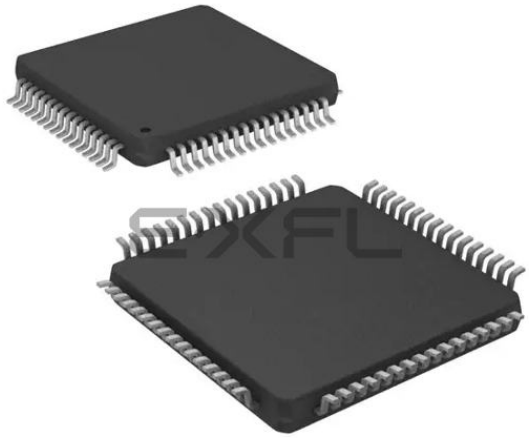


Welcome to E-XFL.COM

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	16-Bit
Speed	16MHz
Connectivity	I²C, PMP, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	53
Program Memory Size	96KB (32K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj96ga006-i-pt

TABLE 1-2: PIC24FJ128GA010 FAMILY PINOUT DESCRIPTIONS (CONTINUED)

Function	Pin Number			I/O	Input Buffer	Description
	64-Pin	80-Pin	100-Pin			
PMA0	30	36	44	I/O	ST/TTL	Parallel Master Port Address Bit 0 Input (Buffered Slave modes) and Output (Master modes).
PMA1	29	35	43	I/O	ST/TTL	Parallel Master Port Address Bit 1 Input (Buffered Slave modes) and Output (Master modes).
PMA2	8	10	14	O	—	Parallel Master Port Address (Demultiplexed Master modes).
PMA3	6	8	12	O	—	
PMA4	5	7	11	O	—	
PMA5	4	6	10	O	—	
PMA6	16	24	29	O	—	
PMA7	22	23	28	O	—	
PMA8	32	40	50	O	—	
PMA9	31	39	49	O	—	
PMA10	28	34	42	O	—	
PMA11	27	33	41	O	—	
PMA12	24	30	35	O	—	
PMA13	23	29	34	O	—	
PMBE	51	63	78	O	—	Parallel Master Port Byte Enable Strobe.
PMCS1	45	57	71	I/O	ST/TTL	Parallel Master Port Chip Select 1 Strobe/Address bit 14.
PMCS2	44	56	70	O	—	Parallel Master Port Chip Select 2 Strobe/Address bit 15.
PMD0	60	76	93	I/O	ST/TTL	Parallel Master Port Data (Demultiplexed Master mode) or Address/Data (Multiplexed Master modes).
PMD1	61	77	94	I/O	ST/TTL	
PMD2	62	78	98	I/O	ST/TTL	
PMD3	63	79	99	I/O	ST/TTL	
PMD4	64	80	100	I/O	ST/TTL	
PMD5	1	1	3	I/O	ST/TTL	
PMD6	2	2	4	I/O	ST/TTL	
PMD7	3	3	5	I/O	ST/TTL	
PMRD	53	67	82	I/O	ST/TTL	Parallel Master Port Read Strobe.
PMWR	52	66	81	I/O	ST/TTL	Parallel Master Port Write Strobe.

Legend: TTL = TTL input buffer, ST = Schmitt Trigger input buffer, ANA = Analog level input/output, I^2C™ = I^2C/SMBus input buffer

2.6 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 8.0 "Oscillator Configuration"** for details).

