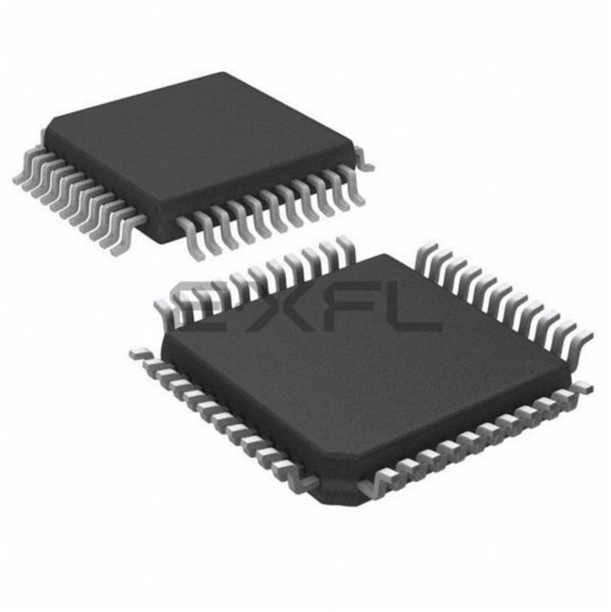


Welcome to E-XFL.COM

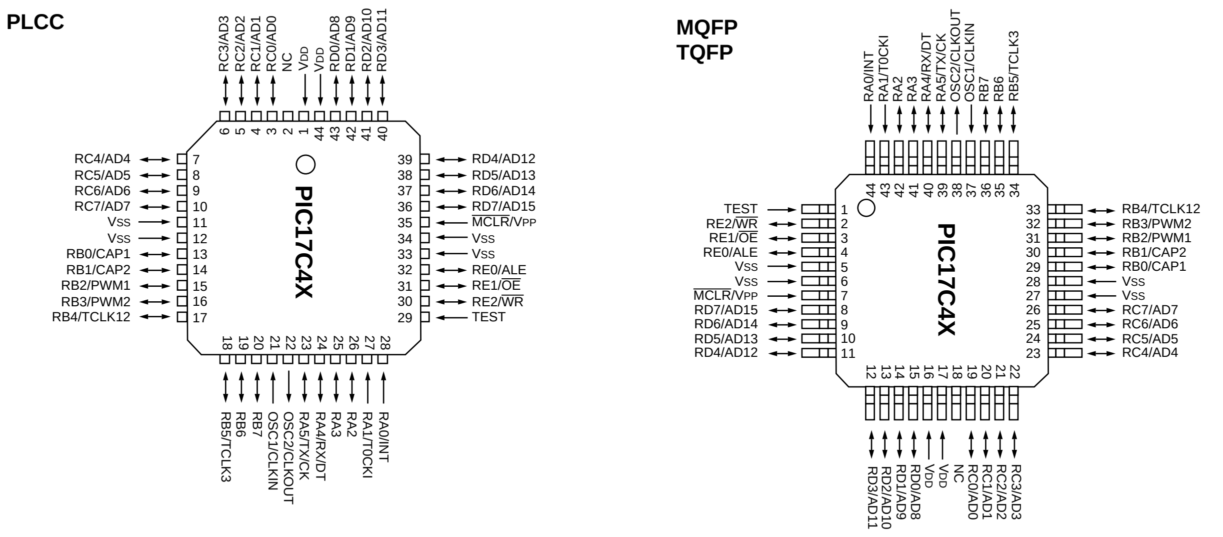
## What is "Embedded - Microcontrollers"?

"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 | Obsolete |
| Core Processor | PIC |
| Core Size | 8-Bit |
| Speed | 25MHz |
| Connectivity | UART/USART |
| Peripherals | POR, PWM, WDT |
| Number of I/O | 33 |
| Program Memory Size | 8KB (4K x 16) |
| Program Memory Type | OTP |
| EEPROM Size | - |
| RAM Size | 454 x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.5V ~ 6V |
| Data Converters | - |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-QFP |
| Supplier Device Package | 44-MQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic17c43t-25i-pq |

