since the analog pins are connected to a digital output, they have reverse biased diodes to  $V_{DD}$  and  $V_{SS}$ . The analog input, therefore, must be between  $V_{SS}$  and  $V_{DD}$ . 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  $10\text{ k}\Omega$  is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 24-3: COMPARATOR ANALOG INPUT MODEL



## 24.5 Comparator Control and Configuration

Each comparator has up to eight possible combinations of inputs: up to four external analog inputs and one of two internal voltage references.

All of the comparators allow a selection of the signal from pin, CxINA, or the voltage from the comparator reference (CVREF) on the non-inverting channel. This is compared to either C1INB, CxINC, C2INB or the microcontroller's fixed internal reference voltage (VBG, 1.024V nominal) on the inverting channel. The comparator inputs and outputs are tied to fixed I/O pins, defined in [Table 24-1](#). The available comparator configurations and their corresponding bit settings are shown in [Figure 24-4](#).

**TABLE 24-1: COMPARATOR INPUTS AND OUTPUTS**

Comparator	Input or Output	I/O Pin <sup>(†)</sup>
1	C1INA (VIN+)	RB0/RD0
	C1INB (VIN-)	RB1/RD1
	C1INC (VIN-)	RA1
	C2INB(VIN-)	RA5/RD3
	C1OUT	RB2/RE1
2	C2INA(VIN+)	RB4/RD2
	C2INB(VIN-)	RA5/RD3
	C2INC(VIN-)	RA2
	C2OUT	RB3/RE2

<sup>†</sup> The I/O pin is dependent on package type.

### 24.5.1 COMPARATOR ENABLE AND INPUT SELECTION

Setting the CON bit of the CMxCON register (CMxCON<7>) enables the comparator for operation. Clearing the CON bit disables the comparator, resulting in minimum current consumption.

The CCH<1:0> bits in the CMxCON register (CMxCON<1:0>) direct either one of three analog input pins, or the Internal Reference Voltage (VBG), to the comparator, VIN-. Depending on the comparator operating mode, either an external or internal voltage reference may be used.

The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly.

The external reference is used when CREF = 0 (CMxCON<2>) and VIN+ is connected to the CxINA pin. When external voltage references are used, the comparator module can be configured to have the reference sources externally. The reference signal must be between Vss and VDD and can be applied to either pin of the comparator.

The comparator module also allows the selection of an internally generated voltage reference (CVREF) from the comparator voltage reference module. This module is described in more detail in [Section 25.0 “Comparator Voltage Reference Module”](#). The reference from the comparator voltage reference module is only available when CREF = 1. In this mode, the internal voltage reference is applied to the comparator's VIN+ pin.

**Note:** The comparator input pin selected by CCH<1:0> must be configured as an input by setting both the corresponding TRIS bit and the corresponding ANSELx bit in the ANCONx register.

### 24.5.2 COMPARATOR ENABLE AND OUTPUT SELECTION

The comparator outputs are read through the CMSTAT register. The CMSTAT<6> bit reads the Comparator 1 output, CMSTAT<7> reads the Comparator 2 output. These bits are read-only.

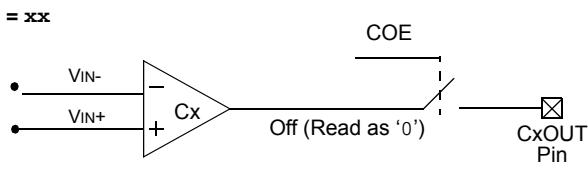
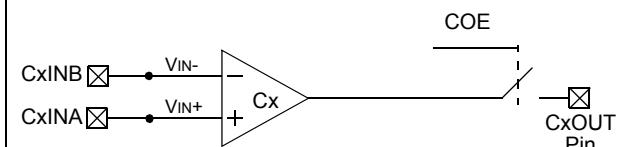
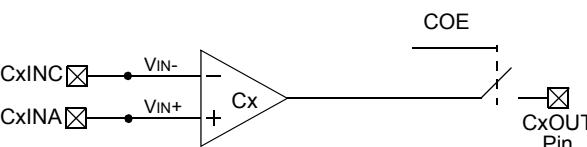
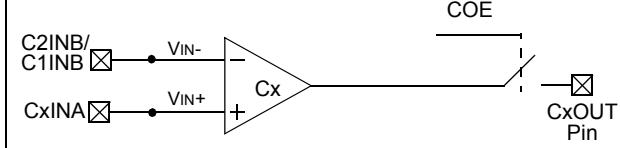
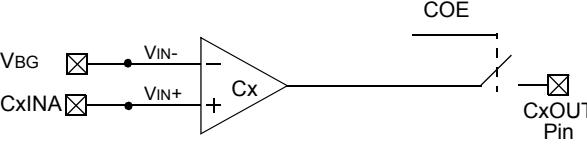
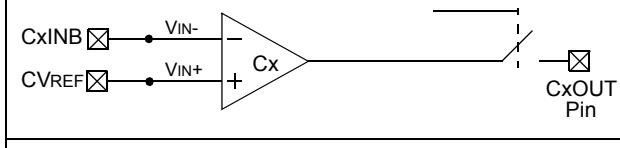
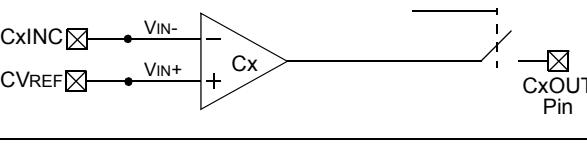
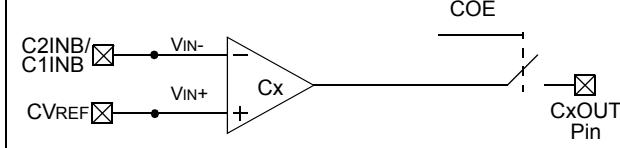
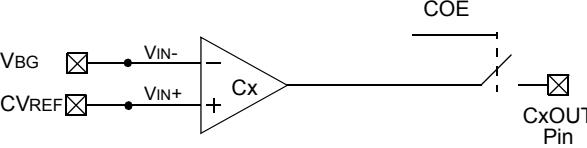
The comparator outputs may also be directly output to the RE2 and RE1 pins by setting the COE bit (CMxCON<6>). When enabled, multiplexers in the output path of the pins switch to the output of the comparator. While in this mode, the TRISE<2:1> bits still function as the digital output enable bits for the RE2, and RE1 pins.

By default, the comparator's output is at logic high whenever the voltage on VIN+ is greater than on VIN-. The polarity of the comparator outputs can be inverted using the CPOL bit (CMxCON<5>).

The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications, as discussed in [Section 24.2 “Comparator Operation”](#).

# PIC18F66K80 FAMILY

**FIGURE 24-4: COMPARATOR CONFIGURATIONS**

<b>Comparator Off</b> <b>CON = 0, CREF = x, CCH&lt;1:0&gt; = xx</b>	
	
<b>Comparator CxINB &gt; CxINA Compare</b> <b>CON = 1, CREF = 0, CCH&lt;1:0&gt; = 00</b>	<b>Comparator CxINC &gt; CxINA Compare</b> <b>CON = 1, CREF = 0, CCH&lt;1:0&gt; = 01</b>
	
<b>Comparator C2INB/C1INB &gt; CxINA Compare</b> <b>CON = 1, CREF = 0, CCH&lt;1:0&gt; = 10</b>	<b>Comparator VBG &gt; CxINA Compare</b> <b>CON = 1, CREF = 0, CCH&lt;1:0&gt; = 11</b>
	
<b>Comparator CxINB &gt; CVREF Compare</b> <b>CON = 1, CREF = 1, CCH&lt;1:0&gt; = 00</b>	<b>Comparator CxINC &gt; CVREF Compare</b> <b>CON = 1, CREF = 1, CCH&lt;1:0&gt; = 01</b>
	
<b>Comparator C2INB/C1INB &gt; CVREF Compare</b> <b>CON = 1, CREF = 1, CCH&lt;1:0&gt; = 10</b>	<b>Comparator VBG &gt; CVREF Compare</b> <b>CON = 1, CREF = 1, CCH&lt;1:0&gt; = 11</b>
	
<b>Note 1:</b> VBG is the Internal Reference Voltage (see <a href="#">Table 31-2</a> ).	

## 24.6 Comparator Interrupts

The comparator interrupt flag is set whenever any of the following occurs:

- Low-to-high transition of the comparator output
- High-to-low transition of the comparator output
- Any change in the comparator output

The comparator interrupt selection is done by the EVPOL<1:0> bits in the CMxCON register (CMxCON<4:3>).

In order to provide maximum flexibility, the output of the comparator may be inverted using the CPOL bit in the CMxCON register (CMxCON<5>). This is functionally identical to reversing the inverting and non-inverting inputs of the comparator for a particular mode.

An interrupt is generated on the low-to-high or high-to-low transition of the comparator output. This mode of interrupt generation is dependent on EVPOL<1:0> in the CMxCON register. When EVPOL<1:0> = 01 or 10, the interrupt is generated on a low-to-high or high-to-low transition of the comparator output. Once the interrupt is generated, it is required to clear the interrupt flag by software.

When EVPOL<1:0> = 11, the comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMSTAT<7:6>, to determine the actual change that occurred.

The CMPxIF<2:0> (PIR4<5:4>) bits are the Comparator Interrupt Flags. The CMPxIF bits must be reset by clearing them. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated. [Table 24-2](#) shows the interrupt generation with respect to comparator input voltages and EVPOL bit settings.

Both the CMPxIE bits (PIE4<5:4>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. 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 bits will still be set if an interrupt condition occurs.

A simplified diagram of the interrupt section is shown in [Figure 24-3](#).

**Note:** CMPxIF will not be set when EVPOL<1:0> = 00.

**TABLE 24-2: COMPARATOR INTERRUPT GENERATION**

CPOL	EVPOL<1:0>	Comparator Input Change	CxOUT Transition	Interrupt Generated
0	00	VIN+ > VIN-	Low-to-High	No
		VIN+ < VIN-	High-to-Low	No
	01	VIN+ > VIN-	Low-to-High	Yes
		VIN+ < VIN-	High-to-Low	No
	10	VIN+ > VIN-	Low-to-High	No
		VIN+ < VIN-	High-to-Low	Yes
	11	VIN+ > VIN-	Low-to-High	Yes
		VIN+ < VIN-	High-to-Low	Yes
1	00	VIN+ > VIN-	High-to-Low	No
		VIN+ < VIN-	Low-to-High	No
	01	VIN+ > VIN-	High-to-Low	No
		VIN+ < VIN-	Low-to-High	Yes
	10	VIN+ > VIN-	High-to-Low	Yes
		VIN+ < VIN-	Low-to-High	No
	11	VIN+ > VIN-	High-to-Low	Yes
		VIN+ < VIN-	Low-to-High	Yes

# PIC18F66K80 FAMILY

---

## 24.7 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.

To minimize power consumption while in Sleep mode, turn off the comparators (CON = 0) before entering Sleep. If the device wakes up from Sleep, the contents of the CMxCON register are not affected.

## 24.8 Effects of a Reset

A device Reset forces the CMxCON registers to their Reset state. This forces both comparators and the voltage reference to the OFF state.

TABLE 24-3: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
CM1CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
CM2CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
CVRCON	CVREN	CVROE	CVRSS	CVR4	CVR3	CVR2	CVR1	CVR0
CMSTAT	CMP2OUT	CMP1OUT	—	—	—	—	—	—
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
ANCON0	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0
ANCON1	—	ANSEL14	ANSEL13	ANSEL12	ANSEL11	ANSEL10	ANSEL9	ANSEL8
PMD2	—	—	—	—	MODMD	ECANMD	CMP2MD	CMP1MD

Legend: — = unimplemented, read as '0'.

## 25.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 32-tap resistor ladder network that provides a selectable reference voltage. Although its primary purpose is to provide a reference for the analog comparators, it may also be used independently of them.

A block diagram of the module is shown in [Figure 25-1](#). The resistor ladder is segmented to provide a range 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.

### 25.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register ([Register 25-1](#)). The comparator voltage reference provides a range of output voltage with 32 levels.

The CVR<4:0> selection bits (CVRCON<4:0>) offer a range of output voltages. [Equation 25-1](#) shows the how the comparator voltage reference is computed.

#### REGISTER 25-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| CVREN | CVROE | CVRSS | CVR4  | 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 **CVROE:** Comparator VREF Output Enable bit

1 = CVREF voltage level is output on CVREF pin  
0 = CVREF voltage level is disconnected from CVREF pin

bit 5 **CVRSS:** Comparator VREF Source Selection bit

1 = Comparator reference source, CVRSRC = VREF+ – VREF-  
0 = Comparator reference source, CVRSRC = AVDD – AVSS

bit 4-0 **CVR<4:0>:** Comparator VREF Value Selection  $0 \leq \text{CVR}<4:0> \leq 31$  bits

When CVRSS = 1:

$\text{CVREF} = (\text{VREF}-) + (\text{CVR}<4:0>/32) \cdot (\text{VREF}+ - \text{VREF}-)$

When CVRSS = 0:

$\text{CVREF} = (\text{AVss}) + (\text{CVR}<4:0>/32) \cdot (\text{AVDD} - \text{AVss})$

#### EQUATION 25-1:

If CVRSS = 1:

$$\text{CVREF} = \left( \text{VREF}- + \frac{\text{CVR}<4:0>}{32} \right) \cdot (\text{VREF}+ - \text{VREF}-)$$

If CVRSS = 0:

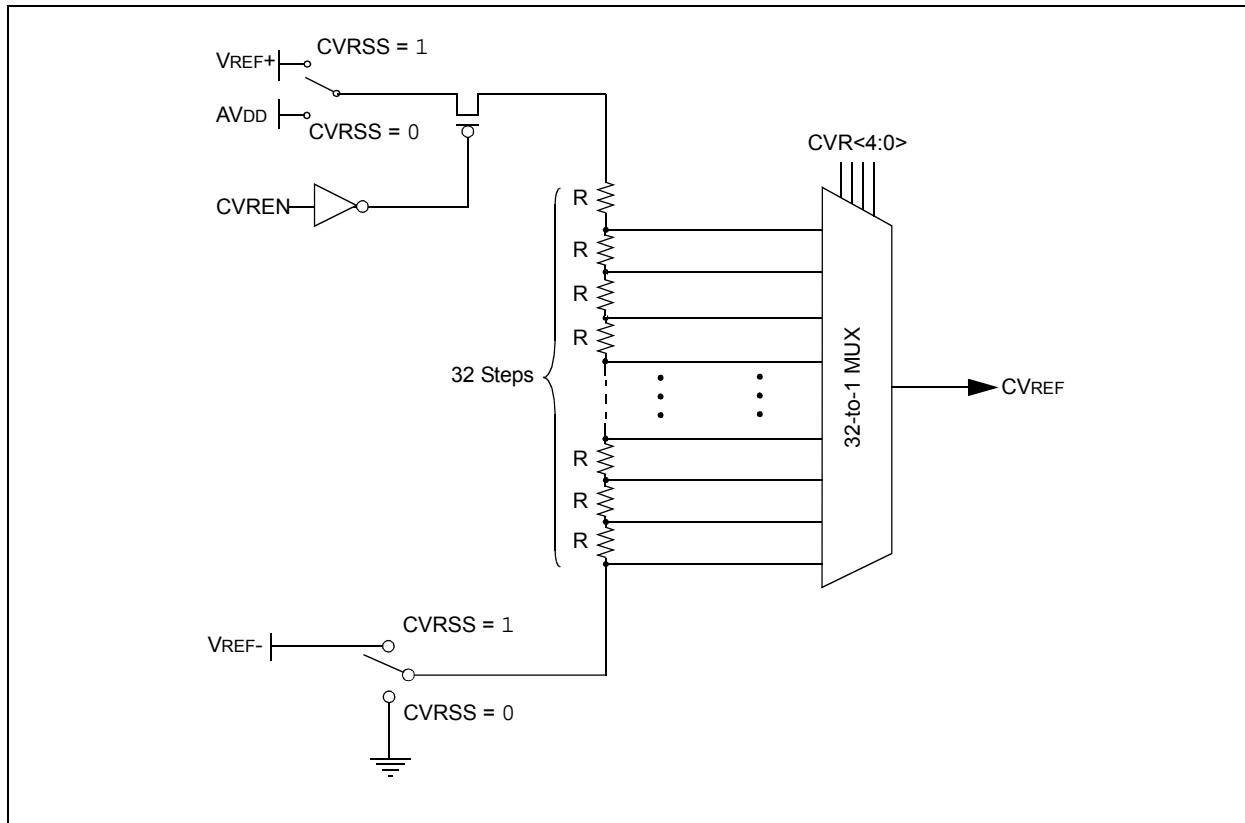
$$\text{CVREF} = \left( \text{AVss} + \frac{\text{CVR}<4:0>}{32} \right) \cdot (\text{AVDD} - \text{AVss})$$

The comparator reference supply voltage can come from either VDD and Vss, or the external VREF+ and VREF- that are multiplexed with RA3 and RA2. The voltage source is selected by the CVRSS bit (CVRCON<5>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see [Table 31-2](#) in [Section 31.0 “Electrical Characteristics”](#)).

# PIC18F66K80 FAMILY

FIGURE 25-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM



## 25.2 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 25-1) 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 31.0 “Electrical Characteristics”](#).

## 25.3 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.

## 25.4 Effects of a Reset

A device Reset disables the voltage reference by clearing bit, CVREN (CVRCON<7>). This Reset also disconnects the reference from the RF5 pin by clearing bit, CVROE (CVRCON<6>).

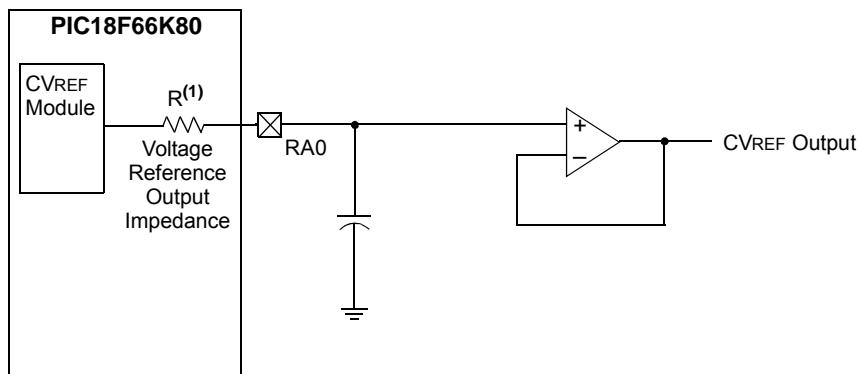
## 25.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RA0 pin if the CVROE bit is set. Enabling the voltage reference output onto RA0 when it is configured as a digital input will increase current consumption. Connecting RA0 as a digital output with CVRSS enabled will also increase current consumption.

The RA0 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 25-2 shows an example buffering technique.

# PIC18F66K80 FAMILY

FIGURE 25-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE



Note 1: R is dependent upon the Voltage Reference Configuration bits, CVRCON<3:0> and CVRCON<5>.

TABLE 25-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CVRCON	CVREN	CVROE	CVRSS	CVR4	CVR3	CVR2	CVR1	CVR0
CM1CON	CON	COE	CPOL	EVPOL1	EVPOLO	CREF	CCH1	CCH0
CM2CON	CON	COE	CPOL	EVPOL1	EVPOLO	CREF	CCH1	CCH0
TRISA	TRISA7	TRISA6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0
ANCON0	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0

Legend: — = unimplemented, read as '0'. Shaded cells are not used with the comparator voltage reference.

# PIC18F66K80 FAMILY

---

---

## NOTES:

## 26.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18F66K80 family of devices has a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point. If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

The High/Low-Voltage Detect Control register ([Register 26-1](#)) completely controls the operation of the HLVD module. This allows the circuitry to be “turned off” by the user under software control, which minimizes the current consumption for the device.

The module’s block diagram is shown in [Figure 26-1](#).

### REGISTER 26-1: HLVDCON: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER

R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL3 <sup>(1)</sup>	HLVDL2 <sup>(1)</sup>	HLVDL1 <sup>(1)</sup>	HLVDL0 <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	<b>VDIRMAG:</b> Voltage Direction Magnitude Select bit 1 = Event occurs when voltage equals or exceeds trip point (HLVDL<3:0>) 0 = Event occurs when voltage equals or falls below trip point (HLVDL<3:0>)
bit 6	<b>BGVST:</b> Band Gap Reference Voltages Stable Status Flag bit 1 = Internal band gap voltage references are stable 0 = Internal band gap voltage references are not stable
bit 5	<b>IRVST:</b> Internal Reference Voltage Stable Flag bit 1 = Indicates that the voltage detect logic will generate the interrupt flag at the specified voltage range 0 = Indicates that the voltage detect logic will not generate the interrupt flag at the specified voltage range and the HLVD interrupt should not be enabled
bit 4	<b>HLVDEN:</b> High/Low-Voltage Detect Power Enable bit 1 = HLVD enabled 0 = HLVD disabled
bit 3-0	<b>HLVDL&lt;3:0&gt;:</b> Voltage Detection Limit bits <sup>(1)</sup> 1111 = External analog input is used (input comes from the HLVDIN pin) 1110 = Maximum setting ... 0000 = Minimum setting

**Note 1:** For the electrical specifications, see Parameter D420 in [Section 31.0 “Electrical Characteristics”](#).

# PIC18F66K80 FAMILY

The module is enabled by setting the HLVDEN bit (HLVDCON<4>). Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit (HLVDCON<5>) is a read-only bit used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

The VDIRMAG bit (HLVDCON<7>) determines the overall operation of the module. When VDIRMAG is cleared, the module monitors for drops in VDD below a predetermined set point. When the bit is set, the module monitors for rises in VDD above the set point.

## 26.1 Operation

When the HLVD 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

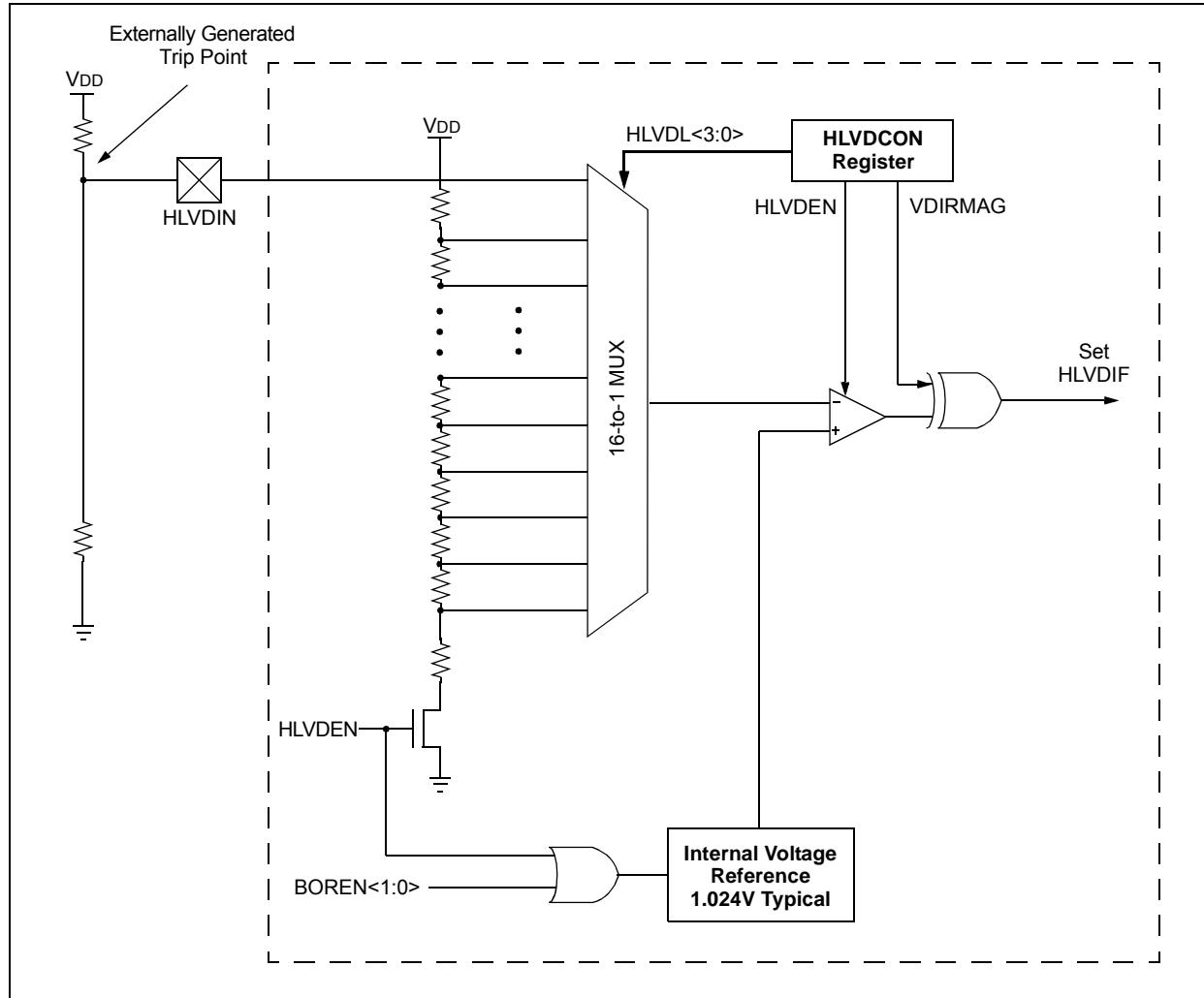
trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

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 HLVDIF bit.

The trip point voltage is software programmable to any one of 16 values. The trip point is selected by programming the HLVDL<3:0> bits (HLVDCON<3:0>).

The HLVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits, HLVDL<3:0>, are set to ‘1111’. In this state, the comparator input is multiplexed from the external input pin, HLVDIN. This gives users the flexibility of configuring the High/Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

**FIGURE 26-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)**



## 26.2 HLVD Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the HLVDL<3:0> bits.
2. Set the VDIRMAG bit to detect high voltage (VDIRMAG = 1) or low voltage (VDIRMAG = 0).
3. Enable the HLVD module by setting the HLVDEN bit.
4. Clear the HLVD interrupt flag (PIR2<2>), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE and GIE bits (PIE2<2> and INTCON<7>, respectively).

An interrupt will not be generated until the IRVST bit is set.

**Note:** Before changing any module settings (VDIRMAG, HLVDL<3:0>), first disable the module (HLVDEN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events.

## 26.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and consume static current. The total current consumption, when enabled, is specified in electrical specification Parameter D022B ([Table 31-11](#)).

Depending on the application, the HLVD module does not need to operate constantly. To reduce current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After such a check, the module could be disabled.

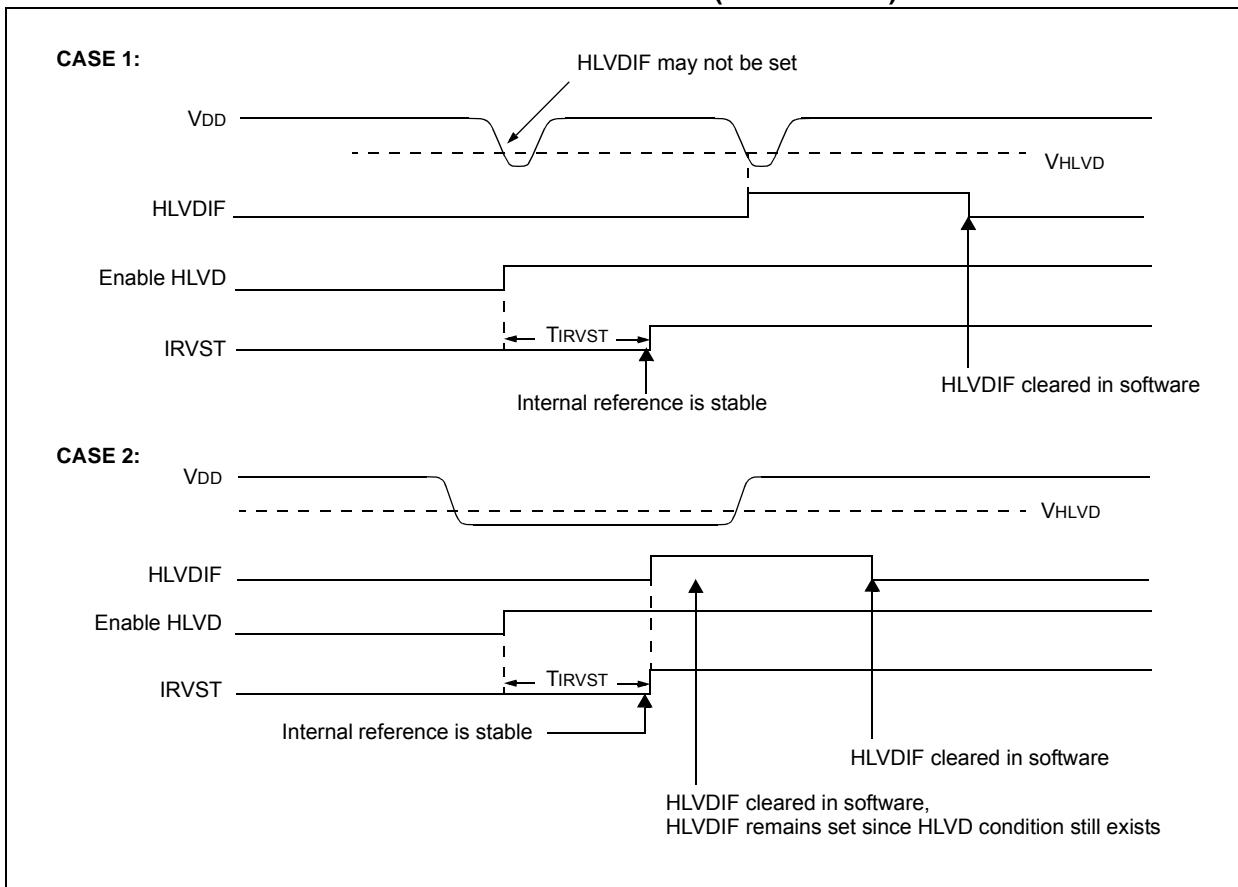
## 26.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification Parameter 37 ([Section 31.0 “Electrical Characteristics”](#)), may be used by other internal circuitry, such as the programmable Brown-out Reset. If the HLVD 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 or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed. It is specified in electrical specification Parameter 37 ([Table 31-11](#)).

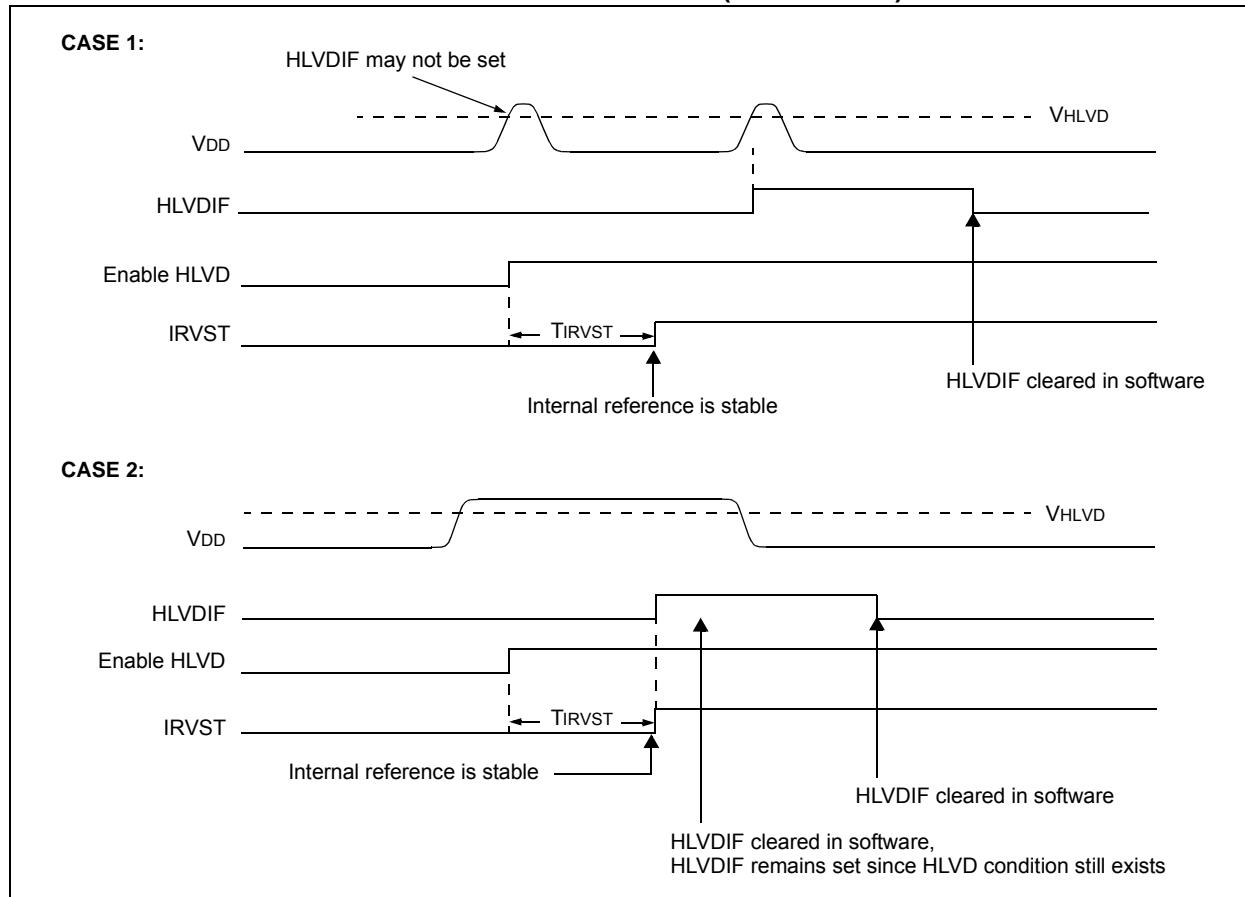
The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (see [Figure 26-2](#) or [Figure 26-3](#)).

# PIC18F66K80 FAMILY

FIGURE 26-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)



**FIGURE 26-3: HIGH-VOLTAGE DETECT OPERATION (VDIRMAG = 1)**

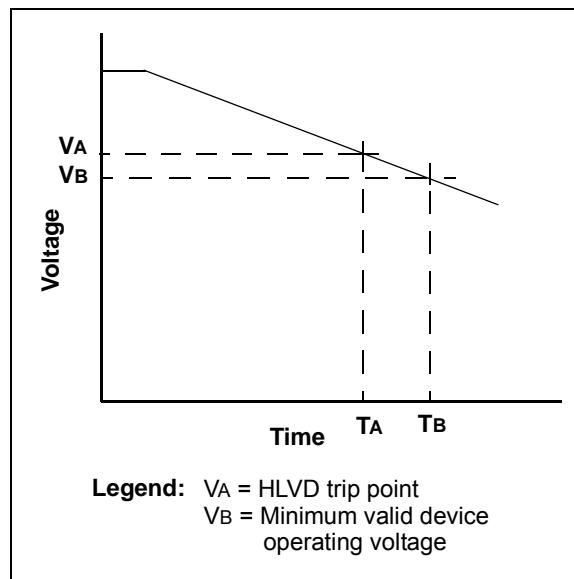


## 26.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a high-voltage detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 26-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage,  $V_A$ , the HLVD logic generates an interrupt at time,  $T_A$ . The interrupt could cause the execution of an ISR, which would allow the application to perform “house-keeping tasks” and a controlled shutdown before the device voltage exits the valid operating range at  $T_B$ . This would give the application a time window, represented by the difference between  $T_A$  and  $T_B$ , to safely exit.

**FIGURE 26-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION**



# PIC18F66K80 FAMILY

---

## 26.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF 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.

## 26.7 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

**TABLE 26-1: REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
HLVDCON	VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL3	HLVDL2	HLVDL1	HLVDL0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR2	OSCFIF	—	—	—	BCLIF	HLVDIF	TMR3IF	TMR3GIF
PIE2	OSCFIE	—	—	—	BCLIE	HLVDIE	TMR3IE	TMR3GIE
IPR2	OSCFIP	—	—	—	BCLIP	HLVDIP	TMR3IP	TMR3GIP
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0

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

**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'.

## 27.0 ECAN MODULE

PIC18F66K80 family devices contain an Enhanced Controller Area Network (ECAN) module. The ECAN module is fully backward compatible with the CAN module available in PIC18CXX8 and PIC18FXX8 devices and the ECAN module in PIC18Fxx80 devices.

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

The ECAN module is a communication controller, implementing the CAN 2.0A or B protocol as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN 2.0B Passive and CAN 2.0B Active versions of the protocol. The module implementation is a full CAN system; however, the CAN specification is not covered within this data sheet. Refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol, CAN 1.2, CAN 2.0A and CAN 2.0B
- DeviceNet™ data bytes filter support
- Standard and extended data frames
- 0-8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Fully backward compatible with the PIC18XXX8 CAN module
- Three modes of operation:
  - Mode 0 – Legacy mode
  - Mode 1 – Enhanced Legacy mode with DeviceNet support
  - Mode 2 – FIFO mode with DeviceNet support
- Support for remote frames with automated handling
- Double-buffered receiver with two prioritized received message storage buffers
- Six buffers programmable as RX and TX message buffers
- 16 full (standard/extended identifier) acceptance filters that can be linked to one of four masks
- Two full acceptance filter masks that can be assigned to any filter
- One full acceptance filter that can be used as either an acceptance filter or acceptance filter mask
- Three dedicated transmit buffers with application specified prioritization and abort capability
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low-power Sleep mode

## 27.1 Module Overview

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

The CAN module supports the following frame types:

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

The CAN module uses the RB2/CANTX and RB3/CANRX pins to interface with the CAN bus. The CANTX and CANRX pins can be placed on alternate I/O pins by setting the CANMX (CONFIG3H<0>) Configuration bit.

For the PIC18F2XK80 and PIC18F4XK80, the alternate pin locations are RC6/CANTX and RC7/CANRX. For the PIC18F6XK80, the alternate pin locations are RE4/CANRX and RE5/CANTX.

In normal mode, the CAN module automatically overrides the appropriate TRIS bit for CANTX. The user must ensure that the appropriate TRIS bit for CANRX is set.

### 27.1.1 MODULE FUNCTIONALITY

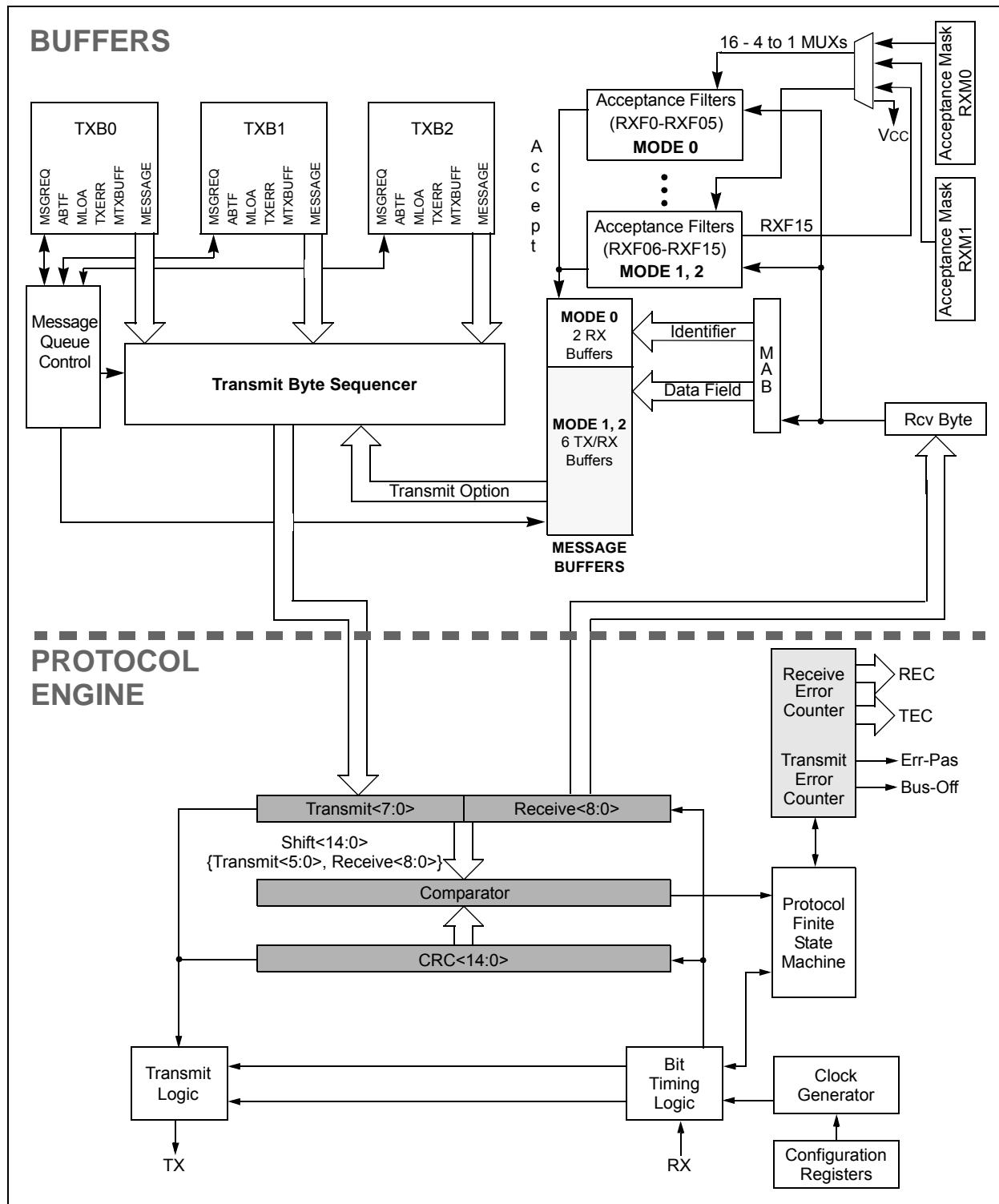
The CAN bus module consists of a protocol engine, message buffering and control (see [Figure 27-1](#)). The protocol engine can best be understood by defining the types of data frames to be transmitted and received by the module.

The following sequence illustrates the necessary initialization steps before the ECAN module can be used to transmit or receive a message. Steps can be added or removed depending on the requirements of the application.

1. Initial LAT and TRIS bits for RX and TX CAN.
2. Ensure that the ECAN module is in Configuration mode.
3. Select ECAN Operational mode.
4. Set up the Baud Rate registers.
5. Set up the Filter and Mask registers.
6. Set the ECAN module to normal mode or any other mode required by the application logic.

# PIC18F66K80 FAMILY

FIGURE 27-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM



## 27.2 CAN Module Registers

**Note:** Not all CAN registers are available in the Access Bank.

There are many control and data registers associated with the CAN module. For convenience, their descriptions have been grouped into the following sections:

- Control and Status Registers
- Dedicated Transmit Buffer Registers
- Dedicated Receive Buffer Registers
- Programmable TX/RX and Auto RTR Buffers
- Baud Rate Control Registers
- I/O Control Register
- Interrupt Status and Control Registers

Detailed descriptions of each register and their usage are described in the following sections.

### 27.2.1 CAN CONTROL AND STATUS REGISTERS

The registers described in this section control the overall operation of the CAN module and show its operational status.

# PIC18F66K80 FAMILY

---

## REGISTER 27-1: CANCON: CAN CONTROL REGISTER

Mode 0	R/W-1	R/W-0	R/W-0	R/S-0	R/W-0	R/W-0	R/W-0	U-0
	REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	—
Mode 1	R/W-1	R/W-0	R/W-0	R/S-0	U0	U-0	U-0	U-0
	REQOP2	REQOP1	REQOP0	ABAT	—	—	—	—
Mode 2	R/W-1	R/W-0	R/W-0	R/S-0	R-0	R-0	R-0	R-0
	REQOP2	REQOP1	REQOP0	ABAT	FP3	FP2	FP1	FP0
bit 7								bit 0

<b>Legend:</b>	S = Settable bit	
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    **REQOP<2:0>**: Request CAN Operation Mode bits

1xx = Requests Configuration mode  
011 = Requests Listen Only mode  
010 = Requests Loopback mode  
001 = Disabled/Sleep mode  
000 = Requests Normal mode

bit 4    **ABAT**: Abort All Pending Transmissions bit

1 = Abort all pending transmissions (in all transmit buffers)<sup>(1)</sup>  
0 = Transmissions proceeding as normal

bit 3-1    Mode 0:

**WIN<2:0>**: Window Address bits

These bits select which of the CAN buffers to switch into the Access Bank area. This allows access to the buffer registers from any data memory bank. After a frame has caused an interrupt, the ICODE<3:0> bits can be copied to the WIN<2:0> bits to select the correct buffer. See [Example 27-2](#) for a code example.

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

bit 0    Mode 0:

**Unimplemented**: Read as '0'

bit 4-0    Mode 1:

**Unimplemented**: Read as '0'

Mode 2:

**FP<3:0>**: FIFO Read Pointer bits

These bits point to the message buffer to be read.

0000 = Receive Message Buffer 0  
0001 = Receive Message Buffer 1  
0010 = Receive Message Buffer 2  
0011 = Receive Message Buffer 3  
0100 = Receive Message Buffer 4  
0101 = Receive Message Buffer 5  
0110 = Receive Message Buffer 6  
0111 = Receive Message Buffer 7  
1000:1111 Reserved

**Note 1:** This bit will clear when all transmissions are aborted.

# PIC18F66K80 FAMILY

## **REGISTER 27-2: CANSTAT: CAN STATUS REGISTER**

<b>Mode 0</b>	R-1	R-0	R-0	R-0	R-0	R-0	R-0	U-0
	OPMODE2 <sup>(1)</sup>	OPMODE1 <sup>(1)</sup>	OPMODE0 <sup>(1)</sup>	—	ICODE2	ICODE1	ICODE0	—
<b>Mode 1,2</b>	R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	OPMODE2 <sup>(1)</sup>	OPMODE1 <sup>(1)</sup>	OPMODE0 <sup>(1)</sup>	EICODE4	EICODE3	EICODE2	EICODE1	EICODE0
	bit 7							bit 0

**Legend:**

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

**U** = Unimplemented bit, read as '0'

‘0’ = Bit is cleared

$x$  = Bit is unknown

bit 7-5      **OPMODE<2:0>**: Operation Mode Status bits<sup>(1)</sup>

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

bit 4 Mode 0:

**Unimplemented:** Read as ‘0’

bit 3-1.4-0 Mode 0:

**ICODE<2:0>**: Interrupt Code bits

When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. By copying ICODE<3:1> to WIN<3:0> (Mode 0) or EICODE<4:0> to EWIN<4:0> (Mode 1 and 2), it is possible to select the correct buffer to map into the Access Bank area. See [Example 27-2](#) for a code example. To simplify the description, the following table lists all five bits.

	<b>Mode 0</b>	<b>Mode 1</b>	<b>Mode 2</b>
No interrupt	00000	00000	00000
CAN bus error interrupt	00010	00010	00010
TXB2 interrupt	00100	00100	00100
TXB1 interrupt	00110	00110	00110
TXB0 interrupt	01000	01000	01000
RXB1 interrupt	01010	10001	----
RXB0 interrupt	01100	10000	10000
Wake-up interrupt	00010	01110	01110
RXB0 interrupt	-----	10000	10000
RXB1 interrupt	-----	10001	10000
RX/TX B0 interrupt	-----	10010	10010 <sup>(2)</sup>
RX/TX B1 interrupt	-----	10011	10011 <sup>(2)</sup>
RX/TX B2 interrupt	-----	10100	10100 <sup>(2)</sup>
RX/TX B3 interrupt	-----	10101	10101 <sup>(2)</sup>
RX/TX B4 interrupt	-----	10110	10110 <sup>(2)</sup>
RX/TX B5 interrupt	-----	10111	10111 <sup>(2)</sup>

bit 0 Mode 0:

**Unimplemented:** Read as '0'

bit 4-0              Mode 1-2

**EICODE<4:0>**: Interrupt Code bits

See ICODE<3:1> above.

**Note 1:** To achieve maximum power saving and/or able to wake-up on CAN bus activity, switch the CAN module in Disable/Sleep mode before putting the device to Sleep.

**2:** If the buffer is configured as a receiver, the EICODE bits will contain '10000' upon interrupt.

# PIC18F66K80 FAMILY

## EXAMPLE 27-1: CHANGING TO CONFIGURATION MODE

```
; Request Configuration mode.  
MOVlw B'10000000' ; Set to Configuration Mode.  
MOVwf CANCON  
;  
; A request to switch to Configuration mode may not be immediately honored.  
; Module will wait for CAN bus to be idle before switching to Configuration Mode.  
; Request for other modes such as Loopback, Disable etc. may be honored immediately.  
; It is always good practice to wait and verify before continuing.  
ConfigWait:  
MOVf CANSTAT, W ; Read current mode state.  
ANDlw B'10000000' ; Interested in OPMODE bits only.  
TSTFSZ WREG ; Is it Configuration mode yet?  
BRA ConfigWait ; No. Continue to wait...  
;  
; Module is in Configuration mode now.  
; Modify configuration registers as required.  
; Switch back to Normal mode to be able to communicate.
```

## EXAMPLE 27-2: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS

```
; Save application required context.  
; Poll interrupt flags and determine source of interrupt  
; This was found to be CAN interrupt  
; TempCANCON and TempCANSTAT are variables defined in Access Bank low  
MOVff CANCON, TempCANCON ; Save CANCON.WIN bits  
; This is required to prevent CANCON  
; from corrupting CAN buffer access  
; in-progress while this interrupt  
; occurred  
MOVff CANSTAT, TempCANSTAT ; Save CANSTAT register  
; This is required to make sure that  
; we use same CANSTAT value rather  
; than one changed by another CAN  
; interrupt.  
MOVf TempCANSTAT, W ; Retrieve ICODE bits  
ANDlw B'00000110'  
ADDwf PCL, F ; Perform computed GOTO  
; to corresponding interrupt cause  
BRA NoInterrupt ; 000 = No interrupt  
BRA ErrorInterrupt ; 001 = Error interrupt  
BRA TXB2Interrupt ; 010 = TXB2 interrupt  
BRA TXB1Interrupt ; 011 = TXB1 interrupt  
BRA TXB0Interrupt ; 100 = TXB0 interrupt  
BRA RXB1Interrupt ; 101 = RXB1 interrupt  
BRA RXB0Interrupt ; 110 = RXB0 interrupt  
; 111 = Wake-up on interrupt  
;  
WakeupInterrupt  
BCF PIR3, WAKIF ; Clear the interrupt flag  
;  
; User code to handle wake-up procedure  
;  
;  
; Continue checking for other interrupt source or return from here  
...  
NoInterrupt  
...  
; PC should never vector here. User may  
; place a trap such as infinite loop or pin/port  
; indication to catch this error.
```

# PIC18F66K80 FAMILY

## EXAMPLE 27-2: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS (CONTINUED)

```
ErrorInterrupt
    BCF      PIR3, ERRIF           ; Clear the interrupt flag
    ...
    RETFIE

TXB2Interrupt
    BCF      PIR3, TXB2IF          ; Clear the interrupt flag
    GOTO    AccessBuffer

TXB1Interrupt
    BCF      PIR3, TXB1IF          ; Clear the interrupt flag
    GOTO    AccessBuffer

TXB0Interrupt
    BCF      PIR3, TXB0IF          ; Clear the interrupt flag
    GOTO    AccessBuffer

RXB1Interrupt
    BCF      PIR3, RXB1IF          ; Clear the interrupt flag
    GOTO    Accessbuffer

RXB0Interrupt
    BCF      PIR3, RXB0IF          ; Clear the interrupt flag
    GOTO    AccessBuffer

AccessBuffer
    ; This is either TX or RX interrupt
    ; Copy CANSTAT.ICODE bits to CANCON.WIN bits
    MOVF    TempCANCON, W          ; Clear CANCON.WIN bits before copying
                                    ; new ones.
    ANDLW   B'11110001'            ; Use previously saved CANCON value to
                                    ; make sure same value.
    MOVWF   TempCANCON
    MOVF    TempCANSTAT, W          ; Copy masked value back to TempCANCON
                                    ; Retrieve ICODE bits
    ANDLW   B'00001110'            ; Use previously saved CANSTAT value
                                    ; to make sure same value.
    IORWF   TempCANCON            ; Copy ICODE bits to WIN bits.
    MOVFF   TempCANCON, CANCON
    ; Access current buffer...
    ; User code
    ; Restore CANCON.WIN bits
    MOVF    CANCON, W              ; Preserve current non WIN bits
    ANDLW   B'11110001'
    IORWF   TempCANCON            ; Restore original WIN bits
    ; Do not need to restore CANSTAT - it is read-only register.
    ; Return from interrupt or check for another module interrupt source
```

# PIC18F66K80 FAMILY

## REGISTER 27-3: ECANCON: ENHANCED CAN CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
MDSEL1 <sup>(1)</sup>	MDSEL0 <sup>(1)</sup>	FIFOWM <sup>(2)</sup>	EWIN4	EWIN3	EWIN2	EWIN1	EWIN0
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      **MDSEL<1:0>**: Mode Select bits<sup>(1)</sup>

- 00 = Legacy mode (Mode 0, default)
- 01 = Enhanced Legacy mode (Mode 1)
- 10 = Enhanced FIFO mode (Mode 2)
- 11 = Reserved

bit 5      **FIFOWM**: FIFO High Water Mark bit<sup>(2)</sup>

- 1 = Will cause FIFO interrupt when one receive buffer remains
- 0 = Will cause FIFO interrupt when four receive buffers remain<sup>(3)</sup>

bit 4-0      **EWIN<4:0>**: Enhanced Window Address bits

These bits map the group of 16 banked CAN SFRs into Access Bank addresses, 0F60-0F6Dh. The exact group of registers to map is determined by the binary value of these bits.

#### Mode 0:

**Unimplemented:** Read as '0'

#### Mode 1, 2:

- 00000 = Acceptance Filters 0, 1, 2 and BRGCON2, 3
- 00001 = Acceptance Filters 3, 4, 5 and BRGCON1, CIOCON
- 00010 = Acceptance Filter Masks, Error and Interrupt Control
- 00011 = Transmit Buffer 0
- 00100 = Transmit Buffer 1
- 00101 = Transmit Buffer 2
- 00110 = Acceptance Filters 6, 7, 8
- 00111 = Acceptance Filters 9, 10, 11
- 01000 = Acceptance Filters 12, 13, 14
- 01001 = Acceptance Filter 15
- 01010-01110 = Reserved
- 01111 = RXINT0, RXINT1
- 10000 = Receive Buffer 0
- 10001 = Receive Buffer 1
- 10010 = TX/RX Buffer 0
- 10011 = TX/RX Buffer 1
- 10100 = TX/RX Buffer 2
- 10101 = TX/RX Buffer 3
- 10110 = TX/RX Buffer 4
- 10111 = TX/RX Buffer 5
- 11000-11111 = Reserved

**Note 1:** These bits can only be changed in Configuration mode. See [Register 27-1](#) to change to Configuration mode.

**2:** This bit is used in Mode 2 only.

**3:** If FIFO is configured to contain four or less buffers, then the FIFO interrupt will trigger.

## REGISTER 27-4: COMSTAT: COMMUNICATION STATUS REGISTER

Mode 0	R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
	RXB0OVFL	RXB1OVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
Mode 1	R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
	—	RXBnOVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
Mode 2	R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
	FIFOEMPTY	RXBnOVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
	bit 7							bit 0

### Legend:

R = Readable bit

-n = Value at POR

C = Clearable bit

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7      Mode 0:

**RXB0OVFL:** Receive Buffer 0 Overflow bit

1 = Receive Buffer 0 has overflowed

0 = Receive Buffer 0 has not overflowed

Mode 1:

**Unimplemented:** Read as '0'

Mode 2:

**FIFOEMPTY:** FIFO Not Empty bit

1 = Receive FIFO is not empty

0 = Receive FIFO is empty

bit 6      Mode 0:

**RXB1OVFL:** Receive Buffer 1 Overflow bit

1 = Receive Buffer 1 has overflowed

0 = Receive Buffer 1 has not overflowed

Mode 1, 2:

**RXBnOVFL:** Receive Buffer n Overflow bit

1 = Receive Buffer n has overflowed

0 = Receive Buffer n has not overflowed

bit 5      **TXBO:** Transmitter Bus-Off bit

1 = Transmit error counter > 255

0 = Transmit error counter  $\leq$  255

bit 4      **TXBP:** Transmitter Bus Passive bit

1 = Transmit error counter > 127

0 = Transmit error counter  $\leq$  127

bit 3      **RXBP:** Receiver Bus Passive bit

1 = Receive error counter > 127

0 = Receive error counter  $\leq$  127

bit 2      **TXWARN:** Transmitter Warning bit

1 = Transmit error counter > 95

0 = Transmit error counter  $\leq$  95

bit 1      **RXWARN:** Receiver Warning bit

1 =  $127 \geq$  Receive error counter > 95

0 = Receive error counter  $\leq$  95

bit 0      **EWARN:** Error Warning bit

This bit is a flag of the RXWARN and TXWARN bits.

1 = The RXWARN or the TXWARN bits are set

0 = Neither the RXWARN or the TXWARN bits are set

# PIC18F66K80 FAMILY

## 27.2.2 DEDICATED CAN TRANSMIT BUFFER REGISTERS

This section describes the dedicated CAN Transmit Buffer registers and their associated control registers.

### REGISTER 27-5: TXBnCON: TRANSMIT BUFFER n CONTROL REGISTERS [0 ≤ n ≤ 2]

Mode 0	U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
	TXBIF	TXABT <sup>(1)</sup>	TXLARB <sup>(1)</sup>	TXERR <sup>(1)</sup>	TXREQ <sup>(2)</sup>	—	TXPRI1 <sup>(3)</sup>	TXPRI0 <sup>(3)</sup>

Mode 1,2	R/C-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
	TXBIF	TXABT <sup>(1)</sup>	TXLARB <sup>(1)</sup>	TXERR <sup>(1)</sup>	TXREQ <sup>(2)</sup>	—	TXPRI1 <sup>(3)</sup>	TXPRI0 <sup>(3)</sup>
	bit 7							
	bit 0							

#### Legend:

C = Clearable bit

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      **TXBIF:** Transmit Buffer Interrupt Flag bit  
1 = Transmit buffer has completed transmission of a message and may be reloaded  
0 = Transmit buffer has not completed transmission of a message
- bit 6      **TXABT:** Transmission Aborted Status bit<sup>(1)</sup>  
1 = Message was aborted  
0 = Message was not aborted
- bit 5      **TXLARB:** Transmission Lost Arbitration Status bit<sup>(1)</sup>  
1 = Message lost arbitration while being sent  
0 = Message did not lose arbitration while being sent
- bit 4      **TXERR:** Transmission Error Detected Status bit<sup>(1)</sup>  
1 = A bus error occurred while the message was being sent  
0 = A bus error did not occur while the message was being sent
- bit 3      **TXREQ:** Transmit Request Status bit<sup>(2)</sup>  
1 = Requests sending a message; clears the TXABT, TXLARB and TXERR bits  
0 = Automatically cleared when the message is successfully sent
- bit 2      **Unimplemented:** Read as '0'
- bit 1-0     **TXPRI<1:0>:** Transmit Priority bits<sup>(3)</sup>  
11 = Priority Level 3 (highest priority)  
10 = Priority Level 2  
01 = Priority Level 1  
00 = Priority Level 0 (lowest priority)

- Note 1:** This bit is automatically cleared when TXREQ is set.
- 2:** While TXREQ is set, Transmit Buffer registers remain read-only. Clearing this bit in software while the bit is set will request a message abort.
- 3:** These bits define the order in which transmit buffers will be transferred. They do not alter the CAN message identifier.