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

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

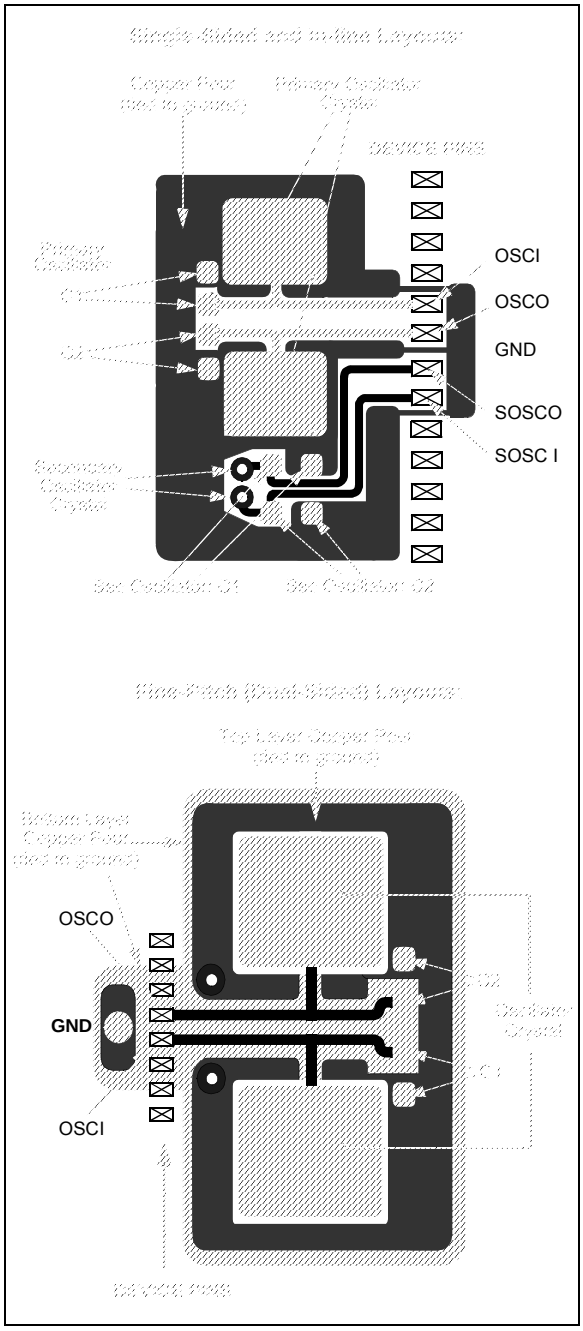
Layout suggestions are shown in Figure 2-5. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

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

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

- *AN826, "Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices"*
- *AN849, "Basic PICmicro® Oscillator Design"*
- *AN943, "Practical PICmicro® Oscillator Analysis and Design"*
- *AN949, "Making Your Oscillator Work"*

FIGURE 2-5: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT



3.0 CPU

Note: This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. Refer to **Section 2. "CPU"** (DS39703) in the *"PIC24F Family Reference Manual"* for more information.

The PIC24F CPU has a 16-bit (data) modified Harvard architecture with an enhanced instruction set, and a 24-bit instruction word with a variable length opcode field. The Program Counter (PC) is 23 bits wide and addresses up to 4M instructions of user program memory space. A single-cycle instruction prefetch mechanism is used to help maintain throughput and provides predictable execution. All instructions execute in a single cycle, with the exception of instructions that change the program flow, the double-word move (`MOV.D`) instruction and the table instructions. Overhead-free program loop constructs are supported using the `REPEAT` instructions, which are interruptible at any point.

PIC24F devices have sixteen 16-bit working registers in the programmer's model. Each of the working registers can act as a data, address or address offset register. The 16th working register (W15) operates as a Software Stack Pointer for interrupts and calls.

The upper 32 Kbytes of the data space memory map can optionally be mapped into program space at any 16K word boundary defined by the 8-bit Program Space Visibility Page (PSVPAG) register. The program to data space mapping feature lets any instruction access program space as if it were data space.

The Instruction Set Architecture (ISA) has been significantly enhanced beyond that of the PIC18, but maintains an acceptable level of backward compatibility. All PIC18 instructions and addressing modes are supported either directly or through simple macros. Many of the ISA enhancements have been driven by compiler efficiency needs.

The core supports Inherent (no operand), Relative, Literal, Memory Direct and three groups of addressing modes. All modes support Register Direct and various Register Indirect modes. Each group offers up to 7 addressing modes. Instructions are associated with predefined addressing modes depending upon their functional requirements.

For most instructions, the core is capable of executing a data (or program data) memory read, a working register (data) read, a data memory write and a program (instruction) memory read per instruction cycle. As a result, three parameter instructions can be supported, allowing trinary operations (that is, A + B = C) to be executed in a single cycle.

A high-speed, 17-bit by 17-bit multiplier has been included to significantly enhance the core arithmetic capability and throughput. The multiplier supports signed, unsigned and Mixed mode 16-bit by 16-bit or 8-bit by 8-bit integer multiplication. All multiply instructions execute in a single cycle.

The 16-bit ALU has been enhanced with integer divide assist hardware that supports an iterative, non-restoring divide algorithm. It operates in conjunction with the `REPEAT` instruction looping mechanism, and a selection of iterative divide instructions, to support 32-bit (or 16-bit) divided by 16-bit integer signed and unsigned division. All divide operations require 19 cycles to complete but are interruptible at any cycle boundary.

The PIC24F has a vectored exception scheme with up to 8 sources of non-maskable traps and up to 118 interrupt sources. Each interrupt source can be assigned to one of seven priority levels.

A block diagram of the CPU is shown in Figure 3-1.

3.1 Programmer's Model

The programmer's model for the PIC24F is shown in Figure 3-2. All registers in the programmer's model are memory mapped and can be manipulated directly by instructions. A description of each register is provided in Table 3-1. All registers associated with the programmer's model are memory mapped.

