



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



18-pin Flash/EEPROM 8-Bit Microcontrollers

Devices Included in this Data Sheet:

- PIC16F83
- PIC16F84
- PIC16CR83
- PIC16CR84
- Extended voltage range devices available (PIC16LF8X, PIC16LCR8X)

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single cycle except for program branches which are two-cycle
- Operating speed: DC - 10 MHz clock input
DC - 400 ns instruction cycle

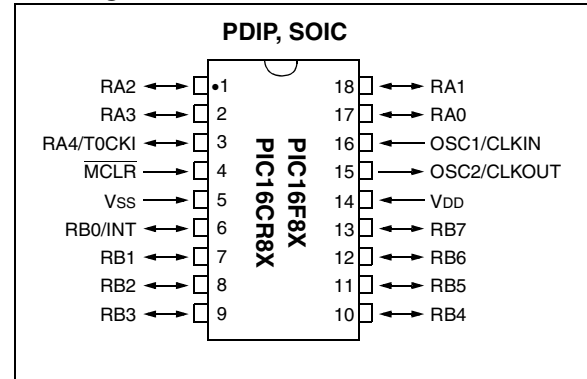
Device	Program Memory (words)	Data RAM (bytes)	Data EEPROM (bytes)	Max. Freq (MHz)
PIC16F83	512 Flash	36	64	10
PIC16F84	1 K Flash	68	64	10
PIC16CR83	512 ROM	36	64	10
PIC16CR84	1 K ROM	68	64	10

- 14-bit wide instructions
- 8-bit wide data path
- 15 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
 - External RB0/INT pin
 - TMR0 timer overflow
 - PORTB<7:4> interrupt on change
 - Data EEPROM write complete
- 1000 erase/write cycles Flash program memory
- 10,000,000 erase/write cycles EEPROM data memory
- EEPROM Data Retention > 40 years

Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
 - 25 mA sink max. per pin
 - 20 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Pin Diagrams



Special Microcontroller Features:

- In-Circuit Serial Programming (ICSP™) - via two pins (ROM devices support only Data EEPROM programming)
- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code-protection
- Power saving SLEEP mode
- Selectable oscillator options

CMOS Flash/EEPROM Technology:

- Low-power, high-speed technology
- Fully static design
- Wide operating voltage range:
 - Commercial: 2.0V to 6.0V
 - Industrial: 2.0V to 6.0V
- Low power consumption:
 - < 2 mA typical @ 5V, 4 MHz
 - 15 μA typical @ 2V, 32 kHz
 - < 1 μA typical standby current @ 2V

PIC16F8X

Table of Contents

1.0	General Description	3
2.0	PIC16F8X Device Varieties	5
3.0	Architectural Overview	7
4.0	Memory Organization	11
5.0	I/O Ports	21
6.0	Timer0 Module and TMR0 Register	27
7.0	Data EEPROM Memory	33
8.0	Special Features of the CPU	37
9.0	Instruction Set Summary	53
10.0	Development Support	69
11.0	Electrical Characteristics for PIC16F83 and PIC16F84	73
12.0	Electrical Characteristics for PIC16CR83 and PIC16CR84	85
13.0	DC & AC Characteristics Graphs/Tables	97
14.0	Packaging Information	109
Appendix A:	Feature Improvements - From PIC16C5X To PIC16F8X	113
Appendix B:	Code Compatibility - from PIC16C5X to PIC16F8X	113
Appendix C:	What's New In This Data Sheet	114
Appendix D:	What's Changed In This Data Sheet	114
Appendix E:	Conversion Considerations - PIC16C84 to PIC16F83/F84 And PIC16CR83/CR84	115
Index	117
On-Line Support	119
Reader Response	120
PIC16F8X Product Identification System	121
Sales and Support	121

To Our Valued Customers

We constantly strive to improve the quality of all our products and documentation. We have spent a great deal of time to ensure that these documents are correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error, please use the reader response form in the back of this data sheet to inform us. We appreciate your assistance in making this a better document.

1.0 GENERAL DESCRIPTION

The PIC16F8X is a group in the PIC16CXX family of low-cost, high-performance, CMOS, fully-static, 8-bit microcontrollers. This group contains the following devices:

- PIC16F83
- PIC16F84
- PIC16CR83
- PIC16CR84

All PIC[®] microcontrollers employ an advanced RISC architecture. PIC16F8X devices have enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with a separate 8-bit wide data bus. The two stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available. Additionally, a large register set is used to achieve a very high performance level.

PIC16F8X microcontrollers typically achieve a 2:1 code compression and up to a 4:1 speed improvement (at 20 MHz) over other 8-bit microcontrollers in their class.

The PIC16F8X has up to 68 bytes of RAM, 64 bytes of Data EEPROM memory, and 13 I/O pins. A timer/counter is also available.

The PIC16CXX family has special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for High Speed crystals. The SLEEP (power-down) mode offers power saving. The user can wake the chip from sleep through several external and internal interrupts and resets.

A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software lock-up.

The devices with Flash program memory allow the same device package to be used for prototyping and production. In-circuit reprogrammability allows the code to be updated without the device being removed from the end application. This is useful in the development of many applications where the device may not be easily accessible, but the prototypes may require code updates. This is also useful for remote applications where the code may need to be updated (such as rate information).

Table 1-1 lists the features of the PIC16F8X. A simplified block diagram of the PIC16F8X is shown in Figure 3-1.

The PIC16F8X fits perfectly in applications ranging from high speed automotive and appliance motor control to low-power remote sensors, electronic locks, security devices and smart cards. The Flash/EEPROM technology makes customization of application programs (transmitter codes, motor speeds, receiver frequencies, security codes, etc.) extremely fast and convenient. The small footprint packages make this microcontroller series perfect for all applications with space limitations. Low-cost, low-power, high performance, ease-of-use and I/O flexibility make the PIC16F8X very versatile even in areas where no microcontroller use has been considered before (e.g., timer functions; serial communication; capture, compare and PWM functions; and co-processor applications).

The serial in-system programming feature (via two pins) offers flexibility of customizing the product after complete assembly and testing. This feature can be used to serialize a product, store calibration data, or program the device with the current firmware before shipping.

1.1 Family and Upward Compatibility

Those users familiar with the PIC16C5X family of microcontrollers will realize that this is an enhanced version of the PIC16C5X architecture. Please refer to Appendix A for a detailed list of enhancements. Code written for PIC16C5X devices can be easily ported to PIC16F8X devices (Appendix B).

1.2 Development Support

The PIC16CXX family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a low-cost development programmer and a full-featured programmer. A “C” compiler and fuzzy logic support tools are also available.

PIC16F8X

TABLE 1-1 PIC16F8X FAMILY OF DEVICES

		PIC16F83	PIC16CR83	PIC16F84	PIC16CR84
Clock	Maximum Frequency of Operation (MHz)	10	10	10	10
	Flash Program Memory	512	—	1K	—
Memory	EEPROM Program Memory	—	—	—	—
	ROM Program Memory	—	512	—	1K
	Data Memory (bytes)	36	36	68	68
Peripherals	Data EEPROM (bytes)	64	64	64	64
	Timer Module(s)	TMR0	TMR0	TMR0	TMR0
	Interrupt Sources	4	4	4	4
Features	I/O Pins	13	13	13	13
	Voltage Range (Volts)	2.0-6.0	2.0-6.0	2.0-6.0	2.0-6.0
	Packages	18-pin DIP, SOIC	18-pin DIP, SOIC	18-pin DIP, SOIC	18-pin DIP, SOIC

All PIC® Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability. All PIC16F8X Family devices use serial programming with clock pin RB6 and data pin RB7.