## Pin Diagrams Cont.'d

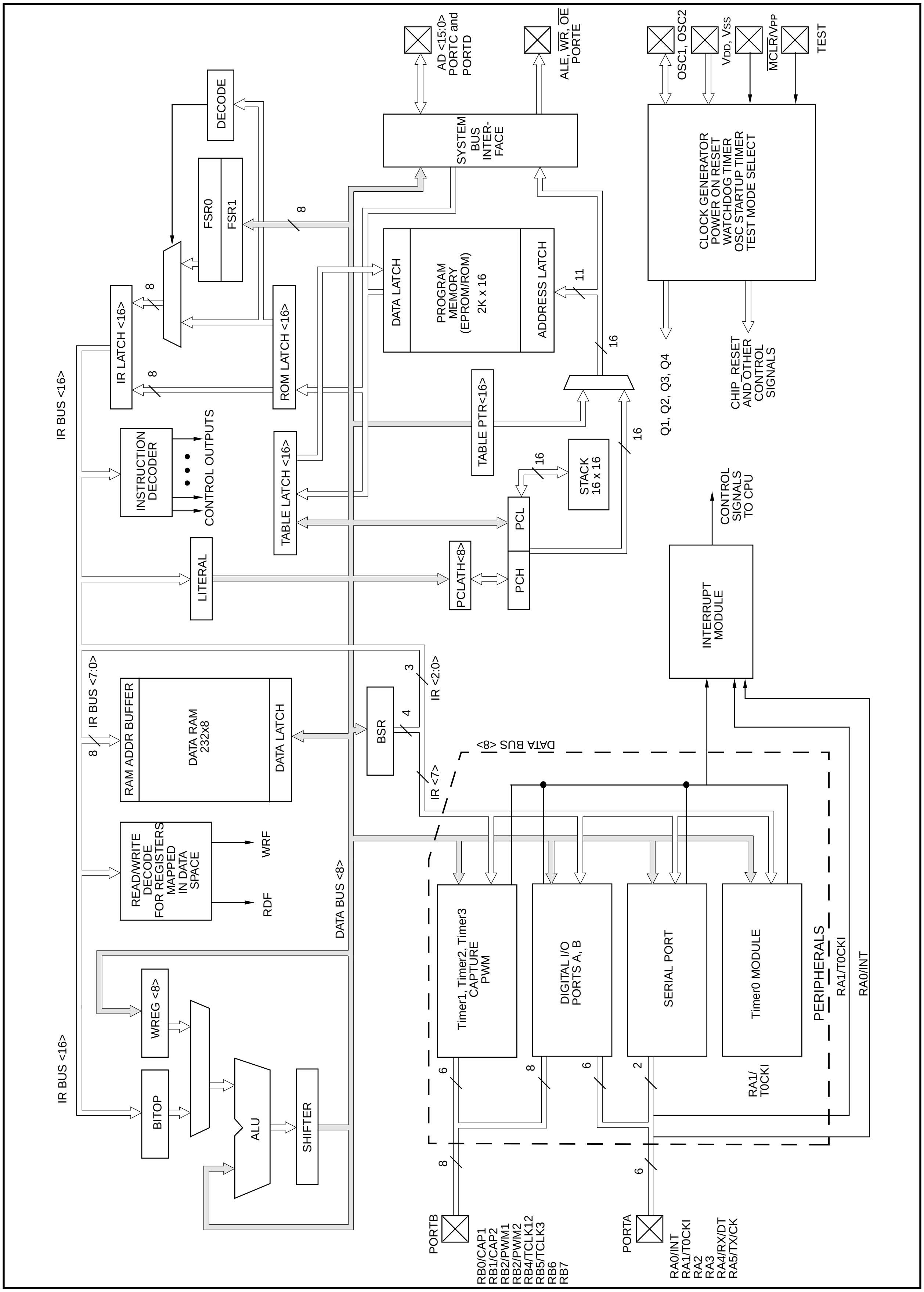


All devices are available in all package types, listed in Section 21.0, with the following exceptions:

- ROM devices are not available in Windowed CERDIP Packages
- TQFP is not available for the PIC17C42.

NOTES:

**FIGURE 3-1: PIC17C42 BLOCK DIAGRAM**



## 5.1 Interrupt Status Register (INTSTA)

The Interrupt Status/Control register (INTSTA) records the individual interrupt requests in flag bits, and contains the individual interrupt enable bits (not for the peripherals).

The PEIF bit is a read only, bit wise OR of all the peripheral flag bits in the PIR register (Figure 5-4).

> **Note:** T0IF, INTF, T0CKIF, or PEIF will be set by the specified condition, even if the corresponding interrupt enable bit is clear (interrupt disabled) or the GLINTD bit is set (all interrupts disabled).

Care should be taken when clearing any of the INTSTA register enable bits when interrupts are enabled (GLINTD is clear). If any of the INTSTA flag bits (T0IF, INTF, T0CKIF, or PEIF) are set in the same instruction cycle as the corresponding interrupt enable bit is cleared, the device will vector to the reset address (0x00).

When disabling any of the INTSTA enable bits, the GLINTD bit should be set (disabled).

### FIGURE 5-2: INTSTA REGISTER (ADDRESS: 07h, UNBANKED)

| R - 0 | R/W - 0 | R/W - 0 | R/W - 0 | R/W - 0 | R/W - 0 | R/W - 0 | R/W - 0 |
|---|---|---|---|---|---|---|---|
| PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
- n = Value at POR reset

bit 7: **PEIF**: Peripheral Interrupt Flag bit
This bit is the OR of all peripheral interrupt flag bits AND'ed with their corresponding enable bits.
1 = A peripheral interrupt is pending
0 = No peripheral interrupt is pending

bit 6: **T0CKIF**: External Interrupt on T0CKI Pin Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to vector (18h).
1 = The software specified edge occurred on the RA1/T0CKI pin
0 = The software specified edge did not occur on the RA1/T0CKI pin

bit 5: **T0IF**: TMR0 Overflow Interrupt Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to vector (10h).
1 = TMR0 overflowed
0 = TMR0 did not overflow

bit 4: **INTF**: External Interrupt on INT Pin Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to vector (08h).
1 = The software specified edge occurred on the RA0/INT pin
0 = The software specified edge did not occur on the RA0/INT pin

bit 3: **PEIE**: Peripheral Interrupt Enable bit
This bit enables all peripheral interrupts that have their corresponding enable bits set.
1 = Enable peripheral interrupts
0 = Disable peripheral interrupts

bit 2: **T0CKIE**: External Interrupt on T0CKI Pin Enable bit
1 = Enable software specified edge interrupt on the RA1/T0CKI pin
0 = Disable interrupt on the RA1/T0CKI pin

bit 1: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enable TMR0 overflow interrupt
0 = Disable TMR0 overflow interrupt

bit 0: **INTE**: External Interrupt on RA0/INT Pin Enable bit
1 = Enable software specified edge interrupt on the RA0/INT pin
0 = Disable software specified edge interrupt on the RA0/INT pin

### 6.8 Bank Select Register (BSR)

The BSR is used to switch between banks in the data memory area (Figure 6-13). In the PIC17C42, PIC17CR42, and PIC17C42A only the lower nibble is implemented. While in the PIC17C43, PIC17CR43, and PIC17C44 devices, the entire byte is implemented. The lower nibble is used to select the peripheral register bank. The upper nibble is used to select the general purpose memory bank.