REGISTER 3-2: CORCON: CORE CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/C-0	R/W-0	U-0	U-0
—	—	—	—	IPL3(1)	PSV	—	—
bit 7							bit 0

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

bit 15-4 **Unimplemented:** Read as '0'

bit 3 **IPL3:** CPU Interrupt Priority Level Status bit(1)
1 = CPU Interrupt Priority Level is greater than 7
0 = CPU Interrupt Priority Level is 7 or less

bit 2 **PSV:** Program Space Visibility in Data Space Enable bit
1 = Program space is visible in data space
0 = Program space is not visible in data space

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

Note 1: User interrupts are disabled when IPL3 = 1.

REGISTER 7-4: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
ALTIVT	DISI	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **ALTIVT:** Enable Alternate Interrupt Vector Table bit
1 = Use alternate vector table
0 = Use standard (default) vector table

bit 14 **DISI:** DISI Instruction Status bit
1 = DISI instruction is active
0 = DISI is not active

bit 13-5 **Unimplemented:** Read as ‘0’

bit 4 **INT4EP:** External Interrupt 4 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 3 **INT3EP:** External Interrupt 3 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 2 **INT2EP:** External Interrupt 2 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 1 INT1EP: External Interrupt 1 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 0 **INT0EP:** External Interrupt 0 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

REGISTER 7-18: IPC3: INTERRUPT PRIORITY CONTROL REGISTER 3

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	AD1IP2	AD1IP1	AD1IP0	—	U1TXIP2	U1TXIP1	U1TXIP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-7 **Unimplemented:** Read as ‘0’

bit 6-4 **AD1IP<2:0>:** A/D Conversion Complete Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **U1TXIP<2:0>:** UART1 Transmitter Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

REGISTER 7-30: IPC16: INTERRUPT PRIORITY CONTROL REGISTER 16

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	CRCIP2	CRCIP1	CRCIP0	—	U2ERIP2	U2ERIP1	U2ERIP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	U1ERIP2	U1ERIP1	U1ERIP0	—	—	—	—
bit 7							bit 0

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

bit 15 **Unimplemented:** Read as '0'

bit 14-12 **CRCIP2:0>:** CRC Generator Error Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **U2ERIP<2:0>:** UART2 Error Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

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

bit 6-4 **U1ERIP<2:0>:** UART1 Error Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

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

10.0 I/O PORTS

Note: This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. Refer to **Section 12. "I/O Ports with Peripheral Pin Select (PPS)"** (DS39711) in the *"PIC24F Family Reference Manual"* for more information.

All of the device pins (except VDD, VSS, $\overline{\text{MCLR}}$ and OSC1/CLKI) are shared between the peripherals and the parallel I/O ports. All I/O input ports feature Schmitt Trigger inputs for improved noise immunity.

10.1 Parallel I/O (PIO) Ports

A parallel I/O port that shares a pin with a peripheral is, in general, subservient to the peripheral. The peripheral's output buffer data and control signals are provided to a pair of multiplexers. The multiplexers select whether the peripheral or the associated port has ownership of the output data and control signals of the I/O pin. The logic also prevents "loop through", in which a port's digital output can drive the input of a peripheral that shares the same pin. Figure 10-1 shows how ports are shared with other peripherals and the associated I/O pin to which they are connected.