2.0 PIC16F8X DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements the proper device option can be selected using the information in this section. When placing orders, please use the "PIC16F8X Product Identification System" at the back of this data sheet to specify the correct part number.

There are four device "types" as indicated in the device number.

1. **F**, as in PIC16**F**84. These devices have Flash program memory and operate over the standard voltage range.
2. **LF**, as in PIC16**LF**84. These devices have Flash program memory and operate over an extended voltage range.
3. **CR**, as in PIC16**CR**83. These devices have ROM program memory and operate over the standard voltage range.
4. **LCR**, as in PIC16**LCR**84. These devices have ROM program memory and operate over an extended voltage range.

When discussing memory maps and other architectural features, the use of **F** and **CR** also implies the **LF** and **LCR** versions.

2.1 Flash Devices

These devices are offered in the lower cost plastic package, even though the device can be erased and reprogrammed. This allows the same device to be used for prototype development and pilot programs as well as production.

A further advantage of the electrically-erasable Flash version is that it can be erased and reprogrammed in-circuit, or by device programmers, such as Microchip's PICSTART[®] Plus or PRO MATE[®] II programmers.

2.2 Quick-Turnaround-Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who choose not to program a medium to high quantity of units and whose code patterns have stabilized. The devices have all Flash locations and configuration options already programmed by the factory. Certain code and prototype verification procedures do apply before production shipments are available.

For information on submitting a QTP code, please contact your Microchip Regional Sales Office.

2.3 Serialized Quick-Turnaround-Production (SQTPSM) Devices

Microchip offers the unique programming service where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number which can serve as an entry-code, password or ID number.

For information on submitting a SQTP code, please contact your Microchip Regional Sales Office.

2.4 ROM Devices

Some of Microchip's devices have a corresponding device where the program memory is a ROM. These devices give a cost savings over Microchip's traditional user programmed devices (EPROM, EEPROM).

ROM devices (PIC16CR8X) do not allow serialization information in the program memory space. The user may program this information into the Data EEPROM.

For information on submitting a ROM code, please contact your Microchip Regional Sales Office.

PIC16F8X

NOTES:

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16CXX family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16CXX uses a Harvard architecture. This architecture has the program and data accessed from separate memories. So the device has a program memory bus and a data memory bus. This improves bandwidth over traditional von Neumann architecture where program and data are fetched from the same memory (accesses over the same bus). Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. PIC16CXX opcodes are 14-bits wide, enabling single word instructions. The full 14-bit wide program memory bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions (Example 3-1). Consequently, all instructions execute in a single cycle except for program branches.

The PIC16F83 and PIC16CR83 address 512 x 14 of program memory, and the PIC16F84 and PIC16CR84 address 1K x 14 program memory. All program memory is internal.

The PIC16CXX can directly or indirectly address its register files or data memory. All special function registers including the program counter are mapped in the data memory. An orthogonal (symmetrical) instruction set makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC16CXX simple yet efficient. In addition, the learning curve is reduced significantly.

PIC16F8X

PIC16CXX devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8-bits wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register), and the other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register. The C and DC bits operate as a borrow and digit borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

A simplified block diagram for the PIC16F8X is shown in Figure 3-1, its corresponding pin description is shown in Table 3-1.