# PIC18F66K80 FAMILY

## REGISTER 27-6: TXBnSIDH: TRANSMIT BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

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

**SID<10:3>**: Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)

Extended Identifier bits, EID<28:21> (if EXIDE = 1).

## REGISTER 27-7: TXBnSIDL: TRANSMIT BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]

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

**SID<2:0>**: Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)

Extended Identifier bits, EID<20:18> (if EXIDE = 1).

bit 4

**Unimplemented**: Read as '0'

bit 3

**EXIDE**: Extended Identifier Enable bit

1 = Message will transmit extended ID, SID<10:0> become EID<28:18>

0 = Message will transmit standard ID, EID<17:0> are ignored

bit 2

**Unimplemented**: Read as '0'

bit 1-0

**EID<17:16>**: Extended Identifier bits

## REGISTER 27-8: TXBnEIDH: TRANSMIT BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

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

**EID<15:8>**: Extended Identifier bits (not used when transmitting standard identifier message)

# PIC18F66K80 FAMILY

## REGISTER 27-9: TXBnEIDL: TRANSMIT BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7  | EID6  | EID5  | EID4  | EID3  | EID2  | EID1  | EID0  |
| 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 7-0      **EID<7:0>**: Extended Identifier bits (not used when transmitting standard identifier message)

## REGISTER 27-10: TXBnDm: TRANSMIT BUFFER ‘n’ DATA FIELD BYTE ‘m’ REGISTERS [0 ≤ n ≤ 2, 0 ≤ m ≤ 7]

| R/W-x   |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TXBnDm7 | TXBnDm6 | TXBnDm5 | TXBnDm4 | TXBnDm3 | TXBnDm2 | TXBnDm1 | TXBnDm0 |
| 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 7-0      **TXBnDm<7:0>**: Transmit Buffer n Data Field Byte m bits (where 0 ≤ n < 3 and 0 ≤ m < 8)

Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

# PIC18F66K80 FAMILY

## REGISTER 27-11: TXBnDLC: TRANSMIT BUFFER 'n' DATA LENGTH CODE REGISTERS [0 ≤ n ≤ 2]

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

### 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	<b>Unimplemented:</b> Read as '0'
bit 6	<b>TXRTR:</b> Transmit Remote Frame Transmission Request bit 1 = Transmitted message will have the TXRTR bit set 0 = Transmitted message will have the TXRTR bit cleared
bit 5-4	<b>Unimplemented:</b> Read as '0'
bit 3-0	<b>DLC&lt;3:0&gt;:</b> Data Length Code bits 1111 = Reserved 1110 = Reserved 1101 = Reserved 1100 = Reserved 1011 = Reserved 1010 = Reserved 1001 = Reserved 1000 = Data length = 8 bytes 0111 = Data length = 7 bytes 0110 = Data length = 6 bytes 0101 = Data length = 5 bytes 0100 = Data length = 4 bytes 0011 = Data length = 3 bytes 0010 = Data length = 2 bytes 0001 = Data length = 1 bytes 0000 = Data length = 0 bytes

## REGISTER 27-12: TXERRCNT: TRANSMIT ERROR COUNT REGISTER

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

### 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 7-0	<b>TEC&lt;7:0&gt;:</b> Transmit Error Counter bits
	This register contains a value which is derived from the rate at which errors occur. When the error count overflows, the bus-off state occurs. When the bus has 128 occurrences of 11 consecutive recessive bits, the counter value is cleared.

# PIC18F66K80 FAMILY

---

## EXAMPLE 27-3: TRANSMITTING A CAN MESSAGE USING BANKED METHOD

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.  
; To successfully transmit, CAN module must be either in Normal or Loopback mode.  
; TXB0 buffer is not in access bank. And since we want banked method, we need to make sure  
; that correct bank is selected.  
BANKSEL TXB0CON ; One BANKSEL in beginning will make sure that we are  
; in correct bank for rest of the buffer access.  
;  
; Now load transmit data into TXB0 buffer.  
MOVLW MY_DATA_BYTE1 ; Load first data byte into buffer  
MOVWF TXB0D0 ; Compiler will automatically set "BANKED" bit  
;  
; Load rest of data bytes - up to 8 bytes into TXB0 buffer.  
...  
;  
; Load message identifier  
MOVLW 60H ; Load SID2:SID0, EXIDE = 0  
MOVWF TXB0SIDL  
MOVLW 24H ; Load SID10:SID3  
MOVWF TXB0SIDH  
;  
; No need to load TXB0EIDL:TXB0EIDH, as we are transmitting Standard Identifier Message only.  
;  
; Now that all data bytes are loaded, mark it for transmission.  
MOVLW B'00001000' ; Normal priority; Request transmission  
MOVWF TXB0CON  
;  
; If required, wait for message to get transmitted  
BTFS C TXB0CON, TXREQ ; Is it transmitted?  
BRA $-2 ; No. Continue to wait...  
;  
; Message is transmitted.
```

## EXAMPLE 27-4: TRANSMITTING A CAN MESSAGE USING WIN BITS

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.  
; To successfully transmit, CAN module must be either in Normal or Loopback mode.  
; TXB0 buffer is not in access bank. Use WIN bits to map it to RXB0 area.  
MOVF    CANCON, W           ; WIN bits are in lower 4 bits only. Read CANCON  
                           ; register to preserve all other bits. If operation  
                           ; mode is already known, there is no need to preserve  
                           ; other bits.  
ANDLW   B'11110000'        ; Clear WIN bits.  
IORLW   B'00001000'        ; Select Transmit Buffer 0  
MOVWF   CANCON            ; Apply the changes.  
                           ; Now TXB0 is mapped in place of RXB0. All future access to RXB0 registers will actually  
                           ; yield TXB0 register values.  
  
; Load transmit data into TXB0 buffer.  
MOVLW   MY_DATA_BYTE1      ; Load first data byte into buffer  
MOVWF   RXB0D0              ; Access TXB0D0 via RXB0D0 address.  
                           ; Load rest of the data bytes - up to 8 bytes into "TXB0" buffer using RXB0 registers.  
...  
; Load message identifier  
MOVLW   60H                 ; Load SID2:SID0, EXIDE = 0  
MOVWF   RXB0SIDL            ;  
MOVLW   24H                 ; Load SID10:SID3  
MOVWF   RXB0SIDH            ;  
                           ; No need to load RXB0EIDL:RXB0EIDH, as we are transmitting Standard Identifier Message only.  
  
; Now that all data bytes are loaded, mark it for transmission.  
MOVLW   B'00001000'          ; Normal priority; Request transmission  
MOVWF   RXB0CON            ;  
  
; If required, wait for message to get transmitted  
BTFSR  RXB0CON, TXREQ       ; Is it transmitted?  
BRA    $-2                  ; No. Continue to wait...  
  
; Message is transmitted.  
; If required, reset the WIN bits to default state.
```

# PIC18F66K80 FAMILY

## 27.2.3 DEDICATED CAN RECEIVE BUFFER REGISTERS

This section shows the dedicated CAN Receive Buffer registers with their associated control registers.

### REGISTER 27-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER

Mode 0	R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
	RXFUL <sup>(1)</sup>	RXM1	RXM0	—	RXTRRRO	RXB0DBEN	JTOFF <sup>(2)</sup>	FILHITO

Mode 1,2	R/C-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
	RXFUL <sup>(1)</sup>	RXM1	RTRRRO	FILHITF4	FILHIT3	FILHIT2	FILHIT1	FILHITO

<b>Legend:</b>	C = Clearable 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      **RXFUL:** Receive Full Status bit<sup>(1)</sup>  
1 = Receive buffer contains a received message  
0 = Receive buffer is open to receive a new message
- bit 6,6-5    Mode 0:  
**RXM<1:0>:** Receive Buffer Mode bit 1 (combines with RXM0 to form RXM<1:0> bits, see bit 5)  
11 = Receive all messages (including those with errors); filter criteria is ignored  
10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be '1'  
01 = Receive only valid messages with standard identifier; EXIDEN in RXFnSIDL must be '0'  
00 = Receive all valid messages as per the EXIDEN bit in the RXFnSIDL register
- Mode 1, 2:  
**RXM1:** Receive Buffer Mode bit 1  
1 = Receive all messages (including those with errors); acceptance filters are ignored  
0 = Receive all valid messages as per acceptance filters
- bit 5      Mode 0:  
**RXM0:** Receive Buffer Mode bit 0 (combines with RXM1 to form RXM<1:0>bits, see bit 6)
- Mode 1, 2:  
**RTRRRO:** Remote Transmission Request bit for Received Message (read-only)  
1 = A remote transmission request is received  
0 = A remote transmission request is not received
- bit 4      Mode 0:  
**Unimplemented:** Read as '0'
- Mode 1, 2:  
**FILHIT<4:0>:** Filter Hit bit 4  
This bit combines with other bits to form filter acceptance bits<4:0>.
- bit 3      Mode 0:  
**RXTRRRO:** Remote Transmission Request bit for Received Message (read-only)  
1 = A remote transmission request is received  
0 = A remote transmission request is not received
- Mode 1, 2:  
**FILHIT<4:0>:** Filter Hit bit 3  
This bit combines with other bits to form filter acceptance bits<4:0>.

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full. After clearing the RXFUL flag, the PIR5 bit, RXB0IF, can be cleared. If RXB0IF is cleared, but RXFUL is not cleared, then RXB0IF is set again.

**2:** This bit allows the same filter jump table for both RXB0CON and RXB1CON.

## REGISTER 27-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER (CONTINUED)

bit 2	<p><u>Mode 0:</u></p> <p><b>RB0DBEN:</b> Receive Buffer 0 Double-Buffer Enable bit</p> <p>1 = Receive Buffer 0 overflow will write to Receive Buffer 1 0 = No Receive Buffer 0 overflow to Receive Buffer 1</p> <p><u>Mode 1, 2:</u></p> <p><b>FILHIT&lt;4:0&gt;:</b> Filter Hit bit 2</p> <p>This bit combines with other bits to form filter acceptance bits&lt;4:0&gt;.</p>
bit 1	<p><u>Mode 0:</u></p> <p><b>JTOFF:</b> Jump Table Offset bit (read-only copy of RXB0DBEN)<sup>(2)</sup></p> <p>1 = Allows jump table offset between 6 and 7 0 = Allows jump table offset between 1 and 0</p> <p><u>Mode 1, 2:</u></p> <p><b>FILHIT&lt;4:0&gt;:</b> Filter Hit bit 1</p> <p>This bit combines with other bits to form filter acceptance bits&lt;4:0&gt;.</p>
bit 0	<p><u>Mode 0:</u></p> <p><b>FILHITO:</b> Filter Hit bit 0</p> <p>This bit indicates which acceptance filter enabled the message reception into Receive Buffer 0.</p> <p>1 = Acceptance Filter 1 (RXF1) 0 = Acceptance Filter 0 (RXF0)</p> <p><u>Mode 1, 2:</u></p> <p><b>FILHIT&lt;4:0&gt;:</b> Filter Hit bit 0</p> <p>This bit, in combination with FILHIT&lt;4:1&gt;, indicates which acceptance filter enabled the message reception into this receive buffer.</p> <p>01111 = Acceptance Filter 15 (RXF15) 01110 = Acceptance Filter 14 (RXF14) ... 00000 = Acceptance Filter 0 (RXF0)</p>

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full. After clearing the RXFUL flag, the PIR5 bit, RXB0IF, can be cleared. If RXB0IF is cleared, but RXFUL is not cleared, then RXB0IF is set again.

**2:** This bit allows the same filter jump table for both RXB0CON and RXB1CON.

# PIC18F66K80 FAMILY

---

## REGISTER 27-14: RXB1CON: RECEIVE BUFFER 1 CONTROL REGISTER

Mode 0	R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
	RXFUL <sup>(1)</sup>	RXM1	RXM0	—	RXRTRRO	FILHIT2	FILHIT1	FILHITO

Mode 1,2	R/C-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
	RXFUL <sup>(1)</sup>	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHITO
	bit 7							bit 0

<b>Legend:</b>	C = Clearable bit
R = Readable bit	W = Writable bit
-n = Value at POR	‘1’ = Bit is set      ‘0’ = Bit is cleared      x = Bit is unknown

- bit 7      **RXFUL:** Receive Full Status bit<sup>(1)</sup>  
           1 = Receive buffer contains a received message  
           0 = Receive buffer is open to receive a new message
- bit 6-5, 6    Mode 0:  
**RXM<1:0>:** Receive Buffer Mode bit 1 (combines with RXM0 to form RXM<1:0> bits, see bit 5)  
           11 = Receive all messages (including those with errors); filter criteria is ignored  
           10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be ‘1’  
           01 = Receive only valid messages with standard identifier, EXIDEN in RXFnSIDL must be ‘0’  
           00 = Receive all valid messages as per EXIDEN bit in RXFnSIDL register
- Mode 1, 2:  
**RXM1:** Receive Buffer Mode bit  
           1 = Receive all messages (including those with errors); acceptance filters are ignored  
           0 = Receive all valid messages as per acceptance filters
- bit 5      Mode 0:  
**RXM<1:0>:** Receive Buffer Mode bit 0 (combines with RXM1 to form RXM<1:0> bits, see bit 6)
- Mode 1, 2:  
**RTRRO:** Remote Transmission Request bit for Received Message (read-only)  
           1 = A remote transmission request is received  
           0 = A remote transmission request is not received
- bit 4      Mode 0:  
**FILHIT24:** Filter Hit bit 4
- Mode 1, 2:  
**FILHIT<4:0>:** Filter Hit bit 4  
           This bit combines with other bits to form the filter acceptance bits<4:0>.
- bit 3      Mode 0:  
**RXRTRRO:** Remote Transmission Request bit for Received Message (read-only)  
           1 = A remote transmission request is received  
           0 = A remote transmission request is not received
- Mode 1, 2:  
**FILHIT<4:0>:** Filter Hit bit 3  
           This bit combines with other bits to form the filter acceptance bits<4:0>.

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full.

## REGISTER 27-14: RXB1CON: RECEIVE BUFFER 1 CONTROL REGISTER (CONTINUED)

bit 2-0    Mode 0:

**FILHIT<2:0>:** Filter Hit bits

These bits indicate which acceptance filter enabled the last message reception into Receive Buffer 1.

111 = Reserved

110 = Reserved

101 = Acceptance Filter 5 (RXF5)

100 = Acceptance Filter 4 (RXF4)

011 = Acceptance Filter 3 (RXF3)

010 = Acceptance Filter 2 (RXF2)

001 = Acceptance Filter 1 (RXF1), only possible when RXB0DBEN bit is set

000 = Acceptance Filter 0 (RXF0), only possible when RXB0DBEN bit is set

Mode 1, 2:

**FILHIT<4:0>:** Filter Hit bits<2:0>

These bits, in combination with FILHIT<4:3>, indicate which acceptance filter enabled the message reception into this receive buffer.

01111 = Acceptance Filter 15 (RXF15)

01110 = Acceptance Filter 14 (RXF14)

...

00000 = Acceptance Filter 0 (RXF0)

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full.

## REGISTER 27-15: RXBnSIDH: RECEIVE BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
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 7-0

**SID<10:3>:** Standard Identifier bits (if EXID (RXBnSIDL<3>) = 0)

Extended Identifier bits, EID<28:21> (if EXID = 1).

# PIC18F66K80 FAMILY

---

## REGISTER 27-16: RXBnSIDL: RECEIVE BUFFER 'n' STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXID	—	EID17	EID16
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      **SID<2:0>**: Standard Identifier bits (if EXID = 0)  
Extended Identifier bits, EID<20:18> (if EXID = 1).
- bit 4      **SRR**: Substitute Remote Request bit
- bit 3      **EXID**: Extended Identifier bit  
1 = Received message is an extended data frame, SID<10:0> are EID<28:18>  
0 = Received message is a standard data frame
- bit 2      **Unimplemented**: Read as '0'
- bit 1-0     **EID<17:16>**: Extended Identifier bits

## REGISTER 27-17: RXBnEIDH: RECEIVE BUFFER 'n' EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
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 7-0      **EID<15:8>**: Extended Identifier bits

## REGISTER 27-18: RXBnEIDL: RECEIVE BUFFER 'n' EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
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 7-0      **EID<7:0>**: Extended Identifier bits

# PIC18F66K80 FAMILY

## REGISTER 27-19: RXBnDLC: RECEIVE BUFFER ‘n’ DATA LENGTH CODE REGISTERS [0 ≤ n ≤ 1]

U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
—	RXRTR	RB1	R0	DLC3	DLC2	DLC1	DLC0
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	<b>Unimplemented:</b> Read as ‘0’
bit 6	<b>RXRTR:</b> Receiver Remote Transmission Request bit 1 = Remote transfer request 0 = No remote transfer request
bit 5	<b>RB1:</b> Reserved bit 1 Reserved by CAN Spec and read as ‘0’.
bit 4	<b>RB0:</b> Reserved bit 0 Reserved by CAN Spec and read as ‘0’.
bit 3-0	<b>DLC&lt;3:0&gt;:</b> Data Length Code bits 1111 = Invalid 1110 = Invalid 1101 = Invalid 1100 = Invalid 1011 = Invalid 1010 = Invalid 1001 = Invalid 1000 = Data length = 8 bytes 0111 = Data length = 7 bytes 0110 = Data length = 6 bytes 0101 = Data length = 5 bytes 0100 = Data length = 4 bytes 0011 = Data length = 3 bytes 0010 = Data length = 2 bytes 0001 = Data length = 1 byte 0000 = Data length = 0 bytes

## REGISTER 27-20: RXBnDm: RECEIVE BUFFER ‘n’ DATA FIELD BYTE ‘m’ REGISTERS [0 ≤ n ≤ 1, 0 ≤ m ≤ 7]

| R-x     |
|---------|---------|---------|---------|---------|---------|---------|---------|
| RXBnDm7 | RXBnDm6 | RXBnDm5 | RXBnDm4 | RXBnDm3 | RXBnDm2 | RXBnDm1 | RXBnDm0 |
| 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 7-0	<b>RXBnDm&lt;7:0&gt;:</b> Receive Buffer n Data Field Byte m bits (where 0 ≤ n < 1 and 0 < m < 7) Each receive buffer has an array of registers. For example, Receive Buffer 0 has 8 registers: RXB0D0 to RXB0D7.
---------	--

# PIC18F66K80 FAMILY

## REGISTER 27-21: RXERRCNT: RECEIVE ERROR COUNT REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
bit 7	bit 0						

### 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 7-0      **REC<7:0>**: Receive Error Counter bits

This register contains the receive error value as defined by the CAN specifications. When RXERRCNT > 127, the module will go into an error-passive state. RXERRCNT does not have the ability to put the module in "bus-off" state.

## EXAMPLE 27-5: READING A CAN MESSAGE

```
; Need to read a pending message from RXB0 buffer.  
; To receive any message, filter, mask and RXM1:RXM0 bits in RXB0CON registers must be  
; programmed correctly.  
;  
; Make sure that there is a message pending in RXB0.  
BTFS  RXB0CON, RXFUL ; Does RXB0 contain a message?  
BRA  NoMessage ; No. Handle this situation...  
; We have verified that a message is pending in RXB0 buffer.  
; If this buffer can receive both Standard or Extended Identifier messages,  
; identify type of message received.  
BTFS  RXB0SIDL, EXID ; Is this Extended Identifier?  
BRA  StandardMessage ; No. This is Standard Identifier message.  
; Yes. This is Extended Identifier message.  
; Read all 29-bits of Extended Identifier message.  
...  
; Now read all data bytes  
MOVFF  RXB0DO, MY_DATA_BYTE1  
...  
; Once entire message is read, mark the RXB0 that it is read and no longer FULL.  
BCF  RXB0CON, RXFUL ; This will allow CAN Module to load new messages  
; into this buffer.  
...
```

### 27.2.3.1 Programmable TX/RX and Auto-RTR Buffers

The ECAN module contains 6 message buffers that can be programmed as transmit or receive buffers. Any of these buffers can also be programmed to automatically handle RTR messages.

**Note:** These registers are not used in Mode 0.

### REGISTER 27-22: BnCON: TX/RX BUFFER 'n' CONTROL REGISTERS IN RECEIVE MODE [ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFUL <sup>(2)</sup>	RXM1	RXRTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0
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	<b>RXFUL:</b> Receive Full Status bit <sup>(2)</sup> 1 = Receive buffer contains a received message 0 = Receive buffer is open to receive a new message
bit 6	<b>RXM1:</b> Receive Buffer Mode bit 1 = Receive all messages including partial and invalid (acceptance filters are ignored) 0 = Receive all valid messages as per acceptance filters
bit 5	<b>RXRTRRO:</b> Read-Only Remote Transmission Request for Received Message bit 1 = Received message is a remote transmission request 0 = Received message is not a remote transmission request
bit 4-0	<b>FILHIT&lt;4:0&gt;:</b> Filter Hit bits These bits indicate which acceptance filter enabled the last message reception into this buffer. 01111 = Acceptance Filter 15 (RXF15) 01110 = Acceptance Filter 14 (RXF14) ... 00001 = Acceptance Filter 1 (RXF1) 00000 = Acceptance Filter 0 (RXF0)

**Note 1:** These registers are available in Mode 1 and 2 only.

**2:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full.

# PIC18F66K80 FAMILY

REGISTER 27-23: BnCON: TX/RX BUFFER ‘n’ CONTROL REGISTERS IN TRANSMIT MODE  
[ $0 \leq n \leq 5$ , TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>

R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
TXBIF <sup>(3)</sup>	TXABT <sup>(3)</sup>	TXLARB <sup>(3)</sup>	TXERR <sup>(3)</sup>	TXREQ <sup>(2,4)</sup>	RTREN	TXPRI1 <sup>(5)</sup>	TXPRI0 <sup>(5)</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      **TXBIF:** Transmit Buffer Interrupt Flag bit<sup>(3)</sup>  
1 = A message was successfully transmitted  
0 = No message was transmitted
- bit 6      **TXABT:** Transmission Aborted Status bit<sup>(3)</sup>  
1 = Message was aborted  
0 = Message was not aborted
- bit 5      **TXLARB:** Transmission Lost Arbitration Status bit<sup>(3)</sup>  
1 = Message lost arbitration while being sent  
0 = Message did not lose arbitration while being sent
- bit 4      **TXERR:** Transmission Error Detected Status bit<sup>(3)</sup>  
1 = A bus error occurred while the message was being sent  
0 = A bus error did not occur while the message was being sent
- bit 3      **TXREQ:** Transmit Request Status bit<sup>(2,4)</sup>  
1 = Requests sending a message; clears the TXABT, TXLARB and TXERR bits  
0 = Automatically cleared when the message is successfully sent
- bit 2      **RTREN:** Automatic Remote Transmission Request Enable bit  
1 = When a remote transmission request is received, TXREQ will be automatically set  
0 = When a remote transmission request is received, TXREQ will be unaffected
- bit 1-0     **TXPRI<1:0>:** Transmit Priority bits<sup>(5)</sup>  
11 = Priority Level 3 (highest priority)  
10 = Priority Level 2  
01 = Priority Level 1  
00 = Priority Level 0 (lowest priority)

- Note 1:** These registers are available in Mode 1 and 2 only.
- 2:** Clearing this bit in software while the bit is set will request a message abort.
- 3:** This bit is automatically cleared when TXREQ is set.
- 4:** While TXREQ is set or a transmission is in progress, Transmit Buffer registers remain read-only.
- 5:** These bits set the order in which the Transmit Buffer register will be transferred. They do not alter the CAN message identifier.

# PIC18F66K80 FAMILY

**REGISTER 27-24: BnSIDH: TX/RX BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS,  
HIGH BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>**

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
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 7-0      **SID<10:3>**: Standard Identifier bits (if EXIDE (BnSIDL<3>) = 0)  
Extended Identifier bits, EID<28:21> (if EXIDE = 1).

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 27-25: BnSIDH: TX/RX BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS,  
HIGH BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>**

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9  | SID8  | SID7  | SID6  | SID5  | SID4  | SID3  |
| 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 7-0      **SID<10:3>**: Standard Identifier bits (if EXIDE (BnSIDL<3>) = 0)  
Extended Identifier bits, EID<28:21> (if EXIDE = 1).

**Note 1:** These registers are available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

**REGISTER 27-26: BnSIDL: TX/RX BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS,  
LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>**

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXIDE	—	EID17	EID16
bit 7	bit 0						

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
‘1’ = Bit is set

U = Unimplemented bit, read as ‘0’  
‘0’ = Bit is cleared

x = Bit is unknown

- bit 7-5      **SID<2:0>**: Standard Identifier bits (if EXID = 0)  
Extended Identifier bits, EID<20:18> (if EXID = 1).
- bit 4      **SRR**: Substitute Remote Transmission Request bit  
This bit is always ‘1’ when EXID = 1 or equal to the value of RXRTRRO (BnCON<5>) when EXID = 0.
- bit 3      **EXIDE**: Extended Identifier Enable bit  
1 = Received message is an extended identifier frame (SID<10:0> are EID<28:18>  
0 = Received message is a standard identifier frame
- bit 2      **Unimplemented**: Read as ‘0’
- bit 1-0      **EID<17:16>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 27-27: BnSIDL: TX/RX BUFFER ‘n’ STANDARD IDENTIFIER REGISTERS,  
LOW BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>**

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

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
‘1’ = Bit is set

U = Unimplemented bit, read as ‘0’  
‘0’ = Bit is cleared

x = Bit is unknown

- bit 7-5      **SID<2:0>**: Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)  
Extended Identifier bits, EID<20:18> (if EXIDE = 1).
- bit 4      **Unimplemented**: Read as ‘0’
- bit 3      **EXIDE**: Extended Identifier Enable bit  
1 = Message will transmit extended ID, SID<10:0> bits become EID<28:18>  
0 = Received will transmit standard ID, EID<17:0> are ignored
- bit 2      **Unimplemented**: Read as ‘0’
- bit 1-0      **EID<17:16>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

**REGISTER 27-28: BnEIDH: TX/RX BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS,  
HIGH BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0]<sup>(1)</sup>**

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
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 7-0      **EID<15:8>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 27-29: BnEIDH: TX/RX BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS,  
HIGH BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1]<sup>(1)</sup>**

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9  | EID8  |
| 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 7-0      **EID<15:8>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 27-30: BnEIDL: TX/RX BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS,  
LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0]<sup>(1)</sup>**

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
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 7-0      **EID<7:0>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

**REGISTER 27-31: BnEIDL: TX/RX BUFFER ‘n’ EXTENDED IDENTIFIER REGISTERS,  
LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 1]<sup>(1)</sup>**

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7  | EID6  | EID5  | FEID4 | EID3  | EID2  | EID1  | EID0  |
| bit 7 | bit 0 |       |       |       |       |       |       |

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **EID<7:0>**: Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 27-32: BnDm: TX/RX BUFFER ‘n’ DATA FIELD BYTE ‘m’ REGISTERS IN RECEIVE MODE  
[0 ≤ n ≤ 5, 0 ≤ m ≤ 7, TXnEN (BSEL<n>) = 0]<sup>(1)</sup>**

| R-x   |
|-------|-------|-------|-------|-------|-------|-------|-------|
| BnDm7 | BnDm6 | BnDm5 | BnDm4 | BnDm3 | BnDm2 | BnDm1 | BnDm0 |
| bit 7 | bit 0 |       |       |       |       |       |       |

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **BnDm<7:0>**: Receive Buffer n Data Field Byte m bits (where 0 ≤ n < 3 and 0 < m < 8)

Each receive buffer has an array of registers. For example, Receive Buffer 0 has 7 registers: B0D0 to B0D7.

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 27-33: BnDm: TX/RX BUFFER ‘n’ DATA FIELD BYTE ‘m’ REGISTERS IN TRANSMIT MODE  
[0 ≤ n ≤ 5, 0 ≤ m ≤ 7, TXnEN (BSEL<n>) = 1]<sup>(1)</sup>**

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| BnDm7 | BnDm6 | BnDm5 | BnDm4 | BnDm3 | BnDm2 | BnDm1 | BnDm0 |
| bit 7 | bit 0 |       |       |       |       |       |       |

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **BnDm<7:0>**: Transmit Buffer n Data Field Byte m bits (where 0 ≤ n < 3 and 0 < m < 8)

Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

**Note 1:** These registers are available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

**REGISTER 27-34: BnDLC: TX/RX BUFFER 'n' DATA LENGTH CODE REGISTERS IN RECEIVE MODE  
[0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0]<sup>(1)</sup>**

U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0
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      **RXRTR:** Receiver Remote Transmission Request bit  
1 = This is a remote transmission request  
0 = This is not a remote transmission request
- bit 5      **RB1:** Reserved bit 1  
Reserved by CAN Spec and read as '0'.
- bit 4      **RB0:** Reserved bit 0  
Reserved by CAN Spec and read as '0'.
- bit 3-0     **DLC<3:0>:** Data Length Code bits  
1111 = Reserved  
1110 = Reserved  
1101 = Reserved  
1100 = Reserved  
1011 = Reserved  
1010 = Reserved  
1001 = Reserved  
1000 = Data length = 8 bytes  
0111 = Data length = 7 bytes  
0110 = Data length = 6 bytes  
0101 = Data length = 5 bytes  
0100 = Data length = 4 bytes  
0011 = Data length = 3 bytes  
0010 = Data length = 2 bytes  
0001 = Data length = 1 byte  
0000 = Data length = 0 bytes

**Note 1:** These registers are available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

**REGISTER 27-35: BnDLC: TX/RX BUFFER ‘n’ DATA LENGTH CODE REGISTERS IN TRANSMIT MODE  
[0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 1]<sup>(1)</sup>**

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

**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      **TXRTR:** Transmitter Remote Transmission Request bit

1 = Transmitted message will have the RTR bit set

0 = Transmitted message will have the RTR bit cleared

bit 5-4      **Unimplemented:** Read as ‘0’

bit 3-0      **DLC<3:0>:** Data Length Code bits

1111-1001 = Reserved

1000 = Data length = 8 bytes

0111 = Data length = 7 bytes

0110 = Data length = 6 bytes

0101 = Data length = 5 bytes

0100 = Data length = 4 bytes

0011 = Data length = 3 bytes

0010 = Data length = 2 bytes

0001 = Data length = 1 byte

0000 = Data length = 0 bytes

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 27-36: BSEL0: BUFFER SELECT REGISTER 0<sup>(1)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
B5TXEN	B4TXEN	B3TXEN	B2TXEN	B1TXEN	B0TXEN	—	—
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 7-2      **B<5:0>TXEN:** Buffer 5 to Buffer 0 Transmit Enable bits

1 = Buffer is configured in Transmit mode

0 = Buffer is configured in Receive mode

bit 1-0      **Unimplemented:** Read as ‘0’

**Note 1:** These registers are available in Mode 1 and 2 only.

### 27.2.3.2 Message Acceptance Filters and Masks

This section describes the message acceptance filters and masks for the CAN receive buffers.

#### REGISTER 27-37: RXFnSIDH: RECEIVE ACCEPTANCE FILTER ‘n’ STANDARD IDENTIFIER FILTER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SID10 | SID9  | SID8  | SID7  | SID6  | SID5  | SID4  | SID3  |
| 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 7-0      **SID<10:3>**: Standard Identifier Filter bits (if EXIDEN = 0)  
Extended Identifier Filter bits, EID<28:21> (if EXIDEN = 1).

**Note 1:** Registers, RXF6SIDH:RXF15SIDH, are available in Mode 1 and 2 only.

#### REGISTER 27-38: RXFnSIDL: RECEIVE ACCEPTANCE FILTER ‘n’ STANDARD IDENTIFIER FILTER REGISTERS, LOW BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDEN <sup>(2)</sup>	—	EID17	EID16
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      **SID<2:0>**: Standard Identifier Filter bits (if EXIDEN = 0)  
Extended Identifier Filter bits, EID<20:18> (if EXIDEN = 1).

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

bit 3      **EXIDEN:** Extended Identifier Filter Enable bit<sup>(2)</sup>  
1 = Filter will only accept extended ID messages  
0 = Filter will only accept standard ID messages

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

bit 1-0      **EID<17:16>**: Extended Identifier Filter bits

**Note 1:** Registers, RXF6SIDL:RXF15SIDL, are available in Mode 1 and 2 only.

**2:** In Mode 0, this bit must be set/cleared as required, irrespective of corresponding mask register value.

# PIC18F66K80 FAMILY

## REGISTER 27-39: RXFnEIDH: RECEIVE ACCEPTANCE FILTER ‘n’ EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

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

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **EID<15:8>**: Extended Identifier Filter bits

**Note 1:** Registers, RXF6EIDH:RXF15EIDH, are available in Mode 1 and 2 only.

## REGISTER 27-40: RXFnEIDL: RECEIVE ACCEPTANCE FILTER ‘n’ EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 15]<sup>(1)</sup>

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

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **EID<7:0>**: Extended Identifier Filter bits

**Note 1:** Registers, RXF6EIDL:RXF15EIDL, are available in Mode 1 and 2 only.

## REGISTER 27-41: RXMnSIDH: RECEIVE ACCEPTANCE MASK ‘n’ STANDARD IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

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

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **SID<10:3>**: Standard Identifier Mask bits or Extended Identifier Mask bits (EID<28:21>)

# PIC18F66K80 FAMILY

## REGISTER 27-42: RXMnSIDL: RECEIVE ACCEPTANCE MASK ‘n’ STANDARD IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	U-0	R/W-0	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDEN <sup>(1)</sup>	—	EID17	EID16
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      **SID<2:0>**: Standard Identifier Mask bits or Extended Identifier Mask bits (EID<20:18>)

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

bit 3      **Mode 0**:

**Unimplemented**: Read as ‘0’

**Mode 1, 2**:

**EXIDEN**: Extended Identifier Filter Enable Mask bit<sup>(1)</sup>

1 = Messages selected by the EXIDEN bit in RXFnSIDL will be accepted

0 = Both standard and extended identifier messages will be accepted

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

bit 1-0      **EID<17:16>**: Extended Identifier Mask bits

**Note 1:** This bit is available in Mode 1 and 2 only.

## REGISTER 27-43: RXMnEIDH: RECEIVE ACCEPTANCE MASK ‘n’ EXTENDED IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9  | EID8  |
| 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 7-0      **EID<15:8>**: Extended Identifier Mask bits

## REGISTER 27-44: RXMnEIDL: RECEIVE ACCEPTANCE MASK ‘n’ EXTENDED IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| EID7  | EID6  | EID5  | EID4  | EID3  | EID2  | EID1  | EID0  |
| 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 7-0      **EID<7:0>**: Extended Identifier Mask bits

# PIC18F66K80 FAMILY

## REGISTER 27-45: RXFCONn: RECEIVE FILTER CONTROL REGISTER ‘n’ [0 ≤ n ≤ 1]<sup>(1)</sup>

RXFCON0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXF7EN	RXF6EN	RXF5EN	RXF4EN	RXF3EN	RXF2EN	RXF1EN	RXF0EN
RXFCON1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXF15EN	RXF14EN	RXF13EN	RXF12EN	RXF11EN	RXF10EN	RXF9EN	RXF8EN
	bit 7							bit 0

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared      x = Bit is unknown

bit 7-0    **RXF<7:0>EN:** Receive Filter n Enable bits  
0 = Filter is disabled  
1 = Filter is enabled

**Note 1:** This register is available in Mode 1 and 2 only.

**Note:** Register 27-46 through Register 27-51 are writable in Configuration mode only.

## REGISTER 27-46: SDFLC: STANDARD DATA BYTES FILTER LENGTH COUNT REGISTER<sup>(1)</sup>

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLC4	FLC3	FLC2	FLC1	FLC0
bit 7							bit 0

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared      x = Bit is unknown

bit 7-5    **Unimplemented:** Read as '0'  
bit 4-0    **FLC<4:0>:** Filter Length Count bits  
Mode 0:  
Not used; forced to '00000'.  
00000-10010 = 0    18 bits are available for standard data byte filter. Actual number of bits used depends on the DLC<3:0> bits (RXBnDLC<3:0> or BnDLC<3:0> if configured as RX buffer) of the message being received.  
If DLC<3:0> = 0000    No bits will be compared with incoming data bits.  
If DLC<3:0> = 0001    Up to 8 data bits of RXFnEID<7:0>, as determined by FLC<2:0>, will be compared with the corresponding number of data bits of the incoming message.  
If DLC<3:0> = 0010    Up to 16 data bits of RXFnEID<15:0>, as determined by FLC<3:0>, will be compared with the corresponding number of data bits of the incoming message.  
If DLC<3:0> = 0011    Up to 18 data bits of RXFnEID<17:0>, as determined by FLC<4:0>, will be compared with the corresponding number of data bits of the incoming message.

**Note 1:** This register is available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

## REGISTER 27-47: RXFBCONn: RECEIVE FILTER BUFFER CONTROL REGISTER 'n'<sup>(1)</sup>

RXFBCON0	R/W-0 F1BP_3	R/W-0 F1BP_2	R/W-0 F1BP_1	R/W-0 F1BP_0	R/W-0 F0BP_3	R/W-0 F0BP_2	R/W-0 F0BP_1	R/W-0 F0BP_0
RXFBCON1	R/W-0 F3BP_3	R/W-0 F3BP_2	R/W-0 F3BP_1	R/W-1 F3BP_0	R/W-0 F2BP_3	R/W-0 F2BP_2	R/W-0 F2BP_1	R/W-1 F2BP_0
RXFBCON2	R/W-0 F5BP_3	R/W-0 F5BP_2	R/W-0 F5BP_1	R/W-1 F5BP_0	R/W-0 F4BP_3	R/W-0 F4BP_2	R/W-0 F4BP_1	R/W-1 F4BP_0
RXFBCON3	R/W-0 F7BP_3	R/W-0 F7BP_2	R/W-0 F7BP_1	R/W-0 F7BP_0	R/W-0 F6BP_3	R/W-0 F6BP_2	R/W-0 F6BP_1	R/W-0 F6BP_0
RXFBCON4	R/W-0 F9BP_3	R/W-0 F9BP_2	R/W-0 F9BP_1	R/W-0 F9BP_0	R/W-0 F8BP_3	R/W-0 F8BP_2	R/W-0 F8BP_1	R/W-0 F8BP_0
RXFBCON5	R/W-0 F11BP_3	R/W-0 F11BP_2	R/W-0 F11BP_1	R/W-0 F11BP_0	R/W-0 F10BP_3	R/W-0 F10BP_2	R/W-0 F10BP_1	R/W-0 F10BP_0
RXFBCON6	R/W-0 F13BP_3	R/W-0 F13BP_2	R/W-0 F13BP_1	R/W-0 F13BP_0	R/W-0 F12BP_3	R/W-0 F12BP_2	R/W-0 F12BP_1	R/W-0 F12BP_0
RXFBCON7	R/W-0 F15BP_3	R/W-0 F15BP_2	R/W-0 F15BP_1	R/W-0 F15BP_0	R/W-0 F14BP_3	R/W-0 F14BP_2	R/W-0 F14BP_1	R/W-0 F14BP_0
	bit 7							
	bit 0							

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

bit 7-0      **F<15:2>BP\_<3:0>**: Filter n Buffer Pointer Nibble bits

0000 = Filter n is associated with RXB0

0001 = Filter n is associated with RXB1

0010 = Filter n is associated with B0

0011 = Filter n is associated with B1

...

0111 = Filter n is associated with B5

1111-1000 = Reserved

**Note 1:** This register is available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

## REGISTER 27-48: MSEL0: MASK SELECT REGISTER 0<sup>(1)</sup>

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0
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      **FIL3\_<1:0>**: Filter 3 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4      **FIL2\_<1:0>**: Filter 2 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2      **FIL1\_<1:0>**: Filter 1 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0      **FIL0\_<1:0>**: Filter 0 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

## REGISTER 27-49: MSEL1: MASK SELECT REGISTER 1<sup>(1)</sup>

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
FIL7_1	FIL7_0	FIL6_1	FIL6_0	FIL5_1	FIL5_0	FIL4_1	FIL4_0
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      **FIL7\_<1:0>**: Filter 7 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4      **FIL6\_<1:0>**: Filter 6 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2      **FIL5\_<1:0>**: Filter 5 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0      **FIL4\_<1:0>**: Filter 4 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

## REGISTER 27-50: MSEL2: MASK SELECT REGISTER 2<sup>(1)</sup>

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FIL11_1	FIL11_0	FIL10_1	FIL10_0	FIL9_1	FIL9_0	FIL8_1	FIL8_0
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      **FIL11\_<1:0>**: Filter 11 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4      **FIL10\_<1:0>**: Filter 10 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2      **FIL9\_<1:0>**: Filter 9 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0      **FIL8\_<1:0>**: Filter 8 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

## REGISTER 27-51: MSEL3: MASK SELECT REGISTER 3<sup>(1)</sup>

| R/W-0   |
|---------|---------|---------|---------|---------|---------|---------|---------|
| FIL15_1 | FIL15_0 | FIL14_1 | FIL14_0 | FIL13_1 | FIL13_0 | FIL12_1 | FIL12_0 |
| 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      **FIL15\_<1:0>**: Filter 15 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 5-4      **FIL14\_<1:0>**: Filter 14 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 3-2      **FIL13\_<1:0>**: Filter 13 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

bit 1-0      **FIL12\_<1:0>**: Filter 12 Select bits 1 and 0

11 = No mask

10 = Filter 15

01 = Acceptance Mask 1

00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

# PIC18F66K80 FAMILY

---

## 27.2.4 CAN BAUD RATE REGISTERS

This section describes the CAN Baud Rate registers.

**Note:** These registers are writable in Configuration mode only.

### REGISTER 27-52: BRGCON1: BAUD RATE CONTROL REGISTER 1

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| SJW1  | SJW0  | BRP5  | BRP4  | BRP3  | BRP2  | BRP1  | BRP0  |
| 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      **SJW<1:0>**: Synchronized Jump Width bits

11 = Synchronization jump width time = 4 x TQ

10 = Synchronization jump width time = 3 x TQ

01 = Synchronization jump width time = 2 x TQ

00 = Synchronization jump width time = 1 x TQ

bit 5-0      **BRP<5:0>**: Baud Rate Prescaler bits

111111 = TQ = (2 x 64)/Fosc

111110 = TQ = (2 x 63)/Fosc

:

:

000001 = TQ = (2 x 2)/Fosc

000000 = TQ = (2 x 1)/Fosc

# PIC18F66K80 FAMILY

## REGISTER 27-53: BRGCON2: BAUD RATE CONTROL REGISTER 2

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

### 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           **SEG2PHTS:** Phase Segment 2 Time Select bit  
1 = Freely programmable  
0 = Maximum of PHEG1 or Information Processing Time (IPT), whichever is greater
- bit 6           **SAM:** Sample of the CAN bus Line bit  
1 = Bus line is sampled three times prior to the sample point  
0 = Bus line is sampled once at the sample point
- bit 5-3          **SEG1PH<2:0>:** Phase Segment 1 bits  
111 = Phase Segment 1 time = 8 x TQ  
110 = Phase Segment 1 time = 7 x TQ  
101 = Phase Segment 1 time = 6 x TQ  
100 = Phase Segment 1 time = 5 x TQ  
011 = Phase Segment 1 time = 4 x TQ  
010 = Phase Segment 1 time = 3 x TQ  
001 = Phase Segment 1 time = 2 x TQ  
000 = Phase Segment 1 time = 1 x TQ
- bit 2-0          **PRSEG<2:0>:** Propagation Time Select bits  
111 = Propagation time = 8 x TQ  
110 = Propagation time = 7 x TQ  
101 = Propagation time = 6 x TQ  
100 = Propagation time = 5 x TQ  
011 = Propagation time = 4 x TQ  
010 = Propagation time = 3 x TQ  
001 = Propagation time = 2 x TQ  
000 = Propagation time = 1 x TQ

# PIC18F66K80 FAMILY

## REGISTER 27-54: BRGCON3: BAUD RATE CONTROL REGISTER 3

R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKDIS	WAKFIL	—	—	—	SEG2PH2 <sup>(1)</sup>	SEG2PH1 <sup>(1)</sup>	SEG2PH0 <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           **WAKDIS:** Wake-up Disable bit

1 = Disable CAN bus activity wake-up feature

0 = Enable CAN bus activity wake-up feature

bit 6           **WAKFIL:** Selects CAN bus Line Filter for Wake-up bit

1 = Use CAN bus line filter for wake-up

0 = CAN bus line filter is not used for wake-up

bit 5-3       **Unimplemented:** Read as '0'

bit 2-0       **SEG2PH<2:0>:** Phase Segment 2 Time Select bits<sup>(1)</sup>

111 = Phase Segment 2 time = 8 x TQ

110 = Phase Segment 2 time = 7 x TQ

101 = Phase Segment 2 time = 6 x TQ

100 = Phase Segment 2 time = 5 x TQ

011 = Phase Segment 2 time = 4 x TQ

010 = Phase Segment 2 time = 3 x TQ

001 = Phase Segment 2 time = 2 x TQ

000 = Phase Segment 2 time = 1 x TQ

**Note 1:** These bits are ignored if SEG2PHTS bit (BRGCON2<7>) is '0'.

## 27.2.5 CAN MODULE I/O CONTROL REGISTER

This register controls the operation of the CAN module's I/O pins in relation to the rest of the microcontroller.

### REGISTER 27-55: CIOCON: CAN I/O CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0
TX2SRC	TX2EN	ENDRHI <sup>(1)</sup>	CANCAP	—	—	—	CLKSEL
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	<b>TX2SRC:</b> CANTX2 Pin Data Source bit 1 = CANTX2 pin will output the CAN clock 0 = CANTX2 pin will output CANTX
bit 6	<b>TX2EN:</b> CANTX Pin Enable bit 1 = CANTX2 pin will output CANTX or CAN clock as selected by the TX2SRC bit 0 = CANTX2 pin will have digital I/O function
bit 5	<b>ENDRHI:</b> Enable Drive High bit <sup>(1)</sup> 1 = CANTX pin will drive VDD when recessive 0 = CANTX pin will be tri-state when recessive
bit 4	<b>CANCAP:</b> CAN Message Receive Capture Enable bit 1 = Enable CAN capture; CAN message receive signal replaces input on RC2/CCP1 0 = Disable CAN capture; RC2/CCP1 input to CCP1 module
bit 3-1	<b>Unimplemented:</b> Read as '0'
bit 0	<b>CLKSEL:</b> CAN Clock Source Selection bit 1 = Use the oscillator as the source of the CAN system clock 0 = Use the PLL as the source of the CAN system clock

**Note 1:** Always set this bit when using a differential bus to avoid signal crosstalk in CANTX from other nearby pins.

# PIC18F66K80 FAMILY

## 27.2.6 CAN INTERRUPT REGISTERS

The registers in this section are the same as described in [Section 10.0 “Interrupts”](#). They are duplicated here for convenience.

### REGISTER 27-56: PIR5: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 5

Mode 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
	IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF <sup>(1)</sup>	TXB0IF <sup>(1)</sup>	RXB1IF	RXB0IF
Mode 1,2	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IRXIF	WAKIF	ERRIF	TXBnIF	TXB1IF <sup>(1)</sup>	TXB0IF <sup>(1)</sup>	RXBnIF	FIFOWMIF

#### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

- bit 7      **IRXIF:** CAN Bus Error Message Received Interrupt Flag bit  
1 = An invalid message has occurred on the CAN bus  
0 = No invalid message on the CAN bus
- bit 6      **WAKIF:** CAN Bus Activity Wake-up Interrupt Flag bit  
1 = Activity on the CAN bus has occurred  
0 = No activity on the CAN bus
- bit 5      **ERRIF:** CAN Module Error Interrupt Flag bit  
1 = An error has occurred in the CAN module (multiple sources; refer to [Section 27.15.6 “Error Interrupt”](#))  
0 = No CAN module errors
- bit 4      When CAN is in Mode 0:  
**TXB2IF:** CAN Transmit Buffer 2 Interrupt Flag bit  
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded  
0 = Transmit Buffer 2 has not completed transmission of a message
- When CAN is in Mode 1 or 2:  
**TXBnIF:** Any Transmit Buffer Interrupt Flag bit  
1 = One or more transmit buffers have completed transmission of a message and may be reloaded  
0 = No transmit buffer is ready for reload
- bit 3      **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit<sup>(1)</sup>  
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded  
0 = Transmit Buffer 1 has not completed transmission of a message
- bit 2      **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit<sup>(1)</sup>  
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded  
0 = Transmit Buffer 0 has not completed transmission of a message
- bit 1      When CAN is in Mode 0:  
**RXB1IF:** CAN Receive Buffer 1 Interrupt Flag bit  
1 = Receive Buffer 1 has received a new message  
0 = Receive Buffer 1 has not received a new message
- When CAN is in Mode 1 or 2:  
**RXBnIF:** Any Receive Buffer Interrupt Flag bit  
1 = One or more receive buffers has received a new message  
0 = No receive buffer has received a new message
- bit 0      When CAN is in Mode 0:  
**RXB0IF:** CAN Receive Buffer 0 Interrupt Flag bit  
1 = Receive Buffer 0 has received a new message  
0 = Receive Buffer 0 has not received a new message
- When CAN is in Mode 1:  
**Unimplemented:** Read as '0'
- When CAN is in Mode 2:  
**FIFOWMIF:** FIFO Watermark Interrupt Flag bit  
1 = FIFO high watermark is reached  
0 = FIFO high watermark is not reached

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

## REGISTER 27-57: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

Mode 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
	IRXIE	WAKIE	ERRIE	TXB2IE	TXB1IE <sup>(1)</sup>	TXB0IE <sup>(1)</sup>	RXB1IE	RXB0IE
Mode 1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IRXIE	WAKIE	ERRIE	TXBnIE	TXB1IE <sup>(1)</sup>	TXB0IE <sup>(1)</sup>	RXBnIE	FIFOWMIE
	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      **IRXIE:** CAN Bus Error Message Received Interrupt Enable bit  
               1 = Enable invalid message received interrupt  
               0 = Disable invalid message received interrupt
- bit 6      **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit  
               1 = Enable bus activity wake-up interrupt  
               0 = Disable bus activity wake-up interrupt
- bit 5      **ERRIE:** CAN bus Error Interrupt Enable bit  
               1 = Enable CAN module error interrupt  
               0 = Disable CAN module error interrupt
- bit 4      When CAN is in Mode 0:  
**TXB2IE:** CAN Transmit Buffer 2 Interrupt Enable bit  
               1 = Enable Transmit Buffer 2 interrupt  
               0 = Disable Transmit Buffer 2 interrupt  
When CAN is in Mode 1 or 2:  
**TXBnIE:** CAN Transmit Buffer Interrupts Enable bit  
               1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0  
               0 = Disable all transmit buffer interrupts
- bit 3      **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit<sup>(1)</sup>  
               1 = Enable Transmit Buffer 1 interrupt  
               0 = Disable Transmit Buffer 1 interrupt
- bit 2      **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit<sup>(1)</sup>  
               1 = Enable Transmit Buffer 0 interrupt  
               0 = Disable Transmit Buffer 0 interrupt
- bit 1      When CAN is in Mode 0:  
**RXB1IE:** CAN Receive Buffer 1 Interrupt Enable bit  
               1 = Enable Receive Buffer 1 interrupt  
               0 = Disable Receive Buffer 1 interrupt  
When CAN is in Mode 1 or 2:  
**RXBnIE:** CAN Receive Buffer Interrupts Enable bit  
               1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0  
               0 = Disable all receive buffer interrupts
- bit 0      When CAN is in Mode 0:  
**RXB0IE:** CAN Receive Buffer 0 Interrupt Enable bit  
               1 = Enable Receive Buffer 0 interrupt  
               0 = Disable Receive Buffer 0 interrupt  
When CAN is in Mode 1:  
**Unimplemented:** Read as '0'  
When CAN is in Mode 2:  
**FIFOWMIE:** FIFO Watermark Interrupt Enable bit  
               1 = Enable FIFO watermark interrupt  
               0 = Disable FIFO watermark interrupt

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

# PIC18F66K80 FAMILY

## REGISTER 27-58: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

Mode 0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	IRXIP	WAKIP	ERRIP	TXB2IP	TXB1IP <sup>(1)</sup>	TXB0IP <sup>(1)</sup>	RXB1IP	RXB0IP
Mode 1,2	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	IRXIP	WAKIP	ERRIP	TXBnIP	TXB1IP <sup>(1)</sup>	TXB0IP <sup>(1)</sup>	RXBnIP	FIFOWMIP

### 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      **IRXIP:** CAN Bus Error Message Received Interrupt Priority bit

1 = High priority

0 = Low priority

bit 6      **WAKIP:** CAN Bus Activity Wake-up Interrupt Priority bit

1 = High priority

0 = Low priority

bit 5      **ERRIP:** CAN Module Error Interrupt Priority bit

1 = High priority

0 = Low priority

bit 4      When CAN is in Mode 0:

**TXB2IP:** CAN Transmit Buffer 2 Interrupt Priority bit

1 = High priority

0 = Low priority

When CAN is in Mode 1 or 2:

**TXBnIP:** CAN Transmit Buffer Interrupt Priority bit

1 = High priority

0 = Low priority

bit 3      **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit<sup>(1)</sup>

1 = High priority

0 = Low priority

bit 2      **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit<sup>(1)</sup>

1 = High priority

0 = Low priority

bit 1      When CAN is in Mode 0:

**RXB1IP:** CAN Receive Buffer 1 Interrupt Priority bit

1 = High priority

0 = Low priority

When CAN is in Mode 1 or 2:

**RXBnIP:** CAN Receive Buffer Interrupts Priority bit

1 = High priority

0 = Low priority

bit 0      When CAN is in Mode 0:

**RXB0IP:** CAN Receive Buffer 0 Interrupt Priority bit

1 = High priority

0 = Low priority

When CAN is in Mode 1:

**Unimplemented:** Read as '0'

When CAN is in Mode 2:

**FIFOWMIP:** FIFO Watermark Interrupt Priority bit

1 = High priority

0 = Low priority

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

# PIC18F66K80 FAMILY

## REGISTER 27-59: TXBIE: TRANSMIT BUFFERS INTERRUPT ENABLE REGISTER<sup>(1)</sup>

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0	U-0
—	—	—	TXB2IE <sup>(2)</sup>	TXB1IE <sup>(2)</sup>	TXB0IE <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 4-2      **TXB2IE:TXB0IE:** Transmit Buffer 2-0 Interrupt Enable bits<sup>(2)</sup>

1 = Transmit buffer interrupt is enabled

0 = Transmit buffer interrupt is disabled

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

**Note 1:** This register is available in Mode 1 and 2 only.

2: TXBnIE in PIE5 register must be set to get an interrupt.

## REGISTER 27-60: BIE0: BUFFER INTERRUPT ENABLE REGISTER 0<sup>(1)</sup>

R/W-0	R/W-0						
B5IE <sup>(2)</sup>	B4IE <sup>(2)</sup>	B3IE <sup>(2)</sup>	B2IE <sup>(2)</sup>	B1IE <sup>(2)</sup>	B0IE <sup>(2)</sup>	RXB1IE <sup>(2)</sup>	RXB0IE <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 7-2      **B<5:0>IE:** Programmable Transmit/Receive Buffer 5-0 Interrupt Enable bits<sup>(2)</sup>

1 = Interrupt is enabled

0 = Interrupt is disabled

bit 1-0      **RXB<1:0>IE:** Dedicated Receive Buffer 1-0 Interrupt Enable bits<sup>(2)</sup>

1 = Interrupt is enabled

0 = Interrupt is disabled

**Note 1:** This register is available in Mode 1 and 2 only.

2: Either TXBnIE or RXBnIE, in the PIE5 register, must be set to get an interrupt.

# PIC18F66K80 FAMILY

---

## 27.3 CAN Modes of Operation

The PIC18F66K80 family has six main modes of operation:

- Configuration mode
- Disable/Sleep mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode

All modes, except Error Recognition, are requested by setting the REQOP bits (CANCON<7:5>). Error Recognition mode is requested through the RXM bits of the Receive Buffer register(s). Entry into a mode is Acknowledged by monitoring the OPMODE bits.

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

### 27.3.1 CONFIGURATION MODE

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

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

- Configuration Registers
- Functional Mode Selection Registers
- Bit Timing Registers
- Identifier Acceptance Filter Registers
- Identifier Acceptance Mask Registers
- Filter and Mask Control Registers
- Mask Selection Registers

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

### 27.3.2 DISABLE/SLEEP MODE

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

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

The WAKIF interrupt is the only module interrupt that is still active in the Disable/Sleep mode. If the WAKDIS is cleared and WAKIE is set, the processor will receive an interrupt whenever the module detects recessive to dominant transition. On wake-up, the module will automatically be set to the previous mode of operation. For example, if the module was switched from Normal to Disable/Sleep mode on bus activity wake-up, the module will automatically enter into Normal mode and the first message that caused the module to wake-up is lost. The module will not generate any error frame. Firmware logic must detect this condition and make sure that retransmission is requested. If the processor receives a wake-up interrupt while it is sleeping, more than one message may get lost. The actual number of messages lost would depend on the processor oscillator start-up time and incoming message bit rate.

The TXCAN pin will stay in the recessive state while the module is in Disable/Sleep mode.

### 27.3.3 NORMAL MODE

This is the standard operating mode of the PIC18F66K80 family devices. In this mode, the device actively monitors all bus messages and generates Acknowledge bits, error frames, etc. This is also the only mode in which the PIC18F66K80 family devices will transmit messages over the CAN bus.

### 27.3.4 LISTEN ONLY MODE

Listen Only mode provides a means for the PIC18F66K80 family devices to receive all messages, including messages with errors. This mode can be used for bus monitor applications or for detecting the baud rate in 'hot plugging' situations. For auto-baud detection, it is necessary that there are at least two other nodes which are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received. The Listen Only mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. In Listen Only mode, both valid and invalid messages will be received, regardless of RXMn bit settings. The filters and masks can still be used to allow only particular valid messages to be loaded into the Receive registers, or the filter masks can be set to all zeros to allow a message with any identifier to pass. All invalid messages will be received in this mode, regardless of filters and masks or RXMn Receive Buffer mode bits. The error counters are reset and deactivated in this state. The Listen Only mode is activated by setting the mode request bits in the CANCON register.

### 27.3.5 LOOPBACK MODE

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

### 27.3.6 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. In functional Mode 0, the Error Recognition mode is activated by setting the RXM<1:0> bits in the RXBnCON registers to '11'. In this mode, the data which is in the message assembly buffer until the error time, is copied in the receive buffer and can be read via the CPU interface.

### 27.4 CAN Module Functional Modes

In addition to CAN modes of operation, the ECAN module offers a total of 3 functional modes. Each of these modes are identified as Mode 0, Mode 1 and Mode 2.

#### 27.4.1 MODE 0 – LEGACY MODE

Mode 0 is designed to be fully compatible with CAN modules used in PIC18CXX8 and PIC18FXX8 devices. This is the default mode of operation on all Reset conditions. As a result, module code written for the PIC18XX8 CAN module may be used on the ECAN module without any code changes.

The following is the list of resources available in Mode 0:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Two acceptance masks, one for each receive buffer: RXM0, RXM1
- Six acceptance filters, 2 for RXB0 and 4 for RXB1: RXF0, RXF1, RXF2, RXF3, RXF4, RXF5

#### 27.4.2 MODE 1 – ENHANCED LEGACY MODE

Mode 1 is similar to Mode 0, with the exception that more resources are available in Mode 1. There are 16 acceptance filters and two acceptance mask registers. Acceptance Filter 15 can be used as either an acceptance filter or an acceptance mask register. In addition to three transmit and two receive buffers, there are six more message buffers. One or more of these additional buffers can be programmed as transmit or receive buffers. These additional buffers can also be programmed to automatically handle RTR messages.

