What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	10MHz
Connectivity	-
Peripherals	POR, WDT
Number of I/O	5
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 4x8b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.209", 5.30mm Width)
Supplier Device Package	8-SOIJ
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic12c672-10-sm

3.0   ARCHITECTURAL OVERVIEW

The high performance of the PIC12C67X family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC12C67X uses a Harvard architecture, in which program and data are accessed from separate memories using separate buses. This improves bandwidth over traditional von Neumann architecture in which program and data are fetched from the same memory using the same bus. Separating program and data buses also allow 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 instruction cycle. A two-stage pipeline overlaps fetch and execution of instructions (Example 3-1). Consequently, all instructions (35) execute in a single cycle (200 ns @ 20 MHz) except for program branches.

The table below lists program memory (EPROM), data memory (RAM), and non-volatile memory (EEPROM) for each PIC12C67X device.

Device	Program Memory	RAM Data Memory	EEPROM Data Memory
PIC12C671	1K x 14	128 x 8	—
PIC12C672	2K x 14	128 x 8	—
PIC12CE673	1K x 14	128 x 8	16x8
PIC12CE674	2K x 14	128 x 8	16x8

The PIC12C67X 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. The PIC12C67X has 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 PIC12C67X simple yet efficient. In addition, the learning curve is reduced significantly.

PIC12C67X devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between the 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). 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 $\overline{\text{borrow}}$ bit and a $\overline{\text{digit borrow}}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

PIC12C67X

FIGURE 4-2: **PIC12C67X REGISTER FILE MAP**

File Address			File Address
00h	INDF[1]	INDF[1]	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	GPIO	TRIS	85h
06h			86h
07h			87h
08h			88h
09h			89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	PIR1	PIE1	8Ch
0Dh			8Dh
0Eh		PCON	8Eh
0Fh		OSCCAL	8Fh
10h			90h
11h			91h
12h			92h
13h			93h
14h			94h
15h			95h
16h			96h
17h			97h
18h			98h
19h			99h
1Ah			9Ah
1Bh			9Bh
1Ch			9Ch
1Dh			9Dh
1Eh	ADRES		9Eh
1Fh	ADCON0	ADCON1	9Fh
20h		General Purpose Register	A0h
	General Purpose Register		BFh
			C0h
			EFh
70h		Mapped in Bank 0	F0h
7Fh			FFh
	Bank 0	Bank 1	

Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM.

The Special Function Registers can be classified into two sets (core and peripheral). Those registers associated with the "core" functions are described in this section, and those related to the operation of the peripheral features are described in the section of that peripheral feature.
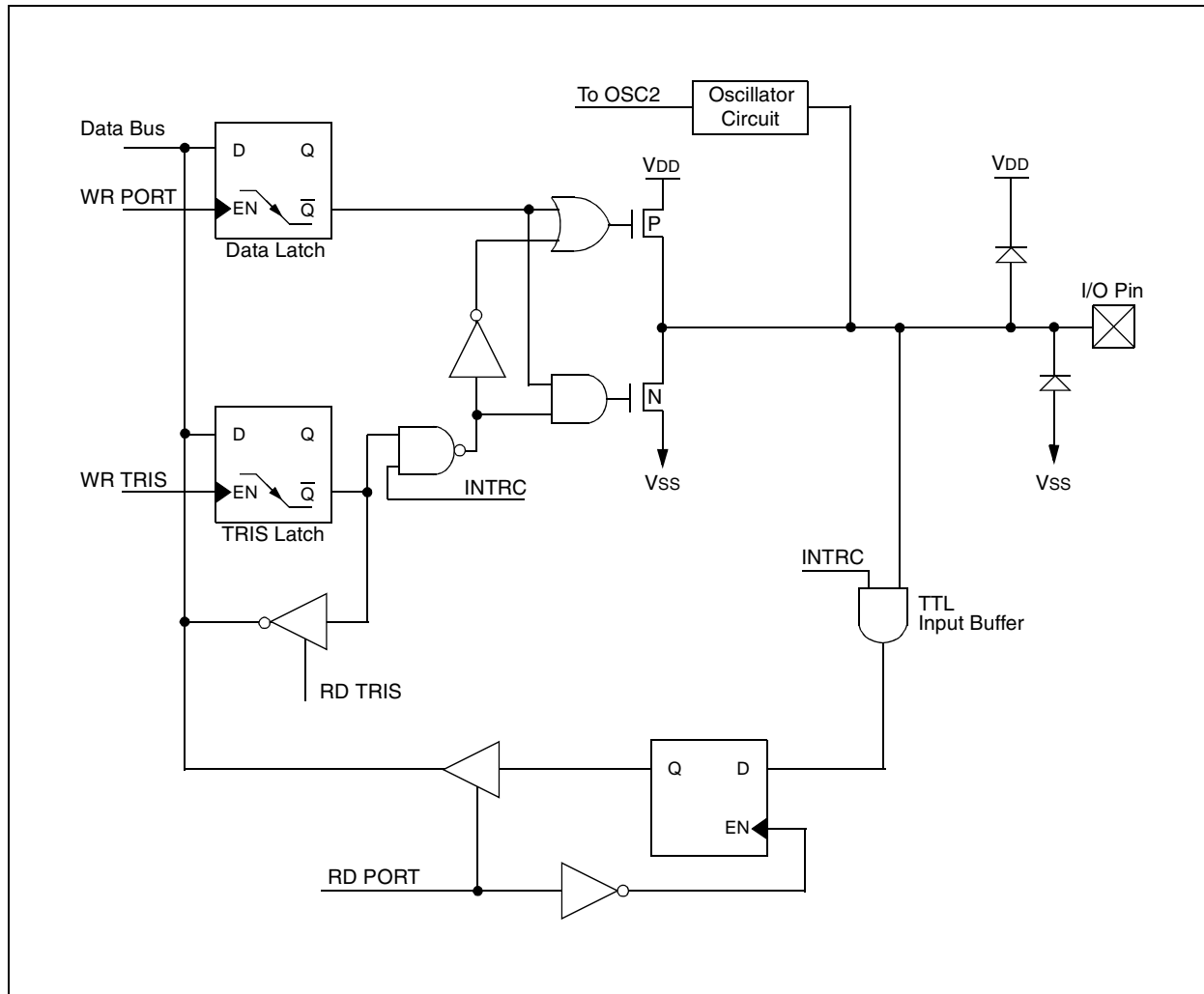
FIGURE 5-5: **BLOCK DIAGRAM OF GP5/OSC1/CLKIN PIN**