FIGURE 3-1: PIC16F8X BLOCK DIAGRAM

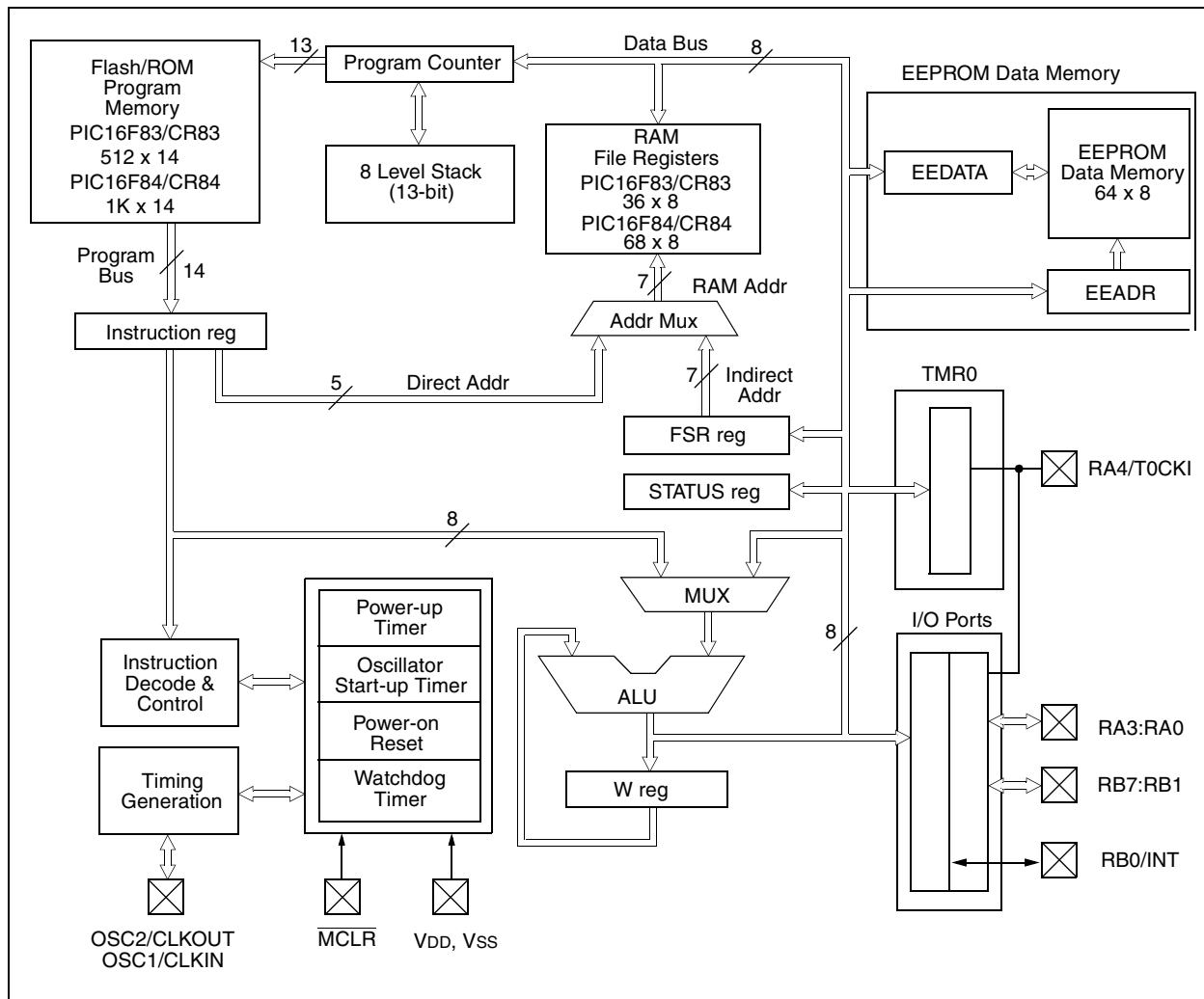


TABLE 3-1 PIC16F8X PINOUT DESCRIPTION

Pin Name	DIP No.	SOIC No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	I	ST/CMOS ⁽³⁾	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
RA0	17	17	I/O	TTL	PORTA is a bi-directional I/O port. Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
RA1	18	18	I/O	TTL	
RA2	1	1	I/O	TTL	
RA3	2	2	I/O	TTL	
RA4/T0CKI	3	3	I/O	ST	
RB0/INT	6	6	I/O	TTL/ST ⁽¹⁾	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. RB0/INT can also be selected as an external interrupt pin. Interrupt on change pin. Interrupt on change pin. Interrupt on change pin. Serial programming clock. Interrupt on change pin. Serial programming data.
RB1	7	7	I/O	TTL	
RB2	8	8	I/O	TTL	
RB3	9	9	I/O	TTL	
RB4	10	10	I/O	TTL	
RB5	11	11	I/O	TTL	
RB6	12	12	I/O	TTL/ST ⁽²⁾	
RB7	13	13	I/O	TTL/ST ⁽²⁾	
Vss	5	5	P	—	Ground reference for logic and I/O pins.
Vdd	14	14	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = Input/Output P = power
 — = Not used TTL = TTL input ST = Schmitt Trigger input

- Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
 Note 2: This buffer is a Schmitt Trigger input when used in serial programming mode.
 Note 3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

PIC16F8X

3.1 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2.

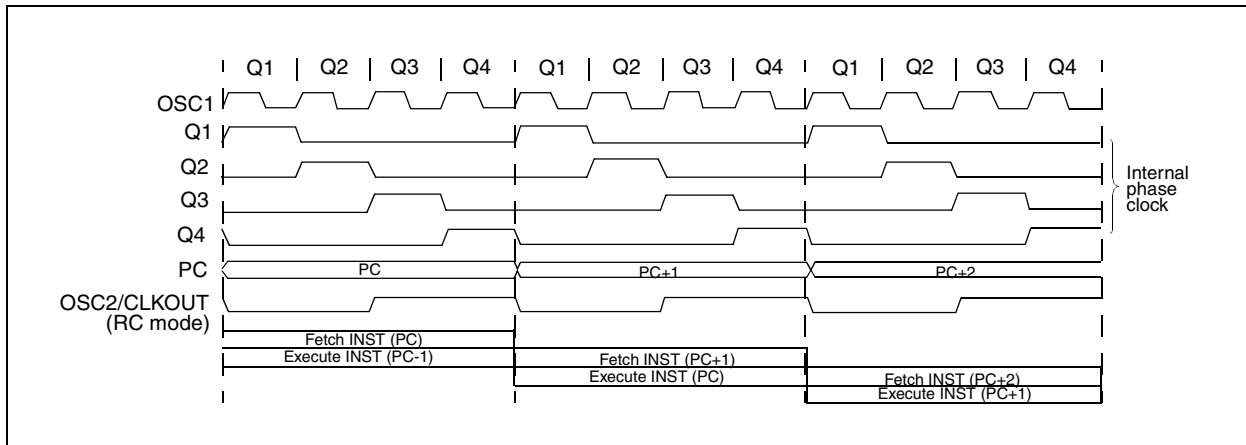
3.2 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO) then two cycles are required to complete the instruction (Example 3-1).

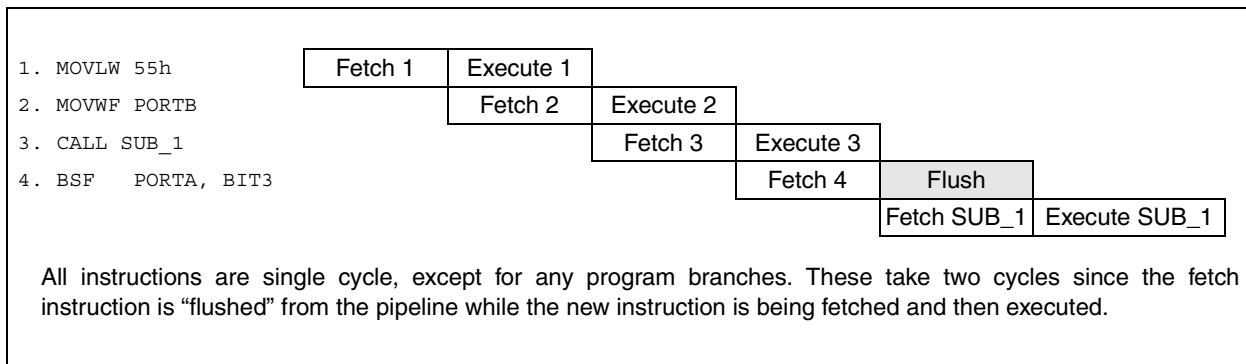
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the "Instruction Register" 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 3-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW



4.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F8X. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 7.0.

4.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F83 and PIC16CR83, the first 512 x 14 (0000h-01FFh) are physically implemented (Figure 4-1). For the PIC16F84 and PIC16CR84, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 4-2). Accessing a location above the physically implemented address will cause a wrap-around. For example, for the PIC16F84 locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h will be the same instruction.

The reset vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK - PIC16F83/CR83

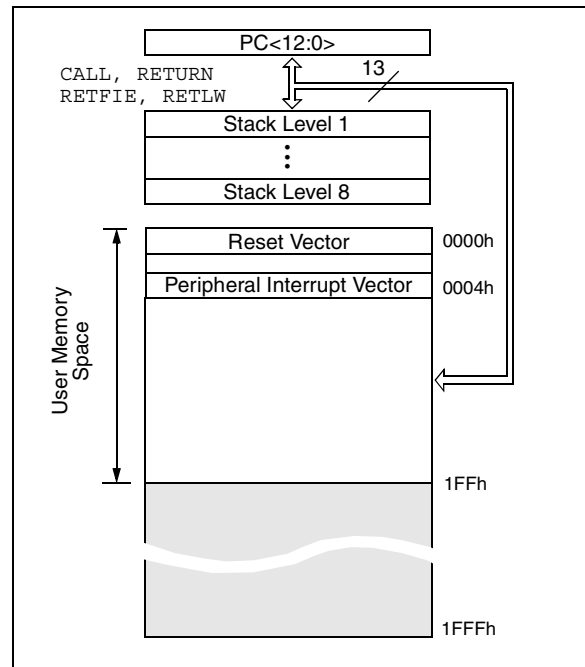
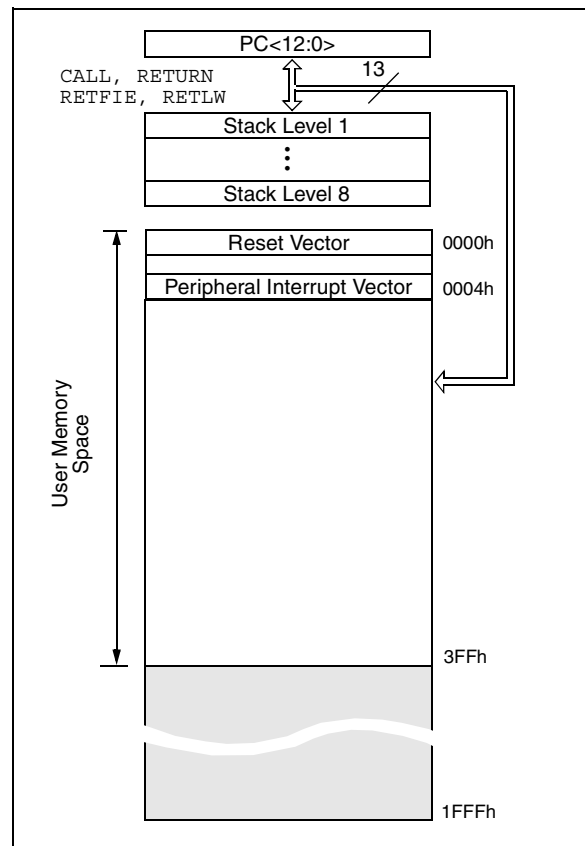


FIGURE 4-2: PROGRAM MEMORY MAP AND STACK - PIC16F84/CR84



4.2 Data Memory Organization

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. This is for both the SFR area and the GPR area. The GPR area is banked to allow greater than 116 bytes of general purpose RAM. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Figure 4-1 and Figure 4-2 show the data memory map organization.