Fourteen of sixteen acceptance filter registers can be dynamically associated to any receive buffer and acceptance mask register. One can use this capability to associate more than one filter to any one buffer.

When a receive buffer is programmed to use standard identifier messages, part of the full acceptance filter register can be used as a data byte filter. The length of the data byte filter is programmable from 0 to 18 bits. This functionality simplifies implementation of high-level protocols, such as the DeviceNet™ protocol.

The following is the list of resources available in Mode 1:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen dynamically assigned acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC

# PIC18F66K80 FAMILY

---

---

## 27.4.3 MODE 2 – ENHANCED FIFO MODE

In Mode 2, two or more receive buffers are used to form the receive FIFO (first in, first out) buffer. There is no one-to-one relationship between the receive buffer and acceptance filter registers. Any filter that is enabled and linked to any FIFO receive buffer can generate acceptance and cause FIFO to be updated.

FIFO length is user-programmable, from 2-8 buffers deep. FIFO length is determined by the very first programmable buffer that is configured as a transmit buffer. For example, if Buffer 2 (B2) is programmed as a transmit buffer, FIFO consists of RXB0, RXB1, B0 and B1, creating a FIFO length of 4. If all programmable buffers are configured as receive buffers, FIFO will have the maximum length of 8.

The following is the list of resources available in Mode 2:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX; receive buffers form FIFO: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC, useful for DeviceNet protocol

## 27.5 CAN Message Buffers

### 27.5.1 DEDICATED TRANSMIT BUFFERS

The PIC18F66K80 family devices implement three dedicated transmit buffers – TXB0, TXB1 and TXB2. Each of these buffers occupies 14 bytes of SRAM and are mapped into the SFR memory map. These are the only transmit buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each transmit buffer contains one Control register (TXBnCON), four Identifier registers (TXBnSIDL, TXBnSIDH, TXBnEIDL, TXBnEIDH), one Data Length Count register (TXBnDLC) and eight Data Byte registers (TXBnDm).

### 27.5.2 DEDICATED RECEIVE BUFFERS

The PIC18F66K80 family devices implement two dedicated receive buffers: RXB0 and RXB1. Each of these buffers occupies 14 bytes of SRAM and are mapped into SFR memory map. These are the only receive buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each receive buffer contains one Control register (RXBnCON), four Identifier registers (RXBnSIDL, RXBnSIDH, RXBnEIDL, RXBnEIDH), one Data Length Count register (RXBnDLC) and eight Data Byte registers (RXBnDm).

There is also a separate Message Assembly Buffer (MAB) which acts as an additional receive buffer. MAB is always committed to receiving the next message from the bus and is not directly accessible to user firmware. The MAB assembles all incoming messages one by one. A message is transferred to appropriate receive buffers only if the corresponding acceptance filter criteria is met.

### 27.5.3 PROGRAMMABLE TRANSMIT/RECEIVE BUFFERS

The ECAN module implements six new buffers: B0-B5. These buffers are individually programmable as either transmit or receive buffers. These buffers are available only in Mode 1 and 2. As with dedicated transmit and receive buffers, each of these programmable buffers occupies 14 bytes of SRAM and are mapped into SFR memory map.

Each buffer contains one Control register (BnCON), four Identifier registers (BnSIDL, BnSIDH, BnEIDL, BnEIDH), one Data Length Count register (BnDLC) and eight Data Byte registers (BnDm). Each of these registers contains two sets of control bits. Depending on whether the buffer is configured as transmit or receive, one would use the corresponding control bit set. By default, all buffers are configured as receive buffers. Each buffer can be individually configured as a transmit or receive buffer by setting the corresponding TXENn bit in the BSEL0 register.

When configured as transmit buffers, user firmware may access transmit buffers in any order similar to accessing dedicated transmit buffers. In receive configuration with Mode 1 enabled, user firmware may also access receive buffers in any order required. But in Mode 2, all receive buffers are combined to form a single FIFO. Actual FIFO length is programmable by user firmware. Access to FIFO must be done through the FIFO Pointer bits (FP<4:0>) in the CANCON register. It must be noted that there is no hardware protection against out of order FIFO reads.

## 27.5.4 PROGRAMMABLE AUTO-RTR BUFFERS

In Mode 1 and 2, any of six programmable transmit/receive buffers may be programmed to automatically respond to predefined RTR messages without user firmware intervention. Automatic RTR handling is enabled by setting the TX2EN bit in the BSEL0 register and the RTREN bit in the BnCON register. After this setup, when an RTR request is received, the TXREQ bit is automatically set and the current buffer content is automatically queued for transmission as a RTR response. As with all transmit buffers, once the TXREQ bit is set, buffer registers become read-only and any writes to them will be ignored.

The following outlines the steps required to automatically handle RTR messages:

1. Set buffer to Transmit mode by setting the TXnEN bit to '1' in the BSEL0 register.
2. At least one acceptance filter must be associated with this buffer and preloaded with the expected RTR identifier.
3. Bit, RTREN in the BnCON register, must be set to '1'.
4. Buffer must be preloaded with the data to be sent as a RTR response.

Normally, user firmware will keep buffer data registers up to date. If firmware attempts to update the buffer while an automatic RTR response is in the process of transmission, all writes to buffers are ignored.

## 27.6 CAN Message Transmission

### 27.6.1 INITIATING TRANSMISSION

For the MCU to have write access to the message buffer, the TXREQ bit must be clear, indicating that the message buffer is clear of any pending message to be transmitted. At a minimum, the SIDH, SIDL and DLC registers must be loaded. If data bytes are present in the message, the Data registers must also be loaded. If the message is to use extended identifiers, the EIDH:EIDL registers must also be loaded and the EXIDE bit set.

To initiate message transmission, the TXREQ bit must be set for each buffer to be transmitted. When TXREQ is set, the TXABT, TXLARB and TXERR bits will be cleared. To successfully complete the transmission, there must be at least one node with matching baud rate on the network.

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

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

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

### 27.6.2 ABORTING TRANSMISSION

The MCU can request to abort a message by clearing the TXREQ bit associated with the corresponding message buffer (TXBnCON<3> or BnCON<3>). Setting the ABAT bit (CANCON<4>) will request an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets the TXABT bit for the corresponding buffer (TXBnCON<6> or BnCON<6>). If the message has started to transmit, it will attempt to transmit the current message fully. If the current message is transmitted fully and is not lost to arbitration or an error, the TXABT bit will not be set because the message was transmitted successfully. Likewise, if a message is being transmitted during an abort request and the message is lost to arbitration or an error, the message will not be retransmitted and the TXABT bit will be set, indicating that the message was successfully aborted.

Once an abort is requested by setting the ABAT or TXABT bits, it cannot be cleared to cancel the abort request. Only CAN module hardware or a POR condition can clear it.

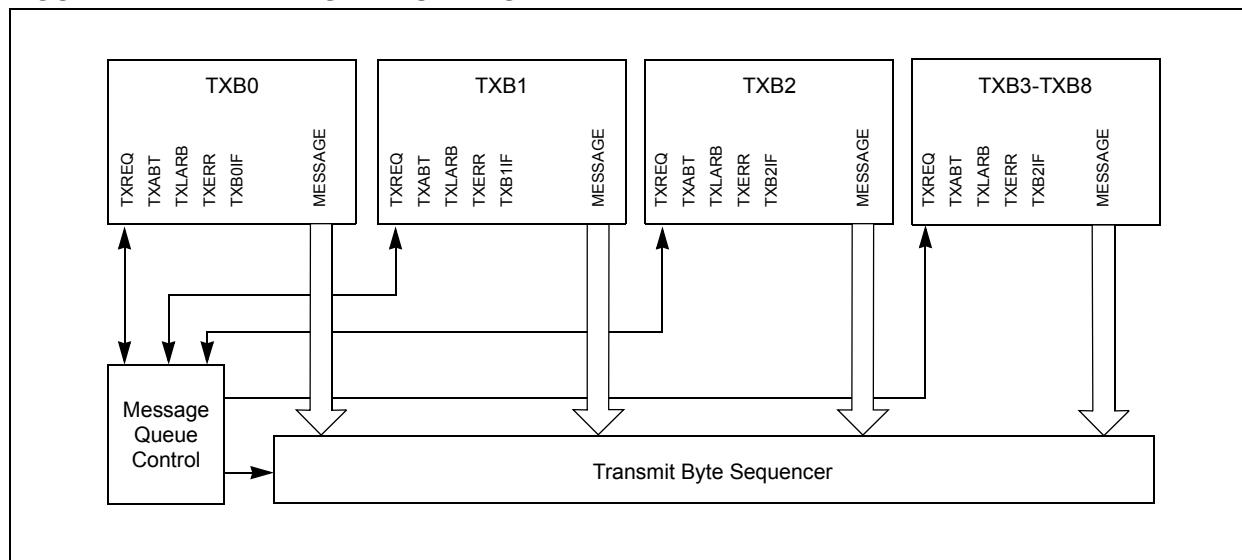
# PIC18F66K80 FAMILY

## 27.6.3 TRANSMIT PRIORITY

Transmit priority is a prioritization within the PIC18F66K80 family devices of the pending transmittable messages. This is independent from, and not related to, any prioritization implicit in the message arbitration scheme built into the CAN protocol. Prior to sending the Start-of-Frame (SOF), the priority of all buffers that are queued for transmission is compared. The

transmit buffer with the highest priority will be sent first. If two buffers have the same priority setting, the buffer with the highest buffer number will be sent first. There are four levels of transmit priority. If the TXP bits for a particular message buffer are set to '11', that buffer has the highest possible priority. If the TXP bits for a particular message buffer are set to '00', that buffer has the lowest possible priority.

**FIGURE 27-2: TRANSMIT BUFFERS**



## 27.7 Message Reception

### 27.7.1 RECEIVING A MESSAGE

Of all receive buffers, the MAB is always committed to receiving the next message from the bus. The MCU can access one buffer while the other buffer is available for message reception or holding a previously received message.

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

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

Once a message is loaded into any matching buffer, user firmware may determine exactly what filter caused this reception by checking the filter hit bits in the RXBnCON or BnCON registers. In Mode 0, FILHIT<2:0> of RXBnCON serve as filter hit bits. In Mode 1 and 2, FILHIT<4:0> bits of BnCON serve as filter hit bits. The same registers also indicate whether the current message is an RTR frame or not. A received message is considered a standard identifier message if the EXID/EXIDE bit in the RXBnSIDL or the BnSIDL register is cleared. Conversely, a set EXID bit indicates an extended identifier message. If the received message is a standard identifier message, user firmware needs to read the SIDL and SIDH registers. In the case of an extended identifier message, firmware should read the SIDL, SIDH, EIDL and EIDH registers. If the RXBnDLC or BnDLC register contain non-zero data count, user firmware should also read the corresponding number of data bytes by accessing the RXBnDm or the BnDm registers. When a received message is an RTR, and if the current buffer is not configured for automatic RTR handling, user firmware must take appropriate action and respond manually.

Each receive buffer contains RXM bits to set special Receive modes. In Mode 0, RXM<1:0> bits in RXBnCON define a total of four Receive modes. In Mode 1 and 2, RXM1 bit, in combination with the EXID mask and filter bit, define the same four receive modes.

Normally, these bits are set to '00' to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the acceptance filter register. In Mode 0, if the RXM bits are set to '01' or '10', the receiver will accept only messages with standard or extended identifiers, respectively. If an acceptance filter has the EXIDE bit set, such that it does not correspond with the RXM mode, that acceptance filter is rendered useless. In Mode 1 and 2, setting EXID in the SIDL Mask register will ensure that only standard or extended identifiers are received. These two modes of RXM bits can be used in systems where it is known that only standard or extended messages will be on the bus. If the RXM bits are set to '11' (RXM1 = 1 in Mode 1 and 2), the buffer will receive all messages regardless of the values of the acceptance filters. Also, if a message has an error before the end of frame, that portion of the message assembled in the MAB before the error frame will be loaded into the buffer. This mode may serve as a valuable debugging tool for a given CAN network. It should not be used in an actual system environment as the actual system will always have some bus errors and all nodes on the bus are expected to ignore them.

In Mode 1 and 2, when a programmable buffer is configured as a transmit buffer and one or more acceptance filters are associated with it, all incoming messages matching this acceptance filter criteria will be discarded. To avoid this scenario, user firmware must make sure that there are no acceptance filters associated with a buffer configured as a transmit buffer.

### 27.7.2 RECEIVE PRIORITY

When in Mode 0, RXB0 is the higher priority buffer and has two message acceptance filters associated with it. RXB1 is the lower priority buffer and has four acceptance filters associated with it. The lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer. Additionally, the RXB0CON register can be configured such that if RXB0 contains a valid message and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1 regardless of the acceptance criteria of RXB1. There are also two programmable acceptance filter masks available, one for each receive buffer (see [Section 27.5 "CAN Message Buffers"](#)).

In Mode 1 and 2, there are a total of 16 acceptance filters available and each can be dynamically assigned to any of the receive buffers. A buffer with a lower number has higher priority. Given this, if an incoming message matches with two or more receive buffer acceptance criteria, the buffer with the lower number will be loaded with that message.

# PIC18F66K80 FAMILY

---

## 27.7.3 ENHANCED FIFO MODE

When configured for Mode 2, two of the dedicated receive buffers in combination with one or more programmable transmit/receive buffers, are used to create a maximum of an 8 buffers deep FIFO buffer. In this mode, there is no direct correlation between filters and receive buffer registers. Any filter that has been enabled can generate an acceptance. When a message has been accepted, it is stored in the next available receive buffer register and an internal Write Pointer is incremented. The FIFO can be a maximum of 8 buffers deep. The entire FIFO must consist of contiguous receive buffers. The FIFO head begins at RXB0 buffer and its tail spans toward B5. The maximum length of the FIFO is limited by the presence or absence of the first transmit buffer starting from B0. If a buffer is configured as a transmit buffer, the FIFO length is reduced accordingly. For instance, if B3 is configured as a transmit buffer, the actual FIFO will consist of RXB0, RXB1, B0, B1 and B2, a total of 5 buffers. If B0 is configured as a transmit buffer, the FIFO length will be 2. If none of the programmable buffers are configured as a transmit buffer, the FIFO will be 8 buffers deep. A system that requires more transmit buffers should try to locate transmit buffers at the very end of B0-B5 buffers to maximize available FIFO length.

When a message is received in FIFO mode, the interrupt flag code bits (EICODE<4:0>) in the CANSTAT register will have a value of '10000', indicating the FIFO has received a message. FIFO Pointer bits, FP<3:0> in the CANCON register, point to the buffer that contains data not yet read. The FIFO Pointer bits, in this sense, serve as the FIFO Read Pointer. The user should use the FP bits and read corresponding buffer data. When receive data is no longer needed, the RXFUL bit in the current buffer must be cleared, causing FP<3:0> to be updated by the module.

To determine whether FIFO is empty or not, the user may use the FP<3:0> bits to access the RXFUL bit in the current buffer. If RXFUL is cleared, the FIFO is considered to be empty. If it is set, the FIFO may contain one or more messages. In Mode 2, the module also provides a bit called FIFO High Water Mark (FIFOWM) in the ECANCON register. This bit can be used to cause an interrupt whenever the FIFO contains only one or four empty buffers. The FIFO high water mark interrupt can serve as an early warning to a full FIFO condition.

## 27.7.4 TIME-STAMPING

The CAN module can be programmed to generate a time-stamp for every message that is received. When enabled, the module generates a capture signal for CCP1, which in turn captures the value of either Timer1 or Timer3. This value can be used as the message time-stamp.

To use the time-stamp capability, the CANCAP bit (CIOCON<4>) must be set. This replaces the capture input for CCP1 with the signal generated from the CAN module. In addition, CCP1CON<3:0> must be set to '0011' to enable the CCP Special Event Trigger for CAN events.

## 27.8 Message Acceptance Filters and Masks

The message acceptance filters and masks are used to determine if a message in the Message Assembly Buffer should be loaded into any of the receive buffers. Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifier are examined with the filters. A truth table is shown below in [Table 27-1](#) that indicates how each bit in the identifier is compared to the masks and filters to determine if a message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, then that bit will automatically be accepted regardless of the filter bit.

**TABLE 27-1: FILTER/MASK TRUTH TABLE**

Mask bit n	Filter bit n	Message Identifier bit n001	Accept or Reject bit n
0	x	x	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

**Legend:** x = don't care

In Mode 0, acceptance filters, RXF0 and RXF1, and filter mask, RXM0, are associated with RXB0. Filters, RXF2, RXF3, RXF4 and RXF5, and mask, RXM1, are associated with RXB1.

In Mode 1 and 2, there are an additional 10 acceptance filters, RXF6-RXF15, creating a total of 16 available filters. RXF15 can be used either as an acceptance filter or acceptance mask register. Each of these acceptance filters can be individually enabled or disabled by setting or clearing the RXFENN bit in the RXFCONn register. Any of these 16 acceptance filters can be dynamically associated with any of the receive buffers. Actual association is made by setting the appropriate bits in the RXFBCONn register. Each RXFBCONn register contains a nibble for each filter. This nibble can be used to associate a specific filter to any of available receive buffers. User firmware may associate more than one filter to any one specific receive buffer.

In addition to dynamic filter to buffer association, in Mode 1 and 2, each filter can also be dynamically associated to available Acceptance Mask registers. The FILLn\_m bits in the MSELn register can be used to link a specific acceptance filter to an acceptance mask register. As with filter to buffer association, one can also associate more than one mask to a specific acceptance filter.

When a filter matches and a message is loaded into the receive buffer, the filter number that enabled the message reception is loaded into the FILHIT bit(s). In Mode 0 for RXB1, the RXB1CON register contains the FILHIT<2:0> bits. They are coded as follows:

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

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

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

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

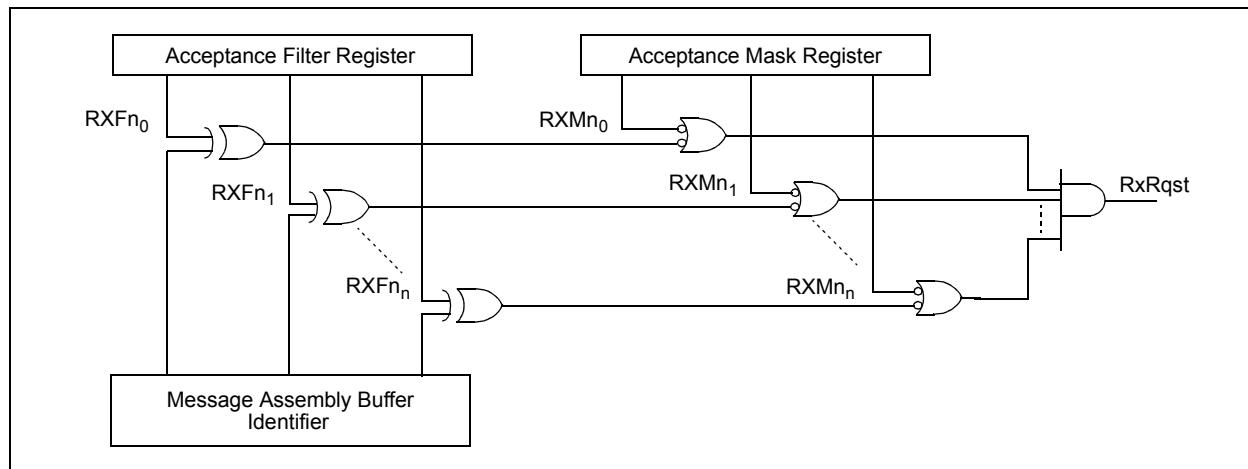
If the RXB0DBEN bit is clear, there are six codes corresponding to the six filters. If the RXB0DBEN bit is set, there are six codes corresponding to the six filters, plus two additional codes corresponding to RXF0 and RXF1 filters, that rollover into RXB1.

In Mode 1 and 2, each buffer control register contains 5 bits of filter hit bits (FILHIT<4:0>). A binary value of '0' indicates a hit from RXF0 and 15 indicates RXF15.

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

The mask and filter registers can only be modified when the PIC18F66K80 family devices are in Configuration mode.

**FIGURE 27-3: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION**



# PIC18F66K80 FAMILY

## 27.9 Baud Rate Setting

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

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

The bit timing of the PIC18F66K80 family is implemented using a DPLL that is configured to synchronize to the incoming data and provides the nominal timing for the transmitted data. The DPLL breaks each bit time into multiple segments made up of minimal periods of time called the *Time Quanta* ( $T_Q$ ).

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

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

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

The "Nominal Bit Time" is defined as:

### EQUATION 27-1: NOMINAL BIT TIME

$$T_{BIT} = 1/\text{Nominal Bit Rate}$$

The Nominal Bit Time can be thought of as being divided into separate, non-overlapping time segments. These segments (Figure 27-4) include:

- Synchronization Segment (Sync\_Seg)
- Propagation Time Segment (Prop\_Seg)
- Phase Buffer Segment 1 (Phase\_Seg1)
- Phase Buffer Segment 2 (Phase\_Seg2)

The time segments (and thus, the Nominal Bit Time) are, in turn, made up of integer units of time called Time Quanta or  $T_Q$  (see Figure 27-4). By definition, the Nominal Bit Time is programmable from a minimum of 8  $T_Q$  to a maximum of 25  $T_Q$ . Also by definition, the minimum Nominal Bit Time is 1  $\mu s$ , corresponding to a maximum 1 Mb/s rate. The actual duration is given by the following relationship:

### EQUATION 27-2: NOMINAL BIT TIME DURATION

$$\text{Nominal Bit Time} = T_Q * (\text{Sync}_\text{Seg} + \text{Prop}_\text{Seg} + \text{Phase}_\text{Seg1} + \text{Phase}_\text{Seg2})$$

The Time Quantum is a fixed unit derived from the oscillator period. It is also defined by the programmable baud rate prescaler, with integer values from 1 to 64, in addition to a fixed divide-by-two for clock generation. Mathematically, this is:

### EQUATION 27-3: TIME QUANTUM

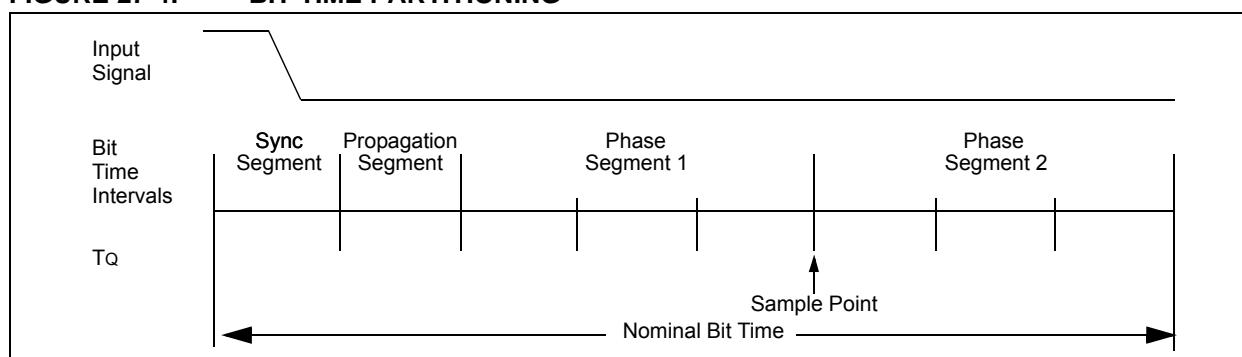
$$T_Q (\mu s) = (2 * (\text{BRP} + 1)) / \text{FOSC (MHz)}$$

or

$$T_Q (\mu s) = (2 * (\text{BRP} + 1)) * \text{TOSC (\mu s)}$$

where Fosc is the clock frequency, Tosc is the corresponding oscillator period and BRP is an integer (0 through 63) represented by the binary values of BRGCON1<5:0>. The equation above refers to the effective clock frequency used by the microcontroller. If, for example, a 10 MHz crystal in HS mode is used, then Fosc = 10 MHz and Tosc = 100 ns. If the same 10 MHz crystal is used in HS-PLL mode, then the effective frequency is Fosc = 40 MHz and Tosc = 25 ns.

FIGURE 27-4: BIT TIME PARTITIONING



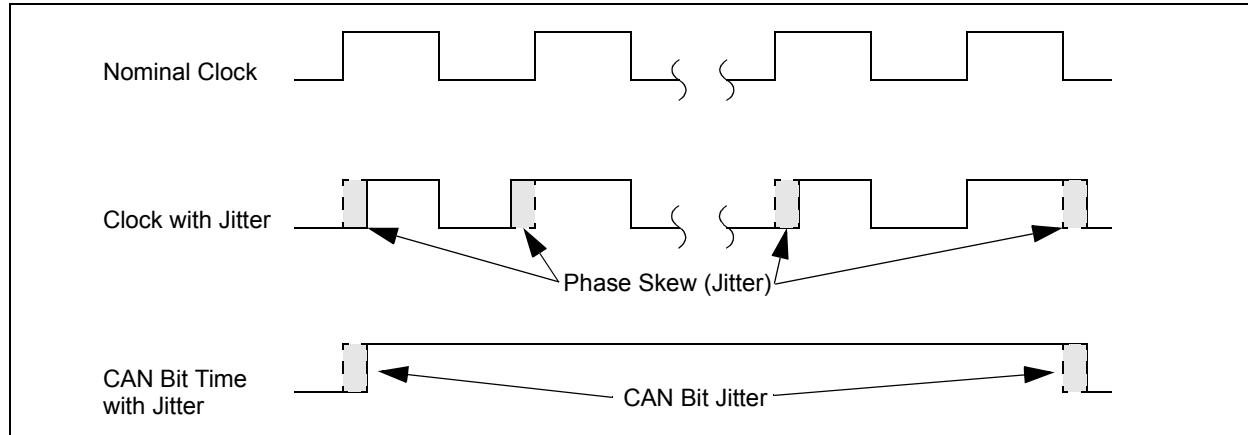
## 27.9.1 EXTERNAL CLOCK, INTERNAL CLOCK AND MEASURABLE JITTER IN HS-PLL BASED OSCILLATORS

The microcontroller clock frequency generated from a PLL circuit is subject to a jitter, also defined as Phase Jitter or Phase Skew. For its PIC18 Enhanced microcontrollers, Microchip specifies phase jitter ( $P_{\text{jitter}}$ ) as being 2% (Gaussian distribution, within 3 standard deviations, see Parameter F13 in Table 31-7) and Total Jitter ( $T_{\text{jitter}}$ ) as being  $2 * P_{\text{jitter}}$ .

The CAN protocol uses a bit-stuffing technique that inserts a bit of a given polarity following five bits with the opposite polarity. This gives a total of 10 bits transmitted without resynchronization (compensation for jitter or phase error).

Given the random nature of the added jitter error, it can be shown that the total error caused by the jitter tends to cancel itself over time. For a period of 10 bits, it is necessary to add only two jitter intervals to correct for jitter induced error: one interval in the beginning of the 10-bit period and another at the end. The overall effect is shown in Figure 27-5.

**FIGURE 27-5: EFFECTS OF PHASE JITTER ON THE MICROCONTROLLER CLOCK AND CAN BIT TIME**



Once these considerations are taken into account, it is possible to show that the relation between the jitter and the total frequency error can be defined as:

**EQUATION 27-4: JITTER AND TOTAL FREQUENCY ERROR**

$$\Delta f = \frac{T_{\text{jitter}}}{10 \times \text{NBT}} = \frac{2 \times P_{\text{jitter}}}{10 \times \text{NBT}}$$

where jitter is expressed in terms of time and NBT is the Nominal Bit Time.

For example, assume a CAN bit rate of 125 Kb/s, which gives an NBT of 8  $\mu$ s. For a 16 MHz clock generated from a 4x PLL, the jitter at this clock frequency is:

**EQUATION 27-5: 16 MHz CLOCK FROM 4x PLL JITTER:**

$$2\% \times \frac{1}{16 \text{ MHz}} = \frac{0.02}{16 \times 10^6} = 1.25 \text{ ns}$$

and resultant frequency error is:

**EQUATION 27-6: RESULTANT FREQUENCY ERROR:**

$$\frac{2 \times (1.25 \times 10^{-9})}{10 \times (8 \times 10^{-6})} = 3.125 \times 10^{-5} = 0.0031\%$$

# PIC18F66K80 FAMILY

---

Table 27-2 shows the relation between the clock generated by the PLL and the frequency error from jitter (measured jitter-induced error of 2%, Gaussian distribution, within 3 standard deviations), as a percentage of the nominal clock frequency.

This is clearly smaller than the expected drift of a crystal oscillator, typically specified at 100 ppm or 0.01%. If we add jitter to oscillator drift, we have a total frequency drift of 0.0132%. The total oscillator frequency errors for common clock frequencies and bit rates, including both drift and jitter, are shown in Table 27-3.

TABLE 27-2: FREQUENCY ERROR FROM JITTER AT VARIOUS PLL GENERATED CLOCK SPEEDS

PLL Output	$P_{\text{jitter}}$	$T_{\text{jitter}}$	Frequency Error at Various Nominal Bit Times (Bit Rates)			
			8 $\mu\text{s}$ (125 Kb/s)	4 $\mu\text{s}$ (250 Kb/s)	2 $\mu\text{s}$ (500 Kb/s)	1 $\mu\text{s}$ (1 Mb/s)
40 MHz	0.5 ns	1 ns	0.00125%	0.00250%	0.005%	0.01%
24 MHz	0.83 ns	1.67 ns	0.00209%	0.00418%	0.008%	0.017%
16 MHz	1.25 ns	2.5 ns	0.00313%	0.00625%	0.013%	0.025%

TABLE 27-3: TOTAL FREQUENCY ERROR AT VARIOUS PLL GENERATED CLOCK SPEEDS  
(100 PPM OSCILLATOR DRIFT, INCLUDING ERROR FROM JITTER)

Nominal PLL Output	Frequency Error at Various Nominal Bit Times (Bit Rates)			
	8 $\mu\text{s}$ (125 Kb/s)	4 $\mu\text{s}$ (250 Kb/s)	2 $\mu\text{s}$ (500 Kb/s)	1 $\mu\text{s}$ (1 Mb/s)
40 MHz	0.01125%	0.01250%	0.015%	0.02%
24 MHz	0.01209%	0.01418%	0.018%	0.027%
16 MHz	0.01313%	0.01625%	0.023%	0.035%

## 27.9.2 TIME QUANTA

As already mentioned, the Time Quanta is a fixed unit derived from the oscillator period and baud rate prescaler. Its relationship to TBIT and the Nominal Bit Rate is shown in [Example 27-6](#).

### EXAMPLE 27-6: CALCULATING TQ, NOMINAL BIT RATE AND NOMINAL BIT TIME

$$TQ (\mu s) = (2 * (BRP + 1)) / FOSC (MHz)$$

$$TBIT (\mu s) = TQ (\mu s) * \text{number of } TQ \text{ per bit interval}$$

$$\text{Nominal Bit Rate (bits/s)} = 1/TBIT$$

This frequency (Fosc) refers to the effective frequency used. If, for example, a 10 MHz external signal is used along with a PLL, then the effective frequency will be  $4 \times 10$  MHz which equals 40 MHz.

#### CASE 1:

For  $FOSC = 16$  MHz,  $BRP<5:0> = 00h$  and Nominal Bit Time = 8 TQ:

$$TQ = (2 * 1)/16 = 0.125 \mu s (125 ns)$$

$$TBIT = 8 * 0.125 = 1 \mu s (10^{-6}s)$$

$$\text{Nominal Bit Rate} = 1/10^{-6} = 10^6 \text{ bits/s (1 Mb/s)}$$

#### CASE 2:

For  $FOSC = 20$  MHz,  $BRP<5:0> = 01h$  and Nominal Bit Time = 8 TQ:

$$TQ = (2 * 2)/20 = 0.2 \mu s (200 ns)$$

$$TBIT = 8 * 0.2 = 1.6 \mu s (1.6 * 10^{-6}s)$$

$$\text{Nominal Bit Rate} = 1/1.6 * 10^{-6}s = 625,000 \text{ bits/s (625 Kb/s)}$$

#### CASE 3:

For  $FOSC = 25$  MHz,  $BRP<5:0> = 3Fh$  and Nominal Bit Time = 25 TQ:

$$TQ = (2 * 64)/25 = 5.12 \mu s$$

$$TBIT = 25 * 5.12 = 128 \mu s (1.28 * 10^{-4}s)$$

$$\text{Nominal Bit Rate} = 1/1.28 * 10^{-4} = 7813 \text{ bits/s (7.8 Kb/s)}$$

The frequencies of the oscillators in the different nodes must be coordinated in order to provide a system wide specified Nominal Bit Time. This means that all oscillators must have a Tosc that is an integral divisor of TQ. It should also be noted that although the number of TQ is programmable from 4 to 25, the usable minimum is 8 TQ. There is no assurance that a bit time of less than 8 TQ in length will operate correctly.

## 27.9.3 SYNCHRONIZATION SEGMENT

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

## 27.9.4 PROPAGATION SEGMENT

This part of the bit time is used to compensate for physical delay times within the network. These delay times consist of the signal propagation time on the bus line and the internal delay time of the nodes. The length of the propagation segment can be programmed from 1 TQ to 8 TQ by setting the PRSEG<2:0> bits.

## 27.9.5 PHASE BUFFER SEGMENTS

The phase buffer segments are used to optimally locate the sampling point of the received bit within the Nominal Bit Time. The sampling point occurs between Phase Segment 1 and Phase Segment 2. These segments can be lengthened or shortened by the resynchronization process. The end of Phase Segment 1 determines the sampling point within a bit time. Phase Segment 1 is programmable from 1 TQ to 8 TQ in duration. Phase Segment 2 provides a delay before the next transmitted data transition and is also programmable from 1 TQ to 8 TQ in duration. However, due to IPT requirements, the actual minimum length of Phase Segment 2 is 2 TQ, or it may be defined to be equal to the greater of Phase Segment 1 or the Information Processing Time (IPT). The sampling point should be as late as possible or approximately 80% of the bit time.

## 27.9.6 SAMPLE POINT

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

## 27.9.7 INFORMATION PROCESSING TIME

The Information Processing Time (IPT) is the time segment starting at the sample point that is reserved for calculation of the subsequent bit level. The CAN specification defines this time to be less than or equal to 2 TQ. The PIC18F66K80 family devices define this time to be 2 TQ. Thus, Phase Segment 2 must be at least 2 TQ long.

# PIC18F66K80 FAMILY

## 27.10 Synchronization

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

### 27.10.1 HARD SYNCHRONIZATION

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

### 27.10.2 RESYNCHRONIZATION

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to Phase Segment 1 (see Figure 27-6) or subtracted from Phase Segment 2 (see Figure 27-7). The SJW is programmable between 1 TQ and 4 TQ.

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

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

- $e = 0$  if the edge lies within Sync\_Seg.
- $e > 0$  if the edge lies before the sample point.
- $e < 0$  if the edge lies after the sample point of the previous bit.

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

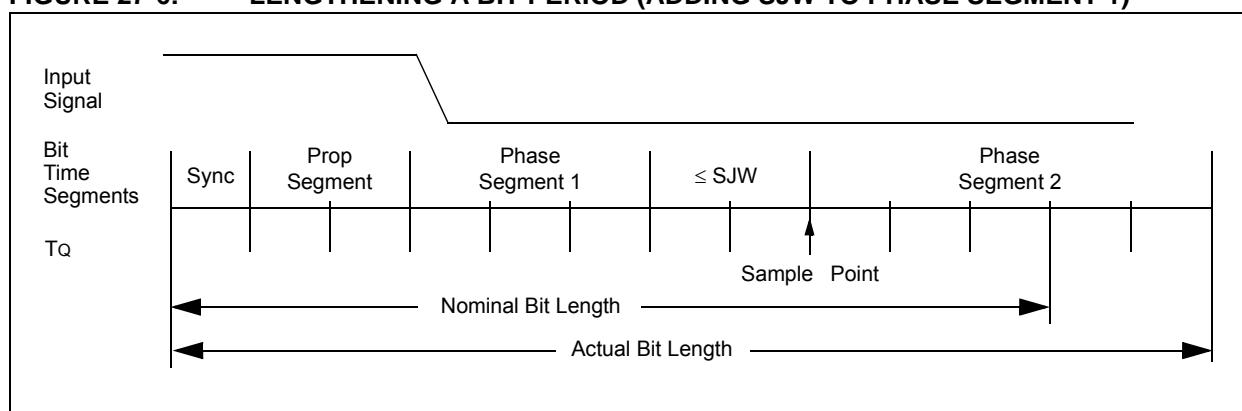
If the magnitude of the phase error is larger than the Synchronization Jump Width and if the phase error is positive, then Phase Segment 1 is lengthened by an amount equal to the Synchronization Jump Width.

If the magnitude of the phase error is larger than the resynchronization jump width and if the phase error is negative, then Phase Segment 2 is shortened by an amount equal to the Synchronization Jump Width.

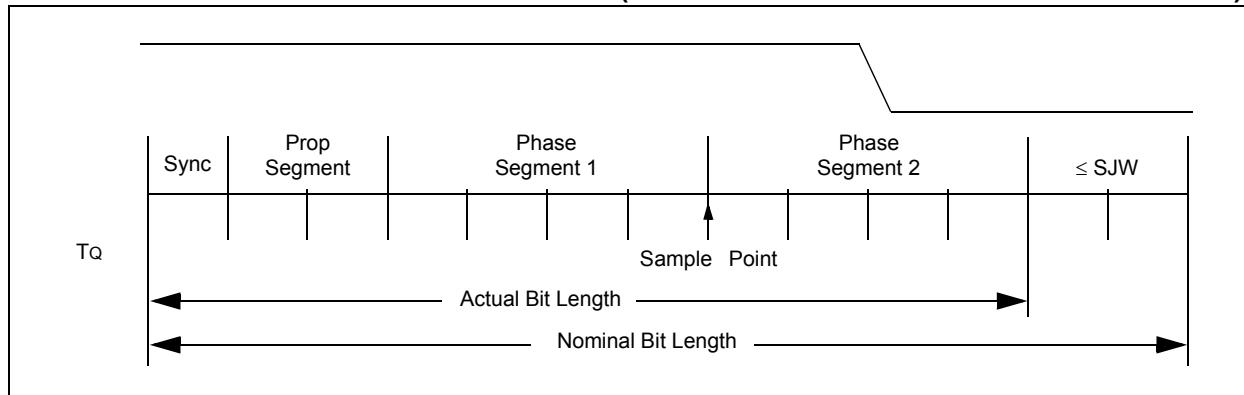
### 27.10.3 SYNCHRONIZATION RULES

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

**FIGURE 27-6: LENGTHENING A BIT PERIOD (ADDING SJW TO PHASE SEGMENT 1)**



**FIGURE 27-7: SHORTENING A BIT PERIOD (SUBTRACTING SJW FROM PHASE SEGMENT 2)**



## 27.11 Programming Time Segments

Some requirements for programming of the time segments:

- Prop\_Seg + Phase\_Seg 1  $\geq$  Phase\_Seg 2
- Phase\_Seg 2  $\geq$  Sync Jump Width.

For example, assume that a 125 kHz CAN baud rate is desired, using 20 MHz for Fosc. With a Tosc of 50 ns, a baud rate prescaler value of 04h gives a TQ of 500 ns. To obtain a Nominal Bit Rate of 125 kHz, the Nominal Bit Time must be 8  $\mu$ s or 16 TQ.

Using 1 TQ for the Sync\_Seg, 2 TQ for the Prop\_Seg and 7 TQ for Phase Segment 1 would place the sample point at 10 TQ after the transition. This leaves 6 TQ for Phase Segment 2.

By the rules above, the Sync Jump Width could be the maximum of 4 TQ. However, normally a large SJW is only necessary when the clock generation of the different nodes is inaccurate or unstable, such as using ceramic resonators. Typically, an SJW of 1 is enough.

## 27.12 Oscillator Tolerance

As a rule of thumb, the bit timing requirements allow ceramic resonators to be used in applications with transmission rates of up to 125 Kbit/sec. For the full bus speed range of the CAN protocol, a quartz oscillator is required. Refer to ISO11898-1 for oscillator tolerance requirements.

# PIC18F66K80 FAMILY

---

## 27.13 Bit Timing Configuration Registers

The Baud Rate Control registers (BRGCON1, BRGCON2, BRGCON3) control the bit timing for the CAN bus interface. These registers can only be modified when the PIC18F66K80 family devices are in Configuration mode.

### 27.13.1 BRGCON1

The BRP bits control the baud rate prescaler. The SJW<1:0> bits select the synchronization jump width in terms of multiples of TQ.

### 27.13.2 BRGCON2

The PRSEG bits set the length of the propagation segment in terms of TQ. The SEG1PH bits set the length of Phase Segment 1 in TQ. The SAM bit controls how many times the RXCAN pin is sampled. Setting this bit to a '1' causes the bus to be sampled three times: twice at TQ/2 before the sample point and once at the normal sample point (which is at the end of Phase Segment 1). The value of the bus is determined to be the value read during at least two of the samples. If the SAM bit is set to a '0', then the RXCAN pin is sampled only once at the sample point. The SEG2PHTS bit controls how the length of Phase Segment 2 is determined. If this bit is set to a '1', then the length of Phase Segment 2 is determined by the SEG2PH bits of BRGCON3. If the SEG2PHTS bit is set to a '0', then the length of Phase Segment 2 is the greater of Phase Segment 1 and the information processing time (which is fixed at 2 TQ for the PIC18F66K80 family).

### 27.13.3 BRGCON3

The PHSEG2<2:0> bits set the length (in TQ) of Phase Segment 2 if the SEG2PHTS bit is set to a '1'. If the SEG2PHTS bit is set to a '0', then the PHSEG2<2:0> bits have no effect.

## 27.14 Error Detection

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

### 27.14.1 CRC ERROR

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

### 27.14.2 ACKNOWLEDGE ERROR

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

### 27.14.3 FORM ERROR

If a node detects a dominant bit in one of the four segments, including End-of-Frame (EOF), interframe space, Acknowledge delimiter or CRC delimiter, then a form error has occurred and an error frame is generated. The message is repeated.

### 27.14.4 BIT ERROR

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

### 27.14.5 STUFF BIT ERROR

If, between the Start-of-Frame (SOF) and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit stuffing rule has been violated. A stuff bit error occurs and an error frame is generated. The message is repeated.

### 27.14.6 ERROR STATES

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

### 27.14.7 ERROR MODES AND ERROR COUNTERS

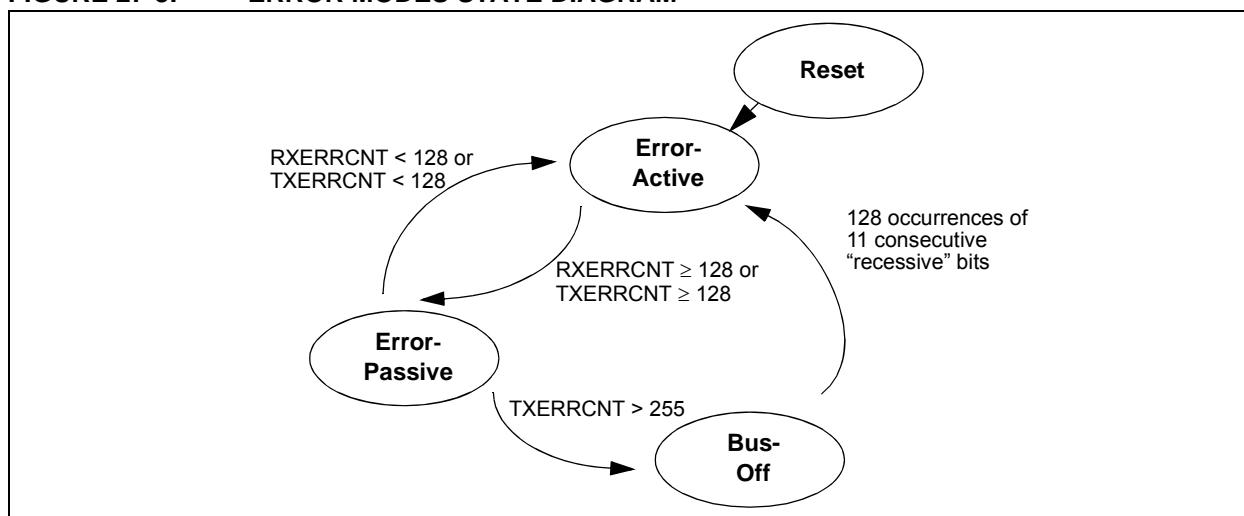
The PIC18F66K80 family devices contain two error counters: the Receive Error Counter (RXERRCNT) and the Transmit Error Counter (TXERRCNT). The values of both counters can be read by the MCU. These counters are incremented or decremented in accordance with the CAN bus specification.

The PIC18F66K80 family devices are error-active if both error counters are below the error-passive limit of 128. They are error-passive if at least one of the error counters equals or exceeds 128. They go to bus-off if the transmit error counter equals or exceeds the bus-off limit of 256. The devices remain in this state until the bus-off recovery sequence is finished. The bus-off recovery sequence consists of 128 occurrences of 11 consecutive recessive bits (see Figure 27-8). Note that the CAN module, after going bus-off, will recover back to error-active without any intervention by the

MCU if the bus remains idle for  $128 \times 11$  bit times. If this is not desired, the error Interrupt Service Routine should address this. The current Error mode of the CAN module can be read by the MCU via the COMSTAT register.

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

**FIGURE 27-8: ERROR MODES STATE DIAGRAM**



## 27.15 CAN Interrupts

The module has several sources of interrupts. Each of these interrupts can be individually enabled or disabled. The PIR5 register contains interrupt flags. The PIE5 register contains the enables for the 8 main interrupts. A special set of read-only bits in the CANSTAT register, the ICODE bits, can be used in combination with a jump table for efficient handling of interrupts.

All interrupts have one source, with the exception of the error interrupt and buffer interrupts in Mode 1 and 2. Any of the error interrupt sources can set the error interrupt flag. The source of the error interrupt can be determined by reading the Communication Status register, COMSTAT. In Mode 1 and 2, there are two interrupt enable/disable and flag bits – one for all transmit buffers and the other for all receive buffers.

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

The receive related interrupts are:

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

The transmit related interrupts are:

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

# PIC18F66K80 FAMILY

## 27.15.1 INTERRUPT CODE BITS

To simplify the interrupt handling process in user firmware, the ECAN module encodes a special set of bits. In Mode 0, these bits are ICODE<3:1> in the CANSTAT register. In Mode 1 and 2, these bits are EICODE<4:0> in the CANSTAT register. Interrupts are internally prioritized such that the higher priority interrupts are assigned lower values. Once the highest priority interrupt condition has been cleared, the code for the next highest priority interrupt that is pending (if any) will be reflected by the ICODE bits (see [Table 27-4](#)). Note that only those interrupt sources that have their associated interrupt enable bit set will be reflected in the ICODE bits.

In Mode 2, when a receive message interrupt occurs, the EICODE bits will always consist of '10000'. User firmware may use FIFO Pointer bits to actually access the next available buffer.

## 27.15.2 TRANSMIT INTERRUPT

When the transmit interrupt is enabled, an interrupt will be generated when the associated transmit buffer becomes empty and is ready to be loaded with a new message. In Mode 0, there are separate interrupt enable/disable and flag bits for each of the three dedicated transmit buffers. The TXBnIF bit will be set to indicate the source of the interrupt. The interrupt is cleared by the MCU, resetting the TXBnIF bit to a '0'. In Mode 1 and 2, all transmit buffers share one interrupt enable/disable bit and one flag bit. In Mode 1 and 2, TXBnIE in PIE5 and TXBnIF in PIR5 indicate when a transmit buffer has completed transmission of its message. TXBnIF, TXBnIE and TXBnIP in PIR5, PIE5 and IPR5, respectively, are not used in Mode 1 and 2. Individual transmit buffer interrupts can be enabled or disabled by setting or clearing TXBnIE and B0IE register bits. When a shared interrupt occurs, user firmware must poll the TXREQ bit of all transmit buffers to detect the source of interrupt.

## 27.15.3 RECEIVE INTERRUPT

When the receive interrupt is enabled, an interrupt will be generated when a message has been successfully received and loaded into the associated receive buffer. This interrupt is activated immediately after receiving the End-of-Frame (EOF) field.

In Mode 0, the RXBnIF bit is set to indicate the source of the interrupt. The interrupt is cleared by the MCU, resetting the RXBnIF bit to a '0'.

In Mode 1 and 2, all receive buffers share RXBnIE, RXBnIF and RXBnIP in PIE5, PIR5 and IPR5, respectively. Bits, RXBnIE, RXBnIF and RXBnIP, are not used. Individual receive buffer interrupts can be controlled by the TXBnIE and BIE0 registers. In Mode 1, when a shared receive interrupt occurs, user firmware must poll the RXFUL bit of each receive buffer to detect the source of interrupt. In Mode 2, a receive interrupt indicates that the new message is loaded into FIFO. FIFO can be read by using FIFO Pointer bits, FP.

**TABLE 27-4: VALUES FOR ICODE<2:0>**

IODEC <2:0>	Interrupt	Boolean Expression
000	None	ERR•WAK•TX0•TX1•TX2•RX0•RX1
001	Error	ERR
010	TXB2	ERR•TX0•TX1•TX2
011	TXB1	ERR•TX0•TX1
100	TXB0	ERR•TX0
101	RXB1	ERR•TX0•TX1•TX2•RX0•RX1
110	RXB0	ERR•TX0•TX1•TX2•RX0
111	Wake on Interrupt	ERR•TX0•TX1•TX2•RX0•RX1•WAK

**Legend:**

ERR = ERRIF \* ERRIE      RX0 = RXB0IF \* RXB0IE  
TX0 = TXB0IF \* TXB0IE      RX1 = RXB1IF \* RXB1IE  
TX1 = TXB1IF \* TXB1IE      WAK = WAKIF \* WAKIE  
TX2 = TXB2IF \* TXB2IE

## 27.15.4 MESSAGE ERROR INTERRUPT

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

## 27.15.5 BUS ACTIVITY WAKE-UP INTERRUPT

When the PIC18F66K80 family devices are in Sleep mode and the bus activity wake-up interrupt is enabled, an interrupt will be generated and the WAKIF bit will be set when activity is detected on the CAN bus. This interrupt causes the PIC18F66K80 family devices to exit Sleep mode. The interrupt is reset by the MCU, clearing the WAKIF bit.

## 27.15.6 ERROR INTERRUPT

When the CAN module error interrupt (ERRIE in PIE5) is enabled, an interrupt is generated if an overflow condition occurs, or if the error state of the transmitter or receiver has changed. The error flags in COMSTAT will indicate one of the following conditions.

## 27.15.6.1 Receiver Overflow

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

## 27.15.6.2 Receiver Warning

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

## 27.15.6.3 Transmitter Warning

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

## 27.15.6.4 Receiver Bus Passive

This will occur when the device has gone to the error-passive state because the receive error counter is greater or equal to 128.

## 27.15.6.5 Transmitter Bus Passive

This will occur when the device has gone to the error-passive state because the transmit error counter is greater or equal to 128.

## 27.15.6.6 Bus-Off

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

# PIC18F66K80 FAMILY

---

---

## NOTES:

## 28.0 SPECIAL FEATURES OF THE CPU

The PIC18F66K80 family of devices includes several features intended to maximize reliability and minimize cost through elimination of external components. These include:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT) and On-Chip Regulator
- 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, the PIC18F66K80 family of devices has 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 (LF-INTOSC) 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.

## 28.1 Configuration Bits

The Configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped starting at program memory location, 300000h.

The user will note that address, 300000h, is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Software 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 the 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”](#).

# PIC18F66K80 FAMILY

---

**TABLE 28-1: 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
300000h	CONFIG1L	—	XINST	—	SOSCSEL1	SOSCSEL0	INTOSCSEL	—	REten	-1-1 11-1
300001h	CONFIG1H	IESO	FCMEN	—	PLLCFG	FOSC3	FOSC2	FOSC1	FOSCO	00-0 1000
300002h	CONFIG2L	—	BORPWR1	BORWPR0	BORV1	BORV0	BOREN1	BOREN0	PWRten	-111 1111
300003h	CONFIG2H	—	WDTPS4	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN1	WDTEN0	-111 1111
300005h	CONFIG3H	MCLRE	—	—	—	MSSPMsk	T3CKMX <sup>(1,3)</sup>	TOCKMX <sup>(1)</sup>	CANMX	1--- 1qq1
300006h	CONFIG4L	DEBUG	—	—	BBSIZ0	—	—	—	STVREN	1--1 ---1
300008h	CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0	---- 1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0	---- 1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0	---- 1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1 <sup>(2)</sup>	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	xxxx xxxx
3FFFFFh	DEVID2 <sup>(2)</sup>	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	xxxx xxxx

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

**Note 1:** Implemented only on the 64-pin devices (PIC18F6XK80).

**2:** See [Register 28-13](#) for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

**3:** Maintain as '0' on 28-pin, 40-pin and 44-pin devices.

# PIC18F66K80 FAMILY

## REGISTER 28-1: CONFIG1L: CONFIGURATION REGISTER 1 LOW (BYTE ADDRESS 300000h)

U-0	R/P-1	U-0	R/P-1	R/P-1	R/P-1	U-0	R/P-1
—	XINST	—	SOSCSEL1	SOSCSEL0	INTOSCSEL	—	RETN
bit 7	bit 0						

<b>Legend:</b>	P = Programmable bit
R = Readable bit	W = Writable bit
-n = Value at POR	‘1’ = Bit is set ‘0’ = Bit is cleared x = Bit is unknown

- bit 7           **Unimplemented:** Read as ‘0’
- bit 6           **XINST:** Extended Instruction Set Enable bit  
    1 = Instruction set extension and Indexed Addressing mode are enabled  
    0 = Instruction set extension and Indexed Addressing mode are disabled (Legacy mode)
- bit 5           **Unimplemented:** Read as ‘0’
- bit 4-3       **SOSCSEL<1:0>:** SOSC Power Selection and Mode Configuration bits  
    11 = High-power SOSC circuit is selected  
    10 = Digital (SCLKI) mode; I/O port functionality of RC0 and RC1 is enabled  
    01 = Low-power SOSC circuit is selected  
    00 = Reserved
- bit 2           **INTOSCSEL:** LF-INTOSC Low-power Enable bit  
    1 = LF-INTOSC in High-Power mode during Sleep  
    0 = LF-INTOSC in Low-Power mode during Sleep
- bit 1           **Unimplemented:** Read as ‘0’
- bit 0           **RETN:** VREG Sleep Enable bit  
    1 = Ultra low-power regulator is disabled. Regulator power in Sleep mode is controlled by REGSLP (WDTCON<7>).  
    0 = Ultra low-power regulator is enabled. Regulator power in Sleep mode is controlled by SRETEN (WDTCON<4>).

# PIC18F66K80 FAMILY

## REGISTER 28-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

R/P-0 IESO	R/P-0 FCMEN	U-0 —	U-0 PLLCFG <sup>(1)</sup>	R/P-1 FOSC3 <sup>(2)</sup>	R/P-0 FOSC2 <sup>(2)</sup>	R/P-0 FOSC1 <sup>(2)</sup>	R/P-0 FOSC0 <sup>(2)</sup>
bit 7							bit 0

<b>Legend:</b>	P = Programmable bit		
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      **IESO:** Internal/External Oscillator Switchover bit  
1 = Two-Speed Start-up is enabled  
0 = Two-Speed Start-up is disabled
- bit 6      **FCMEN:** Fail-Safe Clock Monitor Enable bit  
1 = Fail-Safe Clock Monitor is enabled  
0 = Fail-Safe Clock Monitor is disabled
- bit 5      **Unimplemented:** Read as '0'
- bit 4      **PLLCFG:** 4X PLL Enable bit<sup>(1)</sup>  
1 = Oscillator is multiplied by 4  
0 = Oscillator is used directly
- bit 3-0     **FOSC<3:0>:** Oscillator Selection bits<sup>(2)</sup>  
1101 = EC1, EC oscillator (**low power, DC-160 kHz**)  
1100 = EC1IO, EC oscillator with CLKOUT function on RA6 (**low power, DC-160 kHz**)  
1011 = EC2, EC oscillator (**medium power, 160 kHz-16 MHz**)  
1010 = EC2IO, EC oscillator with CLKOUT function on RA6 (**medium power, 160 kHz-16 MHz**)  
0101 = EC3, EC oscillator (**high power, 16 MHz-64 MHz**)  
0100 = EC3IO, EC oscillator with CLKOUT function on RA6 (**high power, 16 MHz-64 MHz**)  
0011 = HS1, HS oscillator (**medium power, 4 MHz-16 MHz**)  
0010 = HS2, HS oscillator (**high power, 16 MHz-25 MHz**)  
0001 = XT oscillator  
0000 = LP oscillator  
0111 = RC, external RC oscillator  
0110 = RCIO, external RC oscillator with CKLOUT function on RA6  
1000 = INTIO2, internal RC oscillator  
1001 = INTIO1, internal RC oscillator with CLKOUT function on RA6

- Note 1:** Not valid for the INTIOx PLL mode.
- 2:** INTIO + PLL can be enabled only by the PLLEN bit (OSCTUNE<6>). Other PLL modes can be enabled by either the PLLEN bit or the PLLCFG (CONFIG1H<4>) bit.

# PIC18F66K80 FAMILY

## REGISTER 28-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)

U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	BORPWR1 <sup>(1)</sup>	BORPWR0 <sup>(1)</sup>	BORV1 <sup>(1)</sup>	BORV0 <sup>(1)</sup>	BOREN1 <sup>(2)</sup>	BOREN0 <sup>(2)</sup>	PWRTE <sup>(2)</sup>
bit 7							bit 0

<b>Legend:</b>	P = Programmable bit		
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	<b>Unimplemented:</b> Read as '0'
bit 6-5	<b>BORPWR&lt;1:0&gt;:</b> BORMV Power-Level bits <sup>(1)</sup> 11 = ZPBORVMV instead of BORMV is selected 10 = BORMV is set to a high-power level 01 = BORMV is set to a medium power level 00 = BORMV is set to a low-power level
bit 4-3	<b>BORV&lt;1:0&gt;:</b> Brown-out Reset Voltage bits <sup>(1)</sup> 11 = BVDD is set to 1.8V 10 = BVDD is set to 2.0V 01 = BVDD is set to 2.7V 00 = BVDD is set to 3.0V
bit 2-1	<b>BOREN&lt;1:0&gt;:</b> Brown-out Reset Enable bits <sup>(2)</sup> 11 = Brown-out Reset is enabled in hardware only (SBOREN is disabled) 10 = Brown-out Reset is enabled in hardware only and disabled in Sleep mode (SBOREN is disabled) 01 = Brown-out Reset is enabled and controlled by software (SBOREN is enabled) 00 = Brown-out Reset is disabled in hardware and software
bit 0	<b>PWRTE:</b> Power-up Timer Enable bit <sup>(2)</sup> 1 = PWRT disabled 0 = PWRT enabled

**Note 1:** For the specifications, see [Section 31.1 “DC Characteristics: Supply Voltage PIC18F66K80 Family \(Industrial/Extended\)”](#).