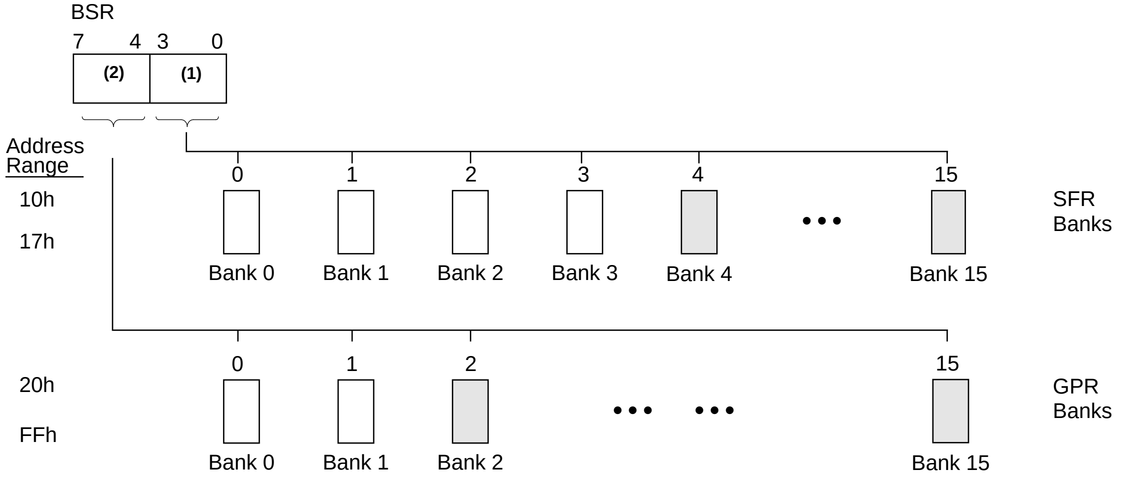
All the Special Function Registers (SFRs) are mapped into the data memory space. In order to accommodate the large number of registers, a banking scheme has been used. A segment of the SFRs, from address 10h to address 17h, is banked. The lower nibble of the bank select register (BSR) selects the currently active "peripheral bank." Effort has been made to group the peripheral registers of related functionality in one bank. However, it will still be necessary to switch from bank to bank in order to address all peripherals related to a single task. To assist this, a `MOVLB bank` instruction is in the instruction set.

For the PIC17C43, PIC17CR43, and PIC17C44 devices, the need for a large general purpose memory space dictated a general purpose RAM banking scheme. The upper nibble of the BSR selects the currently active general purpose RAM bank. To assist this, a `MOVLR bank` instruction has been provided in the instruction set.

If the currently selected bank is not implemented (such as Bank 13), any read will read all '0's. Any write is completed to the bit bucket and the ALU status bits will be set/cleared as appropriate.

**Note:** Registers in Bank 15 in the Special Function Register area, are reserved for Microchip use. Reading of registers in this bank may cause random values to be read.

**FIGURE 6-13: BSR OPERATION (PIC17C43/R43/44)**



Note 1: Only Banks 0 through Bank 3 are implemented. Selection of an unimplemented bank is not recommended. **Bank 15 is reserved for Microchip use, reading of registers in this bank may cause random values to be read.**

2: Only Banks 0 and Bank 1 are implemented. Selection of an unimplemented bank is not recommended.

## 7.1 Table Writes to Internal Memory

A table write operation to internal memory causes a long write operation. The long write is necessary for programming the internal EPROM. Instruction execution is halted while in a long write cycle. The long write will be terminated by any enabled interrupt. To ensure that the EPROM location has been well programmed, a minimum programming time is required (see specification #D114 ). Having only one interrupt enabled to terminate the long write ensures that no unintentional interrupts will prematurely terminate the long write.

The sequence of events for programming an internal program memory location should be:

1. Disable all interrupt sources, except the source to terminate EPROM program write.
2. Raise $\overline{\text{MCLR}}$/VPP pin to the programming voltage.
3. Clear the WDT.
4. Do the table write. The interrupt will terminate the long write.
5. Verify the memory location (table read).

**Note:** Programming requirements must be met. See timing specification in electrical specifications for the desired device. Violating these specifications (including temperature) may result in EPROM locations that are not fully programmed and may lose their state over time.

### 7.1.1 TERMINATING LONG WRITES

An interrupt source or reset are the only events that terminate a long write operation. Terminating the long write from an interrupt source requires that the interrupt enable and flag bits are set. The GLINTD bit only enables the vectoring to the interrupt address.

If the T0CKI, RA0/INT, or TMR0 interrupt source is used to terminate the long write; the interrupt flag, of the highest priority enabled interrupt, will terminate the long write and automatically be cleared.

**Note 1:** If an interrupt is pending, the TABLWT is aborted (an NOP is executed). The highest priority pending interrupt, from the T0CKI, RA0/INT, or TMR0 sources that is enabled, has its flag cleared.