Instructions `MOVWF` and `MOVF` can move values from the W register to any location in the register file ("F"), and vice-versa.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR) (Section 4.5). Indirect addressing uses the present value of the RP1:RP0 bits for access into the banked areas of data memory.

Data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers implemented as static RAM.

4.2.1 GENERAL PURPOSE REGISTER FILE

All devices have some amount of General Purpose Register (GPR) area. Each GPR is 8 bits wide and is accessed either directly or indirectly through the FSR (Section 4.5).

The GPR addresses in bank 1 are mapped to addresses in bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR.

4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (Figure 4-1, Figure 4-2 and Table 4-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

FIGURE 4-1: REGISTER FILE MAP - PIC16F83/CR83

File Address			File Address
00h	Indirect addr. ⁽¹⁾	Indirect addr. ⁽¹⁾	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h			87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2 ⁽¹⁾	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch			8Ch
	36 General Purpose registers (SRAM)	Mapped (accesses) in Bank 0	
2Fh			AFh
30h			B0h
7Fh			FFh
	Bank 0	Bank 1	

Unimplemented data memory location; read as '0'.

Note 1: Not a physical register.

FIGURE 4-2: REGISTER FILE MAP - PIC16F84/CR84

File Address			File Address
00h	Indirect addr. ⁽¹⁾	Indirect addr. ⁽¹⁾	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h			87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2 ⁽¹⁾	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch			8Ch
	68 General Purpose registers (SRAM)	Mapped (accesses) in Bank 0	
4Fh			CFh
50h			D0h
7Fh			FFh
	Bank 0	Bank 1	

Unimplemented data memory location; read as '0'.

Note 1: Not a physical register.

PIC16F8X

TABLE 4-1 REGISTER FILE SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)		
Bank 0													
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								----	----		
01h	TMR0	8-bit real-time clock/counter								xxxx	xxxx	uuuu	uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000	0000	0000	0000
03h	STATUS ⁽²⁾	IRP	RP1	RP0	\overline{TO}	\overline{PD}	Z	DC	C	0001	1xxx	000q	quuu
04h	FSR	Indirect data memory address pointer 0								xxxx	xxxx	uuuu	uuuu
05h	PORTA	—	—	—	RA4/ \overline{TOCKI}	RA3	RA2	RA1	RA0	---x	xxxx	---	uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx	xxxx	uuuu	uuuu
07h		Unimplemented location, read as '0'								----	----	----	----
08h	EEDATA	EEPROM data register								xxxx	xxxx	uuuu	uuuu
09h	EEADR	EEPROM address register								xxxx	xxxx	uuuu	uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC ⁽¹⁾				---	0000	---	0000	
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000	000x	0000	000u
Bank 1													
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								----	----	----	----
81h	OPTION_REG	\overline{RBPU}	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111	1111	1111	1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000	0000	0000	0000
83h	STATUS ⁽²⁾	IRP	RP1	RP0	\overline{TO}	\overline{PD}	Z	DC	C	0001	1xxx	000q	quuu
84h	FSR	Indirect data memory address pointer 0								xxxx	xxxx	uuuu	uuuu
85h	TRISA	—	—	—	PORTA data direction register				---	1111	---	1111	
86h	TRISB	PORTB data direction register								1111	1111	1111	1111
87h		Unimplemented location, read as '0'								----	----	----	----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---	0x000	---	0q000
89h	EECON2	EEPROM control register 2 (not a physical register)								----	----	----	----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC ⁽¹⁾				---	0000	---	0000	
0Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000	000x	0000	000u

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

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The \overline{TO} and \overline{PD} status bits in the STATUS register are not affected by a \overline{MCLR} reset.

3: Other (non power-up) resets include: external reset through \overline{MCLR} and the Watchdog Timer Reset.

4.2.2.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the \overline{TO} and \overline{PD} bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where u = unchanged).

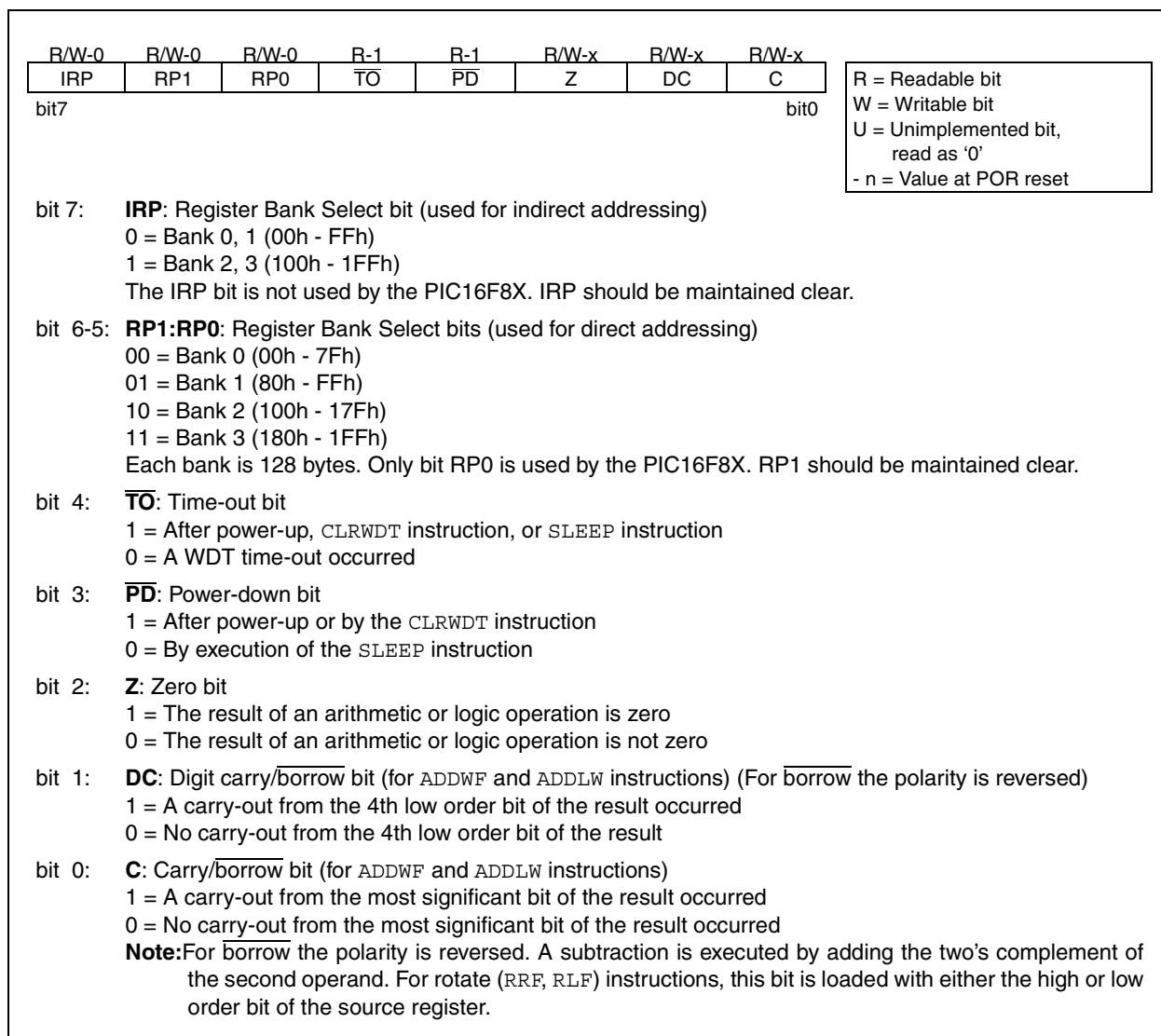
Only the `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions should be used to alter the STATUS register (Table 9-2) because these instructions do not affect any status bit.

