PIC16C84

8-Bit CMOS EEPROM Microcontroller

High Performance RISC-like CPU Features

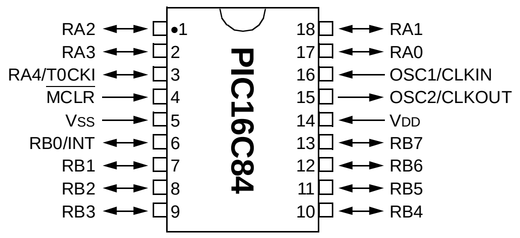
- Only 35 single word instructions to learn
- All instructions single cycle (400 ns) except for program branches which are two-cycle
- Operating speed: DC - 10 MHz clock input
  DC - 400 ns instruction cycle
- 14-bit wide instructions
- 8-bit wide data path
- 1K x 14 On-chip EEPROM program memory
- 36 x 8 general purpose registers (SRAM)
- 15 special function hardware registers
- 64 x 8 EEPROM data memory
- 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
- 1,000,000 ERASE/WRITE cycles - Typical (Data Memory)
- 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 Configuration

PDIP, SOIC



Special Microcontroller Features

- 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:
  - RC: Low-cost RC oscillator
  - XT: Standard crystal/resonator
  - HS: High-speed crystal/resonator
  - LP: Power saving, low frequency crystal
- Serial In-System Programming (via two pins)

CMOS Technology

- Low-power, high-speed CMOS EEPROM 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 @ 5V, 4 MHz
  - 60 µA typical @ 2V, 32 kHz
  - 26 µA typical standby current @ 2V

TABLE OF CONTENTS

To Our Valued Customers

We constantly strive to improve the quality of all our products and documentation. To this end, we recently converted to a new publishing software package which we believe will enhance our entire documentation process and product. As in any conversion process, information may have accidently been altered or deleted. We have spent an exceptional amount 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 from the previous version of the PIC16C84 Data Sheet (Literature Number DS30081D), 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.

To assist you in the use of this document, Appendix C contains a list of new information in this data sheet, while Appendix D contains information that has changed

1.0 GENERAL DESCRIPTION

The PIC16C84 is a low-cost, high-performance, CMOS, fully-static, 8-bit microcontroller. All PIC16/17 microcontrollers employ an advanced RISC-like architecture. PIC16C84 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 gives some of the architectural innovations used to achieve a very high performance.

PIC16C84 microcontrollers typically achieve a 2:1 code compression and a 2:1 speed improvement over other 8-bit microcontrollers in its class.

The PIC16CXX has 36 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 consumpton. 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 EEPROM program memory allows 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 PIC16C84.

Figure 3-1 is a simplified block diagram of the PIC16C84.

The PIC16C84 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 EEPROM technology makes customization of application programs (transmitter codes, motor speeds, receiver frequencies, 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 PIC16C84 very versatile even in areas where no microcontroller use has been considered before (e.g. timer functions, serial communication, capture and compare, PWM functions and co-processor applications).

The in-system programming feature (via two pins) offers flexibility of customizing the product after complete assemble and test. 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 can be easily ported to the PIC16C84 (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.

TABLE 1-1: PIC16CXX FAMILY OF DEVICES

	Clock	Memory				Peripherals							Features				
	Maximum Frequency of Operation (MHz)	EPROM Program Memory	EEPROM Program Memory	Data Memory (bytes)	Data EEPROM (bytes)	Timer Module(s)	Capture/Compare/PWM Module(s)	Serial Port(s) (SPI/I²C, SCI)	Parallel Slave Port	Analog to Digital Converter (8-bit)	Comparator(s)	Internal Reference Voltage	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Brown-out	Packages
PIC16C61	20	1K	—	36	—	TMR0	—	—	—	—	—	—	3	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C62*	20	2K	—	128	—	TMR0, TMR1, TMR2	2	SPI/I²C	—	—	—	—	10	22	2.5-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C63*	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	—	—	—	—	10	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C64	20	2K	—	128	—	TMR0, TMR1, TMR2	1	SPI/I²C	Yes	—	—	—	8	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C65	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	Yes	—	—	—	11	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C620*	20	512	—	80	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C621*	20	1K	—	80	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C622	20	2K	—	128	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C71	20	1K	—	36	—	TMR0	—	—	—	4 ch	—	—	4	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C73	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	—	5 ch	—	—	11	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C74	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	Yes	8 ch	—	—	12	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C84	10	—	1K	36	64	TMR0	—	—	—	—	—	—	4	13	2.0-6.0	—	18-pin DIP, 18-pin SOIC

* Please contact your local sales office for availability of these devices.

Note 1: All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

2: The PIC16CXX Timer1 has its own oscillator circuit and can operate asynchronously to the device. Timer1 can increment while the device is in SLEEP mode. This allows a Real Time Clock to be implemented.

3: PORTB has software-configurable weak pull-ups.

2.0 PIC16C84 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 “PIC16C84 Product Identification System” on the back page of this data sheet to specify the correct part number.

2.1 Electrically Erasable Devices

All PIC16C84 versions are electrically erasable. 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 version can be erased and reprogrammed in-circuit, or by Microchip's PICSTART™ or PRO MATE™ 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 EEPROM locations and configuration options already programmed by the factory. Certain code and prototype verification procedures do apply before production shipments are available. Please contact your Microchip's Technology sales office for more details.

2.3 Serialized Quick-Turnaround-Production (SQTP℠) 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.

NOTES:

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16CXX can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16CXX uses a Harvard architecture, in which, program and data are accessed from separate memories. This improves bandwidth over traditional von Neumann architecture where program and data are fetched from the same memory. Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. Instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions (see Example 3-1). Consequently, all instructions execute in a single cycle (400 ns @ 10 MHz) except for program branches.

The PIC16C84 addresses 1K x 14 program memory. All program memory is internal.

The PIC16C84 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 that 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 PIC16C84 simple yet efficient. In addition, the learning curve is reduced significantly.

PIC16C84 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 $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

A simplified block diagram for the PIC16C84 is shown in Figure 3-1, its corresponding pin description is shown in Table 3-1.

FIGURE 3-1: PIC16C84 BLOCK DIAGRAM

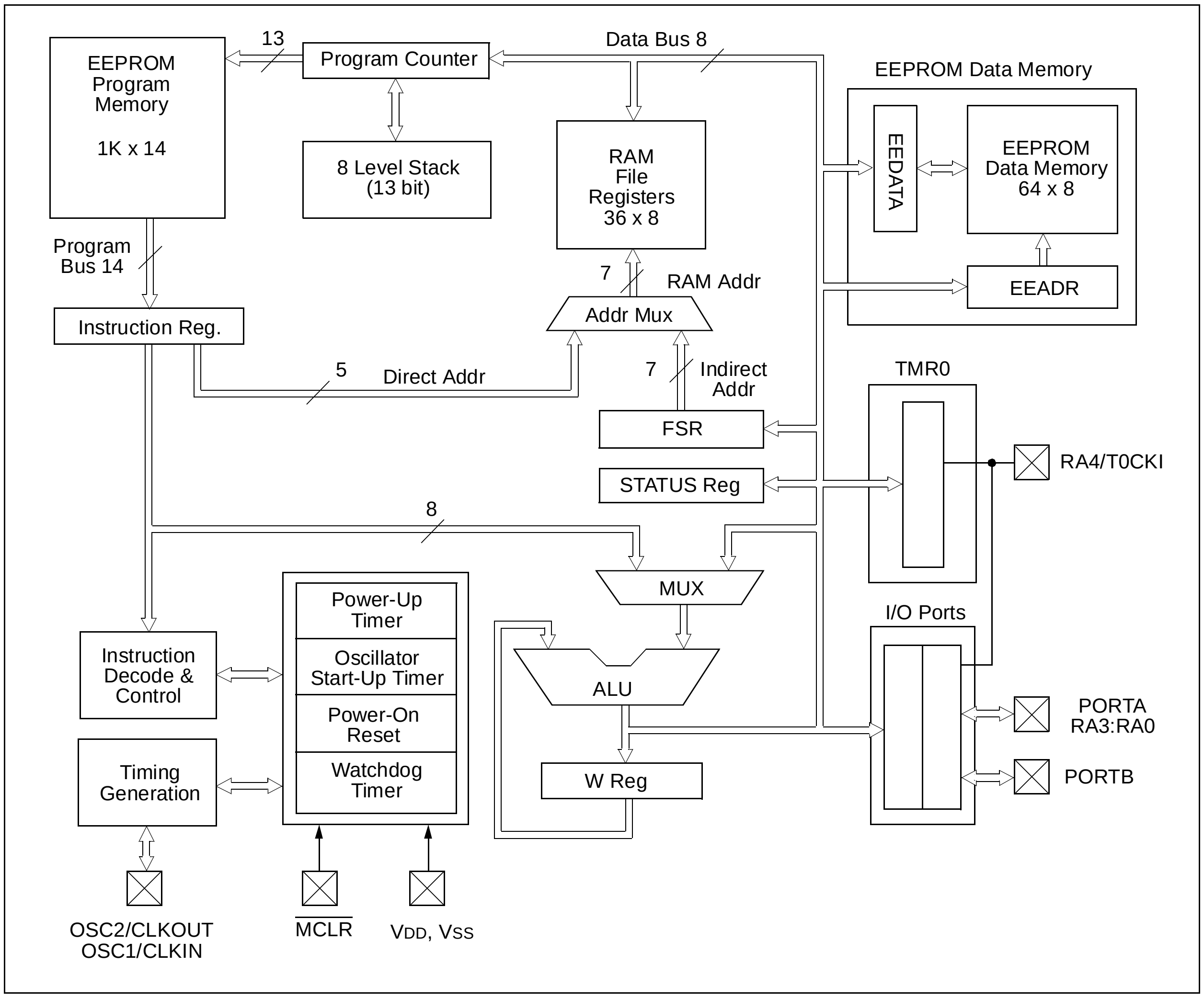


TABLE 3-1: PIC16C84 PINOUT DESCRIPTION

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.
$\overline{\text{MCLR}}$	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
					PORTA is a bi-directional I/O port.
RA0	17	17	I/O	TTL	
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	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open collector type.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	I/O	TTL	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	I/O	TTL	
RB2	8	8	I/O	TTL	
RB3	9	9	I/O	TTL	
RB4	10	10	I/O	TTL	Interrupt on change pin.
RB5	11	11	I/O	TTL	Interrupt on change pin.
RB6	12	12	I/O	TTL/ST[2]	Interrupt on change pin. Serial programming clock.
RB7	13	13	I/O	TTL/ST[2]	Interrupt on change pin. Serial programming data.
V_{SS}	5	5	P	—	Ground reference for logic and I/O pins.
V_{DD}	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.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

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.

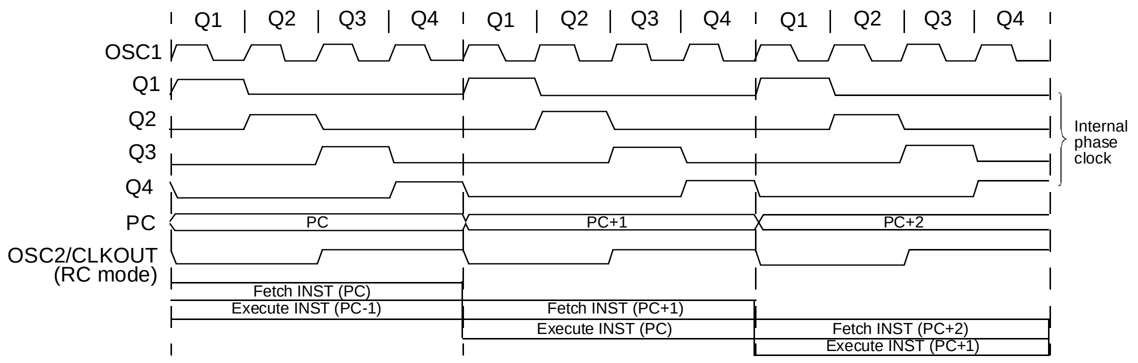
3.2 Instruction Flow/Pipelining

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 (see Example 3-1).

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

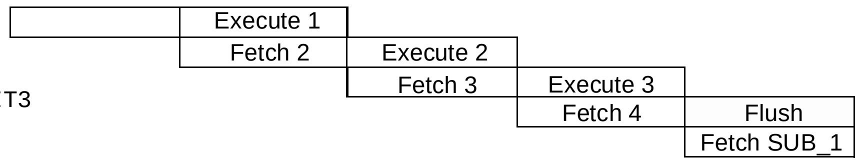
In the execution cycle, the fetched instruction is latched into the “Instruction Register” in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 3-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW

```
1. MOVLW  55h
2. MOVWF  PORTB
3. CALL   SUB_1
4. BSF    PORTA, BIT3
```



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.

4.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16C84. 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.

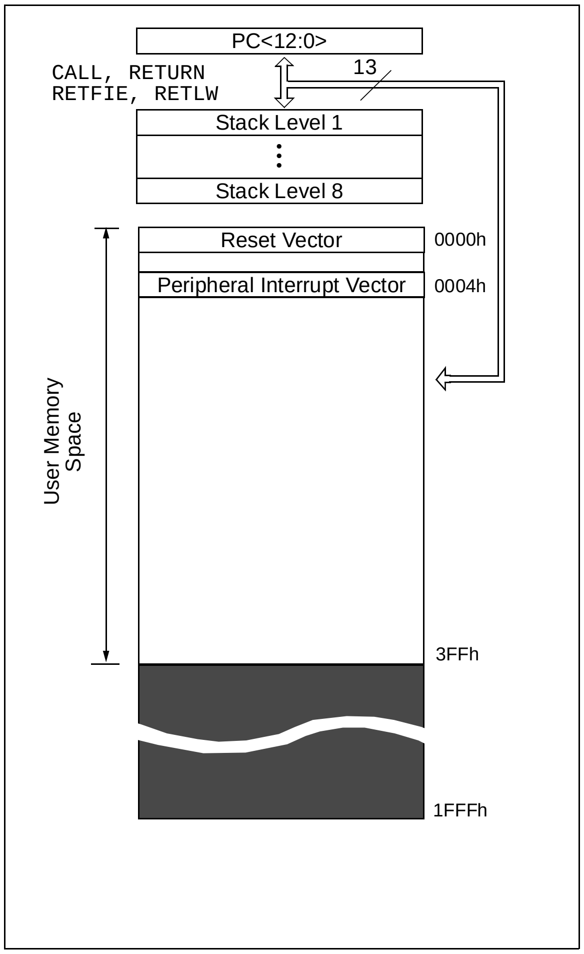
The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0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

4.1 Program Memory Organization

The PIC16C84 has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16C84 only the first 1K x 14 (0000-03FFh) are physically implemented. Accessing a location above the physically implemented address will cause a wrap-around. For example, 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).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK



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 96 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 4-2 shows 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 or indirectly through the File Select Register (FSR) (Section 4.4). 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 lower locations of each Bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers implemented as static RAM. (Figure 4-2)

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is accessed either directly, or indirectly through the FSR (Section 4.4).

All devices have some amount of GPR area. The GPR is 8-bits wide. When the GPR area is greater then 96, banking must be performed to access the additional memory space.

PIC16C84 devices do not have banked memory in the GPR area. Any access to Bank 1 will cause the access to occur in Bank 0. That is, the MSb of the direct address will be ignored.

4.2.2 SPECIAL FUNCTION REGISTERS

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

The special registers can be classified into two sets. 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 4-2: PIC16C84 REGISTER FILE MAP

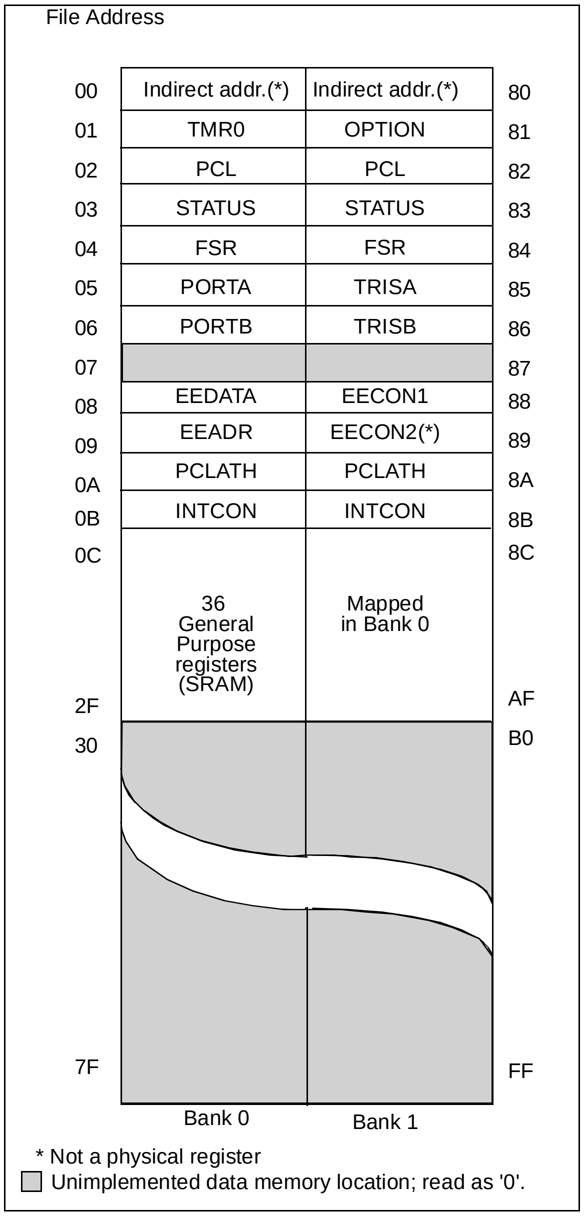


TABLE 4-1: REGISTER FILE SUMMARY

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 PC								`0000 0000`	`0000 0000`
03h	STATUS [2]	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	`0001 1xxx`	`000? ?uuu`
04h	FSR	Indirect data memory address pointer 0								`xxxx xxxx`	`uuuu uuuu`
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	`---x xxxx`	`---u uuuu`
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	`xxxx xxxx`	`uuuu uuuu`
07h										`---- ----`	`---- ----`
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]					`---0 0000`	`---0 0000`
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	`0000 000x`	`0000 0000`
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								`---- ----`	`---- ----`
81h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	`1111 1111`	`1111 1111`
82h	PCL	Low order 8 bits of PC								`0000 0000`	`0000 0000`
83h	STATUS [2]	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	`0001 1xxx`	`000? ?uuu`
84h	FSR	Indirect data memory address pointer 0								`xxxx xxxx`	`uuuu uuuu`
85h	TRISA	—	—	—	PORTA data direction register					`---1 1111`	`---1 1111`
86h	TRISB	PORTB data direction register								`1111 1111`	`1111 1111`
87h										`---- ----`	`---- ----`
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	`---0 x000`	`---0 ?000`
89h	EECON2	EEPROM control register 2 (not a physical register)								`---- ----`	`---- ----`
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC [1]					`---0 0000`	`---0 0000`
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	`0000 000x`	`0000 0000`

Legend: x = unknown, u = unchanged. - = unimplemented, read as '0', ? = 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 $\overline{\text{TO}}$ and $\overline{\text{PD}}$ status bits in STATUS are not affected by a $\overline{\text{MCLR}}$ reset.

3: Other (non power-up) resets include: external reset through $\overline{\text{MCLR}}$ and the Watchdog Timer time-out 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 $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

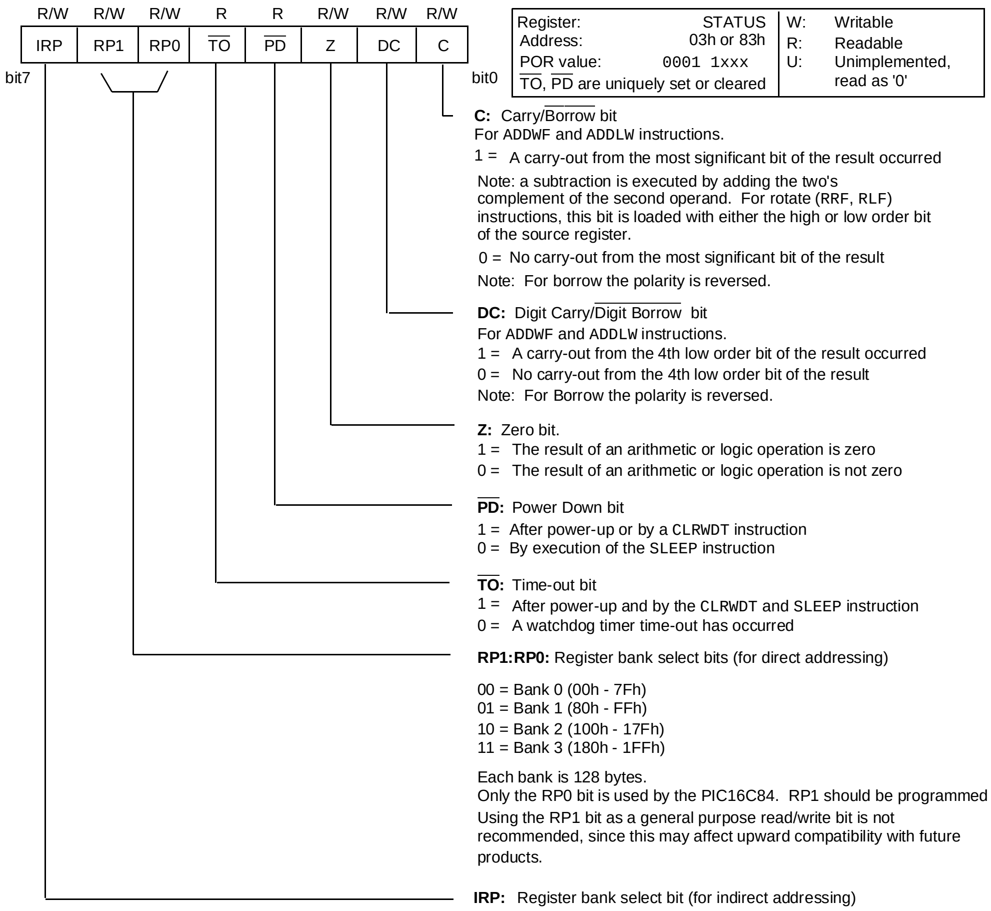
For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where u = unchanged).

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

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16C84 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.

FIGURE 4-3: STATUS REGISTER



R/W R/W R/W R R R/W R/W R/W
IRP RP1 RP0 $\overline{\text{TO}}$ $\overline{\text{PD}}$ Z DC C
bit7
bit0

Register: STATUS	W: Writable
Address: 03h or 83h	R: Readable
POR value: `0001 1xxx`	U: Unimplemented, read as '0'
$\overline{\text{TO}}$, $\overline{\text{PD}}$ are uniquely set or cleared	

C: Carry/$\overline{\text{Borrow}}$ bit
For `ADDWF` and `ADDLW` instructions.
1 = A carry-out from the most significant bit of the result occurred
Note: a subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.
0 = No carry-out from the most significant bit of the result
Note: For borrow the polarity is reversed.

DC: Digit Carry/$\overline{\text{Digit Borrow}}$ bit
For `ADDWF` and `ADDLW` instructions.
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result
Note: For Borrow the polarity is reversed.

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

$\overline{\text{PD}}$: Power Down bit
1 = After power-up or by a `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

$\overline{\text{TO}}$: Time-out bit
1 = After power-up and by the `CLRWDT` and `SLEEP` instruction
0 = A watchdog timer time-out has occurred

RP1:RP0: Register bank select bits (for direct addressing)

00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
10 = Bank 2 (100h - 17Fh)
11 = Bank 3 (180h - 1FFh)

Each bank is 128 bytes.
Only the RP0 bit is used by the PIC16C84. RP1 should be programmed as '0'.
Using the RP1 bit as a general purpose read/write bit is not recommended, since this may affect upward compatibility with future products.

IRP: Register bank select bit (for indirect addressing)
0 = Bank 0,1 (00h - FFh)
1 = Bank 2,3 (100h - 1FFh)

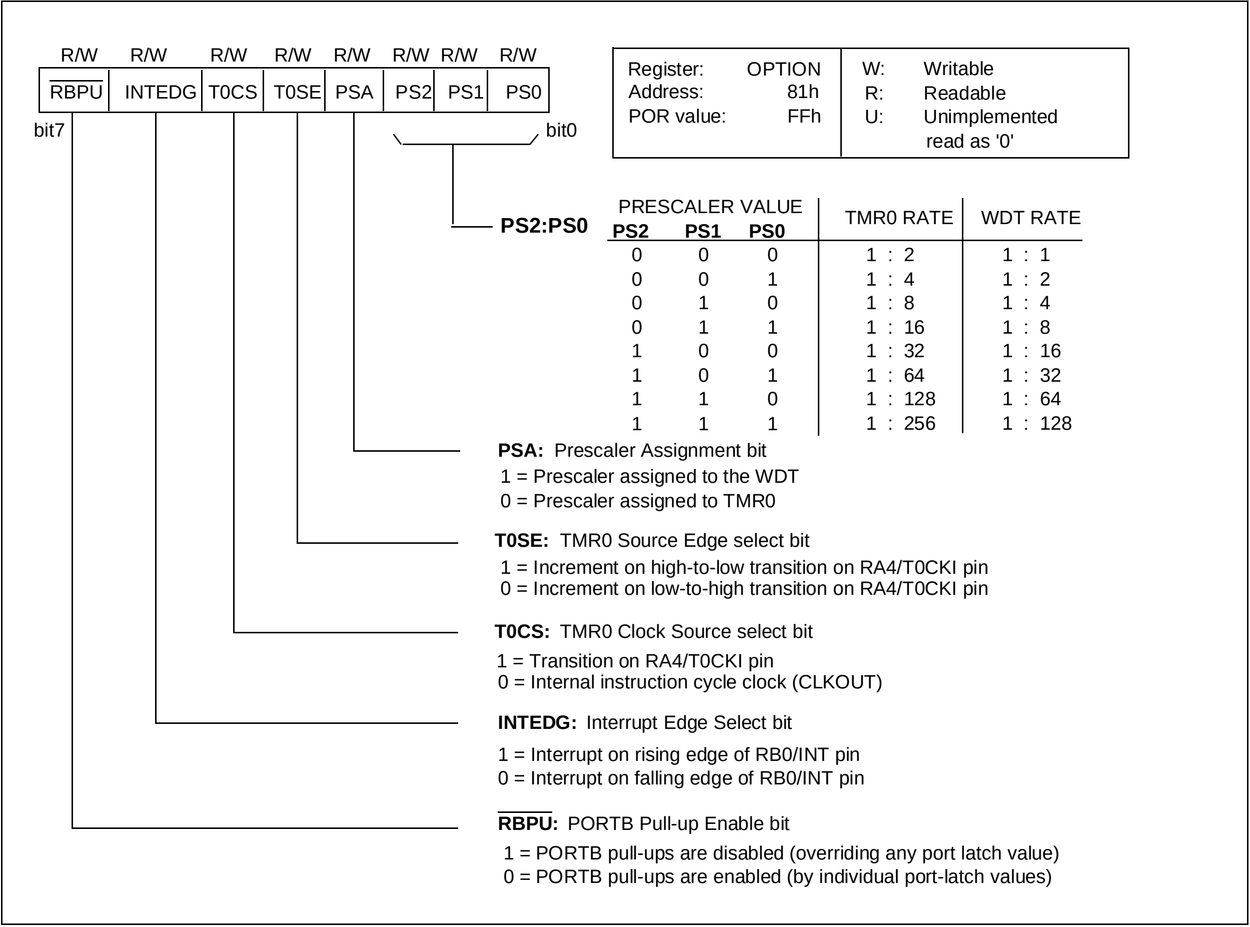
The IRP bit is not used by the PIC16C84. IRP should be programmed as '0'.
Use of the IRP bit as a general purpose read/write bit is not recommended, since this may affect upward compatibility with future products.

4.2.2.2 OPTION REGISTER

The OPTION 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 4-4: OPTION REGISTER


R/W R/W R/W R/W R/W R/W R/W R/W
$\overline{\text{RBPU}}$ INTEDG T0CS T0SE PSA PS2 PS1 PS0
bit7
bit0
PS2:PS0

Register:	OPTION	W:	Writable
Address:	81h	R:	Readable
POR value:	FFh	U:	Unimplemented read as '0'

PRESCALER VALUE			TMR0 RATE	WDT RATE
PS2	**PS1**	**PS0**		
0	0	0	1 : 2	1 : 1
0	0	1	1 : 4	1 : 2
0	1	0	1 : 8	1 : 4
0	1	1	1 : 16	1 : 8
1	0	0	1 : 32	1 : 16
1	0	1	1 : 64	1 : 32
1	1	0	1 : 128	1 : 64
1	1	1	1 : 256	1 : 128

PSA: Prescaler Assignment bit
1 = Prescaler assigned to the WDT
0 = Prescaler assigned to TMR0

T0SE: TMR0 Source Edge select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

T0CS: TMR0 Clock Source select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

INTEDG: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

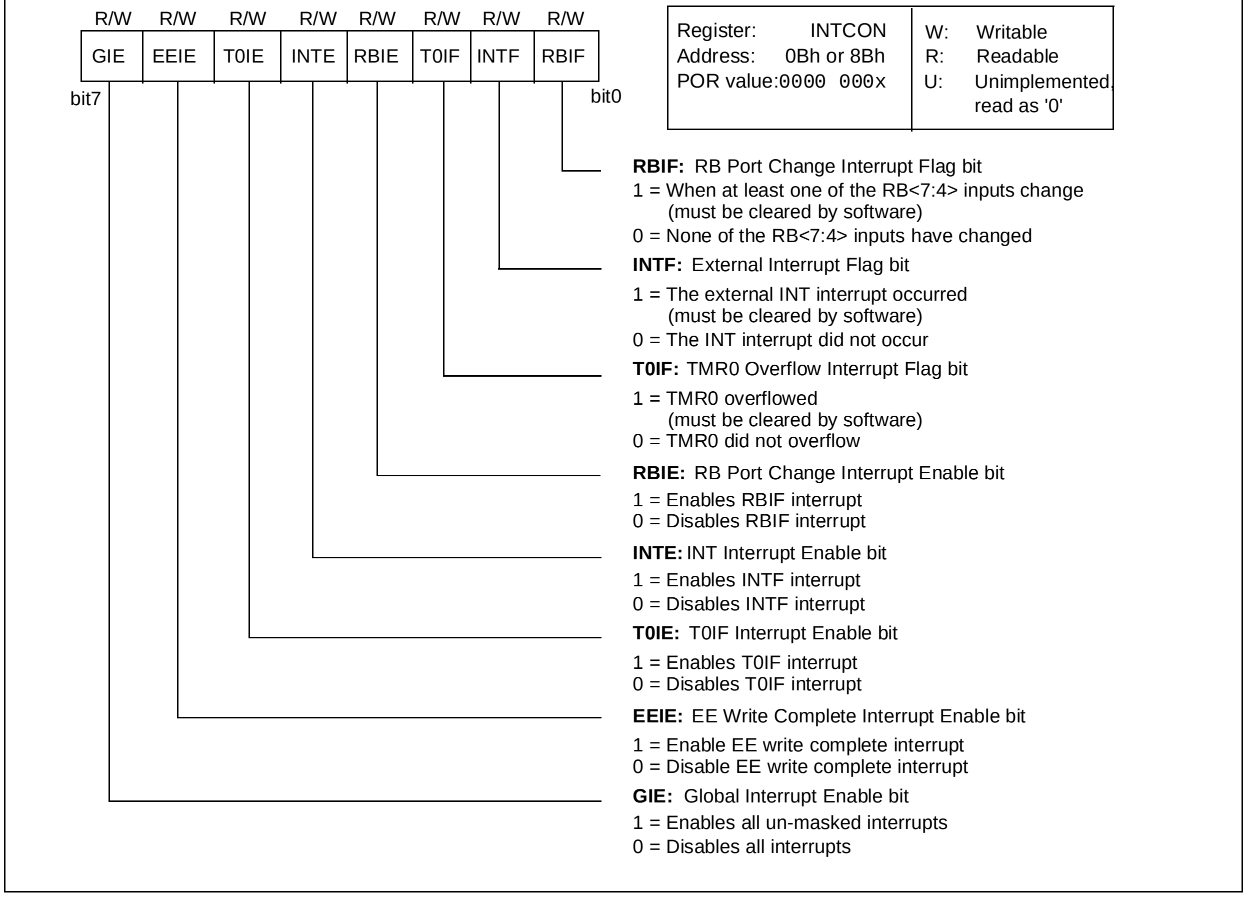
$\overline{\text{RBPU}}$: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled (overriding any port latch value)
0 = PORTB pull-ups are enabled (by individual port-latch values)

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: T0IF, INTF, or RBIF will be set by the specified condition even if the corresponding interrupt enable bit is cleared (interrupt disabled) or the GIE bit is cleared (all interrupts disabled).