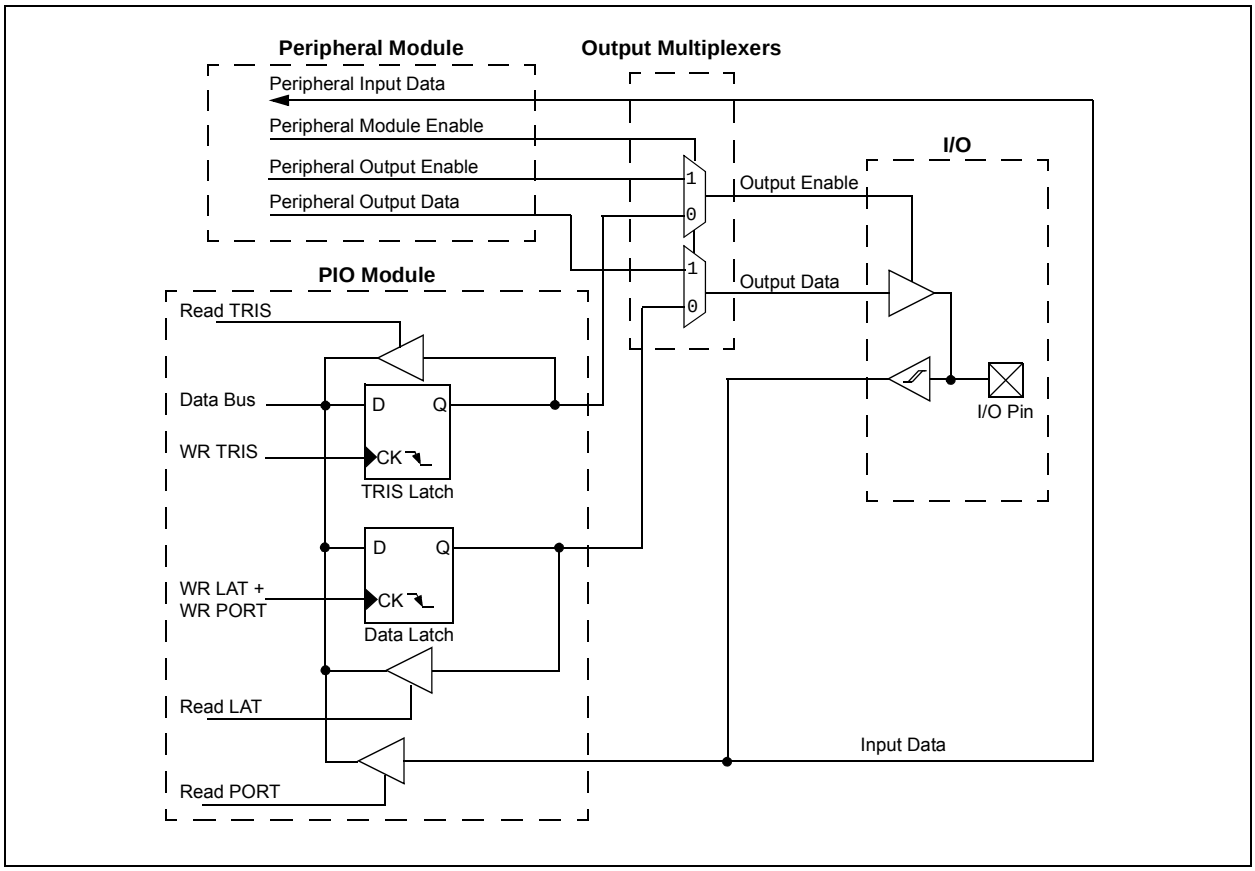
When a peripheral is enabled and the peripheral is actively driving an associated pin, the use of the pin as a general purpose output pin is disabled. The I/O pin may be read, but the output driver for the parallel port bit will be disabled. If a peripheral is enabled, but the peripheral is not actively driving a pin, that pin may be driven by a port.

All port pins have three registers directly associated with their operation as digital I/O. The Data Direction register (TRISx) determines whether the pin is an input or an output. If the data direction bit is a '1', then the pin is an input. All port pins are defined as inputs after a Reset. Reads from the latch (LATx), read the latch. Writes to the latch, write the latch. Reads from the port (PORTx), read the port pins, while writes to the port pins, write the latch.

Any bit and its associated data and control registers that are not valid for a particular device will be disabled. That means the corresponding LATx and TRISx registers, and the port pin will read as zeros.

When a pin is shared with another peripheral or function that is defined as an input only, it is nevertheless, regarded as a dedicated port because there is no other competing source of outputs. An example is the INT4 pin.

FIGURE 10-1: BLOCK DIAGRAM OF A TYPICAL SHARED PORT STRUCTURE



17.1 UARTx Baud Rate Generator (BRG)

The UARTx module includes a dedicated, 16-bit Baud Rate Generator. The UBRGx register controls the period of a free-running, 16-bit timer. Equation 17-1 shows the formula for computation of the baud rate with BRGH = 0.

EQUATION 17-1: UARTx BAUD RATE WITH BRGH = 0(1)

$$\text{Baud Rate} = \frac{F_{CY}}{16 \cdot (\text{UBRGx} + 1)}$$

$$\text{UBRGx} = \frac{F_{CY}}{16 \cdot \text{Baud Rate}} - 1$$

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

Example 17-1 shows the calculation of the baud rate error for the following conditions:

- FCY = 4 MHz
- Desired Baud Rate = 9600

The maximum baud rate (BRGH = 0) possible is FCY/16 (for UBRGx = 0) and the minimum baud rate possible is FCY/(16 * 65536).

Equation 17-2 shows the formula for computation of the baud rate with BRGH = 1.