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F8X and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

Note 2: The C and DC bits operate as a $\overline{\text{borrow}}$ and digit borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

Note 3: When the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. The specified bit(s) will be updated according to device logic

FIGURE 4-1: STATUS REGISTER (ADDRESS 03h, 83h)



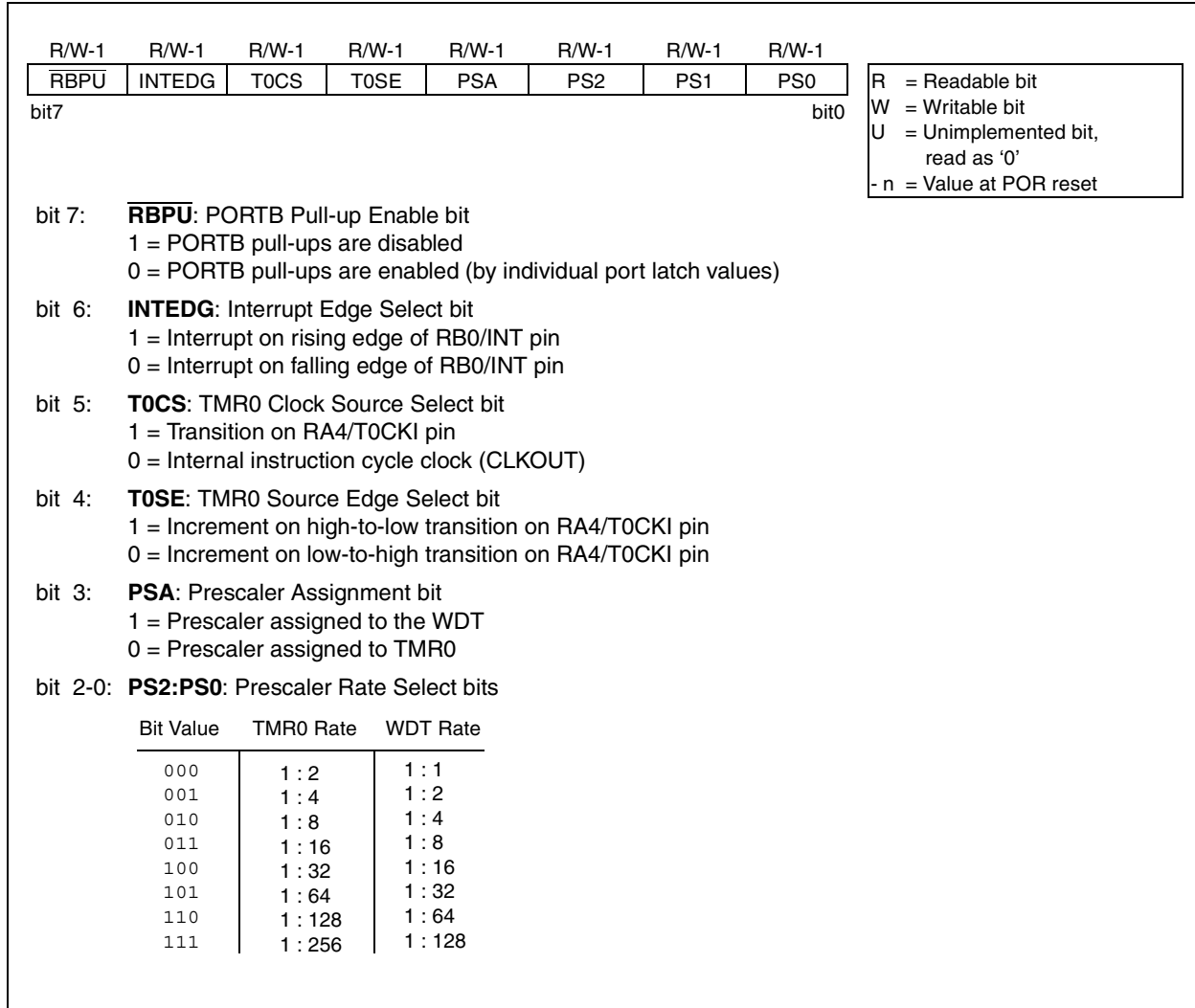
PIC16F8X

4.2.2.2 OPTION_REG REGISTER

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external INT interrupt, TMR0, and the weak pull-ups on PORTB.

Note: When the prescaler is assigned to the WDT (PSA = '1'), TMR0 has a 1:1 prescaler assignment.

FIGURE 4-1: OPTION_REG REGISTER (ADDRESS 81h)