**Note 2:** If the interrupt is not being used for the program write timing, the interrupt should be disabled. This will ensure that the interrupt is not lost, nor will it terminate the long write prematurely.

If a peripheral interrupt source is used to terminate the long write, the interrupt enable and flag bits must be set. The interrupt flag will not be automatically cleared upon the vectoring to the interrupt vector address.

If the GLINTD bit is cleared prior to the long write, when the long write is terminated, the program will branch to the interrupt vector.

If the GLINTD bit is set prior to the long write, when the long write is terminated, the program will not vector to the interrupt address.

**TABLE 7-1: INTERRUPT - TABLE WRITE INTERACTION**

| Interrupt Source | GLINTD | Enable Bit | Flag Bit | Action |
|---|---|---|---|---|
| RA0/INT, TMR0, T0CKI | 0 | 1 | 1 | Terminate long table write (to internal program memory), branch to interrupt vector (branch clears flag bit). |
| | 0 | 1 | 0 | None |
| | 1 | 0 | x | None |
| | 1 | 1 | 1 | Terminate table write, do not branch to interrupt vector (flag is automatically cleared). |
| Peripheral | 0 | 1 | 1 | Terminate table write, branch to interrupt vector. |
| | 0 | 1 | 0 | None |
| | 1 | 0 | x | None |
| | 1 | 1 | 1 | Terminate table write, do not branch to interrupt vector (flag is set). |

Example 8-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers RES3:RES0.

### EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

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

### EXAMPLE 8-3: 16 x 16 MULTIPLY ROUTINE

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L       ; ARG1L * ARG2L ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H       ; ARG1H * ARG2H ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H       ; ARG1L * ARG2H ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L       ; ARG1H * ARG2L ->
                        ;   PRODH:PRODL

   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
```

## 9.3 PORTC and DDRC Registers

PORTC is an 8-bit bi-directional port. The corresponding data direction register is DDRC. A '1' in DDRC configures the corresponding port pin as an input. A '0' in the DDRC register configures the corresponding port pin as an output. Reading PORTC reads the status of the pins, whereas writing to it will write to the port latch. PORTC is multiplexed with the system bus. When operating as the system bus, PORTC is the low order byte of the address/data bus (AD7:AD0). The timing for the system bus is shown in the Electrical Characteristics section.

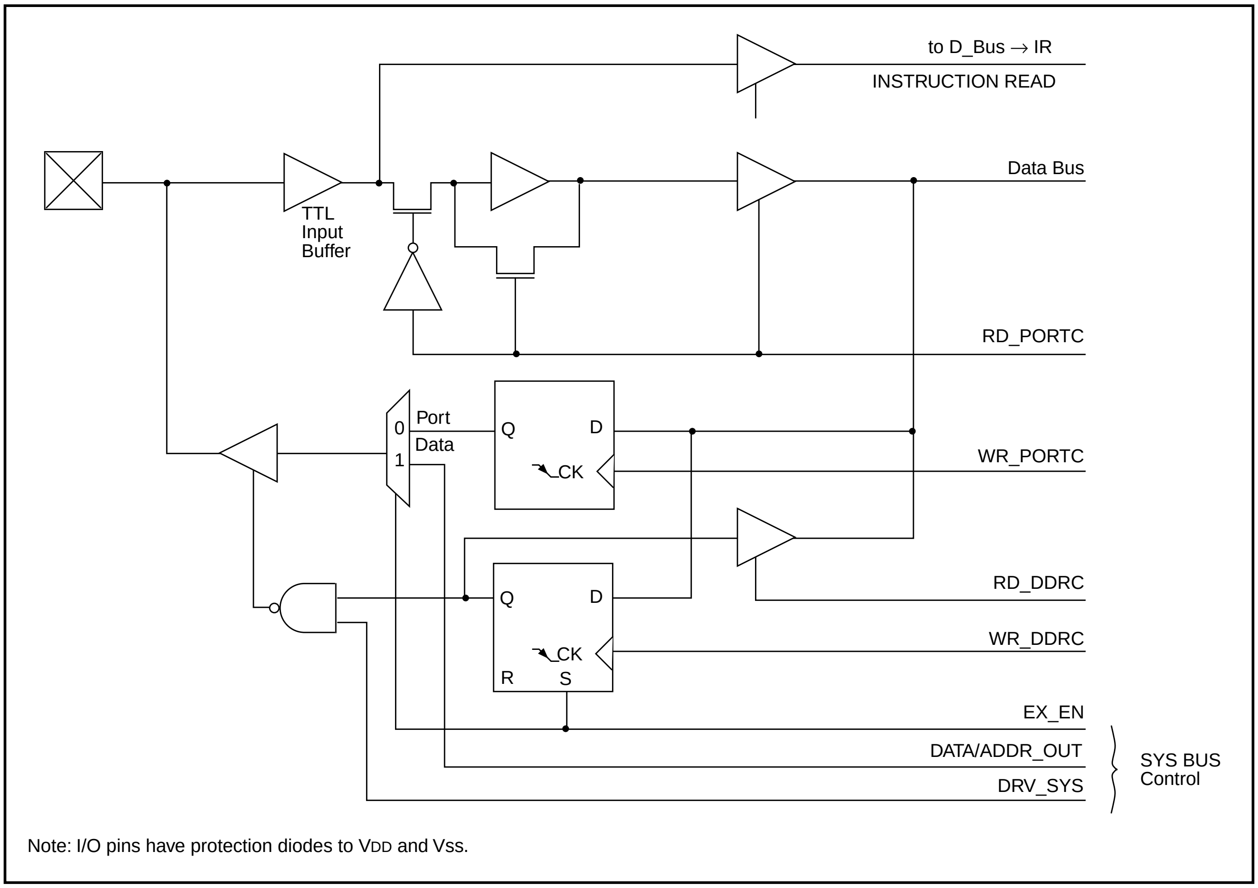
| **Note:** | This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O. |
|---|---|

Example 9-2 shows the instruction sequence to initialize PORTC. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized.

**EXAMPLE 9-2: INITIALIZING PORTC**