EQUATION 17-2: UARTx BAUD RATE WITH BRGH = 1(1)

$$\text{Baud Rate} = \frac{F_{CY}}{4 \cdot (\text{UBRGx} + 1)}$$

$$\text{UBRGx} = \frac{F_{CY}}{4 \cdot \text{Baud Rate}} - 1$$

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

The maximum baud rate (BRGH = 1) possible is FCY/4 (for UBRGx = 0) and the minimum baud rate possible is FCY/(4 * 65536).

Writing a new value to the UBRGx register causes the BRG timer to be reset (cleared). This ensures the BRG does not wait for a timer overflow before generating the new baud rate.

EXAMPLE 17-1: BAUD RATE ERROR CALCULATION (BRGH = 0)(1)

Desired Baud Rate = FCY/(16 (UBRGx + 1))

Solving for UBRGx value:

BRGx = ((FCY/Desired Baud Rate)/16) – 1
BRGx = ((4000000/9600)/16) – 1
BRGx = 25

Calculated Baud Rate = 4000000/(16 (25 + 1))
= 9615

Error = (Calculated Baud Rate – Desired Baud Rate) / Desired Baud Rate
= (9615 – 9600)/9600
= 0.16%

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

REGISTER 17-2: UxSTA: UARTx STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	U-0	R/W-0, HC	R/W-0	R-0	R-1
UTXISEL1	TXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7							bit 0

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

bit 15,13 **UTXISEL<1:0>:** Transmission Interrupt Mode Selection bits
- 11 = Reserved; do not use
- 10 = Interrupt when a character is transferred to the Transmit Shift Register and as a result, the transmit buffer becomes empty
- 01 = Interrupt when the last character is shifted out of the Transmit Shift Register; all transmit operations are completed
- 00 = Interrupt when a character is transferred to the Transmit Shift Register (this implies there is at least one character open in the transmit buffer)

bit 14 **TXINV:** Transmit Polarity Inversion bit

IREN = 0:
- 1 = TX Idle state is '0'
- 0 = TX Idle state is '1'

IREN = 1:
- 1 = IrDA® encoded TX Idle state is '1'
- 0 = IrDA encoded TX Idle state is '0'

bit 12 **Unimplemented:** Read as '0'

bit 11 **UTXBRK:** Transmit Break bit
- 1 = Sends Sync Break on next transmission – Start bit, followed by twelve '0' bits, followed by Stop bit; cleared by hardware upon completion
- 0 = Sync Break transmission is disabled or completed

bit 10 **UTXEN:** Transmit Enable bit
- 1 = Transmit is enabled, UxTX pin controlled by UARTx
- 0 = Transmit is disabled, any pending transmission is aborted and the buffer is reset. UxTX pin is controlled by the PORT.

bit 9 **UTXBF:** Transmit Buffer Full Status bit (read-only)
- 1 = Transmit buffer is full
- 0 = Transmit buffer is not full, at least one more character can be written

bit 8 **TRMT:** Transmit Shift Register Empty bit (read-only)
- 1 = Transmit Shift Register is empty and transmit buffer is empty (the last transmission has completed)
- 0 = Transmit Shift Register is not empty, a transmission is in progress or queued

bit 7-6 **URXISEL<1:0>:** Receive Interrupt Mode Selection bits
- 11 = Interrupt is set on RSR transfer, making the receive buffer full (i.e., has 4 data characters)
- 10 = Interrupt is set on RSR transfer, making the receive buffer 3/4 full (i.e., has 3 data characters)
- 0x = Interrupt is set when any character is received and transferred from the RSR to the receive buffer; receive buffer has one or more characters

bit 5 **ADDEN:** Address Character Detect bit (bit 8 of received data = 1)
- 1 = Address Detect mode is enabled. If 9-bit mode is not selected, this does not take effect.
- 0 = Address Detect mode is disabled

REGISTER 21-4: AD1CHS: A/D INPUT SELECT REGISTER

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NB	—	—	—	CH0SB3	CH0SB2	CH0SB1	CH0SB0
bit 15							bit 8

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NA	—	—	—	CH0SA3	CH0SA2	CH0SA1	CH0SA0
bit 7							bit 0

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

bit 15 **CH0NB:** Channel 0 Negative Input Select for MUX B Multiplexor Setting bit
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VR-

bit 14-12 **Unimplemented:** Read as '0'

bit 11-8 **CH0SB<3:0>:** Channel 0 Positive Input Select for MUX B Multiplexor Setting bits
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
.....
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0

bit 7 **CH0NA:** Channel 0 Negative Input Select for MUX A Multiplexor Setting bit
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VR-

bit 6-4 **Unimplemented:** Read as '0'

bit 3-0 **CH0SA<3:0>:** Channel 0 Positive Input Select for MUX A Multiplexor Setting bits
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
.....
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0