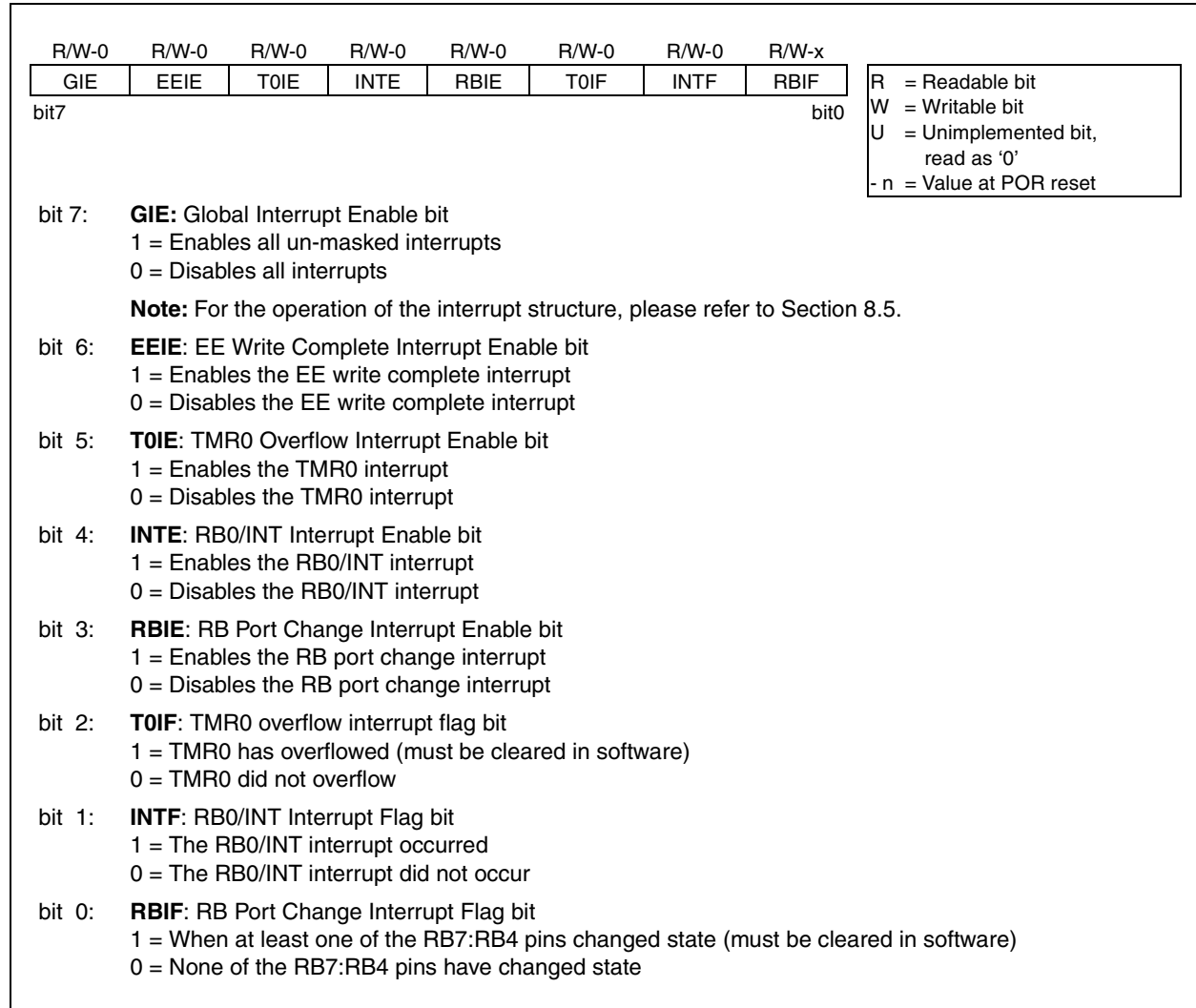


4.2.2.3 INTCON REGISTER

The INTCON register is a readable and writable register which contains the various enable bits for all interrupt sources.

Note: Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

FIGURE 4-1: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

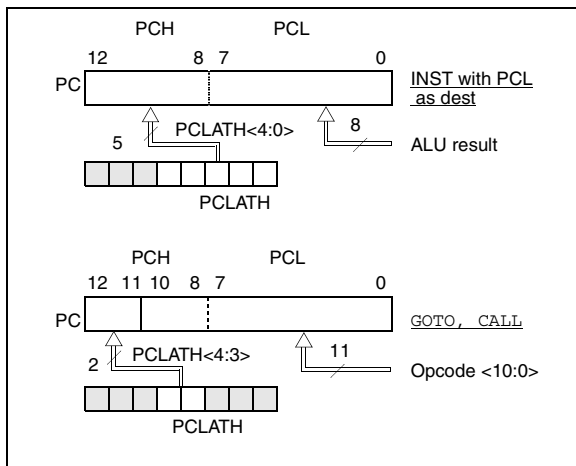


PIC16F8X

4.3 Program Counter: PCL and PCLATH

The Program Counter (PC) is 13-bits wide. The low byte is the PCL register, which is a readable and writable register. The high byte of the PC (PC<12:8>) is not directly readable nor writable and comes from the PCLATH register. The PCLATH (PC latch high) register is a holding register for PC<12:8>. The contents of PCLATH are transferred to the upper byte of the program counter when the PC is loaded with a new value. This occurs during a `CALL`, `GOTO` or a write to PCL. The high bits of PC are loaded from PCLATH as shown in Figure 4-1.

FIGURE 4-1: LOADING OF PC IN DIFFERENT SITUATIONS



4.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (`ADDWF PCL`). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256 word block). Refer to the application note *“Implementing a Table Read”* (AN556).

4.3.2 PROGRAM MEMORY PAGING

The PIC16F83 and PIC16CR83 have 512 words of program memory. The PIC16F84 and PIC16CR84 have 1K of program memory. The `CALL` and `GOTO` instructions have an 11-bit address range. This 11-bit address range allows a branch within a 2K program memory page size. For future PIC16F8X program memory expansion, there must be another two bits to specify the program memory page. These paging bits come from the PCLATH<4:3> bits (Figure 4-1). When doing a `CALL` or a `GOTO` instruction, the user must ensure that these page bits (PCLATH<4:3>) are programmed to the desired program memory page. If a `CALL` instruction (or interrupt) is executed, the entire 13-bit PC is “pushed” onto the stack (see next section). Therefore,

manipulation of the PCLATH<4:3> is not required for the return instructions (which “pops” the PC from the stack).

Note: The PIC16F8X ignores the PCLATH<4:3> bits, which are used for program memory pages 1, 2 and 3 (0800h - 1FFFh). The use of PCLATH<4:3> as general purpose R/W bits is not recommended since this may affect upward compatibility with future products.

4.4 Stack

The PIC16FXX has an 8 deep x 13-bit wide hardware stack (Figure 4-1). The stack space is not part of either program or data space and the stack pointer is not readable or writable.

The entire 13-bit PC is “pushed” onto the stack when a `CALL` instruction is executed or an interrupt is acknowledged. The stack is “popped” in the event of a `RETURN`, `RETLW` or a `RETFIE` instruction execution. PCLATH is not affected by a push or a pop operation.

Note: There are no instruction mnemonics called push or pop. These are actions that occur from the execution of the `CALL`, `RETURN`, `RETLW`, and `RETFIE` instructions, or the vectoring to an interrupt address.

The stack operates as a circular buffer. That is, after the stack has been pushed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

If the stack is effectively popped nine times, the PC value is the same as the value from the first pop.

Note: There are no status bits to indicate stack overflow or stack underflow conditions.

4.5 Indirect Addressing: INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

EXAMPLE 4-1: INDIRECT ADDRESSING

- Register file 05 contains the value 10h
- Register file 06 contains the value 0Ah
- Load the value 05 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 06)
- A read of the INDF register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 4-2.

