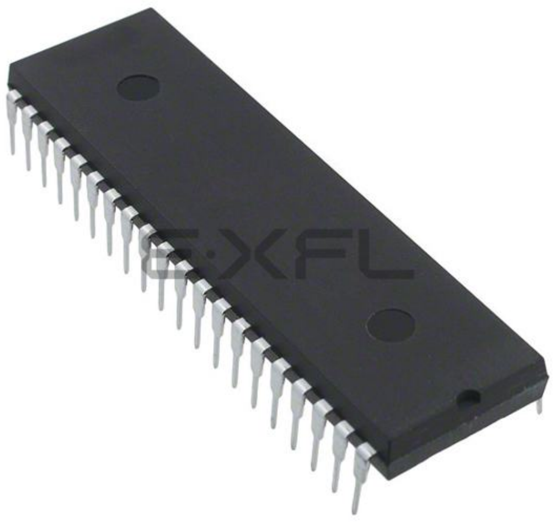


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

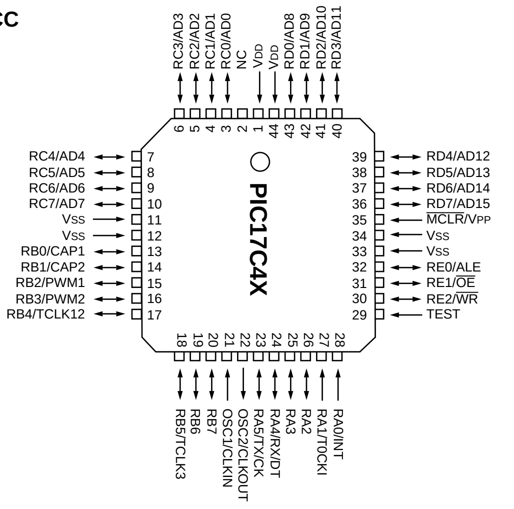
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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"

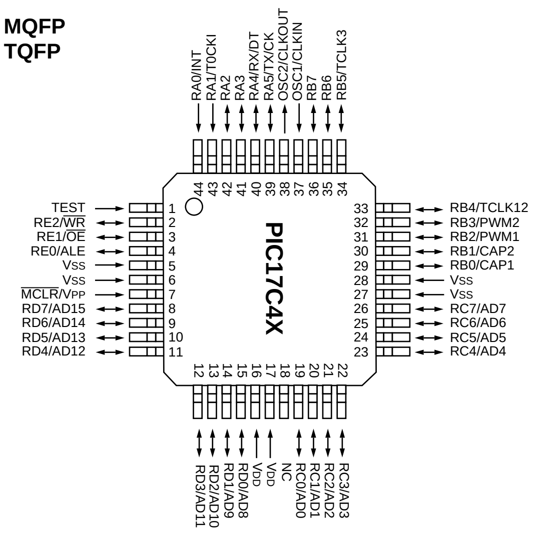
Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	16MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	8KB (4K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	454 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c43-16e-p

Pin Diagrams Cont.'d

PLCC



MQFP
TQFP



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:

6.4.1 INDIRECT ADDRESSING REGISTERS

The PIC17C4X has four registers for indirect addressing. These registers are:

- INDF0 and FSR0
- INDF1 and FSR1

Registers INDF0 and INDF1 are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data. The FSR is an 8-bit register and allows addressing anywhere in the 256-byte data memory address range. For banked memory, the bank of memory accessed is specified by the value in the BSR.

If file INDF0 (or INDF1) itself is read indirectly via an FSR, all '0's are read (Zero bit is set). Similarly, if INDF0 (or INDF1) is written to indirectly, the operation will be equivalent to a NOP, and the status bits are not affected.

6.4.2 INDIRECT ADDRESSING OPERATION

The indirect addressing capability has been enhanced over that of the PIC16CXX family. There are two control bits associated with each FSR register. These two bits configure the FSR register to:

- Auto-decrement the value (address) in the FSR after an indirect access
- Auto-increment the value (address) in the FSR after an indirect access
- No change to the value (address) in the FSR after an indirect access

These control bits are located in the ALUSTA register. The FSR1 register is controlled by the FS3:FS2 bits and FSR0 is controlled by the FS1:FS0 bits.

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the ALUSTA register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

If the FSR register contains a value of 0h, an indirect read will read 0h (Zero bit is set) while an indirect write will be equivalent to a NOP (status bits are not affected).

Indirect addressing allows single cycle data transfers within the entire data space. This is possible with the use of the `MOVPF` and `MOVFP` instructions, where either 'p' or 'f' is specified as INDF0 (or INDF1).

If the source or destination of the indirect address is in banked memory, the location accessed will be determined by the value in the BSR.

A simple program to clear RAM from 20h - FFh is shown in Example 6-1.