```
MOVLB  1             ;  Select Bank 1
CLRF   PORTC         ;  Initialize PORTC data
                     ;   latches before setting
                     ;   the data direction
                     ;   register
MOVLW  0xCF          ;  Value used to initialize
                     ;   data direction
MOVWF  DDRC          ;  Set RC<3:0> as inputs
                     ;   RC<5:4> as outputs
                     ;   RC<7:6> as inputs
```

**FIGURE 9-6: BLOCK DIAGRAM OF RC<7:0> PORT PINS**



Note: I/O pins have protection diodes to VDD and Vss.

#### 12.1.3.1 PWM PERIODS

The period of the PWM1 output is determined by Timer1 and its period register (PR1). The period of the PWM2 output can be software configured to use either Timer1 or Timer2 as the time-base. When TM2PW2 bit (PW2DCL<5>) is clear, the time-base is determined by TMR1 and PR1. When TM2PW2 is set, the time-base is determined by Timer2 and PR2.

Running two different PWM outputs on two different timers allows different PWM periods. Running both PWMs from Timer1 allows the best use of resources by freeing Timer2 to operate as an 8-bit timer. Timer1 and Timer2 can not be used as a 16-bit timer if either PWM is being used.

The PWM periods can be calculated as follows:

period of PWM1 =[(PR1) + 1] x 4TOSC

period of PWM2 =[(PR1) + 1] x 4TOSC or
[(PR2) + 1] x 4TOSC

The duty cycle of PWMx is determined by the 10-bit value DCx<9:0>. The upper 8-bits are from register PWxDCH and the lower 2-bits are from PWxDCL<7:6> (PWxDCH:PWxDCL<7:6>). Table 12-3 shows the maximum PWM frequency (FPWM) given the value in the period register.

The number of bits of resolution that the PWM can achieve depends on the operation frequency of the device as well as the PWM frequency (FPWM).

Maximum PWM resolution (bits) for a given PWM frequency:

$$= \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \quad \text{bits}$$

The PWMx duty cycle is as follows:

PWMx Duty Cycle = (DCx) x TOSC

where DCx represents the 10-bit value from PWxDCH:PWxDCL.

If DCx = 0, then the duty cycle is zero. If PRx = PWxDCH, then the PWM output will be low for one to four Q-clock (depending on the state of the PWxDCL<7:6> bits). For a Duty Cycle to be 100%, the PWxDCH value must be greater then the PRx value.

The duty cycle registers for both PWM outputs are double buffered. When the user writes to these registers, they are stored in master latches. When TMR1 (or TMR2) overflows and a new PWM period begins, the master latch values are transferred to the slave latches and the PWMx pin is forced high.

**Note:** For PW1DCH, PW1DCL, PW2DCH and PW2DCL registers, a write operation writes to the "master latches" while a read operation reads the "slave latches". As a result, the user may not read back what was just written to the duty cycle registers.

The user should also avoid any "read-modify-write" operations on the duty cycle registers, such as: `ADDWF PW1DCH`. This may cause duty cycle outputs that are unpredictable.

**TABLE 12-3: PWM FREQUENCY vs. RESOLUTION AT 25 MHz**

| PWM Frequency | Frequency (kHz) | | | | |
|---|---|---|---|---|---|
| | **24.4** | **48.8** | **65.104** | **97.66** | **390.6** |
| PRx Value | 0xFF | 0x7F | 0x5F | 0x3F | 0x0F |
| High Resolution | 10-bit | 9-bit | 8.5-bit | 8-bit | 6-bit |
| Standard Resolution | 8-bit | 7-bit | 6.5-bit | 6-bit | 4-bit |