TABLE 5-1: **SUMMARY OF PORT REGISTERS**

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
85h	TRIS	—	—	GPIO Data Direction Register						`--11 1111`	`--11 1111`
81h	OPTION	$\overline{GPPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	`1111 1111`	`1111 1111`
03h	STATUS	IRP[1]	RP1[1]	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	`0001 1xxx`	`000q quuu`
05h	GPIO	SCL[2]	SDA[2]	GP5	GP4	GP3	GP2	GP1	GP0	`11xx xxxx`	`11uu uuuu`

Legend: Shaded cells not used by Port Registers, read as '0', — = unimplemented, read as '0', x = unknown, u = unchanged,
   q = see tables in Section 9.4 for possible values.

Note 1: The IRP and RP1 bits are reserved on the PIC12C67X; always maintain these bits clear.
   2: The SCL and SDA bits are unimplemented on the PIC12C671 and PIC12C672.

5.4   I/O Programming Considerations

5.4.1   BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read followed by a write operation. The BCF and BSF instructions, for example, read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of GPIO will cause all eight bits of GPIO to be read into the CPU. Then the BSF operation takes place on bit5 and GPIO is written to the output latches. If another bit of GPIO 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 to an output, the content of the data latch may now be unknown.

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

Example 5-1 shows the effect of two sequential read-modify-write instructions on an I/O port.

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

```
;Initial GPIO Settings
; GPIO<5:3> Inputs
; GPIO<2:0> Outputs
;
;                 GPIO latch  GPIO pins
;                 ----------  ----------
  BCF   GPIO, 5   ;--01 -ppp   --11 pppp
  BCF   GPIO, 4   ;--10 -ppp   --11 pppp
  MOVLW 007h      ;
  TRIS  GPIO      ;--10 -ppp   --10 pppp
;
;Note that the user may have expected the pin
;values to be --00 pppp. The 2nd BCF caused
;GP5 to be latched as the pin value (High).
```

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.

6.5   Read Operations

Read operations are initiated in the same way as write operations with the exception that the R/$\overline{W}$ bit of the EEPROM address is set to one. There are three basic types of read operations; current address read, random read and sequential read.

6.5.1   CURRENT ADDRESS READ

The EEPROM contains an address counter that maintains the address of the last word accessed, internally incremented by one. Therefore, if the previous read access was to address n, the next current address read operation would access data from address n + 1. Upon receipt of the EEPROM address with the R/$\overline{W}$ bit set to one, the EEPROM issues an acknowledge and transmits the 8-bit data word. The processor will not acknowledge the transfer, but does generate a stop condition and the EEPROM discontinues transmission (Figure 6-8).

6.5.2   RANDOM READ

Random read operations allow the processor to access any memory location in a random manner. To perform this type of read operation, first the word address must be set. This is done by sending the word address to the EEPROM as part of a write operation. After the word address is sent, the processor generates a start condition following the acknowledge. This terminates the write operation, but not before the internal address pointer is set. Then the processor issues the control byte again, but with the R/$\overline{W}$ bit set to a one. The EEPROM will then issue an acknowledge and transmits the 8-bit data word. The processor will not acknowledge the transfer, but does generate a stop condition and the EEPROM discontinues transmission (Figure 6-9). After this command, the internal address counter will point to the address location following the one that was just read.

6.5.3   SEQUENTIAL READ

Sequential reads are initiated in the same way as a random read, except that after the device transmits the first data byte, the processor issues an acknowledge as opposed to a stop condition in a random read. This directs the EEPROM to transmit the next sequentially addressed 8-bit word (Figure 6-10).

To provide sequential reads, the EEPROM contains an internal address pointer, which is incremented by one at the completion of each read operation. This address pointer allows the entire memory contents to be serially read during one operation.
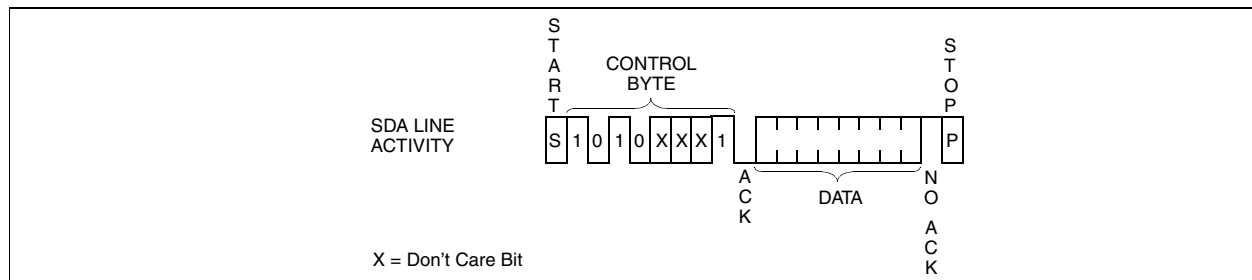
FIGURE 6-8:   CURRENT ADDRESS READ