EXAMPLE 4-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

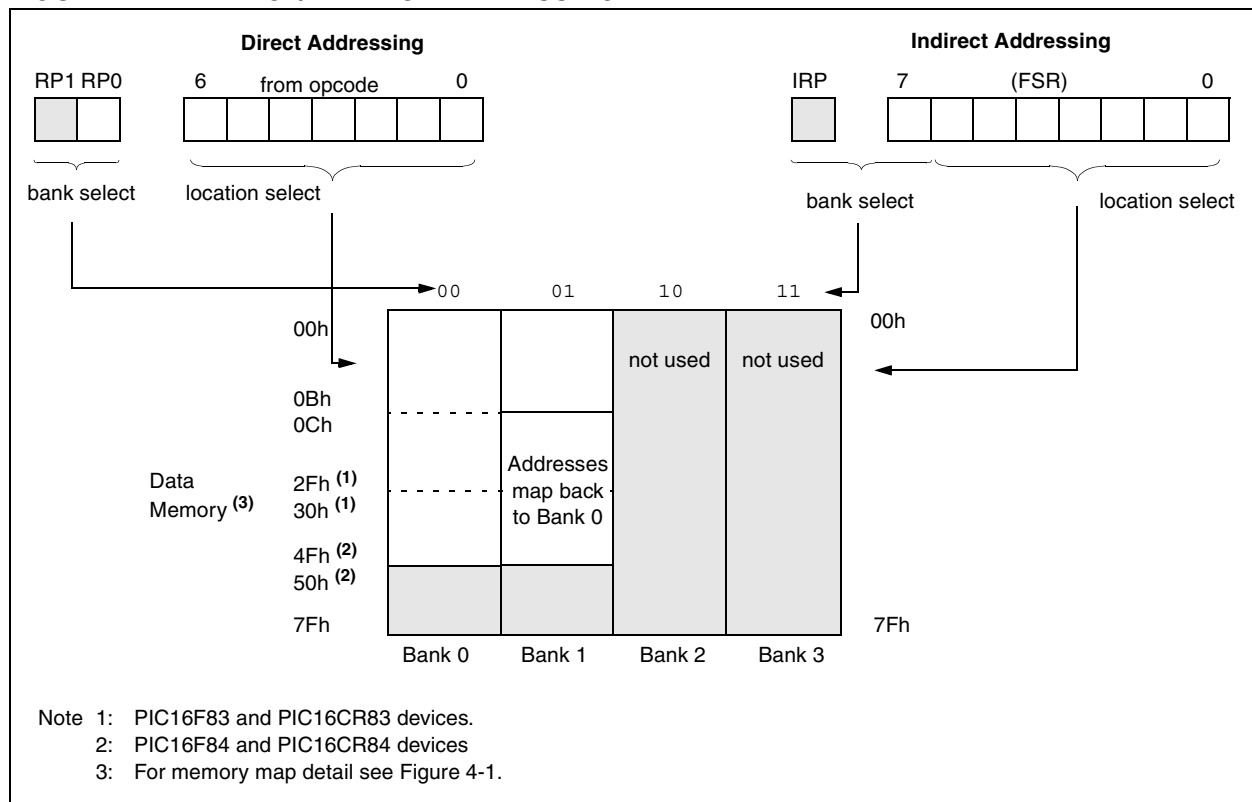
```

movlw 0x20 ;initialize pointer
movwf FSR ; to RAM
NEXT  clrf INDF ;clear INDF register
      incf FSR ;inc pointer
      btfss FSR,4 ;all done?
      goto NEXT ;NO, clear next

CONTINUE : ;YES, continue
    
```

An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-1. However, IRP is not used in the PIC16F8X.

FIGURE 4-1: DIRECT/INDIRECT ADDRESSING



PIC16F8X

NOTES:

5.0 I/O PORTS

The PIC16F8X has two ports, PORTA and PORTB. Some port pins are multiplexed with an alternate function for other features on the device.

5.1 PORTA and TRISA Registers

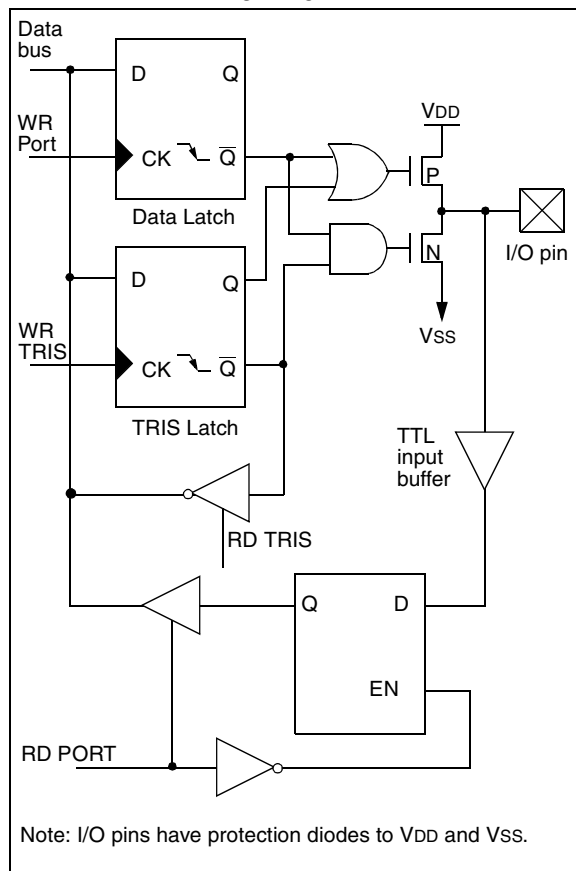
PORTA is a 5-bit wide latch. RA4 is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers. All pins have data direction bits (TRIS registers) which can configure these pins as output or input.

Setting a TRISA bit (=1) will make the corresponding PORTA pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISA bit (=0) will make the corresponding PORTA pin an output, i.e., put the contents of the output latch on the selected pin.

Reading the PORTA register reads the status of the pins whereas writing to it will write to the port latch. All write operations are read-modify-write operations. So a write to a port implies that the port pins are first read, then this value is modified and written to the port data latch.

The RA4 pin is multiplexed with the TMR0 clock input.

FIGURE 5-1: BLOCK DIAGRAM OF PINS RA3:RA0



EXAMPLE 5-1: INITIALIZING PORTA

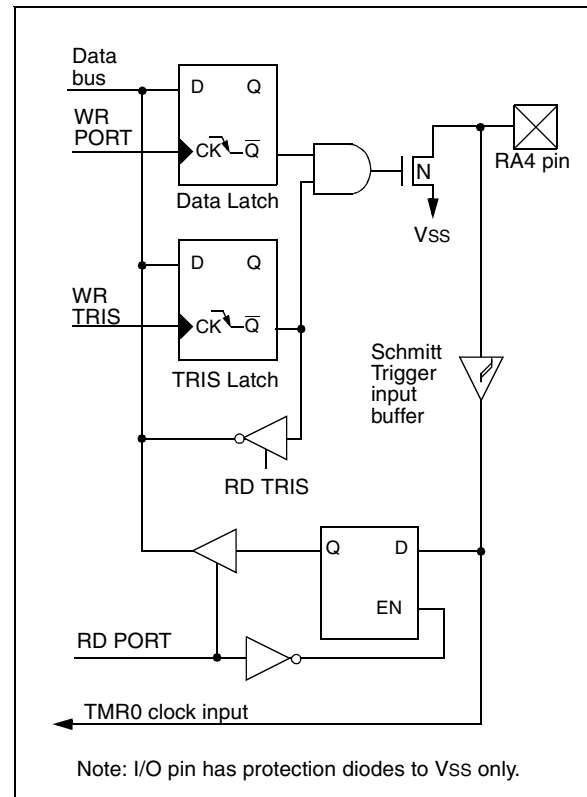
```

CLRF  PORTA      ; Initialize PORTA by
                ; setting output
                ; data latches

BSF   STATUS, RP0 ; Select Bank 1
MOVLW 0x0F      ; Value used to
                ; initialize data
                ; direction

MOVWF TRISA     ; Set RA<3:0> as inputs
                ; RA4 as outputs
                ; TRISA<7:5> are always
                ; read as '0'.
    
```

FIGURE 5-2: BLOCK DIAGRAM OF PIN RA4



PIC16F8X

TABLE 5-1 PORTA FUNCTIONS

Name	Bit0	Buffer Type	Function
RA0	bit0	TTL	Input/output
RA1	bit1	TTL	Input/output
RA2	bit2	TTL	Input/output
RA3	bit3	TTL	Input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0. Output is open drain type.