FIGURE 4-5: INTCON REGISTER


R/W R/W R/W R/W R/W R/W R/W R/W
GIE EEIE T0IE INTE RBIE T0IF INTF RBIF
bit7
bit0
Register: INTCON
Address: 0Bh or 8Bh
POR value: 0000 000x
W: Writable
R: Readable
U: Unimplemented, read as '0'

RBIF: RB Port Change Interrupt Flag bit
1 = When at least one of the RB<7:4> inputs change
(must be cleared by software)
0 = None of the RB<7:4> inputs have changed

INTF: External Interrupt Flag bit
1 = The external INT interrupt occurred
(must be cleared by software)
0 = The INT interrupt did not occur

T0IF: TMR0 Overflow Interrupt Flag bit
1 = TMR0 overflowed
(must be cleared by software)
0 = TMR0 did not overflow

RBIE: RB Port Change Interrupt Enable bit
1 = Enables RBIF interrupt
0 = Disables RBIF interrupt

INTE: INT Interrupt Enable bit
1 = Enables INTF interrupt
0 = Disables INTF interrupt

T0IE: T0IF Interrupt Enable bit
1 = Enables T0IF interrupt
0 = Disables T0IF interrupt

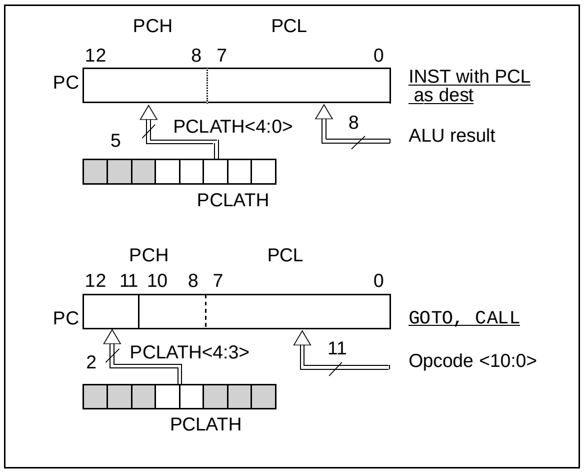
EEIE: EE Write Complete Interrupt Enable bit
1 = Enable EE write complete interrupt
0 = Disable EE write complete interrupt

GIE: Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

4.3 PCL and PCLATH

The program counter (PC) is 13-bits wide. The low byte, PCL, is a readable and writable register. The high byte of the PC (PCH) is not directly readable nor writable. PCLATH (PC latch high) is a holding register for PC<12:8> where contents are transferred to the upper byte of the program counter. When the PC is loaded with a new value during a `CALL, GOTO` or a write to PCL, the high bits of PC are loaded from PCLATH as shown in Figure 4-6.

FIGURE 4-6: LOADING OF PC IN DIFFERENT SITUATIONS



4.3.1 COMPUTED GOTO

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 byte block). Refer to the application note “Table Read Using the PIC16CXX” (AN556).

4.3.2 STACK

The PIC16C84 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 PUSH'ed onto the stack when a `CALL` instruction is executed or an interrupt is acknowledged. The stack is POP'ed in the event of a `RETURN, RETLW` or a `RETFIE` instruction execution. PCLATH is not affected by a PUSH or a POP operation.

The stack operates as a circular buffer. That is, after the stack has been PUSH'ed 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 POP'ed nine times, the PC value is the same as the value from the first POP.

Note 1: There are no status bits to indicate stack overflow or stack underflow conditions.

Note 2: There are no instructions 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

4.3.3 PROGRAM MEMORY PAGING

The PIC16C84 has 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 PIC16C8X 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 4-6). 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. Therefore, manipulation of the PCLATH<4:3> is not required for the return instructions (which POPs the PC from the stack).

Note: The PIC16C84 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 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register and is used in conjunction with the FSR register to perform indirect addressing.

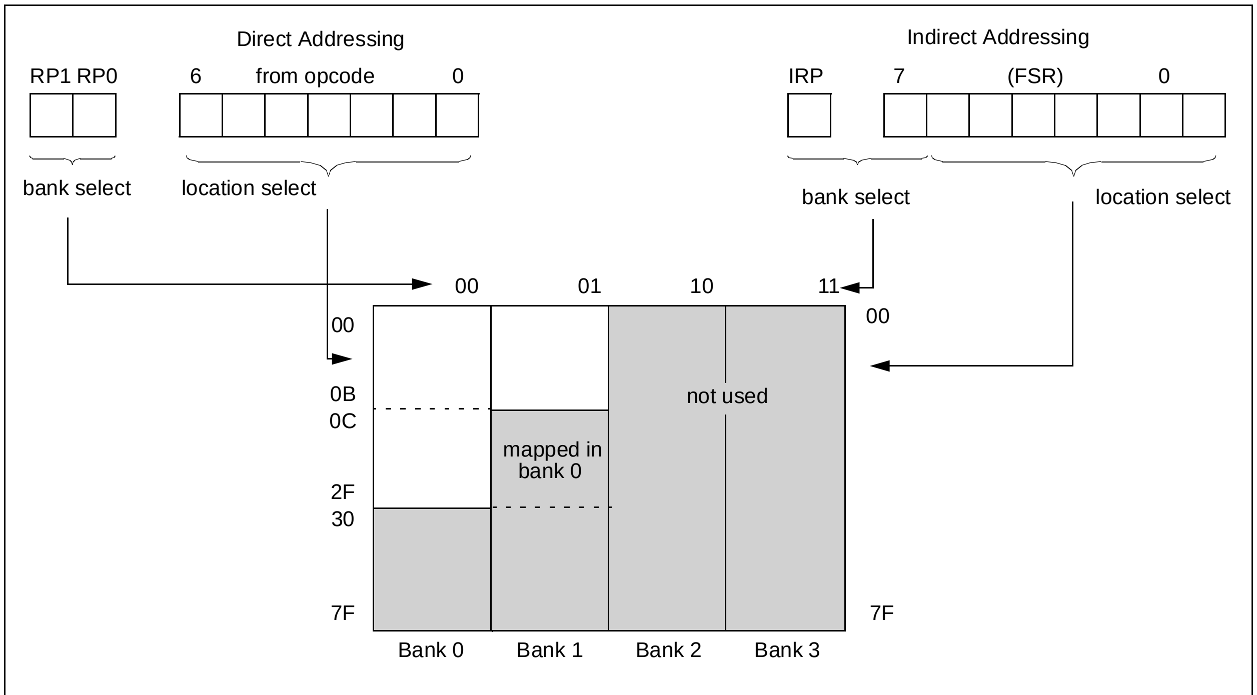
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses data pointed to by the file select register (FSR). Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 4-7. However, IRP is not used in the PIC16C84.

A simple program to clear RAM location 20h-2Fh using indirect addressing is shown in Example 4-1.

EXAMPLE 4-1: INDIRECT ADDRESSING

```
         movlw   0x20    ; initialize pointer
         movf    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
```

FIGURE 4-7: DIRECT/INDIRECT ADDRESSING



Note: For memory map detail, see Figure 4-2.

5.0 I/O PORTS

The PIC16C84 device 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

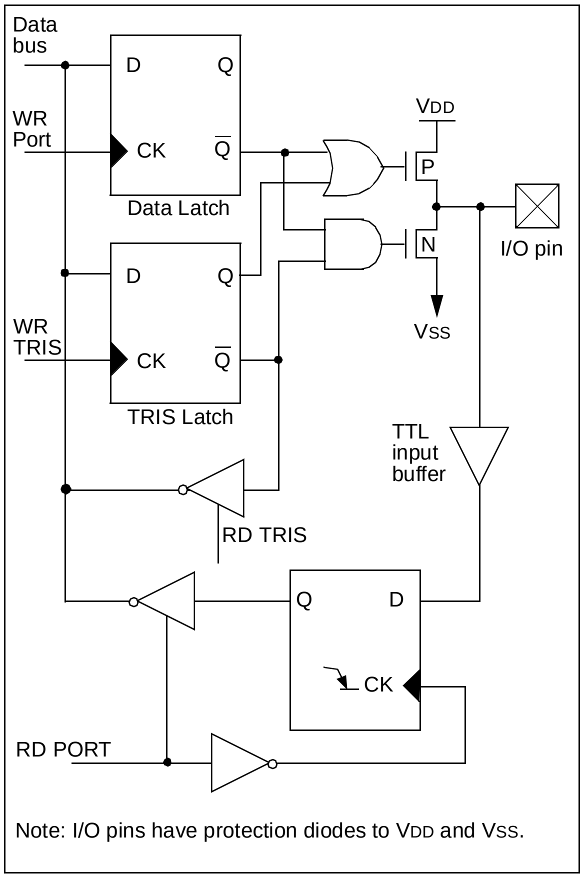
PORTA is a 5-bit wide latch. RA4 is a Schmitt trigger input and an open collector output. All other RA port pins have TTL input levels and full CMOS output drivers. All pins have data direction bits (TRIS registers) which can configure these pins as output or input.

A '1' on any bit in the TRISA register puts the corresponding output driver in a hi-impedance mode. A '0' on any bit in the TRISA register puts the contents of the output latch on the selected pin(s).

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.

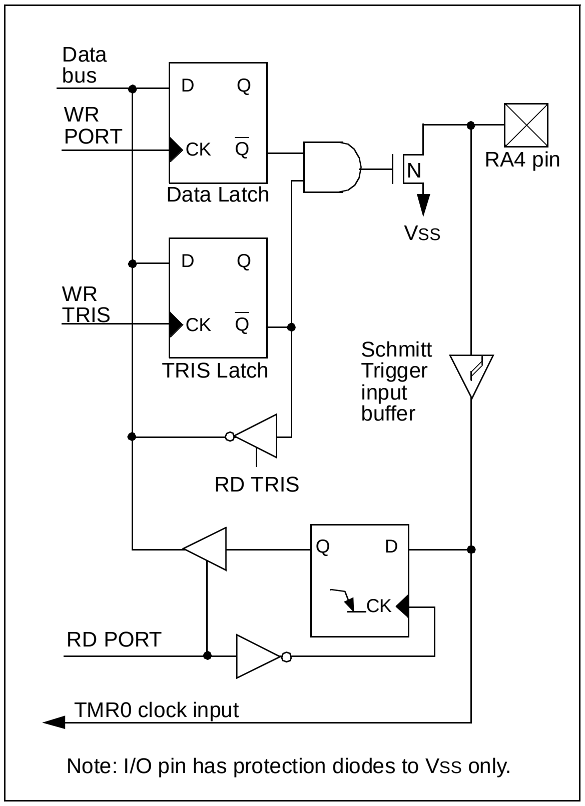
FIGURE 5-1: BLOCK DIAGRAM OF RA<3:0>



EXAMPLE 5-1: INITIALIZING PORTA

```
CLRF   PORTA       ;Initialize PORTA by setting
                   ; output data latches
BSF    STATUS, RP0 ;Select Bank1
MOVLW  0xCF        ;Value used to initialize
                   ;data direction
MOVWF  TRISA       ;Set RA<3:0> as inputs
                   ;RA<5:4> as outputs
                   ;TRISA<7:6> are always
                   ;read as '0'.
```

FIGURE 5-2: BLOCK DIAGRAM OF RA4 PIN



Note:	For crystal oscillator configurations operating below 500 kHz, the device may generate a spurious internal Q-clock when PORTA<0> switches state. This does not occur with an external clock in RC mode. To avoid this, the RA0 pin should be kept static, i.e. in input/output mode, RA0 should not be toggled.

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 collector type.

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

TABLE 5-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Register Name	Function	Address	Power-on Reset Value
PORTA	PORTA pins when read PORTA data latch when written	05h	`---x xxxx`
TRISA	PORTA data direction register 0 = output, 1 = input	85h	`---1 1111`

Legend: x = unknown, - = unimplemented, read as '0'.

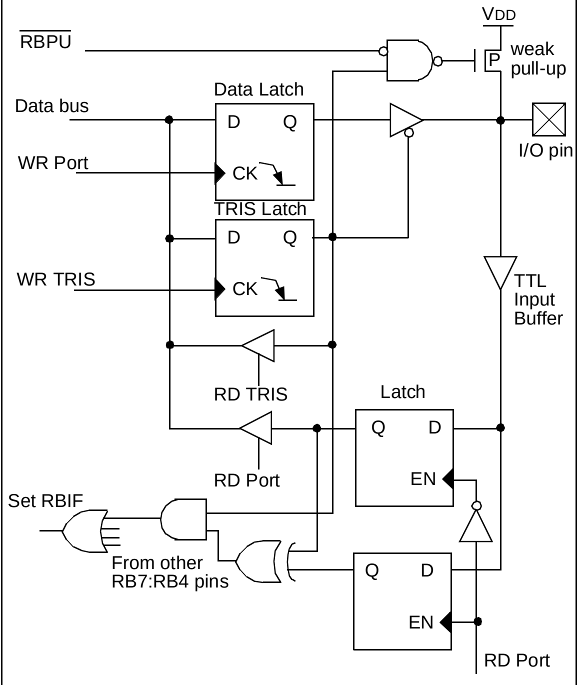
5.2 PORTB and TRISB Registers

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 $\overline{\text{RBPU}}$ (OPTION<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 RBIF interrupt (INTCON<0>).

FIGURE 5-3: BLOCK DIAGRAM OF RB<7:4> PINS



Note 1: TRISB = 1 enables weak pull-up (if $\overline{\text{RBPU}}$ = 0 in the OPTION register).

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

This interrupt can wake the device from SLEEP. The user, in the interrupt service routine, can clear the interrupt in one of two ways:

1. Disable the interrupt by clearing the RBIE (INTCON<3>) bit.
2. Read PORTB, then clear the RBIF bit.

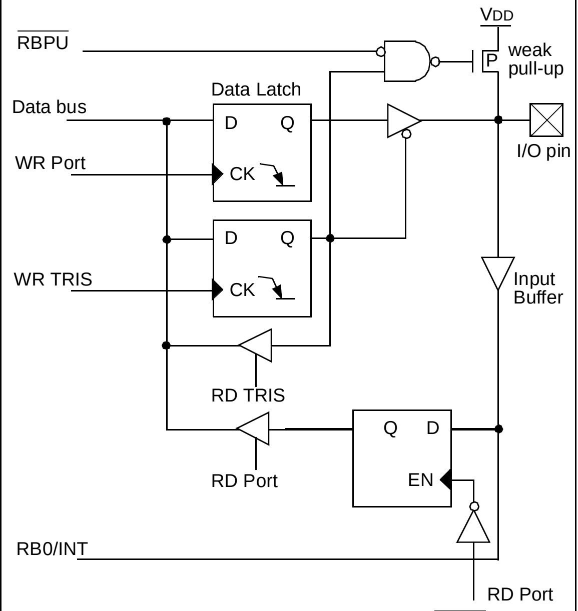
A mismatch condition will continue to set the RBIF bit. Reading PORTB will end the mismatch condition, and allow the RBIF bit to be cleared.

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:	If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), the RBIF interrupt flag may not be set.

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 RB<3:0> PINS



Note 1: TRISB = 1 enables weak pull-up (if $\overline{\text{RBPU}}$ = 0 in the OPTION register).

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

3: For RB0/INT pin, the INT input comes through a Schmitt Trigger input buffer.

EXAMPLE 5-2: INITIALIZING PORTB

```
CLRF  PORTB       ;Initialize PORTB by setting
                  ; output data latches
BSF   STATUS, RP0 ;Select Bank1
MOVLW 0xCF        ;Value used to initialize
                  ;data direction
MOVWF TRISB       ;Set RB<3:0> as inputs
                  ;RB<5:4> as outputs
                  ;TRISB<7:6> are always
                  ;read as '0'.
```

TABLE 5-3: PORTB FUNCTIONS

Name	Bit	Buffer Type	I/O Consistancy Function
RB0/INT	bit0	TTL	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‡	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST‡	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger.
† This buffer is a Schmitt Trigger input when configured as the external interrupt.
‡ This buffer is a Schmitt Trigger input when used in serial programming mode.

TABLE 5-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Register Name	Function	Address	Power-on Reset Value
PORTB	PORTB pins when read PORTB data latch when written	06h	`xxxx xxxx`
TRISB	PORTB data direction register 0 = output, 1 = input	86h	`1111 1111`
OPTION	Weak pull-up on/off control ($\overline{\text{RBPU}}$ bit)	81h	`1111 1111`

Legend: x = unknown.

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

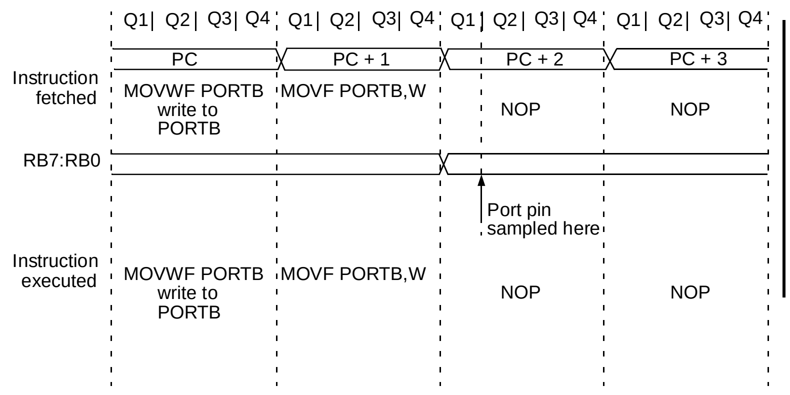
Example 5-3 shows the effect of two sequential read-modify-write instructions (e.g., BCF, BSF, etc.) on an I/O port.

EXAMPLE 5-3: 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 pppp   11pp pppp
    BCF PORTB, 6       ;10pp pppp   11pp pppp
    BCF STATUS, RP0    ;
    BCF TRISB, 7       ;10pp pppp   11pp pppp
    BCF TRISB, 6       ;10pp pppp   10pp pppp
;
;Note that the user may have expected the pin
;values to be 00pp pppp. The 2nd BCF caused
;RB7 to be latched as the pin value (High).
```

FIGURE 5-5: SUCCESSIVE I/O OPERATION



Note:

This example shows a write to PORTB followed by a read from PORTB.

Note that:
data setup time = (0.25 T_{CY} - T_{PD})
where T_{CY} = instruction cycle.
T_{PD} = propagation delay

Therefore, at higher clock frequencies, a write followed by a read may be problematic.

NOTES:

6.0 TIMER0 (TMR0) MODULE

The TMR0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Figure 6-1 is a simplified block diagram of the TMR0 module.

Timer mode is selected by clearing the T0CS bit (OPTION<5>). In timer mode, the TMR0 module will increment every instruction cycle (without prescaler). If TMR0 is written, the increment is inhibited for the following two cycles (Figure 6-2 and Figure 6-3). The user can work around this by writing an adjusted value to the TMR0 module.

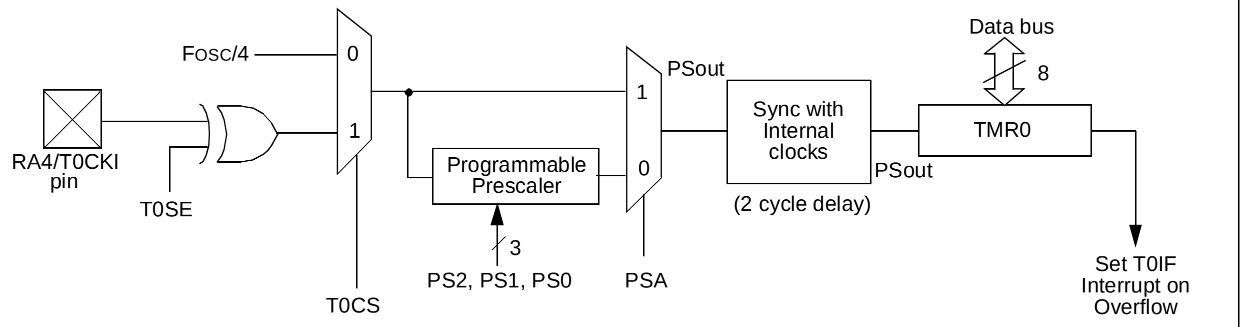
Counter mode is selected by setting the T0CS bit (OPTION<5>). In this mode TMR0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the T0 source edge (T0SE) control bit (OPTION<4>). Clearing the T0SE bit (OPTION<4>) selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 6.2.

The prescaler is shared between the TMR0 module and the Watchdog Timer. The prescaler assignment is controlled, in software, by control bit PSA (OPTION<3>). Clearing the PSA bit will assign the prescaler to TMR0. The prescaler is not readable nor writable. When the prescaler is assigned to the TMR0 module, the prescale value (1:2, 1:4, ..., 1:256) is software selectable. Section 6.3 details the operation of the prescaler.

6.1 TIMER0 (TMR0) Interrupt

TMR0 interrupt is generated when the TMR0 module timer/counter overflows from FFh to 00h. This overflow sets the T0IF bit (INTCON<2>). The interrupt can be masked by clearing the T0IE bit (INTCON<5>). The T0IF bit must be cleared in software by the TMR0 module interrupt service routine before re-enabling this interrupt. The TMR0 module interrupt cannot wake the processor from SLEEP since the timer is shut off during SLEEP. Figure 6-4 shows the TMR0 interrupt timing.

FIGURE 6-1: TMR0 BLOCK DIAGRAM



Note 1: Bits T0SE, T0CS, PS2, PS1, PS0 and PSA are located in the OPTION register.
2: The prescaler is shared with the Watchdog Timer (Figure 6-6)

FIGURE 6-2: TMR0 TIMING: INTERNAL CLOCK/NO PRESCALE

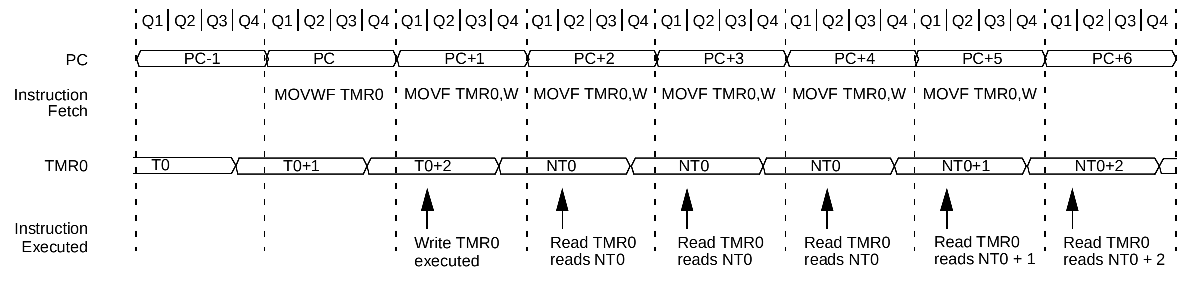


FIGURE 6-3: TMR0 TIMING: INTERNAL CLOCK/PRESCALE 1:2

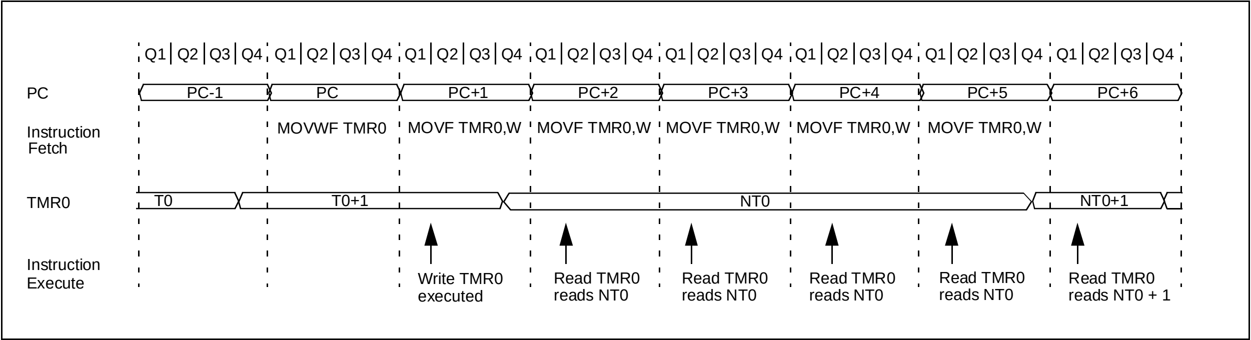
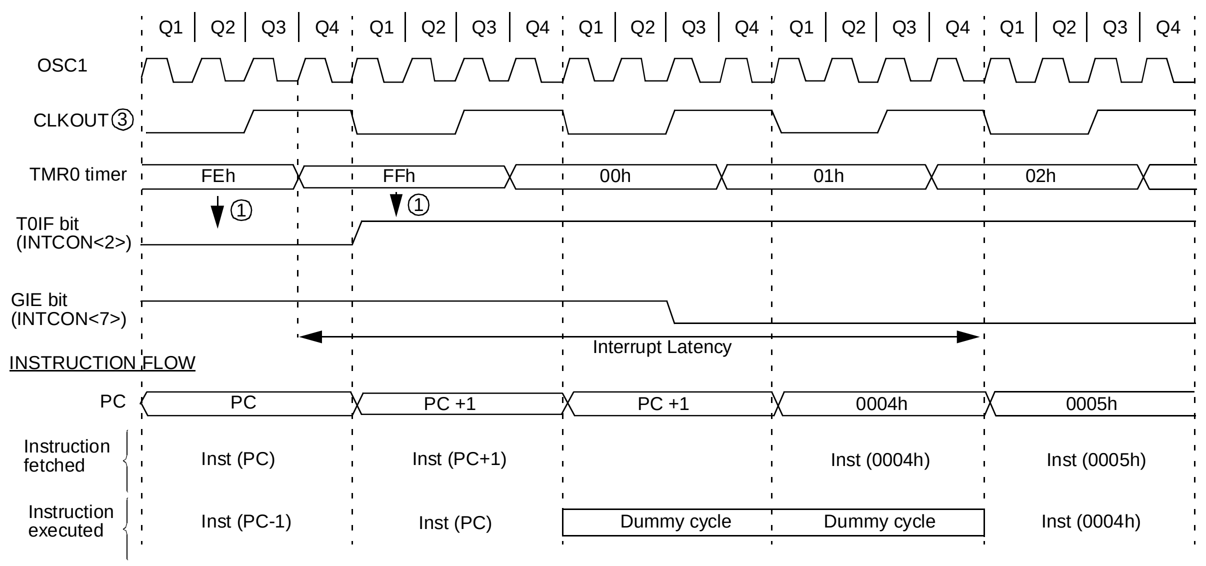


FIGURE 6-4: TMR0 INTERRUPT TIMING



Note 1: T0IF interrupt flag is sampled here (every Q1).
2: Interrupt latency = 4Tcy, where Tcy = instruction cycle time.
3: CLKOUT is available only in RC oscillator mode.

6.2 Using TMR0 with External Clock

When an external clock input is used for TMR0, it must meet certain requirements. The external clock requirement is due to internal phase clock (TOSC) synchronization. Also, there is a delay in the actual incrementing of TMR0 after synchronization.

6.2.1 EXTERNAL CLOCK SYNCHRONIZATION

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 6-5). Therefore, it is necessary for T0CKI to be high for at least 2 Tosc (plus a small RC delay) and low for at least 2 Tosc (plus a small RC delay). Refer to the electrical specification of the desired device.

When a prescaler is used, the external clock input is divided by an asynchronous ripple counter type prescaler so that the prescaler output is symmetrical. For the external clock to meet the sampling requirement, the ripple counter must be taken into account. Therefore, it is necessary for T0CKI to have a period of at least 4 Tosc (plus a small RC delay) divided by the prescaler value. The only requirement on T0CKI high and low time is that they do not violate the minimum pulse width requirement of 10 ns. Refer to parameters 40, 41 and 42 in the AC Electrical Specifications of the desired device.

6.2.2 TMR0 INCREMENT DELAY

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time the TMR0 module is actually incremented. Figure 6-5 shows the delay from the external clock edge to the timer incrementing.

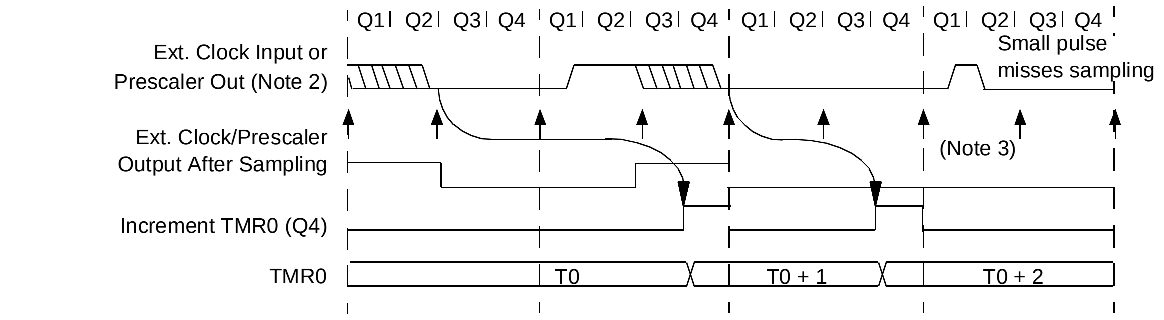
6.3 Prescaler

An 8-bit counter is available as a prescaler for the TMR0 module, or as a post-scaler for the Watchdog Timer (Figure 6-6). For simplicity, this counter is being referred to as “prescaler” throughout this data sheet. Note that there is only one prescaler available which is mutually exclusive between the TMR0 module and the Watchdog Timer. Thus, a prescaler assignment for the TMR0 module means that there is no prescaler for the Watchdog Timer, and vice-versa.

The PSA and PS2:PS0 bits (OPTION<3:0>) determine the prescaler assignment and prescale ratio.

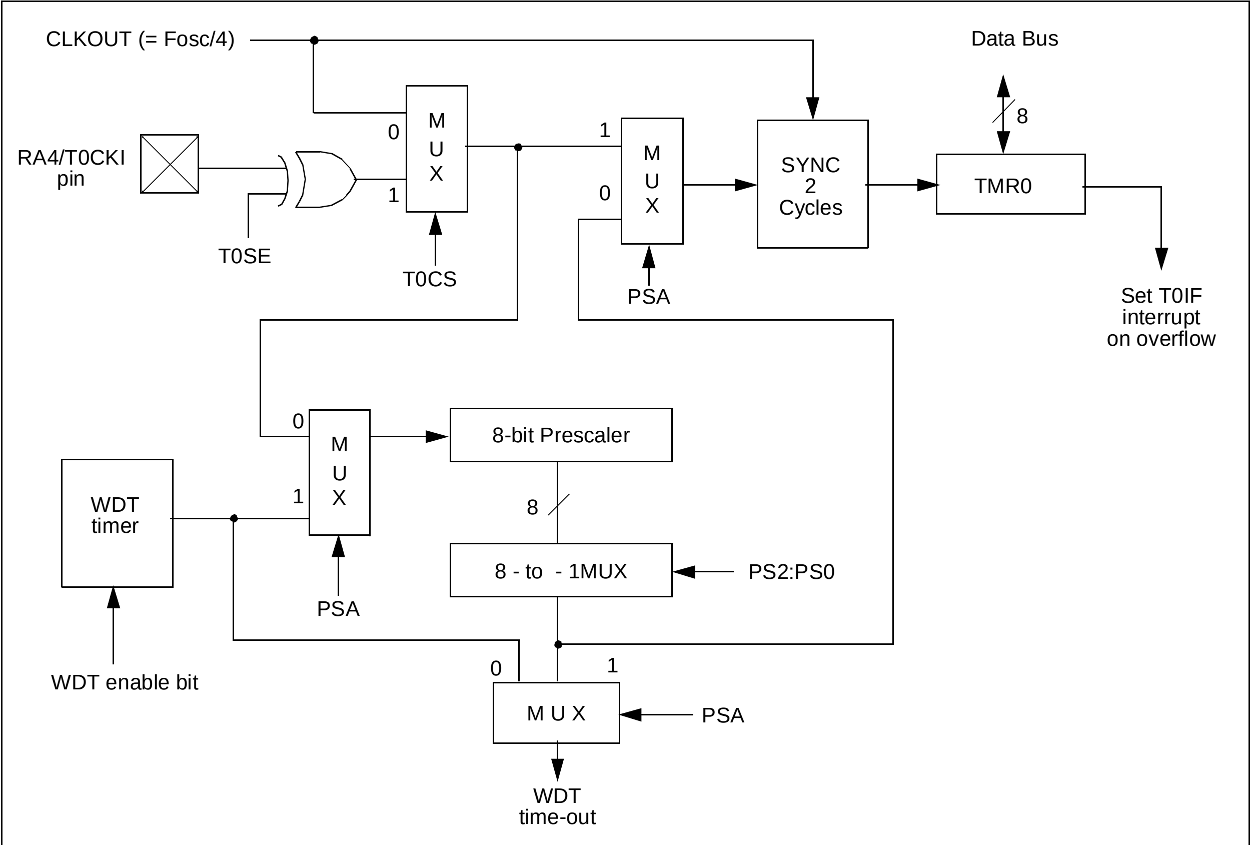
When assigned to the TMR0 module, all instructions writing to the TMR0 module (e.g. `CLRF 1, MOVWF 1, BSF 1,x` ....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable nor writable.

FIGURE 6-5: TIMER0 TIMING WITH EXTERNAL CLOCK


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
Ext. Clock Input or Prescaler Out (Note 2)
Small pulse misses sampling
Ext. Clock/Prescaler Output After Sampling
(Note 3)
Increment TMR0 (Q4)
TMR0
T0
T0 + 1
T0 + 2

Note 1: Delay from clock input change to TMR0 increment is 3 Tosc to 7 Tosc. (Duration of Q = tosc). Therefore, the error in measuring the interval between two edges on TMR0 input = ± 4 Tosc max.
2: External clock if no prescaler selected, Prescaler output otherwise.
3: The arrows ↑ indicate where sampling occurs.

FIGURE 6-6: BLOCK DIAGRAM OF THE TMR0/WDT PRESCALER



Note: T0SE, T0CS, PSA, PS2:PS0 are bits in the OPTION register.

6.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control, i.e., it can be changed “on the fly” during program execution. To avoid an unintended device RESET, the following instruction sequence (Example 6-1) must be executed when changing the prescaler assignment from TMR0 to the WDT.

EXAMPLE 6-1: CHANGING PRESCALER (TMR0→WDT)