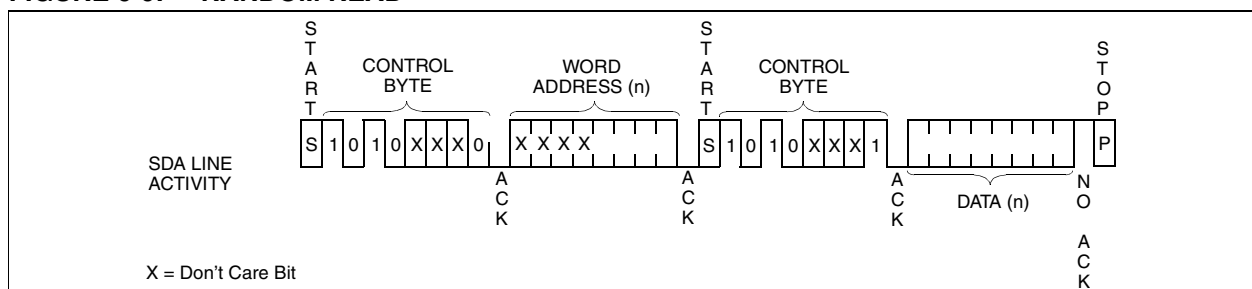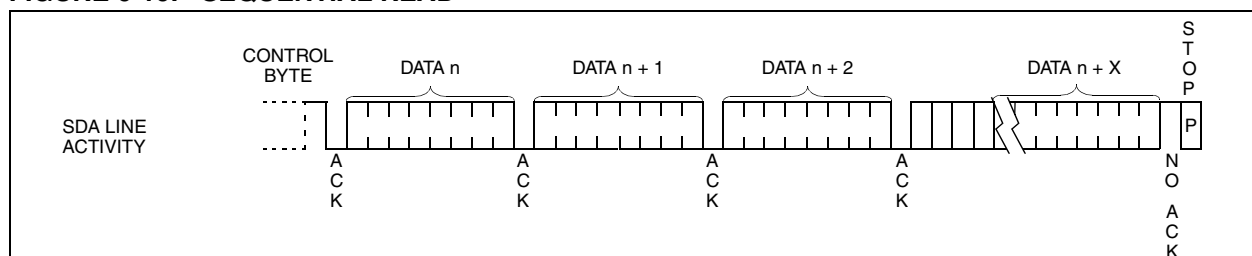

FIGURE 6-9:   RANDOM READ


FIGURE 6-10: SEQUENTIAL READ

The ADRES Register contains the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRES register, the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>) is cleared, and A/D interrupt flag bit ADIF (PIE1<6>) is set. The block diagrams of the A/D module are shown in Figure 8-1.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine sample time, see Section 8.1. After this acquisition time has elapsed, the A/D conversion can be started. The following steps should be followed for doing an A/D conversion:

1. Configure the A/D module:
    - Configure analog pins / voltage reference / and digital I/O (ADCON1 and TRIS)
    - Select A/D input channel (ADCON0)
    - Select A/D conversion clock (ADCON0)
    - Turn on A/D module (ADCON0)

2. Configure A/D interrupt (if desired):
    - Clear ADIF bit
    - Set ADIE bit
    - Set GIE bit

3. Wait the required acquisition time.

4. Start conversion:
    - Set GO/$\overline{\text{DONE}}$ bit (ADCON0)

5. Wait for A/D conversion to complete, by either:
    - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

    OR

    - Waiting for the A/D interrupt

6. Read A/D Result Register (ADRES), clear bit ADIF if required.

7. For the next conversion, go to step 1, step 2 or step 3 as required. The A/D conversion time per bit is defined as T_{AD}. A minimum wait of $2T_{AD}$ is required before next acquisition starts.

FIGURE 8-1:   A/D BLOCK DIAGRAM
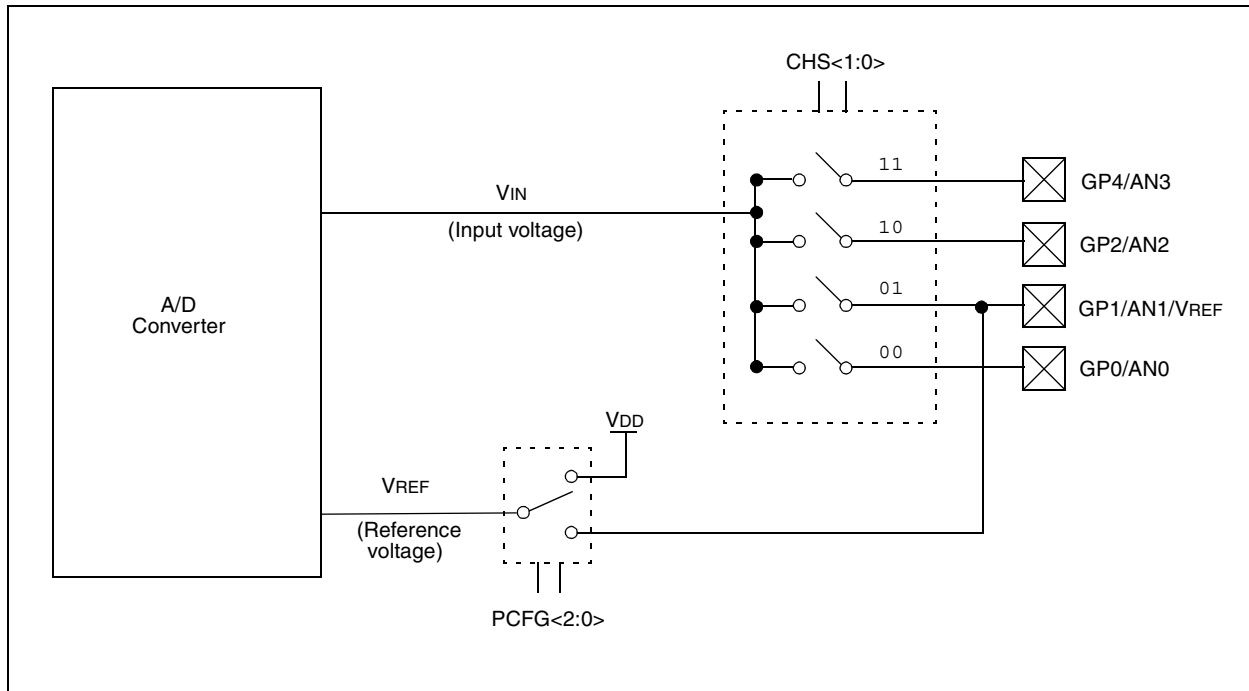
PIC12C67X