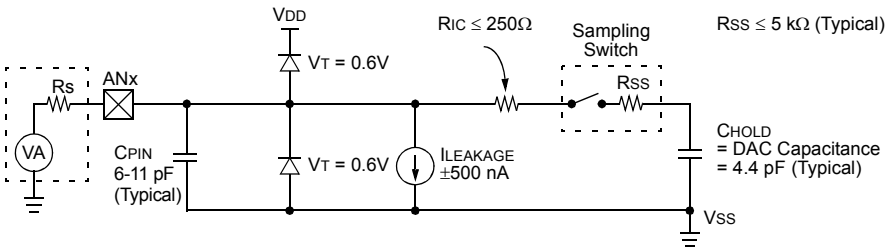
EQUATION 21-1: A/D CONVERSION CLOCK PERIOD(1)

$$T_{AD} = T_{CY}(ADCS + 1)$$

$$ADCS = \frac{T_{AD}}{T_{CY}} - 1$$

Note 1: Based on $T_{CY} = T_{OSC} * 2$; Doze mode and PLL are disabled.

FIGURE 21-2: 10-BIT A/D CONVERTER ANALOG INPUT MODEL



Legend:		
	C_{PIN}	= Input Capacitance
	V_T	= Threshold Voltage
	$I_{LEAKAGE}$	= Leakage Current at the pin due to various junctions
	R_{IC}	= Interconnect Resistance
	R_{SS}	= Sampling Switch Resistance
	C_{HOLD}	= Sample/Hold Capacitance (from DAC)

Note: C_{PIN} value depends on the device package and is not tested. The effect of C_{PIN} is negligible if Rs ≤ 5 kΩ.

NOTES:

REGISTER 22-1: CMCON: COMPARATOR CONTROL REGISTER (CONTINUED)

bit 5 **C2INV:** Comparator 2 Output Inversion bit
1 = C2 output is inverted
0 = C2 output is not inverted

bit 4 **C1INV:** Comparator 1 Output Inversion bit
1 = C1 output is inverted
0 = C1 output is not inverted

bit 3 **C2NEG:** Comparator 2 Negative Input Configure bit
1 = C2IN+ is connected to VIN-
0 = C2IN- is connected to VIN-
See Figure 22-1 for the Comparator modes.

bit 2 **C2POS:** Comparator 2 Positive Input Configure bit
1 = C2IN+ is connected to VIN+
0 = CVREF is connected to VIN+
See Figure 22-1 for the Comparator modes.

bit 1 **C1NEG:** Comparator 1 Negative Input Configure bit
1 = C1IN+ is connected to VIN-
0 = C1IN- is connected to VIN-
See Figure 22-1 for the Comparator modes.

bit 0 **C1POS:** Comparator 1 Positive Input Configure bit
1 = C1IN is connected to VIN+
0 = CVREF is connected to VIN+
See Figure 22-1 for the Comparator modes.

REGISTER 24-3: DEVID: DEVICE ID REGISTER

U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

U	U	R	R	R	R	R	R
—	—	FAMID7	FAMID6	FAMID5	FAMID4	FAMID3	FAMID2
bit 15							bit 8

R	R	R	R	R	R	R	R
FAMID1	FAMID0	DEV5	DEV4	DEV3	DEV2	DEV1	DEV0
bit 7							bit 0

Legend:	x = Bit is unknown		
R = Readable bit	PO = Program Once bit	U = Unimplemented bit, read as ‘1’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 23-14 **Unimplemented:** Read as ‘0’

bit 13-6 **FAMID<7:0>:** Device Family Identifier bits
00010000 = PIC24FJ128GA010 family

bit 5-0 **DEV<5:0>:** Individual Device Identifier bits
000101 = PIC24FJ64GA006
000110 = PIC24FJ96GA006
000111 = PIC24FJ128GA006
001000 = PIC24FJ64GA008
001001 = PIC24FJ96GA008
001010 = PIC24FJ128GA008
001011 = PIC24FJ64GA010
001100 = PIC24FJ96GA010
001101 = PIC24FJ128GA010

25.0 INSTRUCTION SET SUMMARY

The PIC24F instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from previous PIC MCU instruction sets. Most instructions are a single program memory word. Only three instructions require two program memory locations.