```
BCF    STATUS, RP0  ; Bank 0
CLRF   TMR0         ; Clear TMR0 & Prescaler
BSF    STATUS, RP0  ; Bank 1
CLRWDT              ; Clears WDT
MOVLW  'xxxx1xxx'b  ; Select new prescaler
MOVWF  OPTION       ; value
BCF    STATUS, RP0  ; Bank 0
```

To change prescaler from the WDT to the TMR0 module use the sequence shown in Example 6-2. This sequence must be taken even if the WDT is disabled.

EXAMPLE 6-2: CHANGING PRESCALER (WDT→TMR0)

```
CLRWDT                ; Clear WDT and
                      ;   prescaler
BSF     STATUS, RP0   ;
MOVLW   'xxxx0xxx'b   ; Select TMR0, new
                      ;   prescale value
                      ;   and clock source
MOVWF   OPTION        ;
BCF     STATUS, RP0   ;
```

TABLE 6-1: SUMMARY OF TMR0 REGISTERS

Register Name	Function	Address	Power-on Reset Value
TMR0	Timer/counter register	01h	xxxx xxxx
OPTION	Configuration and prescaler assignment bits for TMR0. (Figure 6-5)	81h	1111 1111
INTCON	TMR0 overflow interrupt flag and mask bits. (Figure 6-6)	0Bh	0000 000x

Legend: x = unknown.
Note: For reset values of registers in other reset situations refer to the Special Features of the CPU section.

TABLE 6-2: REGISTERS ASSOCIATED WITH TMR0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01h	TMR0	Timer0							
0Bh/8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
81h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
85h	TRISA	—	—	—	TRISA 4	TRISA 3	TRISA 2	TRISA 1	TRISA 0

Legend: — = Unimplemented locations, read as '0'.
Note 1: Shaded cells are not used by TMR0 module.
2: The PIC16C84 device does not have an RA5 pin.

NOTES:

7.0 DATA EEPROM MEMORY

The EEPROM data memory is readable and writable during normal operation (full VDD range). This memory is not directly mapped in the register file space. Instead it is indirectly addressed through the Special Function Registers. There are four SFRs used to read and write this memory. These registers are:

- EECON1
- EECON2
- EEDATA
- EEADR

EEDATA holds the 8-bit data for read/write, and EEADR holds the address of the EEPROM location being accessed. PIC16C84 devices have 64 bytes of data EEPROM with an address range from 0h to 3Fh.

The EEPROM data memory allows byte read and write. A byte write automatically erases the location and writes the new data (erase before write). The EEPROM data memory is rated for high erase/write cycles. The write time is nominally 10 ms, and is controlled by an on-chip timer. The actual write-time will vary with voltage and temperature as well as from chip to chip. Please refer to AC specifications for exact limits.

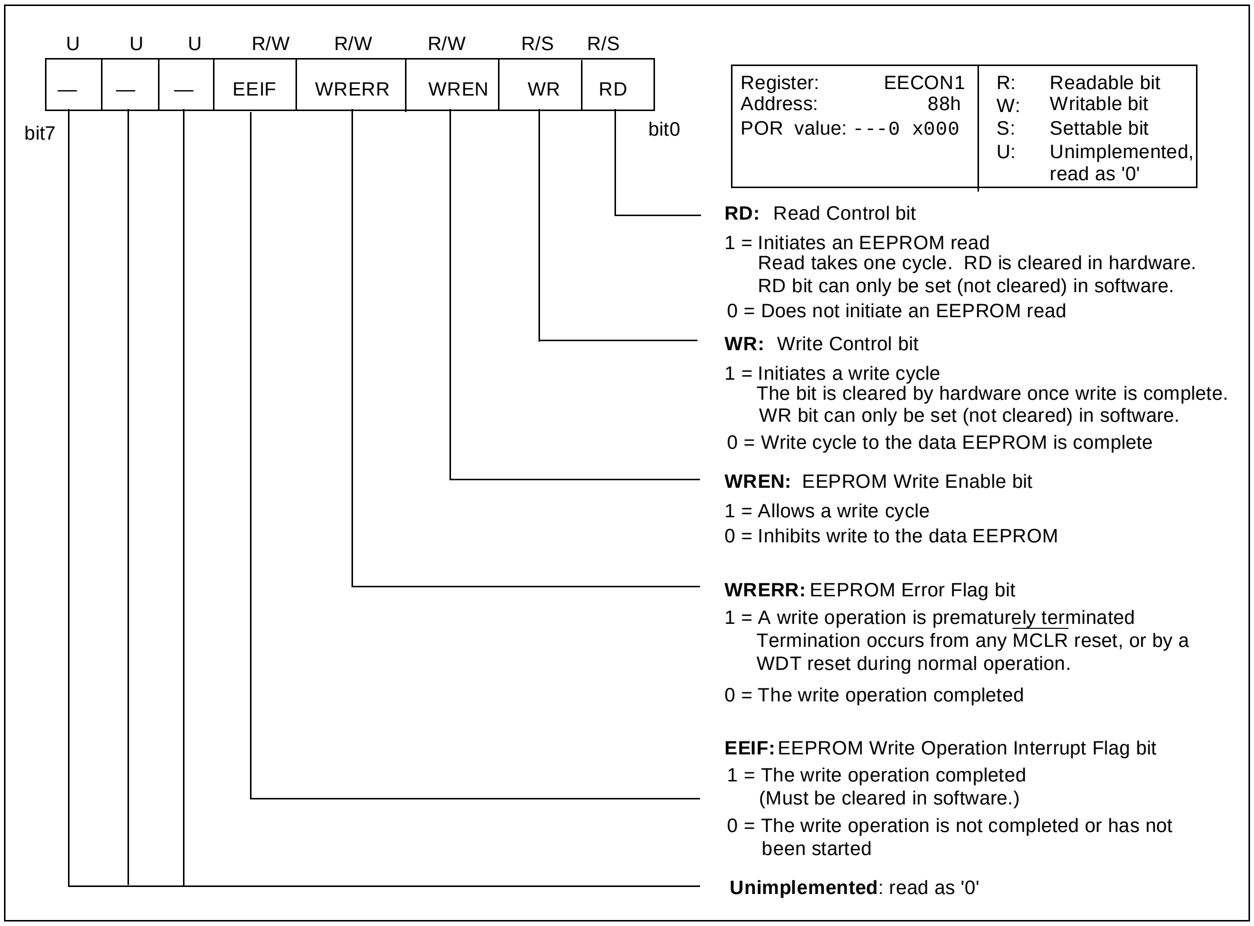
When the device is code protected, only the CPU may complete reads or writes of the data memory. That is, the device programmer can no longer access this memory (external reads/writes are disabled).

7.1 EEADR

The EEADR register can address up to a maximum of 256 bytes of data EEPROM. Only the first 64 bytes of data EEPROM are implemented and only six of the eight bits in the register (EEADR<5:0>) are required.

The upper two bits are not address decoded. This allows four addresses to map into the 64 byte memory space. We recommend using the absolute address (0 - 3Fh) to ensure future upward compatibility.

FIGURE 7-1: EECON1 REGISTER



	U	U	U	R/W	R/W	R/W	R/S	R/S	
bit7	—	—	—	EEIF	WRERR	WREN	WR	RD	bit0

Register: EECON1	R: Readable bit
Address: 88h	W: Writable bit
POR value: `---0 x000`	S: Settable bit
	U: Unimplemented, read as '0'

RD: Read Control bit
1 = Initiates an EEPROM read
Read takes one cycle. RD is cleared in hardware.
RD bit can only be set (not cleared) in software.
0 = Does not initiate an EEPROM read

WR: Write Control bit
1 = Initiates a write cycle
The bit is cleared by hardware once write is complete.
WR bit can only be set (not cleared) in software.
0 = Write cycle to the data EEPROM is complete

WREN: EEPROM Write Enable bit
1 = Allows a write cycle
0 = Inhibits write to the data EEPROM

WRERR: EEPROM Error Flag bit
1 = A write operation is prematurely terminated
Termination occurs from any MCLR reset, or by a WDT reset during normal operation.
0 = The write operation completed

EEIF: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed
(Must be cleared in software.)
0 = The write operation is not completed or has not been started

Unimplemented: read as '0'

7.2 EECON1 and EECON2 Registers

EECON1 is the control register with five low order bits physically implemented. The upper-three bits are non-existent and read as '0's.

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

The WREN bit, when set, will allow a write operation. On power-up WREN is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ reset or a WDT time-out reset during normal operation. In these situations, following reset, the user can check the WRERR bit and rewrite the location. The data and address will be unchanged in the EEDATA and EEADR registers.

The EEIF interrupt flag bit is set when write is complete. It must be cleared in software.

EECON2 is not a physical register. Reading EECON2 will read all '0's.

7.3 Reading the EEPROM Data Memory

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

EXAMPLE 7-1: DATA EEPROM READ

```
    BCF      STATUS, RP0  ; Bank 0
    MOVLW    CONFIG_ADDR  ;
    MOVWF    EEADR        ; Address to read
    BSF      STATUS, RP0  ; Bank 1
    BSF      EECON, RD    ; EE Read
    BCF      STATUS, RP0  ; Bank 0
    MOVF     EEDATA, W    ; W = EEDATA
```

7.4 Writing to the EEPROM Data Memory

To write an EEPROM data location, the user must first write the address to the EEADR register and the data to the EEDATA register. Then the user must follow a specific sequence to initiate write.

EXAMPLE 7-2: DATA EEPROM WRITE

```
    BSF      STATUS, RP0  ; Bank 1
    BCF      INTCON, GIE  ; Disable INTs.
    MOVLW    55h          ;
    MOVWF    EECON2       ; Write 55h
    MOVLW    AAh          ;
    MOVWF    EECON2       ; Write AAh
    BSF      EECON1,WR    ; Set WR bit
                          ;   begin write
    BSF      INTCON, GIE  ; Enable INTs.
```

Write will not initiate if this sequence (write 55h to EECON2, write AAh to EECON2, then set WR bit) is not followed with exact timing. We strongly recommend that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable write. This mechanism prevents accidental writes to data EEPROM due to errant (unexpected) code execution (i.e., lost programs). The user should keep the WREN bit clear at all times, except when updating EEPROM.

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. The WR bit will be inhibited from being set until the WREN bit has been set.

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

Note: The data EEPROM memory E/W cycle time may occasionally exceed the 10 ms specification (typical). To ensure that the write cycle is complete, use the EE interrupt or poll the WR bit (EECON1<1>). Both these events signify the completion of the write cycle.

7.5 Protection Against Spurious Write

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

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

7.6 Power Consumption Considerations

.

Note:	It is recommended that the EEADR<7:6> bits be cleared. When either of these bits is set, the maximum IDD for the device is higher than when both are cleared. The specification is 400 μA. With EEADR<7:6> cleared, the maximum is approximately 150 μA.

TABLE 7-1: REGISTERS/BITS ASSOCIATED WITH DATA EEPROM

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 (Note1)
08h	EEDATA	EEPROM data register								`xxxx xxxx`	`uuuu uuuu`
09h	EEADR	EEPROM address register								`xxxx xxxx`	`uuuu uuuu`
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	`---0 x000`	`---0 ?000`
89h	EECON2	EEPROM control register 2								`---- ----`	`---- ----`

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', ? = Value depends upon condition.

NOTES:

8.0 SPECIAL FEATURES OF THE CPU

What sets a microcontroller apart from other processors are special circuits to deal with the needs of real time applications. The PIC16C84 has a host of such features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection. These features are:

- OSC selection
- Reset
  - Power-On Reset (POR)
  - Power-Up Timer (PWRT)
  - Oscillator Start-Up Timer (OST)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code protection
- ID locations
- In-circuit serial programming

The PIC16C84 has a Watchdog Timer which can be shut off only through configuration bits. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-Up Timer (OST), intended to keep the chip in reset until the crystal oscillator is stable. The other is the Power-Up Timer (PWRT), which provides a fixed delay of 72 ms (nominal) on power-up only. This design keeps the device in reset while the power supply stabilizes. With these two timers on-chip, most applications need no external reset circuitry.

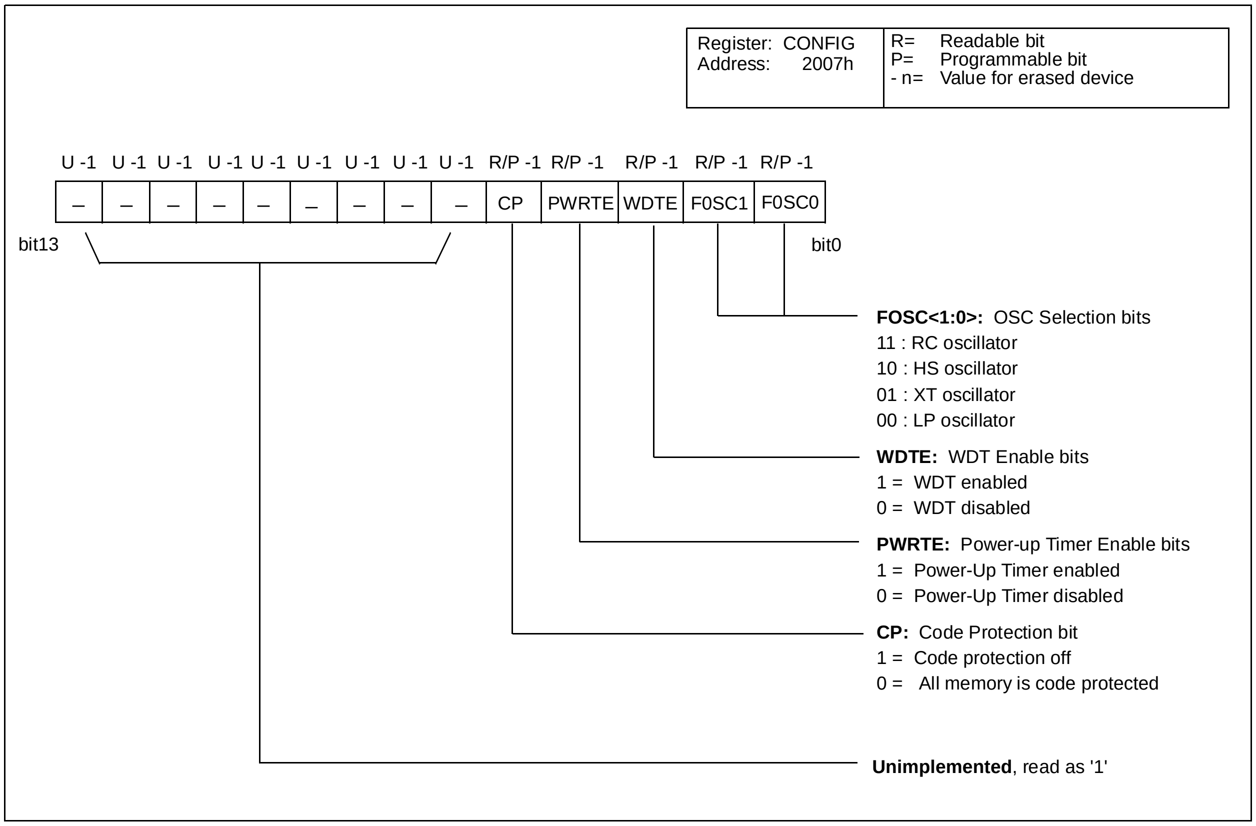
SLEEP mode offers a very low current power-down mode. The user can wake up from SLEEP through external reset, Watchdog Timer time-out or through an interrupt. Several oscillator options are provided to allow the part to fit the application. The RC oscillator option saves system cost while the LP crystal option saves power. A set of configuration bits are used to select the various options.

8.1 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 in program memory location 2007h.

Address 2007h is beyond the user program memory space and it belongs to the special test/configuration memory space (2000h - 3FFFh). This space can only be accessed during programming.

FIGURE 8-1: CONFIGURATION WORD



Register: CONFIG
Address: 2007h
R= Readable bit
P= Programmable bit
- n= Value for erased device
U -1 U -1 U -1 U -1 U -1 U -1 U -1 U -1 U -1 R/P -1 R/P -1 R/P -1 R/P -1 R/P -1
– – – – – – – – – CP PWRTE WDTE F0SC1 F0SC0
bit13
bit0
FOSC<1:0>: OSC Selection bits
11 : RC oscillator
10 : HS oscillator
01 : XT oscillator
00 : LP oscillator
WDTE: WDT Enable bits
1 = WDT enabled
0 = WDT disabled
PWRTE: Power-up Timer Enable bits
1 = Power-Up Timer enabled
0 = Power-Up Timer disabled
CP: Code Protection bit
1 = Code protection off
0 = All memory is code protected
Unimplemented, read as '1'

8.2 Oscillator Configurations

8.2.1 OSCILLATOR TYPES

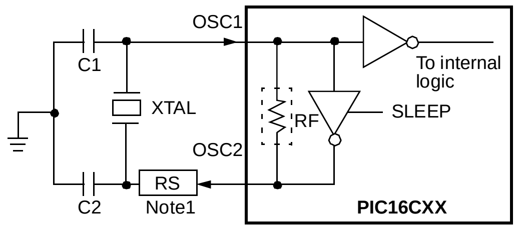
The PIC16C84 can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:

• LP	Low Power Crystal
• XT	Crystal/Resonator
• HS	High Speed Crystal/Resonator
• RC	Resistor/Capacitor

8.2.2 CRYSTAL OSCILLATOR / CERAMIC RESONATORS

In XT, LP or HS modes a crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins to establish oscillation (Figure 8-2). The PIC16C84 oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications. When in XT, LP or HS modes, the device can have an external clock source to drive the OSC1/CLKIN pin. This is shown in Figure 8-3.

FIGURE 8-2: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



See Table 8-1 and Table 8-2 for recommended values of C1 and C2.

Note 1: A series resistor may be required for AT strip cut crystals.

FIGURE 8-3: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)

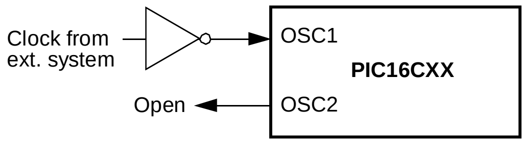


TABLE 8-1: PIC16C84 CAPACITOR SELECTION FOR CERAMIC RESONATORS

Ranges Tested:

Mode	Freq	OSC1/C1	OSC2/C2
XT	455 kHz	47 - 100 pF	47 - 100 pF
	2.0 MHz	15 - 68 pF	15 - 68 pF
	4.0 MHz	15 - 68 pF	15 - 68 pF
HS	8.0 MHz	15 - 68 pF	15 - 68 pF
	10.0 MHz	10 - 47 pF	10 - 47 pF

Note: Recommended values of C1 and C2 are identical to the ranges tested table.

Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for the appropriate values of external components.

Resonators Tested:

455 kHz	Panasonic EFO-A455K04B	± 0.3%
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%
10.0 MHz	Murata Erie CSA10.00MTZ	± 0.5%

None of the resonators had built-in capacitors.

TABLE 8-2: PIC16C84 CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz	68 - 100 pF	68 - 100 pF
	200 kHz	15 - 30 pF	15 - 30 pF
XT	100 kHz	68 - 150 pF	150 - 200 pF
	2 MHz	15 - 33 pF	15 - 33 pF
	4 MHz	15 - 33 pF	15 - 33 pF
HS	4 MHz	15 - 33 pF	15 - 33 pF
	10 MHz	15 - 47 pF	15 - 47 pF

Note: Higher capacitance increases the stability of oscillator but also increases the start-up time. These values are for design guidance only. Rs may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

For $V_{DD} > 4.5V$, $C1 = C2 \approx 30$ pF is recommended.

Crystals Tested:

32.768 kHz	Epson C-001R32.768K-A	± 20 PPM
100 kHz	Epson C-2 100.00 KC-P	± 20 PPM
200 kHz	STD XTL 200.000 KHz	± 20 PPM
1.0 MHz	ECS ECS-10-13-2	± 50 PPM
2.0 MHz	ECS ECS-20-S-2	± 50 PPM
4.0 MHz	ECS ECS-40-S-4	± 50 PPM
10.0 MHz	ECS ECS-100-S-4	± 50 PPM

8.2.3 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator can be used or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator will provide good performance with TTL gates. Two types of crystal oscillator circuits are available; one with series resonance, or one with parallel resonance.

Figure 8-4 shows a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180-degree phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides negative feedback for stability. The 10 kΩ potentiometer biases the 74AS04 in the linear region. This could be used for external oscillator designs.

FIGURE 8-4: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT

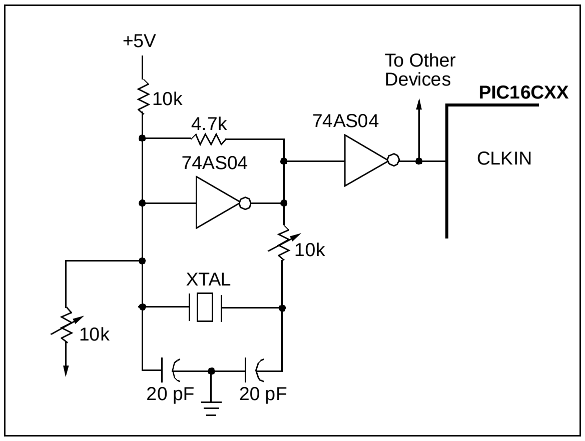
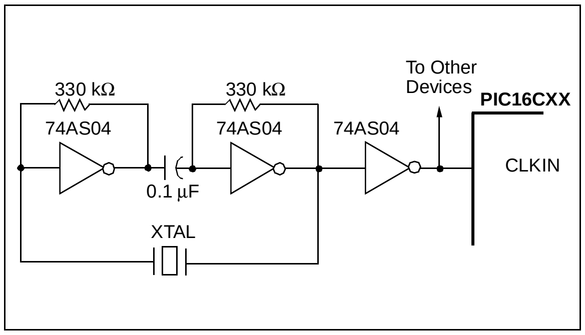


Figure 8-5 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift. The 330 kΩ resistors provide the negative feedback to bias the inverters in their linear region.

FIGURE 8-5: EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT



8.2.4 RC OSCILLATOR

For timing insensitive applications the RC device option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (Rext) values, capacitor (Cext) values, and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types also affects the oscillation frequency, especially for low Cext values. The user needs to take into account variation due to tolerance of the external R and C components. Figure 8-6 shows how an R/C combination is connected to the PIC16C84. For Rext values below 2.2 kΩ, the oscillator operation may become unstable, or stop completely. For very high Rext values (e.g. 1 MΩ), the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend keeping Rext between 3 kΩ and 100 kΩ.

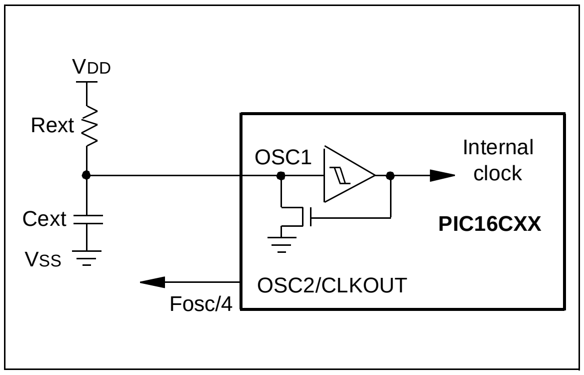
Although the oscillator will operate with no external capacitor (Cext = 0 pF), we recommend using values above 20 pF for noise and stability reasons. With little or no external capacitance, the oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance or package lead frame capacitance.

See the electrical specification section for RC frequency variation from part to part due to normal process variation. The variation is larger for larger R (since leakage current variation will affect RC frequency more for large R) and for smaller C (since variation of input capacitance has a greater affect on RC frequency).

See the electrical specification section for variation of oscillator frequency due to VDD for given Rext/Cext values as well as frequency variation due to operating temperature.

The oscillator frequency, divided by 4, is available on the OSC2/CLKOUT pin, and can be used for test purposes or to synchronize other logic (see Figure 3-2 for waveform).

FIGURE 8-6: RC OSCILLATOR MODE



8.3 Reset

The PIC16C84 differentiates between various kinds of reset:

- Power-On Reset (POR)
- $\overline{\text{MCLR}}$ reset during normal operation
- $\overline{\text{MCLR}}$ reset during SLEEP
- WDT time-out reset during normal operation
- WDT time-out reset during SLEEP

Some registers are not affected in any reset condition; their status is unknown on POR reset and unchanged in any other reset. Most other registers are reset to a "reset state" on POR, $\overline{\text{MCLR}}$ or WDT reset during normal operation and on $\overline{\text{MCLR}}$ reset during SLEEP. They are not affected by a WDT reset during SLEEP, since this reset is viewed as the resumption of normal operation. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are set or cleared differently in different reset situations as indicated in Table 8-4. These bits are used in software to determine the nature of the reset. Table 8-6 gives a full description of reset states for all registers.

Figure 8-7 shows a simplified block diagram of the on-chip reset circuit.

8.4 Power-On Reset (POR), Power-Up-Timer (PWRT) and Oscillator Start-Up Timer (OST)

8.4.1 POWER-ON RESET (POR)

A Power-On Reset pulse is generated on-chip when VDD rise is detected (in the range of 1.6V - 1.8V). To take advantage of the POR, just tie the $\overline{\text{MCLR}}$ pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create Power-On Reset. A maximum rise time for VDD is required for this to operate properly. See Electrical Specifications for details.

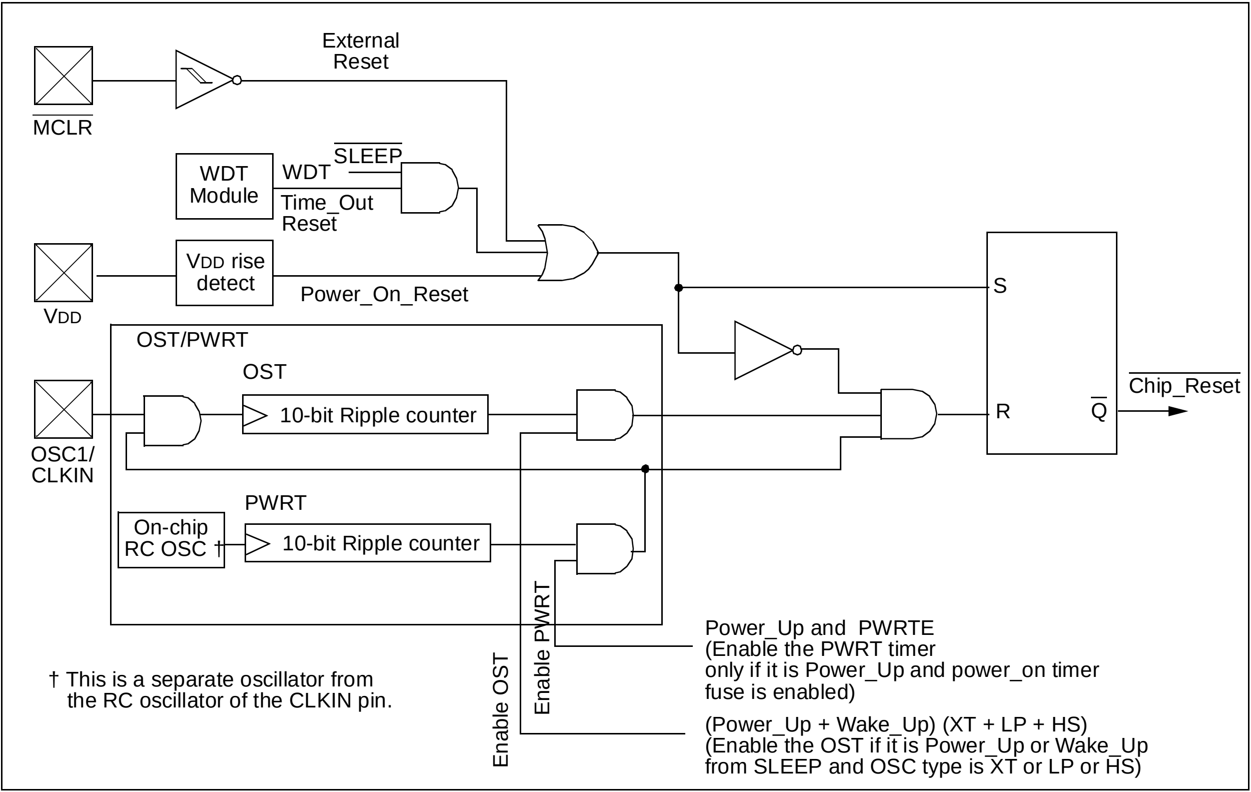
The POR circuit does not produce an internal reset when VDD declines.

8.4.2 POWER-UP TIMER (PWRT)

The Power-Up Timer provides a fixed 72 ms nominal time-out on power-up only, from POR. The power-up timer operates on an internal RC oscillator. The chip is kept in reset as long as PWRT is active. The PWRT delay allows the VDD to rise to an acceptable level. A configuration fuse, PWRTE, can enable (if set) or disable (if cleared or programmed) the power-up timer.

The Power-Up Time delay will vary from chip to chip due to VDD, temperature, and process variation. See DC parameters for details.

FIGURE 8-7: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT


External Reset
MCLR
WDT Module
WDT
SLEEP
Time_Out Reset
VDD rise detect
VDD
Power_On_Reset
OST/PWRT
OST
10-bit Ripple counter
OSC1/ CLKIN
PWRT
On-chip RC OSC †
10-bit Ripple counter
S
R
Q
Chip_Reset
Enable PWRT
Enable OST
Power_Up and PWRTE (Enable the PWRT timer only if it is Power_Up and power_on timer fuse is enabled)
(Power_Up + Wake_Up) (XT + LP + HS) (Enable the OST if it is Power_Up or Wake_Up from SLEEP and OSC type is XT or LP or HS)
† This is a separate oscillator from the RC oscillator of the CLKIN pin.

8.4.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-Up Timer (OST) provides a 1024 oscillator cycle delay (from OSC1 input) after the PWRT delay ends. This ensures the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP and HS modes and only on power-on reset or wake-up from SLEEP.

8.4.4 TIME-OUT SEQUENCE

On power-up the time-out sequence is as follows: First PWRT time-out is invoked after a POR has expired. Then the OST is activated. The total time-out will vary based on oscillator configuration and PWRTE fuse status. For example, in RC mode with the PWRTE fuse cleared (PWRT disabled), there will be no time-out at all. Figure 8-8, Figure 8-9, and Figure 8-10 depict time-out sequences on power-up.

TABLE 8-3: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up		Wake up from SLEEP
	PWRTE = 1	PWRTE = 0	
XT, HS, LP	72 ms + 1024 Tosc	1024 Tosc	1024 tosc
RC	72 ms	—	—

Since the time-outs occur from the POR reset pulse, if $\overline{\text{MCLR}}$ is kept low long enough, the time-outs will expire. Then bringing $\overline{\text{MCLR}}$ high will begin immediately (see Figure 8-9). This is useful for testing purposes or to synchronize more than one PIC16CXX device when operating in parallel.

Table 8-4 shows the significance of the $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits. Table 8-5 lists the reset conditions for some special registers, while Table 8-6 lists the reset conditions for all the registers.

TABLE 8-4: STATUS BITS AND THEIR SIGNIFICANCE

$\overline{\text{TO}}$	$\overline{\text{PD}}$	Condition
1	1	Power-On Reset
0	x	Illegal, $\overline{\text{TO}}$ is set on $\overline{\text{POR}}$
x	0	Illegal, $\overline{\text{PD}}$ is set on $\overline{\text{POR}}$
0	1	WDT reset during normal operation
0	0	WDT timeout wakeup from SLEEP
1	1	$\overline{\text{MCLR}}$ reset during normal operation
1	0	$\overline{\text{MCLR}}$ reset during SLEEP or interrupt wake-up from SLEEP

TABLE 8-5: RESET CONDITION FOR PCL AND STATUS REGISTERS

Condition	PCL Addr: 02h	STATUS Addr: 03h/83h
Power-On Reset	000h	`0001 1xxx`
$\overline{\text{MCLR}}$ reset during normal operation	000h	`0001 1uuu`
$\overline{\text{MCLR}}$ reset during SLEEP	000h	`0001 0uuu`
WDT reset during normal operation	000h	`0000 1uuu`
WDT during SLEEP	PC + 1	`uuu0 0uuu`
Interrupt wake-up from SLEEP	PC + 1 (Note1)	`uuu1 0uuu`

Legend: u = unchanged, x = unknown.
Note 1: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

TABLE 8-6: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Address	Power-On Reset	$\overline{\text{MCLR}}$ Reset during: – normal operation – SLEEP WDT timeout during normal operation	Wake up from SLEEP: – through interrupt – through WDT timeout
W	—	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INDF	00h	`---- ----`	`---- ----`	`---- ----`
TMR0	01h	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PCL	02h	`0000h`	`0000h`	`PC + 1`[2]
STATUS	03h	`0001 1xxx`	`000? ?uuu`[3]	`uuu? ?uuu`[3]
FSR	04h	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PORTA	05h	`---u uuuu`	`---u uuuu`	`---u uuuu`
PORTB	06h	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
EEDATA	08h	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
EEADR	09h	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PCLATH	0Ah	`---0 0000`	`---0 0000`	`---u uuuu`
INTCON	0Bh	`0000 000x`	`0000 000u`	`uuuu uuuu`[1]
INDF	80h	`---- ----`	`---- ----`	`---- ----`
OPTION	81h	`1111 1111`	`1111 1111`	`uuuu uuuu`
PCL	82h	`0000h`	`0000h`	`PC + 1`
STATUS	83h	`0001 1xxx`	`000? ?uuu`[3]	`uuu? ?uuu`[3]
FSR	84h	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
TRISA	85h	`---1 1111`	`---1 1111`	`---u uuuu`
TRISB	86h	`1111 1111`	`1111 1111`	`uuuu uuuu`
EECON1	88h	`---0 0000`	`---0 ?000`	`---0 ?uuu`
EECON2	89h	`---- ----`	`---- ----`	`---- ----`
PCLATH	8Ah	`---0 0000`	`---0 0000`	`---u uuuu`
INTCON	8Bh	`0000 000x`	`0000 000u`	`uuuu uuuu`[1]

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

Note 1: One or more bits in INTCON will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

3: Table 8-5 lists the reset value for each specific condition.

FIGURE 8-8: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): CASE 1

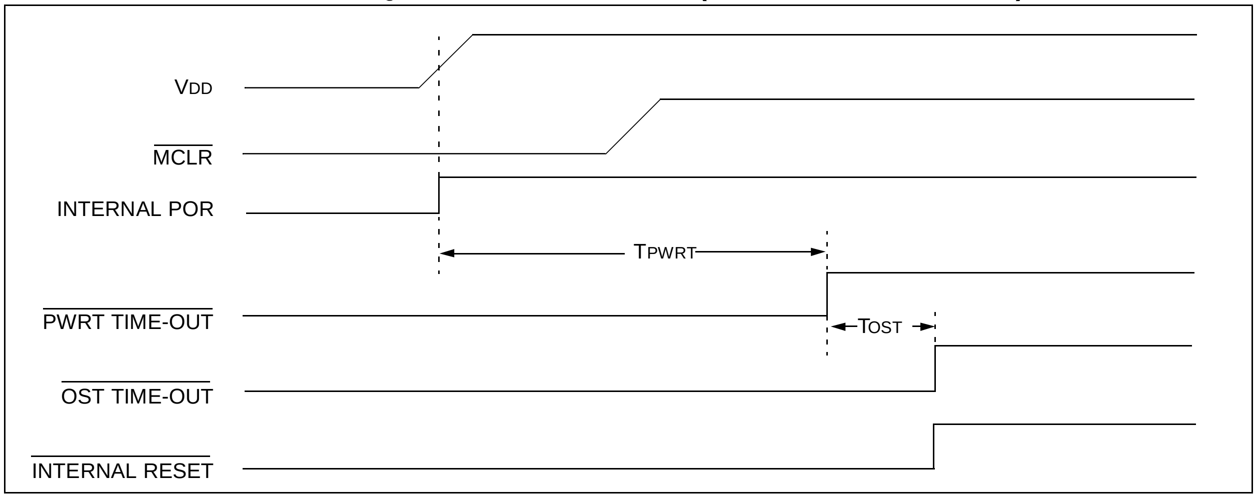


FIGURE 8-9: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): CASE 2

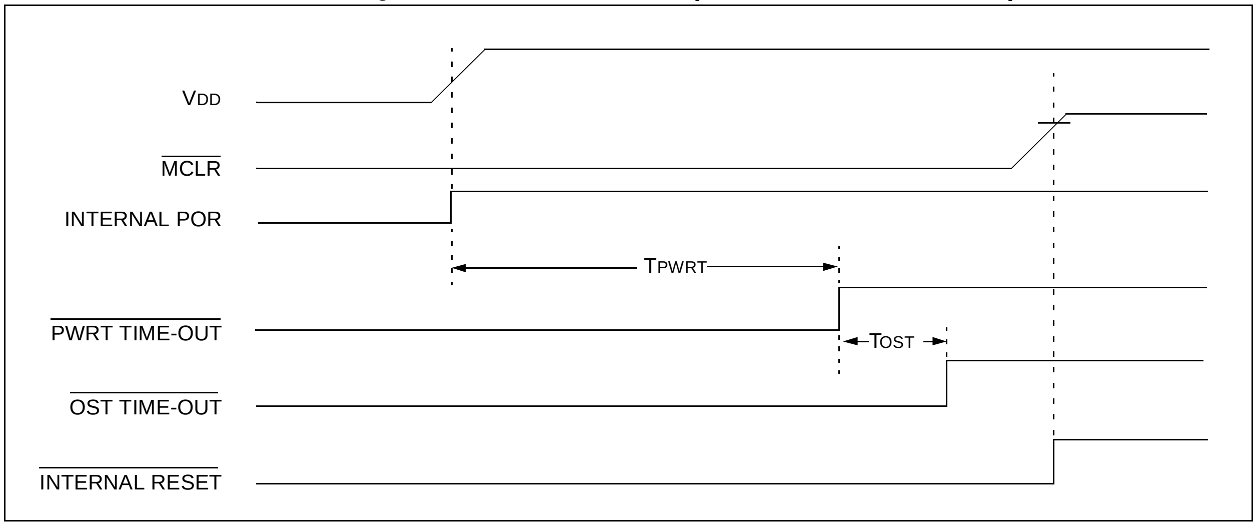


FIGURE 8-10: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO VDD)

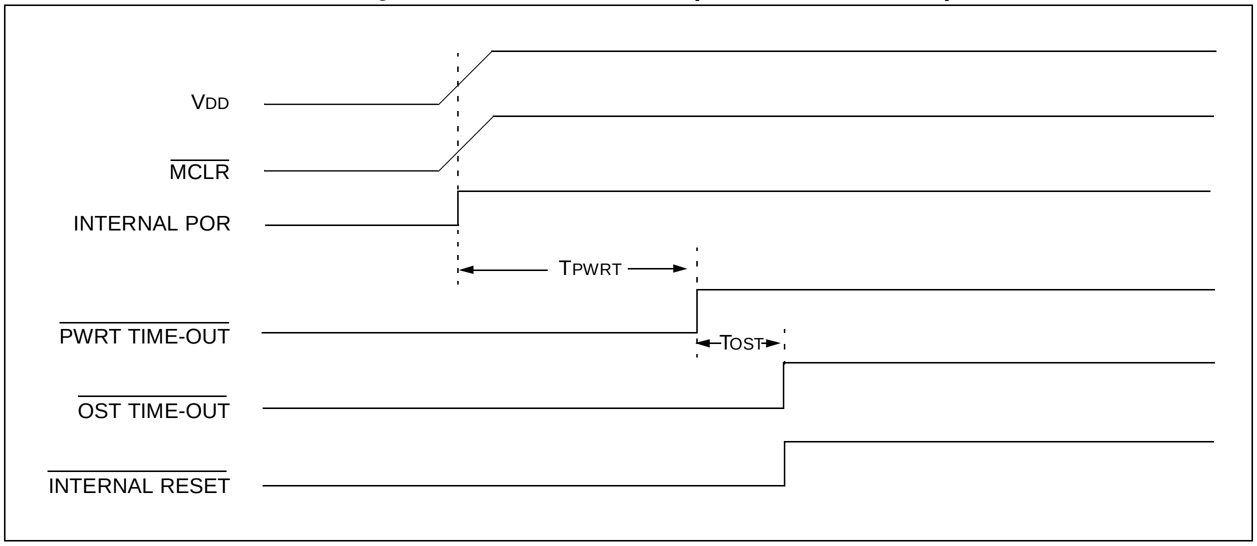
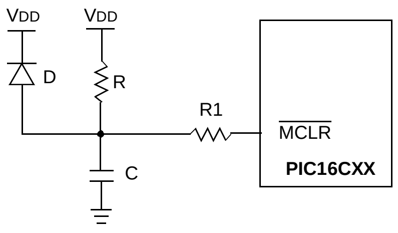


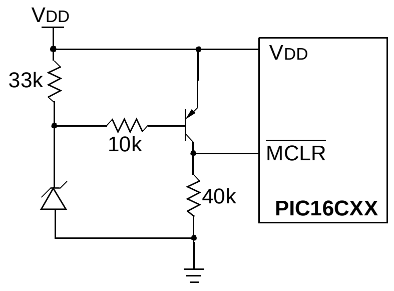
FIGURE 8-11: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



Note:

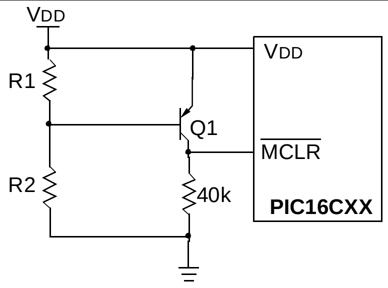
1. External Power-On Reset circuit is required only if VDD power-up rate is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.
2. R < 40 kΩ is recommended to make sure that voltage drop across R does not exceed 0.2V (max leakage current spec on $\overline{\text{MCLR}}$ pin is 5 μA). A larger voltage drop will degrade VIH level on the $\overline{\text{MCLR}}$ pin.
3. R1 = 100Ω to 1 kΩ will limit any current flowing into $\overline{\text{MCLR}}$ from external capacitor C in the event of an $\overline{\text{MCLR}}$ pin breakdown due to ESD or EOS.

FIGURE 8-12: BROWN-OUT PROTECTION CIRCUIT 1



This circuit will activate reset when VDD goes below (Vz + 0.7V) where Vz = Zener voltage.

FIGURE 8-13: BROWN-OUT PROTECTION CIRCUIT 2



This brown-out circuit is less expensive, although less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

$$V_{DD} \bullet \frac{R1}{R1 + R2} = 0.7V$$

8.5 Interrupts

The PIC16C84 family has up to 4 sources of interrupt:

- External interrupt RB0/INT pin
- TMR0 overflow interrupt
- PORTB change interrupts (pins RB7:RB4)
- EEPROM write complete interrupt

The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also contains the individual and global interrupt enable bits.

The global interrupt enable bit, GIE (INTCON<7>) enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding enable bits in INTCON register. GIE is cleared on reset.

The “return from interrupt” instruction, `RETFIE`, exits interrupt routine as well as sets the GIE bit, which re-enable interrupts.

The RB0/INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flags are contained in the INTCON register.

When an interrupt is responded to; the GIE bit is cleared to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with 0004h. For external interrupt events, such as the RB0/INT pin or PORTB change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency depends when the interrupt event occurs (Figure 8-15). The latency is the same for one or two cycle instructions. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid infinite interrupt requests.

Note 1: Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

Note 2: If an interrupt occurs while the Global Interrupt Enable (GIE) bit is being cleared, the GIE bit may unintentionally be re-enabled by the user’s Interrupt Service Routine (the `RETFIE` instruction). The events that would cause this to occur are:

1. An instruction clears the GIE bit while an interrupt is acknowledged
2. The program branches to the Interrupt vector and executes the Interrupt Service Routine.
3. The Interrupt Service Routine completes with the execution of the `RETFIE` instruction. This causes the GIE bit to be set (enables interrupts), and the program returns to the instruction after the one which was meant to disable interrupts.

The method to ensure that interrupts are globally disabled is:

1. Ensure that the GIE bit is cleared by the instruction, as shown in the following code:

```
LOOP    BCF   INTCON,GIE  ; Disable Global
                          ;   Interrupts
       BTFSC INTCON,GIE  ; Global Interrupts
                          ;   Disabled?
        GOTO  LOOP        ; NO, try again
                          ;   Yes, continue
                          ;   with program
                          ;   flow
```

FIGURE 8-14: INTERRUPT LOGIC

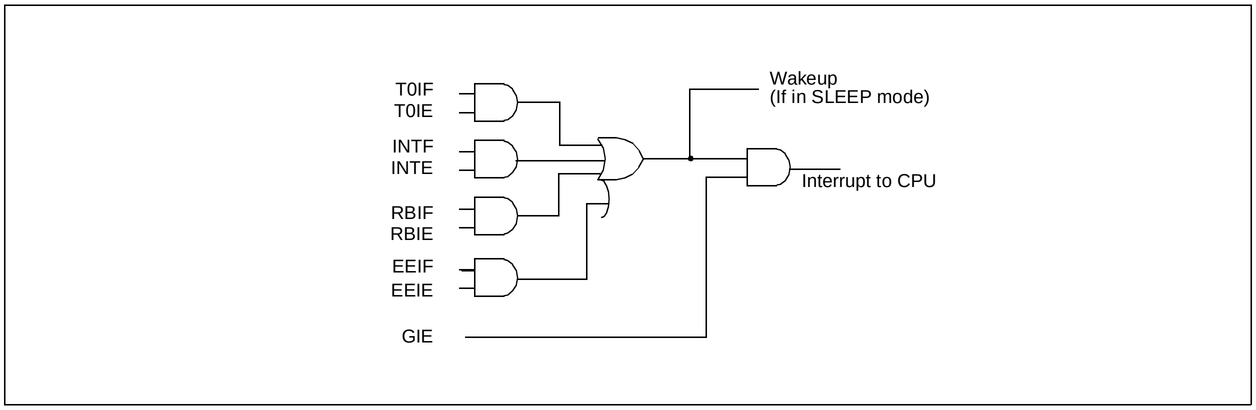
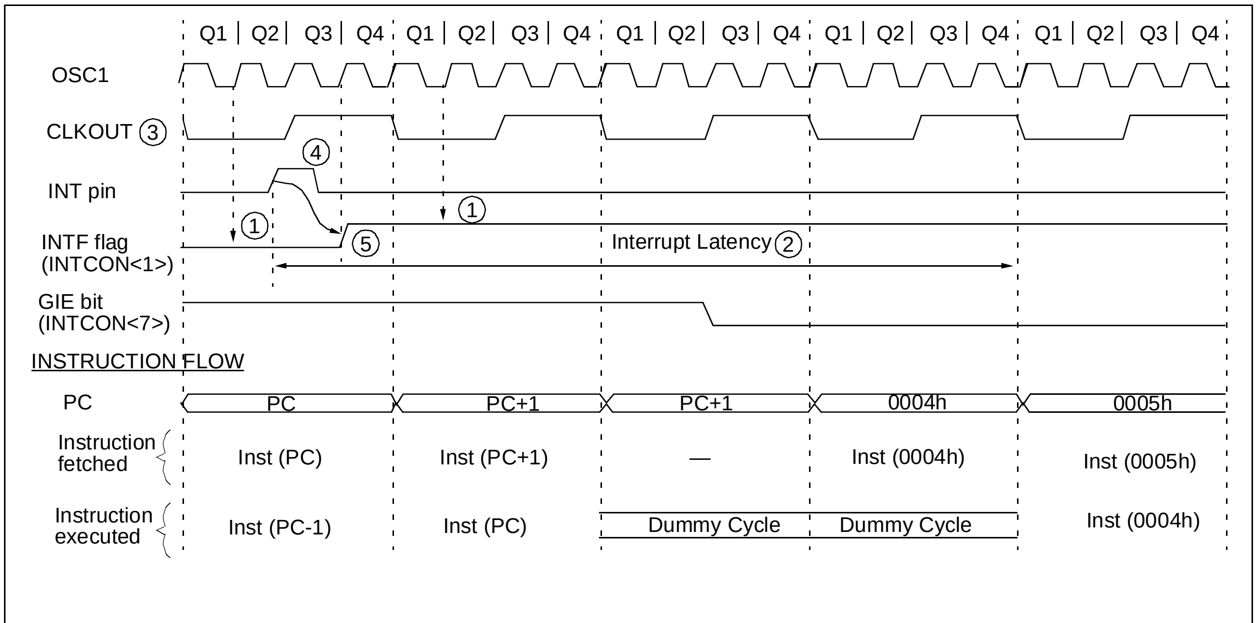


FIGURE 8-15: INT PIN INTERRUPT TIMING



Note 1: INTF flag is sampled here (every Q1).
2: Interrupt latency = 3-4 Tcy where Tcy = instruction cycle time.
Latency is the same whether Inst (PC) is a single cycle or a 2-cycle instruction.
3: CLKOUT is available only in RC oscillator mode.
4: For minimum width of INT pulse, refer to AC specs.
5: INTF is enabled to be set anytime during the Q4-Q1 cycles.

8.5.1 INT INTERRUPT

External interrupt on RB0/INT pin is edge triggered: either rising if INTEDG bit (OPTION<6>) is set, or falling, if INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, the INTF bit (INTCON<1>) is set . This interrupt can be disabled by clearing the INTE control bit (INTCON<4>). The INTF bit must be cleared in software via the interrupt service routine before re-enabling this interrupt. The INT interrupt can wake the processor from SLEEP only if the INTE bit was set prior to going into SLEEP. The status of the GIE bit decides whether the processor branches to the interrupt vector following wake-up. Section 8.8 details SLEEP mode.

8.5.2 TMR0 INTERRUPT

An overflow (FFh $\rightarrow$ 00h) in TMR0 will set the T0IF (INTCON<2>) bit. The interrupt can be enabled/disabled by setting/clearing T0IE (INTCON<5>) bit (Section 6.0).

8.5.3 PORT RB INTERRUPT

An input change on PORTB<7:4> sets the RBIF (INTCON<0>) bit. The interrupt can be enabled/disabled by setting/clearing the RBIE (INTCON<3>) bit (Section 5.2).

Note:	If a change on an I/O pin should occur when a read operation is being executed (start of the Q2 cycle), then the RBIF interrupt flag may not be set.

8.6 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users wish to save key register values during an interrupt (e.g. W register and STATUS register). This is implemented in software.

Example 8-1 stores and restores the STATUS and W register's values. The register, W_TEMP, must be defined in both banks and must be defined at the same offset from the bank base address (i.e., if W_TEMP is defined at 0x20 in bank 0, it must also be defined at 0xA0 in bank 1). User register, STATUS_TEMP, must be defined in bank 0.

Example 8-1 does the following:

a) Stores the W register.
b) Stores the STATUS register in bank 0.
c) Executes the Interrupt Service Routine code.
d) Restores the STATUS (and bank select bit) register.
e) Restores the W register.

EXAMPLE 8-1: SAVING STATUS AND W REGISTERS IN RAM

```
MOVWF   W_TEMP          ; Copy W to TEMP register, could be bank one or zero
SWAPF   STATUS, W       ; Swap status to be saved into W
BCF     STATUS, RP0     ; Change to bank zero, regardless of current bank
MOVWF   STATUS_TEMP     ; Save status to bank zero STATUS_TEMP register
:                       :
:                       ; Interrupt Service Routine
:                       ;
SWAPF   STATUS_TEMP, W  ; Swap STATUS_TEMP register into W
                        ;   (sets bank to    original state)
MOVWF   STATUS          ; Move W into STATUS register
SWAPF   W_TEMP, F       ; Swap W_TEMP
SWAPF   W_TEMP, W       ; Swap W_TEMP into W
```

8.7 Watchdog Timer (WDT)

The watchdog timer is realized as a free running on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin. That means that the WDT will run even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation a WDT time-out generates a device RESET. If the device is in SLEEP mode, a WDT time-out causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by programming configuration fuse WDTE as a '0' (Section 8.1).

8.7.1 WDT PERIOD

The WDT has a nominal time-out period of 18 ms, (with no prescaler). The time-out periods vary with temperature, V_{DD} and process variations from part to part (see DC specs). If longer time-out periods are desired, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT under software control by writing to the OPTION register. Thus, time-out periods up to 2.3 seconds can be realized.

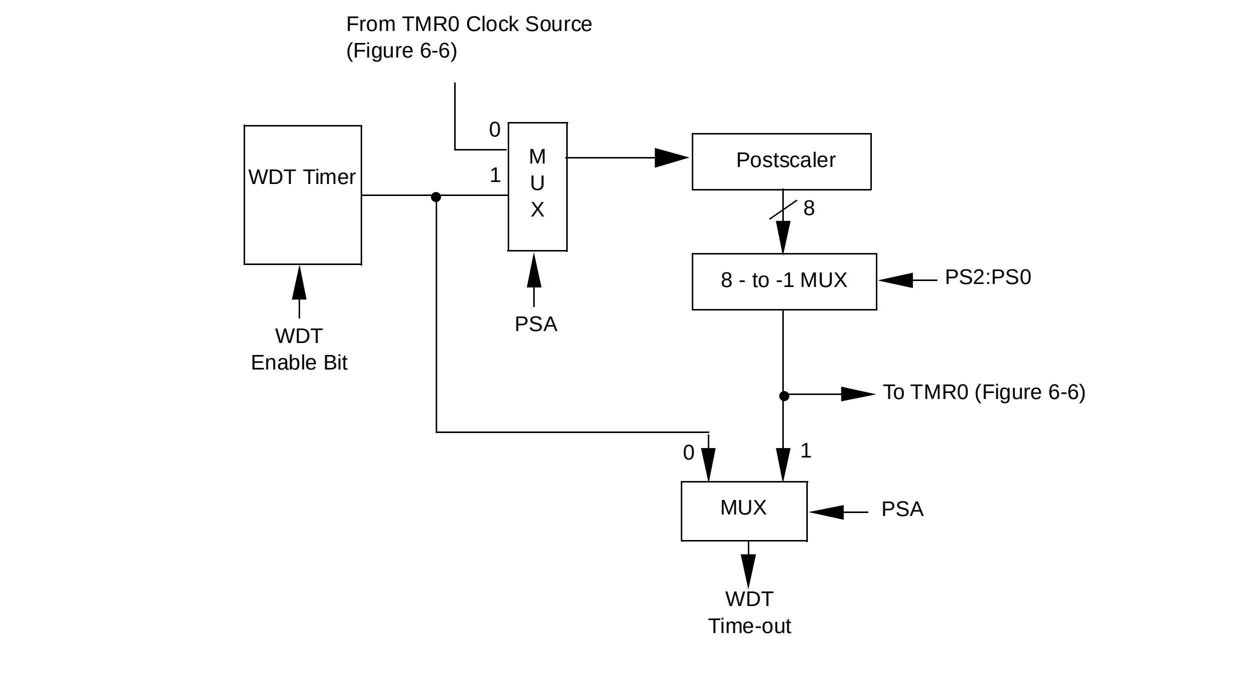
The `CLRWDT` and `SLEEP` instructions clear the WDT and the postscaler (if assigned to the WDT) and prevent it from timing out and generating a device RESET condition.

The $\overline{\text{TO}}$ bit in the STATUS register will be cleared upon a WDT time-out.

8.7.2 WDT PROGRAMMING CONSIDERATIONS

It should also be taken into account that under worst case conditions (V_{DD} = Min., Temperature = Max., max. WDT prescaler) it may take several seconds before a WDT time-out occurs.

FIGURE 8-16: WATCHDOG TIMER BLOCK DIAGRAM



Note: PSA and PS2:PS0 are bits in the OPTION register.

TABLE 8-7: SUMMARY OF REGISTERS ASSOCIATED WITH THE WATCHDOG TIMER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
2007h	Config. bits	–	–	–	CP	PWRTE	WDTE	FOSC1	FOSC0
81h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Note 1: The shaded cells are not used by the Watchdog Timer.

8.8 Power-Down Mode (SLEEP)

The Power-Down mode is entered by executing the SLEEP instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the $\overline{\text{PD}}$ bit in the STATUS register is cleared, the $\overline{\text{TO}}$ bit is set, and the oscillator driver is turned off. The I/O ports maintain the status they had, before the SLEEP instruction was executed (driving high, low, or hi-impedance).

For lowest current consumption, in this mode, all I/O pins should be either at VDD, or VSS, with no external circuitry drawing current from the I/O pin, and disable external clocks. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should also be at VDD or VSS. The contribution from on chip pull-ups on PORTB should be considered.

The $\overline{\text{MCLR}}$ pin must be at a logic high level (VIHMC).

It should be noted that a RESET generated by a WDT time-out does not drive the $\overline{\text{MCLR}}$ pin low.

8.8.1 WAKE-UP FROM SLEEP

The device can wake up from SLEEP through one of the following events:

1. External reset input on $\overline{\text{MCLR}}$ pin.
2. WDT time-out reset (if WDT was enabled).
3. Interrupt from RB0/INT pin, RB port change, or data EEPROM write complete.

Peripherals cannot generate interrupts during SLEEP, since no on-chip Q clocks are present.

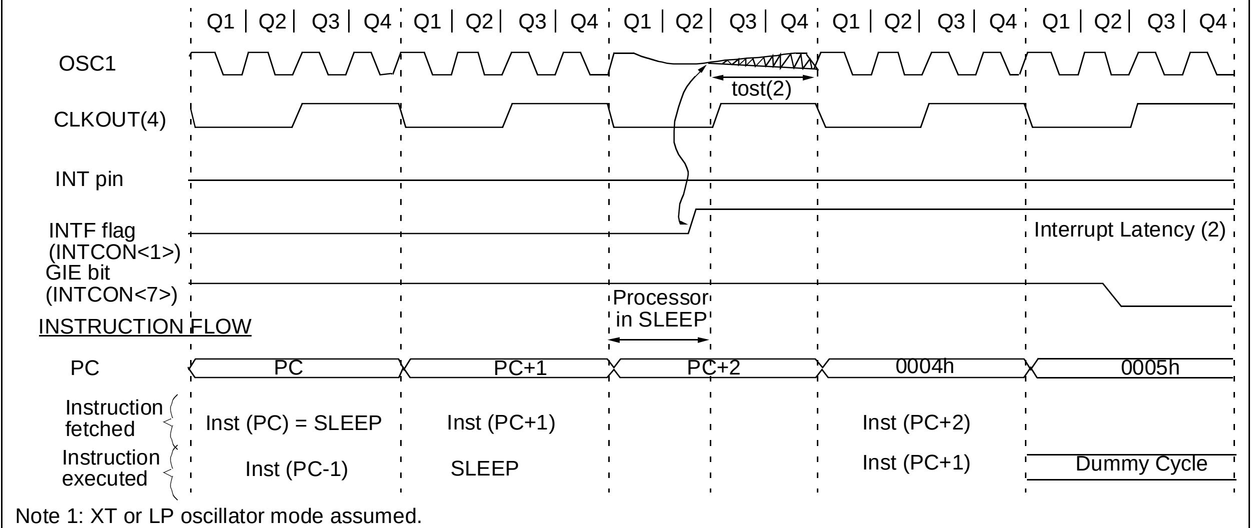
The first event ($\overline{\text{MCLR}}$ reset) will cause a device reset. The two latter events are considered a continuation of program execution. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits can be used to determine the cause of device reset. The $\overline{\text{PD}}$ bit, which is set on power-up, is cleared when SLEEP is invoked. The $\overline{\text{TO}}$ bit is cleared if a WDT time-out occurred (and caused wake-up).

While the SLEEP instruction is being executed, the next instruction (PC + 1) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up occurs regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is set (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt address (0004h). In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

Note:	If the global interrupts are disabled (GIE is cleared), but any interrupt source has both its interrupt enable bit and the corresponding interrupt flag bits set, the device will immediately wake from sleep. The SLEEP instruction is completely executed.

The WDT is cleared when the device wakes-up from sleep, regardless of the source of wake-up.

FIGURE 8-17: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: XT or LP oscillator mode assumed.
2: Tost = 1024 Tosc (drawing not to scale). This delay will not be there for RC osc mode.
3: When GIE is set, processor jumps to interrupt routine after wake-up. If GIE is clear, execution will continue in line.
4: CLKOUT is not available in these osc modes, but shown here for timing reference.

8.9 Code Protection

The code in the program memory and data EEPROM memory can be protected by programming the code protect bit.

Refer to Figure 8-1 for the code protection bit assignment for the PIC16C84.

8.10 ID Locations

Four memory locations (2000h - 2003h) are designated as ID locations to store checksum or other code identification numbers. These locations are not accessible during normal execution but are readable and writable only during program/verify. Only the 4 least significant bits of ID location are usable.

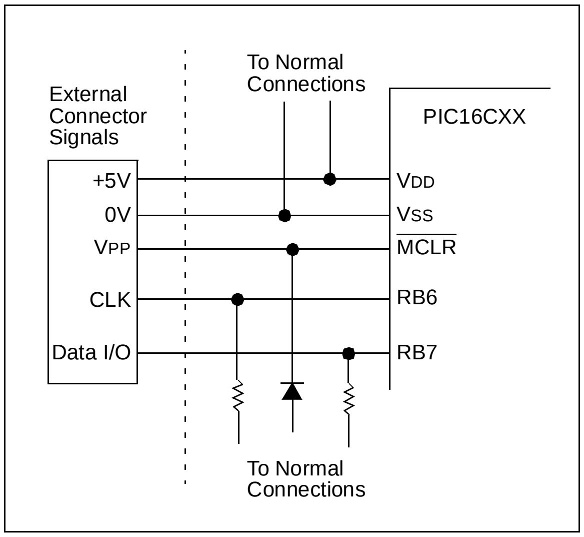
8.11 In-Circuit Serial Programming

PIC16C84 microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground, and the programming voltage. Customers can manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product, allowing the most recent firmware or custom firmware to be programmed.

The device is placed into a program/verify mode by holding the RB6 and RB7 pins low, while raising the $\overline{\text{MCLR}}$ (V_{PP}) pin from V_{IL} to V_{IH} (see programming specification). RB6 becomes the programming clock and RB7 becomes the programming data. Both RB6 and RB7 are Schmitt Trigger inputs in this mode.

After reset, to place the device into programming/verify mode, the program counter (PC) points to location 00h. A 6-bit command is then supplied to the device, 14-bits of program data is then supplied to or from the device, using load or a read-type instructions. For complete details of serial programming, please refer to the PIC16CXX Programming Specifications (Literature #DS30189).

FIGURE 8-18: TYPICAL IN-SYSTEM SERIAL PROGRAMMING CONNECTION



NOTES:

9.0 INSTRUCTION SET SUMMARY

Each PIC16CXX instruction is a 14-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 PIC16CXX instruction set summary in Table 9-2 lists byte-oriented, bit-oriented, and literal and control operations. Table 9-1 shows the opcode field descriptions.

Byte-oriented instructions: 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed in the file register specified by the instruction.

Bit-oriented instructions: 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

Literal and control operations: 'k' represents an eight or eleven bit constant or literal value.

TABLE 9-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 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.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
dest	Destination (Either the W register or the specified register file location)
[ ]	Options
( )	Contents
$\rightarrow$	Assigned to
< >	Register bit field
$\in$	In the set of
italics	User defined term (font is courier)

The instruction set is highly orthogonal and is grouped into three basic categories:

- Byte-oriented
- Bit-oriented
- Literal and control

All instructions are executed within a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. The execution takes two instruction cycles with the second cycle executed as a NOP. Each cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µsec. The instruction execution time is 2 µsec for program branches.

Table 9-2 lists the instructions recognized by Microchip's assembler (MPASM).

Figure 9-1 shows the three general formats of instructions.

Note:	To maintain upward compatibility with future PIC16CXX products, do not use the `OPTION` and `TRIS` instructions.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

FIGURE 9-1: GENERAL FORMAT FOR INSTRUCTIONS

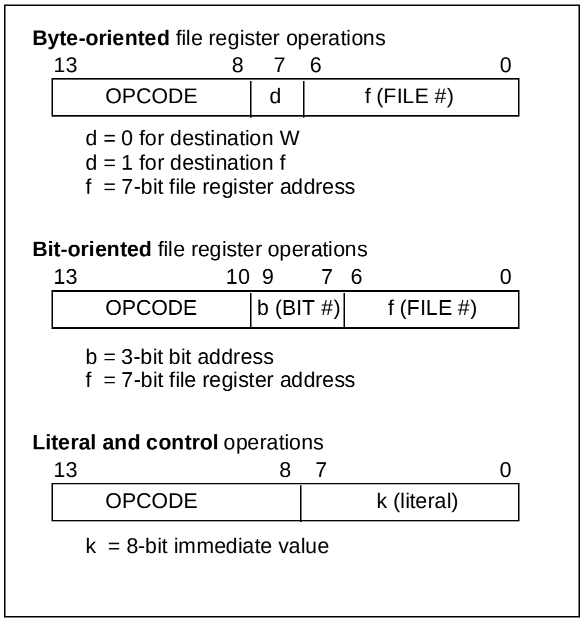

Byte-oriented file register operations
13 8 7 6 0
OPCODE
d
f (FILE #)
d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address
Bit-oriented file register operations
13 10 9 7 6 0
OPCODE
b (BIT #)
f (FILE #)
b = 3-bit bit address
f = 7-bit file register address
Literal and control operations
13 8 7 0
OPCODE
k (literal)
k = 8-bit immediate value

TABLE 9-2: INSTRUCTION SET SUMMARY

Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				msb			lsb		
ADDWF	**f, d**	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	**f, d**	AND W and f	1	00	0101	dfff	ffff	Z	1,2
CLRF	**f**	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	**-**	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	**f, d**	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	**f, d**	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	**f, d**	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff	None	1,2,3
INCF	**f, d**	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	**f, d**	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff	None	1,2,3
IORWF	**f, d**	Inclusive OR W and f	1	00	0100	dfff	ffff	Z	1,2
MOVF	**f, d**	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	**f**	Move W to f	1	00	0000	lfff	ffff	None	
NOP	**-**	No Operation	1	00	0000	0xx0	0000	None	
RLF	**f, d**	Rotate left f through carry	1	00	1101	dfff	ffff	C	1,2
RRF	**f, d**	Rotate right f through carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	**f, d**	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	**f, d**	Swap nibbles in f	1	00	1110	dfff	ffff	None	1,2
XORWF	**f, d**	Exclusive OR W and f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	**f, b**	Bit Clear f	1	01	00bb	bfff	ffff	None	1,2
BSF	**f, b**	Bit Set f	1	01	01bb	bfff	ffff	None	1,2
BTFSC	**f, b**	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff	None	3
BTFSS	**f, b**	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff	None	3
LITERAL AND CONTROL OPERATIONS									
ADDLW	**k**	Add literal to W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	**k**	AND literal to W	1	11	1001	kkkk	kkkk	Z	
CALL	**k**	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	**-**	Clear watchdog timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	**k**	Go to address	2	10	1kkk	kkkk	kkkk	None	
IORLW	**k**	Inclusive OR literal to W	1	11	1000	kkkk	kkkk	Z	
MOVLW	**k**	Move literal to W	1	11	00xx	kkkk	kkkk	None	
RETFIE	**-**	Return from interrupt	2	00	0000	0000	1001	None	
RETLW	**k**	Return with literal in W	2	11	01xx	kkkk	kkkk	None	
RETURN	**-**	Return from subroutine	2	00	0000	0000	1000	None	
SLEEP	**-**	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	**k**	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	**k**	Exclusive OR literal to W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (i.e., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d=1), the prescaler will be cleared if assigned to the TMR0.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

9.1 Instruction Descriptions

ADDLW — Add Literal to W

Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Encoding:	`11` | `111x` | `kkkk` | `kkkk`
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed back in the W register.
Words:	1
Cycles:	1

Example	`ADDLW 0x15`
	Before Instruction
	W = 0x10
	After Instruction
	W = 0x25

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) + (f) → (dest)
Status Affected:	C, DC, Z
Encoding:	`00` | `0111` | `dfff` | `ffff`
Description:	Add the contents of the W register to register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example	`ADDWF FSR, 0`
	Before Instruction
	W = 0x17
	FSR = 0xC2
	After Instruction
	W = 0xD9
	FSR = 0xC2

ANDLW — And Literal to W

Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Encoding:	`11` | `1001` | `kkkk` | `kkkk`
Description:	The contents of W register is AND'ed with the eight bit literal 'k'. The result is placed back in the W register.
Words:	1
Cycles:	1

Example	`ANDLW 0x5F`
	Before Instruction
	W = 0xA3
	After Instruction
	W = 0x03

ANDWF — AND W to f

Syntax:	[*label*] ANDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .AND. (f) → (dest)
Status Affected:	Z
Encoding:	`00` | `0101` | `dfff` | `ffff`
Description:	AND the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example	`ANDWF FSR, 1`
	Before Instruction
	W = 0x17
	FSR = 0xC2
	After Instruction
	W = 0x17
	FSR = 0x02

BCF **Bit Clear f**

Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Encoding:	`01` `00bb` `bfff` `ffff`
Description:	Bit 'b' in register 'f' is cleared.
Words:	1
Cycles:	1
Example	`BCF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

BSF **Bit Set f**

Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Encoding:	`01` `01bb` `bfff` `ffff`
Description:	Bit 'b' in register 'f' is set.
Words:	1
Cycles:	1
Example	`BSF FLAG_REG, 7`

Before Instruction
FLAG_REG= 0x0A
After Instruction
FLAG_REG= 0x8A

BTFSC **Bit Test f, Skip if Clear**

Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Encoding:	`01` `10bb` `bfff` `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 2 cycle instruction.
Words:	1
Cycles:	1(2)
Example	