8.4 A/D Conversions

Example 8-2 shows how to perform an A/D conversion. The GPIO pins are configured as analog inputs. The analog reference (VREF) is the device VDD. The A/D interrupt is enabled and the A/D conversion clock is FRC. The conversion is performed on the GP0 channel.

> **Note:** The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The ADRES register will NOT be updated with the partially completed A/D conversion sample. That is, the ADRES register will continue to contain the value of the last completed conversion (or the last value written to the ADRES register). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, an acquisition is automatically started on the selected channel.

EXAMPLE 8-2: DOING AN A/D CONVERSION

```
        BSF     STATUS, RP0         ; Select Page 1
        CLRF    ADCON1              ; Configure A/D inputs
        BSF     PIE1,   ADIE        ; Enable A/D interrupts
        BCF     STATUS, RP0         ; Select Page 0
        MOVLW   0xC1                ; RC Clock, A/D is on, Channel 0 is selected
        MOVWF   ADCON0              ;
        BCF     PIR1,   ADIF        ; Clear A/D interrupt flag bit
        BSF     INTCON, PEIE        ; Enable peripheral interrupts
        BSF     INTCON, GIE         ; Enable all interrupts
;
;   Ensure that the required sampling time for the selected input channel has elapsed.
;   Then the conversion may be started.
;
        BSF     ADCON0, GO          ; Start A/D Conversion
          :                         ; The ADIF bit will be set and the GO/DONE bit
          :                         ;  is cleared upon completion of the A/D Conversion.
```

PIC12C67X

9.4 Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)

9.4.1 POWER-ON RESET (POR)

The on-chip POR circuit holds the chip in reset until VDD has reached a high enough level for proper operation. To take advantage of the POR, just tie the MCLR pin through a resistor to VDD. This will eliminate external RC components usually needed to create a Power-on Reset. A maximum rise time for VDD is specified. See Electrical Specifications for details.

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

For additional information, refer to Application Note AN607, "*Power-up Trouble Shooting.*"

9.4.2 POWER-UP TIMER (PWRT)

The Power-up Timer provides a fixed 72 ms nominal time-out on power-up only, from the POR. The Power-up Timer operates on an internal RC oscillator. The chip is kept in reset as long as the PWRT is active. The PWRT's time delay allows VDD to rise to an acceptable level. A configuration bit is provided to enable/disable the PWRT.

The power-up time delay will vary from chip to chip due to VDD, temperature and process variation. See Table 11-4.

9.4.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-up Timer (OST) provides 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over. This ensures that 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.

9.4.4 TIME-OUT SEQUENCE

On power-up, the Time-out Sequence is as follows: first, PWRT time-out is invoked after the POR time delay has expired; then, OST is activated. The total time-out will vary, based on oscillator configuration and the status of the PWRT. For example, in RC mode with the PWRT disabled, there will be no time-out at all. Figure 9-7, Figure 9-8, and Figure 9-9 depict time-out sequences on power-up.

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, the time-outs will expire. Then bringing MCLR high will begin execution immediately (Figure 9-9). This is useful for testing purposes or to synchronize more than one PIC12C67X device operating in parallel.

9.4.5 POWER CONTROL (PCON)/STATUS REGISTER

The Power Control/Status Register, PCON (address 8Eh), has one bit. See Register 4-6 for register.

Bit1 is POR (Power-on Reset). It is cleared on a Power-on Reset and is unaffected otherwise. The user sets this bit following a Power-on Reset. On subsequent resets, if POR is '0', it will indicate that a Power-on Reset must have occurred.

TABLE 9-4: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up		Wake-up from SLEEP
	PWRTE = 0	PWRTE = 1	
XT, HS, LP	72 ms + 1024TOSC	1024TOSC	1024TOSC
INTRC, EXTRC	72 ms	—	—

TABLE 9-5: STATUS/PCON BITS AND THEIR SIGNIFICANCE

POR	TO	PD	
0	1	1	Power-on Reset
0	0	x	Illegal, TO is set on POR
0	x	0	Illegal, PD is set on POR
1	0	u	WDT Reset
1	0	0	WDT Wake-up
1	u	u	MCLR Reset during normal operation
1	1	0	MCLR Reset during SLEEP or interrupt wake-up from SLEEP

Legend: u = unchanged, x = unknown.

9.8 Power-down Mode (SLEEP)

Power-down mode is entered by executing a `SLEEP` instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the $\overline{PD}$ bit (STATUS<3>) is cleared, the $\overline{TO}$ (STATUS<4>) 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, place all I/O pins at either VDD or VSS, ensure no external circuitry is drawing current from the I/O pin, power-down the A/D, and disable external clocks. Pull all I/O pins that are hi-impedance inputs, high or low externally, to avoid switching currents caused by floating inputs. The T0CKI input, if enabled, should also be at VDD or VSS for lowest current consumption. The contribution from on-chip pull-ups on GPIO should be considered.