**2:** The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.

# PIC18F66K80 FAMILY

## REGISTER 28-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)

U-0	R/P-1						
—	WDTPS4	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN1	WDTEN0
bit 7	bit 0						

<b>Legend:</b>	P = Programmable bit		
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	<b>Unimplemented:</b> Read as '0'
bit 6-2	<b>WDTPS&lt;4:0&gt;:</b> Watchdog Timer Postscale Select bits
	11111 = Reserved
	10100 = 1:1,048,576 (4,194.304s)
	10011 = 1:524,288 (2,097.152s)
	10010 = 1:262,144 (1,048.576s)
	10001 = 1:131,072 (524.288s)
	10000 = 1:65,536 (262.144s)
	01111 = 1:32,768 (131.072s)
	01110 = 1:16,384 (65.536s)
	01101 = 1:8,192 (32.768s)
	01100 = 1:4,096 (16.384s)
	01011 = 1:2,048 (8.192s)
	01010 = 1:1,024 (4.096s)
	01001 = 1:512 (2.048s)
	01000 = 1:256 (1.024s)
	00111 = 1:128 (512 ms)
	00110 = 1:64 (256 ms)
	00101 = 1:32 (128 ms)
	00100 = 1:16 (64 ms)
	00011 = 1:8 (32 ms)
	00010 = 1:4 (16 ms)
	00001 = 1:2 (8 ms)
	00000 = 1:1 (4 ms)
bit 1-0	<b>WDTEN&lt;1:0&gt;:</b> Watchdog Timer Enable bits
	11 = WDT is enabled in hardware; SWDTEN bit is disabled
	10 = WDT is controlled by the SWDTEN bit setting
	01 = WDT is enabled only while the device is active and is disabled in Sleep mode; SWDTEN bit is disabled
	00 = WDT is disabled in hardware; SWDTEN bit is disabled

# PIC18F66K80 FAMILY

## REGISTER 28-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

R/P-1	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
MCLRE	—	—	—	MSSPMSK	T3CKMX <sup>(1)</sup>	T0CKMX <sup>(1)</sup>	CANMX
bit 7	bit 0						

<b>Legend:</b>	P = Programmable bit		
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	<b>MCLRE:</b> MCLR Pin Enable bit 1 = MCLR pin is enabled; RE3 input pin is disabled 0 = RE3 input pin is enabled; MCLR is disabled
bit 6-4	<b>Unimplemented:</b> Read as '0'
bit 3	<b>MSSPMSK:</b> MSSP V3 7-Bit Address Masking Mode Enable bit 1 = 7-Bit Address Masking mode is enabled 0 = 5-Bit Address Masking mode is enabled
bit 2	<b>T3CKMX:</b> Timer3 Clock Input MUX bit <sup>(1)</sup> 1 = Timer3 gets its clock input from the RG2/T3CKI pin on 64-pin packages 0 = Timer3 gets its clock input from the RB5/T3CKI pin on 64-pin packages
bit 1	<b>T0CKMX:</b> Timer0 Clock Input MUX bit <sup>(1)</sup> 1 = Timer0 gets its clock input from the RB5/T0CKI pin on 64-pin packages 0 = Timer0 gets its clock input from the RG4/T0CKI pin on 64-pin packages
bit 0	<b>CANMX:</b> ECAN MUX bit 1 = CANTX and CANRX pins are located on RB2 and RB3, respectively 0 = CANTX and CANRX pins are located on RC6 and RC7, respectively (28-pin and 40/44-pin packages) or on RE4 and RE5, respectively (64-pin package)

**Note 1:** These bits are implemented only on the 64-pin devices (PIC18F6XK80); maintain as '0' on 28-pin, 40-pin and 44-pin devices.

# PIC18F66K80 FAMILY

## REGISTER 28-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	U-0	U-0	R/P-0	U-0	U-0	U-0	R/P-1
DEBUG	—	—	BBSIZ0	—	—	—	STVREN
bit 7							bit 0

<b>Legend:</b>	P = Programmable bit
R = Readable bit	W = Writable bit
-n = Value at POR	‘1’ = Bit is set      ‘0’ = Bit is cleared      x = Bit is unknown

- bit 7            **DEBUG:** Background Debugger Enable bit  
1 = Background debugger is disabled, RB6 and RB7 are configured as general purpose I/O pins  
0 = Background debugger is enabled, RB6 and RB7 are dedicated to In-Circuit Debug
- bit 6-5          **Unimplemented:** Read as ‘0’
- bit 4            **BBSIZ0:** Boot Block Size Select bit  
1 = 2 kW boot block size  
0 = 1 kW boot block size
- bit 3-1          **Unimplemented:** Read as ‘0’
- bit 0            **STVREN:** Stack Full/Underflow Reset Enable bit  
1 = Stack full/underflow will cause a Reset  
0 = Stack full/underflow will not cause a Reset

# PIC18F66K80 FAMILY

## REGISTER 28-7: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	CP3	CP2	CP1	CP0
bit 7	bit 0						

<b>Legend:</b>	C = Clearable 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 7-4      **Unimplemented:** Read as '0'
- bit 3      **CP3:** Code Protection bit  
1 = Block 3 is not code-protected<sup>(1)</sup>  
0 = Block 3 is code-protected<sup>(1)</sup>
- bit 2      **CP2:** Code Protection bit  
1 = Block 2 is not code-protected<sup>(1)</sup>  
0 = Block 2 is code-protected<sup>(1)</sup>
- bit 1      **CP1:** Code Protection bit  
1 = Block 1 is not code-protected<sup>(1)</sup>  
0 = Block 1 is code-protected<sup>(1)</sup>
- bit 0      **CP0:** Code Protection bit  
1 = Block 0 is not code-protected<sup>(1)</sup>  
0 = Block 0, is code-protected<sup>(1)</sup>

**Note 1:** For the memory size of the blocks, see [Figure 28-6](#).

# PIC18F66K80 FAMILY

## REGISTER 28-8: 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						

<b>Legend:</b>	C = Clearable bit		
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           **CPD:** Data EEPROM Code Protection bit

1 = Data EEPROM is not code-protected

0 = Data EEPROM is code-protected

bit 6           **CPB:** Boot Block Code Protection bit

1 = Boot block is not code-protected<sup>(1)</sup>

0 = Boot block is code-protected<sup>(1)</sup>

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

**Note 1:** For the memory size of the blocks, see [Figure 28-6](#). The boot block size changes with BBSIZ0.

# PIC18F66K80 FAMILY

## REGISTER 28-9: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	WRT3	WRT2	WRT1	WRT0
bit 7	bit 0						

<b>Legend:</b>	C = Clearable bit	
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      **Unimplemented:** Read as '0'
- bit 3      **WRT3:** Write Protection bit  
1 = Block 3 is not write-protected<sup>(1)</sup>  
0 = Block 3 is write-protected<sup>(1)</sup>
- bit 2      **WRT2:** Write Protection bit  
1 = Block 2 is not write-protected<sup>(1)</sup>  
0 = Block 2 is write-protected<sup>(1)</sup>
- bit 1      **WRT1:** Write Protection bit  
1 = Block 1 is not write-protected<sup>(1)</sup>  
0 = Block 1 is write-protected<sup>(1)</sup>
- bit 0      **WRT0:** Write Protection bit  
1 = Block 0 is not write-protected<sup>(1)</sup>  
0 = Block 0 is write-protected<sup>(1)</sup>

**Note 1:** For the memory size of the blocks, see [Figure 28-6](#).

# PIC18F66K80 FAMILY

## REGISTER 28-10: 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

<b>Legend:</b>	C = Clearable bit		
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           **WRTD:** Data EEPROM Write Protection bit

1 = Data EEPROM is not write-protected

0 = Data EEPROM is write-protected

bit 6           **WRTB:** Boot Block Write Protection bit

1 = Boot block is not write-protected<sup>(2)</sup>

0 = Boot block is write-protected<sup>(2)</sup>

bit 5           **WRTC:** Configuration Register Write Protection bit<sup>(1)</sup>

1 = Configuration registers are not write-protected<sup>(2)</sup>

0 = Configuration registers are write-protected<sup>(2)</sup>

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.

**2:** For the memory size of the blocks, see [Figure 28-6](#).

# PIC18F66K80 FAMILY

## REGISTER 28-11: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0
bit 7	bit 0						

<b>Legend:</b>	C = Clearable 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 7-4      **Unimplemented:** Read as '0'
- bit 3      **EBTR3:** Table Read Protection bit  
1 = Block 3 is not protected from table reads executed in other blocks<sup>(1)</sup>  
0 = Block 3 is protected from table reads executed in other blocks<sup>(1)</sup>
- bit 2      **EBTR2:** Table Read Protection bit  
1 = Block 2 is not protected from table reads executed in other blocks<sup>(1)</sup>  
0 = Block 2 is protected from table reads executed in other blocks<sup>(1)</sup>
- bit 1      **EBTR1:** Table Read Protection bit  
1 = Block 1 is not protected from table reads executed in other blocks<sup>(1)</sup>  
0 = Block 1 is protected from table reads executed in other blocks<sup>(1)</sup>
- bit 0      **EBTR0:** Table Read Protection bit  
1 = Block 0 is not protected from table reads executed in other blocks<sup>(1)</sup>  
0 = Block 0 is protected from table reads executed in other blocks<sup>(1)</sup>

**Note 1:** For the memory size of the blocks, see [Figure 28-6](#).

# PIC18F66K80 FAMILY

## REGISTER 28-12: 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	—	—	—	—	—	—
bit 7	bit 0						

<b>Legend:</b>	C = Clearable 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      **Unimplemented:** Read as '0'

bit 6      **EBTRB:** Boot Block Table Read Protection bit

1 = Boot block is not protected from table reads executed in other blocks<sup>(1)</sup>

0 = Boot block is protected from table reads executed in other blocks<sup>(1)</sup>

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

**Note 1:** For the memory size of the blocks, see [Figure 28-6](#).

# PIC18F66K80 FAMILY

## REGISTER 28-13: DEVID1: DEVICE ID REGISTER 1 FOR THE PIC18F66K80 FAMILY

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
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

### DEV<2:0>: Device ID bits

These bits are used with the DEV<10:3> bits in the Device ID Register 2 to identify the part number:

000 = PIC18F46K80, PIC18LF26K80  
001 = PIC18F26K80, PIC18LF65K80  
010 = PIC18F65K80, PIC18LF45K80  
011 = PIC18F45K80, PIC18LF25K80  
100 = PIC18F25K80  
110 = PIC18LF66K80  
111 = PIC18F66K80, PIC18LF46K80

bit 4-0

### REV<4:0>: Revision ID bits

These bits are used to indicate the device revision.

## REGISTER 28-14: DEVID2: DEVICE ID REGISTER 2 FOR THE PIC18F66K80 FAMILY

R	R	R	R	R	R	R	R	R
DEV10 <sup>(1)</sup>	DEV9 <sup>(1)</sup>	DEV8 <sup>(1)</sup>	DEV7 <sup>(1)</sup>	DEV6 <sup>(1)</sup>	DEV5 <sup>(1)</sup>	DEV4 <sup>(1)</sup>	DEV3 <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 7-0

### DEV<10:3>: Device ID bits<sup>(1)</sup>

These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.

**Note 1:** These values for DEV<10:3> may be shared with other devices. The specific device is always identified by using the entire DEV<10:0> bit sequence.

# PIC18F66K80 FAMILY

## 28.2 Watchdog Timer (WDT)

For the PIC18F66K80 family of devices, the WDT is driven by the LF-INTOSC 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 LF-INTOSC 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 4,194 seconds (about one hour). The WDT and postscaler are cleared when any of the following events occur: a SLEEP or CLRWDW instruction is executed, the IRCFx bits (OSCCON<6:4>) are changed or a clock failure has occurred.

The WDT can be operated in one of four modes as determined by WDTEN<1:0> (CONFIG2H<1:0>). The four modes are:

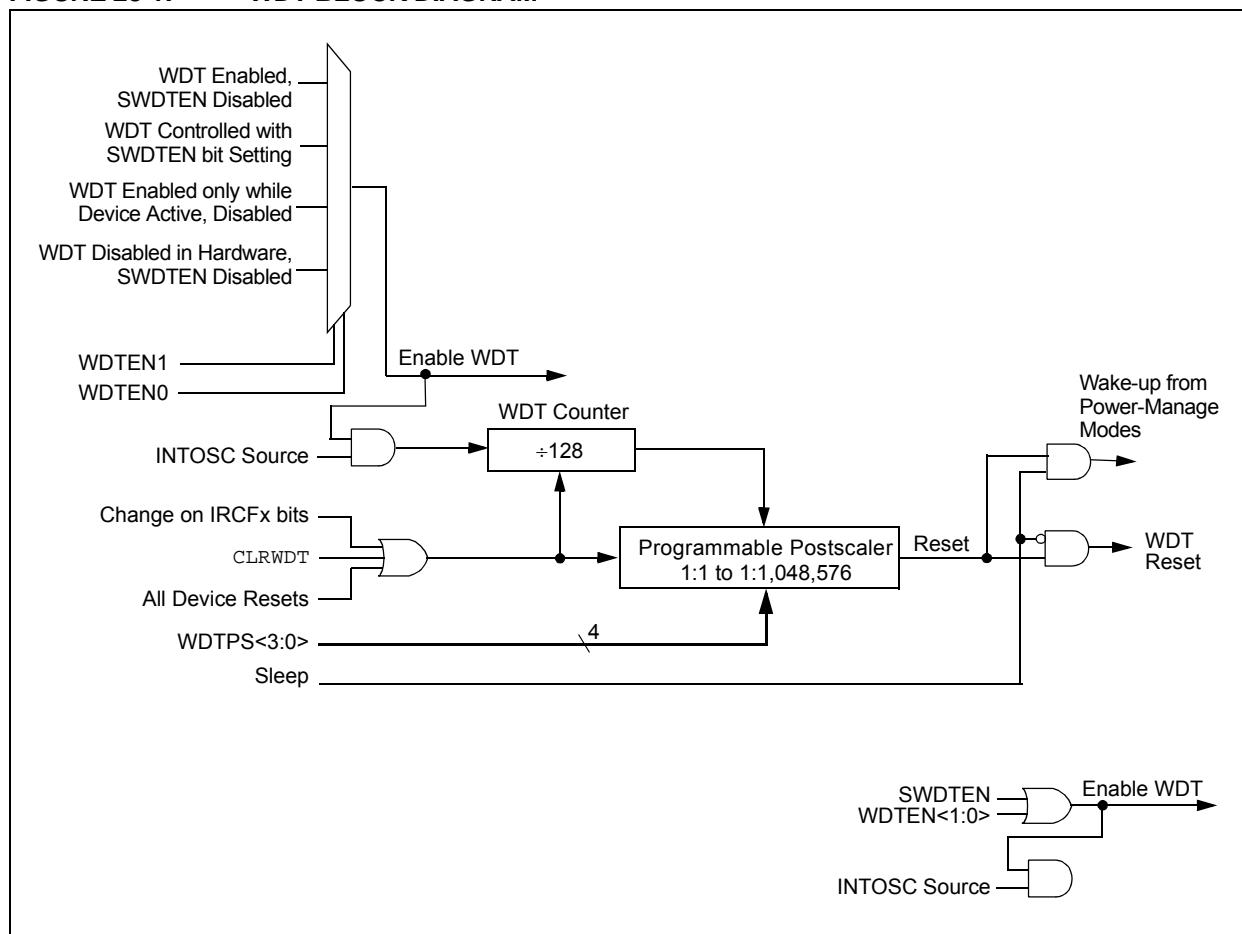
- WDT Enabled
- WDT Disabled
- WDT under Software Control, SWDTEN (WDTCON<0>)
- WDT
  - Enabled during normal operation
  - Disabled during Sleep

**Note 1:** The CLRWDW and SLEEP instructions clear the WDT and postscaler counts when executed.

**2:** Changing the setting of the IRCFx bits (OSCCON<6:4>) clears the WDT and postscaler counts.

**3:** When a CLRWDW instruction is executed, the postscaler count will be cleared.

FIGURE 28-1: WDT BLOCK DIAGRAM



## 28.2.1 CONTROL REGISTER

Register 28-15 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.

### REGISTER 28-15: WDTCON: WATCHDOG TIMER CONTROL REGISTER

R/W-0	U-0	R-x	R/W-0	U-0	R/W-x	R/W-x	R/W-0
REGSLP <sup>(3)</sup>	—	ULPLVL	SRETEN <sup>(2)</sup>	—	ULPEN	ULPSINK	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	<b>REGSLP:</b> Regulator Voltage Sleep Enable bit <sup>(3)</sup> 1 = Regulator goes into Low-Power mode when device's Sleep mode is enabled 0 = Regulator stays in normal mode when device's Sleep mode is activated
bit 6	<b>Unimplemented:</b> Read as '0'
bit 5	<b>ULPLVL:</b> Ultra Low-Power Wake-up Output bit Not valid unless ULPEN = 1. 1 = Voltage on RA0 pin > ~ 0.5V 0 = Voltage on RA0 pin < ~ 0.5V.
bit 4	<b>SRETEN:</b> Regulator Voltage Sleep Disable bit <sup>(2)</sup> 1 = If RETEN (CONFIG1L<0>) = 0 and the regulator is enabled, the device goes into Ultra Low-Power mode in Sleep 0 = The regulator is on when device's Sleep mode is enabled and the Low-Power mode is controlled by REGSLP
bit 3	<b>Unimplemented:</b> Read as '0'
bit 2	<b>ULPEN:</b> Ultra Low-Power Wake-up Module Enable bit 1 = Ultra Low-Power Wake-up module is enabled; ULPLVL bit indicates comparator output 0 = Ultra Low-Power Wake-up module is disabled
bit 1	<b>ULPSINK:</b> Ultra Low-Power Wake-up Current Sink Enable bit Not valid unless ULPEN = 1. 1 = Ultra Low-Power Wake-up current sink is enabled 0 = Ultra Low-Power Wake-up current sink is disabled
bit 0	<b>SWDTEN:</b> 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 bits, WDTEN<1:0>, are enabled.

**2:** This bit is available only when RETEN = 0.

**3:** This bit is disabled on PIC18LF devices.

### TABLE 28-2: SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RCON	IPEN	SBOREN	CM	RI	TO	PD	POR	BOR
WDTCON	REGSLP	—	ULPLVL	SRETEN	—	ULPEN	ULPSINK	SWDTEN

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

# PIC18F66K80 FAMILY

## 28.3 On-Chip Voltage Regulator

All of the PIC18F66K80 family devices power their core digital logic at a nominal 3.3V. For designs that are required to operate at a higher typical voltage, such as 5V, all family devices incorporate two on-chip regulators that allows the device to run its core logic from VDD. Those regulators are:

- Normal on-chip regulator
- Ultra Low-Power, on-chip regulator

The hardware configuration of these regulators are the same and are explained in [Section 28.3.1 “Regulator Enable Mode \(PIC18FXXX devices\)”](#). The regulators' only differences relate to when the device enters Sleep, as explained in [Section 28.3.1 “Regulator Enable Mode \(PIC18FXXX devices\)”](#).

### 28.3.1 REGULATOR ENABLE MODE (PIC18FXXX devices)

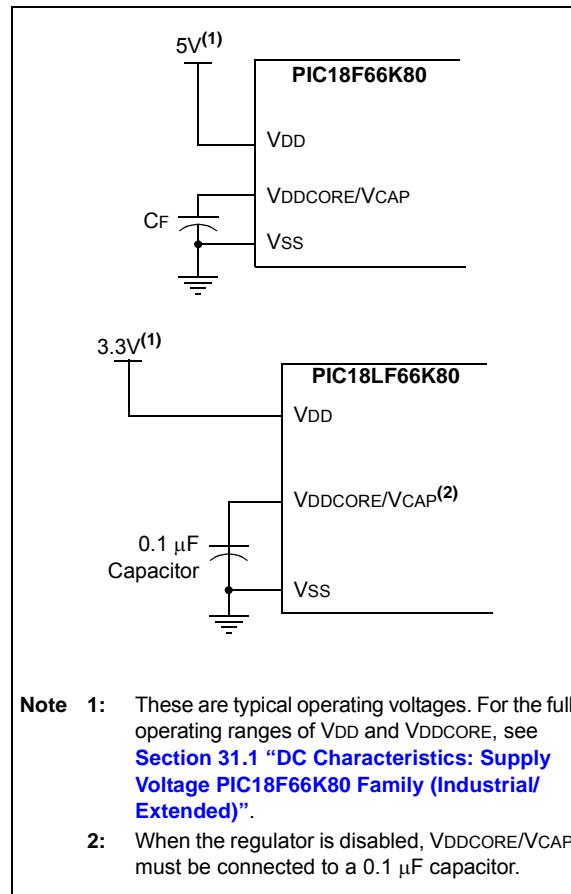
On PIC18FXXX devices, the regulator is enabled and a low-ESR filter capacitor must be connected to the VDDCORE/VCAP pin (see [Figure 28-2](#)). This helps maintain the regulator's stability. The recommended value for the filter capacitor is given in [Section 31.1 “DC Characteristics: Supply Voltage PIC18F66K80 Family \(Industrial/Extended\)”](#).

### 28.3.2 REGULATOR DISABLE MODE (PIC18LFXXX devices)

On PIC18LFXXX devices, the regulator is disabled and the power to the core is supplied directly by VDD. The voltage levels for VDD must not exceed the specified VDDCORE levels. A 0.1  $\mu$ F capacitor should be connected to the VDDCORE/VCAP pin.

On the PIC18FXXX devices, the overall voltage budget is very tight. The regulator should operate the device down to 1.8V. When VDD drops below 3.3V, the regulator no longer regulates, but the output voltage follows the input until VDD reaches 1.8V. Below this voltage, the output of the regulator output may drop to 0V.

**FIGURE 28-2: CONNECTIONS FOR THE F AND LF PARTS**



### 28.3.3 OPERATION OF REGULATOR IN SLEEP

The difference in the two regulators' operation arises with Sleep mode. The ultra low-power regulator gives the device the lowest current in the Regulator Enabled mode.

The on-chip regulator can go into a lower power mode when the device goes to Sleep by setting the REGSLP bit (WDTCON<7>). This puts the regulator in a standby mode so that the device consumes much less current.

The on-chip regulator can also go into the Ultra Low-Power mode, which consumes the lowest current possible with the regulator enabled. This mode is controlled by the RETEN bit (CONFIG1L<0>) and SRETN bit (WDTCON<4>).

The various modes of regulator operation are shown in [Table 28-3](#).

When the ultra low-power regulator is in Sleep mode, the internal reference voltages in the chip will be shut off and any interrupts referring to the internal reference will not wake up the device. If the BOR or LVD is enabled, the regulator will keep the internal references on and the lowest possible current will not be achieved.

When using the ultra low-power regulator in Sleep mode, the device will take about 250  $\mu$ s to start executing code after it wakes up.

**TABLE 28-3: SLEEP MODE REGULATOR SETTINGS<sup>(1)</sup>**

Device	Power Mode	REGSLP WDTCON<7>	SRETN WDTCON<4>	RETEN CONFIG1L<0>
PIC18FXXK80	Normal Operation (Sleep)	0	x	1
PIC18FXXK80	Low-Power mode (Sleep)	1	x	1
PIC18FXXK80	Normal Operation (Sleep)	0	0	0
PIC18FXXK80	Low-Power mode (Sleep)	1	0	0
PIC18FXXK80	Ultra Low-Power mode (Sleep)	x	1	0
PIC18LFXXK80	Reserved <sup>(2)</sup>	x	Don't Care	0
PIC18LFXXK80	Regulator Bypass mode (Sleep) <sup>(2)</sup>	x	x	1

**Note 1:** x — Indicates that VIT status is invalid.

**2:** The ultra low-power regulator should be disabled (RETEN = 1, ULP disabled) on PIC18LFXXK80 devices to obtain the lowest possible Sleep current.

# PIC18F66K80 FAMILY

## 28.4 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 (LF-INTOSC, MF-INTOSC, HF-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 or HS (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, IRCF<2:0>, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:0> bits prior to entering Sleep mode.

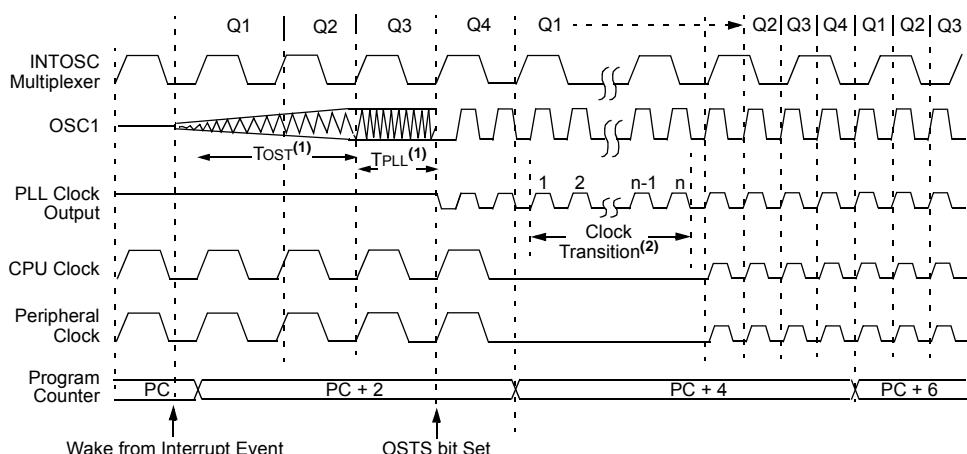
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 28.4.1 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 SCS<1:0> 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 OSTS 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 28-3: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)**



**Note 1:** TOST = 1024 Tosc; TPLL = 2 ms (approx). These intervals are not shown to scale.

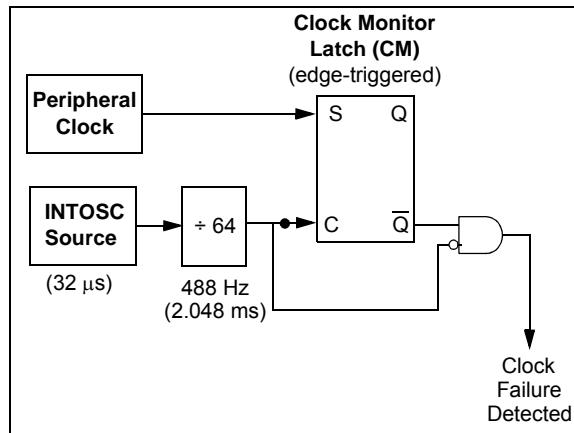
**2:** Clock transition typically occurs within 2-4 Tosc.

## 28.5 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 LF-INTOSC 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 28-4) is accomplished by creating a sample clock signal, which is the output from the LF-INTOSC 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 (CM) latch. The CM is set on the falling edge of the device clock source, but cleared on the rising edge of the sample clock.

**FIGURE 28-4: 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 28-5). This causes the following:

- The FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>)
- The device clock source switches 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 shutdown. See [Section 4.1.4 “Multiple Sleep Commands”](#) and [Section 28.4.1 “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, IRCF<2:0>, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF<2:0> bits prior to entering Sleep mode.

The FSCM will detect only failures of the primary or secondary clock sources. If the internal oscillator block fails, no failure would be detected nor would any action be possible.

### 28.5.1 FSCM AND THE WATCHDOG TIMER

Both the FSCM and the WDT are clocked by the INTOSC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTOSC 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 IRCF<2:0> 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.

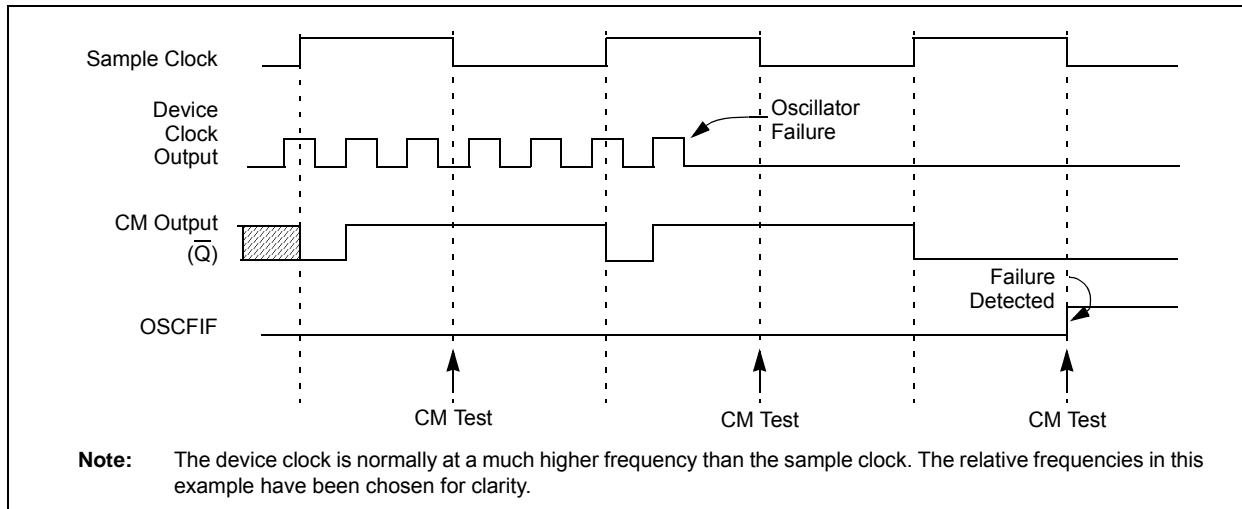
### 28.5.2 EXITING FAIL-SAFE OPERATION

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.

# PIC18F66K80 FAMILY

FIGURE 28-5: FSCM TIMING DIAGRAM



## 28.5.3 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 ( $\text{OSCFIF} = 1$ ), 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.

## 28.5.4 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 INTOSC 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 (when the OST and PLL timers have timed out).

This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTOSC returns to its role as the FSCM source.

**Note:** The same logic that prevents false oscillator failure interrupts on POR, or wake from Sleep, also prevents the detection of the oscillator's failure to start at all following these events. This can be avoided by monitoring the OST 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 28.4.1 “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.

## 28.6 Program Verification and Code Protection

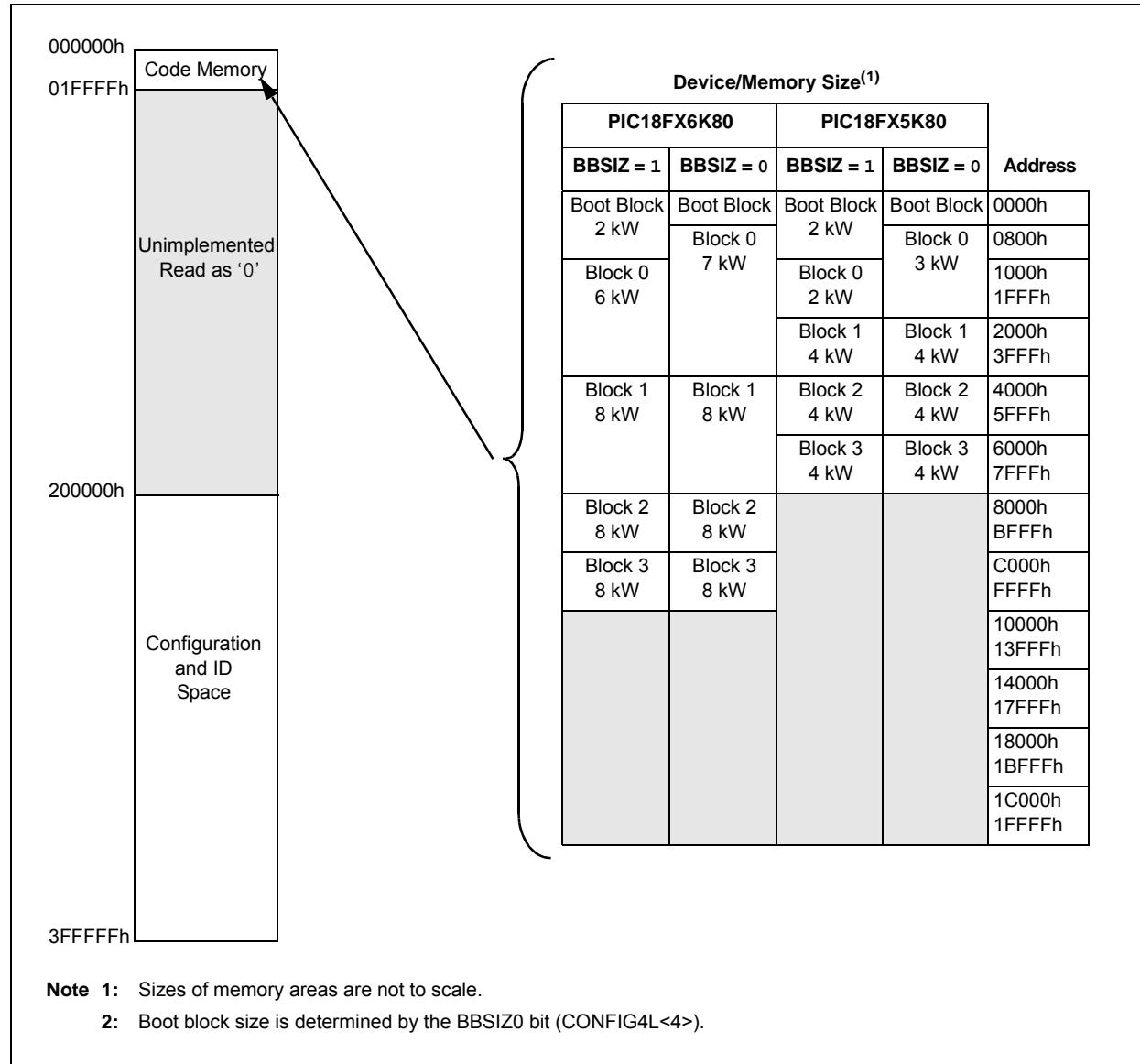
The user program memory is divided into four blocks. One of these is a boot block of 1 or 2 Kbytes. The remainder of the memory is divided into blocks on binary boundaries.

Each of the 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 28-6](#) shows the program memory organization for 48, 64, 96 and 128 Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in [Table 28-4](#).

**FIGURE 28-6: CODE-PROTECTED PROGRAM MEMORY FOR THE PIC18F66K80 FAMILY**



# PIC18F66K80 FAMILY

TABLE 28-4: SUMMARY OF CODE PROTECTION REGISTERS

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0
300009h CONFIG5H	CPD	CPB	—	—	—	—	—	—
3000Ah CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0
3000Bh CONFIG6H	WRTD	WRTB	WRCTC	—	—	—	—	—
3000Ch CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0
3000Dh CONFIG7H	—	EBTRB	—	—	—	—	—	—

**Legend:** Shaded cells are unimplemented.

## 28.6.1 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. [Figure 28-7](#) through [Figure 28-9](#) illustrate table write and table read protection.

**Note:** Code protection bits may only be written to a ‘0’ from a ‘1’ state. It is not possible to write a ‘1’ to a bit in the ‘0’ state. Code protection bits are only set to ‘1’ by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP or an external programmer. Refer to the device programming specification for more information.

FIGURE 28-7: TABLE WRITE (WRTx) DISALLOWED

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h	WRTB, EBTRB = 11
PC = 003FFEh	0007FFh 000800h	WRT0, EBTR0 = 01
PC = 00BFFEh	003FFFh 004000h 007FFFh 008000h 00BFFFh 00C000h 00FFFFh	WRT1, EBTR1 = 11 WRT2, EBTR2 = 11 WRT3, EBTR3 = 11
		<b>Results:</b> All table writes are disabled to Blockn whenever WRTx = 0.

**FIGURE 28-8: EXTERNAL BLOCK TABLE READ (EBTRx) DISALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h 000800h	WRTB, EBTRB = 11
PC = 007FFEh	003FFFh 004000h TBLRD*	WRT0, EBTR0 = 10
	007FFFh 008000h	WRT1, EBTR1 = 11
	00BFFFh 00C000h	WRT2, EBTR2 = 11
	00FFFFh	WRT3, EBTR3 = 11

**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRx = 0.  
The TABLAT register returns a value of '0'.

**FIGURE 28-9: EXTERNAL BLOCK TABLE READ (EBTRx) ALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h 000800h	WRTB, EBTRB = 11
PC = 003FFEh	003FFFh 004000h TBLRD*	WRT0, EBTR0 = 10
	007FFFh 008000h	WRT1, EBTR1 = 11
	00BFFFh 00C000h	WRT2, EBTR2 = 11
	00FFFFh	WRT3, EBTR3 = 11

**Results:** Table reads are permitted within Blockn, even when EBTRBx = 0.  
The TABLAT register returns the value of the data at the location, TBLPTR.

# PIC18F66K80 FAMILY

---

## 28.6.2 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.

## 28.6.3 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 readable only. WRTC can only be written via ICSP or an external programmer.

## 28.7 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.

## 28.8 In-Circuit Serial Programming

The PIC18F66K80 family of devices 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.

For the various programming modes, see the programming specification

## 28.9 In-Circuit Debugger

When the DEBUG 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 28-5](#) shows which resources are required by the background debugger.

**TABLE 28-5: DEBUGGER RESOURCES**

<b>I/O Pins:</b>	RB6, RB7
<b>Stack:</b>	Two levels
<b>Program Memory:</b>	512 bytes
<b>Data Memory:</b>	10 bytes

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to MCLR/RE3, VDD, Vss, RB7 and RB6. This will interface to the In-Circuit Debugger module available from Microchip or one of the third-party development tool companies.

## 29.0 INSTRUCTION SET SUMMARY

The PIC18F66K80 family of devices incorporates 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.

### 29.1 Standard Instruction Set

The standard PIC18 MCU instruction set adds many enhancements to the previous PIC® 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 29-2](#) lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. [Table 29-1](#) 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 29-1](#) 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 29-2](#), lists the standard instructions recognized by the Microchip MPASM™ Assembler.

[Section 29.1.1 “Standard Instruction Set”](#) provides a description of each instruction.

# PIC18F66K80 FAMILY

---

**TABLE 29-1: 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 <b>C</b> arry, <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.
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.
ε	In the set of.
italics	User-defined term (font is Courier New).

# PIC18F66K80 FAMILY

**FIGURE 29-1: GENERAL FORMAT FOR INSTRUCTIONS**

Byte-oriented file register operations	Example Instruction																
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>10</td><td>9</td><td>8</td><td>7</td><td>0</td></tr> <tr> <td>OPCODE</td><td>d</td><td>a</td><td colspan="3">f (FILE #)</td></tr> </table> <p>d = 0 for result destination to be WREG register  d = 1 for result destination to be file register (f)  a = 0 to force Access Bank  a = 1 for BSR to select bank  f = 8-bit file register address</p>	15	10	9	8	7	0	OPCODE	d	a	f (FILE #)			ADDWF MYREG, W, B				
15	10	9	8	7	0												
OPCODE	d	a	f (FILE #)														
<b>Byte to Byte move operations (2-word)</b>																	
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>12</td><td>11</td><td>0</td></tr> <tr> <td>OPCODE</td><td colspan="3">f (Source FILE #)</td></tr> <tr> <td>15</td><td>12</td><td>11</td><td>0</td></tr> <tr> <td>1111</td><td colspan="3">f (Destination FILE #)</td></tr> </table> <p>f = 12-bit file register address</p>	15	12	11	0	OPCODE	f (Source FILE #)			15	12	11	0	1111	f (Destination FILE #)			MOVFF MYREG1, MYREG2
15	12	11	0														
OPCODE	f (Source FILE #)																
15	12	11	0														
1111	f (Destination FILE #)																
<b>Bit-oriented file register operations</b>																	
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>12</td><td>11</td><td>9</td><td>8</td><td>7</td><td>0</td></tr> <tr> <td>OPCODE</td><td>b (BIT #)</td><td>a</td><td colspan="3">f (FILE #)</td><td></td></tr> </table> <p>b = 3-bit position of bit in file register (f)  a = 0 to force Access Bank  a = 1 for BSR to select bank  f = 8-bit file register address</p>	15	12	11	9	8	7	0	OPCODE	b (BIT #)	a	f (FILE #)				BSF MYREG, bit, B		
15	12	11	9	8	7	0											
OPCODE	b (BIT #)	a	f (FILE #)														
<b>Literal operations</b>																	
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>8</td><td>7</td><td>0</td></tr> <tr> <td>OPCODE</td><td colspan="3">k (literal)</td></tr> </table> <p>k = 8-bit immediate value</p>	15	8	7	0	OPCODE	k (literal)			MOVLW 7Fh								
15	8	7	0														
OPCODE	k (literal)																
<b>Control operations</b>																	
<b>CALL, GOTO and Branch operations</b>																	
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>8</td><td>7</td><td>0</td></tr> <tr> <td>OPCODE</td><td colspan="3">n&lt;7:0&gt; (literal)</td></tr> <tr> <td>15</td><td>12</td><td>11</td><td>0</td></tr> <tr> <td>1111</td><td colspan="3">n&lt;19:8&gt; (literal)</td></tr> </table> <p>n = 20-bit immediate value</p>	15	8	7	0	OPCODE	n<7:0> (literal)			15	12	11	0	1111	n<19:8> (literal)			GOTO Label
15	8	7	0														
OPCODE	n<7:0> (literal)																
15	12	11	0														
1111	n<19:8> (literal)																
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>8</td><td>7</td><td>0</td></tr> <tr> <td>OPCODE</td><td>S</td><td colspan="2">n&lt;7:0&gt; (literal)</td></tr> <tr> <td>15</td><td>12</td><td>11</td><td>0</td></tr> <tr> <td>1111</td><td colspan="3">n&lt;19:8&gt; (literal)</td></tr> </table> <p>S = Fast bit</p>	15	8	7	0	OPCODE	S	n<7:0> (literal)		15	12	11	0	1111	n<19:8> (literal)			CALL MYFUNC
15	8	7	0														
OPCODE	S	n<7:0> (literal)															
15	12	11	0														
1111	n<19:8> (literal)																
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>11</td><td>10</td><td>0</td></tr> <tr> <td>OPCODE</td><td colspan="3">n&lt;10:0&gt; (literal)</td></tr> </table>	15	11	10	0	OPCODE	n<10:0> (literal)			BRA MYFUNC								
15	11	10	0														
OPCODE	n<10:0> (literal)																
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>15</td><td>8</td><td>7</td><td>0</td></tr> <tr> <td>OPCODE</td><td colspan="3">n&lt;7:0&gt; (literal)</td></tr> </table>	15	8	7	0	OPCODE	n<7:0> (literal)			BC MYFUNC								
15	8	7	0														
OPCODE	n<7:0> (literal)																

# PIC18F66K80 FAMILY

---

TABLE 29-2: PIC18F66K80 FAMILY 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 01da	fffff	fffff	C, DC, Z, OV, N	1, 2	
ADDWFC f, d, a	Add WREG and Carry bit to f	1	0010 00da	fffff	fffff	C, DC, Z, OV, N	1, 2	
ANDWF f, d, a	AND WREG with f	1	0001 01da	fffff	fffff	Z, N	1, 2	
CLRF f, a	Clear f	1	0110 101a	fffff	fffff	Z	2	
COMF f, d, a	Complement f	1	0001 11da	fffff	fffff	Z, N	1, 2	
CPFSEQ f, a	Compare f with WREG, Skip =	1 (2 or 3)	0110 001a	fffff	fffff	None	4	
CPFSGT f, a	Compare f with WREG, Skip >	1 (2 or 3)	0110 010a	fffff	fffff	None	4	
CPFSLT f, a	Compare f with WREG, Skip <	1 (2 or 3)	0110 000a	fffff	fffff	None	1, 2	
DECF f, d, a	Decrement f	1	0000 01da	fffff	fffff	C, DC, Z, OV, N	1, 2, 3, 4	
DECFSZ f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010 11da	fffff	fffff	None	1, 2, 3, 4	
DCFSNZ f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100 11da	fffff	fffff	None	1, 2	
INCF f, d, a	Increment f	1	0010 10da	fffff	fffff	C, DC, Z, OV, N	1, 2, 3, 4	
INCFSZ f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011 11da	fffff	fffff	None	4	
INFSNZ f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100 10da	fffff	fffff	None	1, 2	
IOWF f, d, a	Inclusive OR WREG with f	1	0001 00da	fffff	fffff	Z, N	1, 2	
MOVF f, d, a	Move f	1	0101 00da	fffff	fffff	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 fffff	fffff	fffff	None		
MOVWF f, a	Move WREG to f	1	0110 111a	fffff	fffff	None		
MULWF f, a	Multiply WREG with f	1	0000 001a	fffff	fffff	None	1, 2	
NEGF f, a	Negate f	1	0110 110a	fffff	fffff	C, DC, Z, OV, N		
RLCF f, d, a	Rotate Left f through Carry	1	0011 01da	fffff	fffff	C, Z, N	1, 2	
RLNCF f, d, a	Rotate Left f (No Carry)	1	0100 01da	fffff	fffff	Z, N		
RRCF f, d, a	Rotate Right f through Carry	1	0011 00da	fffff	fffff	C, Z, N		
RRNCF f, d, a	Rotate Right f (No Carry)	1	0100 00da	fffff	fffff	Z, N		
SETF f, a	Set f	1	0110 100a	fffff	fffff	None	1, 2	
SUBFWB f, d, a	Subtract f from WREG with Borrow	1	0101 01da	fffff	fffff	C, DC, Z, OV, N		
SUBWF f, d, a	Subtract WREG from f	1	0101 11da	fffff	fffff	C, DC, Z, OV, N	1, 2	
SUBWFB f, d, a	Subtract WREG from f with Borrow	1	0101 10da	fffff	fffff	C, DC, Z, OV, N		
SWAPF f, d, a	Swap Nibbles in f	1	0011 10da	fffff	fffff	None	4	
TSTFSZ f, a	Test f, Skip if 0	1 (2 or 3)	0110 011a	fffff	fffff	None	1, 2	
XORWF f, d, a	Exclusive OR WREG with f	1	0001 10da	fffff	fffff	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 an 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.

# PIC18F66K80 FAMILY

TABLE 29-2: PIC18F66K80 FAMILY 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, b, 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		
BNOV n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None		
BNZ 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 1st word 2nd word	2	1110	110s	kkkk	kkkk	None		
			1111	kkkk	kkkk	kkkk			
CLRWDT —	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD		
DAW —	Decimal Adjust WREG	1	0000	0000	0000	0111	C		
GOTO n	Go to Address 1st word 2nd word	2	1110	1111	kkkk	kkkk	None		
			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	TO, 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 an 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.

# PIC18F66K80 FAMILY

---

TABLE 29-2: PIC18F66K80 FAMILY 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 kkkk			None	
			1111 0000	kkkk kkkk				
MOVLB k	Move Literal to BSR<3:0>	1	0000 0001	0000	kkkk		None	
MOVLW k	Move Literal to WREG	1	0000 1110	kkkk kkkk			None	
MULLW k	Multiply Literal with WREG	1	0000 1101	kkkk kkkk			None	
RETLW k	Return with Literal in WREG	2	0000 1100	kkkk kkkk			None	
SUBLW k	Subtract WREG from Literal	1	0000 1000	kkkk kkkk			C, DC, Z, OV, N	
XORLW k	Exclusive OR Literal with WREG	1	0000 1010	kkkk kkkk			Z, N	
<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 an 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.

# PIC18F66K80 FAMILY

## 29.1.1 STANDARD INSTRUCTION SET

<b>ADDLW</b>	<b>ADD Literal to W</b>											
Syntax:	ADDLW k											
Operands:	0 ≤ k ≤ 255											
Operation:	(W) + k → W											
Status Affected:	N, OV, C, DC, Z											
Encoding:	0000	1111	kkkk	kkkk								
Description:	The contents of W are added to the 8-bit literal 'k' and the result is placed in W.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr> </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

<b>ADDWF</b>	<b>ADD W to f</b>											
Syntax:	ADDWF f {,d {,a}}											
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]											
Operation:	(W) + (f) → dest											
Status Affected:	N, OV, C, DC, Z											
Encoding:	0010	01da	ffff	ffff								
Description:	Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4									
Decode	Read register 'f'	Process Data	Write to destination									

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).

# PIC18F66K80 FAMILY

---



---

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:	$(W) + (f) + (C) \rightarrow \text{dest}$								
Status Affected:	N,OV, C, DC, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0010</td> <td>00da</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0010	00da	ffff	ffff				
0010	00da	ffff	ffff						
Description:	<p>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'.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </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: 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:	$(W) .AND. k \rightarrow W$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1011</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	0000	1011	kkkk	kkkk				
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </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: ANDLW 05Fh

Before Instruction  
 W = A3h  
 After Instruction  
 W = 03h

# PIC18F66K80 FAMILY

ANDWF	AND W with f								
Syntax:	ANDWF f {,d {,a}}								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(W) .AND. (f) → dest								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0001</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table>	0001	01da	ffff	ffff				
0001	01da	ffff	ffff						
Description:	<p>The contents of W are ANDed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

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 ≤ n ≤ 127												
Operation:	if Carry bit is '1', (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0010</td><td>nnnn</td><td>nnnn</td></tr> </table>	1110	0010	nnnn	nnnn								
1110	0010	nnnn	nnnn										
Description:	<p>If the Carry bit is '1', then the program will branch.</p> <p>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.</p>												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:													
If Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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 PC = 1;  
If Carry PC = address (HERE + 12)  
If Carry PC = 0;  
If Carry PC = address (HERE + 2)

# PIC18F66K80 FAMILY

---



---

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:	$0 \rightarrow f<b>$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1001</td> <td>bbba</td> <td>ffff</td> <td>ffff</td> </tr> </table>	1001	bbba	ffff	ffff				
1001	bbba	ffff	ffff						
Description:	<p>Bit 'b' in register 'f' is cleared.</p> <p>If 'a' is '0', the Access Bank is selected.  If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1110</td> <td>0110</td> <td>nnnn</td> <td>nnnn</td> </tr> </table>	1110	0110	nnnn	nnnn																
1110	0110	nnnn	nnnn																		
Description:	<p>If the Negative bit is '1', then the program will branch.</p> <p>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 <math>PC + 2 + 2n</math>. This instruction is then a two-cycle instruction.</p>																				
Words:	1																				
Cycles:	1(2)																				
Q Cycle Activity:	<p>If Jump:</p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table> <p>If No Jump:</p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation
Q1	Q2	Q3	Q4																		
Decode	Read literal 'n'	Process Data	Write to PC																		
No operation	No operation	No operation	No operation																		
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 PC = 1;  
PC = address (Jump)  
If Negative PC = 0;  
PC = address (HERE + 2)

# PIC18F66K80 FAMILY

---

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:	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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      PC = 0;  
 If Carry      PC = address (Jump)  
 If Carry      PC = 1;  
 If Carry      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:	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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      PC = 0;  
 If Negative      PC = address (Jump)  
 If Negative      PC = 1;  
 If Negative      PC = address (HERE + 2)

# PIC18F66K80 FAMILY

---



---

## BNOV Branch if Not Overflow

Syntax:	BNOV n															
Operands:	$-128 \leq n \leq 127$															
Operation:	if Overflow bit is '0', $(PC) + 2 + 2n \rightarrow PC$															
Status Affected:	None															
Encoding:	1110	0101	nnnn	nnnn												
Description:	<p>If the Overflow bit is '0', then the program will branch.</p> <p>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 <math>PC + 2 + 2n</math>. This instruction is then a two-cycle instruction.</p>															
Words:	1															
Cycles:	1(2)															
Q Cycle Activity:																
If Jump:	<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 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<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 'n'</td><td>Process Data</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	No operation													

## Example: HERE BNOV Jump

Before Instruction

$$\begin{array}{lcl} PC & = & \text{address (HERE)} \end{array}$$

After Instruction

$$\begin{array}{lcl} \text{If Overflow} & = & 0; \\ PC & = & \text{address (Jump)} \\ \text{If Overflow} & = & 1; \\ PC & = & \text{address (HERE + 2)} \end{array}$$

## BNZ Branch if Not Zero

Syntax:	BNZ n															
Operands:	$-128 \leq n \leq 127$															
Operation:	if Zero bit is '0', $(PC) + 2 + 2n \rightarrow PC$															
Status Affected:	None															
Encoding:	1110	0001	nnnn	nnnn												
Description:	<p>If the Zero bit is '0', then the program will branch.</p> <p>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 <math>PC + 2 + 2n</math>. This instruction is then a two-cycle instruction.</p>															
Words:	1															
Cycles:	1(2)															
Q Cycle Activity:																
If Jump:	<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 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<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 'n'</td><td>Process Data</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	No operation													

## Example: HERE BNZ Jump

Before Instruction

$$\begin{array}{lcl} PC & = & \text{address (HERE)} \end{array}$$

After Instruction

$$\begin{array}{lcl} \text{If Zero} & = & 0; \\ PC & = & \text{address (Jump)} \\ \text{If Zero} & = & 1; \\ PC & = & \text{address (HERE + 2)} \end{array}$$

# PIC18F66K80 FAMILY

---

BRA	Unconditional Branch												
Syntax:	BRA n												
Operands:	-1024 ≤ n ≤ 1023												
Operation:	(PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1101</td> <td>0nnn</td> <td>nnnn</td> <td>nnnn</td> </tr> </table>	1101	0nnn	nnnn	nnnn								
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:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'n'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write to PC</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
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 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]				
Operation:	1 → f<b>				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1000</td> <td>bbba</td> <td>ffff</td> <td>ffff</td> </tr> </table>	1000	bbba	ffff	ffff
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 ≤ 95 (5Fh). See <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> 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

# PIC18F66K80 FAMILY

---

<b>BTFSC</b>	<b>Bit Test File, Skip if Clear</b>												
Syntax:	BTFSC f, b {,a}												
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]												
Operation:	skip if (f<b>) = 0												
Status Affected:	None												
Encoding:	1011 bbba ffff ffff												
Description:	If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.												
Words:	1												
Cycles:	1(2)												
	<b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.												
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read register 'f'	Process Data	No operation										
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4										
No operation	No operation	No operation	No operation										
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
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)

<b>BTFSS</b>	<b>Bit Test File, Skip if Set</b>												
Syntax:	BTFSS f, b {,a}												
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]												
Operation:	skip if (f<b>) = 1												
Status Affected:	None												
Encoding:	1010 bbba ffff ffff												
Description:	If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.												
Words:	1												
Cycles:	1(2)												
	<b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.												
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read register 'f'	Process Data	No operation										
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4										
No operation	No operation	No operation	No operation										
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
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)

# PIC18F66K80 FAMILY

BTG	Bit Toggle f								
Syntax:	BTG f, b {,a}								
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 $a \in [0,1]$								
Operation:	$(\overline{f<b>}) \rightarrow f<b>$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0111</td><td>bbba</td><td>ffff</td><td>ffff</td></tr> </table>	0111	bbba	ffff	ffff				
0111	bbba	ffff	ffff						
Description:	<p>Bit 'b' in data memory location 'f' is inverted.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0100</td><td>nnnn</td><td>nnnn</td></tr> </table>	1110	0100	nnnn	nnnn								
1110	0100	nnnn	nnnn										
Description:	<p>If the Overflow bit is '1', then the program will branch.</p> <p>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 <math>PC + 2 + 2n</math>. This instruction is then a two-cycle instruction.</p>												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:													
If Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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)

# PIC18F66K80 FAMILY

---



---

## BZ Branch if Zero

Syntax:	BZ n				
Operands:	$-128 \leq n \leq 127$				
Operation:	if Zero bit is '1', $(PC) + 2 + 2n \rightarrow PC$				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0000</td><td>nnnn</td><td>nnnn</td></tr> </table>	1110	0000	nnnn	nnnn
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 PC = 1; address (Jump)

If Zero PC = 0;

PC = address (HERE + 2)

## CALL Subroutine Call

Syntax:	CALL k {s}								
Operands:	$0 \leq k \leq 1048575$ $s \in [0,1]$								
Operation:	$(PC) + 4 \rightarrow TOS$ , $k \rightarrow PC<20:1>;$ if $s = 1$ $(W) \rightarrow WS$ , $(STATUS) \rightarrow STATUS$ , $(BSR) \rightarrow BSRS$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>110s</td><td>k<sub>7</sub>kkk</td><td>kkkk<sub>0</sub></td></tr> <tr><td>1111</td><td>k<sub>19</sub>kkk</td><td>kkkk</td><td>kkkk<sub>8</sub></td></tr> </table>	1110	110s	k <sub>7</sub> kkk	kkkk <sub>0</sub>	1111	k <sub>19</sub> kkk	kkkk	kkkk <sub>8</sub>
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, STATUS and BSRS. If ' $s$ ' = 0, no update occurs (default). Then, the 20-bit value ' $k$ ' is loaded into $PC<20:1>$ . CALL is a two-cycle instruction.								

Words: 2

Cycles: 2

### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	Push PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: HERE CALL THERE,1

### Before Instruction

PC = address (HERE)

### After Instruction

PC = address (THERE)

TOS = address (HERE + 4)

WS = W

BSRS = BSR

STATUS = STATUS

# PIC18F66K80 FAMILY

---

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:	<table border="1"><tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr></table>	0110	101a	ffff	ffff				
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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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

CLRWD	Clear Watchdog Timer								
Syntax:	CLRWD								
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:	<table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0100</td></tr></table>	0000	0000	0000	0100				
0000	0000	0000	0100						
Description:	CLRWD 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:	<table border="1"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>No operation</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	No operation	Process Data	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	Process Data	No operation						

Example: CLRWD

Before Instruction	
WDT Counter	= ?
After Instruction	
WDT Counter	= 00h
WDT Postscaler	= 0
TO	= 1
PD	= 1

# PIC18F66K80 FAMILY

---

COMF	Complement f								
Syntax:	COMF 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:	0001 11da ffff ffff								
Description:	The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: COMF REG, 0, 0

Before Instruction  
REG = 13h

After Instruction  
REG = 13h  
W = ECCh

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.
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 ≠ W;  
PC = Address (NEQUAL)

CPFSGT	Compare f with W, Skip if f > W			
Syntax:	CPFSGT f {,a}			
Operands:	0 ≤ f ≤ 255 a ∈ [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 ≤ 95 (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.			
Words:	1			
Cycles:	1(2)			
<b>Note:</b>	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			
<b>Example:</b>	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 ≤ f ≤ 255 a ∈ [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)			
<b>Note:</b>	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			

**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)

# PIC18F66K80 FAMILY

---



---

DAW                    Decimal Adjust W Register									
Syntax:	DAW								
Operands:	None								
Operation:	<p>If <math>[W&lt;3:0&gt; &gt; 9]</math> or <math>[DC = 1]</math>, then  <math>(W&lt;3:0&gt;) + 6 \rightarrow W&lt;3:0&gt;;</math>  else  <math>(W&lt;3:0&gt;) \rightarrow W&lt;3:0&gt;</math></p> <p>If <math>[W&lt;7:4&gt; &gt; 9]</math> or <math>[C = 1]</math>, then  <math>(W&lt;7:4&gt;) + 6 \rightarrow W&lt;7:4&gt;;</math>  C = 1  else  <math>(W&lt;7:4&gt;) \rightarrow W&lt;7:4&gt;</math></p>								
Status Affected:	C								
Encoding:	<table border="1"> <tr> <td>0000</td><td>0000</td><td>0000</td><td>0111</td></tr> </table>	0000	0000	0000	0111				
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:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register W</td><td>Process Data</td><td>Write W</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register W	Process Data	Write W
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:	<table border="1"> <tr> <td>0000</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table>	0000	01da	ffff	ffff				
0000	01da	ffff	ffff						
Description:	<p>Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example:        DECF CNT, 1, 0

Before Instruction

CNT	=	01h
Z	=	0

After Instruction

CNT	=	00h
Z	=	1

# PIC18F66K80 FAMILY

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:	<p>The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).</p> <p>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.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>															
Words:	1															
Cycles:	1(2)															
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.															
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination				
Q1	Q2	Q3	Q4													
Decode	Read register 'f'	Process Data	Write to destination													
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
No operation	No operation	No operation	No operation													
<b>Example:</b>	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 ≠ 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 ≠ 0															
Status Affected:	None															
Encoding:	0100	11da	ffff	ffff												
Description:	<p>The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).</p> <p>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.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>															
Words:	1															
Cycles:	1(2)															
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.															
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination				
Q1	Q2	Q3	Q4													
Decode	Read register 'f'	Process Data	Write to destination													
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
No operation	No operation	No operation	No operation													
<b>Example:</b>	HERE	DCFSNZ	TEMP, 1, 0													
	ZERO	:														
	NZERO	:														

Before Instruction  
 TEMP = ?  
 After Instruction  
 TEMP = TEMP - 1,  
 If TEMP = 0;  
 PC = Address (ZERO)  
 If TEMP ≠ 0;  
 PC = Address (NZERO)

# PIC18F66K80 FAMILY

---



---

GOTO	Unconditional Branch												
Syntax:	GOTO k												
Operands:	$0 \leq k \leq 1048575$												
Operation:	$k \rightarrow PC<20:1>$												
Status Affected:	None												
Encoding:													
1st word ( $k<7:0>$ )	1110												
2nd word ( $k<19:8>$ )	1111 $k_{19}kkk$												
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:													
	<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 '<math>k&lt;7:0&gt;</math>',</td> <td>No operation</td> <td>Read literal '<math>k&lt;19:8&gt;</math>, Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table>	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
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' (default). 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 29.2.3 "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: INCF CNT, 1, 0

Before Instruction

CNT = FFh  
Z = 0  
C = ?  
DC = ?