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

TABLE 5-2 SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are unimplemented, read as '0'

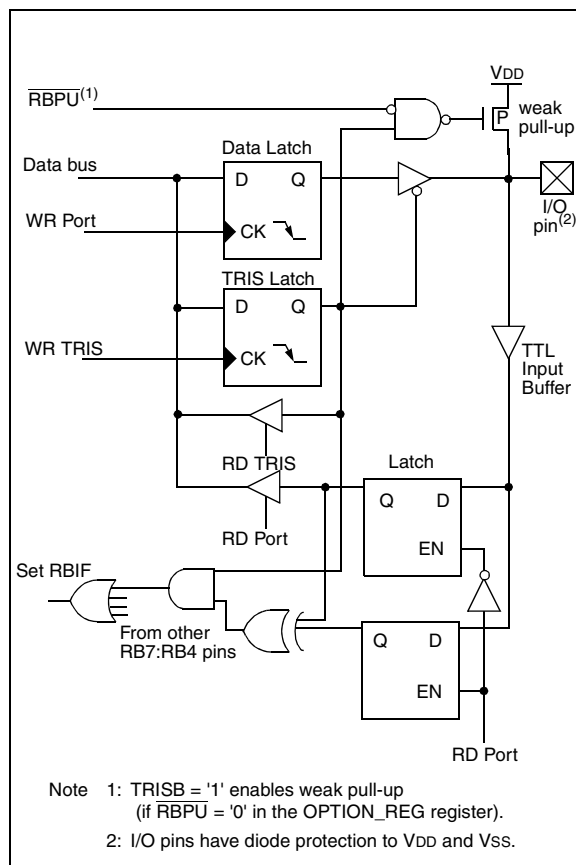
5.2 PORTB and TRISB Registers

PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. A '1' on any bit in the TRISB register puts the corresponding output driver in a hi-impedance mode. A '0' on any bit in the TRISB register puts the contents of the output latch on the selected pin(s).

Each of the PORTB pins have a weak internal pull-up. A single control bit can turn on all the pull-ups. This is done by clearing the $\overline{\text{RBP}}\text{U}$ (OPTION_REG<7>) bit. The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Four of PORTB's pins, RB7:RB4, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt on change comparison). The pins value in input mode are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of the pins are OR'ed together to generate the RB port change interrupt.

FIGURE 5-3: BLOCK DIAGRAM OF PINS RB7:RB4



This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt in the following manner:

- Read (or write) PORTB. This will end the mismatch condition.
- Clear flag bit RBIF.

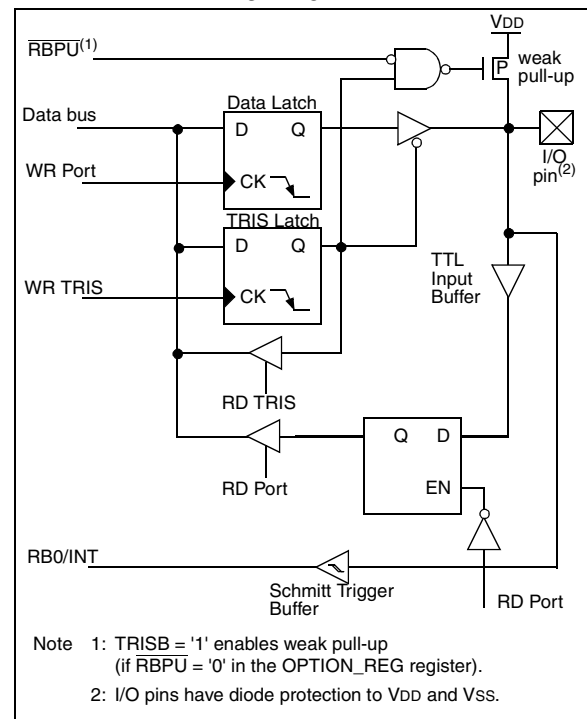
A mismatch condition will continue to set the RBIF bit. Reading PORTB will end the mismatch condition, and allow the RBIF bit to be cleared.

This interrupt on mismatch feature, together with software configurable pull-ups on these four pins allow easy interface to a key pad and make it possible for wake-up on key-depression (see AN552 in the Embedded Control Handbook).

Note 1: For a change on the I/O pin to be recognized, the pulse width must be at least T_{CY} ($4/f_{OSC}$) wide.

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

FIGURE 5-4: BLOCK DIAGRAM OF PINS RB3:RB0



PIC16F8X

EXAMPLE 5-1: INITIALIZING PORTB

```

CLRF   PORTB           ; Initialize PORTB by
                        ; setting output
                        ; data latches
BSF    STATUS, RP0    ; Select Bank 1
MOVLW  0xCF           ; Value used to
                        ; initialize data
                        ; direction
MOVWF  TRISB          ; Set RB<3:0> as inputs
                        ; RB<5:4> as outputs
                        ; RB<7:6> as inputs
    
```

TABLE 5-3 PORTB FUNCTIONS

Name	Bit	Buffer Type	I/O Consistency Function
RB0/INT	bit0	TTL/ST ⁽¹⁾	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3	bit3	TTL	Input/output pin. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB6	bit6	TTL/ST ⁽²⁾	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST ⁽²⁾	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming data.

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

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

2: This buffer is a Schmitt Trigger input when used in serial programming mode.

TABLE 5-4 SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	$\overline{\text{RBP}}\text{U}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

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

5.3 I/O Programming Considerations

5.3.1 BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read followed by a write operation. The `BCF` and `BSF` instructions, for example, read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a `BSF` operation on bit5 of `PORTB` will cause all eight bits of `PORTB` to be read into the CPU. Then the `BSF` operation takes place on bit5 and `PORTB` is written to the output latches. If another bit of `PORTB` is used as a bi-directional I/O pin (i.e., bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and rewritten to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched into output mode later on, the content of the data latch is unknown.

Reading the port register, reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (i.e., `BCF`, `BSF`, etc.) on a port, the value of the port pins is read, the desired operation is done to this value, and this value is then written to the port latch.

A pin actively outputting a Low or High should not be driven from external devices at the same time in order to change the level on this pin ("wired-or", "wired-and"). The resulting high output current may damage the chip.

5.3.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 5-5). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such that the pin voltage stabilizes (load dependent) before the next instruction which causes that file to be read into the CPU is executed. Otherwise, the previous state of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with a `NOP` or another instruction not accessing this I/O port.

Example 5-1 shows the effect of two sequential read-modify-write instructions (e.g., `BCF`, `BSF`, etc.) on an I/O port.

EXAMPLE 5-1: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

```

;Initial PORT settings: PORTB<7:4> Inputs
;                       PORTB<3:0> Outputs
;PORTB<7:6> have external pull-ups and are
;not connected to other circuitry
;
;
;                       PORT latch  PORT pins
;                       -----
;
BCF PORTB, 7      ; 01pp ppp   11pp ppp
BCF PORTB, 6      ; 10pp ppp   11pp ppp
BSF STATUS, RP0   ;
BCF TRISB, 7      ; 10pp ppp   11pp ppp
BCF TRISB, 6      ; 10pp ppp   10pp ppp
;
;Note that the user may have expected the
;pin values to be 00pp ppp. The 2nd BCF
;caused RB7 to be latched as the pin value
;(high).
    
```

FIGURE 5-5: SUCCESSIVE I/O OPERATION