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

9.8.1 WAKE-UP FROM SLEEP

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

1. External reset input on $\overline{MCLR}$ pin.
2. Watchdog Timer Wake-up (if WDT was enabled).
3. GP2/INT interrupt, interrupt GPIO port change or some Peripheral Interrupts.

External $\overline{MCLR}$ Reset will cause a device reset. All other events are considered a continuation of program execution and cause a "wake-up". The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register can be used to determine the cause of device reset. The $\overline{PD}$ bit, which is set on power-up, is cleared when `SLEEP` is invoked. The $\overline{TO}$ bit is cleared if a WDT time-out occurred (and caused wake-up).

The following peripheral interrupt can wake the device from SLEEP:

1. A/D conversion (when A/D clock source is RC).

Other peripherals can not generate interrupts since during SLEEP, no on-chip Q clocks are present.

When 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 is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the `SLEEP` instruction. If the GIE bit is set (enabled), the device executes the instruction after the `SLEEP` instruction and then branches to the interrupt address (0004h). In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

9.8.2 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the the execution of a `SLEEP` instruction, the `SLEEP` instruction will complete as a `NOP`. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bits will not be cleared.
- If the interrupt occurs **during or after** the execution of a `SLEEP` instruction, the device will immediately wake-up from sleep . The `SLEEP` instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit will be cleared.

Even if the flag bits were checked before executing a `SLEEP` instruction, it may be possible for flag bits to become set before the `SLEEP` instruction completes. To determine whether a `SLEEP` instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the `SLEEP` instruction was executed as a `NOP`.

To ensure that the WDT is cleared, a `CLRWDT` instruction should be executed before a `SLEEP` instruction.

10.0   INSTRUCTION SET SUMMARY

Each PIC12C67X 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 PIC12C67X instruction set summary in Table 10-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 10-1 shows the opcode field descriptions.

For **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 zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

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

For **literal and control** operations, 'k' represents an eight or eleven bit constant or literal value.

TABLE 10-1:   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
→	Assigned to
< >	Register bit field
∈	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** operations
- **Bit-oriented** operations
- **Literal and control** operations

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

Table 10-2 lists the instructions recognized by the MPASM assembler.

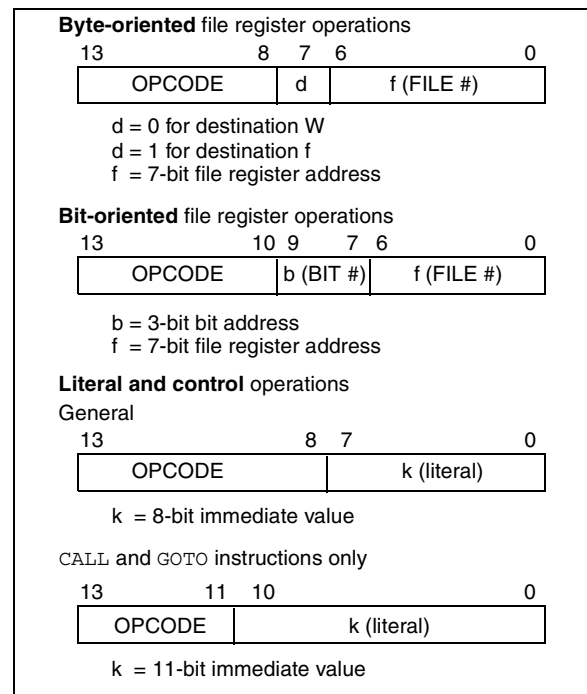Figure 10-1 shows the three general formats that the instructions can have.

Note:	To maintain upward compatibility with future PIC12C67X products, <u>do not use</u> the OPTION and TRIS instructions.

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

    0xhh

where h signifies a hexadecimal digit.

FIGURE 10-1:   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
General

13	8	7	0
OPCODE		k (literal)	

    k = 8-bit immediate value

CALL and GOTO instructions only

13	11	10	0
OPCODE		k (literal)	

    k = 11-bit immediate value

PIC12C67X

10.1 Special Function Registers as Source/Destination

The PIC12C67X's orthogonal instruction set allows read and write of all file registers, including special function registers. There are some special situations the user should be aware of:

10.1.1 STATUS AS DESTINATION

If an instruction writes to STATUS, the Z, C and DC bits may be set or cleared as a result of the instruction and overwrite the original data bits written. For example, executing `CLRF STATUS` will clear register STATUS, and then set the Z bit leaving `0000 0100b` in the register.

10.1.2 TRIS AS DESTINATION

Bit 3 of the TRIS register always reads as a '1' since GP3 is an input only pin. This fact can affect some read-modify-write operations on the TRIS register.

10.1.3 PCL AS SOURCE OR DESTINATION

Read, write or read-modify-write on PCL may have the following results:

Read PC: $PCL \rightarrow dest$

Write PCL: $PCLATH \rightarrow PCH$;
8-bit destination value $\rightarrow PCL$

Read-Modify-Write: $PCL \rightarrow ALU$ operand
$PCLATH \rightarrow PCH$;
8-bit result $\rightarrow PCL$

Where PCH = program counter high byte (not an addressable register), PCLATH = Program counter high holding latch, dest = destination, WREG or f.

10.1.4 BIT MANIPULATION

All bit manipulation instructions are done by first reading the entire register, operating on the selected bit and writing the result back (read-modify-write). The user should keep this in mind when operating on special function registers, such as ports.