#### 12.1.3.2 PWM INTERRUPTS

The PWM module makes use of TMR1 or TMR2 interrupts. A timer interrupt is generated when TMR1 or TMR2 equals its period register and is cleared to zero. This interrupt also marks the beginning of a PWM cycle. The user can write new duty cycle values before the timer roll-over. The TMR1 interrupt is latched into the TMR1IF bit and the TMR2 interrupt is latched into the TMR2IF bit. These flags must be cleared in software.

#### 12.1.3.3 EXTERNAL CLOCK SOURCE

The PWMs will operate regardless of the clock source of the timer. The use of an external clock has ramifications that must be understood. Because the external TCLK12 input is synchronized internally (sampled once per instruction cycle), the time TCLK12 changes to the time the timer increments will vary by as much as TCY (one instruction cycle). This will cause jitter in the duty cycle as well as the period of the PWM output.

This jitter will be ±TCY, unless the external clock is synchronized with the processor clock. Use of one of the PWM outputs as the clock source to the TCLKx input, will supply a synchronized clock.

In general, when using an external clock source for PWM, its frequency should be much less than the device frequency (Fosc).

## 13.4 USART Synchronous Slave Mode

The synchronous slave mode differs from the master mode in the fact that the shift clock is supplied externally at the RA5/TX/CK pin (instead of being supplied internally in the master mode). This allows the device to transfer or receive data in the SLEEP mode. The slave mode is entered by clearing the CSRC (TXSTA<7>) bit.

### 13.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the sync master and slave modes are identical except in the case of the SLEEP mode.

If two words are written to TXREG and then the SLEEP instruction executes, the following will occur. The first word will immediately transfer to the TSR and will transmit as the shift clock is supplied. The second word will remain in TXREG. TXIF will not be set. When the first word has been shifted out of TSR, TXREG will transfer the second word to the TSR and the TXIF flag will now be set. If TXIE is enabled, the interrupt will wake the chip from SLEEP and if the global interrupt is enabled, then the program will branch to interrupt vector (0020h).

Steps to follow when setting up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting the SYNC and SPEN bits and clearing the CSRC bit.
2. Clear the CREN bit.
3. If interrupts are desired, then set the TXIE bit.
4. If 9-bit transmission is desired, then set the TX9 bit.
5. Start transmission by loading data to TXREG.
6. If 9-bit transmission is selected, the ninth bit should be loaded in TX9D.
7. Enable the transmission by setting TXEN.

Writing the transmit data to the TXREG, then enabling the transmit (setting TXEN) allows transmission to start sooner then doing these two events in the reverse order.

| **Note:** | To terminate a transmission, either clear the SPEN bit, or the TXEN bit. This will reset the transmit logic, so that it will be in the proper state when transmit is re-enabled. |
|---|---|

### 13.4.2 USART SYNCHRONOUS SLAVE RECEPTION

Operation of the synchronous master and slave modes are identical except in the case of the SLEEP mode. Also, SREN is a don't care in slave mode.

If receive is enabled (CREN) prior to the SLEEP instruction, then a word may be received during SLEEP. On completely receiving the word, the RSR will transfer the data to RCREG (setting RCIF) and if the RCIE bit is set, the interrupt generated will wake the chip from SLEEP. If the global interrupt is enabled, the program will branch to the interrupt vector (0020h).

Steps to follow when setting up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting the SYNC and SPEN bits and clearing the CSRC bit.