Each single-word instruction is a 24-bit word divided into an 8-bit opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction. The instruction set is highly orthogonal and is grouped into four basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- Control operations

Table 25-1 shows the general symbols used in describing the instructions. The PIC24F instruction set summary in Table 25-2 lists all the instructions, along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand which is typically a register ‘Wb’ without any address modifier
- The second source operand which is typically a register ‘Ws’ with or without an address modifier
- The destination of the result which is typically a register ‘Wd’ with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value, ‘f’
- The destination, which could either be the file register ‘f’ or the W0 register, which is denoted as ‘WREG’

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of ‘Ws’ or ‘f’)
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register, ‘Wb’)

The literal instructions that involve data movement may use some of the following operands:

- A literal value to be loaded into a W register or file register (specified by the value of ‘k’)
- The W register or file register where the literal value is to be loaded (specified by ‘Wb’ or ‘f’)

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand which is a register, ‘Wb’, without any address modifier
- The second source operand which is a literal value
- The destination of the result (only if not the same as the first source operand) which is typically a register, ‘Wd’, with or without an address modifier

The control instructions may use some of the following operands:

- A program memory address
- The mode of the table read and table write instructions

All instructions are a single word, except for certain double-word instructions, which were made double-word instructions so that all of the required information is available in these 48 bits. In the second word, the 8 MSbs are ‘0’s. If this second word is executed as an instruction (by itself), it will execute as a NOP.

Most single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the Program Counter (PC) is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP. Notable exceptions are the BRA (unconditional/computed branch), indirect CALL/GOTO, all table reads and writes, and RETURN/RETFIE instructions, which are single-word instructions but take two or three cycles.

Certain instructions that involve skipping over the subsequent instruction require either two or three cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction. Moreover, double-word moves require two cycles. The double-word instructions execute in two instruction cycles.