PIC12C67X

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b < 7$
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    BTFSS  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	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	$0 \leq k \leq 2047$
Operation:	(PC)+ 1$\rightarrow$ TOS, k $\rightarrow$ PC<10:0>, (PCLATH<4:3>) $\rightarrow$ PC<12:11>
Status Affected:	None
Encoding:	`10` `0kkk` `kkkk` `kkkk`
Description:	Call Subroutine. 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+1`

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	$0 \leq f \leq 127$
Operation:	00h $\rightarrow$ (f) 1 $\rightarrow$ 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
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h $\rightarrow$ (W) 1 $\rightarrow$ Z
Status Affected:	Z
Encoding:	`00` `0001` `0000` `0011`
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

11.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM Assembler
  - MPLAB-C17 and MPLAB-C18 C Compilers
  - MPLINK/MPLIB Linker/Librarian
- Simulators
  - MPLAB-SIM Software Simulator
- Emulators
  - MPLAB-ICE Real-Time In-Circuit Emulator
  - PICMASTER®/PICMASTER-CE In-Circuit Emulator
  - ICEPIC™
- In-Circuit Debugger
  - MPLAB-ICD for PIC16F877
- Device Programmers
  - PRO MATE® II Universal Programmer
  - PICSTART® Plus Entry-Level Prototype Programmer
- Low-Cost Demonstration Boards
  - SIMICE
  - PICDEM-1
  - PICDEM-2
  - PICDEM-3
  - PICDEM-17
  - SEEVAL®
  - KEELOQ®

11.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. MPLAB is a Windows®-based application which contains:

- Multiple functionality
  - editor
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
- A full featured editor
- A project manager
- Customizable tool bar and key mapping
- A status bar
- On-line help

MPLAB allows you to:

- Edit your source files (either assembly or 'C')
- One touch assemble (or compile) and download to PIC MCU tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - object code

The ability to use MPLAB with Microchip's simulator, MPLAB-SIM, allows a consistent platform and the ability to easily switch from the cost-effective simulator to the full featured emulator with minimal retraining.

11.2 MPASM Assembler

MPASM is a full featured universal macro assembler for all PIC MCUs. It can produce absolute code directly in the form of HEX files for device programmers, or it can generate relocatable objects for MPLINK.

MPASM has a command line interface and a Windows shell and can be used as a standalone application on a Windows 3.x or greater system. MPASM generates relocatable object files, Intel standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file which contains source lines and generated machine code, and a COD file for MPLAB debugging.

MPASM features include:

- MPASM and MPLINK are integrated into MPLAB projects.
- MPASM allows user defined macros to be created for streamlined assembly.
- MPASM allows conditional assembly for multi purpose source files.
- MPASM directives allow complete control over the assembly process.

11.3 MPLAB-C17 and MPLAB-C18 C Compilers

The MPLAB-C17 and MPLAB-C18 Code Development Systems are complete ANSI 'C' compilers and integrated development environments for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

11.4 MPLINK/MPLIB Linker/Librarian

MPLINK is a relocatable linker for MPASM and MPLAB-C17 and MPLAB-C18. It can link relocatable objects from assembly or C source files along with precompiled libraries using directives from a linker script.

MPLIB is a librarian for pre-compiled code to be used with MPLINK. When a routine from a library is called from another source file, only the modules that contains that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications. MPLIB manages the creation and modification of library files.

MPLINK features include:

- MPLINK works with MPASM and MPLAB-C17 and MPLAB-C18.
- MPLINK allows all memory areas to be defined as sections to provide link-time flexibility.

MPLIB features include:

- MPLIB makes linking easier because single libraries can be included instead of many smaller files.
- MPLIB helps keep code maintainable by grouping related modules together.
- MPLIB commands allow libraries to be created and modules to be added, listed, replaced, deleted, or extracted.

11.5   MPLAB-SIM Software Simulator

The MPLAB-SIM Software Simulator allows code development in a PC host environment by simulating the PIC series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file or user-defined key press to any of the pins. The execution can be performed in single step, execute until break, or trace mode.

MPLAB-SIM fully supports symbolic debugging using MPLAB-C17 and MPLAB-C18 and MPASM. The Software Simulator offers the flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

11.6   MPLAB-ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB-ICE Universal In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers (MCUs). Software control of MPLAB-ICE is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, "make" and download, and source debugging from a single environment.

Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB-ICE allows expansion to support new PIC microcontrollers.

The MPLAB-ICE Emulator System has been designed as a real-time emulation system with advanced features that are generally found on more expensive development tools. The PC platform and Microsoft® Windows 3.x/95/98 environment were chosen to best make these features available to you, the end user.

MPLAB-ICE 2000 is a full-featured emulator system with enhanced trace, trigger, and data monitoring features. Both systems use the same processor modules and will operate across the full operating speed range of the PIC MCU.

11.7   PICMASTER/PICMASTER CE

The PICMASTER system from Microchip Technology is a full-featured, professional quality emulator system. This flexible in-circuit emulator provides a high-quality, universal platform for emulating Microchip 8-bit PIC microcontrollers (MCUs). PICMASTER systems are sold worldwide, with a CE compliant model available for European Union (EU) countries.

11.8   ICEPIC

ICEPIC is a low-cost in-circuit emulation solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X, and PIC16CXXX families of 8-bit one-time-programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules or daughter boards. The emulator is capable of emulating without target application circuitry being present.

11.9   MPLAB-ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB-ICD, is a powerful, low-cost run-time development tool. This tool is based on the flash PIC16F877 and can be used to develop for this and other PIC microcontrollers from the PIC16CXXX family. MPLAB-ICD utilizes the In-Circuit Debugging capability built into the PIC16F87X. This feature, along with Microchip's In-Circuit Serial Programming protocol, offers cost-effective in-circuit flash programming and debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time. The MPLAB-ICD is also a programmer for the flash PIC16F87X family.