2. If interrupts are desired, then set the RCIE bit.
3. If 9-bit reception is desired, then set the RX9 bit.
4. To enable reception, set the CREN bit.
5. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
6. Read RCSTA to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading RCREG.
8. If any error occurred, clear the error by clearing the CREN bit.

| **Note:** | To abort reception, either clear the SPEN bit, the SREN bit (when in single receive mode), or the CREN bit (when in continuous receive mode). This will reset the receive logic, so that it will be in the proper state when receive is re-enabled. |
|---|---|

### MOVLR — Move Literal to high nibble in BSR

Syntax: [ *label* ] MOVLR k

Operands: 0 ≤ k ≤ 15

Operation: k → (BSR<7:4>)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| `1011` | `101x` | `kkkk` | `uuuu` |

Description: The 4-bit literal 'k' is loaded into the most significant 4-bits of the Bank Select Register (BSR). Only the high 4-bits of the Bank Select Register are affected. The lower half of the BSR is unchanged. The assembler will encode the “u” fields as 0.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k:u' | Execute | Write literal 'k' to BSR<7:4> |

Example: `MOVLR 5`

Before Instruction
BSR register = 0x22

After Instruction
BSR register = 0x52

**Note:** This instruction is not available in the PIC17C42 device.

### MOVLW — Move Literal to WREG

Syntax: [ *label* ] MOVLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (WREG)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| `1011` | `0000` | `kkkk` | `kkkk` |

Description: The eight bit literal 'k' is loaded into WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Execute | Write to WREG |

Example: `MOVLW 0x5A`

After Instruction
WREG = 0x5A

| | | | | | | |
|---|---|---|---|---|---|---|
| **Applicable Devices** | 42 | R42 | 42A | 43 | R43 | 44 |

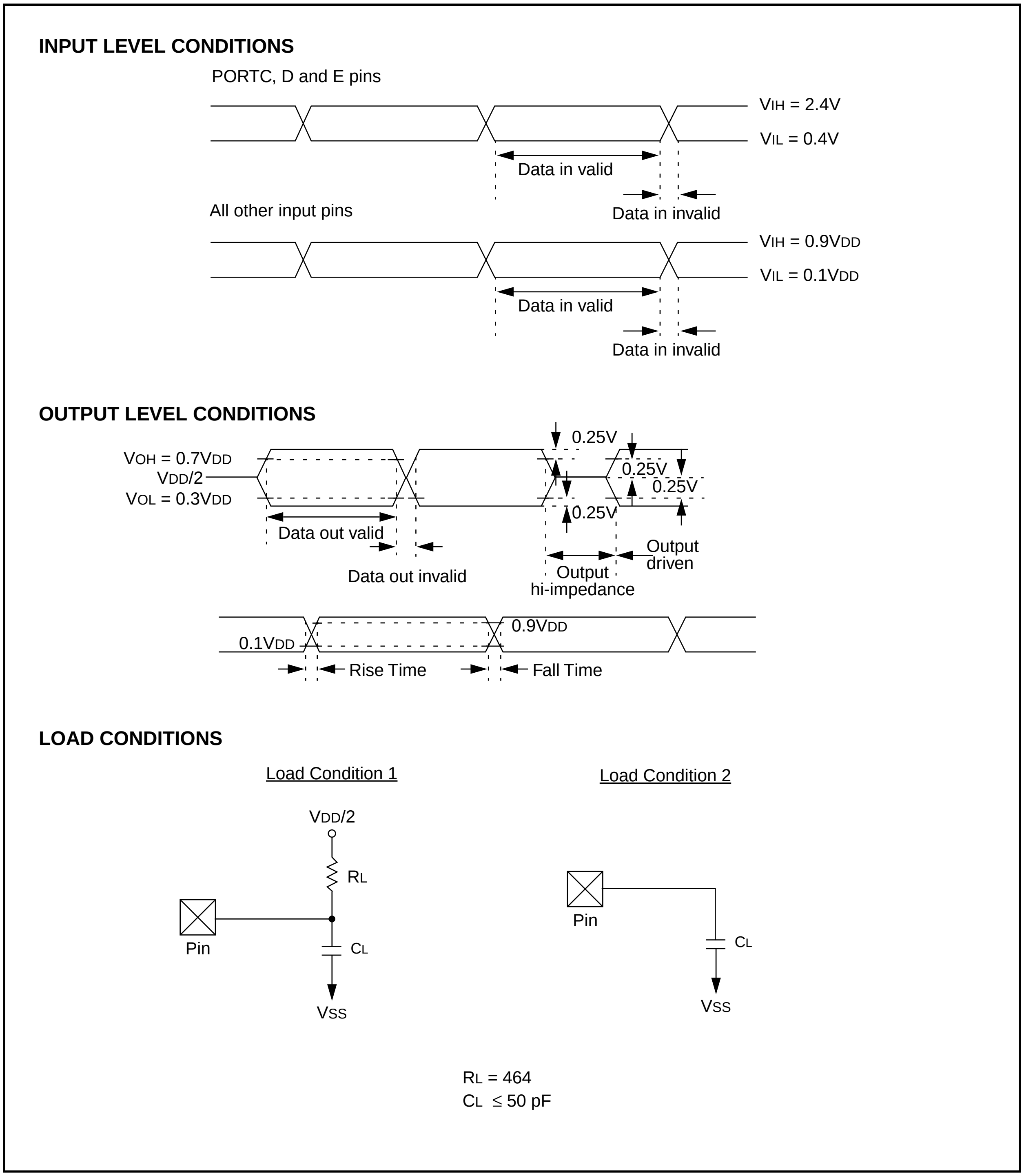
**TABLE 17-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)**

| OSC | PIC17C42-16 | PIC17C42-25 |
|---|---|---|
| RC | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 6 mA max.<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 4 MHz max. | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 6 mA max.<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 4 MHz max. |
| XT | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 24 mA max.<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 16 MHz max. | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 38 mA max.<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 25 MHz max. |
| EC | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 24 mA max.<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 16 MHz max. | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 38 mA max.<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 25 MHz max. |
| LF | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 150 μA max. at 32 kHz (WDT enabled)<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 2 MHz max. | $V_{DD}$: 4.5V to 5.5V<br>$I_{DD}$: 150 μA max. at 32 kHz (WDT enabled)<br>$I_{PD}$: 5 μA max. at 5.5V (WDT disabled)<br>Freq: 2 MHz max. |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Applicable Devices** | 42 | R42 | 42A | 43 | R43 | 44 |

### FIGURE 17-1: PARAMETER MEASUREMENT INFORMATION

All timings are measure between high and low measurement points as indicated in the figures below.



| Applicable Devices | 42 | R42 | 42A | 43 | R43 | 44 |
|---|---|---|---|---|---|---|