After Instruction

CNT = 00h  
Z = 1  
C = 1  
DC = 1

# PIC18F66K80 FAMILY

INCSZ	Increment f, Skip if 0
Syntax:	INCSZ 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'. (default)  If the result is '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.
Words:	1
Cycles:	1(2)
<b>Note:</b>	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 INCSZ CNT, 1, 0  
NZERO :  
ZERO :

Before Instruction

PC = Address (HERE)

After Instruction

CNT = CNT + 1

If CNT = 0;

PC = Address (ZERO)

If CNT ≠ 0;

PC = Address (NZERO)

INFSNZ	Increment f, Skip if Not 0
Syntax:	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 ≠ 0
Status Affected:	None
Encoding:	0100 10da ffff ffff
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).  If the result is not '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.
Words:	1
Cycles:	1(2)
<b>Note:</b>	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 ≠ 0;

PC = Address (NZERO)

If REG = 0;

PC = Address (ZERO)

# PIC18F66K80 FAMILY

---



---

IORLW	Inclusive OR Literal with W								
Syntax:	IORLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	(W) .OR. k → W								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1001</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1001	kkkk	kkkk				
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:	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center;">Q1</th><th style="text-align: center;">Q2</th><th style="text-align: center;">Q3</th><th style="text-align: center;">Q4</th></tr> <tr> <td style="text-align: center;">Decode</td><td style="text-align: center;">Read literal 'k'</td><td style="text-align: center;">Process Data</td><td style="text-align: center;">Write to W</td></tr> </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: 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) → dest								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0001</td><td>00da</td><td>ffff</td><td>ffff</td></tr> </table>	0001	00da	ffff	ffff				
0001	00da	ffff	ffff						
Description:	Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center;">Q1</th><th style="text-align: center;">Q2</th><th style="text-align: center;">Q3</th><th style="text-align: center;">Q4</th></tr> <tr> <td style="text-align: center;">Decode</td><td style="text-align: center;">Read register 'f'</td><td style="text-align: center;">Process Data</td><td style="text-align: center;">Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: IORWF RESULT, 0, 1

Before Instruction

RESULT = 13h  
W = 91h

After Instruction

RESULT = 13h  
W = 93h

# PIC18F66K80 FAMILY

---

LFSR	Load FSR															
Syntax:	LFSR f, k															
Operands:	0 ≤ f ≤ 2 0 ≤ k ≤ 4095															
Operation:	k → FSRf															
Status Affected:	None															
Encoding:	1110 1111	1110 0000	00ff k <sub>7</sub> kkk	k <sub>11</sub> kkk kkkk												
Description:	The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.															
Words:	2															
Cycles:	2															
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'k' MSB</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write literal 'k' MSB to FSRfH</td> </tr> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'k' LSB</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write literal 'k' to FSRfL</td> </tr> </tbody> </table>				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
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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]											
Operation:	f → dest											
Status Affected:	N, Z											
Encoding:	0101	00da	ffff	ffff								
Description:	The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank.											
	If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read register 'f'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write W</td> </tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write W
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

# PIC18F66K80 FAMILY

---



---

<b>MOVFF</b>	<b>Move f to f</b>												
Syntax:	MOVFF f <sub>s</sub> ,f <sub>d</sub>												
Operands:	0 ≤ f <sub>s</sub> ≤ 4095 0 ≤ f <sub>d</sub> ≤ 4095												
Operation:	(f <sub>s</sub> ) → f <sub>d</sub>												
Status Affected:	None												
Encoding:													
1st word (source)	1100 ffff ffff ffff <sub>s</sub>												
2nd word (destin.)	1111 ffff ffff ffff <sub>d</sub>												
Description:	<p>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.</p> <p>Either source or destination can be W (a useful special situation).</p> <p>MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).</p> <p>The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register</p>												
Words:	2												
Cycles:	2												
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 register 'f' (src)</td> <td>Process Data</td> <td>No operation</td> </tr> <tr> <td>Decode</td> <td>No operation No dummy read</td> <td>No operation</td> <td>Write register 'f' (dest)</td> </tr> </tbody> </table>	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)
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

<b>MOVLB</b>	<b>Move Literal to Low Nibble in BSR</b>								
Syntax:	MOVLW k								
Operands:	0 ≤ k ≤ 255								
Operation:	k → BSR								
Status Affected:	None								
Encoding:									
	0000 0001 kkkk kkkk								
Description:	<p>The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR&lt;7:4&gt; always remains '0' regardless of the value of k<sub>7:k<sub>4</sub></sub>.</p>								
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 literal 'k' to BSR</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR
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

# PIC18F66K80 FAMILY

---

## MOVLW

### Move Literal to W

Syntax:	MOVLW k			
Operands:	0 ≤ k ≤ 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 ≤ f ≤ 255			
a ∈ [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 29.2.3 “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

# PIC18F66K80 FAMILY

---

MULLW	Multiply Literal with W								
Syntax:	MULLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	$(W) \times k \rightarrow PRODH:PRODL$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1101</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	0000	1101	kkkk	kkkk				
0000	1101	kkkk	kkkk						
Description:	<p>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.</p> <p>None of the Status flags are affected.</p> <p>Note that neither Overflow nor Carry is possible in this operation. A Zero result is possible but not detected.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'k'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write registers PRODH: PRODL</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL
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:	$(W) \times (f) \rightarrow PRODH:PRODL$				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>001a</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0000	001a	ffff	ffff
0000	001a	ffff	ffff		
Description:	<p>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.</p> <p>None of the Status flags are affected.</p> <p>Note that neither Overflow nor Carry is possible in this operation. A Zero result is possible but not detected.</p>				
Words:	1				
Cycles:	1				
Q Cycle Activity:	<p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>				

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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0110</td> <td>110a</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0110	110a	ffff	ffff				
0110	110a	ffff	ffff						
Description:	<p>Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	No operation	No operation						
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0000</td> </tr> <tr> <td>1111</td> <td>xxxx</td> <td>xxxx</td> <td>xxxx</td> </tr> </table>	0000	0000	0000	0000	1111	xxxx	xxxx	xxxx
0000	0000	0000	0000						
1111	xxxx	xxxx	xxxx						
Description:	No operation.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	No operation	No operation						
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	No operation	No operation						

Example:

None.

# PIC18F66K80 FAMILY

---



---

POP	Pop Top of Return Stack								
Syntax:	POP								
Operands:	None								
Operation:	(TOS) → bit bucket								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0110</td></tr> </table>	0000	0000	0000	0110				
0000	0000	0000	0110						
Description:	<p>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.</p> <p>This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>POP TOS value</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	POP TOS value	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	POP TOS value	No operation						

<u>Example:</u>	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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0101</td></tr> </table>	0000	0000	0000	0101				
0000	0000	0000	0101						
Description:	<p>The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack.</p> <p>This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>PUSH PC + 2 onto return stack</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	PUSH PC + 2 onto return stack	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	PUSH PC + 2 onto return stack	No operation	No operation						

<u>Example:</u>	PUSH	
		Before Instruction
	TOS	= 345Ah
	PC	= 0124h
	After Instruction	
	PC	= 0126h
	TOS	= 0126h
	Stack (1 level down)	= 345Ah

# PIC18F66K80 FAMILY

---

<b>RCALL</b>	<b>Relative Call</b>												
Syntax:	RCALL n												
Operands:	$-1024 \leq n \leq 1023$												
Operation:	$(PC) + 2 \rightarrow TOS,$ $(PC) + 2 + 2n \rightarrow PC$												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1101</td><td>1nnn</td><td>nnnn</td><td>nnnn</td></tr></table>	1101	1nnn	nnnn	nnnn								
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:	<table border="1" style="width: 100%;"><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Read literal 'n' PUSH PC to stack</td><td>Process Data</td><td>Write to PC</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></tbody></table>	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
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)

<b>RESET</b>	<b>Reset</b>								
Syntax:	RESET								
Operands:	None								
Operation:	Reset all registers and flags that are affected by a MCLR Reset.								
Status Affected:	All								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>1111</td><td>1111</td></tr></table>	0000	0000	1111	1111				
0000	0000	1111	1111						
Description:	This instruction provides a way to execute a MCLR Reset in software.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%;"><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Start reset</td><td>No operation</td><td>No operation</td></tr></tbody></table>	Q1	Q2	Q3	Q4	Decode	Start reset	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	Start reset	No operation	No operation						

Example: RESET

After Instruction

Registers = Reset Value  
Flags\* = Reset Value

# PIC18F66K80 FAMILY

---

RETFIE	Return from Interrupt												
Syntax:	RETFIE {s}												
Operands:	$s \in [0,1]$												
Operation:	$(TOS) \rightarrow PC$ , $1 \rightarrow GIE/GIEH$ or $PEIE/GIEL$ ; if $s = 1$ , $(WS) \rightarrow W$ , $(STATUSS) \rightarrow STATUS$ , $(BSRS) \rightarrow BSR$ , $PCLATU$ , $PCLATH$ are unchanged												
Status Affected:	GIE/GIEH, PEIE/GIEL.												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0001</td> <td>000s</td> </tr> </table>	0000	0000	0001	000s								
0000	0000	0001	000s										
Description:	Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low-priority Global Interrupt Enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers W, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">POP PC from stack Set GIEH or GIEL</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table>	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
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	=	STATUSS
GIE/GIEH, PEIE/GIEL	=	1