```
HERE    BTFSC  FLAG,1
FALSE   GOTO   PROCESS_CODE
TRUE    •
        •
        •
```

Before Instruction
PC = address `HERE`
After Instruction
if FLAG<1>=0,
PC=address `TRUE`
if FLAG<1>=1,
PC=address `FALSE`

BTFSS — Bit Test f, skip if Set

Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Encoding:	`01` | `11bb` | `bfff` | `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 2 cycle instruction.
Words:	1
Cycles:	1(2)

Example

```
HERE    BTFSC  FLAG,1
FALSE   GOTO   PROCESS_CODE
TRUE    •
        •
        •
```

Before Instruction
PC = address `HERE`
After Instruction
if FLAG<1>=0,
PC=address `FALSE`
if FLAG<1>=1,
PC=address `TRUE`

CALL — Subroutine Call

Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Encoding:	`10` | `0kkk` | `kkkk` | `kkkk`
Description:	Subroutine call. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two cycle instruction.
Words:	1
Cycles:	2

Example

```
HERE     CALL   THERE
```

Before Instruction
PC = Address `HERE`
After Instruction
PC = Address `THERE`
TOS = Address `HERE`

CLRF — Clear f

Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Encoding:	`00` | `0001` | `1fff` | `ffff`
Description:	The contents of register 'f' are cleared and the Z bit is set.
Words:	1
Cycles:	1

Example

```
CLRF       FLAG_REG
```

Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00
Z = 1

CLRW — Clear W Register

Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Encoding:	`00` | `0001` | `0xxx` | `xxxx`
Description:	W register is cleared. Zero bit (Z) is set.
Words:	1
Cycles:	1

Example

```
CLRW
```

Before Instruction
W = 0x5A
After Instruction
W = 0x00
Z = 1

CLRWDT Clear Watchdog Timer

Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h $\rightarrow$ WDT 0 $\rightarrow$ WDT prescaler, 1 $\rightarrow$ $\overline{\text{TO}}$ 1 $\rightarrow$ $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Encoding:	`00` `0000` `0110` `0100`
Description:	The `CLRWDT` instruction resets the watchdog timer. It also resets the prescaler of the WDT. Status bits $\overline{\text{TO}}$ and $\overline{\text{PD}}$ are set.
Words:	1
Cycles:	1
Example	`CLRWDT`

Before Instruction

WDT counter	=	?

After Instruction

WDT counter	=	0x00
WDT prescale	=	0
$\overline{\text{TO}}$	=	1
$\overline{\text{PD}}$	=	1

COMF Complement f

Syntax:	[*label*] COMF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(\overline{f}) \rightarrow$ (dest)
Status Affected:	Z
Encoding:	`00` `1001` `dfff` `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'.
Words:	1
Cycles:	1
Example	`COMF REG1,0`

Before Instruction

REG1	=	0x13

After Instruction

REG1	=	0x13
W	=	0xEC

DECF Decrement f

Syntax:	[*label*] DECF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f) – 1 $\rightarrow$ (dest)
Status Affected:	Z
Encoding:	`00` `0011` `dfff` `ffff`
Description:	Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	`DECF CNT, 1`

Before Instruction

CNT	=	0x01
Z	=	0

After Instruction

CNT	=	0x00
Z	=	1

DECFSZ Decrement f, Skip if 0

Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f) – 1 $\rightarrow$ (dest); skip if result = 0
Status Affected:	None
Encoding:	`00` `1011` `dfff` `ffff`
Description:	The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'. If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two cycle instruction.
Words:	1
Cycles:	1(2)

Example

```
HERE      DECFSZ   CNT, 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+1`

GOTO **Go to address**

Syntax:	[*label*] GOTO k
Operands:	$0 \le k \le 2047$
Operation:	k → PC<10:0> (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Encoding:	`10` | `1kkk` | `kkkk` | `kkkk`
Description:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.
Words:	1
Cycles:	2

Example	`GOTO THERE`
	After Instruction
	PC = Address THERE

INCFSZ **Increment f, Skip if 0**

Syntax:	[*label*] INCFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → (dest), skip if result = 0
Status Affected:	None
Encoding:	`00` | `1111` | `dfff` | `ffff`
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'. If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two cycle instruction.
Words:	1
Cycles:	1(2)

Example

```
HERE      INCFSZ    CNT,  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 +1

INCF **Increment f**

Syntax:	[*label*] INCF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → (dest)
Status Affected:	Z
Encoding:	`00` | `1010` | `dfff` | `ffff`
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1

Example	`INCF    CNT, 1`
	Before Instruction
	CNT = 0xFF
	Z = 0
	After Instruction
	CNT = 0x00
	Z = 1

IORLW **Inclusive OR Literal to W**

Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. (k) → (W)
Status Affected:	Z
Encoding:	`11` | `1000` | `kkkk` | `kkkk`
Description:	The contents of the W register are OR'ed to the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1

Example	`IORLW   0x35`
	Before Instruction
	W = 0x9A
	After Instruction
	W = 0xBF

IORWF **Inclusive OR W to f**

Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (W)
Status Affected:	Z
Encoding:	`00` | `0100` | `dfff` | `ffff`
Description:	Inclusive OR the W register to register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1
Example	`IORWF RESULT, 0`

Before Instruction
RESULT = 0x13
W = 0x91
After Instruction
RESULT = 0x13
W = 0x93

MOVF **Move f**

Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (dest)
Status Affected:	Z
Encoding:	`00` | `1000` | `dfff` | `ffff`
Description:	The contents of register f is moved to destination d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1
Example	`MOVF FSR, 0`

After Instruction
W = value in FSR register

MOVLW **Move literal to W**

Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Encoding:	`11` | `00XX` | `kkkk` | `kkkk`
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.
Words:	1
Cycles:	1
Example	`MOVLW 0x5A`

After Instruction
W = 0x5A

MOVWF **Move W to f**

Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	`00` | `0000` | `1fff` | `ffff`
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1
Example	`MOVWF OPTION`

Before Instruction
OPTION = 0xFF
W = 0x4F
After Instruction
OPTION = 0x4F
W = 0x4F

NOP — No Operation

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	`00` `0000` `0xx0` `0000`
Description:	No operation.
Words:	1
Cycles:	1
Example	`NOP`

OPTION — Load Option Register

Syntax:	[*label*] OPTION
Operands:	None
Operation:	W → OPTION;
Status Affected:	None
Encoding:	`00` `0000` `0110` `0010`
Description:	The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.
Words:	1
Cycles:	1
Example	

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

RETFIE — Return from Interrupt

Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Encoding:	`00` `0000` `0000` `1001`
Description:	The Stack is popped and Top of Stack (TOS) is loaded into the PC. Interrupts are enabled by setting the Global Interrupt Enable This is a two cycle instruction.
Words:	1
Cycles:	2
Example	`RETFIE`

After Interrupt

PC = TOS
GIE = 1

RETLW — Return Literal to W

Syntax:	[*label*] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → W; TOS → (PC)
Status Affected:	None
Encoding:	`11` `01xx` `kkkk` `kkkk`
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.
Words:	1
Cycles:	2

Example

```
        CALL TABLE  ;W contains table
                    ;offset value
        •           ;W now has table value
        •
        •
TABLE   ADDWF PC   ;W = offset
        RETLW k1   ;Begin table
        RETLW k2   ;
        •
        •
        •
        RETLW kn   ; End of table
```

Before Instruction

W = 0x07

After Instruction

W = value of k7

RETURN Return from Subroutine

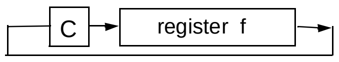
Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → (PC)
Status Affected:	None
Encoding:	`00` | `0000` | `0000` | `1000`
Description:	Return from subroutine. The stack is popped and the Top of Stack (TOS) is loaded into the program counter. This is a two cycle instruction.
Words:	1
Cycles:	2
Example	`RETURN`

After Interrupt

```
PC  =   TOS
```

RRF Rotate Right f through Carry

Syntax:	[*label*] RRF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	`00` | `1100` | `dfff` | `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 the W register. If 'd' is 1 the result is placed back in register 'f'.



Words:	1
Cycles:	1
Example	`RRF         REG1,0`

Before Instruction

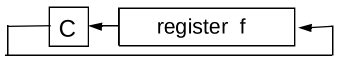
```
REG1   =   1110 0110
C      =   0
```

After Instruction

```
REG1   =   1110 0110
W      =   0111 0011
C      =   1
```

RLF Rotate Left f through Carry

Syntax:	[*label*] RLF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	`00` | `1101` | `dfff` | `ffff`
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.



Words:	1
Cycles:	1
Example	`RLF         REG1,0`

Before Instruction

```
REG1   =   1110 0110
C      =   0
```

After Instruction

```
REG1   =   1110 0110
W      =   1100 1100
C      =   1
```

SLEEP Go into Standby Mode

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler 1 → $\overline{\text{TO}}$, 0 → $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Encoding:	`00` | `0000` | `0110` | `0011`
Description:	The power down status bit ($\overline{\text{PD}}$) is cleared. Time-out status bit ($\overline{\text{TO}}$) is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.
Words:	1
Cycles:	1
Example:	SLEEP

SUBLW — Subtract W from Literal

Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k – (W) → (W)
Status Affected:	C, DC, Z
Encoding:	`11` | `110x` | `kkkk` | `kkkk`
Description:	The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1

Example 1: `SUBLW 0x02`

Before Instruction

```
W  = 1
C  = ?
```

After Instruction

```
W  = 1
C  = 1; result is positive
```

Example 2: Before Instruction

```
W  = 2
C  = ?
```

After Instruction

```
W  = 0
C  = 1;  result is zero
```

Example 3: Before Instruction

```
W  = 3
C  = ?
```

After Instruction

```
W  = FF
C  = 0; result is negative
```

SUBWF — Subtract W from f

Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) – (W) → (dest)
Status Affected:	C, DC, Z
Encoding:	`00` | `0010` | `dfff` | `ffff`
Description:	Subtract (2's complement meth-odize W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example 1: `SUBWF REG1,1`

Before Instruction

```
REG1  = 3
W     = 2
C     = ?
```

After Instruction

```
REG1  = 1
W     = 2
C     = 1; result is positive
```

Example 2: Before Instruction

```
REG1  = 2
W     = 2
C     = ?
```

After Instruction

```
REG1  = 0
W     = 2
C     = 1; result is zero
```

Example 3: Before Instruction