11.10   PRO MATE II Universal Programmer

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

The PRO MATE II has programmable VDD and VPP supplies which allows it to verify programmed memory at VDD min and VDD max for maximum reliability. It has an LCD display for instructions and error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In
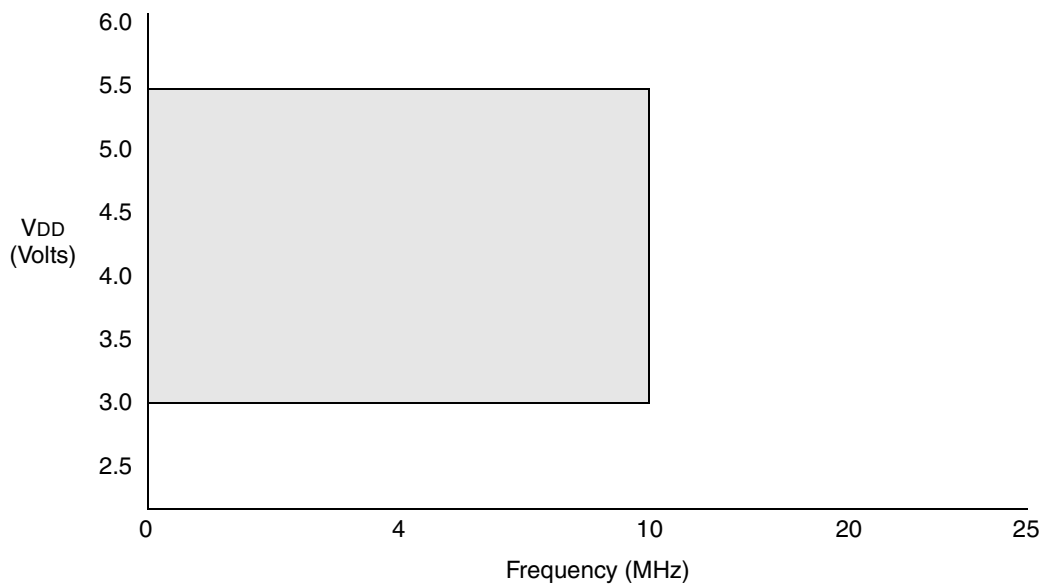
PIC12C67X

FIGURE 12-1: PIC12C67X VOLTAGE-FREQUENCY GRAPH, -40°C ≤ TA < 0°C, +70°C <TA ≤ +125°C
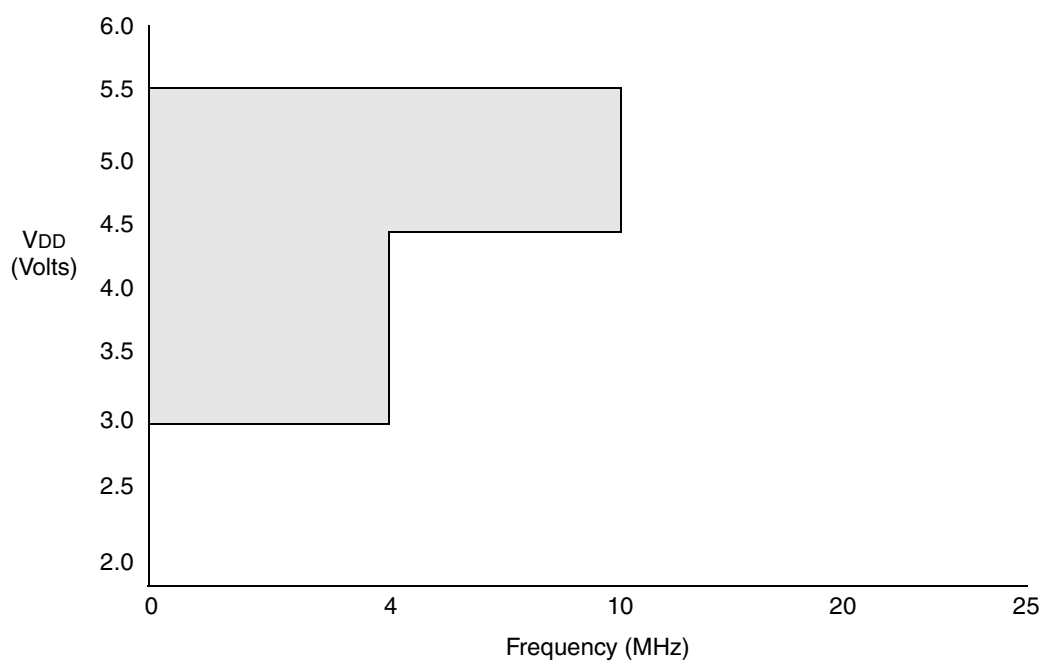
Note 1: The shaded region indicates the permissible combinations of voltage and frequency.

2: The maximum rated speed of the part limits the permissible combinations of voltage and frequency. Please reference the Product Identification System section for the maximum rated speed of the parts.

FIGURE 12-2: PIC12C67X VOLTAGE-FREQUENCY GRAPH, 0°C ≤ TA ≤ +70°C

Note 1: The shaded region indicates the permissible combinations of voltage and frequency.

2: The maximum rated speed of the part limits the permissible combinations of voltage and frequency. Please reference the Product Identification System section for the maximum rated speed of the parts.