RETLW	Return Literal to W												
Syntax:	RETLW k												
Operands:	$0 \leq k \leq 255$												
Operation:	$k \rightarrow W$ , $(TOS) \rightarrow PC$ , $PCLATU$ , $PCLATH$ are unchanged												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1100</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	0000	1100	kkkk	kkkk								
0000	1100	kkkk	kkkk										
Description:	<p>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.</p> <p>Words: 1</p> <p>Cycles: 2</p> <p>Q Cycle Activity:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'k'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">POP PC from stack, write to W</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table>	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
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
  
```

# PIC18F66K80 FAMILY

RETURN	Return from Subroutine	RLCF	Rotate Left f through Carry												
Syntax:	RETURN {s}	Syntax:	RLCF f {,d {,a}}												
Operands:	s ∈ [0,1]	Operands:	0 ≤ f ≤ 255												
Operation:	(TOS) → PC; if s = 1, (WS) → W, (STATUS) → STATUS, (BSRS) → BSR, PCLATU, PCLATH are unchanged		d ∈ [0,1] a ∈ [0,1]												
Status Affected:	None	Operation:	(f<n>) → dest<n + 1>, (f<7>) → C, (C) → dest<0>												
Encoding:	<table border="1"><tr><td>0000</td><td>0000</td><td>0001</td><td>001s</td></tr></table>	0000	0000	0001	001s	Status Affected:	C, N, Z								
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 (default).	Encoding:	<table border="1"><tr><td>0011</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0011	01da	ffff	ffff								
0011	01da	ffff	ffff												
Words:	1	Description:	The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default).												
Cycles:	2		If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.												
Q Cycle Activity:			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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.												
	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>Process Data</td><td>POP PC from stack</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	Process Data	POP PC from stack	No operation	No operation	No operation	No operation		
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

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

# PIC18F66K80 FAMILY

---



---

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:	$(f < n >) \rightarrow \text{dest} < n + 1 >$ , $(f < 7 >) \rightarrow \text{dest} < 0 >$								
Status Affected:	N, Z								
Encoding:	<table border="1"><tr><td>0100</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0100	01da	ffff	ffff				
0100	01da	ffff	ffff						
Description:	The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default).  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.								
									
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

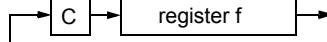
Example:      RLNCF      REG, 1, 0

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:	$(f < n >) \rightarrow \text{dest} < n - 1 >$ , $(f < 0 >) \rightarrow C$ , $(C) \rightarrow \text{dest} < 7 >$								
Status Affected:	C, N, Z								
Encoding:	<table border="1"><tr><td>0011</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>	0011	00da	ffff	ffff				
0011	00da	ffff	ffff						
Description:	The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).  If 'a' is '0', the Access Bank 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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.								
									
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example:      RRCF      REG, 0, 0

Before Instruction

REG = 1110 0110  
C = 0

After Instruction

REG = 1110 0110  
W = 0111 0011  
C = 0

# PIC18F66K80 FAMILY

RRNCF	Rotate Right f (No Carry)								
Syntax:	RRNCF f {,d {,a}}								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f<n>) → dest<n – 1>, (f<0>) → dest<7>								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0100</td><td>00da</td><td>ffff</td><td>ffff</td></tr> </table>	0100	00da	ffff	ffff				
0100	00da	ffff	ffff						
Description:	<p>The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).</p> <p>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.</p> <p>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 <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p> 								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th> </tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write register 'f'						

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 ≤ f ≤ 255 a ∈ [0,1]								
Operation:	FFh → f								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0110</td><td>100a</td><td>ffff</td><td>ffff</td></tr> </table>	0110	100a	ffff	ffff				
0110	100a	ffff	ffff						
Description:	<p>The contents of the specified register are set to FFh.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>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 <a href="#">Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th> </tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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

# PIC18F66K80 FAMILY

---

SLEEP	Enter Sleep Mode								
Syntax:	SLEEP								
Operands:	None								
Operation:	00h → WDT, 0 → WDT postscaler, 1 → $\overline{\text{TO}}$ , 0 → $\overline{\text{PD}}$								
Status Affected:	$\overline{\text{TO}}$ , $\overline{\text{PD}}$								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0011</td></tr> </table>	0000	0000	0000	0011				
0000	0000	0000	0011						
Description:	The Power-Down status bit ( $\overline{\text{PD}}$ ) is cleared. The Time-out status bit ( $\overline{\text{TO}}$ ) is set. The 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:	<table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>Process Data</td><td>Go to Sleep</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	Process Data	Go to Sleep
Q1	Q2	Q3	Q4						
Decode	No operation	Process Data	Go to Sleep						

Example: SLEEP

Before Instruction

$$\begin{array}{lcl} \overline{\text{TO}} & = & ? \\ \overline{\text{PD}} & = & ? \end{array}$$

After Instruction

$$\begin{array}{lcl} \overline{\text{TO}} & = & 1 \dagger \\ \overline{\text{PD}} & = & 0 \end{array}$$

† 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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0101</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table>	0101	01da	ffff	ffff
0101	01da	ffff	ffff		
Description:	<p>Subtract register 'f' and Carry flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>				

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

$$\begin{array}{lcl} \text{REG} & = & 3 \\ \text{W} & = & 2 \\ \text{C} & = & 1 \end{array}$$

After Instruction

$$\begin{array}{lcl} \text{REG} & = & \text{FF} \\ \text{W} & = & 2 \\ \text{C} & = & 0 \\ \text{Z} & = & 0 \\ \text{N} & = & 1 \quad ; \text{result is negative} \end{array}$$

Example 2: SUBFWB REG, 0, 0

Before Instruction

$$\begin{array}{lcl} \text{REG} & = & 2 \\ \text{W} & = & 5 \\ \text{C} & = & 1 \end{array}$$

After Instruction

$$\begin{array}{lcl} \text{REG} & = & 2 \\ \text{W} & = & 3 \\ \text{C} & = & 1 \\ \text{Z} & = & 0 \\ \text{N} & = & 0 \quad ; \text{result is positive} \end{array}$$

Example 3: SUBFWB REG, 1, 0

Before Instruction

$$\begin{array}{lcl} \text{REG} & = & 1 \\ \text{W} & = & 2 \\ \text{C} & = & 0 \end{array}$$

After Instruction

$$\begin{array}{lcl} \text{REG} & = & 0 \\ \text{W} & = & 2 \\ \text{C} & = & 1 \\ \text{Z} & = & 1 \\ \text{N} & = & 0 \quad ; \text{result is zero} \end{array}$$

# PIC18F66K80 FAMILY

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:	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr> </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 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' (default).											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4									
Decode	Read register 'f'	Process Data	Write to destination									

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

# PIC18F66K80 FAMILY

---

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:	$(f) - (W) - (\bar{C}) \rightarrow \text{dest}$								
Status Affected:	N, OV, C, DC, Z								
Encoding:	<table border="1"><tr><td>0101</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0101	10da	ffff	ffff				
0101	10da	ffff	ffff						
Description:	<p>Subtract W and the Carry flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

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)
W	=	0Eh	; [2's comp] (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:	<table border="1"><tr><td>0011</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0011	10da	ffff	ffff				
0011	10da	ffff	ffff						
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: SWAPF REG, 1, 0

Before Instruction

REG	=	53h
-----	---	-----

After Instruction

REG	=	35h
-----	---	-----

# PIC18F66K80 FAMILY

---

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:	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>10nn nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table>	0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*								
0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*										
Description:	<p>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.</p> <p>The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.</p> <p>TBLPTR&lt;0&gt; = 0: Least Significant Byte of Program Memory Word            TBLPTR&lt;0&gt; = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLRD instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> <li>• no change</li> <li>• post-increment</li> <li>• post-decrement</li> <li>• pre-increment</li> </ul>												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> <tr> <td>No operation</td> <td>No operation (Read Program Memory)</td> <td>No operation</td> <td>No operation (Write TABLAT)</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)				
Q1	Q2	Q3	Q4										
Decode	No operation	No operation	No operation										
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)										

TBLRD	Table Read (Continued)
<u>Example 1:</u>	TBLRD *+ ;  Before Instruction TABLAT = 55h TBLPTR = 00A356h MEMORY(00A356h) = 34h  After Instruction TABLAT = 34h TBLPTR = 00A357h
<u>Example 2:</u>	TBLRD +* ;  Before Instruction TABLAT = AAh TBLPTR = 01A357h MEMORY(01A357h) = 12h MEMORY(01A358h) = 34h  After Instruction TABLAT = 34h TBLPTR = 01A358h

# PIC18F66K80 FAMILY

---

TBLWT	Table Write				
Syntax:	TBLWT ( *, *+; *-; +* )				
Operands:	None				
Operation:	<p>if TBLWT*,          (TABLAT) → Holding Register;          TBLPTR – No Change</p> <p>if TBLWT*+,          (TABLAT) → Holding Register;          (TBLPTR) + 1 → TBLPTR</p> <p>if TBLWT*-,          (TABLAT) → Holding Register;          (TBLPTR) – 1 → TBLPTR</p> <p>if TBLWT+*,          (TBLPTR) + 1 → TBLPTR;          (TABLAT) → Holding Register</p>				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>11nn nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table>	0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*
0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*		
Description:	<p>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 <a href="#">Section 6.0 “Memory Organization”</a> for additional details on programming Flash memory.)</p> <p>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.</p> <p>TBLPTR[0] = 0: Least Significant Byte of Program Memory Word</p> <p>TBLPTR[0] = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLWT instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> <li>• no change</li> <li>• post-increment</li> <li>• post-decrement</li> <li>• pre-increment</li> </ul>				
Words:	1				
Cycles:	2				
Q Cycle Activity:					

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT	Table Write (Continued)												
Example 1:	TBLWT *+;												
	Before Instruction												
	<table> <tr> <td>TABLAT</td> <td>=</td> <td>55h</td> </tr> <tr> <td>TBLPTR</td> <td>=</td> <td>00A356h</td> </tr> <tr> <td>HOLDING REGISTER (00A356h)</td> <td>=</td> <td>FFh</td> </tr> </table>	TABLAT	=	55h	TBLPTR	=	00A356h	HOLDING REGISTER (00A356h)	=	FFh			
TABLAT	=	55h											
TBLPTR	=	00A356h											
HOLDING REGISTER (00A356h)	=	FFh											
	After Instructions (table write completion)												
	<table> <tr> <td>TABLAT</td> <td>=</td> <td>55h</td> </tr> <tr> <td>TBLPTR</td> <td>=</td> <td>00A357h</td> </tr> <tr> <td>HOLDING REGISTER (00A356h)</td> <td>=</td> <td>55h</td> </tr> </table>	TABLAT	=	55h	TBLPTR	=	00A357h	HOLDING REGISTER (00A356h)	=	55h			
TABLAT	=	55h											
TBLPTR	=	00A357h											
HOLDING REGISTER (00A356h)	=	55h											
Example 2:	TBLWT +*;												
	Before Instruction												
	<table> <tr> <td>TABLAT</td> <td>=</td> <td>34h</td> </tr> <tr> <td>TBLPTR</td> <td>=</td> <td>01389Ah</td> </tr> <tr> <td>HOLDING REGISTER (01389Ah)</td> <td>=</td> <td>FFh</td> </tr> <tr> <td>HOLDING REGISTER (01389Bh)</td> <td>=</td> <td>FFh</td> </tr> </table>	TABLAT	=	34h	TBLPTR	=	01389Ah	HOLDING REGISTER (01389Ah)	=	FFh	HOLDING REGISTER (01389Bh)	=	FFh
TABLAT	=	34h											
TBLPTR	=	01389Ah											
HOLDING REGISTER (01389Ah)	=	FFh											
HOLDING REGISTER (01389Bh)	=	FFh											
	After Instruction (table write completion)												
	<table> <tr> <td>TABLAT</td> <td>=</td> <td>34h</td> </tr> <tr> <td>TBLPTR</td> <td>=</td> <td>01389Bh</td> </tr> <tr> <td>HOLDING REGISTER (01389Ah)</td> <td>=</td> <td>FFh</td> </tr> <tr> <td>HOLDING REGISTER (01389Bh)</td> <td>=</td> <td>34h</td> </tr> </table>	TABLAT	=	34h	TBLPTR	=	01389Bh	HOLDING REGISTER (01389Ah)	=	FFh	HOLDING REGISTER (01389Bh)	=	34h
TABLAT	=	34h											
TBLPTR	=	01389Bh											
HOLDING REGISTER (01389Ah)	=	FFh											
HOLDING REGISTER (01389Bh)	=	34h											

# PIC18F66K80 FAMILY

---

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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0110</td><td>011a</td><td>ffff</td><td>ffff</td></tr></table>	0110	011a	ffff	ffff
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 <a href="#">Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</a> for details.				
Words:	1				
Cycles:	1(2) <b>Note:</b> 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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>1010</td><td>kkkk</td><td>kkkk</td></tr></table>	0000	1010	kkkk	kkkk				
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:									
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr> </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:      XORLW 0AFh

Before Instruction  
W = B5h  
After Instruction  
W = 1Ah

# PIC18F66K80 FAMILY

---

---

## 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)  $\rightarrow$  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' (default).

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 29.2.3 "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

## 29.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, the PIC18F66K80 family of devices also provides 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 for many of the standard PIC18 instructions.

The additional features of the extended instruction set are enabled by default on unprogrammed devices. Users must properly set or clear the XINST Configuration bit during programming to enable or disable these features.

The instructions in the extended set 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 29-3](#). Detailed descriptions are provided in [Section 29.2.2 “Extended Instruction Set”](#). The opcode field descriptions in [Table 29-1](#) (page 484) 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 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.

### 29.2.1 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 29.2.3.1 “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 29-3: 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	1111	ffff	ffff	ffff	None
PUSHL k	Store Literal at FSR2, Decrement FSR2	1	1110	1011	1zzz	zzzz	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

# PIC18F66K80 FAMILY

---

## 29.2.2 EXTENDED INSTRUCTION SET

<b>ADDFSR</b>	<b>Add Literal to FSR</b>	<b>ADDULNK</b>	<b>Add Literal to FSR2 and Return</b>								
Syntax:	ADDFSR f, k	Syntax:	ADDULNK k								
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$	Operands:	$0 \leq k \leq 63$								
Operation:	$FSR(f) + k \rightarrow FSR(f)$	Operation:	$FSR2 + k \rightarrow FSR2,$ (TOS) $\rightarrow PC$								
Status Affected:	None	Status Affected:	None								
Encoding:	<table border="1"><tr><td>1110</td><td>1000</td><td>ffkk</td><td>kkkk</td></tr></table>	1110	1000	ffkk	kkkk	Encoding:	<table border="1"><tr><td>1110</td><td>1000</td><td>11kk</td><td>kkkk</td></tr></table>	1110	1000	11kk	kkkk
1110	1000	ffkk	kkkk								
1110	1000	11kk	kkkk								
Description:	The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.	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.								
Words:	1	Words:	1								
Cycles:	1	Cycles:	2								
Q Cycle Activity:		Q Cycle Activity:	The instruction takes two cycles to execute; a NOP is performed during the second cycle.								
Q1            Q2            Q3            Q4		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

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 format then becomes: {label} instruction argument(s).

# PIC18F66K80 FAMILY

<b>CALLW</b>	<b>Subroutine Call Using WREG</b>												
Syntax:	CALLW												
Operands:	None												
Operation:	(PC + 2) → TOS, (W) → PCL, (PCLATH) → PCH, (PCLATU) → PCU												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0001</td><td>0100</td></tr> </table>	0000	0000	0001	0100								
0000	0000	0001	0100										
Description	<p>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.</p> <p>Unlike CALL, there is no option to update W, STATUS or BSR.</p>												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read WREG</td> <td style="text-align: center;">Push PC to stack</td> <td style="text-align: center;">No operation</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read WREG	Push PC to stack	No operation				
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

<b>MOVSF</b>	<b>Move Indexed to f</b>
--------------	--------------------------

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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>1011</td><td>0zzz</td><td>zzzz<sub>s</sub></td></tr> <tr><td>1111</td><td>ffff</td><td>ffff</td><td>ffff<sub>d</sub></td></tr> </table>	1110	1011	0zzz	zzzz <sub>s</sub>	1111	ffff	ffff	ffff <sub>d</sub>
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

# PIC18F66K80 FAMILY

---

<b>MOVSS</b>	<b>Move Indexed to Indexed</b>	<b>PUSHL</b>	<b>Store Literal at FSR2, Decrement FSR2</b>								
Syntax:	MOVSS [z <sub>s</sub> ], [z <sub>d</sub> ]	Syntax:	PUSHL k								
Operands:	0 ≤ z <sub>s</sub> ≤ 127 0 ≤ z <sub>d</sub> ≤ 127	Operands:	0 ≤ k ≤ 255								
Operation:	((FSR2) + z <sub>s</sub> ) → ((FSR2) + z <sub>d</sub> )	Operation:	k → (FSR2), FSR2 – 1 → FSR2								
Status Affected:	None	Status Affected:	None								
Encoding:		Encoding:									
1st word (source)	1110 1011 lzzz zzzz <sub>s</sub>	1111 1010 kkkk kkkk									
2nd word (dest.)	1111 xxxx xzzz zzzz <sub>d</sub>										
Description	<p>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).</p> <p>The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p> <p>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.</p>	<p>The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation.</p> <p>This instruction allows users to push values onto a software stack.</p>									
Words:	2	Words:	1								
Cycles:	2	Cycles:	1								
Q Cycle Activity:		Q Cycle Activity:									
		<table border="1"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read 'k'</td> <td>Process data</td> <td>Write to destination</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read 'k'	Process data	Write to destination	
Q1	Q2	Q3	Q4								
Decode	Read 'k'	Process data	Write to destination								

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

Example: PUSHL 08h

Before Instruction

FSR2:H:FSR2L	=	01ECh
Memory (01ECh)	=	00h

After Instruction

FSR2:H:FSR2L	=	01EBh
Memory (01ECh)	=	08h

# PIC18F66K80 FAMILY

---

<b>SUBFSR</b>	<b>Subtract Literal from FSR</b>								
Syntax:	SUBFSR f, k								
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$								
Operation:	$\text{FSR}f - k \rightarrow \text{FSR}f$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1001</td><td>ffkk</td><td>kkkk</td></tr></table>	1110	1001	ffkk	kkkk				
1110	1001	ffkk	kkkk						
Description:	The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: SUBFSR 2, 23h

Before Instruction

FSR2 = 03FFh

After Instruction

FSR2 = 03DCh

<b>SUBULNK</b>	<b>Subtract Literal from FSR2 and Return</b>				
Syntax:	SUBULNK k				
Operands:	$0 \leq k \leq 63$				
Operation:	$\text{FSR}2 - k \rightarrow \text{FSR}2$ , (TOS) $\rightarrow \text{PC}$				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1001</td><td>11kk</td><td>kkkk</td></tr></table>	1110	1001	11kk	kkkk
1110	1001	11kk	kkkk		
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)

# PIC18F66K80 FAMILY

---

## 29.2.3 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 ([Section 6.6.1 “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 29.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”](#)).

Although the Indexed Literal Offset 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.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset 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.

### 29.2.3.1 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 the brackets, will generate an error in the MPASM™ Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, 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.

## 29.2.4 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 PIC18F66K80 family, 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.

# PIC18F66K80 FAMILY

<b>ADDWF</b>	<b>ADD W to Indexed (Indexed Literal Offset mode)</b>								
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' (default).								
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 'k'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read 'k'	Process Data	Write to destination
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

<b>BSF</b>	<b>Bit Set Indexed (Indexed Literal Offset mode)</b>								
Syntax:	BSF [K], b								
Operands:	$0 \leq f \leq 95$ $0 \leq b \leq 7$								
Operation:	$1 \rightarrow ((FSR2) + k)<b>$								
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:	<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 register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: BSF [FLAG\_OFST], 7

Before Instruction	
FLAG_OFST	= 0Ah
FSR2	= 0A00h
Contents of 0A0Ah	= 55h
After Instruction	
Contents of 0A0Ah	= D5h

<b>SETF</b>	<b>Set Indexed (Indexed Literal Offset mode)</b>								
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:	<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 'k'</td> <td>Process Data</td> <td>Write register</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read 'k'	Process Data	Write register
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

# PIC18F66K80 FAMILY

---

## 29.2.5 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 for the PIC18F66K80 family. 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. 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.

## 30.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers and dsPIC® digital signal controllers are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C® for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/  
MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for  
Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards,  
Evaluation Kits, and Starter Kits

## 30.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - In-Circuit Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - 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.

# PIC18F66K80 FAMILY

---

## 30.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

## 30.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, pre-processor, and one-step driver, and can run on multiple platforms.

## 30.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB 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

## 30.5 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

## 30.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB 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 device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

## 30.7 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 C Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 30.8 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® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 30.9 MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE).

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 30.10 PICkit 3 In-Circuit Debugger/Programmer and PICkit 3 Debug Express

The MPLAB PICkit 3 allows debugging and programming of PIC® and dsPIC® Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via an Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

# PIC18F66K80 FAMILY

---

## 30.11 PICkit 2 Development Programmer/Debugger and PICkit 2 Debug Express

The PICkit™ 2 Development Programmer/Debugger is a low-cost development tool with an easy to use interface for programming and debugging Microchip's Flash families of microcontrollers. The full featured Windows® programming interface supports baseline (PIC10F, PIC12F5xx, PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33, and PIC32 families of 8-bit, 16-bit, and 32-bit microcontrollers, and many Microchip Serial EEPROM products. With Microchip's powerful MPLAB Integrated Development Environment (IDE) the PICkit™ 2 enables in-circuit debugging on most PIC® microcontrollers. In-Circuit-Debugging runs, halts and single steps the program while the PIC microcontroller is embedded in the application. When halted at a breakpoint, the file registers can be examined and modified.

The PICkit 2 Debug Express include the PICkit 2, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

## 30.12 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 MMC card for file storage and data applications.

## 30.13 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page ([www.microchip.com](http://www.microchip.com)) for the complete list of demonstration, development and evaluation kits.

## 31.0 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 MCLR with respect to Vss.....	-0.3V to 9.0V
Voltage on any digital only I/O pin with respect to Vss (except VDD).....	-0.3V to 7.5V
Voltage on any combined digital and analog pin with respect to Vss (except VDD and MCLR).....	-0.3V to (VDD + 0.3V)
Voltage on VDD with respect to Vss (PIC18F66K80) .....	-0.3V to 7.5V
Voltage on VDD with respect to Vss (PIC18LF66K80).....	-0.3V to 3.66V
Total power dissipation ( <b>Note 1</b> ) .....	1W
Maximum current out of Vss pin .....	300 mA
Maximum current into VDD pin .....	250 mA
Input clamp current, $I_{IJK}$ ( $V_I < 0$ or $V_I > VDD$ ).....	$\pm 20$ mA
Output clamp current, $I_{OK}$ ( $V_O < 0$ or $V_O > VDD$ ) .....	$\pm 20$ mA
Maximum output current sunk by PORTA<7:6> and any PORTB and PORTC I/O pins.....	25 mA
Maximum output current sunk by any PORTD and PORTE I/O pins.....	8 mA
Maximum output current sunk by PORTA<5:0> and any PORTF and PORTG I/O pins.....	2 mA
Maximum output current sourced by PORTA<7:6> and any PORTB and PORTC I/O pins .....	25 mA
Maximum output current sourced by any PORTD, PORTE and PORTJ I/O pins .....	8 mA
Maximum output current sourced by PORTA<5:0> and any PORTF, PORTG and PORTH I/O pins .....	2 mA
Maximum current sunk by all ports combined.....	200 mA

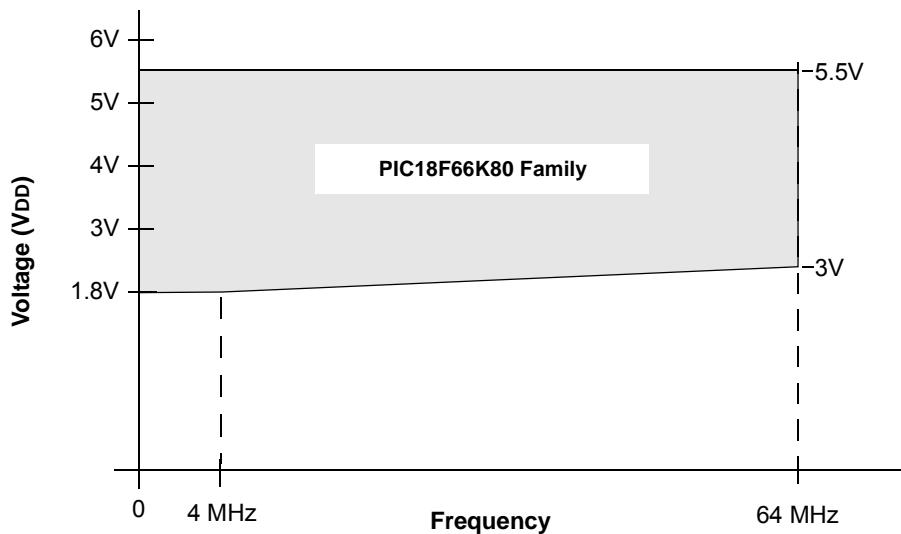
**Note 1:** Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$$

**† 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.

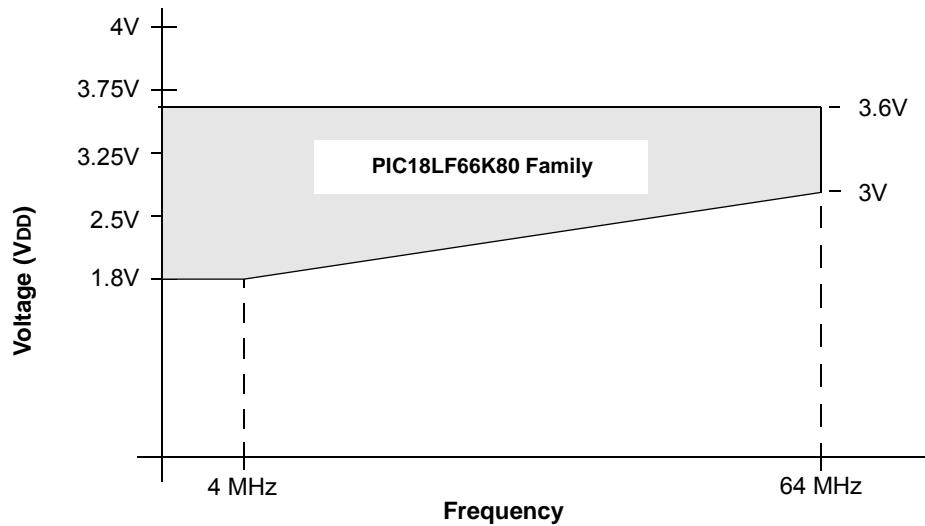
# PIC18F66K80 FAMILY

**FIGURE 31-1: VOLTAGE-FREQUENCY GRAPH,  
REGULATOR ENABLED (INDUSTRIAL/EXTENDED)<sup>(1)</sup>**



**Note 1:** For VDD values 1.8V to 3V,  $F_{MAX} = (VDD - 1.72)/0.02$  MHz.

**FIGURE 31-2: VOLTAGE-FREQUENCY GRAPH,  
REGULATOR DISABLED (INDUSTRIAL/EXTENDED)<sup>(1,2)</sup>**



**Note 1:** When the on-chip voltage regulator is disabled,  $V_{DD}$  must be maintained so that  $V_{DD} \leq 3.6V$ .  
**2:** For VDD values 1.8V to 3V,  $F_{MAX} = (VDD - 1.72)/0.02$  MHz.

# PIC18F66K80 FAMILY

## 31.1 DC Characteristics: Supply Voltage PIC18F66K80 Family (Industrial/Extended)

PIC18F66K80 Family (Industrial, Extended)			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended				
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
D001	VDD	<b>Supply Voltage</b>	1.8 1.8	— —	3.6 5.5	V V	For LF devices For F devices
D001C	AVDD	<b>Analog Supply Voltage</b>	VDD – 0.3	—	VDD + 0.3	V	
D001D	AVSS	<b>Analog Ground Potential</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 <a href="#">Section 5.3 “Power-on Reset (POR)”</a> for details
D004	SVDD	<b>VDD Rise Rate</b> to Ensure Internal Power-on Reset Signal	0.05	—	—	V/ms	See <a href="#">Section 5.3 “Power-on Reset (POR)”</a> for details
D005	BVDD	<b>Brown-out Reset Voltage (High, Medium and Low-Power mode)</b> BORV<1:0> = 11 <sup>(2)</sup> BORV<1:0> = 10 BORV<1:0> = 01 BORV<1:0> = 00	1.69 1.88 2.53 2.82	1.8 2.0 2.7 3.0	1.91 2.12 2.86 3.18	V V V V	

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

**2:** Device will operate normally until Brown-out Reset occurs, even though VDD may be below VDDMIN.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated)				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Power-Down Current (IPD)<sup>(1)</sup></b>						
PIC18LFXXX80	8	400	nA		-40°C	VDD = 1.8V <b>(Sleep mode)</b> Regulator Disabled
	13	500	nA		+25°C	
	35	750	nA		+60°C	
	218	980	nA		+85°C	
	3	6	µA		+125°C	
PIC18LFXXX80	14	500	nA		-40°C	VDD = 3.3V <b>(Sleep mode)</b> Regulator Disabled
	34	600	nA		+25°C	
	92	850	nA		+60°C	
	312	1250	nA		+85°C	
	4	8	µA		+125°C	
PIC18FXXX80	200	700	nA		-40°C	VDD = 3.3V <b>(Sleep mode)</b> Regulator Enabled
	230	800	nA		+25°C	
	320	1050	nA		+60°C	
	510	1500	nA		+85°C	
	5	9	µA		+125°C	
PIC18FXXX80	220	1000	nA		-40°C	VDD = 5V <b>(Sleep mode)</b> Regulator Enabled
	240	1000	nA		+25°C	
	340	1100	nA		+60°C	
	540	1580	nA		+85°C	
	5	10	µA		+125°C	

**Legend:** Shading of rows is to assist in readability of the table.

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** For LF devices, RETEN (CONFIG1L<0>) = 1.

**5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated)				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Supply Current (IDD)<sup>(2,3)</sup></b>						
PIC18LFXXK80		4	8	µA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled
		4	8	µA	+25°C	
		4	8	µA	+60°C	
		5	9	µA	+85°C	
		9	12	µA	+125°C	
PIC18LFXXK80		7	11	µA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled
		7	11	µA	+25°C	
		7	11	µA	+60°C	
		8	12	µA	+85°C	
		13	15	µA	+125°C	
PIC18FXXK80		51	150	µA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled
		70	150	µA	+25°C	
		75	150	µA	+60°C	
		80	170	µA	+85°C	
		88	190	µA	+125°C	
PIC18FXXK80		75	180	µA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled
		75	180	µA	+25°C	
		75	180	µA	+60°C	
		80	190	µA	+85°C	
		95	200	µA	+125°C	

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;
  - MCLR = VDD; WDT enabled/disabled as specified.
- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	274	600	μA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	Fosc = 1 MHz (RC_RUN mode, HF-INTOSC)
	274	600	μA	+25°C		
	274	600	μA	+60°C		
	280	650	μA	+85°C		
	290	700	μA	+125°C		
PIC18LFXXK80	410	820	μA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	410	820	μA	+25°C		
	410	820	μA	+60°C		
	420	840	μA	+85°C		
	430	990	μA	+125°C		
PIC18FXXK80	490	860	μA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	490	860	μA	+25°C		
	490	860	μA	+60°C		
	500	890	μA	+85°C		
	510	1060	μA	+125°C		
PIC18FXXK80	490	910	μA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	490	910	μA	+25°C		
	490	910	μA	+60°C		
	500	970	μA	+85°C		
	510	1125	μA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Supply Current (IDD) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	520	820	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	FOSC = 4 MHz (RC_RUN mode, HF-INTOSC)
	520	820	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	520	820	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	530	880	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	540	1000	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18LFXXK80	941	1600	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	941	1600	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	941	1600	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	950	1610	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	960	1800	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18FXXK80	981	1640	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	981	1640	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	981	1640	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	990	1650	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	1000	1900	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18FXXK80	1	2.2	$\text{mA}$	$-40^{\circ}\text{C}$	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	1	2.2	$\text{mA}$	$+25^{\circ}\text{C}$		
	1	2.2	$\text{mA}$	$+60^{\circ}\text{C}$		
	1	2.2	$\text{mA}$	$+85^{\circ}\text{C}$		
	1	2.2	$\text{mA}$	$+125^{\circ}\text{C}$		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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:
- $\underline{\text{OSC1}} = \text{external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; MCLR} = \text{VDD; WDT enabled/disabled as specified.}$
  - 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of  $-10^{\circ}\text{C}$  to  $+70^{\circ}\text{C}$ . Extended temperature crystals are available at a much higher cost.
  - 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
  - 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	880	1600	nA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	Fosc = 31 kHz (RC_IDLE mode, LF-INTOSC)
	880	1600	nA	+25°C		
	880	1600	nA	+60°C		
	1	2	μA	+85°C		
	5	10	μA	+125°C		
PIC18LFXXK80	1.6	5	μA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	1.6	5	μA	+25°C		
	1.6	5	μA	+60°C		
	2	6	μA	+85°C		
	7	12	μA	+125°C		
PIC18FXXK80	41	130	μA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	59	130	μA	+25°C		
	64	130	μA	+60°C		
	70	150	μA	+85°C		
	80	175	μA	+125°C		
PIC18FXXK80	53	160	μA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	62	160	μA	+25°C		
	70	160	μA	+60°C		
	85	170	μA	+85°C		
	100	180	μA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Supply Current (IDD) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	260	380	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	FOSC = 1 MHz (RC_IDLE mode, HF-INTOSC)
	260	380	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	260	380	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	270	390	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	280	420	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18LFXXK80	400	500	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	400	500	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	400	500	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	410	520	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	420	580	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18FXXK80	430	560	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	430	560	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	430	560	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	450	580	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	480	620	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18FXXK80	450	620	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	450	620	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	450	620	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	470	640	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	500	680	$\mu\text{A}$	$+125^{\circ}\text{C}$		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;
  - MCLR = VDD; WDT enabled/disabled as specified.
- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of  $-10^{\circ}\text{C}$  to  $+70^{\circ}\text{C}$ . Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	330	480	μA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	Fosc = 4 MHz (RC_IDLE mode, Internal HF-INTOSC)
	330	480	μA	+25°C		
	330	480	μA	+60°C		
	340	500	μA	+85°C		
	350	540	μA	+125°C		
PIC18LFXXK80	522	720	μA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	522	720	μA	+25°C		
	522	720	μA	+60°C		
	540	740	μA	+85°C		
	550	780	μA	+125°C		
PIC18FXXK80	540	760	μA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	540	760	μA	+25°C		
	540	760	μA	+60°C		
	560	780	μA	+85°C		
	580	810	μA	+125°C		
PIC18FXXK80	600	1250	μA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	600	1250	μA	+25°C		
	600	1250	μA	+60°C		
	610	1300	μA	+85°C		
	620	1340	μA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Supply Current (IDD) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	90	260	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	$\text{Fosc} = 1 \text{ MHz}$ <b>(PRI_RUN mode,</b> EC oscillator)
	90	260	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	90	260	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	100	270	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	110	300	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18LFXXK80	163	540	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	$\text{Fosc} = 1 \text{ MHz}$ <b>(PRI_RUN mode,</b> EC oscillator)
	163	540	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	163	540	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	170	560	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	180	600	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18FXXK80	201	560	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	$\text{Fosc} = 1 \text{ MHz}$ <b>(PRI_RUN mode,</b> EC oscillator)
	217	560	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	224	560	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	228	580	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	236	620	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18FXXK80	240	740	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5V <sup>(5)</sup> Regulator Enabled	$\text{Fosc} = 1 \text{ MHz}$ <b>(PRI_RUN mode,</b> EC oscillator)
	240	740	$\mu\text{A}$	$+25^{\circ}\text{C}$		
	240	740	$\mu\text{A}$	$+60^{\circ}\text{C}$		
	250	840	$\mu\text{A}$	$+85^{\circ}\text{C}$		
	260	940	$\mu\text{A}$	$+125^{\circ}\text{C}$		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;
  - MCLR** = VDD; WDT enabled/disabled as specified.
- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of  $-10^{\circ}\text{C}$  to  $+70^{\circ}\text{C}$ . Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	270	600	μA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	Fosc = 4 MHz (PRI_RUN mode, EC oscillator)
	270	600	μA	+25°C		
	270	600	μA	+60°C		
	300	700	μA	+85°C		
	320	850	μA	+125°C		
PIC18LFXXK80	540	1000	μA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	540	1000	μA	+25°C		
	540	1000	μA	+60°C		
	550	1100	μA	+85°C		
	560	1200	μA	+125°C		
PIC18FXXK80	566	1020	μA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	585	1020	μA	+25°C		
	590	1020	μA	+60°C		
	595	1120	μA	+85°C		
	600	1220	μA	+125°C		
PIC18FXXK80	630	2000	μA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	630	2000	μA	+25°C		
	630	2000	μA	+60°C		
	640	2000	μA	+85°C		
	650	2000	μA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

**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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** For LF devices, RETEN (CONFIG1L<0>) = 1.

**5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated)					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (IDD) Cont.<sup>(2,3)</sup></b>							
	PIC18LFXXK80	2	5	mA	-40°C to +125°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	Fosc = 16 MHz (PRI_RUN mode, 4 MHz EC oscillator with PLL)
	PIC18FXXK80	2	5	mA	-40°C to +125°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	PIC18FXXK80	2	6	mA	-40°C to +125°C	VDD = 5.5V <sup>(5)</sup> Regulator Enabled	
	PIC18LFXXK80	7	11	mA	-40°C to +125°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	Fosc = 64 MHz (PRI_RUN mode, 16 MHz EC oscillator with PLL)
	PIC18FXXK80	7	11	mA	-40°C to +125°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	PIC18FXXK80	8	12	mA	-40°C to +125°C	VDD = 5.5V <sup>(5)</sup> Regulator Enabled	
	PIC18LFXXK80	7	11	mA	-40°C to +125°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	PIC18FXXK80	7	11	mA	-40°C to +125°C	VDD = 3.3V <sup>(4)</sup> Regulator Enabled	
	PIC18FXXK80	8	12	mA	-40°C to +125°C	VDD = 5.5V <sup>(5)</sup> Regulator Enabled	
	PIC18LFXXK80	2.3	5	mA	-40°C to +125°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	Fosc = 64 MHz (PRI_IDLE mode, EC oscillator)
	PIC18FXXK80	2.3	5	mA	-40°C to +125°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	PIC18FXXK80	2.5	6	mA	-40°C to +125°C	VDD = 5.5V <sup>(5)</sup> Regulator Enabled	

**Legend:** Shading of rows is to assist in readability of the table.

**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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** For LF devices, RETEN (CONFIG1L<0>) = 1.

**5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	20	70	μA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator disabled	Fosc = 1 MHz (PRI_IDLE mode, EC oscillator)
	20	70	μA	+25°C		
	20	70	μA	+60°C		
	25	80	μA	+85°C		
	30	100	μA	+125°C		
PIC18LFXXK80	37	120	μA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator disabled	
	37	120	μA	+25°C		
	37	120	μA	+60°C		
	40	130	μA	+85°C		
	45	150	μA	+125°C		
PIC18FXXK80	85	140	μA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator enabled	
	100	140	μA	+25°C		
	105	140	μA	+60°C		
	110	150	μA	+85°C		
	120	170	μA	+125°C		
PIC18FXXK80	110	225	μA	-40°C	VDD = 5V <sup>(5)</sup> Regulator enabled	
	110	225	μA	+25°C		
	110	225	μA	+60°C		
	120	230	μA	+85°C		
	130	250	μA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	75	160	µA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	FOSC = 4 MHz (PRI_IDLE mode, EC oscillator)
	75	160	µA	+25°C		
	75	160	µA	+60°C		
	76	170	µA	+85°C		
	82	180	µA	+125°C		
PIC18LFXXK80	148	300	µA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	148	300	µA	+25°C		
	148	300	µA	+60°C		
	150	400	µA	+85°C		
	157	460	µA	+125°C		
PIC18FXXK80	187	320	µA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	204	320	µA	+25°C		
	212	320	µA	+60°C		
	218	420	µA	+85°C		
	230	480	µA	+125°C		
PIC18FXXK80	230	500	µA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	230	500	µA	+25°C		
	230	500	µA	+60°C		
	240	600	µA	+85°C		
	250	700	µA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;
  - MCLR = VDD; WDT enabled/disabled as specified.
- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	2	8	μA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	FOSC = 32 kHz <sup>(3)</sup> <b>(SEC_RUN mode, SOSCSELx = 01)</b>
	5	10	μA	+25°C		
	6	15	μA	+60°C		
	8	20	μA	+85°C		
	13	30	μA	+125°C		
PIC18LFXXK80	3	15	μA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	FOSC = 32 kHz <sup>(3)</sup> <b>(SEC_RUN mode, SOSCSELx = 01)</b>
	16	22	μA	+25°C		
	17	28	μA	+60°C		
	19	39	μA	+85°C		
	25	60	μA	+125°C		
PIC18FXXK80	70	170	μA	-40°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	FOSC = 32 kHz <sup>(3)</sup> <b>(SEC_RUN mode, SOSCSELx = 01)</b>
	70	170	μA	+25°C		
	70	170	μA	+60°C		
	75	180	μA	+85°C		
	90	190	μA	+125°C		
PIC18FXXK80	75	180	μA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled	FOSC = 32 kHz <sup>(3)</sup> <b>(SEC_RUN mode, SOSCSELx = 01)</b>
	75	180	μA	+25°C		
	75	180	μA	+60°C		
	80	190	μA	+85°C		
	95	200	μA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

- 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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

- 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
- 4:** For LF devices, RETEN (CONFIG1L<0>) = 1.
- 5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated)				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Supply Current (IDD) Cont.<sup>(2,3)</sup></b>						
PIC18LFXXK80	1.4	4	μA	-40°C	VDD = 1.8V <sup>(4)</sup> Regulator Disabled	Fosc = 32 kHz <sup>(3)</sup> (SEC_IDLE mode, SOSCSELx = 01)
	2.4	6	μA	+25°C		
	3.6	10	μA	+60°C		
	4.6	12	μA	+85°C		
	9.0	20	μA	+125°		
PIC18LFXXK80	2	5	μA	-40°C	VDD = 3.3V <sup>(4)</sup> Regulator Disabled	
	10	18	μA	+25°C		
	11	22	μA	+60°C		
	13	30	μA	+85°C		
	17	40	μA	+125°		
PIC18FXXK80	55	150	μA	+25°C	VDD = 3.3V <sup>(5)</sup> Regulator Enabled	
	55	150	μA	+60°C		
	55	150	μA	+85°C		
	60	160	μA	+125°C		
	75	170	μA	+25°C		
PIC18FXXK80	53	160	μA	-40°C	VDD = 5V <sup>(5)</sup> Regulator Enabled	
	62	160	μA	+25°C		
	70	160	μA	+60°C		
	85	170	μA	+85°C		
	100	180	μA	+125°C		

**Legend:** Shading of rows is to assist in readability of the table.

**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 a high-impedance state and tied to VDD or Vss, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** For LF devices, RETEN (CONFIG1L<0>) = 1.

**5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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 (ΔIWDT)	<b>Module Differential Currents (ΔIWDT, ΔIBOR, ΔHLVD, ΔIADC)</b>						
	Watchdog Timer						
	PIC18LFXKK80	0.4	2	μA	-40°C to +125°C	VDD = 1.8V Regulator Disabled	
	PIC18LFXKK80	0.6	3	μA	-40°C to +125°C	VDD = 3.3V Regulator Disabled	
	PIC18FXXK80	0.6	3	μA	-40°C to +125°C	VDD = 3.3V Regulator Enabled	
D022A (ΔIBOR)	<b>Brown-out Reset</b>						High-Power BOR
	PIC18LFXKK80	4.6	20	μA	-40°C to +125°C	VDD = 1.8V Regulator Disabled	
	PIC18FXXK80	4.6	20	μA	-40°C to +125°C	VDD = 3.3V Regulator Enabled	
	PIC18FXXK80	4.6	20	μA	-40°C to +125°C	VDD = 5.5V Regulator Enabled	
D022B ΔHLVD	<b>High/Low-Voltage Detect</b>						
	PIC18LFXKK80	3.8	10	μA	-40°C to +125°C	VDD = 1.8V Regulator Disabled	
	PIC18LFXKK80	4.5	12	μA	-40°C to +125°C	VDD = 3.3V Regulator Disabled	
	PIC18FXXK80	3.8	12	μA	-40°C to +125°C	VDD = 3.3V Regulator Enabled	
	PIC18FXXK80	4.9	13	μA	-40°C to +125°C	VDD = 5.5V Regulator Enabled	
D026 ΔIADC	<b>A/D Converter</b>						A/D on, not converting
	PIC18LFXKK80	0.4	1.5		-40°C to +125°C	VDD = 1.8V Regulator Disabled	
	PIC18LFXKK80	0.5	2	μA	-40°C to +125°C	VDD = 3.3V Regulator Disabled	
	PIC18FXXK80	0.5	3	μA	-40°C to +125°C	VDD = 3.3V Regulator Enabled	
	PIC18FXXK80	1	3	μA	-40°C to +125°C	VDD = 5.5V Regulator Enabled	

**Legend:** Shading of rows is to assist in readability of the table.

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, SOSC 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 VDD;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** For LF devices, RETEN (CONFIG1L<0>) = 1.

**5:** For F devices, SRETEN (WDTCON<4>) = 1 and RETEN (CONFIG1L<0>) = 0.

# PIC18F66K80 FAMILY

## 31.3 DC Characteristics: PIC18F66K80 Family (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D031	VIL	<b>Input Low Voltage</b> All I/O Ports: Schmitt Trigger Buffer	Vss	0.2 VDD	V	$1.8\text{V} \leq \text{VDD} \leq 5.5\text{V}$
		RC3 and RC4	Vss	0.3 VDD	V	I <sup>2</sup> C™ enabled
		MCLR	Vss	0.8	V	SMBus enabled
		OSC1	Vss	0.2 VDD	V	LP, XT, HS modes
		OSC1	Vss	0.2 VDD	V	EC modes
		SOSCI	Vss	0.3 VDD	V	
D041	VIH	<b>Input High Voltage</b> All I/O Ports: Schmitt Trigger Buffer	0.8	VDD	V	$1.8\text{V} \leq \text{VDD} \leq 5.5\text{V}$