```
REG1  = 1
W     = 2
C     = ?
```

After Instruction

```
REG1  = FF
W     = 2
C     = 0; result is negative
```

SWAPF	**Swap f**			
Syntax:	[*label*] SWAPF f,d			
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]			
Operation:	f<3:0> → d<7:4>, f<7:4> → d<3:0>			
Status Affected:	None			
Encoding:	`00`	`1110`	`dfff`	`ffff`
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.			
Words:	1			
Cycles:	1			
Example	`SWAP F REG, 0`			

Before Instruction

REG1 = 0xA5

After Instruction

REG1 = 0xA5
W = 0x5A

XORLW	**Exclusive OR Literal to W**			
Syntax:	[*label*] XORLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	(W) .XOR. k → (W)			
Status Affected:	Z			
Encoding:	`11`	`1010`	`kkkk`	`kkkk`
Description:	The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.			
Words:	1			
Cycles:	1			
Example:	`XORLW 0xAF`			

Before Instruction

W = 0xB5

After Instruction

W = 0x1A

TRIS	**Load TRIS Register**			
Syntax:	[*label*] TRIS f			
Operands:	5 ≤ f ≤ 7			
Operation:	W → TRIS register f;			
Status Affected:	None			
Encoding:	`00`	`0000`	`0110`	`0fff`
Description:	The instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them.			
Words:	1			
Cycles:	1			
Example				

Note: To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

XORWF	**Exclusive OR W to f**			
Syntax:	[*label*] XORWF f,d			
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]			
Operation:	(W) .XOR. (f) → (dest)			
Status Affected:	Z			
Encoding:	`00`	`0110`	`dfff`	`ffff`
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.			
Words:	1			
Cycles:	1			
Example	`XORWF REG 1`			

Before Instruction

REG = 0xAF
W = 0xB5

After Instruction

REG = 0x1A
W = 0xB5

10.0 DEVELOPMENT SUPPORT

10.1 Development Tools

The PIC16/17 microcontrollers are supported with a full range of hardware and software development tools:

- PICMASTER® Real-Time In-Circuit Emulator
- PRO MATE™ Universal Programmer
- PICSTART® Low-Cost Prototype Programmer
- PICDEM-1 Low-Cost Demonstration Board
- PICDEM-2 Low-Cost Demonstration Board
- MPASM Assembler
- MPSIM Software Simulator
- C Compiler (MP-C)
- Fuzzy logic development system (*fuzzy*TECH®–MP)

10.2 PICMASTER: High Performance Universal In-Circuit Emulator

The PICMASTER Universal In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for all microcontrollers in the PIC16C5X, PIC16CXX and PIC17CXX families. A PICMASTER System configuration is shown in Figure 10-1.

Interchangeable target probes allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the PICMASTER allows expansion to support all new PIC16C5X, PIC16CXX and PIC17CXX microcontrollers.

The Emulator System is designed to operate on PC compatible 386 (and better) machines in the Microsoft Windows™ 3.x environment. Thus, allowing the operator access to a wide range of supporting software and accessories.

The PICMASTER has been designed as a real-time emulation system with advanced features that are generally found on more expensive development tools. The AT platform and Windows 3.x environment was chosen to best make these features available to you, the end user.

The PICMASTER Universal Emulator System consists primarily of four major components:

- Host-Interface Card
- Emulator Control Pod
- Target-Specific Emulator Probe
- PC-Host Emulation Control Software

The Windows 3.x operating system allows the developer to take full advantage of the many powerful features and functions of the PICMASTER system.

PICMASTER emulation can operate in one window, while a text editor is running in a second window.

PC-Host Emulation Control software takes full advantage of Dynamic Data Exchange (DDE), a feature of Windows 3.x. DDE allows data to be dynamically transferred between two or more Windows programs. With this feature, data collected with PICMASTER can be automatically transferred to a spreadsheet or database program for further analysis.

Under Windows 3.x, two or more PICMASTER emulators can be run simultaneously from the same PC making development of multi-microcontroller systems possible (e.g., a system containing a PIC16CXX processor and a PIC17CXX processor).

The PICMASTER probes specifications are shown in Table 10-1.

FIGURE 10-1: PICMASTER SYSTEM CONFIGURATION

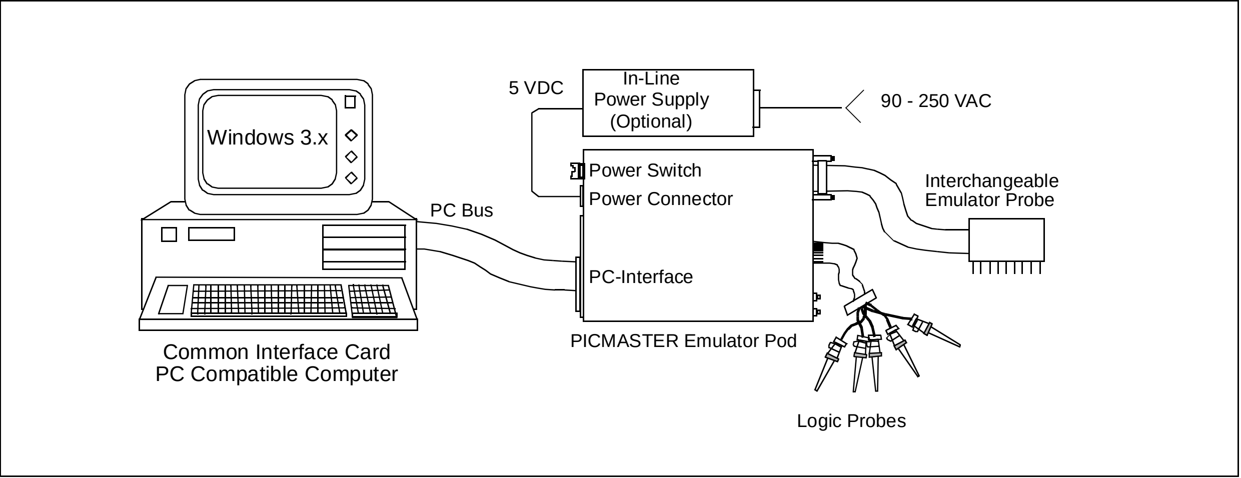


TABLE 10-1: PICMASTER PROBE SPECIFICATION

PICMASTER Probe	Devices Supported	PROBE	
		Maximum Frequency	**Operating Voltage**
PROBE-16B	PIC16C71	10 MHz	4.5V - 5.5V
PROBE-16C	PIC16C84	10 MHz	4.5V - 5.5V
PROBE-16D	PIC16C54, PIC16C54A, PIC16CR54, PIC16C55, PIC16C56, PIC16C57, PIC16CR57A, PIC16C58A, and PIC16CR58A	20 MHz	4.5V - 5.5V
PROBE-16E	PIC16C62 and PIC16C64	10 MHz	4.5V - 5.5V
PROBE-16F	PIC16C65*, PIC16C73 and PIC16C74	10 MHz	4.5V - 5.5V
PROBE-16G	PIC16C61	10 MHz	4.5V - 5.5V
PROBE-16H	PIC16C620, PIC16C621 and PIC16C622	10 MHz	4.5V - 5.5V
PROBE-17A	PIC17C42	16 MHz	4.5V - 5.5V

* PROBE-16F indirectly supports the PIC16C65.

10.3 PRO MATE: Universal Programmer

The PRO MATE Universal Programmer is a full-featured programmer capable of operating in stand-alone mode as well as PC-hosted mode.

The PRO MATE has programmable V_{DD} and V_{PP} supplies which allows it to verify programmed memory at V_{DD} min and V_{DD} max for maximum reliability. It has an LCD display for displaying error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode the PRO MATE can read, verify or program PIC16C5X, PIC16CXX and PIC17CXX devices. It can also set fuse configuration and code-protect bits in this mode.

In PC-hosted mode, the PRO MATE connects to the PC via one of the COM (RS-232) ports. PC based user-interface software makes using the programmer simple and efficient. The user interface is full-screen and menu-based. Full screen display and editing of data, easy selection of fuse configuration and part type, easy selection of V_{DD} min, V_{DD} max and V_{PP} levels, load and store to and from disk files (Intel® hex format) are some of the features of the software. Essential commands such as read, verify, program and blank check can be issued from the screen. Additionally, serial programming support is possible where each part is programmed with a different serial number, sequential or random.

The PRO MATE has a modular “programming socket module”. Different socket modules are required for different processor types and/or package types.

PRO MATE supports all PIC16C5X, PIC16CXX and PIC17CXX processors.

10.4 PICSTART Low-Cost Development System

The PICSTART programmer is an easy to use, very low-cost prototype programmer. It connects to the PC via one of the COM (RS-232) ports. A PC-based user interface software makes using the programmer simple and efficient. The user interface is full-screen and menu-based. PICSTART is not recommended for production programming.

10.5 PICDEM-1 Low-Cost PIC16/17 Demonstration Board

The PICDEM-1 is a simple board which demonstrates the capabilities of several of Microchip’s microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C84, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The users can program the sample microcontrollers provided with the PICDEM-1 board, on a PRO MATE or PICSTART-16B programmer, and easily test firmware. The user can also connect the PICDEM-1 board to the PICMASTER emulator and download the firmware to the emulator for testing. Additional prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push-button switches and eight LEDs connected to PORTB.

10.6 PICDEM-2 Low-Cost PIC16CXX Demonstration Board

The PICDEM-2 is a simple demonstration board that supports the PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-2 board, on a PRO MATE programmer or PICSTART-16C, and easily test firmware. The PICMASTER emulator may also be used with the PICDEM-2 board to test firmware. Additional prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push-button switches, a potentiometer for simulated analog input, a Serial EEPROM to demonstrate usage of the I^2C bus and separate headers for connection to an LCD module and a keypad.

10.7 Assembler (MPASM)

The MPASM Cross Assembler is a PC-hosted symbolic assembler. It supports all microcontroller series including the PIC16C5X, PIC16CXX, and PIC17CXX families.

MPASM offers full featured Macro capabilities, conditional assembly, and several source and listing formats. It generates various object code formats to support Microchip's development tools as well as third party programmers.

MPASM allows full symbolic debugging from the Microchip Universal Emulator System (PICMASTER).

MPASM has the following features to assist in developing software for specific use applications.

- Provides translation of Assembler source code to object code for all Microchip microcontrollers.
- Macro assembly capability
- Produces all the files (Object, Listing, Symbol, and special) required for symbolic debug with Microchip's emulator systems.
- Supports Hex (default), Decimal and Octal source and listing formats.

MPASM provides a full feature directive language represented by four basic classes of directives:

- **Data Directives** are those that control the allocation of memory and provide a way to refer to data items symbolically, i.e., by meaningful names.
- **Listing Directives** control the MPASM listing display. They allow the specification of titles and subtitles, page ejects and other listing control.
- **Control Directives** permit sections of conditionally assembled code.
- **Macro Directives** control the execution and data allocation within macro body definitions.

10.8 Software Simulator (MPSIM)

The MPSIM Software Simulator allows code development in a PC host environment. It allows the user to simulate the PIC16/17 series microcontrollers on an instruction level. On any given instruction, the user may examine or modify any of the data areas or provide external stimulus to any of the pins. The input/output radix can be set by the user and the execution can be performed in; single step, execute until break, or in a trace mode. MPSIM fully supports symbolic debugging using MP-C and MPASM. The Software Simulator offers the low cost flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

10.9 C Compiler (MP-C)

The MP-C Code Development System is a complete 'C' compiler and integrated development environment for Microchip's PIC16/17 family of microcontrollers. The compiler provides powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compiler provides symbol information that is compatible with the PICMASTER Universal Emulator memory display (emulator software versions 1.13 and later).

The MP-C Code Development System is supplied directly by Byte Craft Limited of Waterloo, Ontario, Canada. If you have any questions, please contact your regional Microchip FAE or Microchip technical support personnel at (602) 786-7627.

10.10 Fuzzy Logic Development System (*fuzzy*TECH-MP)

*fuzzy*TECH-MP fuzzy logic development tool is available in two versions - a low cost introductory version, MP Explorer, for designers to gain a comprehensive working knowledge of fuzzy logic system design; and a full-featured version, *fuzzy*TECH-MP Edition, for implementing more complex systems.

Both versions include Microchip's *fuzzy*LAB™ demonstration board for hands-on experience with fuzzy logic systems implementation.

10.11 Development Systems

For convenience, the development tools are packaged into comprehensive systems as listed in Table 10-2.

TABLE 10-2: DEVELOPMENT SYSTEM PACKAGES

Item	Name	System Description
1.	PICMASTER System	PICMASTER In-Circuit Emulator, PRO MATE Programmer, Assembler, Software Simulator, Samples and your choice of Target Probe.
2.	PICSTART System	PICSTART Low-Cost Prototype Programmer, Assembler, Software Simulator and Samples.
3.	PRO MATE System	PRO MATE Universal Programmer, full featured stand-alone or PC-hosted programmer, Assembler, Simulator

11.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias	-55 to +125°C
Storage temperature	-65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	0 to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} (Note 2)	0 to +14V
Voltage on all other pins with respect to V_{SS}	-0.6V to (V_{DD} + 0.6V)
Total power dissipation (Note 1)	800 mW
Maximum current out of V_{SS} pin	150 mA
Maximum current into V_{DD} pin	100 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD})	±20 mA
Output clamp current, I_{OK} (V0 < 0 or V0 >V_{DD})	±20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	20 mA
Maximum current sunk by PORTA	80 mA
Maximum current sourced by PORTA	50 mA
Maximum current sunk by PORTB	150 mA
Maximum current sourced by PORTB	100 mA

Note 1: Power dissipation is calculated as follows: Pdis = V_{DD} x $\{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{Ol} \times I_{OL})$

Note 2: Voltage spikes below V_{SS} at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a “low” level to the $\overline{MCLR}$ pin rather than pulling this pin directly to V_{SS}.

† 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.

TABLE 11-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)

OSC	16C84-04	16C84-10	16LC84-04
RC	V_{DD}: 4.0V to 6.0V I_{DD}: 4.5 mA max. at 5.5V I_{PD}: 100 µA max. at 4V WDT dis Freq: 4 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 1.0 µA typ. at 4V WDT dis Freq: 4 MHz max.	V_{DD}: 2.0V to 6.0V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 1.0 µA typ. at 3V WDT dis Freq: 4 MHz max.
XT	V_{DD}: 4.0V to 6.0V I_{DD}: 4.5 mA max. at 5.5V I_{PD}: 100 µA max. at 4V WDT dis Freq: 4 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 1.0 µA typ. at 4V WDT dis Freq: 4 MHz max.	V_{DD}: 2.0V to 6.0V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 1.0 µA typ. at 3V WDT dis Freq: 4 MHz max.
HS	V_{DD}: 4.5V to 5.5V I_{DD}: 4.5 mA typ. at 5.5V I_{PD}: 1.0 µA typ. at 4.5V WDT dis Freq: 4 MHz	V_{DD}: 4.5V to 5.5V I_{DD}: 10 mA max. at 5.5V typ. I_{PD}: 1.0 µA typ. at 4.5V WDT dis Freq: 10 MHz max.	Do not use in HS mode
LP	V_{DD}: 2.0V to 6.0V I_{DD}: 60 µA typ. at 32 kHz, 2.0V I_{PD}: 26 µA typ. at 2.0V WDT dis Freq: 200 kHz max.	Do not use in LP mode	V_{DD}: 2.0V to 6.0V I_{DD}: 60 µA max. at 32 kHz, 2.0V I_{PD}: 100 µA max. at 4.0V WDT dis Freq: 200 kHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications. It is recommended that the user select the device type that guarantees the specifications required.

11.1 DC CHARACTERISTICS: PIC16C84-04 (Commercial, Industrial) PIC16C84-10 (Commercial, Industrial)

DC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial and 0°C ≤ T_A ≤ +70°C for commercial

Characteristic	Sym	Min	Typ†	Max	Units	Conditions
Supply Voltage	V_{DD}	4.0 4.5	— —	6.0 5.5	V V	XT, RC and LP osc configuration (16C84-04) HS osc configuration (16C84-10)
RAM Data Retention Voltage (Note 1)	V_{DR}	1.5*	—	—	V	Device in SLEEP mode
V_{DD} start voltage to guarantee Power-On Reset	V_{POR}	—	V_{SS}	—	V	See section on Power-On Reset for details