PIC12C67X

FIGURE 12-8: **TIMER0 CLOCK TIMINGS**

Note: Refer to Figure 12-4 for load conditions.
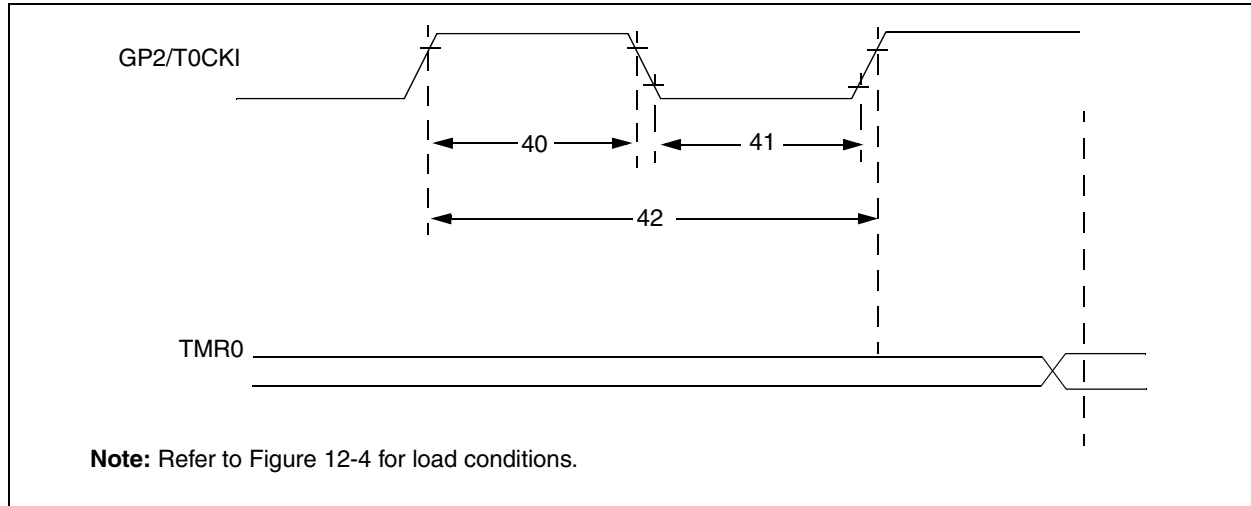
TABLE 12-5: **TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40*	Tt0H	T0CKI High Pulse Width	No Prescaler	$0.5T_{CY} + 20$	—	—	ns	Must also meet parameter 42
			With Prescaler	10	—	—	ns	
41*	Tt0L	T0CKI Low Pulse Width	No Prescaler	$0.5T_{CY} + 20$	—	—	ns	Must also meet parameter 42
			With Prescaler	10	—	—	ns	
42*	Tt0P	T0CKI Period	No Prescaler	$T_{CY} + 40$	—	—	ns	
			With Prescaler	Greater of: 20 or $\frac{T_{CY} + 40}{N}$	—	—	ns	N = prescale value (2, 4,..., 256)
48	TCKE2tmr1	Delay from external clock edge to timer increment		$2T_{OSC}$	—	$7T_{OSC}$	—	

* 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.

TABLE 12-6: **GPIO PULL-UP RESISTOR RANGES**

V_{DD} (Volts)	Temperature (°C)	Min	Typ	Max	Units
GP0/GP1					
2.5	−40	38K	42K	63K	Ω
	25	42K	48K	63K	Ω
	85	42K	49K	63K	Ω
	125	50K	55K	63K	Ω
5.5	−40	15K	17K	20K	Ω
	25	18K	20K	23K	Ω
	85	19K	22K	25K	Ω
	125	22K	24K	28K	Ω
GP3					
2.5	−40	285K	346K	417K	Ω
	25	343K	414K	532K	Ω
	85	368K	457K	532K	Ω
	125	431K	504K	593K	Ω
5.5	−40	247K	292K	360K	Ω
	25	288K	341K	437K	Ω
	85	306K	371K	448K	Ω
	125	351K	407K	500K	Ω

* These parameters are characterized but not tested.

NOTES:

NOTES:

PIC12C67X

PIC12C67X PRODUCT IDENTIFICATION SYSTEM

PART NO.	-XX	X	/XX	XXX			Examples

Pattern: Special Requirements

Package:
- P = 300 mil PDIP
- JW = 300 mil Windowed Ceramic Side Brazed
- SM = 208 mil SOIC

Temperature Range:
- − = 0°C to +70°C
- I = -40°C to +85°C
- E = -40°C to +125°C

Frequency Range:
- 04 = 4 MHz/200 kHz
- 10 = 10 MHz

Device
- PIC12CE673
- PIC12CE674
- PIC12LCE673
- PIC12LCE674
- PIC12C671
- PIC12C672
- PIC12C671T (Tape & reel for SOIC only)
- PIC12C672T (Tape & reel for SOIC only)
- PIC12LC671
- PIC12LC672
- PIC12LC671T (Tape & reel for SOIC only)
- PIC12LC672T (Tape & reel for SOIC only)

Examples

a) PIC12CE673-04/P
Commercial Temp., PDIP Package, 4 MHz, normal V_{DD} limits

b) PIC12CE673-04I/P
Industrial Temp., PDIP package, 4 MHz, normal V_{DD} limits

c) PIC12CE673-10I/P
Industrial Temp., PDIP package, 10 MHz, normal V_{DD} limits

d) PIC12C671-04/P
Commercial Temp., PDIP Package, 4 MHz, normal V_{DD} limits

e) PIC12C671-04I/SM
Industrial Temp., SOIC package, 4 MHz, normal V_{DD} limits

f) PIC12C671-04I/P
Industrial Temp., PDIP package, 4 MHz, normal V_{DD} limits

* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type (including LC devices).

Sales and Support

Data Sheets

Products supported by a preliminary Data Sheet may 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
2. The Microchip Worldwide Site (www.microchip.com)