		RC3 and RC4	0.7 VDD	VDD	V	I <sup>2</sup> C enabled
		MCLR	2.1	VDD	V	SMBus enabled
		OSC1	0.8 VDD	VDD	V	RC mode
		OSC1	0.9 VDD	VDD	V	HS mode
		SOSCI	0.7 VDD	VDD	V	
D060	IIL	<b>Input Leakage Current<sup>(1)</sup></b> I/O Ports	$\pm 50$	$\pm 500$	nA	$\text{VSS} \leq \text{VPIN} \leq \text{VDD}$ , Pin at high-impedance
		MCLR	—	$\pm 500$	nA	$\text{VSS} \leq \text{VPIN} \leq \text{VDD}$
		OSC1	—	1	$\mu\text{A}$	$\text{VSS} \leq \text{VPIN} \leq \text{VDD}$
D070	IPU	<b>Weak Pull-up Current</b> Weak Pull-up Current	50	400	$\mu\text{A}$	$\text{VDD} = 5.5\text{V}$ , $\text{VPIN} = \text{VSS}$
D080	VOL	<b>Output Low Voltage</b> I/O Ports: PORTA, PORTB, PORTC	—	0.6	V	$\text{IOL} = 8.5 \text{ mA}$ , $\text{VDD} = 5.5\text{V}$ , $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
		PORTD, PORTE, PORTF, PORTG	—	0.6	V	$\text{IOL} = 3.5 \text{ mA}$ , $\text{VDD} = 5.5\text{V}$ , $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
		OSC2/CLKO (EC modes)	—	0.6	V	$\text{IOL} = 1.6 \text{ mA}$ , $\text{VDD} = 5.5\text{V}$ , $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$

**Note 1:** Negative current is defined as current sourced by the pin.

# PIC18F66K80 FAMILY

---

## 31.3 DC Characteristics: PIC18F66K80 Family (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D090	VOH	Output High Voltage <sup>(1)</sup>			V	
		I/O Ports: PORTA, PORTB, PORTC	VDD – 0.7	—	V	$\text{IOH} = -3 \text{ mA}, \text{VDD} = 5.5\text{V},$ $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
		PORTD, PORTE, PORTF, PORTG	VDD – 0.7	—	V	$\text{IOH} = -2 \text{ mA}, \text{VDD} = 5.5\text{V},$ $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
D092		OSC2/CLKO (INTOSC, EC modes)	VDD – 0.7	—	V	$\text{IOH} = -1 \text{ mA}, \text{VDD} = 5.5\text{V},$ $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
<b>Capacitive Loading Specs on Output Pins</b>						
D100 <sup>(4)</sup>	COSC2	OSC2 Pin	—	20	pF	In HS mode when external clock is used to drive OSC1
D101	C <sub>IO</sub>	All I/O Pins and OSC2	—	50	pF	To meet the AC Timing Specifications
D102	CB	SCL, SDA	—	400	pF	I <sup>2</sup> C™ Specification

**Note 1:** Negative current is defined as current sourced by the pin.

# PIC18F66K80 FAMILY

## 31.4 DC Characteristics: PIC18F66K80 Family (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions: 1.8V to 5.5V Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for Industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for Extended				
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
D160a	I <sub>IICL</sub>	Input Low Injection Current	0	—	-5 <sup>(1)</sup>	mA	All pins except V <sub>DD</sub> , V <sub>SS</sub> , AV <sub>DD</sub> , AV <sub>SS</sub> , MCLR, V <sub>CAP</sub> , SOSCI, SOSCO
D160b	I <sub>IICH</sub>	Input High Injection Current	0	—	+5 <sup>(1)</sup>	mA	All pins except V <sub>DD</sub> , V <sub>SS</sub> , AV <sub>DD</sub> , AV <sub>SS</sub> , MCLR, V <sub>CAP</sub> , SOSCI, SOSCO
D160c	$\Sigma I_{ICT}$	Total Input Injection Current (sum of all I/O and control pins)	-20 <sup>(1,2)</sup>	—	+20 <sup>(1,2)</sup>	mA	Absolute instantaneous sum of all input injection currents from all I/O pins ( $ I_{IICL} + I_{IICH}  \leq \Sigma I_{ICT}$ )

Note 1: Injection currents  $> |0|$  can affect the A/D results.

2: Any number and/or combination of I/O pins not excluded under I<sub>IICL</sub> or I<sub>IICH</sub> conditions are permitted.

## 31.5 DC Characteristics: CTMU Current Source Specifications

DC CHARACTERISTICS			Standard Operating Conditions: 1.8V to 5.5V Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for Industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for Extended				
Param No.	Sym	Characteristic	Min	Typ <sup>(1)</sup>	Max	Units	Conditions
	I <sub>OUT1</sub>	CTMU Current Source, Base Range	—	550	—	nA	CTMUICON<1:0> = 01
	I <sub>OUT2</sub>	CTMU Current Source, 10x Range	—	5.5	—	$\mu\text{A}$	CTMUICON<1:0> = 10
	I <sub>OUT3</sub>	CTMU Current Source, 100x Range	—	55	—	$\mu\text{A}$	CTMUICON<1:0> = 11

Note 1: Nominal value at center point of current trim range (CTMUICON<7:2> = 000000).

# PIC18F66K80 FAMILY

TABLE 31-1: MEMORY PROGRAMMING REQUIREMENTS

DC CHARACTERISTICS			Standard Operating Conditions Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for Industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for Extended				
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D110	VPP	<b>Internal Program Memory Programming Specifications<sup>(1)</sup></b>	$\text{VDD} + 1.5$	—	10	V	<b>(Note 3, Note 4)</b>
D113	IDDP	Voltage on MCLR/VPP/RE5 pin Supply Current during Programming	—	—	10	mA	
D120	ED	<b>Data EEPROM Memory</b>	100K	1000K	—	E/W	<b>(Note 2)</b> $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
D121	VDRW	Byte Endurance VDD for Read/Write	1.8	—	5.5	V	
			1.8	—	3.6	V	Using EECON to read/write PIC18FXXKXX devices
D122	TDEW	Erase/Write Cycle Time	—	4	—	ms	Using EECON to read/write PIC18LFXXKXX devices
D123	TRETD	Characteristic Retention	20	—	—	Year	
D124	TREF	Number of Total Erase/Write Cycles before Refresh <sup>(2)</sup>	1M	10M	—	E/W	Provided no other specifications are violated $-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
D130	EP	<b>Program Flash Memory</b>	1K	10K	—	E/W	$-40^{\circ}\text{C} \text{ to } +125^{\circ}\text{C}$
D131	VPR	Cell Endurance VDD for Read	1.8	—	5.5	V	
			1.8	—	3.6	V	PIC18FXXKXX devices
D132B	VPEW	Voltage for Self-Timed Erase or Write Operations	—	—	—	—	PIC18FXXKXX devices
	VDD	VDD	1.8	—	5.5	V	
D133A	TIW	Self-Timed Write Cycle Time	—	2	—	ms	PIC18FXXKXX devices
D134	TRETD	Characteristic Retention	20	—	—	Year	
D135	IDDP	Supply Current during Programming	—	—	10	mA	Provided no other specifications are violated
D140	TWE	Writes per Erase Cycle	—	—	1	—	
† Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.							

- Note 1:** These specifications are for programming the on-chip program memory through the use of table write instructions.
- 2:** Refer to [Section 8.8 “Using the Data EEPROM”](#) for a more detailed discussion on data EEPROM endurance.
- 3:** Required only if Single-Supply Programming is disabled.
- 4:** The MPLAB® ICD 2 does not support variable VPP output. Circuitry to limit the ICD2 VPP voltage must be placed between the ICD 2 and target system when programming or debugging with the ICD2.

# PIC18F66K80 FAMILY

**TABLE 31-2: COMPARATOR SPECIFICATIONS**

**Operating Conditions:**  $1.8V \leq VDD \leq 5.5V$ ,  $-40^{\circ}C \leq TA \leq +125^{\circ}C$

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D300	VIOFF	Input Offset Voltage	—	$\pm 5.0$	40	mV	
D301	VICM	Input Common Mode Voltage	—	—	AVDD	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
D303	TRESP	Response Time <sup>(1)</sup>	—	150	400	ns	
D304	Tmc2ov	Comparator Mode Change to Output Valid*	—	—	10	$\mu s$	

**Note 1:** Response time is measured with one comparator input at  $(AVDD - 1.5)/2$ , while the other input transitions from Vss to VDD.

**TABLE 31-3: VOLTAGE REFERENCE SPECIFICATIONS**

**Operating Conditions:**  $1.8V \leq VDD \leq 5.5V$ ,  $-40^{\circ}C \leq TA \leq +125^{\circ}C$

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D310	VRES	Resolution	$VDD/24$	—	$VDD/32$	LSb	
D311	VRAA	Absolute Accuracy	—	—	$1/2$	LSb	
D312	VRUR	Unit Resistor Value (R)	—	2k	—	$\Omega$	
D313	TSET	Settling Time <sup>(1)</sup>	—	—	10	$\mu s$	

**Note 1:** Settling time measured while CVRR = 1 and CVR<3:0> transitions from '0000' to '1111'.

**TABLE 31-4: INTERNAL VOLTAGE REGULATOR SPECIFICATIONS**

**Operating Conditions:**  $-40^{\circ}C \leq TA \leq +125^{\circ}C$

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
	VRGOUT	Regulator Output Voltage	—	3.3	—	V	
	CEFC	External Filter Capacitor Value	4.7	10	—	$\mu F$	Capacitor must be low-ESR, a low series resistance (< $5\Omega$ )

# PIC18F66K80 FAMILY

## 31.6 AC (Timing) Characteristics

### 31.6.1 TIMING PARAMETER SYMOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST      ( $I^2C$  specifications only)
4. Ts            ( $I^2C$  specifications only)

T	F	Frequency	T	Time
---	---	-----------	---	------

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	MCLR	wr	$\overline{WR}$

Uppercase letters and their meanings:

S		P	Period
F	Fall	R	Rise
H	High	V	Valid
I	Invalid (High-impedance)	Z	High-impedance
L	Low	High	High
$I^2C$ only	AA	Low	Low
	BUF	Bus free	

Tcc:ST ( $I^2C$  specifications only)

CC		SU	Setup
HD	Hold	STO	Stop condition
ST			
DAT	DATA input hold		
STA	Start condition		

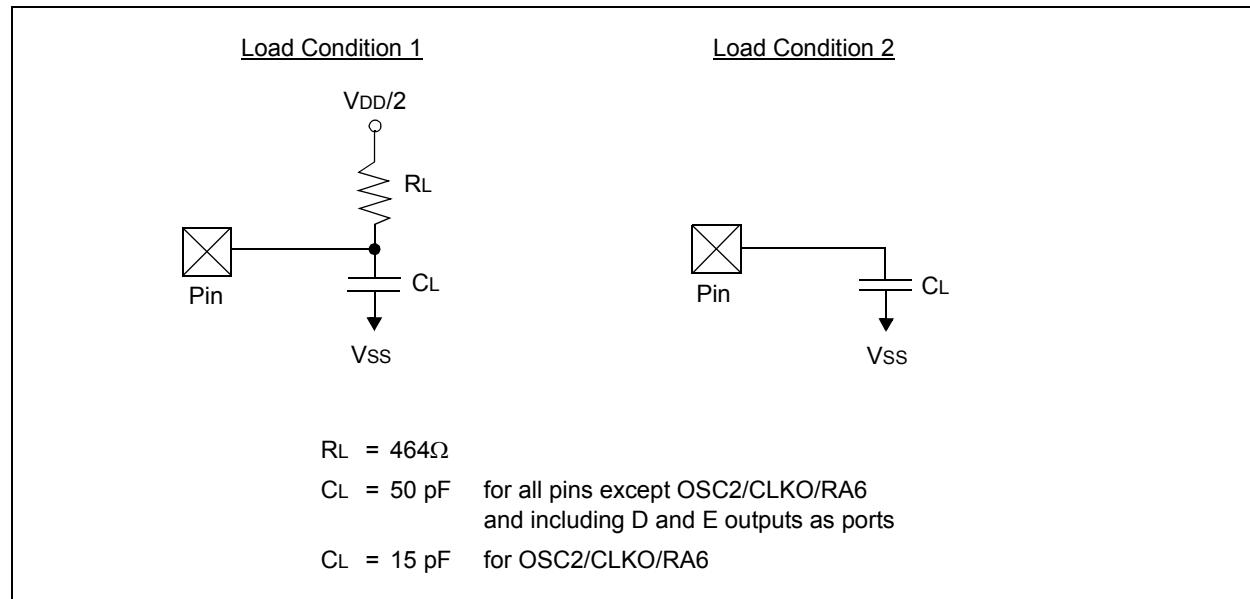
## 31.6.2 TIMING CONDITIONS

The temperature and voltages specified in [Table 31-5](#) apply to all timing specifications unless otherwise noted. [Figure 31-3](#) specifies the load conditions for the timing specifications.

**TABLE 31-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

AC CHARACTERISTICS	Standard Operating Conditions (unless otherwise stated)	
	Operating temperature	-40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended
	Operating voltage VDD range as described in <a href="#">Section 31.1</a> and <a href="#">Section 31.3</a> .	

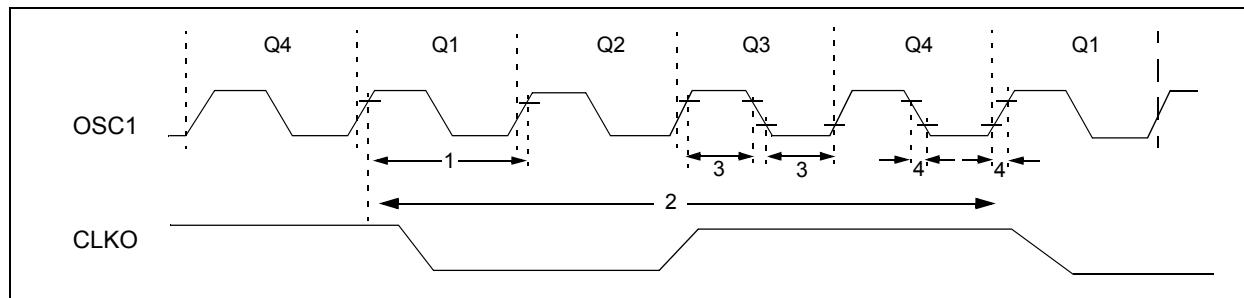
**FIGURE 31-3: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**



# PIC18F66K80 FAMILY

## 31.6.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 31-4: EXTERNAL CLOCK TIMING**



**TABLE 31-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	Fosc	External CLKIN Frequency <sup>(1)</sup>	DC	64	MHz	EC, ECIO Oscillator mode
		Oscillator Frequency <sup>(1)</sup>	DC	4	MHz	RC Oscillator mode
			0.1	4	MHz	XT Oscillator mode
			4	16	MHz	HS Oscillator mode
			4	16	MHz	HS + PLL Oscillator mode
			5	33	kHz	LP Oscillator mode
1	Tosc	External CLKIN Period <sup>(1)</sup>	15.6	—	ns	EC, ECIO Oscillator mode
		Oscillator Period <sup>(1)</sup>	250	—	ns	RC Oscillator mode
			250	10,000	ns	XT Oscillator mode
			40	250	ns	HS Oscillator mode
			62.5	250	ns	HS + PLL Oscillator mode
			5	200	μs	LP Oscillator mode
2	TCY	Instruction Cycle Time <sup>(1)</sup>	62.5	—	ns	TCY = 4/FOSC
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/CLKIN pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.

# PIC18F66K80 FAMILY

**TABLE 31-7: PLL CLOCK TIMING SPECIFICATIONS (VDD = 1.8V TO 5.5V)**

Param No.	Sym	Characteristic	Min	Typ	Max	Units	Conditions
F10	FOSC	Oscillator Frequency Range	4	—	5	MHz	VDD = 1.8-5.5V
			4	—	16	MHz	VDD = 3.0-5.5V, -40°C to +125°C
F11	Fsys	On-Chip VCO System Frequency	16	—	20	MHz	VDD = 1.8-5.5V
			16	—	64	MHz	VDD = 3.0-5.5V, -40°C to +125°C
F12	t <sub>rc</sub>	PLL Start-up Time (Lock Time)	—	—	2	ms	
F13	ΔCLK	CLKOUT Stability (Jitter)	-2	—	+2	%	

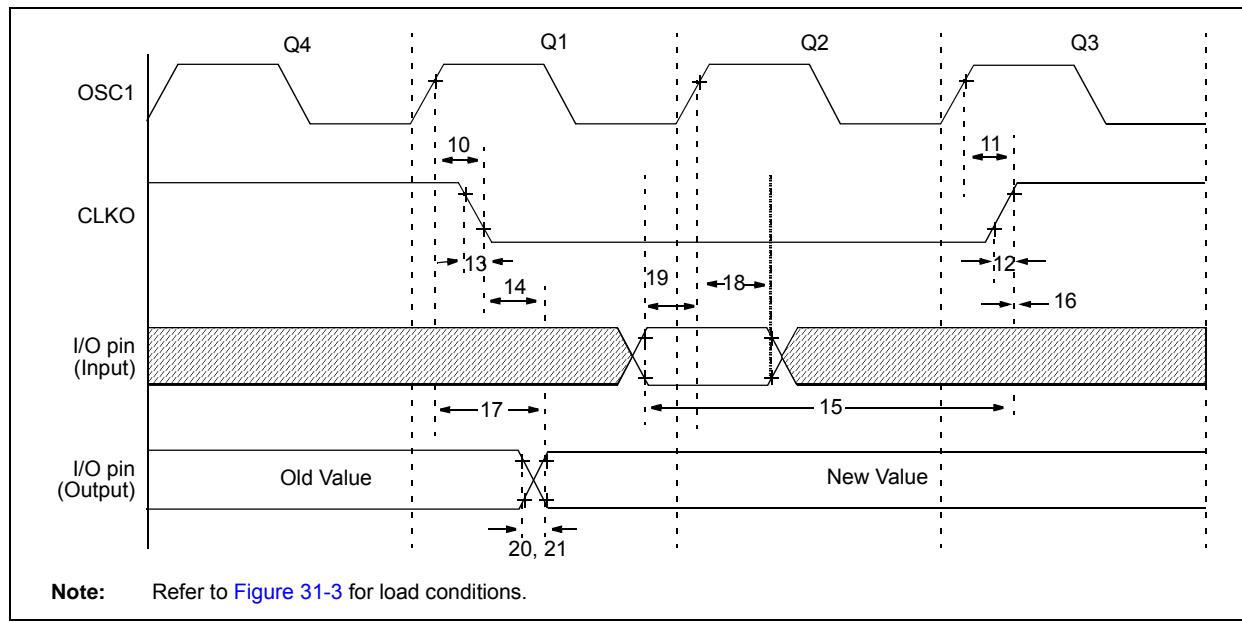
**TABLE 31-8: INTERNAL RC ACCURACY (INTOSC)**

PIC18F66K80 Family		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C						
Param No.		Min	Typ	Max	Units	Conditions		
OA1	<b>HFINTOSC/MFINTOSC Accuracy @ Freq = 16 MHz, 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz<sup>(1)</sup></b>		-2	—	+2	%	+25°C	VDD = 3-5.5V
			-5	—	+5	%	-40°C to +85°C	VDD = 1.8-5.5V
			-10	—	+10	%	-40°C to +125°C	VDD = 1.8-5.5V
OA2	<b>LFINTOSC Accuracy @ Freq = 31 kHz</b>		-15	—	+15	%	-40°C to +125°C	VDD = 1.8-5.5V

**Note 1:** Frequency is calibrated at +25°C. OSCTUNE register can be used to compensate for temperature drift.

# PIC18F66K80 FAMILY

**FIGURE 31-5: CLKO AND I/O TIMING**



**TABLE 31-9: CLKO AND I/O TIMING REQUIREMENTS**

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
10	TosH2CKL	OSC1 $\uparrow$ to CLKO $\downarrow$	—	75	200	ns	(Note 1)
11	TosH2ckH	OSC1 $\uparrow$ to CLKO $\uparrow$	—	75	200	ns	(Note 1)
12	TckR	CLKO Rise Time	—	15	30	ns	(Note 1)
13	TckF	CLKO Fall Time	—	15	30	ns	(Note 1)
14	TckL2ioV	CLKO $\downarrow$ to Port Out Valid	—	—	0.5 TCY + 20	ns	
15	TioV2ckH	Port In Valid before CLKO $\uparrow$	0.25 TCY + 25	—	—	ns	
16	TckH2iol	Port In Hold after CLKO $\uparrow$	0	—	—	ns	
17	TosH2ioV	OSC1 $\uparrow$ (Q1 cycle) to Port Out Valid	—	50	150	ns	
18	TosH2iol	OSC1 $\uparrow$ (Q2 cycle) to Port Input Invalid (I/O in hold time)	100	—	—	ns	
19	TioV2osH	Port Input Valid to OSC1 $\uparrow$ (I/O in setup time)	0	—	—	ns	
20	TioR	Port Output Rise Time	—	10	25	ns	
21	TioF	Port Output Fall Time	—	10	25	ns	
22†	TINP	INTx pin High or Low Time	20	—	—	ns	
23†	TRBP	RB<7:4> Change INTx High or Low Time	TCY	—	—	ns	

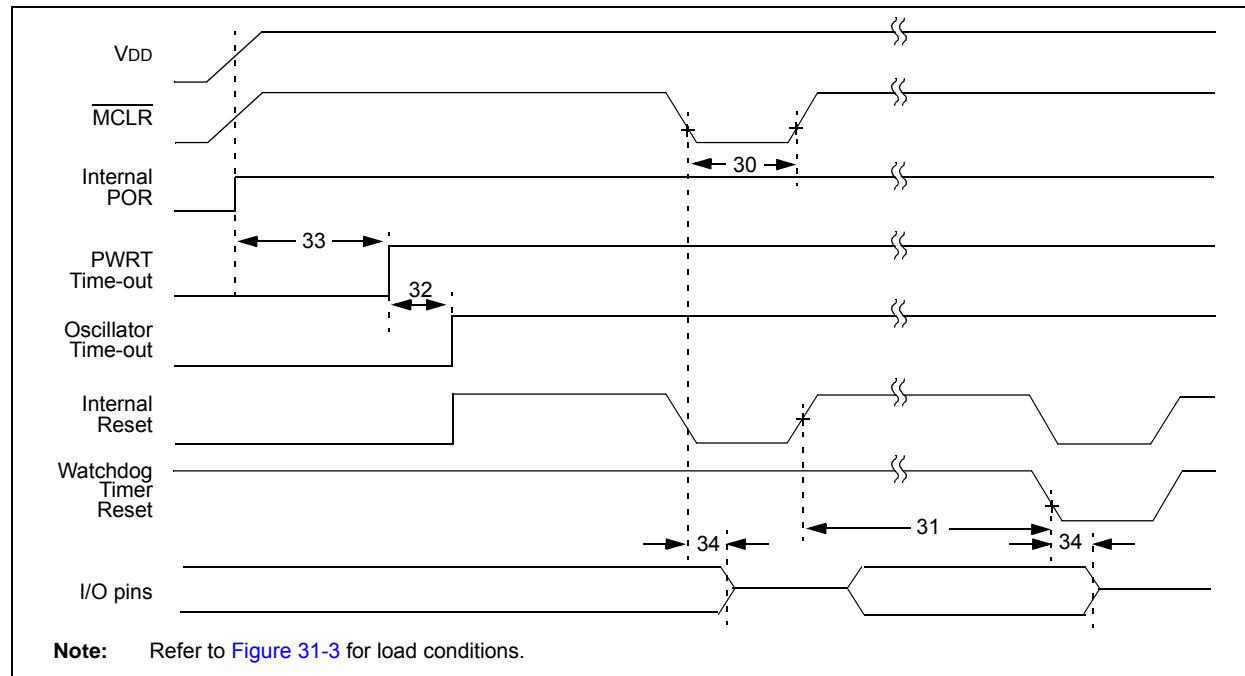
† These parameters are asynchronous events not related to any internal clock edges.

**Note 1:** Measurements are taken in EC mode, where CLKO output is 4 x Tosc.

TABLE 31-10: CLKO AND I/O TIMING REQUIREMENTS

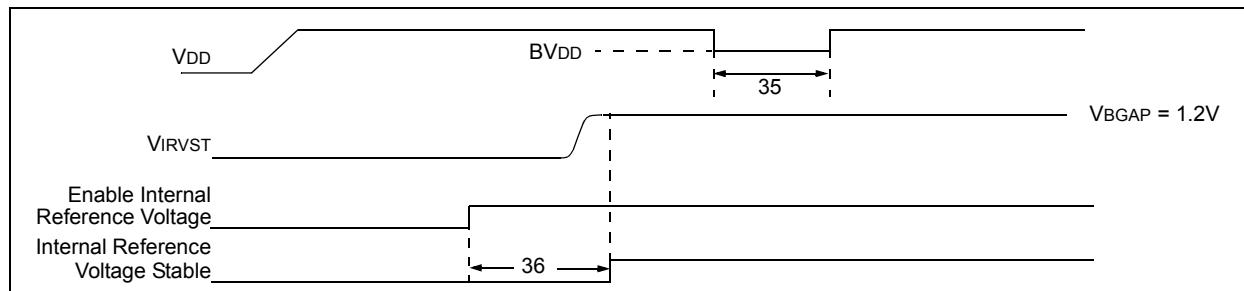
Param. No	Symbol	Characteristics	Min	Typ	Max	Units
150	TadV2aLL	Address Out Valid to ALE ↓ (address setup time)	0.25 TCY – 10	—	—	ns
151	TaI2adL	ALE ↓ to Address Out Invalid (address hold time)	5	—	—	ns
155	TaI2oeL	ALE ↓ to $\overline{OE}$ ↓	10	0.125 TCY	—	ns
160	TadZ2oeL	AD High-Z to $\overline{OE}$ ↓ (bus release to $\overline{OE}$ )	0	—	—	ns
161	ToeH2adD	$\overline{OE}$ ↑ to AD Driven	0.125 TCY – 5	—	—	ns
162	TadV2oeH	LS Data Valid before $\overline{OE}$ ↑ (data setup time)	20	—	—	ns
163	ToeH2adL	$\overline{OE}$ ↑ to Data In Invalid (data hold time)	0	—	—	ns
164	TaI2aLL	ALE Pulse Width	—	0.25 TCY	—	ns
165	ToeL2oeH	$\overline{OE}$ Pulse Width	0.5 TCY – 5	0.5 TCY	—	ns
166	TaI2aLH	ALE ↑ to ALE ↑ (cycle time)	—	TCY	—	ns
167	Tacc	Address Valid to Data Valid	0.75 TCY – 25	—	—	ns
168	Toe	$\overline{OE}$ ↓ to Data Valid	—	—	0.5 TCY – 25	ns
169	TaI2oeH	ALE ↓ to $\overline{OE}$ ↑	0.625 TCY – 10	—	0.625 TCY + 10	ns
171	TaI2csL	Chip Enable Active to ALE ↓	0.25 TCY – 20	—	—	ns
171A	TubL2oeH	AD Valid to Chip Enable Active	—	—	10	ns

FIGURE 31-6: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



# PIC18F66K80 FAMILY

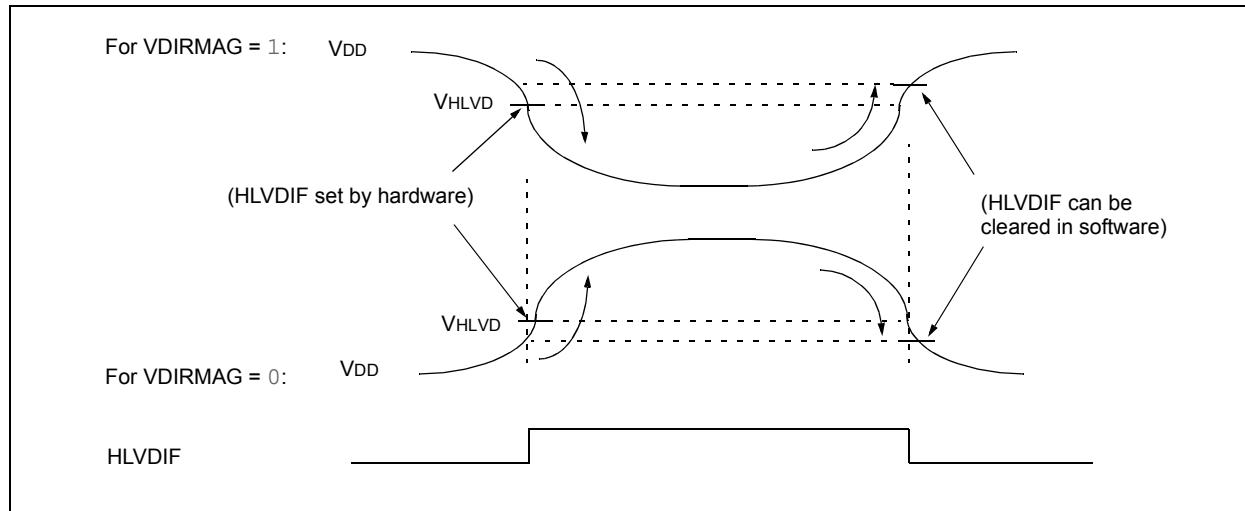
**FIGURE 31-7: BROWN-OUT RESET TIMING**



**TABLE 31-11: 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)	—	4	—	ms	
32	TOST	Oscillation Start-up Timer Period	1024 Tosc	—	1024 Tosc	—	Tosc = OSC1 period
33	TPWRT	Power-up Timer Period	—	1	—	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	TIVRST	Time for Internal Reference Voltage to become Stable	—	25	—	μs	
37	THLVD	High/Low-Voltage Detect Pulse Width	200	—	—	μs	VDD ≤ VHLVD
38	TCSD	CPU Start-up Time	5	—	10	μs	
39	TIOBST	Time for INTOSC to Stabilize	—	1	—	μs	

**FIGURE 31-8: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

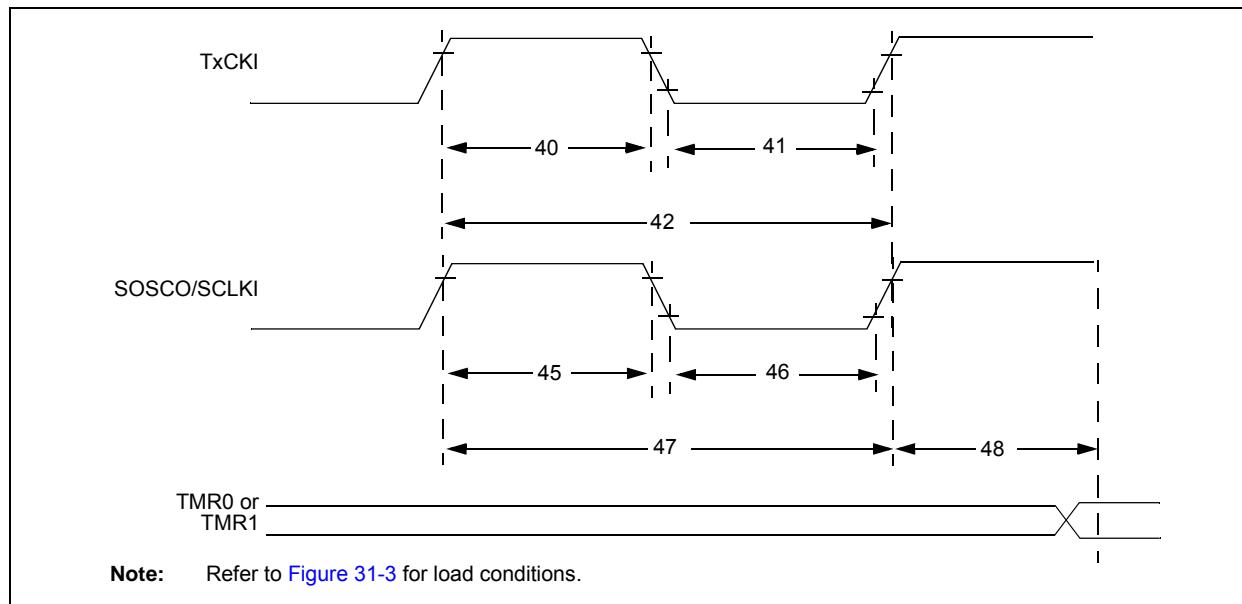


**TABLE 31-12: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended							
Param No.	Sym	Characteristic	Min	Typ	Max	Units	Conditions
D420		HLVD Voltage on VDD Transition High-to-Low	HLVDL<3:0> = 0000	1.80	1.85	1.90	V
			HLVDL<3:0> = 0001	2.03	2.08	2.13	V
			HLVDL<3:0> = 0010	2.24	2.29	2.35	V
			HLVDL<3:0> = 0011	2.40	2.46	2.53	V
			HLVDL<3:0> = 0100	2.50	2.56	2.62	V
			HLVDL<3:0> = 0101	2.70	2.77	2.84	V
			HLVDL<3:0> = 0110	2.82	2.89	2.97	V
			HLVDL<3:0> = 0111	2.95	3.02	3.10	V
			HLVDL<3:0> = 1000	3.24	3.32	3.41	V
			HLVDL<3:0> = 1001	3.42	3.50	3.59	V
			HLVDL<3:0> = 1010	3.61	3.70	3.79	V
			HLVDL<3:0> = 1011	3.82	3.91	4.10	V
			HLVDL<3:0> = 1100	4.06	4.16	4.26	V
			HLVDL<3:0> = 1101	4.33	4.44	4.55	V
			HLVDL<3:0> = 1110	4.64	4.75	4.87	V

# PIC18F66K80 FAMILY

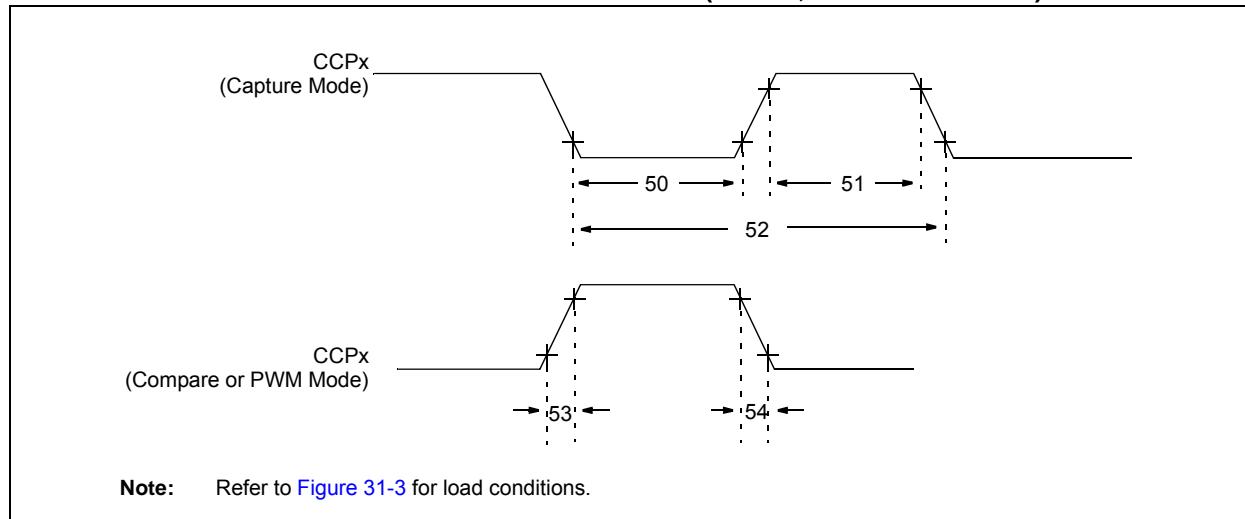
**FIGURE 31-9: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**



**TABLE 31-13: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
40	T <sub>T0H</sub>	T0CKI High Pulse Width	No prescaler	0.5 T <sub>CY</sub> + 20	—	ns	
			With prescaler	10	—	ns	
41	T <sub>T0L</sub>	T0CKI Low Pulse Width	No prescaler	0.5 T <sub>CY</sub> + 20	—	ns	
			With prescaler	10	—	ns	
42	T <sub>T0P</sub>	T0CKI Period	No prescaler	T <sub>CY</sub> + 10	—	ns	N = prescale value (1, 2, 4,..., 256)
			With prescaler	Greater of: 20 ns or (T <sub>CY</sub> + 40)/N	—	ns	
45	T <sub>T1H</sub>	T1CKI High Time	Synchronous, no prescaler	0.5 T <sub>CY</sub> + 20	—	ns	
			Synchronous, with prescaler	10	—	ns	
			Asynchronous	30	—	ns	
46	T <sub>T1L</sub>	T1CKI Low Time	Synchronous, no prescaler	0.5 T <sub>CY</sub> + 5	—	ns	
			Synchronous, with prescaler	10	—	ns	
			Asynchronous	30	—	ns	
47	T <sub>T1P</sub>	T1CKI Input Period	Synchronous	Greater of: 20 ns or (T <sub>CY</sub> + 40)/N	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	60	—	ns	
	F <sub>T1</sub>	T1CKI Oscillator Input Frequency Range		DC	50	KHz	
48	T <sub>CKE2TMRI</sub>	Delay from External T1CKI Clock Edge to Timer Increment		2 T <sub>osc</sub>	7 T <sub>osc</sub>	—	

**FIGURE 31-10: CAPTURE/COMPARE/PWM TIMINGS (ECCP1, ECCP2 MODULES)**

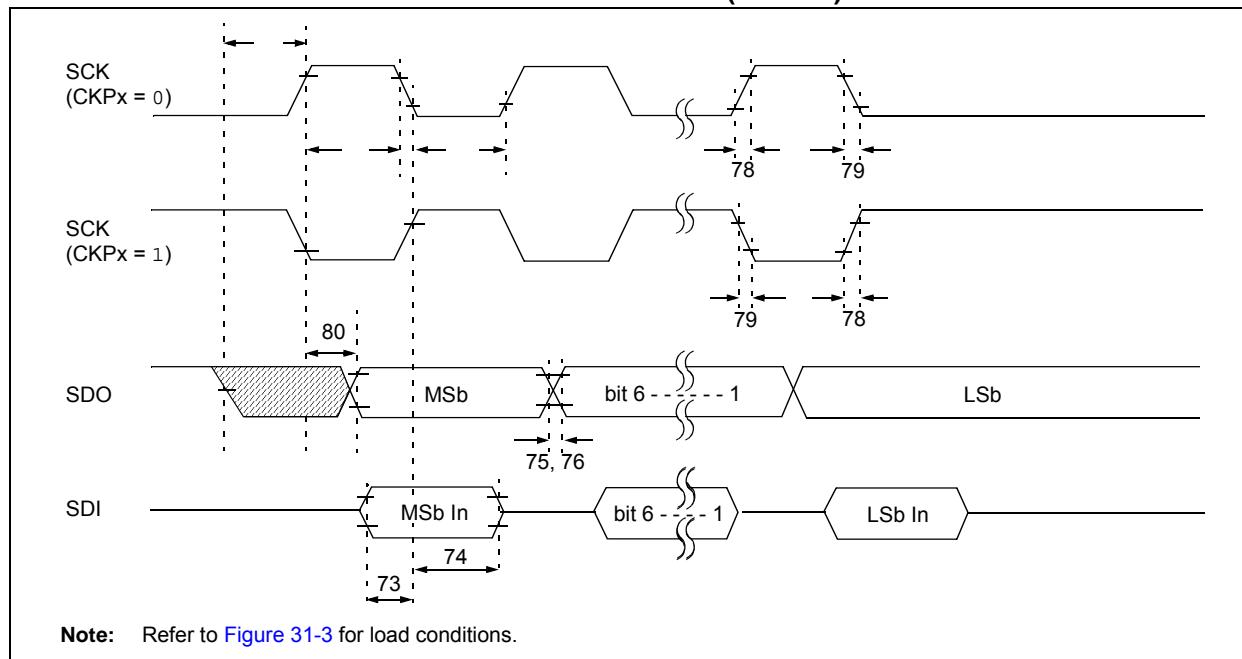


**TABLE 31-14: CAPTURE/COMPARE/PWM REQUIREMENTS (ECCP1, ECCP2 MODULES)**

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
50	TccL	CCPx Input Low Time	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
51	TccH	CCPx Input High Time	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
52	TccP	CCPx Input Period		$\frac{3 \text{ TCY} + 40}{N}$	—	ns	N = prescale value (1, 4 or 16)
53	TccR	CCPx Output Fall Time		—	25	ns	
54	TccF	CCPx Output Fall Time		—	25	ns	

# PIC18F66K80 FAMILY

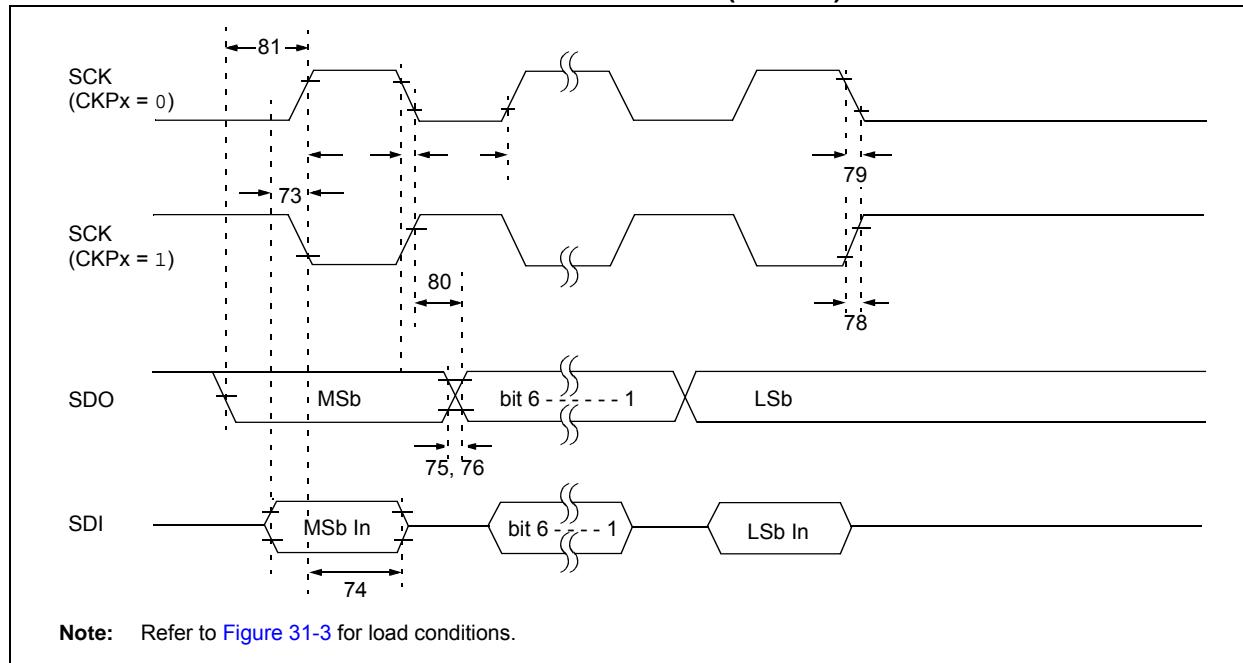
**FIGURE 31-11: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)**



**TABLE 31-15: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
73	T <sub>DIV2SCH</sub> , T <sub>DIV2SCL</sub>	Setup Time of SDI Data Input to SCK Edge	20	—	ns	
73A	T <sub>B2B</sub>	Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2	1.5 T <sub>CY</sub> + 40	—	ns	
74	T <sub>sch2DIL</sub> , T <sub>scL2DIL</sub>	Hold Time of SDI Data Input to SCK Edge	40	—	ns	
75	T <sub>DO</sub> R	SDO Data Output Rise Time	—	25	ns	
76	T <sub>DO</sub> F	SDO Data Output Fall Time	—	25	ns	
78	T <sub>scR</sub>	SCK Output Rise Time (Master mode)	—	25	ns	
79	T <sub>scF</sub>	SCK Output Fall Time (Master mode)	—	25	ns	
80	T <sub>sch2DoV</sub> , T <sub>scL2DoV</sub>	SDO Data Output Valid after SCK Edge	—	50	ns	

**FIGURE 31-12: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)**

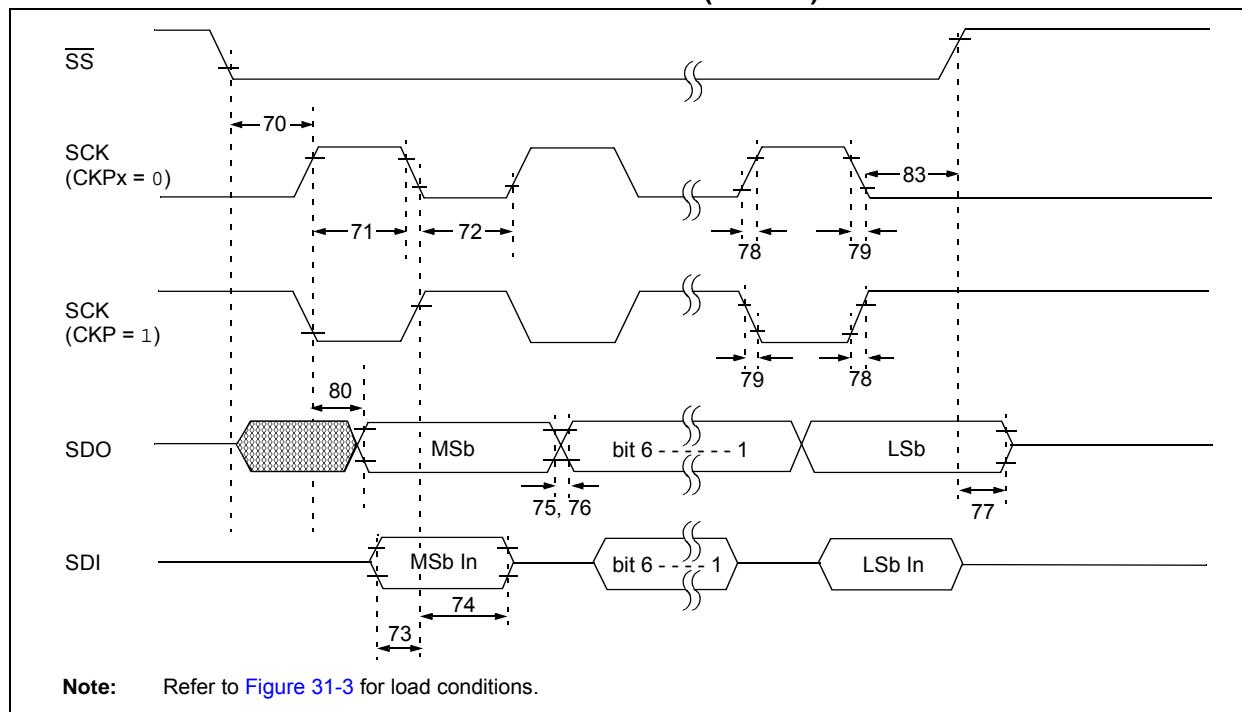


**TABLE 31-16: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
73	TdIV2sCH, TdIV2sCL	Setup Time of SDI Data Input to SCK Edge	20	—	ns	
73A	Tb2B	Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2	1.5 TCY + 40	—	ns	
74	TsCH2DIL, TsCL2DIL	Hold Time of SDI Data Input to SCK Edge	40	—	ns	
75	TdoR	SDO Data Output Rise Time	—	25	ns	
76	TdoF	SDO Data Output Fall Time	—	25	ns	
78	Tscr	SCK Output Rise Time (Master mode)	—	25	ns	
79	TscF	SCK Output Fall Time (Master mode)	—	25	ns	
80	TsCH2DoV, TsCL2DoV	SDO Data Output Valid after SCK Edge	—	50	ns	
81	TdoV2sCH, TdoV2sCL	SDO Data Output Setup to SCK Edge	TCY	—	ns	

# PIC18F66K80 FAMILY

**FIGURE 31-13: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)**



**TABLE 31-17: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)**

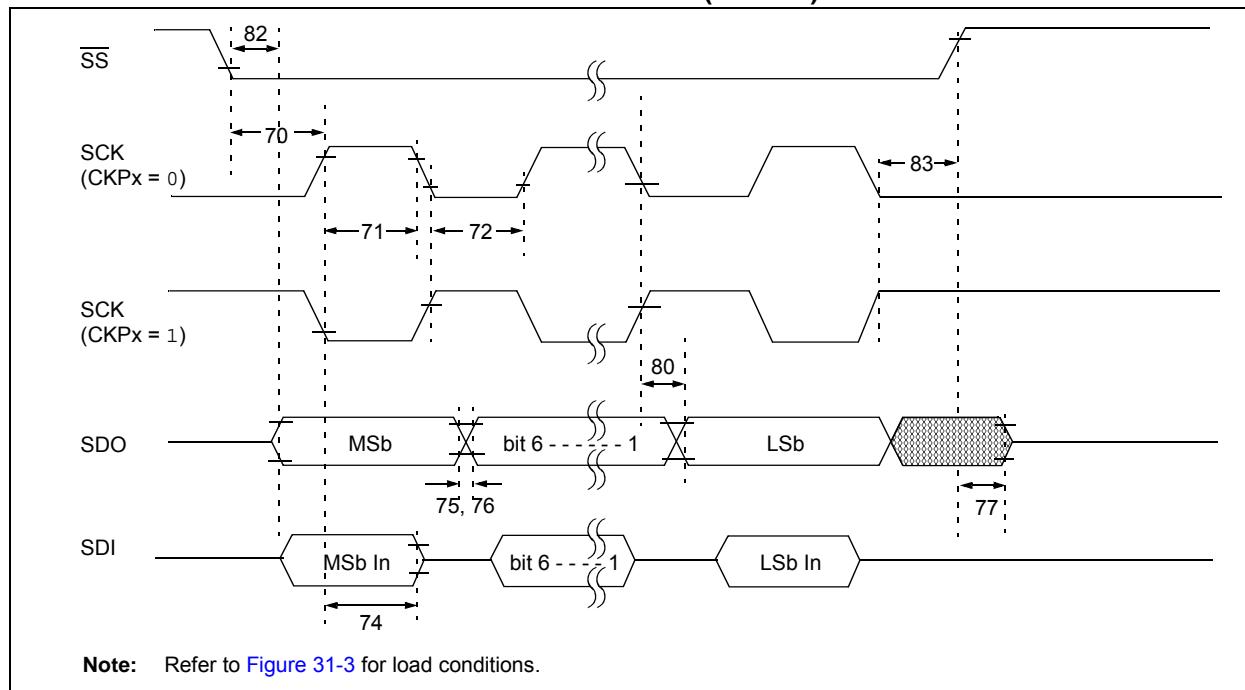
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
70	TssL2sch, TssL2scl	SS $\downarrow$ to SCK $\downarrow$ or SCK $\uparrow$ Input	3 TCY	—	ns	
70A	TssL2wb	SS to write to SSPBUF	3 TCY	—	ns	
71	Tsch	SCK Input High Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns
			Single Byte	40	—	ns (Note 1)
72	Tscl	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns
			Single Byte	40	—	ns (Note 1)
73	Tdiv2sch, Tdiv2scl	Setup Time of SDI Data Input to SCK Edge	20	—	ns	
73A	Tb2b	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2	1.5 TCY + 40	—	ns	(Note 2)
74	Tsch2dil, Tscl2dil	Hold Time of SDI Data Input to SCK Edge	40	—	ns	
75	TdoR	SDO Data Output Rise Time	—	25	ns	
76	TdoF	SDO Data Output Fall Time	—	25	ns	
77	Tssh2doz	SS $\uparrow$ to SDO Output High-impedance	10	50	ns	
78	Tscr	SCK Output Rise Time (Master mode)	—	25	ns	
79	Tscf	SCK Output Fall Time (Master mode)	—	25	ns	
80	Tsch2dov, Tscl2dov	SDO Data Output Valid after SCK Edge	—	50	ns	
83	Tsch2ssh, Tscl2ssh	SS $\uparrow$ after SCK Edge	1.5 TCY + 40	—	ns	

**Note 1:** Requires the use of Parameter 73A.

**2:** Only if Parameter 71A and 72A are used.

# PIC18F66K80 FAMILY

**FIGURE 31-14: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)**



**TABLE 31-18: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)**

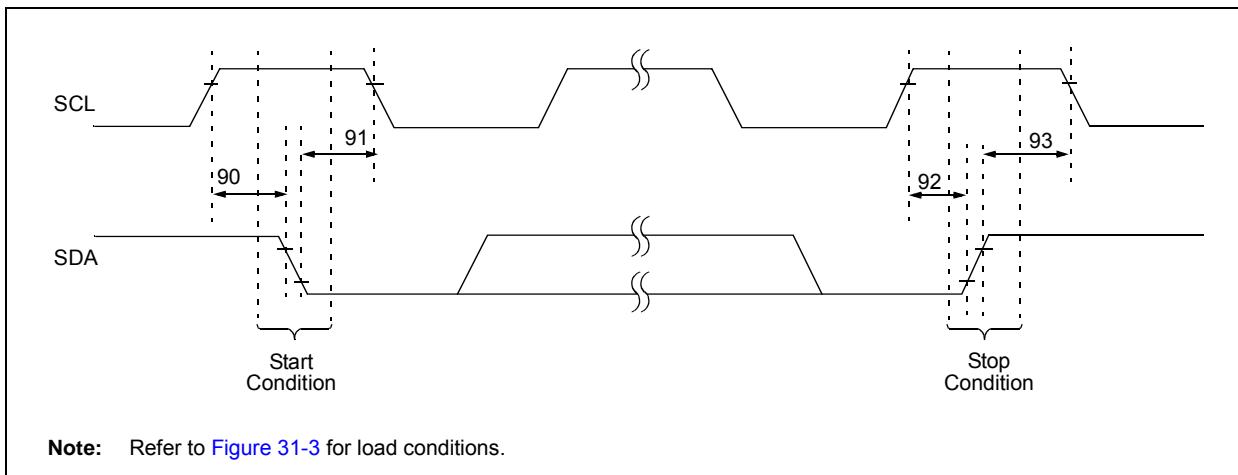
Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2sch, TssL2scl	$\overline{SS} \downarrow$ to $SCK \downarrow$ or $SCK \uparrow$ Input		3 Tcy	—	ns	
70A	TssL2WB	$\overline{SS}$ to write to SSPBUF		3 Tcy	—	ns	
71	TsCH	SCK Input High Time (Slave mode)	Continuous	1.25 Tcy + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
72	TsCL	SCK Input Low Time (Slave mode)	Continuous	1.25 Tcy + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
73A	Tb2B	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2	1.5 Tcy + 40	—	ns		(Note 2)
74	TsCH2DIL, TsCL2DIL	Hold Time of SDI Data Input to SCK Edge	40	—	ns		
75	TdoR	SDO Data Output Rise Time	—	25	ns		
76	TdoF	SDO Data Output Fall Time	—	25	ns		
77	Tssh2doZ	$\overline{SS} \uparrow$ to SDO Output High-Impedance	10	50	ns		
78	Tscr	SCK Output Rise Time (Master mode)	—	25	ns		
79	Tscf	SCK Output Fall Time (Master mode)	—	25	ns		
80	TsCH2dov, TsCL2dov	SDO Data Output Valid after SCK Edge	—	50	ns		
82	TssL2dov	SDO Data Output Valid after $\overline{SS} \downarrow$ Edge	—	50	ns		
83	Tsch2ssH, Tscl2ssH	$\overline{SS} \uparrow$ after SCK Edge	1.5 Tcy + 40	—	ns		

**Note 1:** Requires the use of Parameter 73A.

**2:** Only if Parameter 71A and 72A are used.

# PIC18F66K80 FAMILY

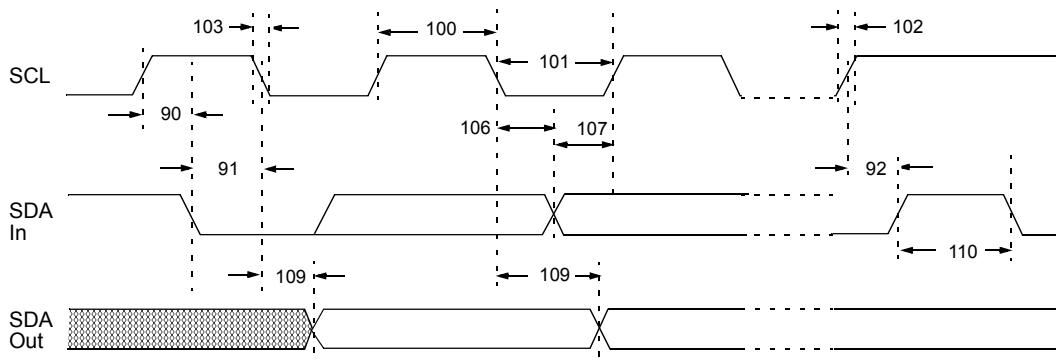
**FIGURE 31-15: I<sup>2</sup>C™ BUS START/STOP BITS TIMING**



**TABLE 31-19: I<sup>2</sup>C™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)**

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4700	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	600	—		
91	THD:STA	Start Condition Hold Time	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—		
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4700	—	ns	
			400 kHz mode	600	—		
93	THD:STO	Stop Condition Hold Time	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—		

**FIGURE 31-16: I<sup>2</sup>C<sup>TM</sup> BUS DATA TIMING**



**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-20: I<sup>2</sup>C<sup>TM</sup> BUS DATA REQUIREMENTS (SLAVE MODE)**

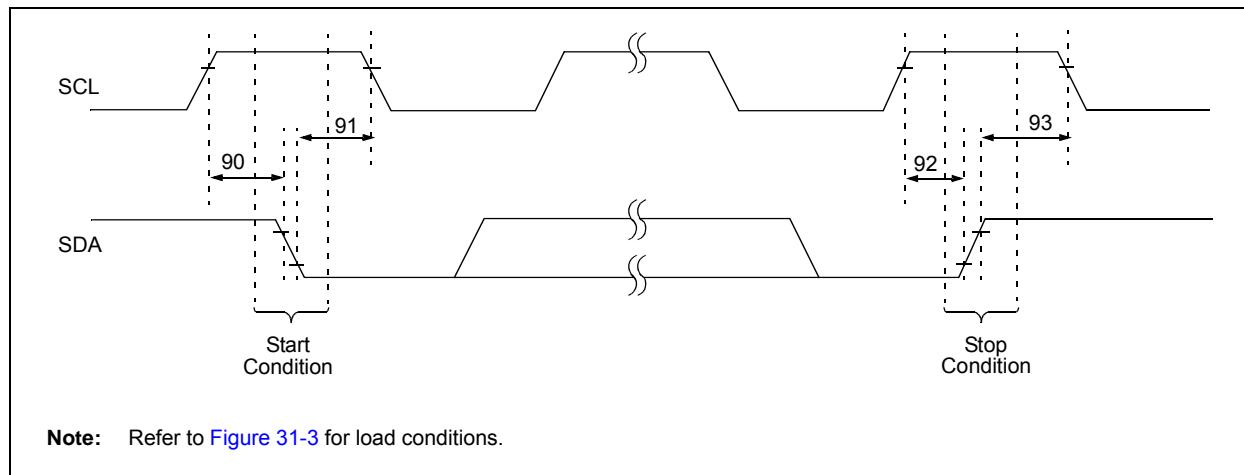
Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	4.0	—	μs	
			400 kHz mode	0.6	—	μs	
			MSSP module	1.5 TCY	—		
101	TLOW	Clock Low Time	100 kHz mode	4.7	—	μs	
			400 kHz mode	1.3	—	μs	
			MSSP module	1.5 TCY	—		
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4.7	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	μs	
91	THD:STA	Start Condition Hold Time	100 kHz mode	4.0	—	μs	After this period, the first clock pulse is generated
			400 kHz mode	0.6	—	μs	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	(Note 2)
			400 kHz mode	100	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4.7	—	μs	
			400 kHz mode	0.6	—	μs	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	(Note 1)
			400 kHz mode	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
D102	CB	Bus Capacitive Loading	—	—	400	pF	

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

**2:** A Fast mode I<sup>2</sup>C<sup>TM</sup> bus device can be used in a Standard mode I<sup>2</sup>C bus system, but the requirement, TSU:DAT ≥ 250 ns, must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line, TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I<sup>2</sup>C bus specification), before the SCL line is released.

# PIC18F66K80 FAMILY

**FIGURE 31-17: MSSP I<sup>2</sup>C™ BUS START/STOP BITS TIMING WAVEFORMS**

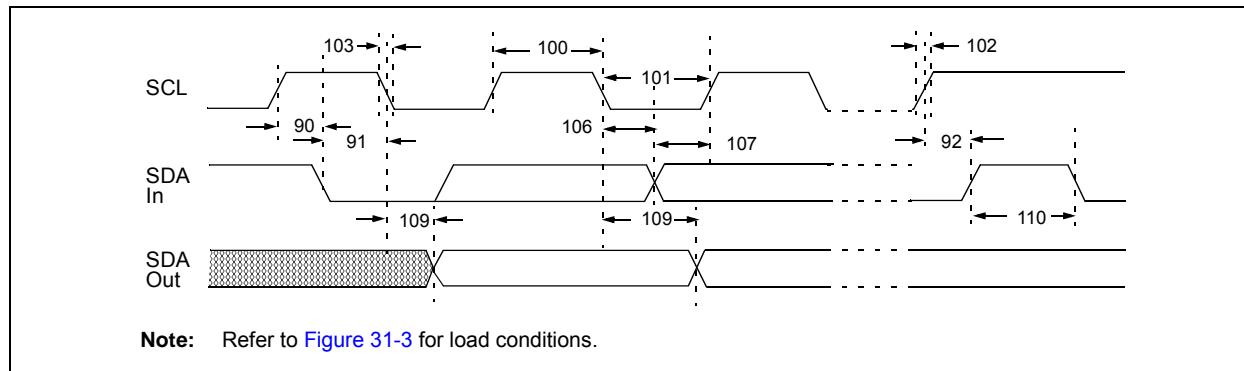


**TABLE 31-21: MSSP I<sup>2</sup>C™ BUS START/STOP BITS REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns Only relevant for Repeated Start condition
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns After this period, the first clock pulse is generated
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	
93	THD:STO	Stop Condition Hold Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	

Note 1: Maximum pin capacitance = 10 pF for all I<sup>2</sup>C™ pins.

**FIGURE 31-18: MSSP I<sup>2</sup>C™ BUS DATA TIMING**



# PIC18F66K80 FAMILY

TABLE 31-22: MSSP I<sup>2</sup>C™ BUS DATA REQUIREMENTS

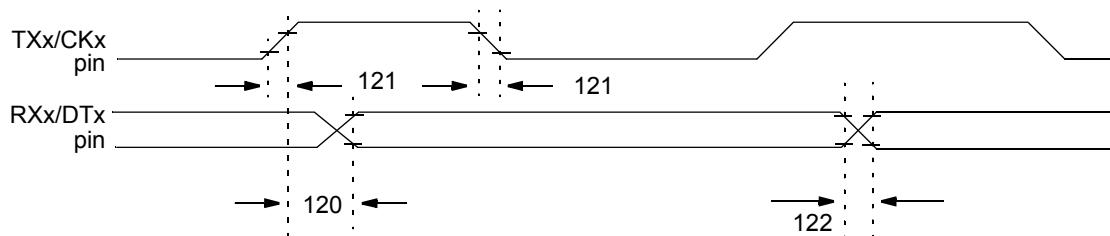
Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	2(Tosc)(BRG + 1)	—	—	
			400 kHz mode	2(Tosc)(BRG + 1)	—	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	—	
101	TLOW	Clock Low Time	100 kHz mode	2(Tosc)(BRG + 1)	—	—	
			400 kHz mode	2(Tosc)(BRG + 1)	—	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	—	
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	C <sub>B</sub> is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 C <sub>B</sub>	300	ns	
			1 MHz mode <sup>(1)</sup>	—	300	ns	
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	C <sub>B</sub> is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 C <sub>B</sub>	300	ns	
			1 MHz mode <sup>(1)</sup>	—	100	ns	
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	—	Only relevant for Repeated Start condition
			400 kHz mode	2(Tosc)(BRG + 1)	—	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	—	
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(Tosc)(BRG + 1)	—	—	After this period, the first clock pulse is generated
			400 kHz mode	2(Tosc)(BRG + 1)	—	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	—	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	—	
			400 kHz mode	0	0.9	μs	
			1 MHz mode <sup>(1)</sup>	—	μs	ns	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	(Note 2)
			400 kHz mode	100	—	ns	
			1 MHz mode <sup>(1)</sup>	—	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	—	
			400 kHz mode	2(Tosc)(BRG + 1)	—	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	—	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	
			400 kHz mode	—	1000	ns	
			1 MHz mode <sup>(1)</sup>	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
			1 MHz mode <sup>(1)</sup>	—	—	μs	
D102	CB	Bus Capacitive Loading		—	400	pF	

Note 1: Maximum pin capacitance = 10 pF for all I<sup>2</sup>C™ pins.

- 2: A Fast mode I<sup>2</sup>C bus device can be used in a Standard mode I<sup>2</sup>C bus system, but Parameter #107  $\geq$  250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line, Parameter #102 + Parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode), before the SCL line is released.

# PIC18F66K80 FAMILY

**FIGURE 31-19: EUSARTx SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

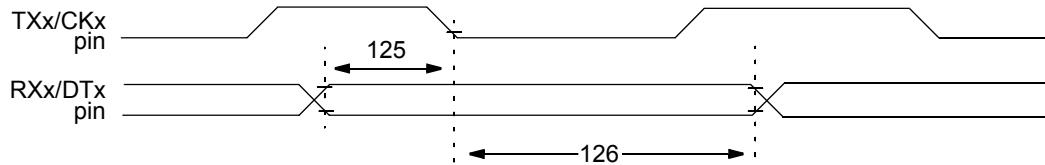


**Note:** Refer to [Figure 31-3](#) for load conditions.

**TABLE 31-23: EUSART/AUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	TckH2dtv	<u>SYNC XMIT (MASTER and SLAVE)</u> Clock High to Data Out Valid	—	40	ns	
121	Tckrf	Clock Out Rise Time and Fall Time (Master mode)	—	20	ns	
122	Tdtrf	Data Out Rise Time and Fall Time	—	20	ns	

**FIGURE 31-20: EUSART/AUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



**Note:** Refer to [Figure 31-3](#) for load conditions.

**TABLE 31-24: EUSART/AUSART SYNCHRONOUS RECEIVE REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	Tdtv2clk	<u>SYNC RCV (MASTER and SLAVE)</u> Data Hold before CKx ↓ (DTx hold time)	10	—	ns	
126	Tckl2dtl	Data Hold after CKx ↓ (DTx hold time)	15	—	ns	

# PIC18F66K80 FAMILY

---

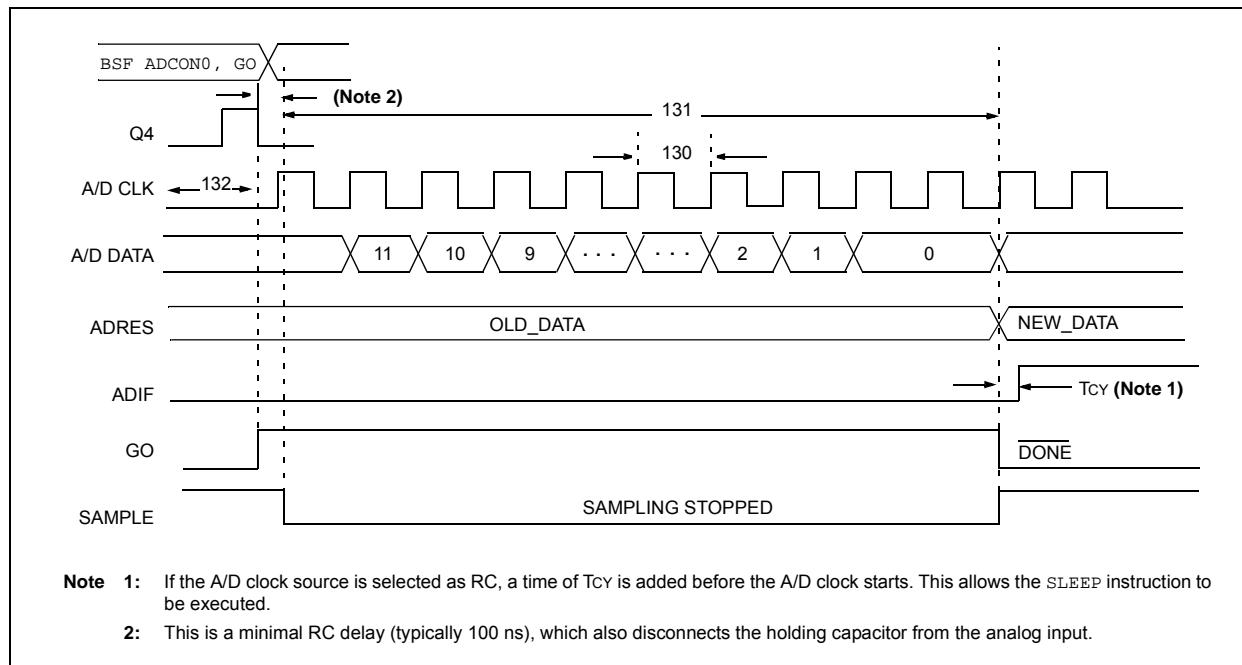
**TABLE 31-25: A/D CONVERTER CHARACTERISTICS: PIC18F66K80 FAMILY  
(INDUSTRIAL/EXTENDED)**

Param No.	Sym	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	—	—	12	bit	$\Delta V_{REF} \geq 3.0V$
A03	EIL	Integral Linearity Error	—	—	$\pm 6.0$	LSB	$V_{DD} = 5.0V$
A04	EDL	Differential Linearity Error	—	$\pm 1$	+3.0/-1.0	LSB	$V_{DD} = 5.0V$
A06	E <sub>OFF</sub>	Offset Error	—	$\pm 1$	$\pm 9$	LSB	$V_{DD} = 5.0V$
A07	E <sub>GN</sub>	Gain Error	—	$< \pm 1$	$\pm 8.00$	LSB	$V_{DD} = 5.0V$
A10	—	Monotonicity	Guaranteed <sup>(1)</sup>			—	$V_{SS} \leq V_{AIN} \leq V_{REF}$
A20	$\Delta V_{REF}$	Reference Voltage Range ( $V_{REFH} - V_{REFL}$ )	3	—	$V_{DD} - V_{SS}$	V	For 12-bit resolution
A21	$V_{REFH}$	Reference Voltage High	$V_{SS} + 3.0V$	—	$V_{DD} + 0.3V$	V	For 12-bit resolution
A22	$V_{REFL}$	Reference Voltage Low	$V_{SS} - 0.3V$	—	$V_{DD} - 3.0V$	V	For 12-bit resolution
A25	$V_{AIN}$	Analog Input Voltage	$V_{REFL}$	—	$V_{REFH}$	V	
A28	$V_{DD}$	Analog Supply Voltage	$V_{DD} - 0.3$	—	$V_{DD} + 0.3$	V	
A29	$V_{SS}$	Analog Supply Voltage	$V_{SS} - 0.3$	—	$V_{SS} + 0.3$	V	
A30	$Z_{AIN}$	Recommended Impedance of Analog Voltage Source	—	—	2.5	kΩ	
A50	I <sub>REF</sub>	$V_{REF}$ Input Current <sup>(2)</sup>	—	—	5 150	μA μA	During $V_{AIN}$ acquisition. During A/D conversion cycle.

- Note** 1: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.
- 2:  $V_{REFH}$  current is from the RA3/AN3/V<sub>REF+</sub> pin or  $V_{DD}$ , whichever is selected as the  $V_{REFH}$  source.  $V_{REFL}$  current is from the RA2/AN2/V<sub>REF-</sub>/CV<sub>REF</sub> pin or  $V_{SS}$ , whichever is selected as the  $V_{REFL}$  source.

# PIC18F66K80 FAMILY

**FIGURE 31-21: A/D CONVERSION TIMING**



**TABLE 31-26: A/D CONVERSION REQUIREMENTS**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
130	TAD	A/D Clock Period	0.8	12.5 <sup>(1)</sup>	μs	TOSC based, VREF ≥ 3.0V
			1.4	25 <sup>(1)</sup>	μs	VDD = 3.0V; TOSC based, VREF full range
			—	1	μs	A/D RC mode
			—	3	μs	VDD = 3.0V; A/D RC mode
131	TCNV	Conversion Time (not including acquisition time) <sup>(2)</sup>	14	15	TAD	
132	TACQ	Acquisition Time <sup>(3)</sup>	1.4	—	μs	-40°C to +125°C
135	TSWC	Switching Time from Convert → Sample	—	(Note 4)		
TBD	TDIS	Discharge Time	0.2	—	μs	-40°C to +125°C

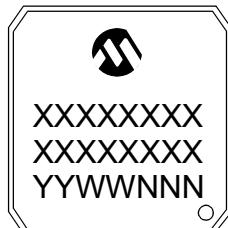
- Note 1:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.
- 2:** ADRES registers may be read on the following TCY cycle.
- 3:** The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (VDD to Vss or Vss to VDD). The source impedance ( $R_s$ ) on the input channels is 50Ω.
- 4:** On the following cycle of the device clock.

# PIC18F66K80 FAMILY

## 32.0 PACKAGING INFORMATION

### 32.1 Package Marking Information

28-Lead QFN



Example



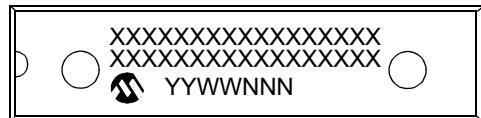
28-Lead SOIC



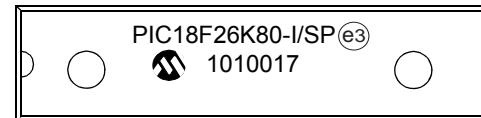
Example



28-Lead SPDIP



Example



28-Lead SSOP



Example



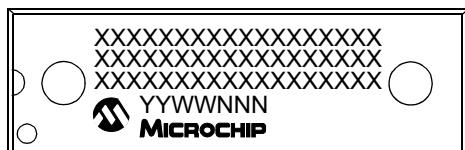
<b>Legend:</b>	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC designator for Matte Tin (Sn)
*		This package is Pb-free. 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.

# PIC18F66K80 FAMILY

## 32.1 Package Marking Information (Continued)

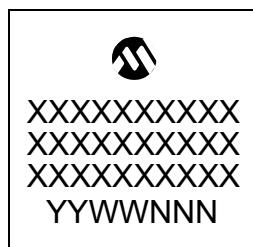
40-Lead PDIP



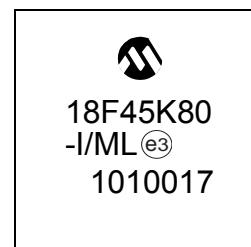
Example



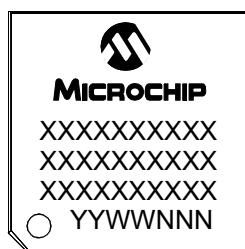
44-Lead QFN



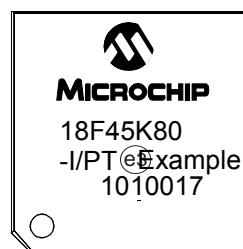
Example



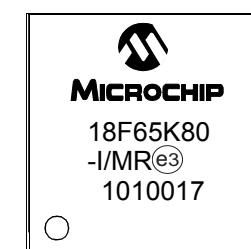
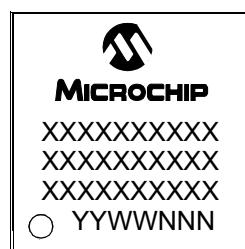
44-Lead TQFP



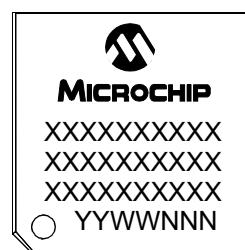
Example



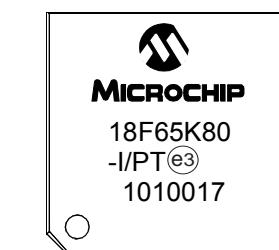
64-Lead QFN



64-Lead TQFP



Example

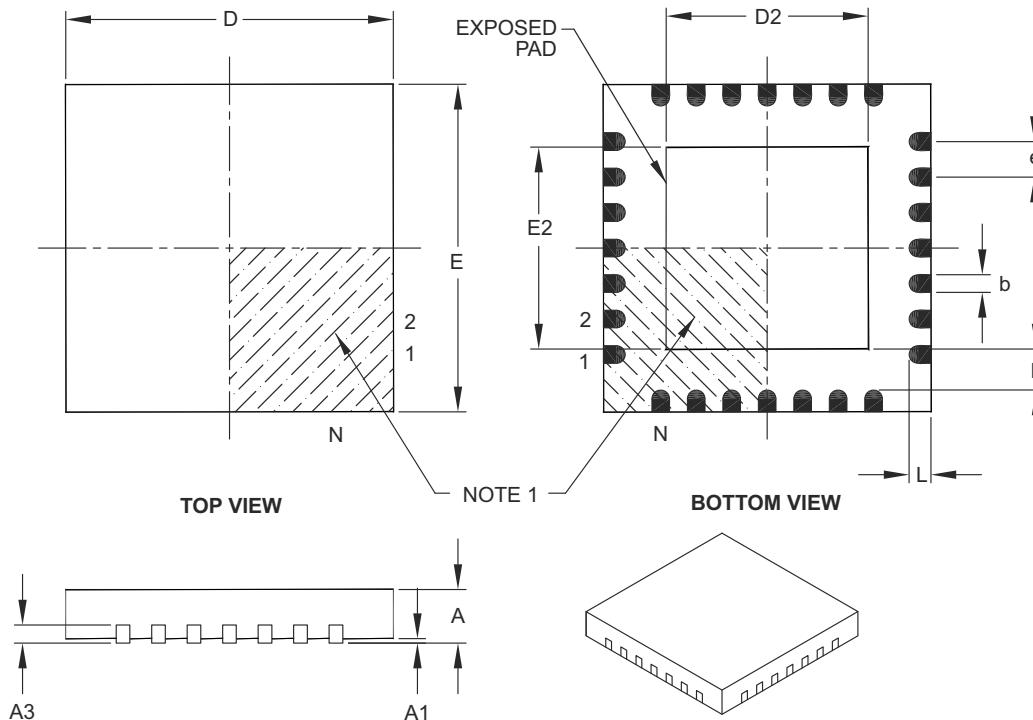


## 32.2 Package Details

The following sections give the technical details of the packages.

### 28-Lead Plastic Quad Flat, No Lead Package (MM) – 6x6x0.9 mm Body [QFN-S] with 0.40 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



		Units	MILLIMETERS		
Dimension Limits			MIN	NOM	MAX
Number of Pins	N		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	4.70	
Overall Length	D		6.00	BSC	
Exposed Pad Length	D2	3.65	3.70	4.70	
Contact Width	b	0.23	0.38	0.43	
Contact Length	L	0.30	0.40	0.50	
Contact-to-Exposed Pad	K	0.20	–	–	

#### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.

2. Package is saw singulated.

3. Dimensioning and tolerancing per ASME Y14.5M.

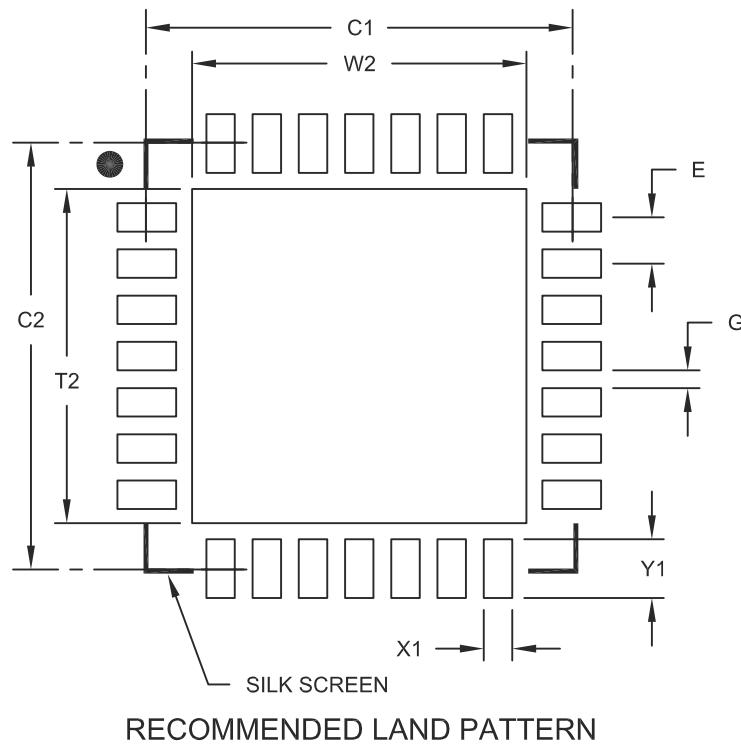
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

# PIC18F66K80 FAMILY

## 28-Lead Plastic Quad Flat, No Lead Package (MM) – 6x6x0.9 mm Body [QFN-S] with 0.40 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65	BSC
Optional Center Pad Width	W2			4.70
Optional Center Pad Length	T2			4.70
Contact Pad Spacing	C1		6.00	
Contact Pad Spacing	C2		6.00	
Contact Pad Width (X28)	X1			0.40
Contact Pad Length (X28)	Y1			0.85
Distance Between Pads	G	0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

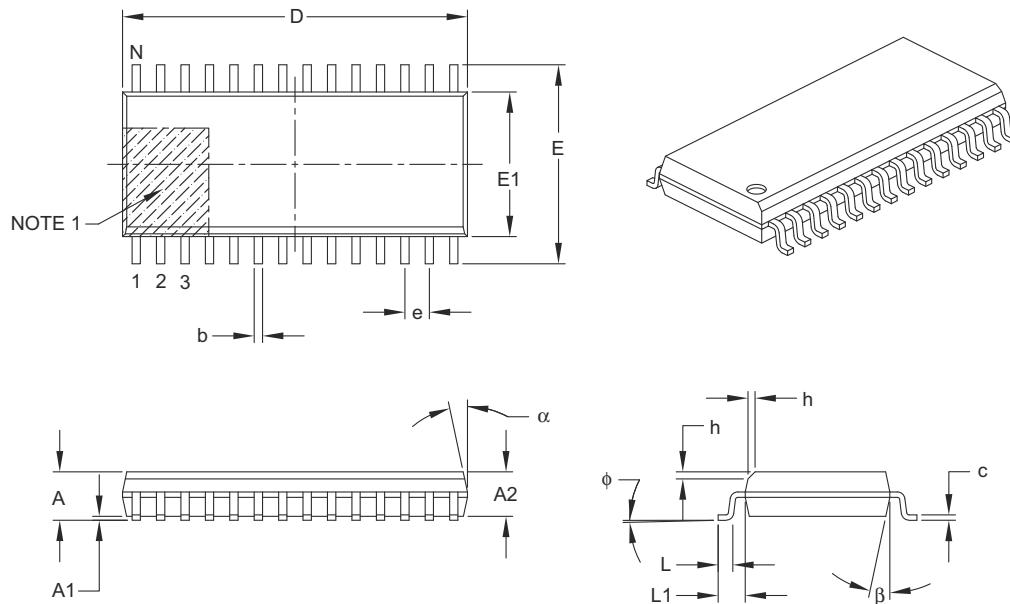
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2124A

# PIC18F66K80 FAMILY

## 28-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>



		Units	MILLIMETERS		
Dimension Limits			MIN	NOM	MAX
Number of Pins	N		28		
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		17.90 BSC		
Chamfer (optional)	h	0.25	—	0.75	
Foot Length	L	0.40	—	1.27	
Footprint	L1		1.40 REF		
Foot Angle Top	ϕ	0°	—	8°	
Lead Thickness	c	0.18	—	0.33	
Lead Width	b	0.31	—	0.51	
Mold Draft Angle Top	α	5°	—	15°	
Mold Draft Angle Bottom	β	5°	—	15°	

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

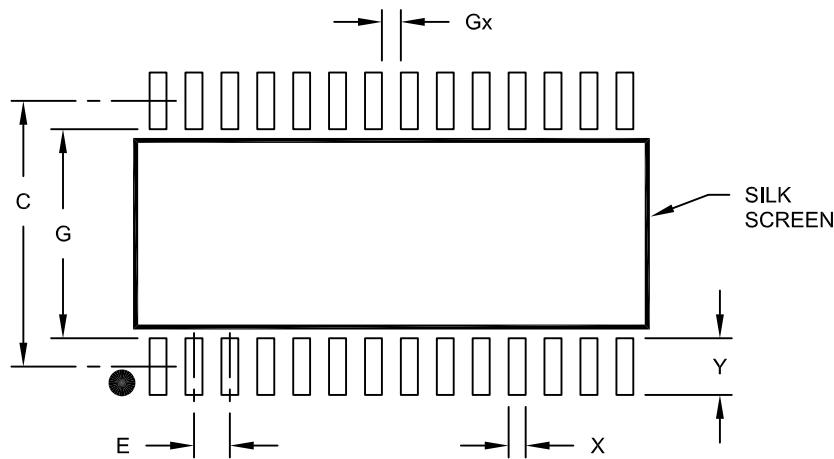
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-052B

# PIC18F66K80 FAMILY

28-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

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch		1.27 BSC		
Contact Pad Spacing	C		9.40	
Contact Pad Width (X28)	X			0.60
Contact Pad Length (X28)	Y			2.00
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.40		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

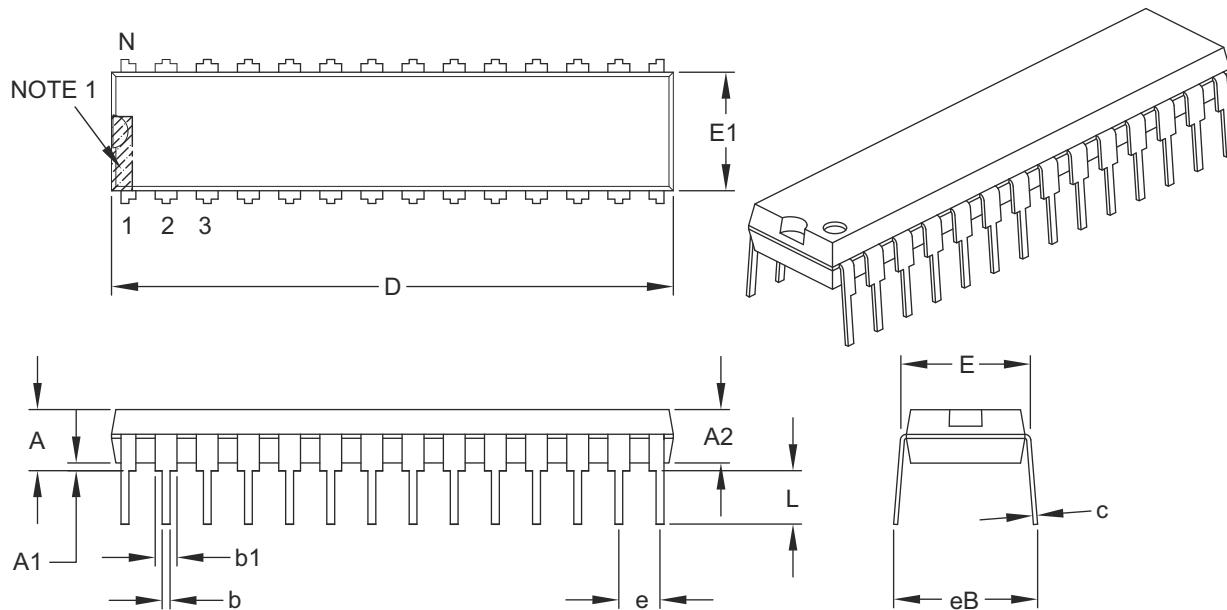
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

# PIC18F66K80 FAMILY

## 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

**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		28		
Pitch	e		.100 BSC		
Top to Seating Plane	A	—	—	.200	
Molded Package Thickness	A2	.120	.135	.150	
Base to Seating Plane	A1	.015	—	—	
Shoulder to Shoulder Width	E	.290	.310	.335	
Molded Package Width	E1	.240	.285	.295	
Overall Length	D	1.345	1.365	1.400	
Tip to Seating Plane	L	.110	.130	.150	
Lead Thickness	c	.008	.010	.015	
Upper Lead Width	b1	.040	.050	.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.

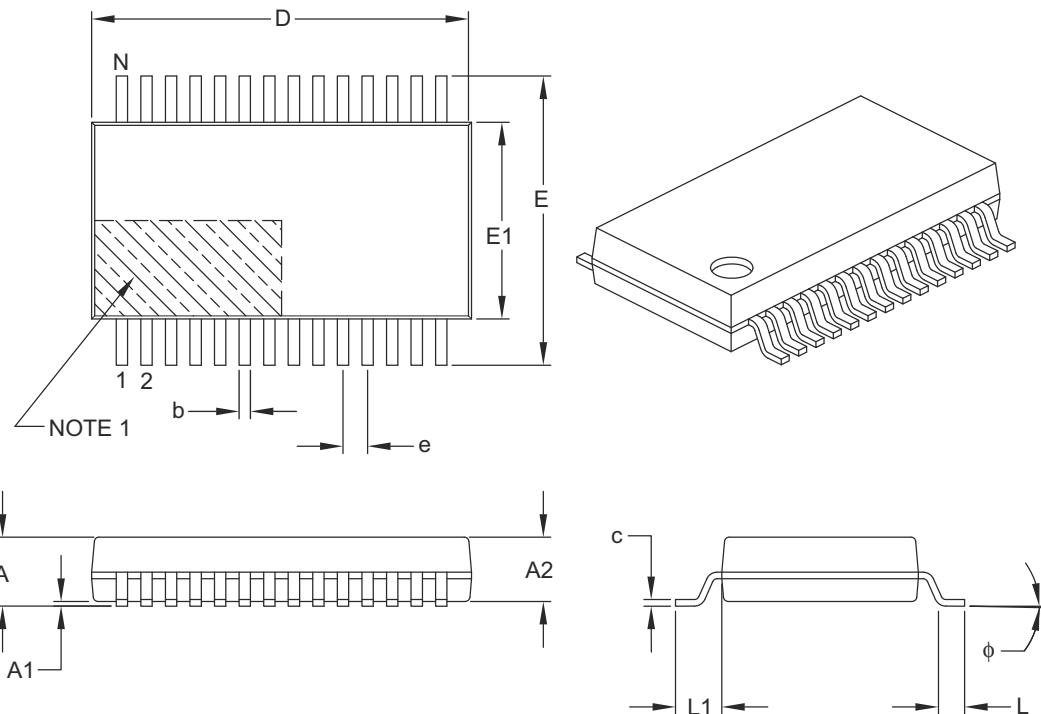
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

# PIC18F66K80 FAMILY

## 28-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>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins		28		
Pitch		0.65 BSC		
Overall Height		A	—	2.00
Molded Package Thickness		A2	1.65	1.75
Standoff		A1	0.05	—
Overall Width		E	7.40	7.80
Molded Package Width		E1	5.00	5.30
Overall Length		D	9.90	10.20
Foot Length		L	0.55	0.75
Footprint		L1	1.25 REF	
Lead Thickness		c	0.09	—
Foot Angle		φ	0°	4°
Lead Width		b	—	0.38

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

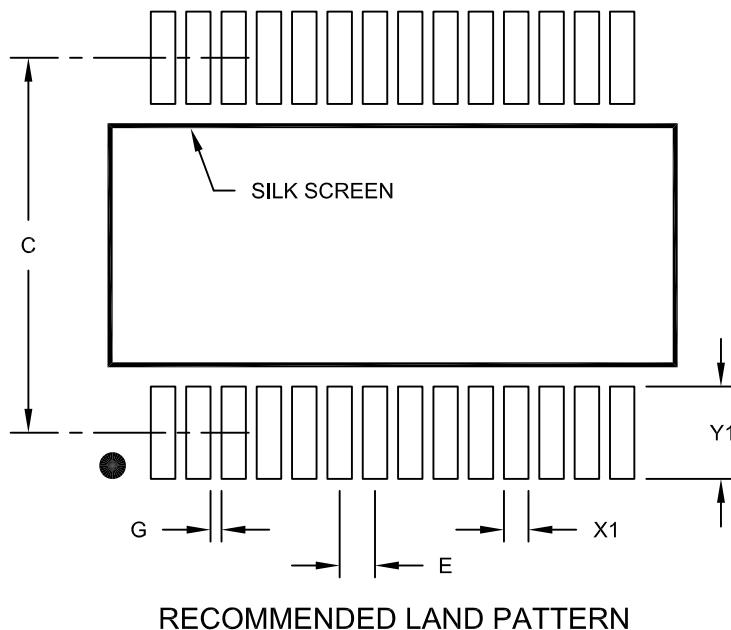
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B

# PIC18F66K80 FAMILY

## 28-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
Contact Pitch	E	0.65	BSC	
Contact Pad Spacing	C		7.20	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.75
Distance Between Pads	G	0.20		

### Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

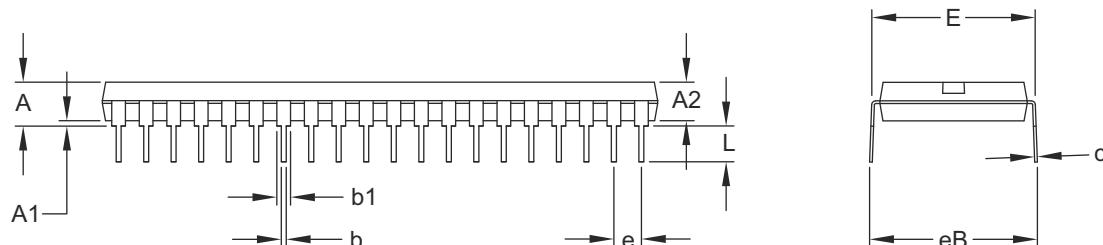
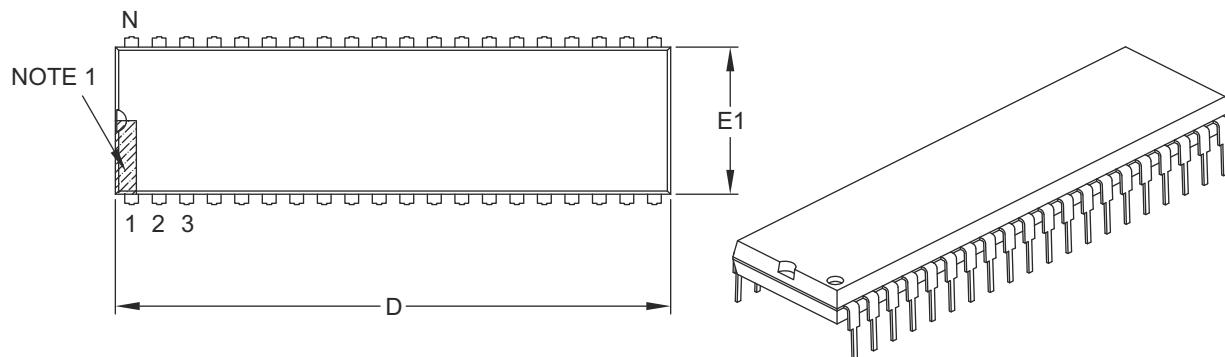
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

# PIC18F66K80 FAMILY

## 40-Lead Plastic Dual In-Line (P) – 600 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		40	
Pitch	e		.100 BSC	
Top to Seating Plane	A	—	—	.250
Molded Package Thickness	A2	.125	—	.195
Base to Seating Plane	A1	.015	—	—
Shoulder to Shoulder Width	E	.590	—	.625
Molded Package Width	E1	.485	—	.580
Overall Length	D	1.980	—	2.095
Tip to Seating Plane	L	.115	—	.200
Lead Thickness	c	.008	—	.015
Upper Lead Width	b1	.030	—	.070
Lower Lead Width	b	.014	—	.023
Overall Row Spacing §	eB	—	—	.700

### 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.

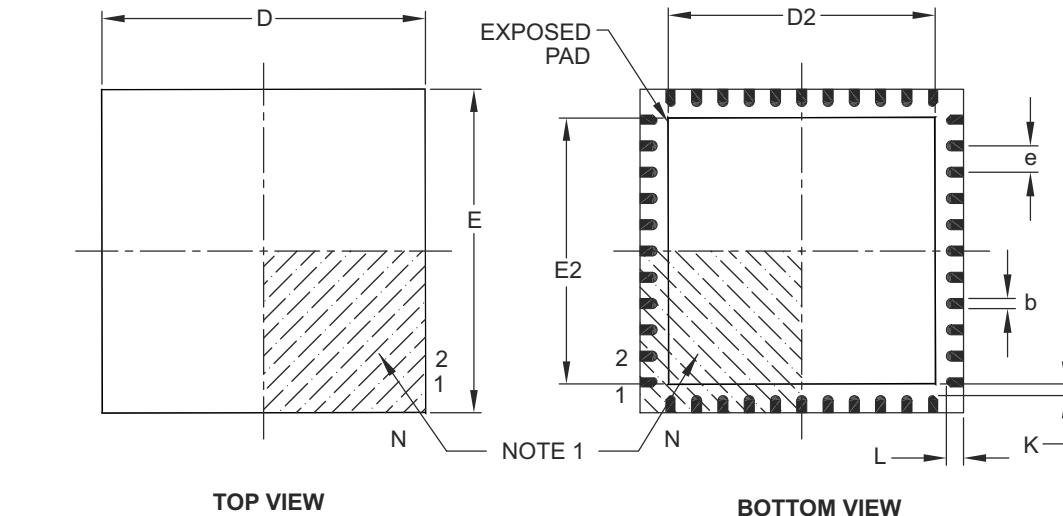
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-016B

# PIC18F66K80 FAMILY

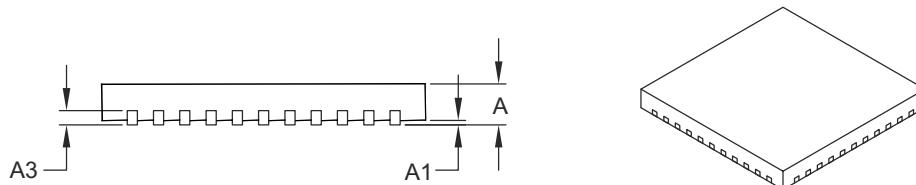
## 44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



TOP VIEW

BOTTOM VIEW



		Units	MILLIMETERS		
Dimension Limits			MIN	NOM	MAX
Number of Pins	N		44		
Pitch	e		0.65	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		8.00	BSC	
Exposed Pad Width	E2	6.30	6.45	6.80	
Overall Length	D		8.00	BSC	
Exposed Pad Length	D2	6.30	6.45	6.80	
Contact Width	b	0.25	0.30	0.38	
Contact Length	L	0.30	0.40	0.50	
Contact-to-Exposed Pad	K	0.20	–	–	

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

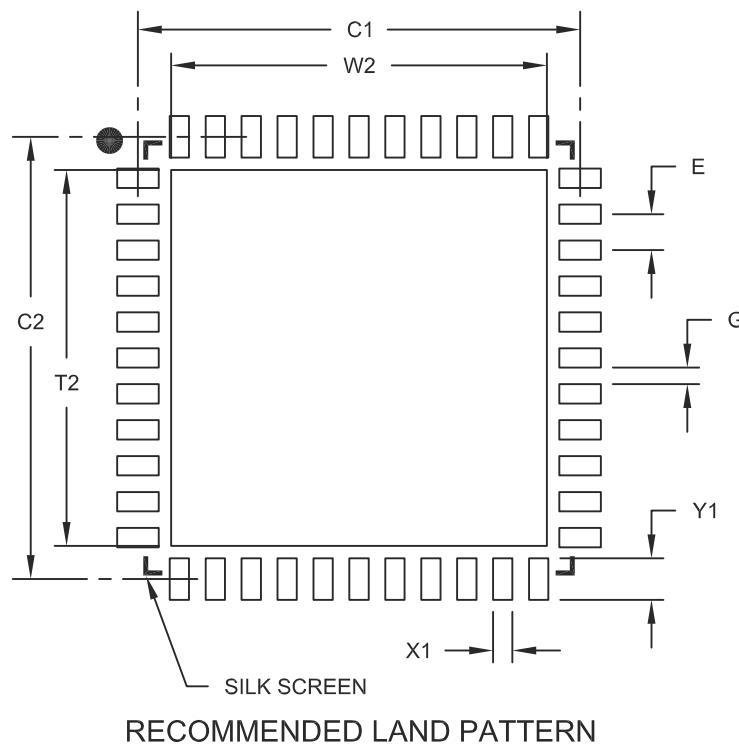
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103B

# PIC18F66K80 FAMILY

## 44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Optional Center Pad Width	W2			6.80
Optional Center Pad Length	T2			6.80
Contact Pad Spacing	C1		8.00	
Contact Pad Spacing	C2		8.00	
Contact Pad Width (X44)	X1			0.35
Contact Pad Length (X44)	Y1			0.80
Distance Between Pads	G	0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

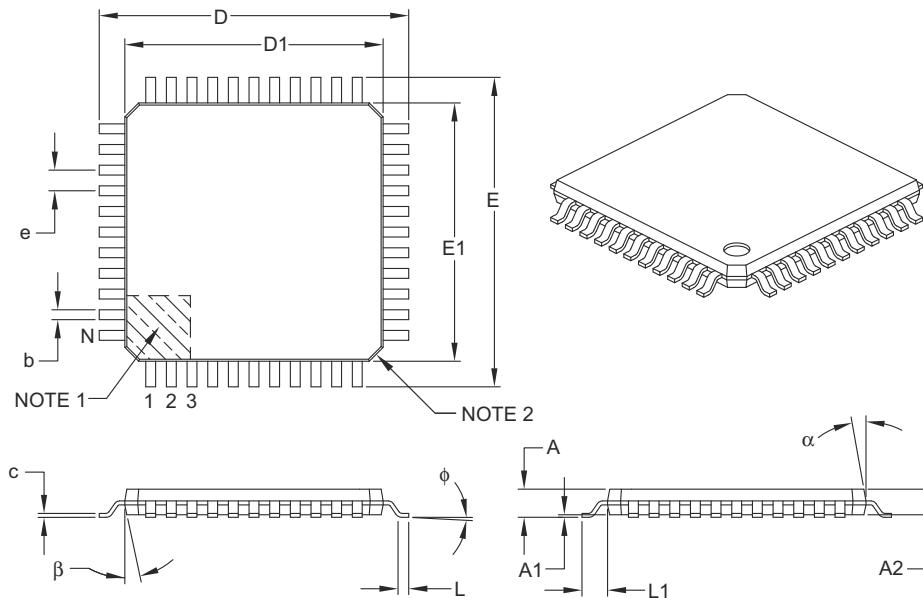
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2103A

# PIC18F66K80 FAMILY

## 44-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension	Limits	MILLIMETERS		
		MIN	NOM	MAX
Number of Leads	N		44	
Lead Pitch	e		0.80 BSC	
Overall Height	A	—	—	1.20
Molded Package Thickness	A2	0.95	1.00	1.05
Standoff	A1	0.05	—	0.15
Foot Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Foot Angle	φ	0°	3.5°	7°
Overall Width	E		12.00 BSC	
Overall Length	D		12.00 BSC	
Molded Package Width	E1		10.00 BSC	
Molded Package Length	D1		10.00 BSC	
Lead Thickness	c	0.09	—	0.20
Lead Width	b	0.30	0.37	0.45
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

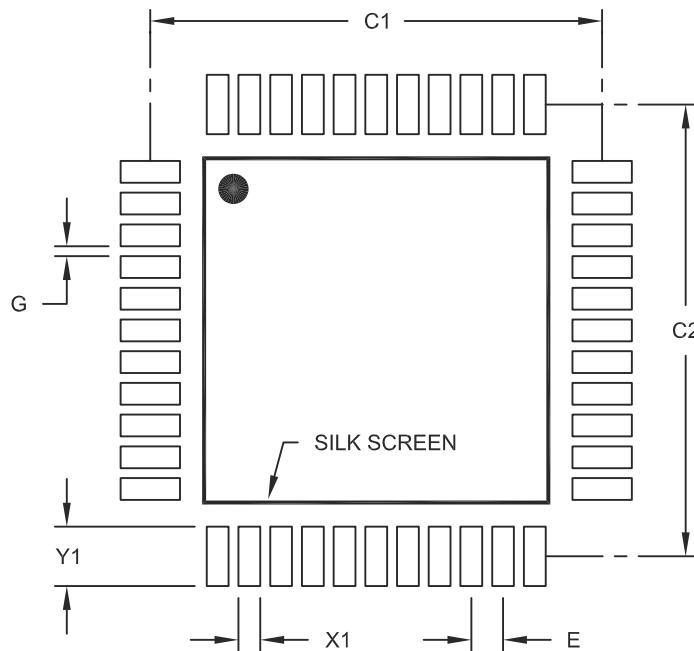
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-076B

# PIC18F66K80 FAMILY

## 44-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

		Units	MILLIMETERS		
Dimension Limits			MIN	NOM	MAX
Contact Pitch	E		0.80 BSC		
Contact Pad Spacing	C1			11.40	
Contact Pad Spacing	C2			11.40	
Contact Pad Width (X44)	X1				0.55
Contact Pad Length (X44)	Y1				1.50
Distance Between Pads	G	0.25			

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

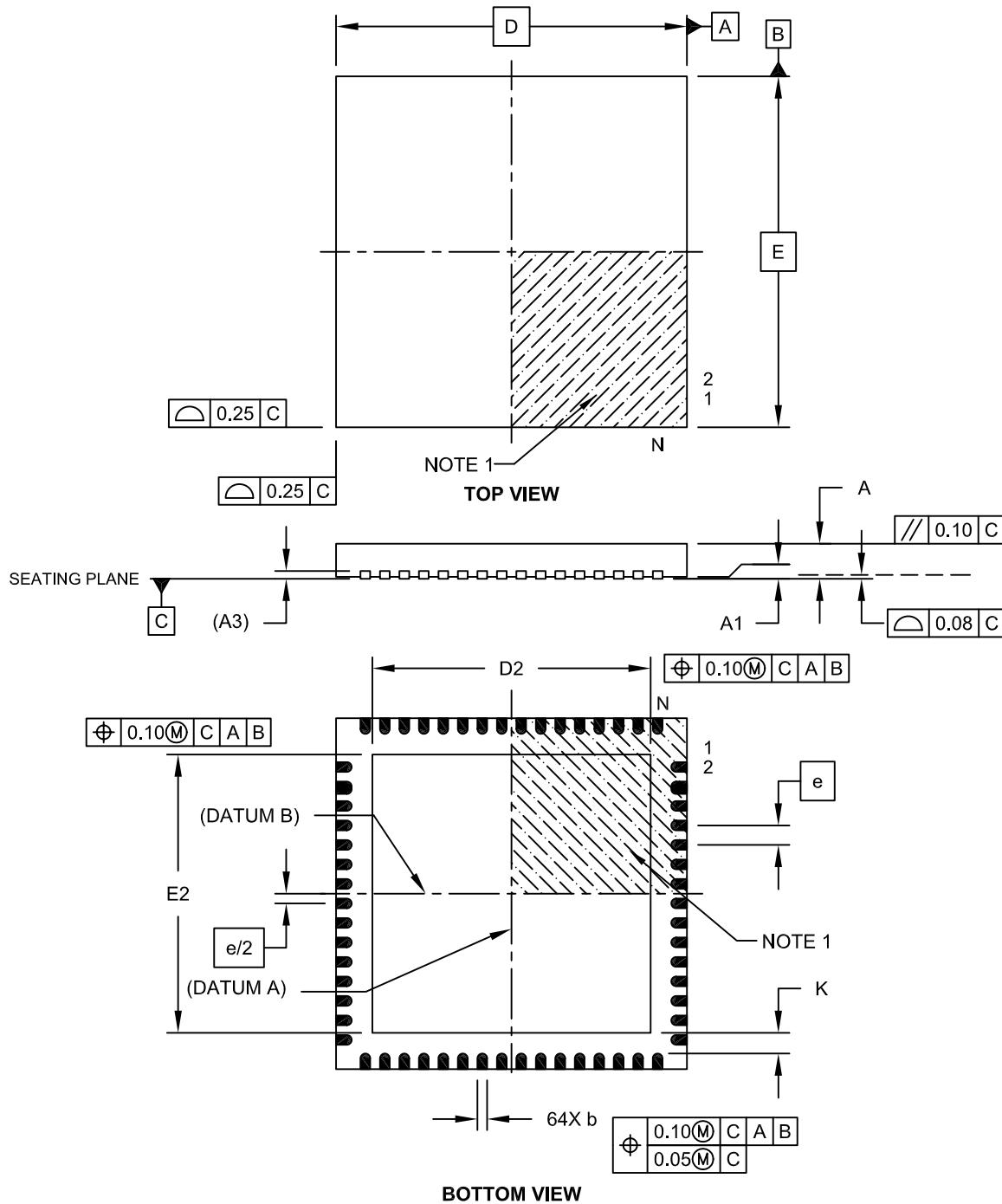
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076A

# **PIC18F66K80 FAMILY**

## **64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

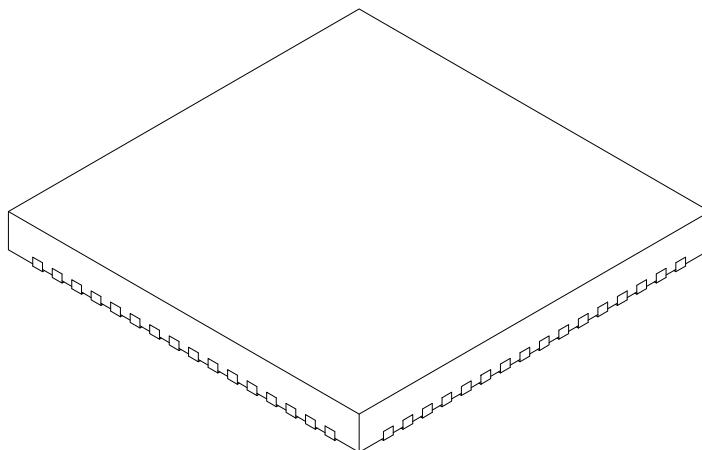


Microchip Technology Drawing C04-149B Sheet 1 of 2

# PIC18F66K80 FAMILY

## 64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	64		
Pitch	e	0.50	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	9.00	BSC	
Exposed Pad Width	E2	7.05	7.15	7.50
Overall Length	D	9.00	BSC	
Exposed Pad Length	D2	7.05	7.15	7.50
Contact Width	b	0.18	0.25	0.30
Contact Length	L	0.30	0.40	0.50
Contact-to-Exposed Pad	K	0.20	-	-

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

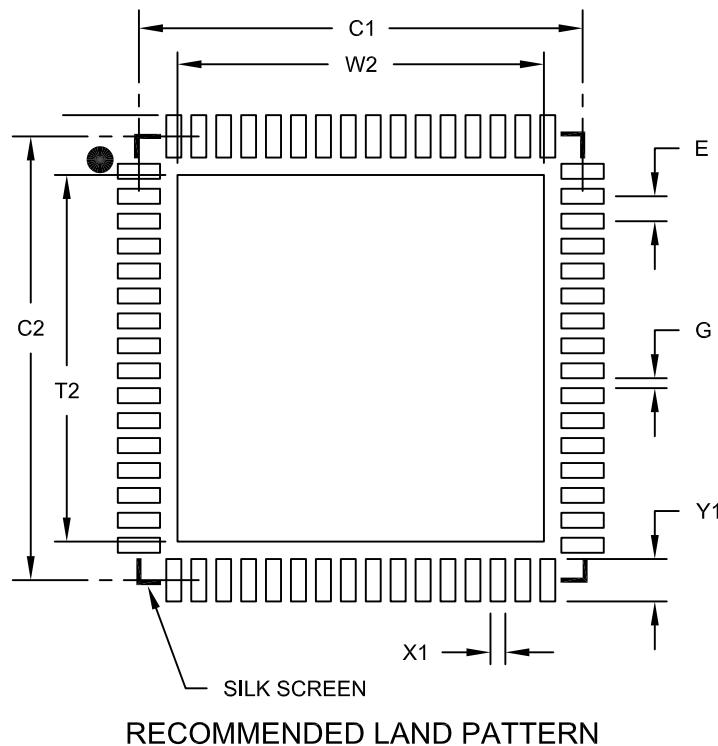
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-149B Sheet 2 of 2

# PIC18F66K80 FAMILY

64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]  
With 0.40 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
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	W2			7.35
Optional Center Pad Length	T2			7.35
Contact Pad Spacing	C1		8.90	
Contact Pad Spacing	C2		8.90	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			0.85
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

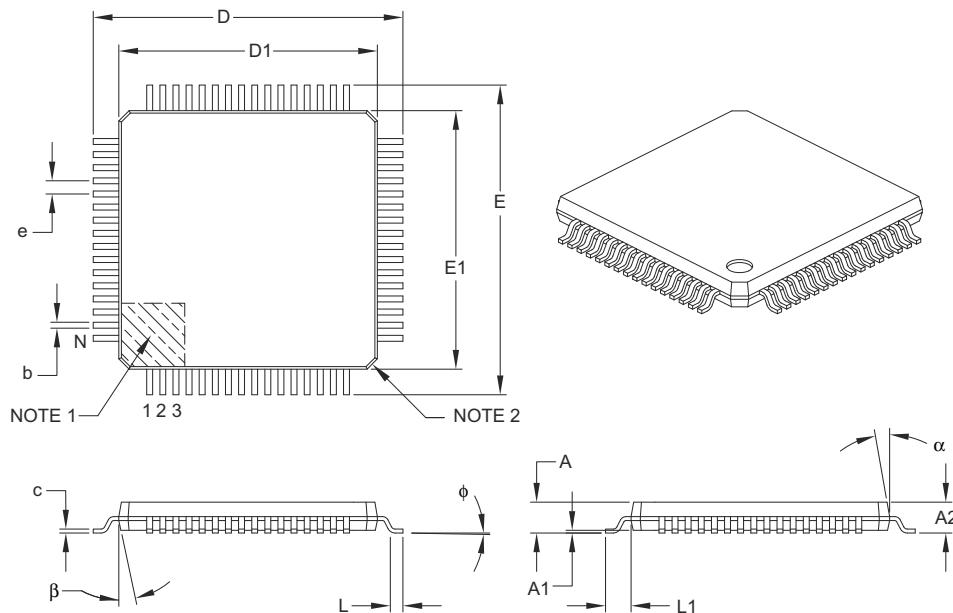
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2149A

# PIC18F66K80 FAMILY

## 64-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Leads	N	64		
Lead Pitch	e	0.50	BSC	
Overall Height	A	–	–	1.20
Molded Package Thickness	A2	0.95	1.00	1.05
Standoff	A1	0.05	–	0.15
Foot Length	L	0.45	0.60	0.75
Footprint	L1	1.00 REF		
Foot Angle	φ	0°	3.5°	7°
Overall Width	E	12.00 BSC		
Overall Length	D	12.00 BSC		
Molded Package Width	E1	10.00 BSC		
Molded Package Length	D1	10.00 BSC		
Lead Thickness	c	0.09	–	0.20
Lead Width	b	0.17	0.22	0.27
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

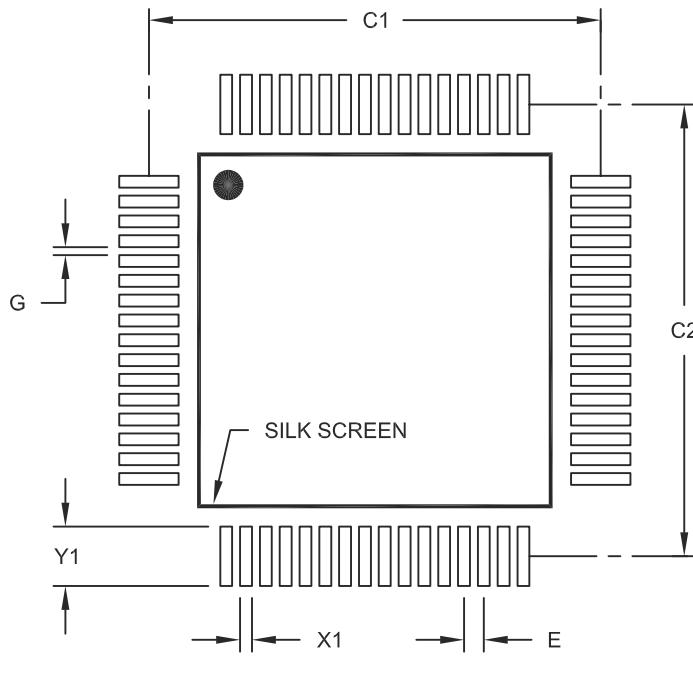
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085B

# PIC18F66K80 FAMILY

## 64-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**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		0.50	BSC
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			1.50
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. C04-2085A

# **PIC18F66K80 FAMILY**

---

---

## **NOTES:**

## APPENDIX A: REVISION HISTORY

### Revision A (August 2010)

Original data sheet for PIC18F66K80 family devices.

### Revision B (December 2010)

Changes to [Section 31.0 “Electrical Characteristics”](#) and minor text edits throughout document.

### Revision C (January 2011)

[Section 2.0 “Guidelines for Getting Started with PIC18FXXKXX Microcontrollers”](#) was added to the data sheet. Changes to [Section 31.0 “Electrical Characteristics”](#) for PIC18F66K80 family devices. Minor text edits throughout document.

### Revision D (November 2011)

Preliminary conditions have been deleted from document.

### Revision E (February 2012)

Added all Data Sheet erratas. Added Current Injection specifications to [Section 31.0 “Electrical Characteristics”](#).

### Revision F (February 2012)

Updated the Reset value for the IOCB register.

# PIC18F66K80 FAMILY

---

## APPENDIX B: MIGRATION TO PIC18F66K80 FAMILY

Devices in the PIC18F66K80, PIC18F4580, PIC18F4680 and 18F8680 families are similar in their functions and features. Code can be migrated from the

other families to the PIC18F66K80 without many changes. The differences between the device families are listed in [Table B-1](#) and [Table B-2](#). For more details on migrating to the PIC18F66K80, refer to "*PIC18FXX80 to PIC18FXXK80 Migration Guide*" (DS39982).

**TABLE B-1: NOTABLE DIFFERENCES BETWEEN 28, 40 AND 44-PIN DEVICES – PIC18F66K80, PIC18F4580 AND PIC18F4680 FAMILIES**

Characteristic	PIC18F66K80 Family	PIC18F4680 Family	PIC18F4580 Family
Max Operating Frequency	64 MHz	40 MHz	40 MHz
Max Program Memory	64 Kbytes	64 Kbytes	32 Kbytes
Data Memory (bytes)	3,648	3,328	1,536
CTMU	Yes	No	No
SOSC Oscillator Options	Low-power oscillator option for SOSC	No options	No options
T1CKI Clock	T1CKI can be used as a clock without enabling the SOSC oscillator	No	No
INTOSC	Up to 16 MHz	Up to 8 MHz	Up to 8 MHz
Timers	Two 8-bit, three 16-bit	One 8-bit, three 16-bit	One 8-bit, three 16-bit
ECCP	One for all devices	40 and 44-pin devices – One 28-pin devices – None	40 and 44-pin devices – One 28-pin devices – None
CCP	Four	One	One
Data EEPROM (bytes)	1,024	1,024	256
WDT Prescale Options	22	16	16
5V Operation	18FXXK80 parts – 5V operation 18LFXXX80 parts – 3.3V operation	Yes	Yes
nanoWatt XLP	Yes	No	No
Regulator	18FXXK80 parts – Yes 18LFXXX80 parts – No	No	No
Low-Power BOR	Yes	No	No
A/D Converter	12-bit signed differential	10-bit	10-bit
A/D Channels	28-pin devices – 8 Channels 40 and 44-pin devices – 11 Channels	8 Channels for 28-pin devices/ 11 Channels for 40 and 44-pin devices	8 Channels for 28-pin devices/ 11 Channels for 40 and 44-pin devices
Internal Temp Sensor	Yes	No	No
EUSART	Two	One	One
Comparators	Two	28-pin devices – None 40 and 44-pin devices – Two	28-pin devices – None 40 and 44-pin devices – Two
Oscillator Options	14	Nine	Nine
Ultra Low-Power Wake-up (ULPW)	Yes	No	No
Adjustable Slew Rate for I/O	Yes	No	No
PLL	Available for all oscillator options	Available only for high-speed crystal and internal oscillator	Available only for high-speed crystal and internal oscillator
TXM Modulator	No	No	No

# PIC18F66K80 FAMILY

---

**TABLE B-2: NOTABLE DIFFERENCES BETWEEN 64-PIN DEVICES – PIC18F66K80 AND PIC18F8680 FAMILIES**

Characteristic	PIC18F66K80 Family	PIC18F8680 Family
Max Operating Frequency	64 MHz	40 MHz
Max Program Memory	64K	64K
Data Memory (bytes)	3,648	3,328
CTMU	Yes	No
SOSC Oscillator Options	Low-power oscillator option for SOSC	No options
T1CKI Clock	T1CKI can be used as a clock without enabling the SOSC oscillator	No
INTOSC	Up to 16 MHz	No Internal Oscillator
SPI/I <sup>2</sup> C™	1 Module	1 Module
Timers	Two 8-bit, Three 16-bit	Two 8-bit, Three 16-bit
ECCP	1	1
CCP	4	1
Data EEPROM (bytes)	1,024	1,024
WDT Prescale Options	22	16
5V Operation	18FXXK80 parts – 5V operation 18LFXXK80 parts – 3.3V operation	Yes
nanoWatt XLP	Yes	No
On-Chip 3.3V Regulator	18FXXK80 parts – Yes 18LFXXK80 parts – No	No
Low-Power BOR	Yes	No
A/D Converter	12-bit signed differential	10-bit
A/D Channels	15 Channels	12 Channels
Internal Temp Sensor	Yes	No
EUSART	Two	One
Comparators	Two	Two
Oscillator Options	14	Seven
Ultra Low-Power Wake-up (ULPW)	Yes	No
Adjustable Slew Rate for I/O	Yes	No
PLL	Available for all oscillator options	Available for only high-speed crystal and external oscillator
Data Signal Modulator	Yes	No

# **PIC18F66K80 FAMILY**

---

---

## **NOTES:**

## INDEX

### A

A/D .....	357
A/D Converter Interrupt, Configuring .....	366
Acquisition Requirements .....	367
ADRESH Register.....	364
Analog Port Pins, Configuring .....	368
Associated Registers .....	371
Automatic Acquisition Time.....	368
Configuring the Module .....	366
Conversion Clock (TAD) .....	368
Conversion Requirements .....	580
Conversion Status (GO/DONE Bit).....	364
Conversions .....	369
Converter Characteristics .....	579
Differential Converter .....	357
Operation in Power-Managed Modes .....	370
Use of the Special Event Triggers .....	370
Absolute Maximum Ratings .....	537
AC (Timing) Characteristics .....	560
Load Conditions for Device Timing Specifications....	561
Parameter Symbology .....	560
Temperature and Voltage Specifications .....	561
Timing Conditions .....	561
ACKSTAT .....	322
ACKSTAT Status Flag .....	322
ADC0N0 Register	
GO/DONE Bit.....	364
ADDFSR .....	526
ADDLW .....	489
ADDULNK .....	526
ADDWFW .....	489
ADDWFC .....	490
ADRESL Register .....	364
Analog-to-Digital Converter. See A/D.	
ANDLW .....	490
ANDWF .....	491
Assembler	
MPASM Assembler .....	534
Auto-Wake-up on Sync Break Character .....	348

### B

Baud Rate Generator .....	318
BC .....	491
BCF .....	492
BF .....	322
BF Status Flag .....	322
Bit Timing Configuration Registers	
BRGCON1 .....	452
BRGCON2 .....	452
BRGCON3 .....	452
Block Diagrams	
A/D .....	365
Analog Input Model .....	366
Baud Rate Generator .....	318
CAN Buffers and Protocol Engine.....	392
Capture Mode Operation .....	257, 269
Comparator Analog Input Model .....	376
Comparator Configurations .....	378
Comparator Module .....	373
Comparator Voltage Reference .....	382
Comparator Voltage Reference Output Buffer.....	383
Compare Mode Operation .....	260, 270
Connections for On-Chip Voltage Regulator.....	474

Crystal/Ceramic Resonator Operation (HS, HSPLL ...	58
CTMU .....	235
CTMU Current Source Calibration Circuit .....	241
CTMU Temperature Measurement Circuit .....	249
CTMU Typical Connections and Internal Configuration for Pulse Delay Generation .....	250
CTMU Typical Connections and Internal Configuration for Time Measurement .....	248
Data Signal Modulator .....	196
Device Clock.....	52
Differential Channel Measurement.....	357
EUSART Receive .....	346
EUSART Transmit .....	343
External Components for the SOSC Oscillator.....	214
External Power-on Reset Circuit (Slow VDD Power-up) .	
81	
Fail-Safe Clock Monitor (FSCM) .....	477
Full-Bridge Application.....	275
Generic I/O Port Operation .....	171
Half-Bridge Applications .....	274, 281
High/Low-Voltage Detect with External Input .....	386
Interrupt Logic.....	148
INTIO1 Oscillator Mode .....	60
INTIO2 Oscillator Mode .....	60
MSSP (I <sup>2</sup> C Master Mode).....	316
MSSP (I <sup>2</sup> C Mode).....	296
MSSP (SPI Mode) .....	287
On-Chip Reset Circuit .....	79
PIC18F2XK80.....	15