V_{DD} rise rate to guarantee Power-On Reset	S_{VDD}	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	I_{DD}	— — — —	7.3 1.8 35 5	10 4.5 400 10	mA mA µA mA	RC and XT osc configuration F_{OSC} = 4 MHz, V_{DD} = 5.5V During EEPROM programming F_{OSC} = 4 MHz, V_{DD} = 5.5V (Note 4) LP osc configuration (PIC16C84-04) F_{OSC} = 32 kHz, V_{DD} = 4.0V, WDT disabled HS osc configuration (PIC16C84-10) F_{OSC} = 10 MHz, V_{DD} = 5.5V
Power Down Current (Note 3)	I_{PD}	— — —	40 38 38	100 100 100	µA µA µA	V_{DD} = 4.0V, WDT enabled, -40°C to +85°C V_{DD} = 4.0V, WDT disabled, -0°C to +70°C V_{DD} = 4.0V, WDT disabled, -40°C to +85°C

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which V_{DD} can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on current consumption.
The test conditions for all I_{DD} measurements in active operation mode are:
$\overline{\text{OSC1}}$ = external square wave, from rail to rail; all I/O pins tristated, pulled to V_{DD}, T0CKI = V_{DD}, $\overline{\text{MCLR}}$ = V_{DD}; WDT enabled/disabled as specified.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to V_{DD} or V_{SS}.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula $I_R = V_{DD}/2Rext$ (mA) with Rext in kOhm.

11.2 DC CHARACTERISTICS: PIC16LC84-04 (Commercial, Industrial)

DC CHARACTERISTICS						**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial and 0°C ≤ T_A ≤ +70°C for commercial
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Supply Voltage	V_{DD}	2.0	—	6.0	V	XT, RC, and LP osc configuration
RAM Data Retention Voltage (Note 1)	V_{DR}	1.5 *	—	—	V	Device in SLEEP mode
V_{DD} start voltage to guarantee Power-On Reset	V_{POR}	—	V_{SS}	—	V	See section on Power-On Reset for details
V_{DD} rise rate to guarantee Power-On Reset	S_{VDD}	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	I_{DD}	—	7.3	10	mA	RC and XT osc configuration F_{OSC} = 4 MHz, V_{DD} = 5.5V
		—	1.8	4.5	mA	During EEPROM programming F_{OSC} = 4 MHz, V_{DD} = 5.5V (Note 4)
		—	60	400	µA	LP osc configuration F_{OSC} = 32 kHz, V_{DD} = 2.0V, WDT disabled
Power Down Current (Note 3)	I_{PD}	—	26	100	µA	V_{DD} = 2.0V, WDT enabled, -40°C to +85°C
		—	26	100	µA	V_{DD} = 2.0V, WDT disabled, 0°C to +70°C
		—	26	100	µA	V_{DD} = 2.0V, WDT disabled, -40°C to +85°C

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which V_{DD} can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
The test conditions for all I_{DD} measurements in active operation mode are:
$\overline{\text{OSC1}}$=external square wave, from rail to rail; all I/O pins tristated, pulled to V_{DD}, T0CKI = V_{DD},
$\overline{\text{MCLR}}$ = V_{DD}; WDT enabled/disabled as specified.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to V_{DD} or V_{SS}.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula $I_R = V_{DD}/2Rext$ (mA) with Rext in kOhm.

11.3 DC CHARACTERISTICS: PIC16C84-04 (Commercial, Industrial) PIC16C84-10 (Commercial, Industrial) PIC16LC84-04 (Commercial, Industrial)

DC CHARACTERISTICS						Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial Operating voltage VDD range as described in DC spec Table 11-1 and Table 11-2.
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Input Low Voltage						
I/O ports	VIL					
with TTL buffer		VSS	—	0.8	V	
with Schmitt Trigger buffer		VSS	—	0.2 VDD	V	
$\overline{\text{MCLR}}$, RA4/T0CKI,OSC1 (in RC mode)		Vss	—	0.2 VDD	V	
OSC1 (in XT, HS and LP)		Vss	—	0.3 VDD	V	Note1
Input High Voltage						
I/O ports	VIH					
with TTL buffer		0.36 VDD	—	VDD	V	VDD ≤ 5.5V (Note 4)
with Schmitt Trigger buffer		0.45 VDD	—	VDD		VDD ≤ 6.0V (Note 4)
$\overline{\text{MCLR}}$, RA4/T0CKI, OSC1 (RC mode)		0.85 VDD	—	VDD	V	
OSC1 (XT, HS and LP)		0.7 VDD	—	VDD	V	Note1
PORTB weak pull-up current	IPURB	50*	250*	400*	µA	VDD = 5V, VPIN = VSS
Input Leakage Current (Notes 2, 3)						
I/O ports	IIL	—	—	±1	µA	Vss ≤ VPIN ≤ VDD, Pin at hi-impedance
$\overline{\text{MCLR}}$, RA4/T0CKI		—	—	±5	µA	Vss ≤ VPIN ≤ VDD
OSC1/CLKIN		—	—	±5	µA	Vss ≤ VPIN ≤ VDD, XT, HS and LP osc configuration
Output Low Voltage						
I/O ports	VOL	—	—	0.6	V	IOL = 8.5 mA, VDD = 4.5V, -40°C to +85°C
OSC2/CLKOUT (RC osc configuration)		—	—	0.6	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
Output High Voltage						
I/O ports (Note 3)	VOH	VDD - 0.7	—	—	V	IOH = -3.0 mA, VDD = 4.5V, -40°C to +85°C
OSC2/CLKOUT (RC osc configuration)		VDD - 0.7	—	—	V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
Capacitive Loading Specs on Output Pins						
OSC2/CLKOUT pin	COSC2	—	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1.
All I/O pins and OSC2 (in RC mode)	CIO	—	—	50	pF	

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC16C84 be driven with external clock in RC mode.

2: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: The user may use better of the two specs.

11.4 DC CHARACTERISTICS: PIC16C84-04 (Commercial, Industrial) PIC16C84-10 (Commercial, Industrial) PIC16LC84-04 (Commercial, Industrial)

DC CHARACTERISTICS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for industrial and 0°C ≤ T_A ≤ +70°C for commercial Operating voltage V_{DD} range as described in DC spec Table 11-1 and Table 11-2					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Data EEPROM Memory						
Endurance	E_D	100,000	1,000,000	—	E/W	
V_{DD} for read/write	V_{DRW}	V_{MIN}	—	0.2 V_{DD}	V	V_{MIN} = Minimum operating voltage
Erase/Write cycle time	T_{DEW}	—	10	—	ms	Note1
Program EEPROM Memory						
Endurance	E_P	100	—	—	E/W	
V_{DD} for read	V_{PR}	V_{MIN}	—	V_{DD}	V	V_{MIN} = Minimum operating voltage
V_{DD} for erase/write	V_{PEW}	4.5	—	5.5	V	
Erase/Write cycle time	T_{PEW}	—	10	-	ms	Note1

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: The user should use interrupts or pull the EEIF or WR bits to ensure the write cycle has completed.

11.5 Timing Parameter Symbology

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS	3. TCC:ST (I²C specifications only)
2. TppS	4. Ts (I²C specifications only)

T			
F	Frequency	T	Time

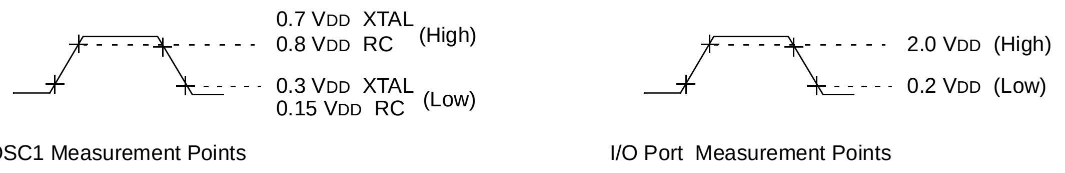
Lowercase symbols (pp) and their meanings:

pp			
ck	CLKOUT	osc	OSC1
io	I/O port	t0	T0CKI
mc	$\overline{\text{MCLR}}$		

Uppercase symbols and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance

FIGURE 11-1: PARAMETER MEASUREMENT INFORMATION



All timings are measured between high and low measurement points as indicated in the figure.

11.6 Timing Diagrams and Specifications

FIGURE 11-2: EXTERNAL CLOCK TIMING

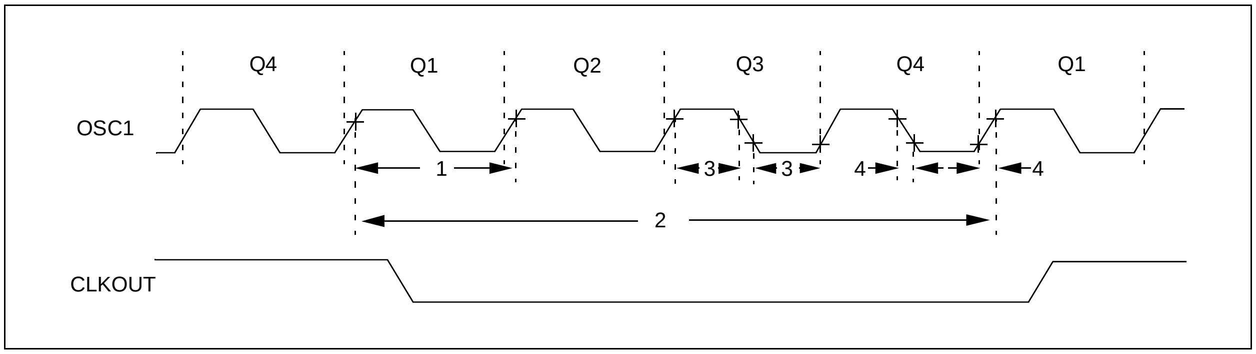


TABLE 11-2: EXTERNAL CLOCK TIMING REQUIREMENTS

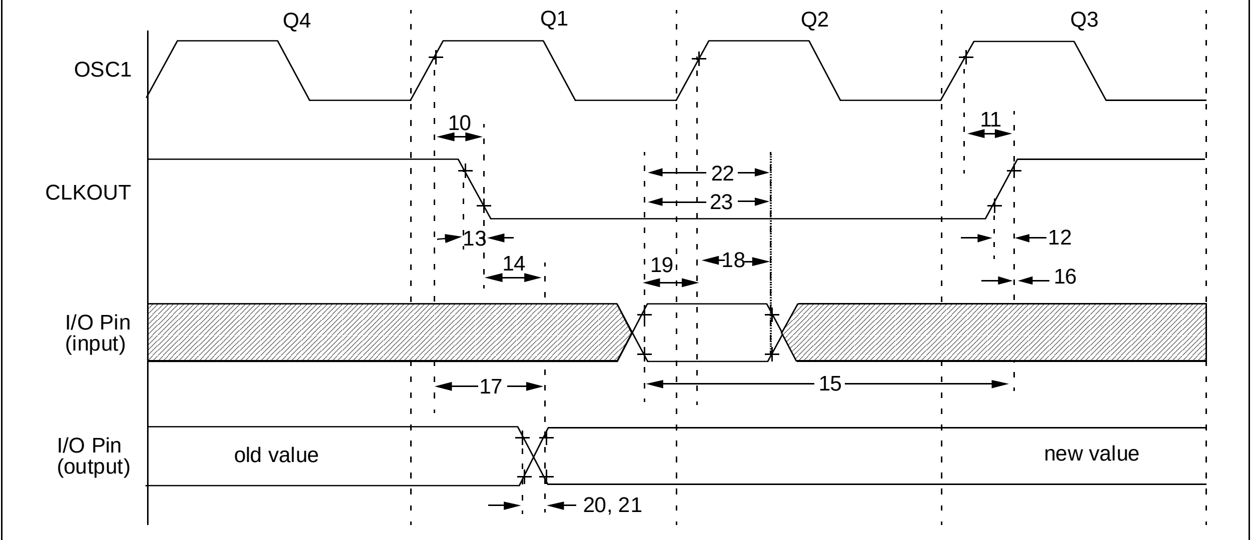
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	**External CLKIN Frequency (Note 1)**	DC	—	2	MHz	XT, RC osc mode, 2V ≤VDD ≤6V
			DC	—	4	MHz	XT, RC osc mode, 3V ≤VDD ≤6V
			DC	—	10	MHz	HS osc mode (PIC16C84-10)
			DC	—	200	kHz	LP osc mode
		Oscillator Frequency (Note 1)	DC	—	2	MHz	RC osc mode, 2V ≤ VDD ≤ 6V
			DC		4	MHz	RC osc mode, 3V ≤ VDD ≤ 6V
			0.1	—	2	MHz	XT osc mode, 2V ≤ VDD ≤ 6V
			0.1	—	4	MHz	XT osc mode, 3V ≤ VDD ≤ 6V
			1		10	MHz	HS osc mode (PIC16C84-04)
			DC	—	200	kHz	LP osc mode (PIC16LC84-04)
1	Tosc	**External CLKIN Period (Note 1)**	250	—	—	ns	XT and RC osc mode
			250	—	—	ns	HS osc mode (PIC16C84-04)
			100	—	—	ns	HS osc mode (PIC16C84-10)
			5	—	—	µs	LP osc mode
		Oscillator Period (Note 1)	250	—	—	ns	RC osc mode
			250	—	10,000	ns	XT osc mode
			250	—	1,000	ns	HS osc mode
			100	—	1,000	ns	HS osc mode (PIC16C84-10)
			5	—	—	µs	LP osc mode
2	TCY	**Instruction Cycle Time (Note 1)**	0.4	4/Fosc	DC	µs	
3	TosL,TosH	Clock in (OSC1) Low or High Time	60 *	—	—	ns	XT oscillator, 2.0V ≤ VDD ≤ 3.0V
			50 *	—	—	ns	XT oscillator, 3.0V ≤ VDD ≤ 6.0V
			2 *	—	—	µs	LP oscillator
			50 *	—	—	ns	HS oscillator
4	TosR,TosF	Clock in (OSC1) Rise or Fall Time	25 *	—	—	ns	XT oscillator
			50 *	—	—	ns	LP oscillator
			25 *	—	—	ns	HS oscillator

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1/CLKIN pin.
When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

FIGURE 11-3: CLKOUT AND I/O TIMING



Note: All tests must be done with specified capacitive loads (see datasheet) 50 pF on I/O pins and CLKOUT.

TABLE 11-3: CLKOUT AND I/O TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1↑ to CLKOUT↓	—	15	30	ns	Note 1
11	TosH2ckH	OSC1↑ to CLKOUT↑	—	15	30	ns	Note 1
12	TckR	CLKOUT rise time	—	5	15	ns	Note 1
13	TckF	CLKOUT fall time	—	5	15	ns	Note 1
14	TckL2ioV	CLKOUT↓ to Port out valid	—	—	0.5TCY+20	ns	Note 1
15	TioV2ckH	Port in valid before CLKOUT↑	0.30TCY+30 *	—	—	ns	Note 1
16	TckH2ioI	Port in hold after CLKOUT↑	0 *	—	—	ns	Note 1
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	—	—	100	ns	
18	TosH2ioI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	TBD	—	—	ns	
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)	TBD	—	—	ns	
20	TioR	Port output rise time	—	10	25	ns	
21	TioF	Port output fall time	—	10	25	ns	
22	Tinp	INT pin high or low time	20 *	—	—	ns	
23	Trbp	RB<7:4> change INT high or low time	20 *	—	—	ns	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Measurements are taken in RC Mode where CLKOUT output is 4 x TOSC.

FIGURE 11-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

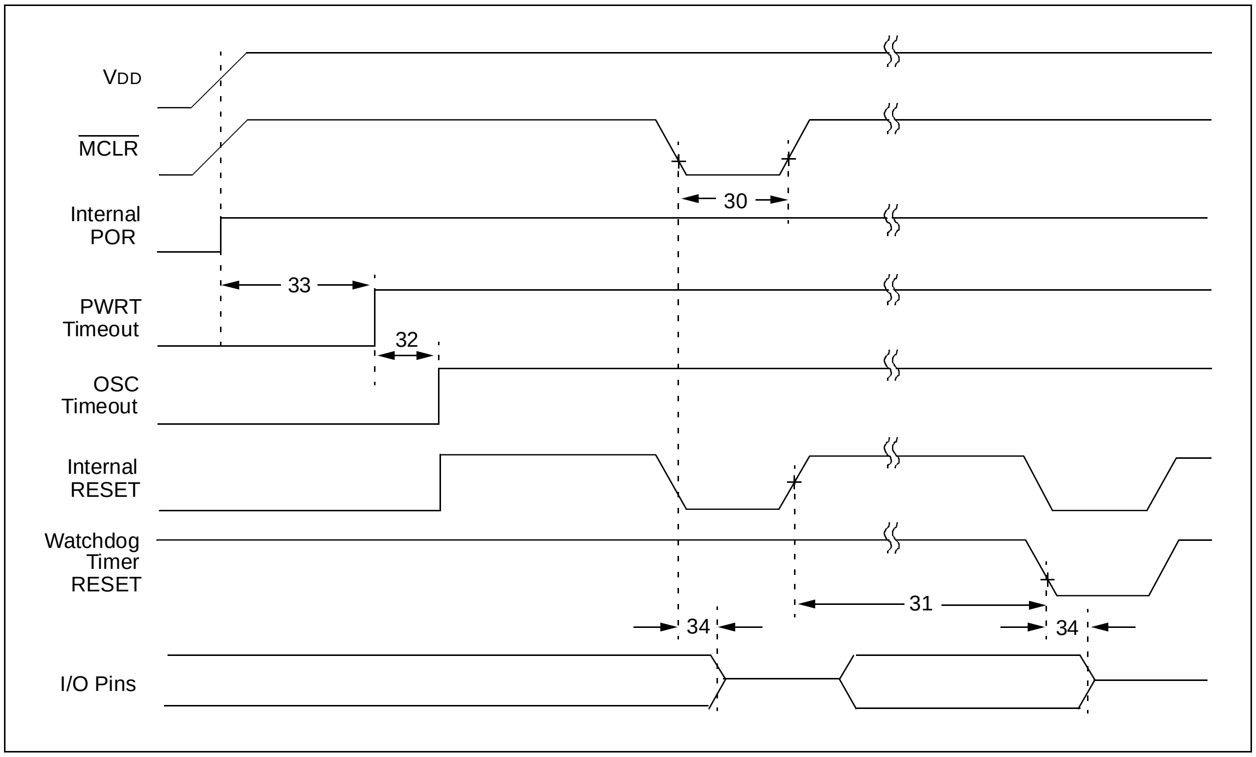


TABLE 11-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)	350 * 150 *	— —	— —	ns ns	2.0V ≤ VDD ≤ 3.0V 3.0V ≤ VDD ≤ 6.0V
31	Twdt	Watchdog Timer Timeout Period (No Prescaler)	7*	18	33*	ms	VDD = 5V, -40°C to +125°C
32	Tost	Oscillation Start-up Timer Period	—	1024 TOSC	—	ms	TOSC = OSC1 period
33	Tpwrt	Power up Timer Period	28*	72	132*	ms	VDD = 5V, -40°C to +125°C
34	TIOZ	I/O Hi-impedance from $\overline{\text{MCLR}}$ Low or reset	—	—	100	ns	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 11-5: TIMER0 CLOCK TIMINGS

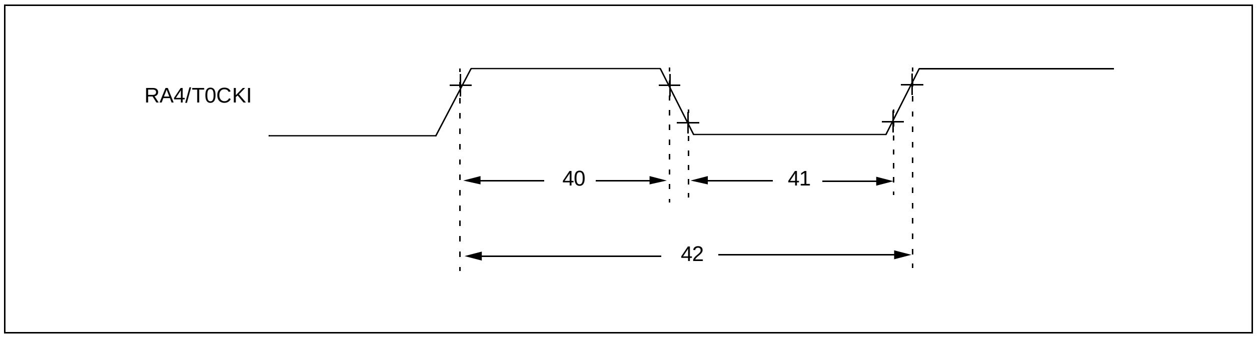


TABLE 11-5: TIMER0 CLOCK REQUIREMENTS

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5 T_{CY} + 20*	—	—	ns	
			With Prescaler	50 * 30 *	— —	— —	ns ns	2.0V ≤ V_{DD} ≤ 3.0V 3.0V ≤ V_{DD} ≤ 6.0V
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5 T_{CY} + 20*	—	—	ns	
			With Prescaler	50 * 20 *	— —	— —	ns ns	2.0V ≤ V_{DD} ≤ 3.0V 3.0V ≤ V_{DD} ≤ 6.0V
42	Tt0P	T0CKI Period		$\frac{T_{CY} + 40}{N}$*	—	—	ns	N = prescale value (2, 4, ..., 256)

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

12.0 DC & AC CHARACTERISTICS GRAPHS/TABLES

The data graphs and tables provided in this section are for design guidance and are not tested or guaranteed. In some graphs or tables the data presented is outside specified operating range (e.g., outside specified VDD range). This is for information only and devices are guaranteed to operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. "Typical" represents the mean of the distribution while 'max' or 'min' represents (mean + 3σ) and (mean - 3σ) respectively where σ is standard deviation.

FIGURE 12-1: TYPICAL RC OSCILLATOR FREQUENCY vs. TEMPERATURE

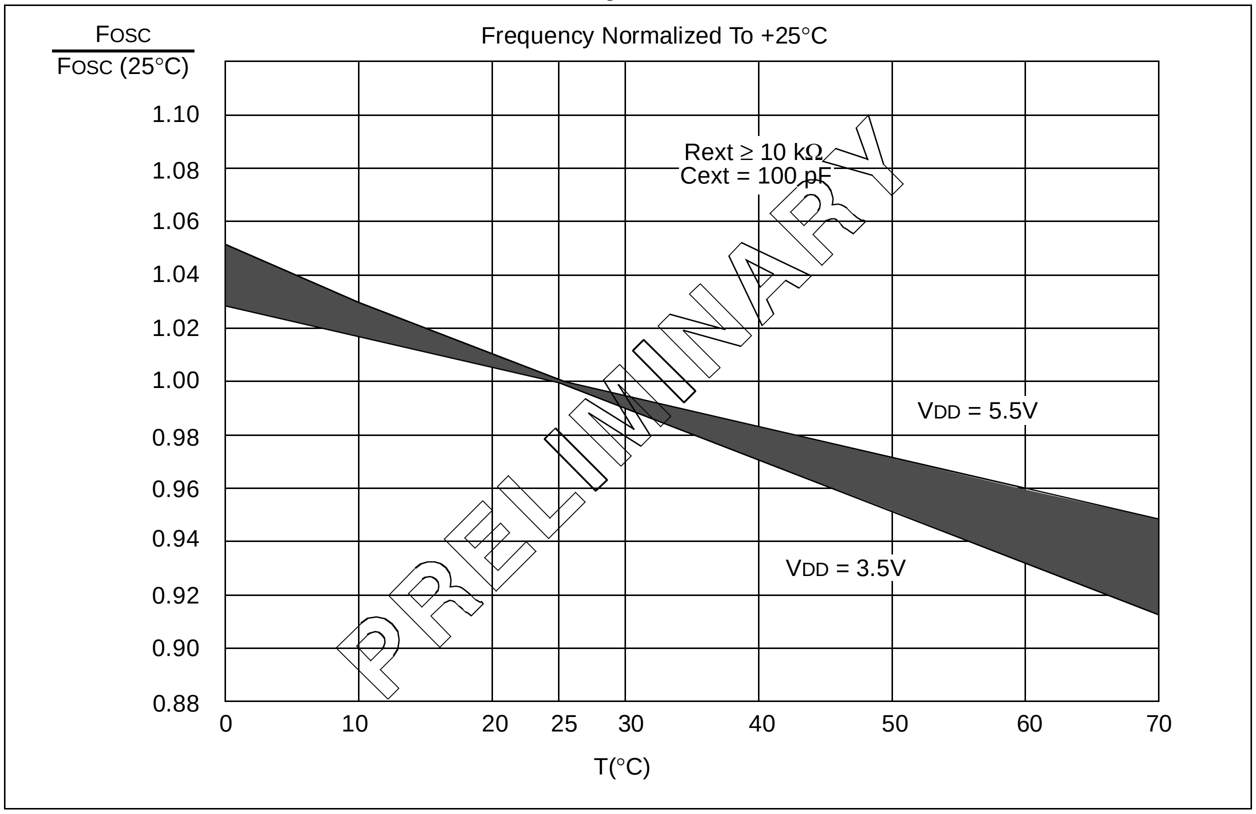


TABLE 12-1: RC OSCILLATOR FREQUENCIES *

Cext	Rext	Average Fosc @ 5V, 25°C	
20 pF	3.3k	4.68 MHz	± 27%
	5.1k	3.94 MHz	± 25%
	10k	2.34 MHz	± 29%
	100k	250.16 kHz	± 33%
100 pF	3.3k	1.49 MHz	± 25%
	5.1k	1.12 MHz	± 25%
	10k	620.31 kHz	± 30%
	100k	90.25 kHz	± 26%
300 pF	3.3k	524.24 kHz	± 28%
	5.1k	415.52 kHz	± 30%
	10k	270.33 kHz	± 26%
	100k	25.37 kHz	± 25%

*Measured in PDIP Packages.The percentage variation indicated here is part to part variation due to normal process distribution. The variation indicated is ±3 standard deviation from average value.

FIGURE 12-2: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD*

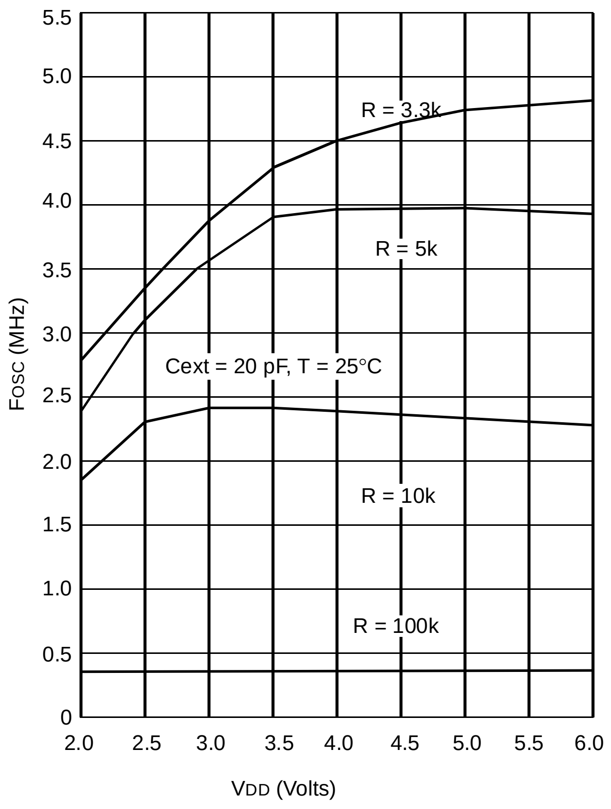


FIGURE 12-3: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD*

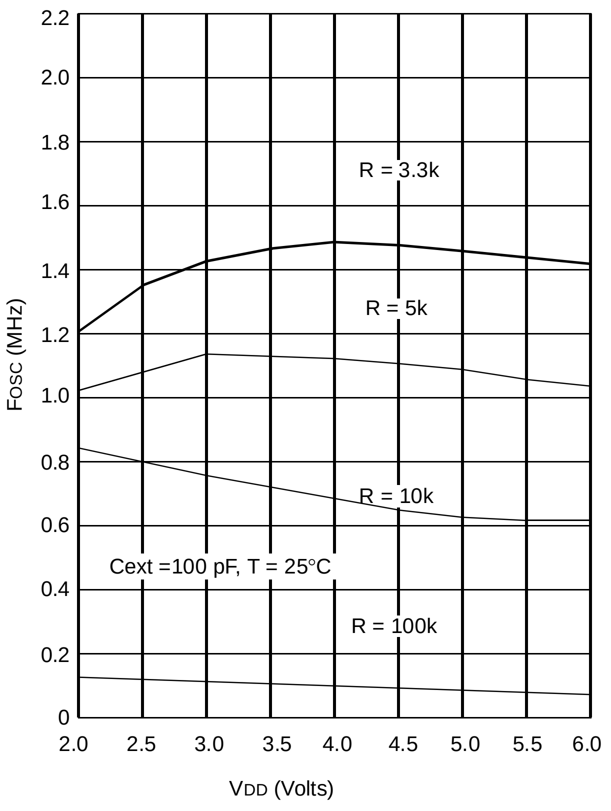


FIGURE 12-4: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD*

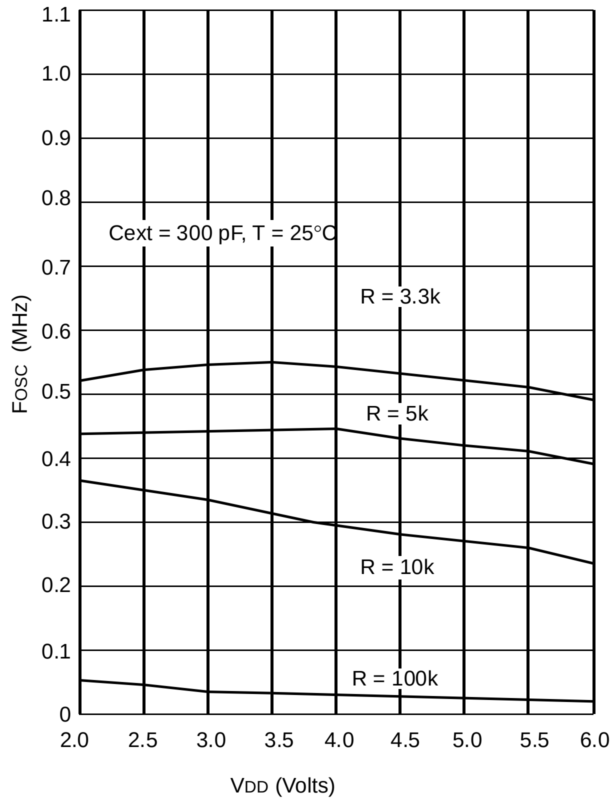


FIGURE 12-5: TYPICAL IPD vs. VDD WATCHDOG DISABLED (25°C)

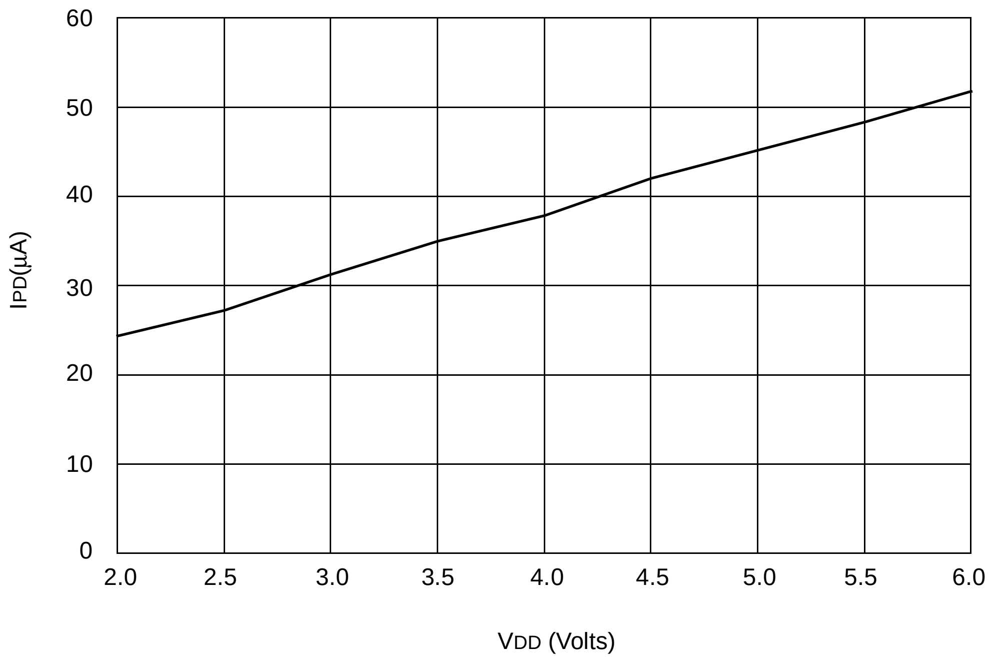


FIGURE 12-6: TYPICAL IPD vs. VDD WATCHDOG ENABLED (25°C)

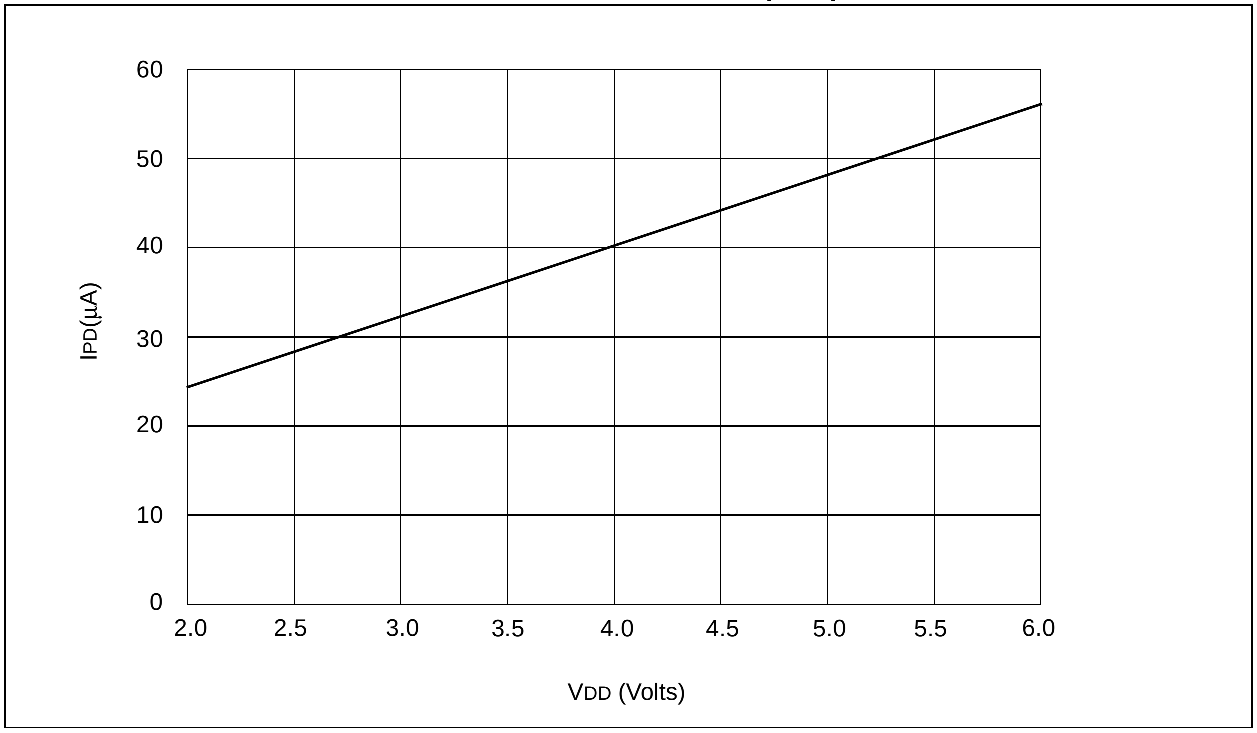


FIGURE 12-7: MAXIMUM IPD vs. VDD WATCHDOG DISABLED

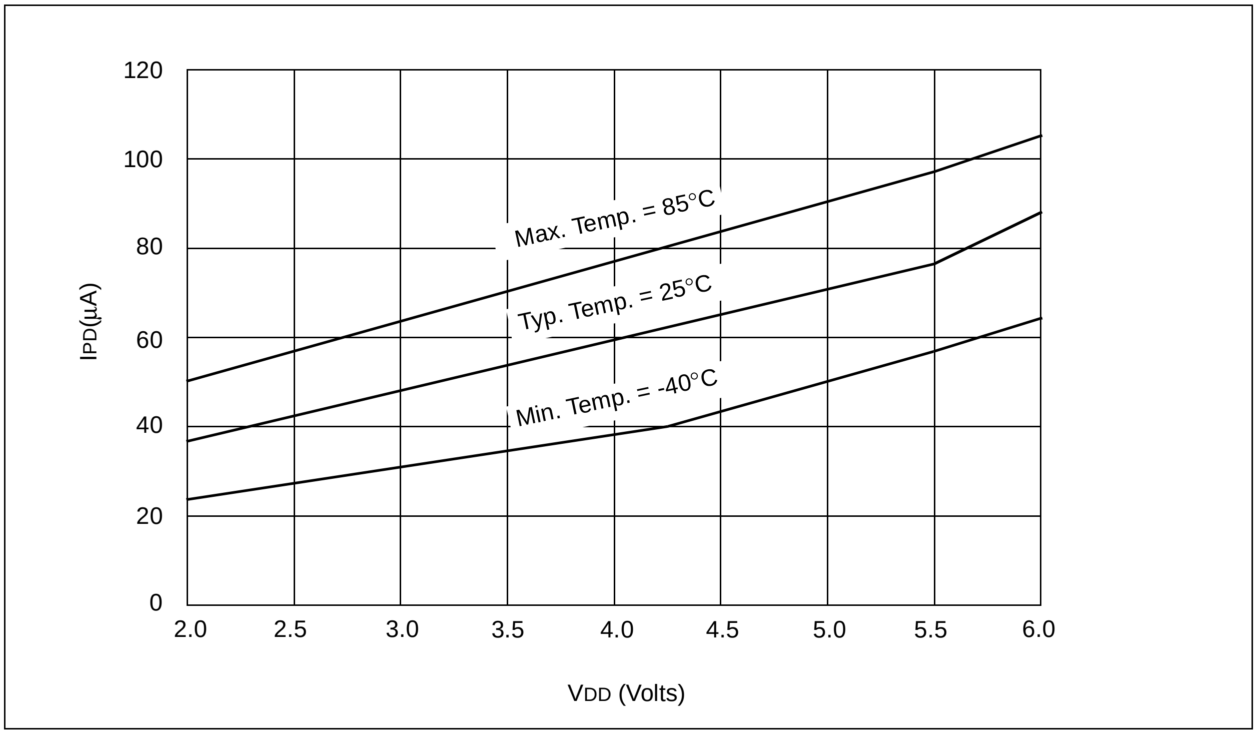
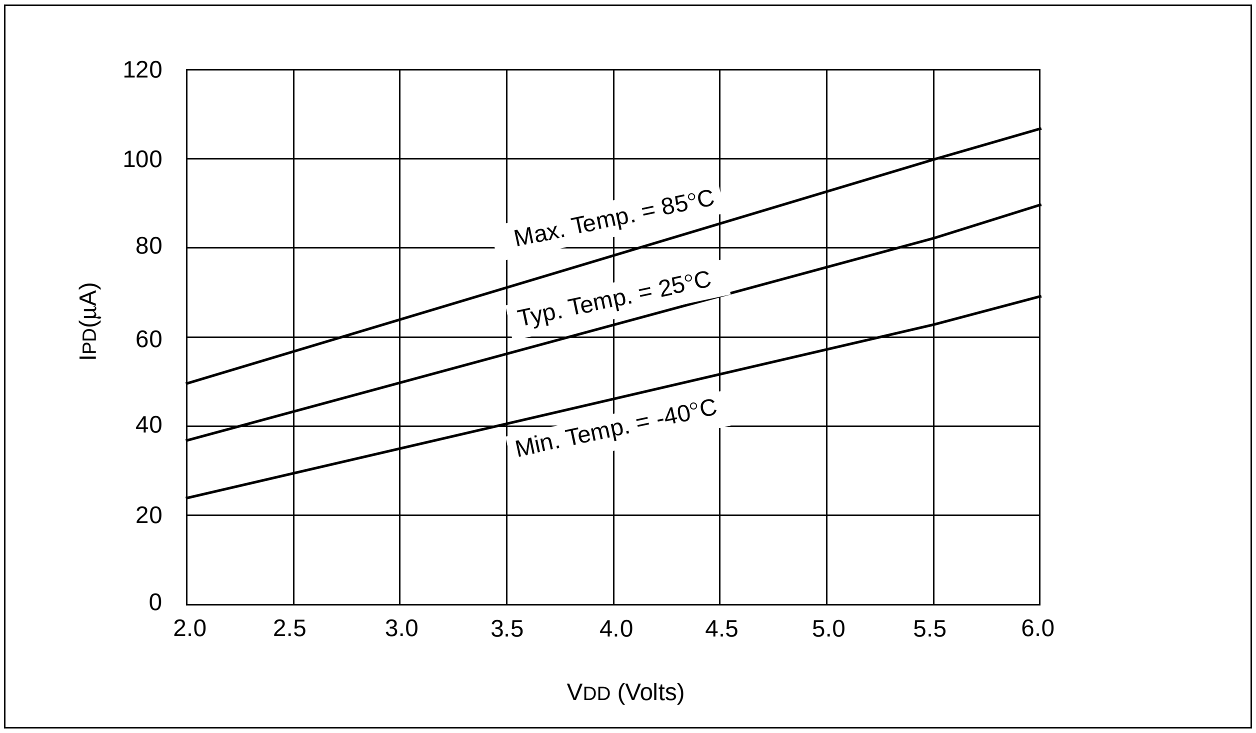


FIGURE 12-8: MAXIMUM IPD vs. VDD WATCHDOG ENABLED*



* IPD, with watchdog timer enabled, has two components: The leakage current which increases with higher temperature and the operating current of the watchdog timer logic which increases with lower temperature. At -40°C, the latter dominates explaining the apparently anomalous behavior.

FIGURE 12-9: VTH (INPUT THRESHOLD VOLTAGE) OF I/O PINS vs. VDD

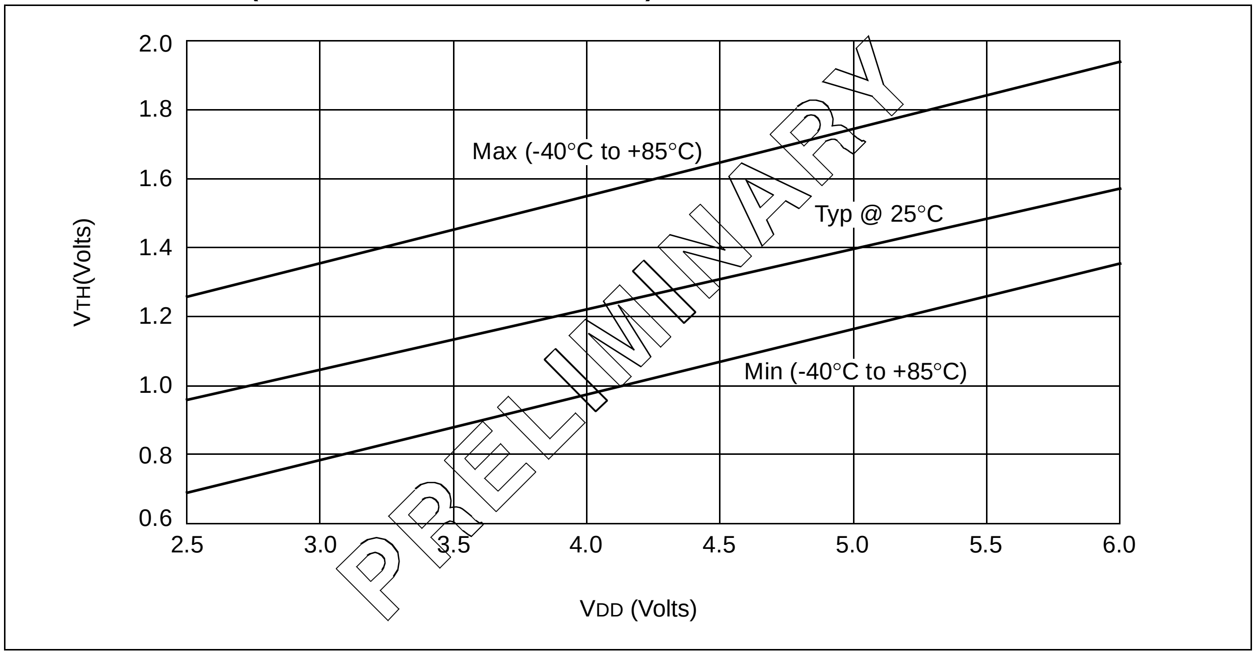


FIGURE 12-10: VTH (INPUT THRESHOLD VOLTAGE) OF OSC1 INPUT (IN XT, HS, AND LP MODES) vs. VDD

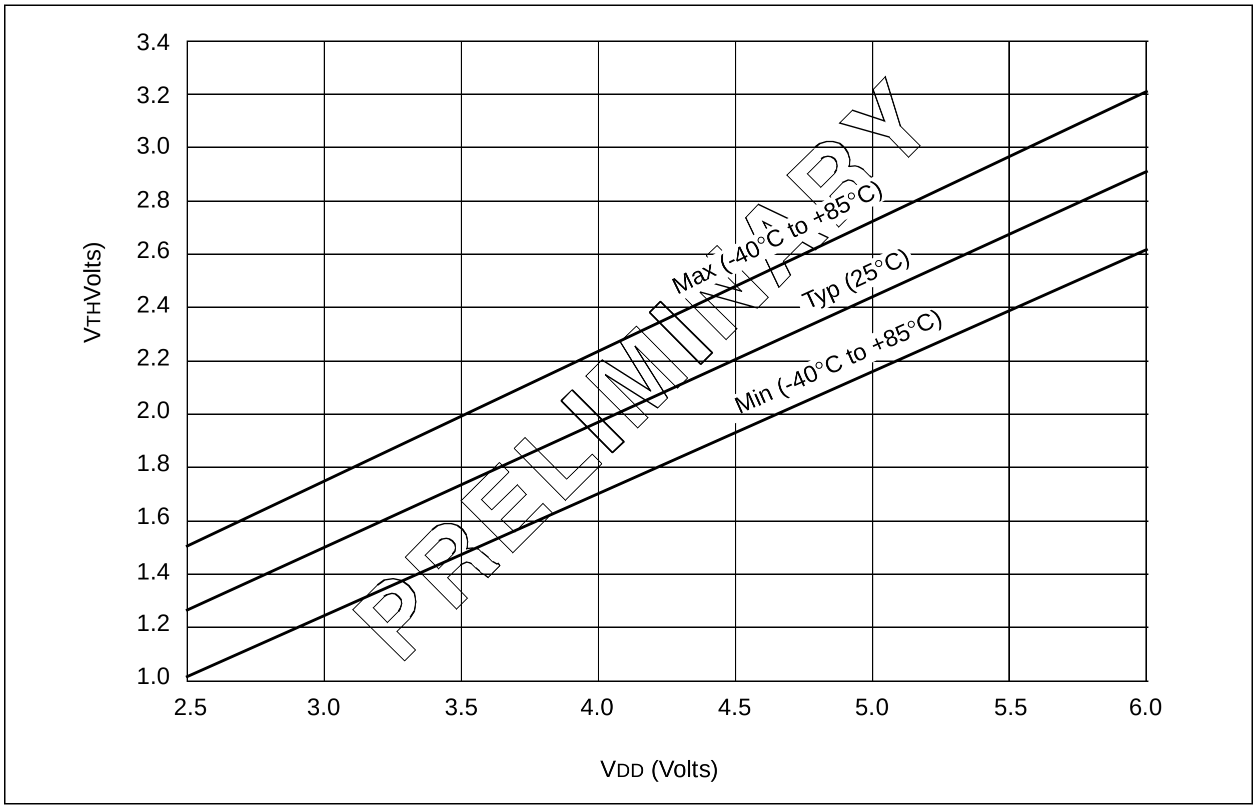


FIGURE 12-11: VIH, VIL OF $\overline{\text{MCLR}}$, T0CKI and OSC1 (IN RC MODE) vs. VDD

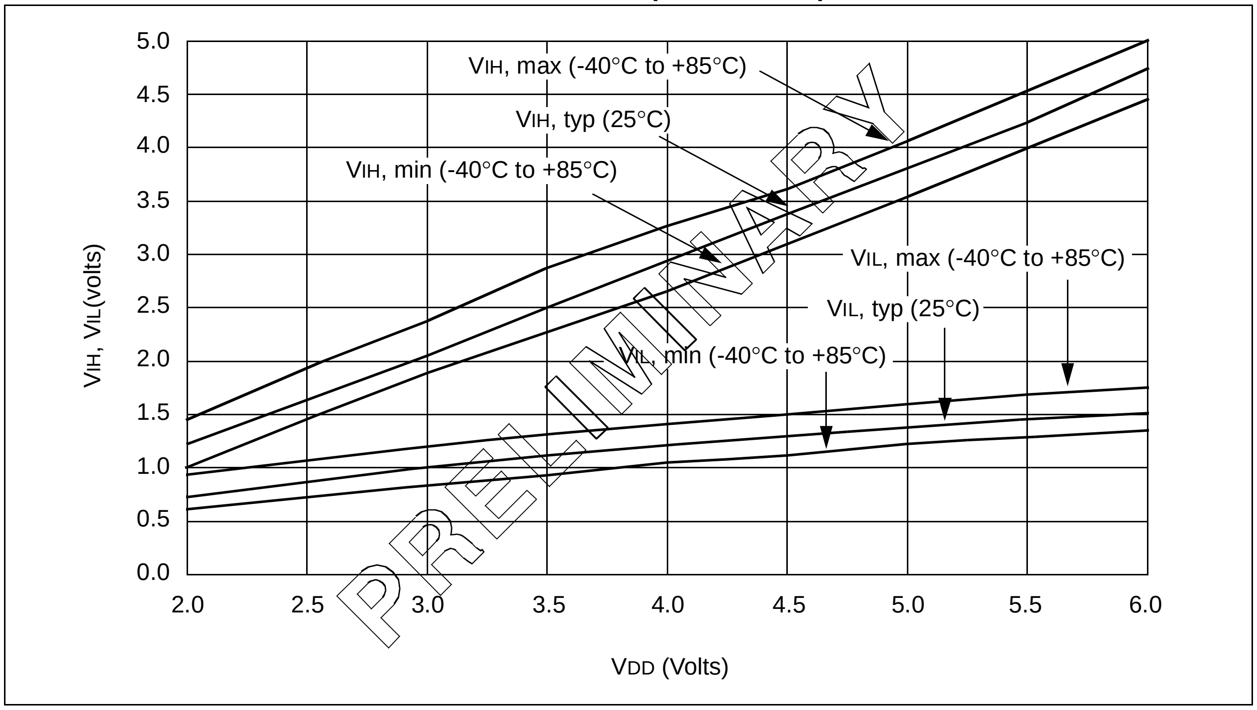


FIGURE 12-12: TYPICAL IDD vs. FREQ (EXT CLOCK, 25°C)

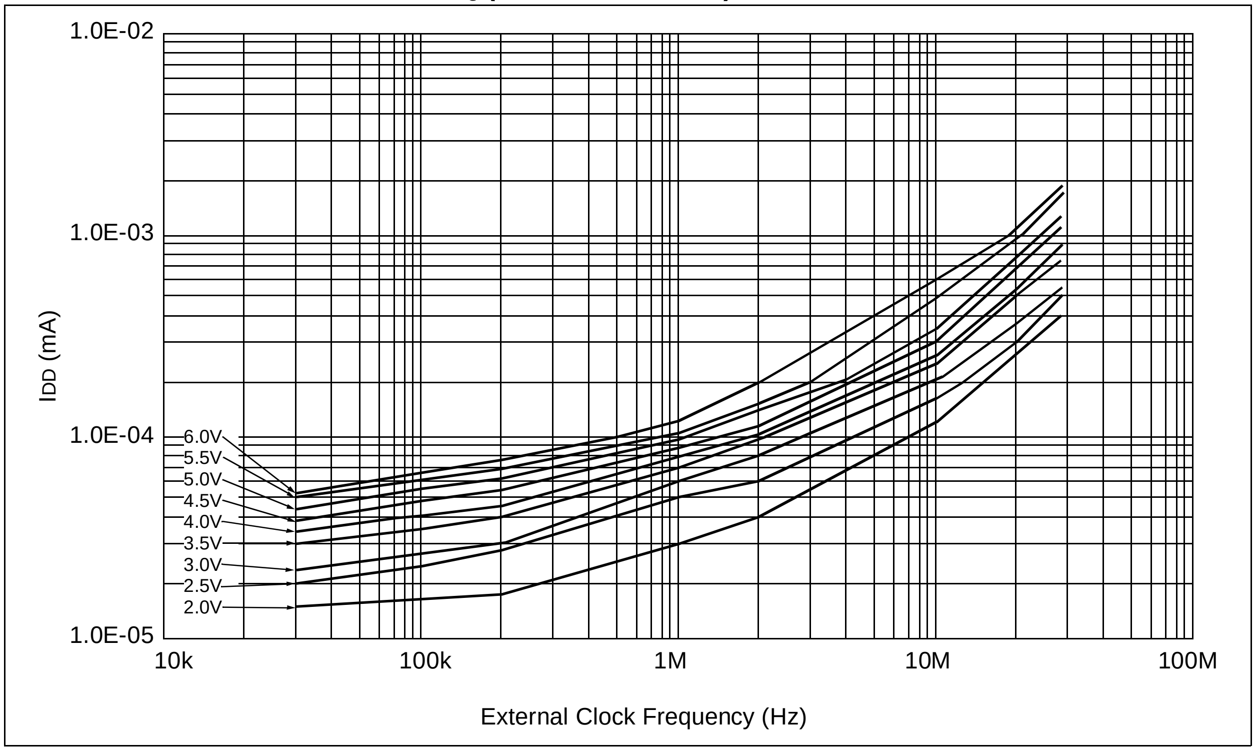


FIGURE 12-13: MAXIMUM IDD vs. FREQ (EXT CLOCK, -40° TO +85°C)

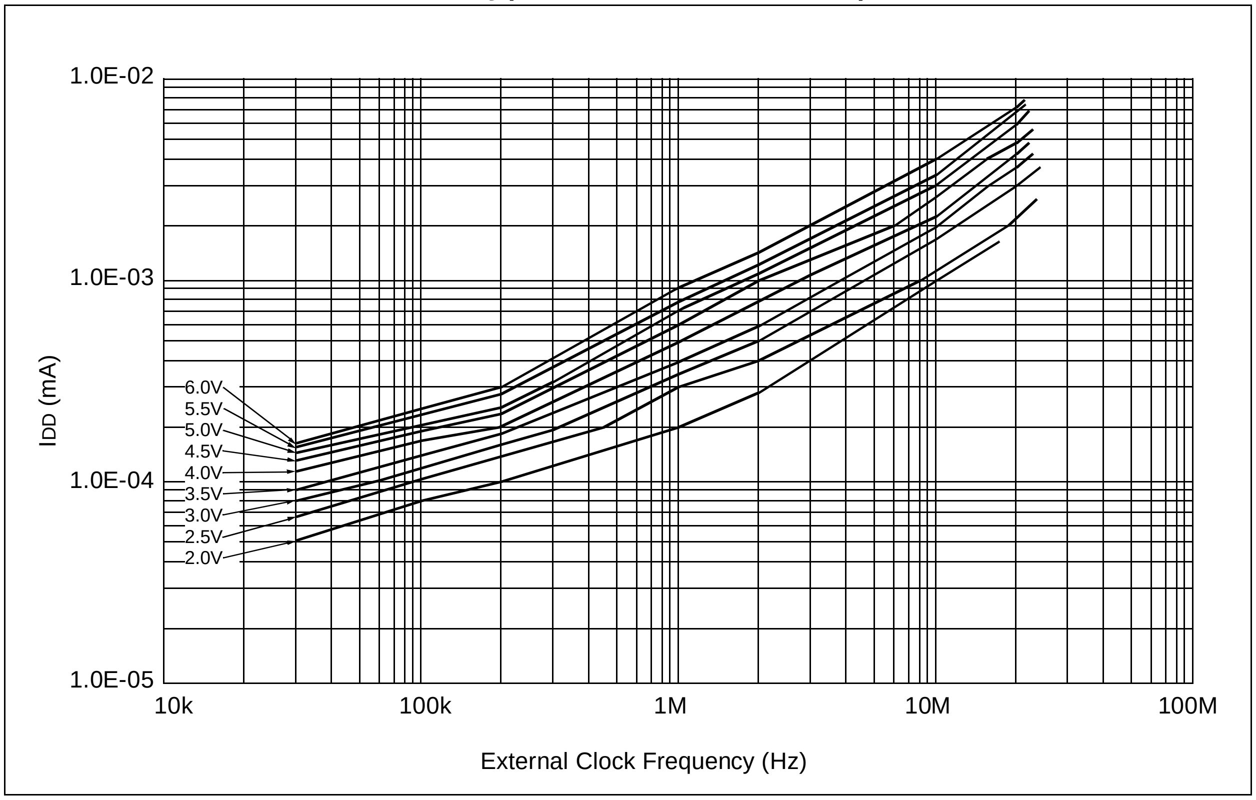


FIGURE 12-14: WDT TIMER TIME-OUT PERIOD vs. VDD

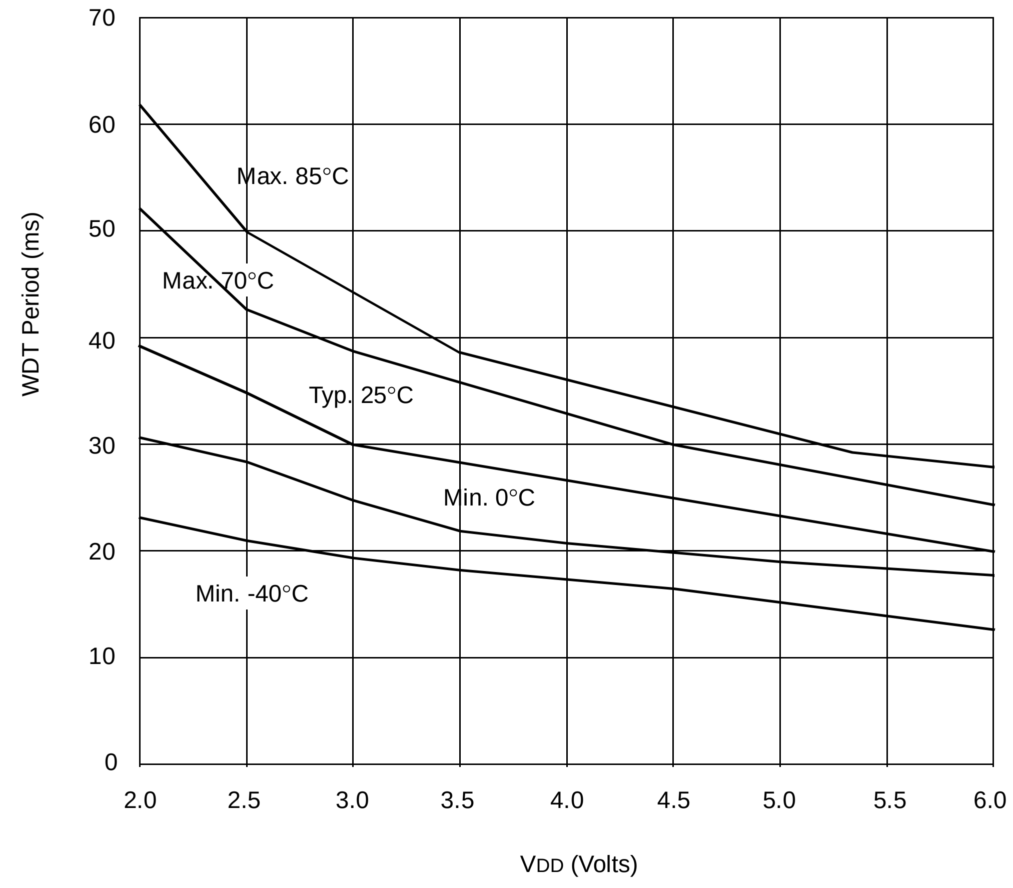


FIGURE 12-15: TRANSCONDUCTANCE (gm) OF HS OSCILLATOR vs. VDD

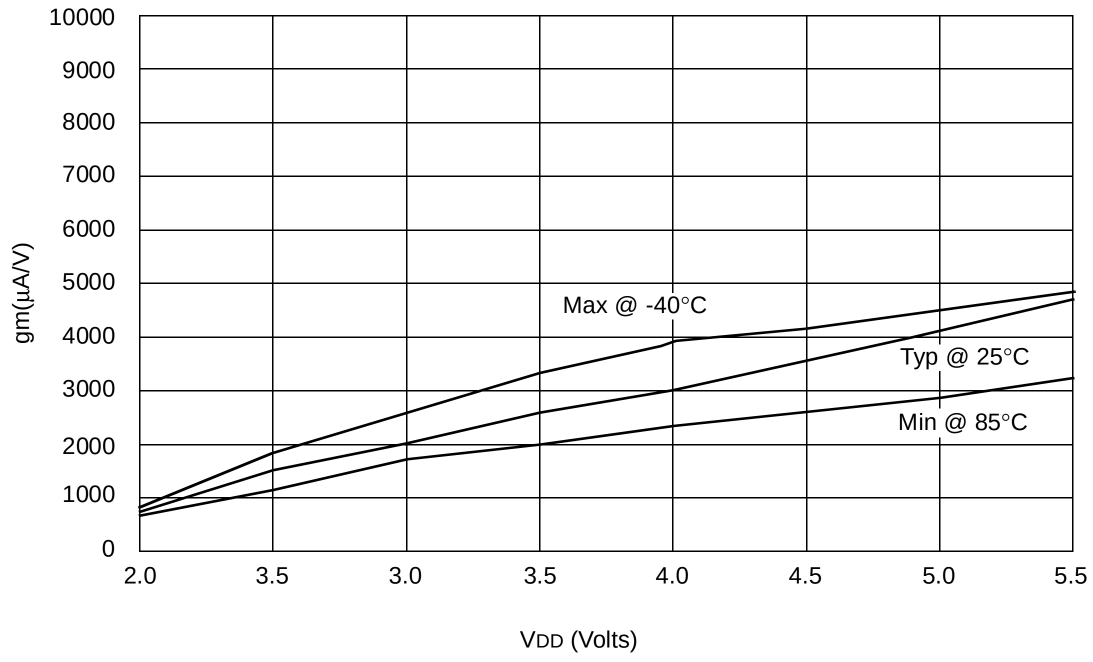


FIGURE 12-16: TRANSCONDUCTANCE (gm) OF LP OSCILLATOR vs. VDD

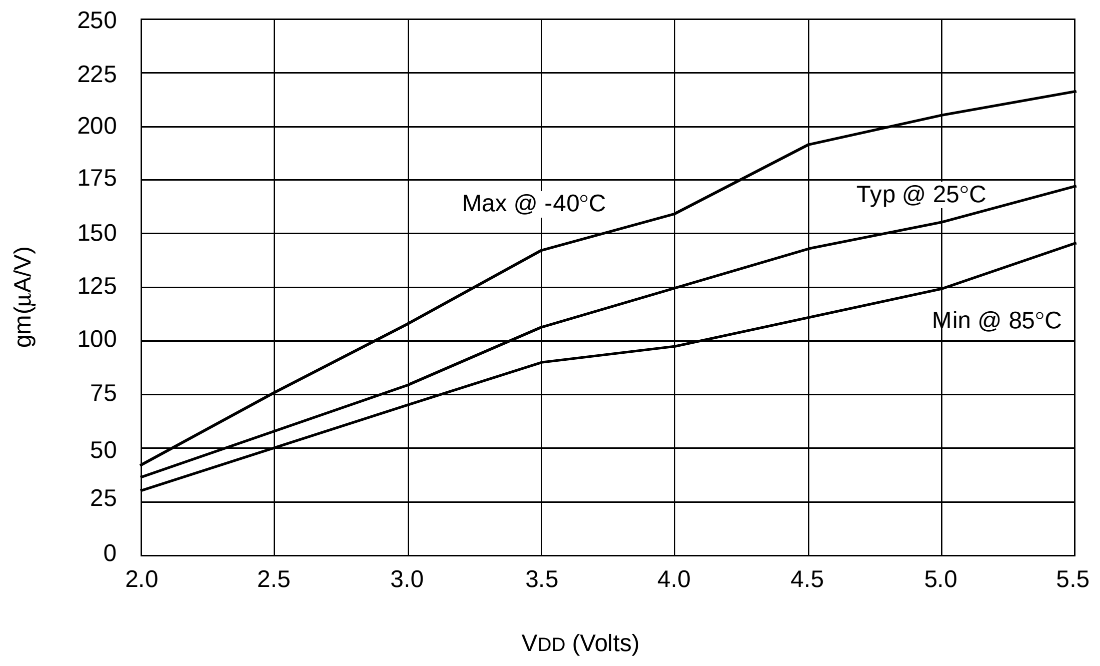


FIGURE 12-17: TRANSCONDUCTANCE (gm) OF XT OSCILLATOR vs. VDD

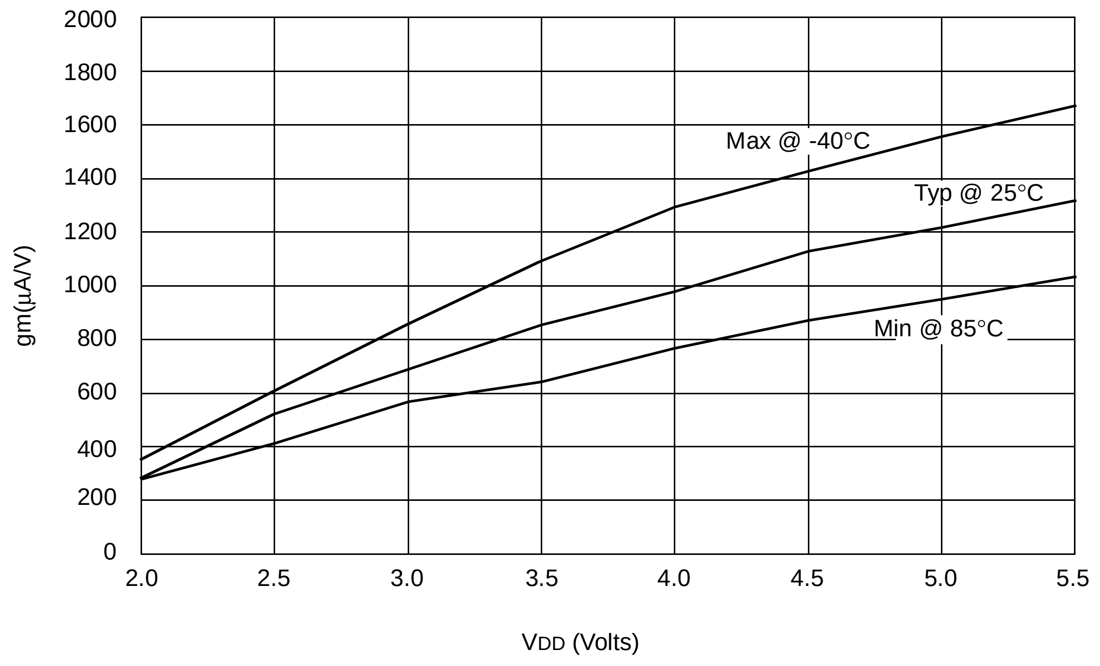


FIGURE 12-18: IOH vs. VOH, VDD = 3V

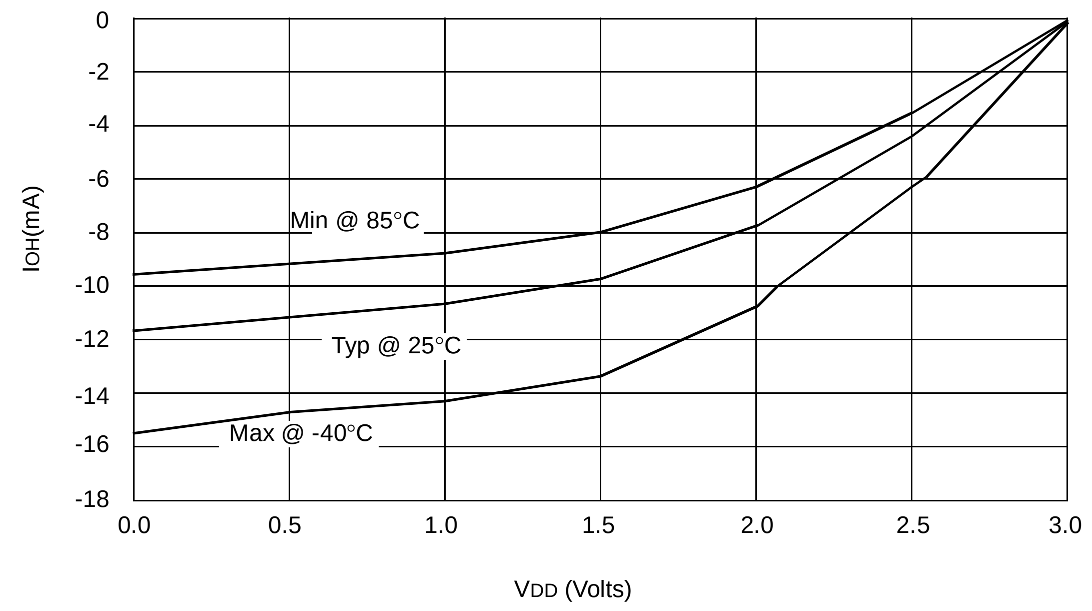


FIGURE 12-19: IOH vs. VOH, VDD = 5V

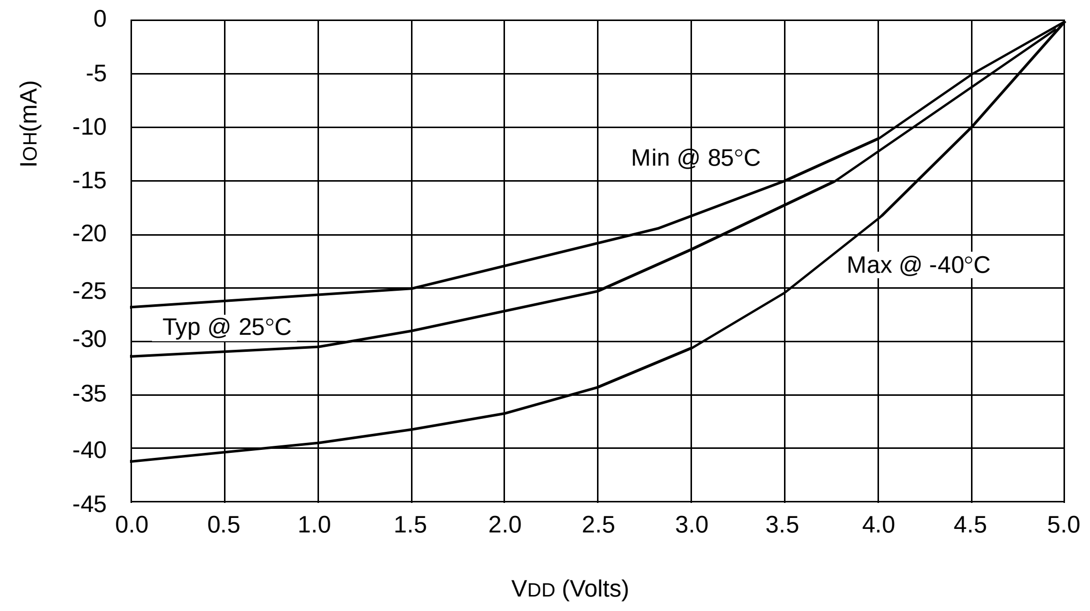


FIGURE 12-20: IOL vs. VOL, VDD = 3V

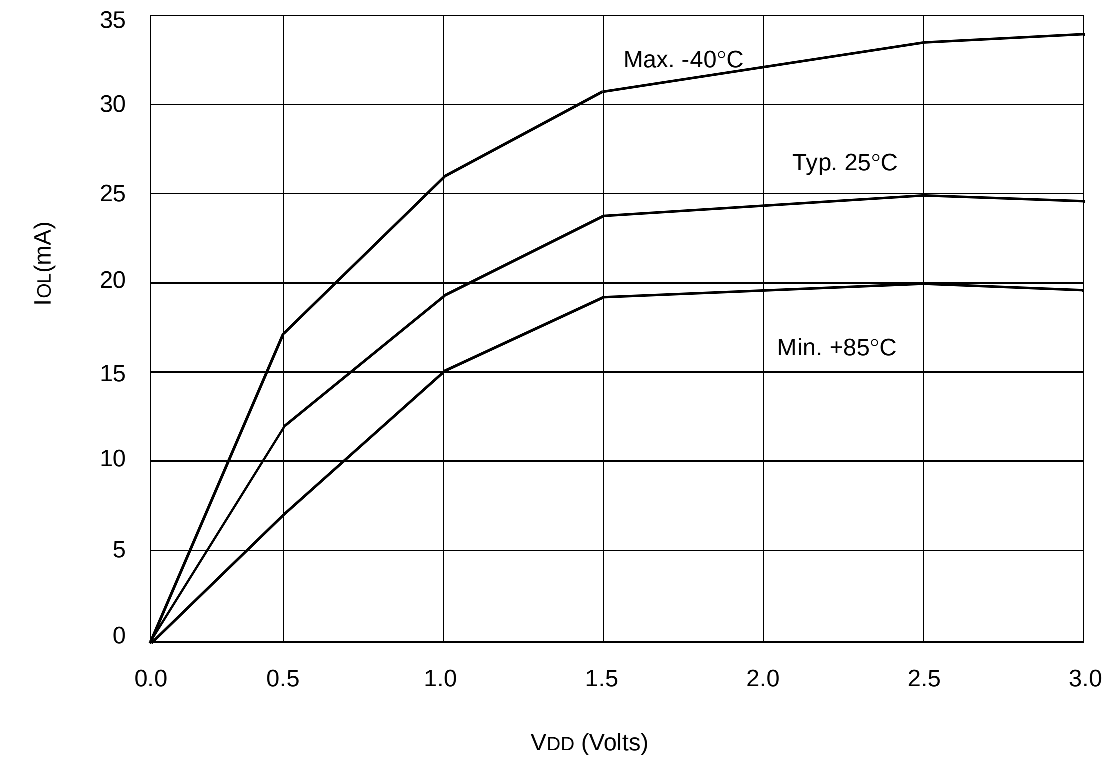


FIGURE 12-21: IOL vs. VOL, VDD = 5V

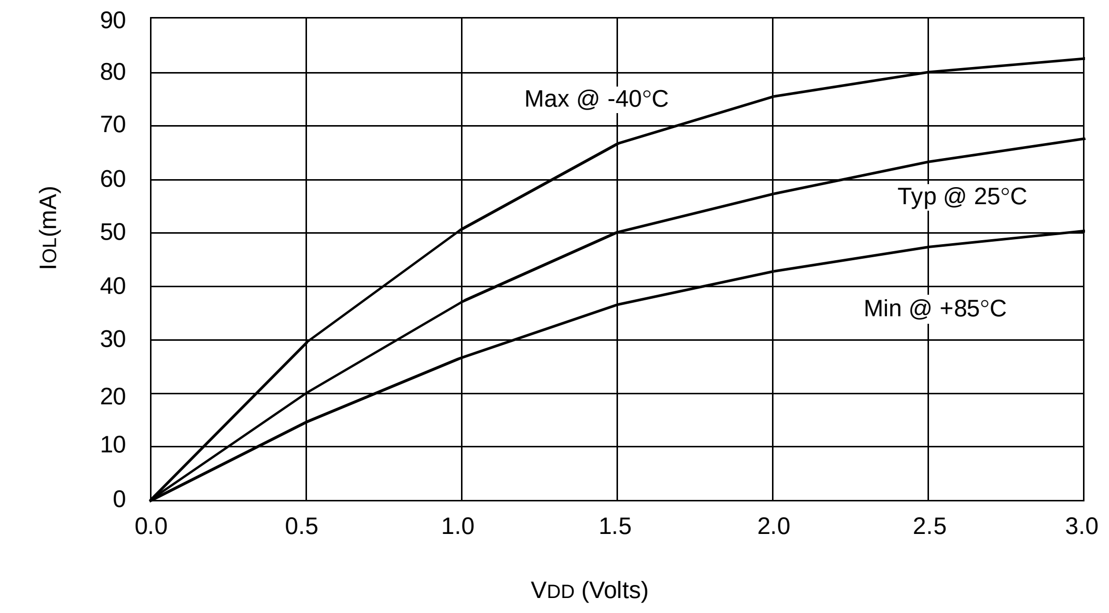


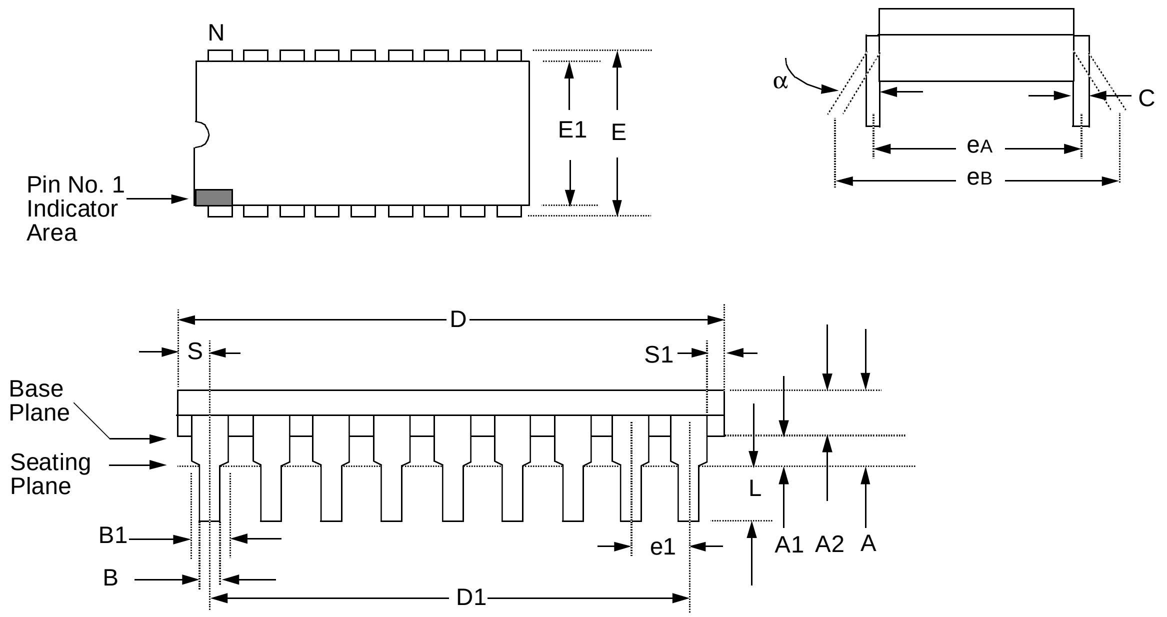
TABLE 12-2: INPUT CAPACITANCE *

Pin Name	Typical Capacitance (pF)	
	18L PDIP	18L SOIC
PORTA	5.0	4.3
PORTB	5.0	4.3
$\overline{\text{MCLR}}$	17.0	17.0
OSC1/CLKIN	4.0	3.5
OSC2/CLKOUT	4.3	3.5
T0CKI	3.2	2.8

* All capacitance values are typical at 25°C. A part to part variation of ±25% (three standard deviations) should be taken into account.

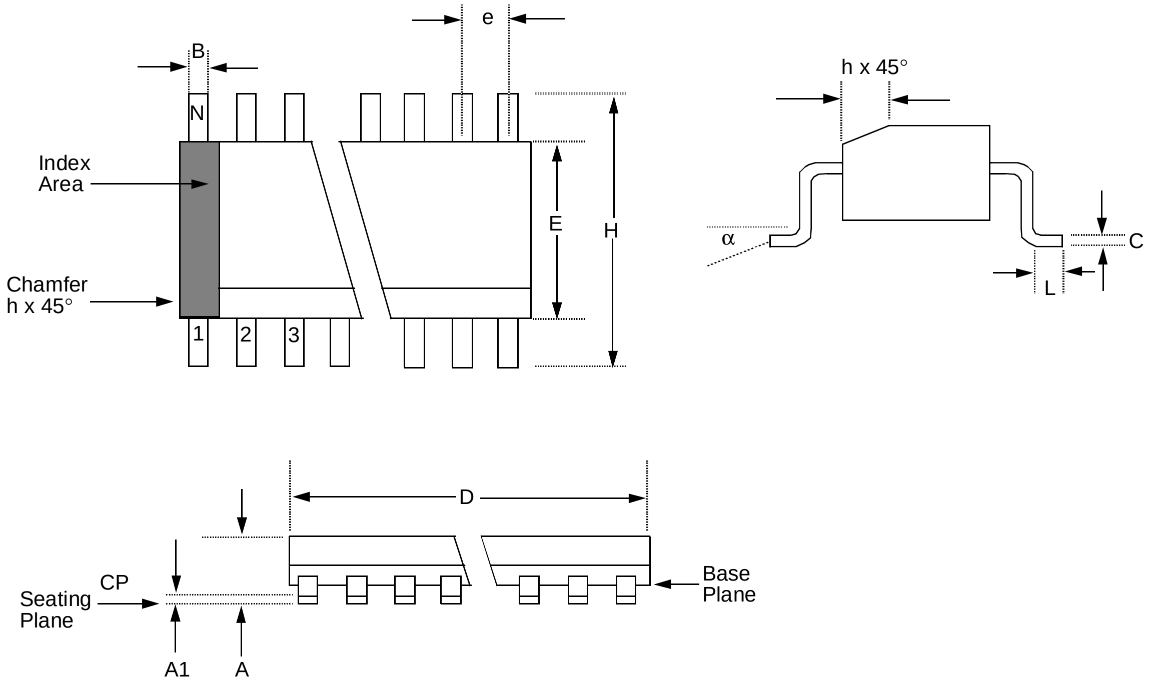
13.0 PACKAGING INFORMATION

13.1 18-Lead Plastic Dual In-line (300 mil)



Package Group: Plastic Dual In-Line (PLA)						
	Millimeters			Inches		
Symbol	Min	Max	Notes	Min	Max	Notes
α	0°	10°		0°	10°	
A	–	4.064		–	0.160	
A1	0.381	–		0.015	–	
A2	3.048	3.810		0.120	0.150	
B	0.355	0.559		0.014	0.022	
B1	1.524	1.524	**Reference**	0.060	0.060	**Reference**
C	0.203	0.381	**Typical**	0.008	0.015	**Typical**
D	22.479	23.495		0.885	0.925	
D1	20.320	20.320	**Reference**	0.800	0.800	**Reference**
E	7.620	8.255		0.300	0.325	
E1	6.096	7.112		0.240	0.280	
e1	2.489	2.591	**Typical**	0.098	0.102	**Typical**
eA	7.620	7.620	**Reference**	0.300	0.300	**Reference**
eB	7.874	9.906		0.310	0.390	
L	3.048	3.556		0.120	0.140	
N	18	18		18	18	
S	0.889	–		0.035	–	
S1	0.127	–		0.005	–	