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

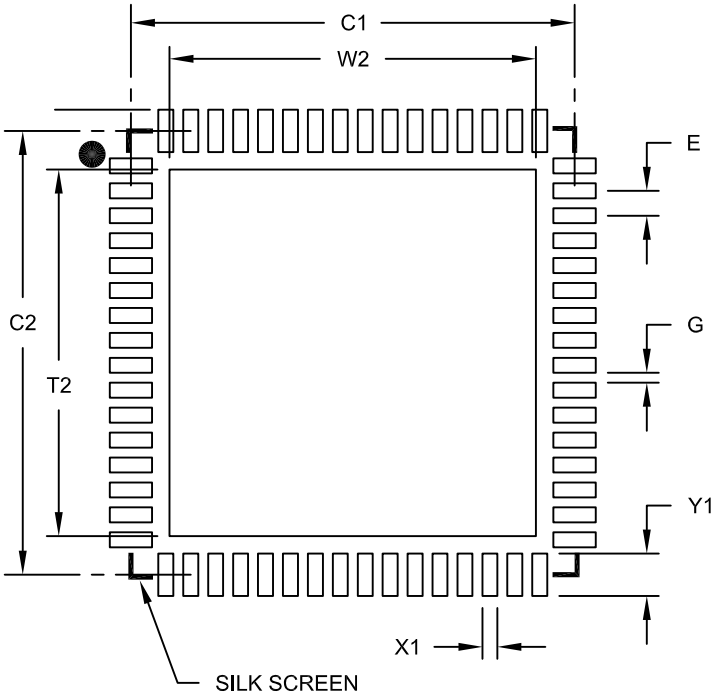
Assembly Mnemonic	Assembly Syntax		Description	# of Words	# of Cycles	Status Flags Affected
GOTO	GOTO	Expr	Go to Address	2	2	None
	GOTO	Wn	Go to Indirect	1	2	None
INC	INC	f	f = f + 1	1	1	C, DC, N, OV, Z
	INC	f,WREG	WREG = f + 1	1	1	C, DC, N, OV, Z
	INC	Ws,Wd	Wd = Ws + 1	1	1	C, DC, N, OV, Z
INC2	INC2	f	f = f + 2	1	1	C, DC, N, OV, Z
	INC2	f,WREG	WREG = f + 2	1	1	C, DC, N, OV, Z
	INC2	Ws,Wd	Wd = Ws + 2	1	1	C, DC, N, OV, Z
IOR	IOR	f	f = f .IOR. WREG	1	1	N, Z
	IOR	f,WREG	WREG = f .IOR. WREG	1	1	N, Z
	IOR	#lit10,Wn	Wd = lit10 .IOR. Wd	1	1	N, Z
	IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1	N, Z
	IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1	N, Z
LNK	LNK	#lit14	Link Frame Pointer	1	1	None
LSR	LSR	f	f = Logical Right Shift f	1	1	C, N, OV, Z
	LSR	f,WREG	WREG = Logical Right Shift f	1	1	C, N, OV, Z
	LSR	Ws,Wd	Wd = Logical Right Shift Ws	1	1	C, N, OV, Z
	LSR	Wb,Wns,Wnd	Wnd = Logical Right Shift Wb by Wns	1	1	N, Z
	LSR	Wb,#lit5,Wnd	Wnd = Logical Right Shift Wb by lit5	1	1	N, Z
MOV	MOV	f,Wn	Move f to Wn	1	1	None
	MOV	[Wns+Slit10],Wnd	Move [Wns+Slit10] to Wnd	1	1	None
	MOV	f	Move f to f	1	1	N, Z
	MOV	f,WREG	Move f to WREG	1	1	N, Z
	MOV	#lit16,Wn	Move 16-Bit Literal to Wn	1	1	None
	MOV.b	#lit8,Wn	Move 8-Bit Literal to Wn	1	1	None
	MOV	Wn,f	Move Wn to f	1	1	None
	MOV	Wns,[Wns+Slit10]	Move Wns to [Wns+Slit10]	1	1	
	MOV	Wso,Wdo	Move Ws to Wd	1	1	None
	MOV	WREG,f	Move WREG to f	1	1	N, Z
	MOV.D	Wns,Wd	Move Double from W(ns):W(ns+1) to Wd	1	2	None
	MOV.D	Ws,Wnd	Move Double from Ws to W(nd+1):W(nd)	1	2	None
MUL	MUL.SS	Wb,Ws,Wnd	{Wnd+1, Wnd} = Signed(Wb) * Signed(Ws)	1	1	None
	MUL.SU	Wb,Ws,Wnd	{Wnd+1, Wnd} = Signed(Wb) * Unsigned(Ws)	1	1	None
	MUL.US	Wb,Ws,Wnd	{Wnd+1, Wnd} = Unsigned(Wb) * Signed(Ws)	1	1	None
	MUL.UU	Wb,Ws,Wnd	{Wnd+1, Wnd} = Unsigned(Wb) * Unsigned(Ws)	1	1	None
	MUL.SU	Wb,#lit5,Wnd	{Wnd+1, Wnd} = Signed(Wb) * Unsigned(lit5)	1	1	None
	MUL.UU	Wb,#lit5,Wnd	{Wnd+1, Wnd} = Unsigned(Wb) * Unsigned(lit5)	1	1	None
	MUL	f	W3:W2 = f * WREG	1	1	None
NEG	NEG	f	f = $\overline{f}$ + 1	1	1	C, DC, N, OV, Z
	NEG	f,WREG	WREG = $\overline{f}$ + 1	1	1	C, DC, N, OV, Z
	NEG	Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1	C, DC, N, OV, Z
NOP	NOP		No Operation	1	1	None
	NOPR		No Operation	1	1	None
POP	POP	f	Pop f from Top-of-Stack (TOS)	1	1	None
	POP	Wdo	Pop from Top-of-Stack (TOS) to Wdo	1	1	None
	POP.D	Wnd	Pop from Top-of-Stack (TOS) to W(nd):W(nd+1)	1	2	None
	POP.S		Pop Shadow Registers	1	1	All
PUSH	PUSH	f	Push f to Top-of-Stack (TOS)	1	1	None
	PUSH	Wso	Push Wso to Top-of-Stack (TOS)	1	1	None
	PUSH.D	Wns	Push W(ns):W(ns+1) to Top-of-Stack (TOS)	1	2	None
	PUSH.S		Push Shadow Registers	1	1	None

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Assembly Mnemonic	Assembly Syntax		Description	# of Words	# of Cycles	Status Flags Affected
TBLRDL	TBLRDL	Ws,Wd	Read Prog<15:0> to Wd	1	2	None
TBLWTH	TBLWTH	Ws,Wd	Write Ws<7:0> to Prog<23:16>	1	2	None
TBLWTL	TBLWTL	Ws,Wd	Write Ws to Prog<15:0>	1	2	None
ULNK	ULNK		Unlink Frame Pointer	1	1	None
XOR	XOR	f	f = f .XOR. WREG	1	1	N, Z
	XOR	f,WREG	WREG = f .XOR. WREG	1	1	N, Z
	XOR	#lit10,Wn	Wd = lit10 .XOR. Wd	1	1	N, Z
	XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1	N, Z
	XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1	N, Z
ZE	ZE	Ws,Wnd	Wnd = Zero-Extend Ws	1	1	C, Z, N

64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]
With 0.40 mm Contact Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	W2			7.35
Optional Center Pad Length	T2			7.35
Contact Pad Spacing	C1		8.90	
Contact Pad Spacing	C2		8.90	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			0.85
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2149A

P

R