PIC18F4XK80.....	16
PIC18F6XK80.....	17
PLL .....	59
PORTD and PORTE (Parallel Slave Port).....	192
PWM (Enhanced Mode) .....	271
PWM Operation (Simplified) .....	262
RC Oscillator Mode .....	57
RCIO Oscillator Mode .....	57
Reads from Flash Program Memory .....	133
Simplified Steering.....	284
Single Channel Measurement .....	357
Single Comparator .....	376
Table Read Operation .....	129
Table Write Operation .....	130
Table Writes to Flash Program Memory .....	135
Timer0 in 16-Bit Mode .....	206
Timer0 in 8-Bit Mode .....	206
Timer1 .....	213
Timer2 .....	222
Timer3 .....	226
Timer4 .....	234
Transmit Buffers .....	442
Ultra Low-Power Wake-up Initialization .....	77
Using Open-Drain Output .....	173
Watchdog Timer .....	472
BN .....	492
BNC .....	493
BNN .....	493
BNOV .....	494
BNZ .....	494
BOR. See Brown-out Reset.	
BOV .....	497
BRA .....	495
Break Character (12-Bit) Transmit and Receive .....	350

# PIC18F66K80 FAMILY

---

BRG. See Baud Rate Generator.	
Brown-out Reset (BOR) .....	82
Detection .....	82
Disabling in Sleep Mode .....	82
Software Enabled.....	82
BSF .....	495
BTFSC .....	496
BTFS... <td>496</td>	496
BTG .....	497
BZ.....	498
<b>C</b>	
C Compilers	
MPLAB C18 .....	534
CALL .....	498
CALLW .....	527
CAN Module	
External-Internal Clock in HS-PLL Based Oscillators	447
Capture (CCP Module).....	257
CCP Pin Configuration.....	257
CCPRxH:CCPRxL Registers .....	257
Software Interrupt .....	258
Timer1/3 Mode Selection .....	257
Capture (ECCP Module) .....	268
CCPR1H:CCPR1L Registers .....	268
ECCP Pin Configuration .....	268
Prescaler.....	269
Software Interrupt .....	269
Timer1/2/3/4 Mode Selection .....	269
Capture, Compare, Timer1/3	
Associated Registers .....	261
Capture/Compare/PWM (CCP).....	253
Capture Mode. See Capture.	
CCP Mode and Timer Resources .....	256
CCPRxH Register .....	256
CCPRxL Register.....	256
Compare Mode. See Compare.	
Configuration.....	256
Open-Drain Output Option .....	256
Charge Time Measurement Unit (CTMU).....	235
Associated Registers .....	252
Calibrating the Module .....	240
Creating a Delay .....	250
Effects of a Reset.....	252
Measuring Capacitance .....	246
Measuring Time .....	248
Module Initialization .....	240
Operation .....	239
During Sleep, Idle Modes.....	252
Temperature Measurement .....	249
Clock Sources .....	56
Default System Clock on Reset .....	57
Selection Using OSCCON Register .....	56
CLRF .....	499
CLRWD... <td>499</td>	499
Code Examples	
16 x 16 Signed Multiply Routine .....	146
16 x 16 Unsigned Multiply Routine .....	146
8 x 8 Signed Multiply Routine .....	145
8 x 8 Unsigned Multiply Routine .....	145
Capacitance Calibration Routine .....	245
Changing Between Capture Prescalers.....	258, 269
Changing to Configuration Mode .....	396
Computed GOTO Using an Offset Value.....	105
Current Calibration Routine .....	243
Data EEPROM Read .....	142
Data EEPROM Refresh Routine.....	143
Data EEPROM Write .....	142
Erasing a Flash Program Memory Row.....	134
Fast Register Stack .....	105
How to Clear RAM (Bank 1) Using Indirect Addressing .	
123	
Initializing PORTA.....	175
Initializing PORTB.....	177
Initializing PORTC .....	181
Initializing PORTD .....	184
Initializing PORTE.....	187
Initializing PORTF .....	189
Initializing PORTG .....	190
Loading the SSPBUF (SSPSR) Register.....	290
Reading a CAN Message .....	412
Reading a Flash Program Memory Word .....	133
Routine for Capacitive Touch Switch.....	247
Routine for Temperature Measurement Using Internal	
Diode .....	249, 251
Saving STATUS, WREG and BSR Registers in RAM ....	
169	
Setup for CTMU Calibration Routines .....	242
Transmitting a CAN Message Using Banked Method ....	
404	
Transmitting a CAN Message Using WIN Bits.....	405
Ultra Low-Power Wake-up Initialization .....	77
WIN and ICODE Bits Usage in Interrupt Service Routine	
to Access TX/RX Buffers .....	396
Writing to Flash Program Memory .....	136–137
Code Protection .....	457
COMF .....	500
Comparator .....	373
Analog Input Connection Considerations .....	376
Associated Registers .....	380
Configuration .....	377
Control .....	377
Effects of a Reset .....	380
Enable and Input Selection .....	377
Enable and Output Selection .....	377
Interrupts .....	379
Operation .....	376
Operation During Sleep .....	380
Response Time .....	376
Comparator Specifications.....	559
Comparator Voltage Reference .....	381
Accuracy and Error .....	382
Associated Registers .....	383
Configuring .....	381
Connection Considerations.....	382
Effects of a Reset .....	382
Operation During Sleep .....	382
Compare (CCP Module) .....	259
CCP Pin Configuration.....	259
Software Interrupt .....	259
Special Event Trigger .....	259
Timer1/3 Mode Selection .....	259
Compare (ECCP Module).....	270
CCPR1 Register .....	270
Pin Configuration .....	270
Software Interrupt .....	270
Special Event Trigger .....	232, 270
Timer1/2/3 Mode Selection .....	270
Computed GOTO.....	105
Configuration Bits .....	457
Configuration Mismatch (CM) Reset .....	83

# PIC18F66K80 FAMILY

Configuration Mode.....	438	DC Characteristics	
Configuration Register Protection .....	482	CTMU Current Source Specifications.....	557
Core Features		PIC18F66K80 Family (Industrial) .....	555, 557
Easy Migration .....	12	Power-Down and Supply Current.....	540
Extended Instruction Set.....	11	Supply Voltage .....	539
Memory Options.....	11	DCFSNZ .....	503
nanoWatt Technology .....	11	DECFSZ .....	503
Oscillator Options and Features .....	11	Default System Clock .....	57
CPFSEQ .....	500	Details on Individual Family Members .....	12
CPFSGT .....	501	Development Support .....	533
CPFSLT .....	501	Device Overview .....	11
Crystal Oscillator/Ceramic Resonator .....	58	Features (28-Pin Devices).....	13
Customer Change Notification Service .....	617	Features (40/44-Pin Devices).....	13
Customer Notification Service.....	617	Features (64-Pin Devices).....	14
Customer Support.....	617	Device Reset Timers .....	83
<b>D</b>		Oscillator Start-up Timer (OST).....	83
Data Addressing Modes.....	123	PLL Lock Time-out .....	83
Comparing Addressing Modes with the Extended In- struction Set Enabled.....	127	Power-up Timer (PWRT) .....	83
Direct.....	123	Direct Addressing .....	124
Indexed Literal Offset.....	126	Disable/Sleep Mode.....	438
BSR .....	128		
Instructions Affected .....	126	<b>E</b>	
Mapping Access Bank .....	128	ECAN Module .....	391
Indirect .....	123	Baud Rate Setting .....	446
Inherent and Literal .....	123	Bit Time Partitioning .....	446
Data EEPROM		Bit Timing Configuration Registers .....	452
Associated Registers .....	144	Calculating TQ, Nominal Bit Rate and Nominal Bit Time .....	449
Code Protection .....	482	CAN Baud Rate Registers .....	430
During Code-Protect .....	143	CAN Control and Status Registers .....	393
EEADR and EEADRH Registers .....	139	CAN I/O Control Register .....	433
EECON1 and EECN2 Registers .....	139	CAN Interrupt Registers .....	434
Overview .....	139	CAN Interrupts .....	453
Reading.....	141	Bus Activity Wake-up .....	454
Spurious Write Protection .....	143	Bus-Off .....	455
Using.....	143	Code Bits .....	454
Write Verify .....	141	Error .....	454
Writing.....	141	Message Error .....	454
Data EEPROM Memory		Receive .....	454
Operation During Code-Protect .....	143	Receiver Bus Passive .....	455
Data Memory .....	108	Receiver Overflow .....	455
Access Bank .....	110	Receiver Warning .....	455
Bank Select Register (BSR).....	108	Transmit .....	454
Extended Instruction Set.....	126	Transmitter Bus Passive .....	455
General Purpose Registers.....	110	Transmitter Warning .....	455
Memory Maps		CAN Message Buffers .....	440
PIC18FX5K80/X6K80 Devices .....	109	Dedicated Receive .....	440
Special Function Registers .....	111	Dedicated Transmit .....	440
Special Function Registers .....	111	Programmable Auto-RTR .....	441
Data Signal Modulator (DSM) .....	195	Programmable Transmit/Receive .....	440
Associated Registers .....	204	CAN Message Transmission .....	441
Carrier Signal Sources.....	197	Aborting .....	441
Carrier Source		Initiating .....	441
Pin Disable.....	200	Priority .....	442
Polarity Select .....	200	CAN Modes of Operation .....	438
Carrier Synchronization .....	197	CAN Registers .....	393
Effects of a Reset.....	200	Configuration Mode .....	438
Modulated Output Polarity .....	200	Dedicated CAN Receive Buffer Registers .....	406
Modulator Signal Sources.....	197	Dedicated CAN Transmit Buffer Registers .....	400
Modulator Source Pin Disable .....	200	Disable/Sleep Mode .....	438
Operation .....	197	Error Detection .....	452
Operation in Sleep Mode .....	200	Acknowledge .....	452
Programmable Modulator Data .....	200	Bit .....	452
Slew Rate Control .....	200	CRC .....	452
DAW .....	502	Error Modes and Counters .....	452

# PIC18F66K80 FAMILY

---

Error States .....	452
Form .....	452
Stuff Bit .....	452
Error Modes State (diagram) .....	453
Error Recognition Mode .....	439
Filter-Mask Truth (table) .....	444
Functional Modes .....	439
Mode 0 (Legacy Mode) .....	439
Mode 1 (Enhanced Legacy Mode) .....	439
Mode 2 (Enhanced FIFO Mode) .....	440
Information Processing Time (IPT) .....	449
Lengthening a Bit Period .....	450
Listen Only Mode .....	439
Loopback Mode .....	439
Message Acceptance Filters and Masks .....	421, 444
Message Acceptance Mask and Filter Operation .....	445
Message Reception .....	443
Enhanced FIFO Mode .....	444
Priority .....	443
Time-Stamping .....	444
Normal Mode .....	438
Oscillator Tolerance .....	451
Overview .....	391
Phase Buffer Segments .....	449
Programmable TX/RX and Auto-RTR Buffers .....	413
Programming Time Segments .....	451
Propagation Segment .....	449
Sample Point .....	449
Shortening a Bit Period .....	451
Synchronization .....	450
Hard .....	450
Resynchronization .....	450
Rules .....	450
Synchronization Segment .....	449
Time Quanta .....	449
Values for ICODE (table) .....	454
Effect on Standard PIC18 Instructions .....	530
Effects of Power-Managed Modes on Various Clock Sources	63
Electrical Characteristics .....	537
Enhanced Capture/Compare/PWM (ECCP) .....	265
Capture Mode. See Capture.	
Compare Mode. See Compare.	
ECCP Mode and Timer Resources .....	268
Enhanced PWM Mode .....	271
Auto-Restart .....	280
Auto-Shutdown .....	278
Direction Change in Full-Bridge Output Mode .....	277
Full-Bridge Application .....	275
Full-Bridge Mode .....	275
Half-Bridge Application .....	274
Half-Bridge Application Examples .....	281
Half-Bridge Mode .....	274
Output Relationships (Active-High and Active-Low)	
272	
Output Relationships Diagram .....	273
Programmable Dead-Band Delay .....	281
Shoot-Through Current .....	281
Start-up Considerations .....	278
Outputs and Configuration .....	268
Enhanced Capture/Compare/PWM (ECCP) and Timer1/2/3/4	
Associated Registers .....	286
Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART). See EUSART.	
Equations .....	
16 x 16 Signed Multiplication Algorithm .....	146
16 x 16 Unsigned Multiplication Algorithm .....	146
16MHz Clock from 4x PLL Jitter .....	447
A/D Acquisition Time .....	367
A/D Minimum Charging Time .....	367
Calculating the Minimum Required Acquisition Time	367
Jitter and Total Frequency Error .....	447
Resultant Frequency Error .....	447
Errata .....	9
Error Recognition Mode .....	438
EUSART	
Asynchronous Mode .....	343
12-Bit Break Transmit and Receive .....	350
Associated Registers, Receive .....	347
Associated Registers, Transmit .....	345
Auto-Wake-up on Sync Break .....	348
Receiver .....	346
Setting up 9-Bit Mode with Address Detect .....	346
Transmitter .....	343
Baud Rate Generator	
Operation in Power-Managed Mode .....	337
Baud Rate Generator (BRG) .....	337
Associated Registers .....	338
Auto-Baud Rate Detect .....	341
Baud Rate Error, Calculating .....	338
Baud Rates, Asynchronous Modes .....	339
High Baud Rate Select (BRGH Bit) .....	337
Sampling .....	337
Synchronous Master Mode .....	351
Associated Registers, Receive .....	354
Associated Registers, Transmit .....	352
Reception .....	353
Transmission .....	351
Synchronous Slave Mode .....	355
Associated Registers, Receive .....	356
Associated Registers, Transmit .....	355
Reception .....	356
Transmission .....	355
Extended Instruction Set	
ADDFSR .....	526
ADDULNK .....	526
CALLW .....	527
MOVSF .....	527
MOVSS .....	528
PUSHL .....	528
SUBFSR .....	529
SUBULNK .....	529
External Oscillator Modes	
Clock Input (EC Modes) .....	59
HS .....	58
F	
Fail-Safe Clock Monitor .....	457, 477
Exiting Operation .....	477
Interrupts in Power-Managed Modes .....	478
POR or Wake from Sleep .....	478
WDT During Oscillator Failure .....	477
Fail-Safe Clock Monitor (FSCM) .....	457
Fast Register Stack .....	105
Firmware Instructions .....	483
Flash Program Memory .....	129
Associated Registers .....	137
Control Registers .....	130
EECON1 and EECON2 .....	130
TABLAT (Table Latch) Register .....	132

# PIC18F66K80 FAMILY

TBLPTR (Table Pointer) Register .....	132	Operation .....	317
Erase Sequence .....	134	Reception .....	322
Erasing .....	134	Repeated Start Condition Timing .....	321
Operation During Code-Protect .....	137	Start Condition Timing .....	320
Reading .....	133	Transmission .....	322
Table Pointer .....	132	Multi-Master Communication, Bus Collision and Arbitration .....	326
Boundaries Based on Operation .....	132	Multi-Master Mode .....	326
Table Pointer Boundaries .....	132	Operation .....	301
Table Reads and Table Writes .....	129	Read/Write Bit Information (R/W Bit) .....	301, 304
Write Sequence .....	135	Registers .....	296
Writing .....	135	Serial Clock (RC3/REF0//SCL/SCK) .....	304
Protection Against Spurious Writes .....	137	Slave Mode .....	301
Unexpected Termination .....	137	Address Masking Modes .....	302
Write Verify .....	137	5-Bit .....	302
FSCM. See Fail-Safe Clock Monitor.		7-Bit .....	303
<b>G</b>		Addressing .....	301
GOTO .....	504	Reception .....	304
<b>H</b>		Transmission .....	304
Hardware Multiplier .....	145	Sleep Operation .....	326
8 x 8 Multiplication Algorithms .....	145	Stop Condition Timing .....	325
Operation .....	145	ID Locations .....	457, 482
Performance Comparison (table) .....	145	Idle Modes .....	70
High/Low-Voltage Detect .....	385	INCF .....	504
Applications .....	389	INCFSZ .....	505
Associated Registers .....	390	In-Circuit Debugger .....	482
Current Consumption .....	387	In-Circuit Serial Programming (ICSP) .....	457, 482
Effects of a Reset .....	390	Indexed Literal Offset Addressing .....	530
Operation .....	386	and Standard PIC18 Instructions .....	530
During Sleep .....	390	Indexed Literal Offset Mode .....	530
Setup .....	387	Indirect Addressing .....	124
Start-up Time .....	387	INFSNZ .....	505
Typical Application .....	389	Initialization Conditions for all Registers .....	88-??
HLVD. See High/Low-Voltage Detect .....	385	Instruction Cycle .....	106
<b>I</b>		Clocking Scheme .....	106
I/O Descriptions .....	18	Flow/Pipelining .....	106
PIC18F2XK80 .....	18	Instruction Set .....	483
PIC18F4XK80 .....	24	ADDLW .....	489
PIC18F6XK80 .....	33	ADDWF .....	489
I/O Ports .....	171	ADDWF (Indexed Literal Offset Mode) .....	531
Analog/Digital Ports .....	174	ADDWFC .....	490
Open-Drain Outputs .....	173	ANDLW .....	490
Output Pin Drive .....	171	ANDWF .....	491
Pin Capabilities .....	171	BC .....	491
Port Slew Rate .....	174	BCF .....	492
Pull-up Configuration .....	171	BN .....	492
I <sup>2</sup> C Mode (MSSP) .....	325	BNC .....	493
Acknowledge Sequence Timing .....	325	BNN .....	493
Associated Registers .....	331	BNOV .....	494
Baud Rate Generator .....	318	BNZ .....	494
Bus Collision .....	311	BOV .....	497
During a Repeated Start Condition .....	329	BRA .....	495
During a Stop Condition .....	330	BSF .....	495
Clock Arbitration .....	319	BSF (Indexed Literal Offset Mode) .....	531
Clock Stretching .....	311	BTFSZ .....	496
10-Bit Slave Receive Mode (SEN = 1) .....	311	BTFSZ .....	496
10-Bit Slave Transmit Mode .....	311	BTG .....	497
7-Bit Slave Receive Mode (SEN = 1) .....	311	BZ .....	498
7-Bit Slave Transmit Mode .....	311	CALL .....	498
Clock Synchronization and the CKP bit .....	312	CLRF .....	499
Effects of a Reset .....	326	CLRWD .....	499
General Call Address Support .....	315	COMF .....	500
I <sup>2</sup> C Clock Rate w/BRG .....	318	CPFSEQ .....	500
Master Mode .....	316	CPFGT .....	501
Slave Mode .....	316	CPFSLT .....	501

# PIC18F66K80 FAMILY

---

DAW.....	502
DCFSNZ .....	503
DEC <sub>F</sub> .....	502
DECFSZ.....	503
Extended Instructions .....	525
Considerations when Enabling .....	530
Syntax.....	525
Use with MPLAB IDE Tools .....	532
General Format.....	485
GOTO .....	504
INCF.....	504
INCFSZ.....	505
INFSNZ.....	505
IORLW .....	506
IORWF .....	506
LFSR.....	507
MOV <sub>F</sub> .....	507
MOVFF .....	508
MOVLB .....	508
MOVLW .....	509
MOVWF .....	509
MULLW .....	510
MULWF .....	510
NEGF .....	511
NOP .....	511
Opcode Field Descriptions.....	484
POP .....	512
PUSH .....	512
RCALL .....	513
RESET .....	513
RETFIE .....	514
RETLW .....	514
RETURN .....	515
RLCF .....	515
RLNCF .....	516
RRCF .....	516
RRNCF .....	517
SET <sub>F</sub> .....	517
SET <sub>F</sub> (Indexed Literal Offset Mode) .....	531
SLEEP .....	518
Standard Instructions .....	483
SUBFWB.....	518
SUBLW .....	519
SUBWF .....	519
SUBWFB.....	520
SWAPF .....	520
TBLRD .....	521
TBLWT .....	522
TSTFSZ .....	523
XORLW .....	523
XORWF .....	524
INTCON Register .....	
RBIF Bit.....	177
Inter-Integrated Circuit. See I <sup>2</sup> C.	
Internal Oscillator Block .....	60
INTIO Modes.....	60
INTOSC Frequency Drift.....	61
INTOSC Output Frequency.....	61
INTPLL Modes .....	60
Internal RC Oscillator .....	
Use with WDT .....	472
Internal Voltage Regulator Specifications .....	559
Internet Address .....	617
Interrupt Sources.....	457
A/D Conversion Complete .....	366
Capture Complete (CCP).....	258
Capture Complete (ECCP).....	269
Compare Complete (CCP).....	259
Compare Complete (ECCP).....	270
ECAN Module .....	453
Interrupt-on-Change (RB7:RB4).....	177
TMR0 Overflow.....	207
TMR1 Overflow.....	215
TMR2 to PR2 Match (PWM).....	262
TMR3 Overflow.....	223
TMRx Overflow.....	232
Interrupts.....	147
Associated Registers.....	169
During, Context Saving.....	169
INTX Pin .....	168
PORTB, Interrupt-on-Change .....	168
TMR0 .....	168
Interrupts, Flag Bits .....	
Interrupt-on-Change (RB7:RB4) Flag (RBIF Bit) .....	177
INTOSC. See Internal Oscillator Block.	
IORLW .....	506
IORWF .....	506
<b>L</b>	
LFSR.....	507
Listen Only Mode .....	438
Loopback Mode .....	438
<b>M</b>	
Master Clear Reset (MCLR) .....	81
Master Synchronous Serial Port (MSSP). See MSSP.	
Memory Organization .....	101
Data Memory .....	108
Program Memory .....	101
Memory Programming Requirements.....	558
Microchip Internet Web Site.....	617
Migration to PIC18F66K80 .....	602
MOV <sub>F</sub> .....	507
MOVFF .....	508
MOVLB .....	508
MOVLW .....	509
MOVSF .....	527
MOVSS .....	528
MOVWF .....	509
MPLAB ASM30 Assembler, Linker, Librarian .....	534
MPLAB Integrated Development Environment Software ..	533
MPLAB PM3 Device Programmer .....	536
MPLAB REAL ICE In-Circuit Emulator System .....	535
MPLINK Object Linker/MPLIB Object Librarian .....	534
<b>MSSP</b> .....	
ACK Pulse .....	301, 304
I <sup>2</sup> C Mode. See I <sup>2</sup> C Mode.	
Module Overview .....	287
SPI Master/Slave Connection .....	291
MULLW .....	510
MULWF .....	510
<b>N</b>	
NEGF .....	511
NOP .....	511
Normal Operation Mode .....	438
Notable Differences Between PIC18F66K80 and PIC18F4580 and PIC18F4680 Families - 28, 40/44-pin Devices ..	602
Notable Differences Between PIC18F66K80 and PIC18F8680 Families - 64-pin Devices ..	603

# PIC18F66K80 FAMILY

---

## O

On-Chip Voltage Regulator .....	474
Oscillator Configuration .....	51
EC .....	51
ECIO .....	51
HS .....	51
Internal Oscillator Block .....	60
INTIO1 .....	51
INTIO2 .....	51
LP .....	51
RC .....	51
RCIO .....	51
XT .....	51
Oscillator Selection .....	457
Oscillator Start-up Timer (OST) .....	63, 84
Oscillator Switching .....	56
Oscillator Transitions .....	57
Oscillator, Timer1 .....	209
Oscillator, Timer3 .....	223

## P

P1A/P1B/P1C/P1D. See Enhanced Capture/Compare/PWM (ECCP) .....	271
Packaging .....	581
Details .....	583
Marking .....	581
Parallel Slave Port (PSP) .....	192
Associated Registers .....	194
PORTD .....	192
Pin Functions	
(Overline)MCLR/RE3 .....	24
AVDD .....	43
AVss .....	43
MCLR/RE3 .....	18
MCLR/RE3 .....	33
OSC1/CLKIN/RA7 .....	18, 24, 33
OSC2/CLKOUT/RA6 .....	18, 24, 33
RA0/CVREF/AN0/ULPWU .....	19, 25
RA0/CVREF/AN0/ULPWU .....	34
RA1/AN1 .....	19
RA1/AN1/C1INC .....	25, 34
RA2/REF-/AN2 .....	19
RA2/VREF-/AN2/C2INC .....	34
RA2/VREF-/AN2/C2INC .....	25
RA3/VREF+/AN3 .....	19, 25
RA3/VREF+/AN3 .....	34
RA5/AN4/C2INB/HLVDIN/T1CKI/SS/CTMUI .....	19
RA5/AN4/HLVDIN/T1CKI/SS .....	25, 34
RB0/AN10/C1INA/FLT0/INT0 .....	20
RB0/AN10/FLT0/INT0 .....	26, 35
RB1/AN8/C1INB/P1B/CTDIN/INT1 .....	20
RB1/AN8/CTDIN/INT1 .....	26, 35
RB2/CANTX/C1OUT/P1C/CTED1/INT2 .....	20
RB2/CANTX/CTED1/INT2 .....	26, 35
RB3/CANRX/C2OUT/P1D/CTED2/INT3 .....	20
RB3/CANRX/CTED2/INT3 .....	26, 35
RB4/AN9/C2INA/ECCP1/P1A/CTPLS/KB10 .....	21
RB4/AN9/CTPLS/KB10 .....	26, 35
RB5/T0CKI/T3CKI/CCP5/KB11 .....	21, 26, 35
RB6/PGC/KB12 .....	27, 36
RB6/PGC/TX2/CK2/KB12 .....	21
RB7/PGD/T3G/KB13 .....	27, 36
RB7/PGD/T3G/RX2/DT2/KB13 .....	21
RC0/SOSCO/SCLKI .....	22, 28, 37
RC1/SOSC .....	28

RC1/SOSCI .....	22, 37
RC2/T1G/CCP2 .....	22, 28, 37
RC3/REFO/SCL/SCK .....	22, 28, 37
RC4/SDA/SDI .....	22, 28, 37
RC5/SDO .....	22, 28, 37
RC6/CANTX/TX1/CK1/CCP3 .....	22, 28
RC6/CCP3 .....	37
RC7/CANRX/RX1/DT1/CCP4 .....	23, 29
RC7/CCP4 .....	37
RD0/C1INA/PSP0 .....	30, 38
RD1/C1INB/PSP1 .....	30, 38
RD2/C2INA/PSP2 .....	30, 38
RD3/C2INB/CTMUI/PSP3 .....	30, 38
RD4/ECCP1/P1A/PSP4 .....	30, 38
RD5/P1B/PSP5 .....	30, 38
RD6/P1C/PSP6 .....	39
RD6/TX2/CK2/P1C/PSP6 .....	31
RD7/P1D/PSP7 .....	39
RD7/RX2/DT2/P1D/PSP7 .....	31
RE0/AN5/RD .....	31, 40
RE1/AN6/C1OUT/wr .....	31, 40
RE2/AN7/C2OUT/CS .....	31, 40
RE4/CANRX .....	40
RE5/CANTX .....	40
RE6/RX2/DT2 .....	40
RE7/TX2/CK2 .....	40
RF0/MDMIN .....	41
RF1 .....	41
RF2/MDCIN1 .....	41
RF3 .....	41
RF4/MDCIN2 .....	41
RF5 .....	41
RF6/MDOUT .....	41
RF7 .....	41
RG0/RX1/DT1 .....	42
RG1/CANTX2 .....	42
RG2/T3CKI .....	42
RG3/TX1/CK1 .....	42
RG4/T0CKI .....	42
VDD .....	32, 43
VDDCORE/VCAP .....	23, 32, 43
Vss .....	23, 32, 43
PLL	
Frequency Multiplier .....	59
HSPLL and ECPLL Oscillator Modes .....	59
Use with HF-INTOSC .....	59
PLL Lock Time-out .....	84
POP .....	512
POR. See Power-on Reset.	
PORTA	
Associated Registers .....	176
LATA Register .....	175
PORTA Register .....	175
TRISA Register .....	175
PORTB	
Associated Registers .....	180
LATB Register .....	177
PORTB Register .....	177
RB7:RB4 Interrupt-on-Change Flag (RBIF Bit) .....	177
TRISB Register .....	177
PORTC	
Associated Registers .....	183
LATC Register .....	181
PORTC Register .....	181
RC3/REFO/SCL/SCK Pin .....	304

# PIC18F66K80 FAMILY

---

TRISC Register.....	181
PORTD	
Associated Registers .....	186
LATD Register .....	184
PORTD Register .....	184
TRISD Register.....	184
PORTE	
Associated Registers .....	188
LATE Register.....	187
PORTE Register .....	187
RE0/AN5/RD Pin.....	192
RE1/AN6/C1OUT/WR Pin.....	192
RE2/AN7/C2OUT/CS Pin.....	192
TRISE Register.....	187
PORTF	
Associated Registers .....	189
LATF Register.....	189
PORTF Register .....	189
TRISF Register.....	189
PORTG	
Associated Registers .....	191
LATG Register .....	190
PORTG Register .....	190
TRISG Register.....	190
Power-Managed Modes .....	65
and EUSART Operation.....	337
and PWM Operation .....	285
and SPI Operation .....	295
Clock Transitions and Status Indicators.....	66
Entering.....	65
Exiting Idle and Sleep Modes .....	76
by Interrupt.....	76
by Reset.....	76
by WDT Time-out.....	76
Without an Oscillator Start-up Delay.....	76
Idle Modes .....	70
PRI_IDLE.....	71
RC_IDLE.....	72
SEC_IDLE.....	71
Multiple Sleep Commands .....	66
Run Modes.....	66
PRI_RUN .....	66
RC_RUN .....	67
SEC_RUN.....	66
Selecting .....	65
Sleep Mode .....	70
OSC1 and OSC2 Pin States .....	63
Summary (table) .....	65
Power-on Reset (POR) .....	81
Oscillator Start-up Timer (OST) .....	84
Power-up Timer (PWRT) .....	83
Time-out Sequence.....	84
Power-up Delays.....	63
Power-up Timer (PWRT).....	63, 83
Prescaler, Capture .....	258
Prescaler, Timer0.....	207
Prescaler, Timer2.....	263
PRI_IDLE Mode .....	71
PRI_RUN Mode .....	66
Program Counter (PC) .....	103
PCL, PCH and PCU Registers.....	103
PCLATH and PCLATU Registers .....	103
Program Memory	
Code Protection .....	480
Extended Instruction Set .....	125
Hard Memory Vectors.....	102
Instructions .....	107
Two-Word .....	107
Interrupt Vector .....	102
Look-up Tables.....	105
Memory Maps .....	101
Hard Vectors.....	102
Reset Vector .....	102
Program Verification and Code Protection .....	479
Associated Registers .....	480
Programming, Device Instructions .....	483
PSP. See Parallel Slave Port.	
Pulse-Width Modulation. See PWM (CCP Module).	
PUSH .....	512
PUSH and POP Instructions .....	104
PUSHL .....	528
PWM (CCP Module)	
Associated Registers .....	264
Duty Cycle .....	263
Example Frequencies/Resolutions .....	263
Period .....	262
Setup for PWM Operation.....	263
TMR2 to PR2 Match .....	262
PWM (ECCP Module)	
Effects of a Reset .....	285
Operation in Power-Managed Modes .....	285
Operation with Fail-Safe Clock Monitor .....	285
Pulse Steering Mode .....	282
Steering Synchronization .....	284
PWM Mode. See Enhanced Capture/Compare/PWM .....	271
Q	
Q Clock .....	263
R	
RAM. See Data Memory.	
RC_IDLE Mode .....	72
RC_RUN Mode .....	67
RCALL .....	513
RCON Register	
Bit Status During Initialization .....	87
Reader Response .....	618
Receiver Warning .....	455
Reference Clock Output .....	61
Register File .....	110
Register File Summary .....	113–121
Registers	
ADCON0 (A/D Control 0).....	358
ADCON1 (A/D Control 1).....	359
ADCON2 (A/D Control 2).....	360
ADRESH (A/D Result High Byte, Left Justified, ADFM = 0) .....	362
ADRESH (A/D Result High Byte, Right Justified, ADFM = 1) .....	362
ADRESL (A/D Result Low Byte, Left Justified, ADFM = 0) .....	362
ADRESL (A/D Result Low Byte, Right Justified, ADFM = 1) .....	363
ANCON0 (A/D Port Configuration 0) .....	363
ANCON1 (A/D Port Configuration 1) .....	364
BAUDCONx (Baud Rate Control) .....	336
BIE0 (Buffer Interrupt Enable 0) .....	437
BnCON (TX/RX Buffer n Control, Receive Mode) .....	413
BnCON (TX/RX Buffer n Control, Transmit Mode) .....	414
BnDLC (TX/RX Buffer n Data Length Code in Receive Mode) .....	419

BnDLC (TX/RX Buffer n Data Length Code in Transmit Mode).....	420
BnDm (TX/RX Buffer n Data Field Byte m in Receive Mode).....	418
BnDm (TX/RX Buffer n Data Field Byte m in Transmit Mode).....	418
BnEIDH (TX/RX Buffer n Extended Identifier, High Byte in Receive Mode).....	417
BnEIDH (TX/RX Buffer n Extended Identifier, High Byte in Transmit Mode).....	417
BnEIDL (TX/RX Buffer n Extended Identifier, Low Byte in Receive Mode).....	417, 418
BnSIDH (TX/RX Buffer n Standard Identifier, High Byte in Receive Mode).....	415
BnSIDH (TX/RX Buffer n Standard Identifier, High Byte in Transmit Mode).....	415
BnSIDL (TX/RX Buffer n Standard Identifier, Low Byte in Receive Mode).....	416
BRGCON1 (Baud Rate Control 1).....	430
BRGCON2 (Baud Rate Control 2).....	431
BRGCON3 (Baud Rate Control 3).....	432
BSEL0 (Buffer Select 0).....	420
CANCON (CAN Control).....	394
CANSTAT (CAN Status).....	395
CCP1CON (Enhanced Capture/Compare/PWM1 Control).....	266
CCPPRxL (CCPx Period Low Byte).....	255
CCPRxH (CCPx Period High Byte).....	255
CCPTMRS (CCP Timer Select).....	254, 267
CCPxCON (CCPx Control, CCP2-CCP5).....	253
CIOCON (CAN I/O Control) .....	433
CMSTAT (Comparator Status).....	375
CMxCON (Comparator Control x).....	374
COMSTAT (CAN Communication Status).....	399
CONFIG1H (Configuration 1 High).....	460
CONFIG1L (Configuration 1 Low).....	459
CONFIG2H (Configuration 2 High).....	462
CONFIG2L (Configuration 2 Low).....	461
CONFIG3H (Configuration 3 High).....	463
CONFIG4L (Configuration 4 Low).....	464
CONFIG5H (Configuration 5 High).....	466
CONFIG5L (Configuration 5 Low).....	465
CONFIG6H (Configuration 6 High).....	468
CONFIG6L (Configuration 6 Low).....	467
CONFIG7H (Configuration 7 High).....	470
CONFIG7L (Configuration 7 Low).....	469
CTMUCONH (CTMU Control High).....	236
CTMUCONL (CTMU Control Low).....	237
CTMUICON (CTMU Current Control) .....	238
CVRCON (Comparator Voltage Reference Control) .....	381
DEVID1 (Device ID 1).....	471
DEVID2 (Device ID 2).....	471
ECANCON (Enhanced CAN Control).....	398
ECCP1AS (ECCP1 Auto-Shutdown Control).....	279
ECCP1DEL (Enhanced PWM Control).....	282
EECON1 (Data EEPROM Control 1).....	140
EECON1 (EEPROM Control 1).....	131
HLVDCON (High/Low-Voltage Detect Control).....	385
INTCON (Interrupt Control).....	149
INTCON2 (Interrupt Control 2).....	150
INTCON3 (Interrupt Control 3).....	151
IOCB (Interrupt-on-Change PORTB Control) .....	168
IPR1 (Peripheral Interrupt Priority 1).....	162
IPR2 (Peripheral Interrupt Priority 2).....	163
IPR3 (Peripheral Interrupt Priority 3).....	164
IPR4 (Peripheral Interrupt Priority 4) .....	165
IPR5 (Peripheral Interrupt Priority 5) .....	166, 436
MDCARH (Modulation High Carrier Control).....	203
MDCARL (Modulation Low Carrier Control).....	204
MDCON (Modulation Control Register) .....	201
MDSRC (Modulation Source Control) .....	202
MSEL0 (Mask Select 0).....	426
MSEL1 (Mask Select 1).....	427
MSEL2 (Mask Select 2).....	428
MSEL3 (Mask Select 3).....	429
ODCON (Peripheral Open-Drain Control) .....	173
OSCCON (Oscillator Control).....	53
OSCCON2 (Oscillator Control 2).....	54, 225
OSCTUNE (Oscillator Tuning).....	55
PADCFG1 (Pad Configuration) .....	172
PIE1 (Peripheral Interrupt Enable 1) .....	157
PIE2 (Peripheral Interrupt Enable 2) .....	158
PIE3 (Peripheral Interrupt Enable 3) .....	159
PIE4 (Peripheral Interrupt Enable 4) .....	160
PIE5 (Peripheral Interrupt Enable 5) .....	161, 435
PIR1 (Peripheral Interrupt Request (Flag) 1).....	152
PIR2 (Peripheral Interrupt Request (Flag) 2).....	153
PIR3 (Peripheral Interrupt Request (Flag) 3).....	154
PIR4 (Peripheral Interrupt Request (Flag) 4).....	155
PIR5 (Peripheral Interrupt Request (Flag) 5).....	156, 434
PMD0 (Peripheral Module Disable 0) .....	75
PMD1 (Peripheral Module Disable 1) .....	74
PMD2 (Peripheral Module Disable 2) .....	73
PSPCON (Parallel Slave Port Control) .....	193
PSTR1CON (Pulse Steering Control) .....	283
RCON (Reset Control).....	80, 167
RCSTAx (Receive Status and Control) .....	335
REFOCON (Reference Oscillator Control) .....	62
RXB0CON (Receive Buffer 0 Control).....	406
RXB1CON (Receive Buffer 1 Control).....	408
RXBnDLC (Receive Buffer n Data Length Code).....	411
RXBnDm (Receive Buffer n Data Field Byte m).....	411
RXBnEIDH (Receive Buffer n Extended Identifier, High Byte) .....	410
RXBnEIDL (Receive Buffer n Extended Identifier, Low Byte) .....	410
RXBnSIDH (Receive Buffer n Standard Identifier, High Byte) .....	409
RXBnSIDL (Receive Buffer n Standard Identifier, Low Byte) .....	410
RXERRCNT (Receive Error Count).....	412
RXFBCONn (Receive Filter Buffer Control n) .....	425
RXFCONn (Receive Filter Control n) .....	424
RXFnEIDH (Receive Acceptance Filter n Extended Identifier, High Byte) .....	422
RXFnEIDL (Receive Acceptance Filter n Extended Identifier, Low Byte) .....	422
RXFnSIDH (Receive Acceptance Filter n Standard Identifier Filter, High Byte) .....	421
RXFnSIDL (Receive Acceptance Filter n Standard Identifier Filter, Low Byte) .....	421
RXMnEIDH (Receive Acceptance Mask n Extended Identifier Mask, High Byte) .....	423
RXMnEIDL (Receive Acceptance Mask n Extended Identifier Mask, Low Byte) .....	423
RXMnSIDH (Receive Acceptance Mask n Standard Identifier Mask, High Byte) .....	422
RXMnSIDL (Receive Acceptance Mask n Standard Identifier Mask, Low Byte) .....	423
SDFLC (Standard Data Bytes Filter Length Count) .....	424

# PIC18F66K80 FAMILY

---

SLRCON (Slew Rate Control).....	174	Software Simulator (MPLAB SIM) .....	535
SSPCON1 (MSSP Control 1, I <sup>2</sup> C Mode) .....	298	Special Event Trigger. See Compare (CCP Module).	
SSPCON1 (MSSP Control 1, SPI Mode).....	289	Special Event Trigger. See Compare (ECCP Mode).	
SSPCON2 (MSSP Control 2, I <sup>2</sup> C Master Mode) .....	299	SPI Mode (MSSP) .....	287
SSPCON2 (MSSP Control 2, I <sup>2</sup> C Slave Mode) .....	300	Associated Registers.....	295
SSPMASK (I <sup>2</sup> C Slave Address Mask).....	300	Bus Mode Compatibility.....	295
SSPSTAT (MSSP Status, I <sup>2</sup> C Mode).....	297	Effects of a Reset .....	295
SSPSTAT (MSSP Status, SPI Mode).....	288	Enabling SPI I/O .....	291
STATUS .....	122	Master Mode.....	292
STKPTR (Stack Pointer).....	104	Master/Slave Connection.....	291
T0CON (Timer0 Control).....	205	Operation .....	290
T1CON (Timer1 Control).....	209	Operation in Power-Managed Modes .....	295
T1GCON (Timer1 Gate Control).....	211	Serial Clock.....	287
T2CON (Timer2 Control).....	221	Serial Data In.....	287
T3CON (Timer3 Control).....	223	Serial Data Out.....	287
T3GCON (Timer3 Gate Control).....	224	Slave Mode.....	293
T4CON (Timer4 Control).....	233	Slave Select.....	287
TXBIE (Transmit Buffers Interrupt Enable) .....	437	Slave Select Synchronization .....	293
TXBnCON (Transmit Buffer n Control) .....	400	SPI Clock .....	292
TXBnDLC (Transmit Buffer n Data Length Code)....	403	SSPBUF Register .....	292
TXBnDm (Transmit Buffer n Data Field Byte m).....	402	SSPSR Register .....	292
TXBnEIDH (Transmit Buffer n Extended Identifier, High Byte).....	401	Typical Connection .....	291
TXBnEIDL (Transmit Buffer n Extended Identifier, Low Byte).....	402	SS .....	287
TXBnSIDH (Transmit Buffer n Standard Identifier, High Byte).....	401	SSPOV .....	322
TXBnSIDL (Transmit Buffer n Standard Identifier, Low Byte).....	401	SSPOV Status Flag .....	322
TXERRCNT (Transmit Error Count).....	403	SSPSTAT Register	
TXSTAx (Transmit Status and Control) .....	334	R/W Bit .....	301, 304
WDTCON (Watchdog Timer Control).....	473	Stack Full/Underflow Resets.....	105
WPUB (Weak Pull-up PORTB Enable).....	172	SUBFSR .....	529
RESET .....	513	SUBFWB .....	518
Resets .....	79, 457	SUBLW .....	519
Brown-out Reset (BOR) .....	457	SUBULNK .....	529
Oscillator Start-up Timer (OST) .....	457	SUBWF .....	519
Power-on Reset (POR) .....	457	SUBWFB .....	520
Power-up Timer (PWRT) .....	457	SWAPF .....	520
RETFIE .....	514	<b>T</b>	
RETLW .....	514	Table Pointer Operations (table).....	132
RETURN .....	515	Table Reads/Table Writes .....	105
Return Address Stack .....	103	TBLRD .....	521
Return Stack Pointer (STKPTR) .....	104	TBLWT .....	522
Revision History .....	601	Time-out in Various Situations (table).....	84
RLCF .....	515	Timer0.....	205
RLNCF .....	516	Associated Registers .....	207
RRCF .....	516	Operation .....	206
RRNCF.....	517	Overflow Interrupt .....	207
<b>S</b>		Prescaler .....	207
SCK.....	287	Switching Assignment .....	207
SDI .....	287	Prescaler Assignment (PSA Bit) .....	207
SDO .....	287	Prescaler Select (T0PS2:T0PS0 Bits) .....	207
SEC_IDLE Mode.....	71	Reads and Writes in 16-Bit Mode .....	206
SEC_RUN Mode.....	66	Source Edge Select (T0SE Bit) .....	206
Selective Peripheral Module Control.....	72	Source Select (T0CS Bit) .....	206
Serial Clock, SCK.....	287	Timer1.....	209
Serial Data In (SDI).....	287	16-Bit Read/Write Mode .....	214
Serial Data Out (SDO) .....	287	Associated Registers .....	220
Serial Peripheral Interface. See SPI Mode.		Clock Source Selection.....	212
SETF .....	517	Gate .....	216
Shoot-Through Current .....	281	Interrupt .....	215
Slave Select (SS).....	287	Operation .....	212
SLEEP.....	518	Oscillator .....	209
Sleep Mode .....	70	Oscillator, as Secondary Clock.....	56
		Resetting, Using the ECCP Special Event Trigger... <td>216</td>	216
		SOSC Oscillator.....	214
		Layout Considerations .....	215
		Use as a Clock Source .....	215

# PIC18F66K80 FAMILY

TMR1H Register .....	209
TMR1L Register .....	209
Timer2 .....	221
Associated Registers .....	222
Interrupt.....	222
Operation .....	221
Output .....	222
PR2 Register.....	262
TMR2 to PR2 Match Interrupt .....	262
Timer3 .....	223
16-Bit Read/Write Mode.....	227
Associated Registers .....	232
Gates .....	228
Operation .....	226
Oscillator .....	223
Overflow Interrupt .....	223, 232
SOSC Oscillator	
Use as the Timer3 Clock Source .....	227
Special Event Trigger (ECCP) .....	232
TMR3H Register .....	223
TMR3L Register.....	223
Timer4 .....	233
Associated Registers .....	234
Interrupt.....	234
Operation .....	233
Output .....	234
Postscaler. See Postscaler, Timer4.	
PR4 Register.....	233
Prescaler. See Prescaler, Timer4.	
TMR4 Register.....	233
Timing Diagrams	
A/D Conversion.....	580
Asynchronous Reception .....	347
Asynchronous Transmission.....	344
Asynchronous Transmission (Back-to-Back) .....	344
Automatic Baud Rate Calculation .....	342
Auto-Wake-up Bit (WUE) During Normal Operation .	349
Auto-Wake-up Bit (WUE) During Sleep .....	349
Baud Rate Generator with Clock Arbitration .....	319
BRG Overflow Sequence .....	342
BRG Reset Due to SDA Arbitration During Start Condition .....	328
Brown-out Reset (BOR) .....	566
Bus Collision During a Repeated Start Condition (Case 1).....	329
Bus Collision During a Repeated Start Condition (Case 2).....	329
Bus Collision During a Start Condition (SCL = 0) .....	328
Bus Collision During a Stop Condition (Case 1) .....	330
Bus Collision During a Stop Condition (Case 2) .....	330
Bus Collision During Start Condition (SDA Only).....	327
Bus Collision for Transmit and Acknowledge.....	326
Capture/Compare/PWM (ECCP1, ECCP2) .....	569
CLKO and I/O .....	564
Clock/Instruction Cycle .....	106
DSM Carrier High Synchronization (MDCHSYNC = 1, MDCLSYNC = 0) .....	198
DSM Carrier Low Synchronization (MDCHSYNC = 0, MDCLSYNC = 1) .....	199
DSM Full Synchronization (MDCHSYNC = 1, MDCLSYNC = 1).....	199
DSM No Synchronization (MDCHSYNC = 0, MDCLSYNC = 0).....	198
DSM On-Off Keying (OOK) Synchronization .....	198
Enhanced PWM Output (Active-High) .....	272
Enhanced PWM Output (Active-Low).....	273
EUSART Synchronous Transmission (Master/Slave) .....	578
EUSART/AUSART Synchronous Receive (Master/Slave) .....	578
Example SPI Master Mode (CKE = 0) .....	570
Example SPI Master Mode (CKE = 1) .....	571
Example SPI Slave Mode (CKE = 0) .....	572
Example SPI Slave Mode (CKE = 1) .....	573
External Clock .....	562
Fail-Safe Clock Monitor (FSCM) .....	478
First Start Bit Timing .....	320
Full-Bridge PWM Output.....	276
Half-Bridge PWM Output .....	274, 281
High-Voltage Detect Operation (VDIRMAG = 1) .....	389
HLVD Characteristics .....	567
I <sup>2</sup> C Acknowledge Sequence .....	325
I <sup>2</sup> C Bus Data.....	575
I <sup>2</sup> C Bus Start/Stop Bits .....	574
I <sup>2</sup> C Master Mode (7 or 10-Bit Transmission) .....	323
I <sup>2</sup> C Master Mode (7-Bit Reception) .....	324
I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 0, ADMSK = 01001) .....	308
I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 0) .....	309
I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 1) .....	314
I <sup>2</sup> C Slave Mode (10-Bit Transmission) .....	310
I <sup>2</sup> C Slave Mode (7-bit Reception, SEN = 0, ADMSK = 01011) .....	306
I <sup>2</sup> C Slave Mode (7-Bit Reception, SEN = 0) .....	305
I <sup>2</sup> C Slave Mode (7-Bit Reception, SEN = 1) .....	313
I <sup>2</sup> C Slave Mode (7-Bit Transmission) .....	307
I <sup>2</sup> C Slave Mode General Call Address Sequence (7 or 10-Bit Addressing Mode) .....	315
I <sup>2</sup> C Stop Condition Receive or Transmit Mode .....	325
Low-Voltage Detect Operation (VDIRMAG = 0) .....	388
MSSP Clock Synchronization .....	312
MSSP I <sup>2</sup> C Bus Data .....	576
MSSP I <sup>2</sup> C Bus Start/Stop Bits .....	576
Parallel Slave Port (PSP) Read .....	194
Parallel Slave Port (PSP) Write .....	193
PWM Auto-Shutdown with Auto-Restart Enabled .....	280
PWM Auto-Shutdown with Firmware Restart .....	280
PWM Direction Change .....	277
PWM Direction Change at Near 100% Duty Cycle... .....	278
PWM Output .....	262
Repeated Start Condition .....	321
Reset, Watchdog Timer (WDT), Oscillator Start-up Timer (OST) and Power-up Timer (PWRT) .....	565
Send Break Character Sequence .....	350
Slave Synchronization .....	293
Slow Rise Time (MCLR Tied to VDD, VDD Rise > TPWRT) .....	85
SPI Mode (Master Mode) .....	292
SPI Mode (Slave Mode, CKE = 0) .....	294
SPI Mode (Slave Mode, CKE = 1) .....	294
Steering Event at Beginning of Instruction (STRSYNC = 1) .....	284
Steering Event at End of Instruction (STRSYNC = 0) .....	284
Synchronous Reception (Master Mode, SREN) .....	353
Synchronous Transmission .....	351
Synchronous Transmission (Through TXEN).....	352
Time-out Sequence on POR w/ PLL Enabled (MCLR Tied to VDD).....	86
Time-out Sequence on Power-up (MCLR Not Tied to	

# PIC18F66K80 FAMILY

---

VDD), Case 1 .....	85
Time-out Sequence on Power-up (MCLR Not Tied to VDD), Case 2 .....	85
Time-out Sequence on Power-up (MCLR Tied to VDD, VDD Rise Tpwrt) .....	84
Timer0 and Timer1 External Clock .....	568
Timer1 Gate Count Enable Mode .....	217
Timer1 Gate Single Pulse Mode .....	219
Timer1 Gate Single Pulse/Toggle Combined Mode ..	220
Timer1 Gate Toggle Mode .....	218
Timer3 Gate Count Enable Mode .....	228
Timer3 Gate Single Pulse Mode .....	230
Timer3 Gate Single Pulse/Toggle Combined Mode ..	231
Timer3 Gate Toggle Mode .....	229
Transition for Entry to Idle Mode .....	71
Transition for Entry to SEC_RUN Mode .....	67
Transition for Entry to Sleep Mode .....	70
Transition for Two-Speed Start-up (INTOSC to HSPLL). 476	
Transition for Wake from Idle to Run Mode .....	71
Transition for Wake from Sleep (HSPLL).....	70
Transition from RC_RUN Mode to PRI_RUN Mode ..	69
Transition from SEC_RUN Mode to PRI_RUN Mode (HSPLL) .....	67
Transition to RC_RUN Mode .....	69
Timing Diagrams and Specifications	
Capture/Compare/PWM Requirements .....	569
CLKO and I/O Requirements .....	564, 565
EUSART/AUSART Synchronous Receive Requirements 578	
EUSART/AUSART Synchronous Transmission Requirements .....	578
Example SPI Mode Requirements (Master Mode, CKE = 0).....	570
Example SPI Mode Requirements (Master Mode, CKE = 1).....	571
Example SPI Mode Requirements (Slave Mode, CKE = 0).....	572
Example SPI Slave Mode Requirements (CKE = 1).573	
External Clock Requirements .....	562
HLVD Characteristics.....	567
I <sup>2</sup> C Bus Data Requirements (Slave Mode) .....	575
I <sup>2</sup> C Bus Start/Stop Bits Requirements (Slave Mode) 574	
Internal RC Accuracy (INTOSC) .....	563
MSSP I <sup>2</sup> C Bus Data Requirements .....	577
MSSP I <sup>2</sup> C Bus Start/Stop Bits Requirements .....	576
PLL Clock.....	563
Reset, Watchdog Timer, Oscillator Start-up Timer, Power-up Timer and Brown-out Reset Requirements ... 566	
Timer0 and Timer1 External Clock Requirements ....	568
Top-of-Stack Access .....	103
TSTFSZ.....	523
Two-Speed Start-up .....	457, 476
IESO (CONFIG1H, Internal/External Oscillator Switchover Bit.....	460
Two-Word Instructions	
Example Cases .....	107
TXSTAx Register	
BRGH Bit .....	337
<b>U</b>	
Ultra Low-Power Mode	
Regulators	
Enable Mode.....	474
Operation in Sleep .....	475
Ultra Low-Power Wake-up	
Exit Delay .....	78
Overview.....	77
<b>V</b>	
Voltage Reference Specifications .....	559
<b>W</b>	
Watchdog Timer (WDT).....	457, 472
Associated Registers.....	473
Control Register.....	473
During Oscillator Failure .....	477
Programming Considerations .....	472
WCOL.....	320, 321, 322, 325
WCOL Status Flag.....	320, 321, 322, 325
WWW Address .....	617
WWW, On-Line Support .....	9
<b>X</b>	
XORLW.....	523
XORWF .....	524

## 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). Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- 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://microchip.com/support>**

# PIC18F66K80 FAMILY

---

---

## READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

TO: Technical Publications Manager                          Total Pages Sent \_\_\_\_\_

RE: Reader Response

From: Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City / State / ZIP / Country \_\_\_\_\_

Telephone: (\_\_\_\_\_) \_\_\_\_\_ - \_\_\_\_\_                          FAX: (\_\_\_\_\_) \_\_\_\_\_ - \_\_\_\_\_

Application (optional):

Would you like a reply?    Y    N

Device: PIC18F66K80 Family

Literature Number: DS39977F

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?

---

# **PIC18F66K80 FAMILY**

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, such as pricing or delivery, refer to the factory or the listed sales office.

<u>PART NO.</u>	X	/XX	XXX	
Device	Temperature Range	Package	Pattern	
Device <sup>(1,2)</sup>	PIC18F25K80, PIC18F26K80, PIC18F45K80, PIC18F46K80, PIC18F65K80, PIC18F66K80 VDD range 1.8V to 5V  PIC18LF25K80, PIC18LF26K80, PIC18LF45K80, PIC18LF46K80, PIC18F65K80, PIC18F66K80 VDD range 1.8V to 3.6V			
Temperature Range	I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended)			
Package	P = PDIP Plastic Dual In-Line ML = QFN Plastic Quad Flat, No Lead Package SO = SOIC Plastic Small Outline SP = SPDIP Skinny Plastic Dual In-Line SS = SSOP Plastic Shrink Small Outline PT = TQFP Plastic Thin Quad Flatpack			
Pattern	a) QTP, SQTP, Code or Special Requirements (blank otherwise)			

# **PIC18F66K80 FAMILY**

---

---

## **NOTES:**

---

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

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

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

---

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, PIC<sup>32</sup> logo, rfPIC 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, chipKIT, chipKIT logo, 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, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rFLAB, 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.

© 2010-2012, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-62076-074-1

---

**QUALITY MANAGEMENT SYSTEM  
CERTIFIED BY DNV  
= ISO/TS 16949 =**

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMS, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*



# MICROCHIP

## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://www.microchip.com/support>  
Web Address:  
[www.microchip.com](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

**Indianapolis**

Noblesville, IN  
Tel: 317-773-8323  
Fax: 317-773-5453

**Los Angeles**

Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**

Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**

Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**

Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**

Tel: 86-10-8569-7000  
Fax: 86-10-8528-2104

**China - Chengdu**

Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Chongqing**

Tel: 86-23-8980-9588  
Fax: 86-23-8980-9500

**China - Hangzhou**

Tel: 86-571-2819-3187  
Fax: 86-571-2819-3189

**China - Hong Kong SAR**

Tel: 852-2401-1200  
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 - Xian**

Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Xiamen**

Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Zhuhai**

Tel: 86-756-3210040

Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4123

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**

Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Osaka**

Tel: 81-66-152-7160  
Fax: 81-66-152-9310

**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: 886-3-5778-366  
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**

Tel: 886-7-536-4818  
Fax: 886-7-330-9305

**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

11/29/11