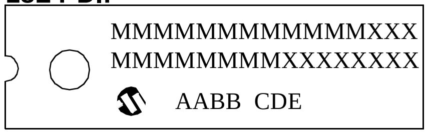
13.2 18-Lead Plastic Surface Mount (SOIC - Wide, 300 mil Body)



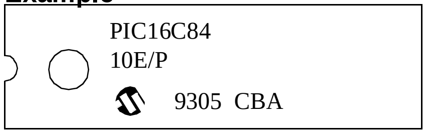
Package Group: Plastic SOIC (SO)						
	Millimeters			Inches		
Symbol	Min	Max	Notes	Min	Max	Notes
α	0°	8°		0°	8°	
A	2.362	2.642		0.093	0.104	
A1	0.101	0.300		0.004	0.012	
B	0.355	0.483		0.014	0.019	
C	0.241	0.318		0.009	0.013	
D	11.353	11.735		0.447	0.462	
E	7.416	7.595		0.292	0.299	
e	1.270	1.270	**Reference**	0.050	0.050	**Reference**
H	10.007	10.643		0.394	0.419	
h	0.381	0.762		0.015	0.030	
L	0.406	1.143		0.016	0.045	
N	18	18		18	18	
CP	–	0.102		–	0.004	

13.3 Package Marking Information

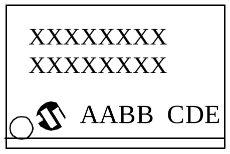
18L PDIP



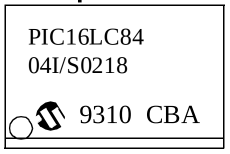
Example



18L SOIC



Example



Legend:	MM...M	Microchip part number information
	XX...X	Customer specific information*
	AA	Year code (last 2 digits of calendar year)
	BB	Week code (week of January 1 is week '01')

	C	Facility code of the plant at which wafer is manufactured
		C = Chandler, Arizona, U.S.A.,
		S = Tempe, Arizona, U.S.A.

	D	Mask revision number
	E	Assembly code of the plant or country of origin in which part was assembled

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

* Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

NOTES:

APPENDIX A: CHANGES

The following is the list of modifications over the PIC16C5X microcontroller family:

1. Instruction word length is increased to 14 bits. This allows larger page sizes both in program memory (2K now as opposed to 512 before) and the register file (128 bytes now versus 32 bytes before).
2. A PC latch register (PCLATH) is added to handle program memory paging. PA2, PA1 and PA0 bits are removed from the status register and placed in the option register.
3. Data memory paging is redefined slightly. The status register is modified.
4. Four new instructions have been added: `RETURN`, `RETFIE`, `ADDLW`, and `SUBLW`.
   Two instructions, `TRIS` and `OPTION`, are being phased out although they are kept for compatibility with PIC16C5X.
5. OPTION and TRIS registers are made addressable.
6. Interrupt capability is added. Interrupt vector is at 0004h.
7. Stack size is increased to 8 deep.
8. Reset vector is changed to 0000h.
9. Reset of all registers is revisited. Five different reset (and wake-up) types are recognized. Registers are reset differently.
10. Wake up from SLEEP through interrupt is added.
11. Two separate timers, the Oscillator Start-Up Timer (OST) and Power-Up Timer (PWRT), are included for more reliable power-up. These timers are invoked selectively to avoid unnecessary delays on power-up and wake-up.
12. PORTB has weak pull-ups and interrupt on change features.
13. T0CKI pin is also a port pin (RA4/T0CKI).
14. FSR is a full 8-bit register.
15. "In system programming" is made possible. The user can program PIC16CXX devices using only five pins: V_{DD}, V_{SS}, V_{PP}, RB6 (clock) and RB7 (data in/out).

APPENDIX B: COMPATIBILITY

To convert code written for PIC16C5X to PIC16C84, the user should take the following steps:

1. Remove any program memory page select operations (PA2, PA1, PA0 bits) for `CALL`, `GOTO`.
2. Revisit any computed jump operations (write to PC or add to PC, etc.) to make sure page bits are set properly under the new scheme.
3. Eliminate any data memory page switching. Redefine data variables for reallocation.
4. Verify all writes to STATUS, OPTION, and FSR registers since these have changed.
5. Change reset vector to 0000h.

APPENDIX C: WHAT'S NEW

The conversion of this Data Sheet into the desktop publishing software package, The structure of the document has been made consistent with other data sheet. This ensures that important topics are covered across all PIC16/17 families. Here is an overview list of new features:

- Data Sheet Structure / Outline

APPENDIX D: WHAT'S CHANGED

To make software more portable across the different PIC16/17 families, some of the registers and control bits have been changed. Now control bits that do the same function, have the same name (regardless of processor family). Care must still be taken, since they may not be in the same special function register. The following lists the register and bit names that have changed:

TABLE 13-1: BIT NAME CHANGES

OLD NAME	NEW NAME
RTS	T0CS
RTE	T0SE

APPENDIX E: PIC16/17 MICROCONTROLLERS

TABLE E-1: PIC17CXX FAMILY OF DEVICES

	Clock	Memory		Peripherals				Features					
	Maximum Frequency of Operation (MHz)	Program Memory EPROM	RAM Data Memory (bytes)	Timer Module(s)	Captures	PWMs	Serial Port(s) (SCI)	External Interrupts	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages
PIC17C42	25	2K	232	TMR0,TMR1, TMR2,TMR3	2	2	Yes	Yes	11	33	4.5-5.5	55	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC17C43*	25	4K	454	TMR0,TMR1, TMR2,TMR3	2	2	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC17C44	25	8K	454	TMR0,TMR1, TMR2,TMR3	2	2	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP, 44-pin PLCC, 44-pin QFP

* Please contact your local sales office for availability of these devices.

Note 1: All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

2: The PIC17C4X devices can also operate in microprocessor and external microcontroller modes.

3: PORTB has software-configurable weak pull-ups.

TABLE E-2: PIC16CXX FAMILY OF DEVICES

	Clock	Memory				Peripherals							Features				
	Maximum Frequency of Operation (MHz)	Program Memory EPROM	Program Memory EEPROM	Data Memory (bytes)	Data EEPROM (bytes)	Timer Module(s)	Capture/Compare/PWM Module(s)	Serial Port(s) (SPI/I^{2}C, SCI)	Parallel Slave Port	Analog to Digital Converter (8-bit)	Comparator(s)	Internal Reference Voltage	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Brown-out	Packages
PIC16C61	20	1K	—	36	—	TMR0	—	—	—	—	—	—	3	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C62*	20	2K	—	128	—	TMR0, TMR1, TMR2	2	SPI/I^{2}C	—	—	—	—	10	22	2.5-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C63*	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I^{2}C/ SCI	—	—	—	—	10	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C64	20	2K	—	128	—	TMR0, TMR1, TMR2	1	SPI/I^{2}C	Yes	—	—	—	8	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C65	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I^{2}C/ SCI	Yes	—	—	—	11	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C620*	20	512	—	80	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C621*	20	1K	—	80	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C622	20	2K	—	128	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C71	20	1K	—	36	—	TMR0	—	—	—	4 ch	—	—	4	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C73	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I^{2}C/ SCI	—	5 ch	—	—	11	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C74	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I^{2}C/ SCI	Yes	8 ch	—	—	12	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C84	10	—	1K	36	64	TMR0	—	—	—	—	—	—	4	13	2.0-6.0	—	18-pin DIP, 18-pin SOIC

* Please contact your local sales office for availability of these devices.

Note 1: All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

2: The PIC16CXX Timer1 has its own oscillator circuit and can operate asynchronously to the device. Timer1 can increment while the device is in SLEEP mode. This allows a Real Time Clock to be implemented.

3: PORTB has software-configurable weak pull-ups.

TABLE E-3: PIC16C5X FAMILY OF DEVICES

	Clock	Memory			Peripherals		Features		
		Program Memory (words)							
	Maximum Frequency of Operation (MHz)	EPROM	ROM	RAM Data Memory (bytes)	Timer Module(s)	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages
PIC16C54	20	512	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16C54A	20	512	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16CR54	20	—	512	25	TMR0	12	2.0-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16C55	20	512	—	25	TMR0	20	2.5-6.25	33	28-pin DIP, 28-pin SOIC, 28 pin SSOP
PIC16C56	20	1K	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C57	20	2K	—	72	TMR0	20	2.5-6.25	33	28-pin DIP, 28-pin SOIC, 28 pin SSOP
PIC16CR57A	20	—	2K	72	TMR0	20	2.0-6.25	33	28-pin DIP, 28-pin SOIC, 28 pin SSOP
PIC16C58A	20	2K	—	73	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16CR58A	20	—	2K	73	TMR0	12	2.0-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP

Note: All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

E.1 Pin Compatibility

Devices that have the same package type; and VDD, VSS, and $\overline{\text{MCLR}}$ pin locations, are said to be pin compatible. This allows these different devices to operate in the same socket. Compatible devices may only requires minor software modification to allow proper operation in the application socket (ex., PIC16C56 and PIC16C61 devices). Not all devices in the same package size are pin compatible; for example, the PIC16C62 is compatible with the PIC16C63, but not the PIC16C55.

Pin compatibility does not mean that the devices offer the same features. As an example, the PIC16C54 is pin compatible with the PIC16C71, but does not have an A/D converter, weak pull-ups on PORTB, or interrupts.

TABLE E-4: PIN COMPATIBILE DEVICES

Pin Compatible Devices	Package
PIC16C61, PIC16C620, PIC16C621, PIC16C622, PIC16C71, PIC16C84, PIC16C54, PIC16C54A, PIC16CR54, PIC16C56, PIC16C58A, PIC16CR58A	18 pin 20 pin
PIC16C62, PIC16C63, PIC16C73	28 pin
PIC16C55, PIC16C57, PIC16CR57A	28 pin
PIC17C42, PIC17C43, PIC17C44	40 pin
PIC16C64, PIC16C65, PIC16C74	40 pin

INDEX

LIST OF EXAMPLES

LIST OF FIGURES

LIST OF TABLES

CONNECTING TO MICROCHIP BBS

Connect worldwide to the Microchip BBS using the CompuServe® communications network. In most cases a local call is your only expense. The Microchip BBS connection does not use CompuServe membership services, therefore **you do not need CompuServe membership to join Microchip's BBS**.

There is **no charge** for connecting to the BBS, except for a toll charge to the CompuServe access number, where applicable. You do not need to be a CompuServe member to take advantage of this connection (you never actually log in to CompuServe).

The procedure to connect will vary slightly from country to country. Please check with your local CompuServe agent for details if you have a problem. CompuServe service allows multiple users at baud rates up to 14400 bps.

The following connect procedure applies in most locations:

1. Set your modem to 8 bit, No parity, and One stop (8N1). This is not the normal CompuServe setting which is 7E1.
2. Dial your local CompuServe access number.
3. Depress **<ENTER↵>** and a garbage string will appear because CompuServe is expecting a 7E1 setting.
4. Type **+,** depress **<ENTER↵>** and `Host Name:` will appear.
5. Type **MCHIPBBS,** depress **< ENTER↵ >** and you will be connected to the Microchip BBS.

In the United States, to find CompuServe's phone number closest to you, set your modem to 7E1 and dial (800) 848-4480 for 300-2400 baud or (800) 331-7166 for 9600-14400 baud connection. After the system responds with `Host Name:`, type

NETWORK, depress **< ENTER↵ >**
and follow CompuServe's directions.

For voice information (or calling from overseas), you may call (614) 457-1550 for your local CompuServe number.

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 (602) 786-7578.

Please list the following information, and use this outline to provide us with your comments about this Data Sheet.

To: Technical Publications Manager   Total Pages Sent ________

RE: Reader Response

From: Name ______________________________

Company ______________________________

Address ______________________________

City / State / ZIP / Country ______________________________

Telephone: (_______) _________ - _________   FAX: (______) _________ - _________

Application (optional):

Would you like a reply?____Y ____N

Device: PIC16C84   Literature Number: DS30081E

Questions:

1. What are the best features of this document? ______________________________

2. How does this document meet your hardware and software development needs? ______________________________

3. Do you find the organization of this data sheet easy to follow? If not, why? ______________________________

4. What additions to the data sheet do you think would enhance the structure and subject? ______________________________

5. What deletions from the data sheet could be made without affecting the overall usefullness? ______________________________

6. Is there any incorrect or misleading information (what and where)? ______________________________

7. How would you improve this document? ______________________________

8. How would you improve our software, systems, and silicon products? ______________________________

PIC16C84 Product Identification System

To order or obtain information, e.g., on pricing or delivery refer to the factory or the listed sales office.

PART NO. -XX X /XX XXX

Pattern:	QTP, SQTP, ROM Code or Special Requirements		
Package:	P	=	PDIP
	SO	=	SOIC (Gull Wing, 300 mil body)
Temperature Range:	-	=	0°C to +70°C
	I	=	-40°C to +85°C
Frequency Range:	04	=	4 MHz -RC, -XT / 200 kHz -LP
	10	=	10 MHz
Device	PIC16C84		:VDD range 4.0V to 6.0V
	PIC16C84T		:Tape and Reel
	PIC16LC84		:VDD range 2.0V to 6.0V
	PIC16LC84T		:Tape and Reel

Examples

a) PIC16C84 - 04/P 301 Commercial Temp., PDIP Package, 4 MHz, normal VDD limits, QTP pattern #301

b) PIC16LC84 - 04I/SO Industrial Temp., SOIC package, 200 kHz, extended VDD limits

c) PIC16C84 - 10I/P Industrial Temp., PDIP package, 10 MHz, normal VDD limits

Sales and Support

Products supported by a preliminary Data Sheet may possibly have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office (see below)
2. The Microchip Corporate Literature Center U.S. FAX: (602) 786-7277
3. The Microchip's Bulletin Board, via your local CompuServe number (CompuServe membership NOT required).

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.
For latest version information and upgrade kits for Microchip Development Tools, please call 1-800-755-2345 or 1-602-786-7302.

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 404 640-0034 Fax: 404 640-0307

Boston
Microchip Technology Inc.
Five The Mountain Road, Suite 120
Framingham, MA 01701
Tel: 508 820-3334 Fax: 508 820-4326

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408 436-7950 Fax: 408 436-7955

ASIA/PACIFIC

Hong Kong
Microchip Technology Inc.
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku,
Seoul, Korea
Tel: 82 2 554 7200 Fax: 82 2 558 5934

Singapore
Microchip Technology Inc.
200 Middle Road
#10-03 Prime Centre
Singapore 0718
Tel (mobile): 65 634 2305

Taiwan
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire
SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

Germany
Arizona MicrochipTechnology GmbH
Gustav-Heinemann-Ring 125
D-81739 Muenchen, Germany
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

MICROCHIP

Printed in the USA, 5/95
© 1995, Microchip Technology Inc.