## 19.2 DC CHARACTERISTICS: PIC17LC42A/43/LC44 (Commercial, Industrial) PIC17LCR42/43 (Commercial, Industrial)

| DC CHARACTERISTICS | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature<br>-40˚C ≤ $T_A$ ≤ +85˚C for industrial and<br>0˚C ≤ $T_A$ ≤ +70˚C for commercial | | | | |
|---|---|---|---|---|---|---|---|
| **Parameter No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| D001 | $V_{DD}$ | Supply Voltage | 2.5 | – | 6.0 | V | |
| D002 | $V_{DR}$ | RAM Data Retention Voltage (Note 1) | 1.5 * | – | – | V | Device in SLEEP mode |
| D003 | $V_{POR}$ | $V_{DD}$ start voltage to ensure internal Power-on Reset signal | – | $V_{SS}$ | – | V | See section on Power-on Reset for details |
| D004 | $S_{VDD}$ | $V_{DD}$ rise rate to ensure internal Power-on Reset signal | 0.060 * | – | – | mV/ms | See section on Power-on Reset for details |
| D010<br>D011<br>D014 | $I_{DD}$ | Supply Current (Note 2) | –<br>–<br>– | 3<br>6<br>95 | 6<br>12 *<br>150 | mA<br>mA<br>µA | $F_{OSC}$ = 4 MHz (Note 4)<br>$F_{OSC}$ = 8 MHz<br>$F_{OSC}$ = 32 kHz, WDT disabled (EC osc configuration) |
| D020<br>D021 | $I_{PD}$ | Power-down Current (Note 3) | –<br>– | 10<br>< 1 | 40<br>5 | µA<br>µA | $V_{DD}$ = 5.5V, WDT enabled<br>$V_{DD}$ = 5.5V, WDT disabled |

* 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.

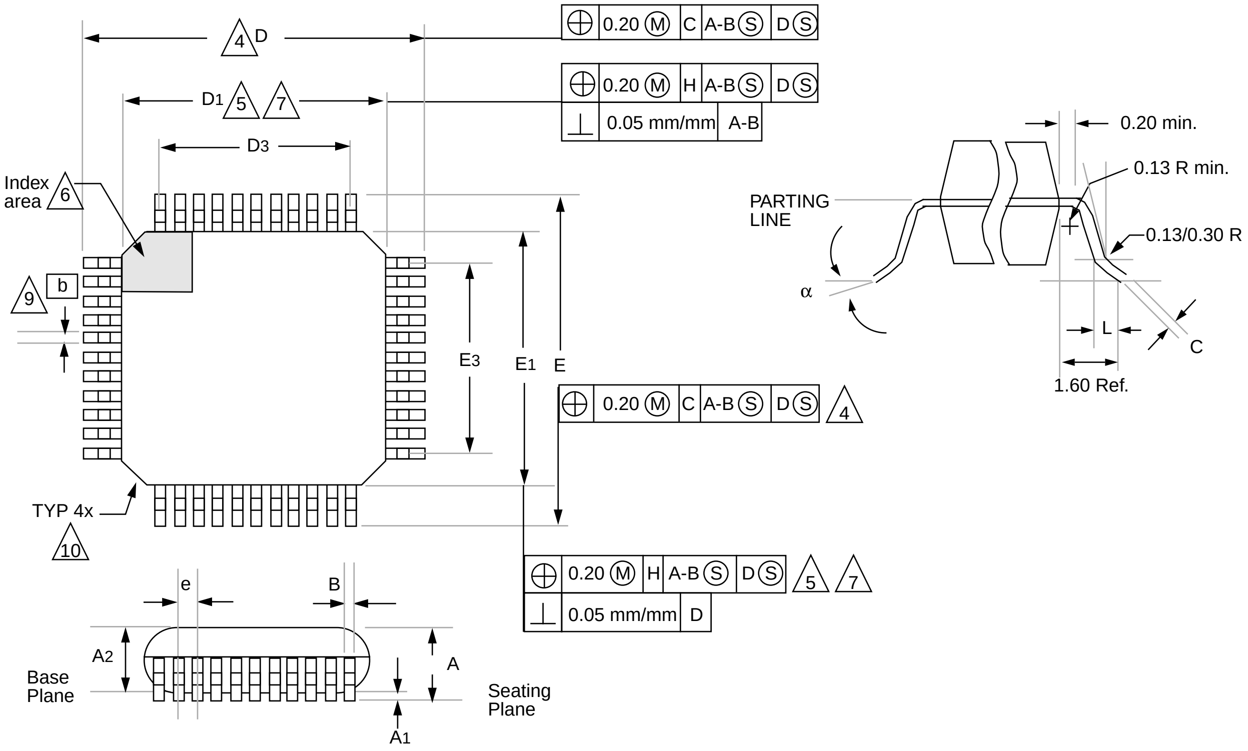
Note 1: This is the limit to which $V_{DD}$ can be lowered in SLEEP mode without losing RAM data.

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:
OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to $V_{DD}$ or $V_{SS}$, T0CKI = $V_{DD}$, $\overline{MCLR}$ = $V_{DD}$; WDT enabled/disabled as specified.
Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.
For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as: $V_{DD}$ / (2 • R).
For capacitive loads, the current can be estimated (for an individual I/O pin) as ($C_L$ • $V_{DD}$) • f
$C_L$ = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.
The capacitive currents are most significant when the device is configured for external execution (includes extended microcontroller mode).

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.

## 21.4 44-Lead Plastic Surface Mount (MQFP 10x10 mm Body 1.6/0.15 mm Lead Form)



| Package Group: Plastic MQFP | | | | | | |
|---|---|---|---|---|---|---|
| | Millimeters | | | Inches | | |
| Symbol | Min | Max | Notes | Min | Max | Notes |
| α | 0° | 7° | | 0° | 7° | |
| A | 2.000 | 2.350 | | 0.078 | 0.093 | |
| A1 | 0.050 | 0.250 | | 0.002 | 0.010 | |
| A2 | 1.950 | 2.100 | | 0.768 | 0.083 | |
| b | 0.300 | 0.450 | **Typical** | 0.011 | 0.018 | **Typical** |
| C | 0.150 | 0.180 | | 0.006 | 0.007 | |
| D | 12.950 | 13.450 | | 0.510 | 0.530 | |
| D1 | 9.900 | 10.100 | | 0.390 | 0.398 | |
| D3 | 8.000 | 8.000 | **Reference** | 0.315 | 0.315 | **Reference** |
| E | 12.950 | 13.450 | | 0.510 | 0.530 | |
| E1 | 9.900 | 10.100 | | 0.390 | 0.398 | |
| E3 | 8.000 | 8.000 | **Reference** | 0.315 | 0.315 | **Reference** |
| e | 0.800 | 0.800 | | 0.031 | 0.032 | |
| L | 0.730 | 1.030 | | 0.028 | 0.041 | |
| N | 44 | 44 | | 44 | 44 | |
| CP | 0.102 | – | | 0.004 | – | |

**Note the following details of the code protection feature on PICmicro® MCUs.**

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

**Trademarks**

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

♻ Printed on recycled paper.

DNV Certification, Inc.
USA

ANSI★RAB
QMS
ACCREDITED

DNV

DNV MSC
The Netherlands
Accredited by the RvA

ISO 9001 / QS-9000
REGISTERED FIRM

*Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.*