EXAMPLE 6-1: INDIRECT ADDRESSING

```
    MOVLW    0x20         ;
    MOVWF    FSR0         ; FSR0 = 20h
    BCF      ALUSTA, FS1  ; Increment FSR
    BSF      ALUSTA, FS0  ; after access
    BCF      ALUSTA, C    ; C = 0
    MOVLW    END_RAM + 1  ;
LP  CLRF     INDF0        ; Addr(FSR) = 0
    CPFSEQ   FSR0         ; FSR0 = END_RAM+1?
    GOTO     LP           ; NO, clear next
    :                     ; YES, All RAM is
    :                     ; cleared
```

6.5 Table Pointer (TBLPTRL and TBLPTRH)

File registers TBLPTRL and TBLPTRH form a 16-bit pointer to address the 64K program memory space. The table pointer is used by instructions `TABLWT` and `TABLRD`.

The `TABLRD` and the `TABLWT` instructions allow transfer of data between program and data space. The table pointer serves as the 16-bit address of the data word within the program memory. For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.

6.6 Table Latch (TBLATH, TBLATL)

The table latch (TBLAT) is a 16-bit register, with TBLATH and TBLATL referring to the high and low bytes of the register. It is not mapped into data or program memory. The table latch is used as a temporary holding latch during data transfer between program and data memory (see descriptions of instructions `TABLRD`, `TABLWT`, `TLRD` and `TLWT`). For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.

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.

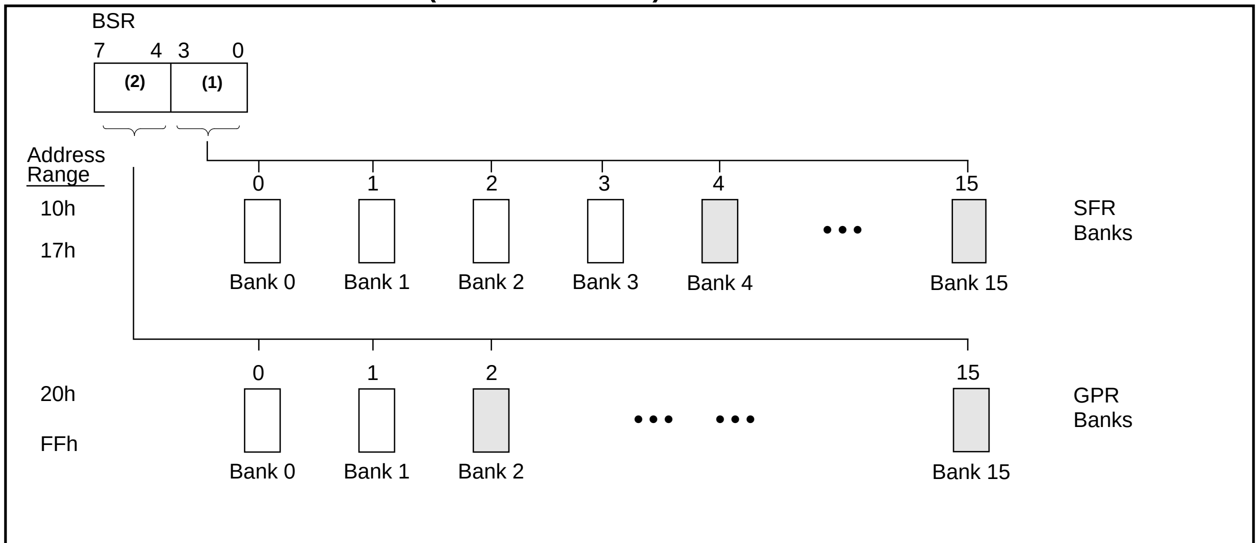
All the Special Function Registers (SFRs) are mapped into the data memory space. In order to accommodate the large number of registers, a banking scheme has been used. A segment of the SFRs, from address 10h to address 17h, is banked. The lower nibble of the bank select register (BSR) selects the currently 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.

7.3 Table Reads

The table read allows the program memory to be read. This allows constant data to be stored in the program memory space, and retrieved into data memory when needed. Example 7-2 reads the 16-bit value at program memory address TBLPTR. After the dummy byte has been read from the TABLATH, the TABLATH is loaded with the 16-bit data from program memory address TBLPTR + 1. The first read loads the data into the latch, and can be considered a dummy read (unknown data loaded into 'f'). INDF0 should be configured for either auto-increment or auto-decrement.

EXAMPLE 7-2: TABLE READ

```
MOVLW   HIGH (TBL_ADDR) ; Load the Table
MOVWF   TBLPTRH         ;   address
MOVLW   LOW (TBL_ADDR)  ;
MOVWF   TBLPTRL         ;
TABLRD  0,0,DUMMY       ; Dummy read,
                        ;  Updates TABLATCH
TLRD    1, INDF0        ; Read HI byte
                        ;   of TABLATCH
TABLRD  0,1,INDF0       ; Read LO byte
                        ;   of TABLATCH and
                        ;   Update TABLATCH
```

FIGURE 7-7: TABLRD TIMING

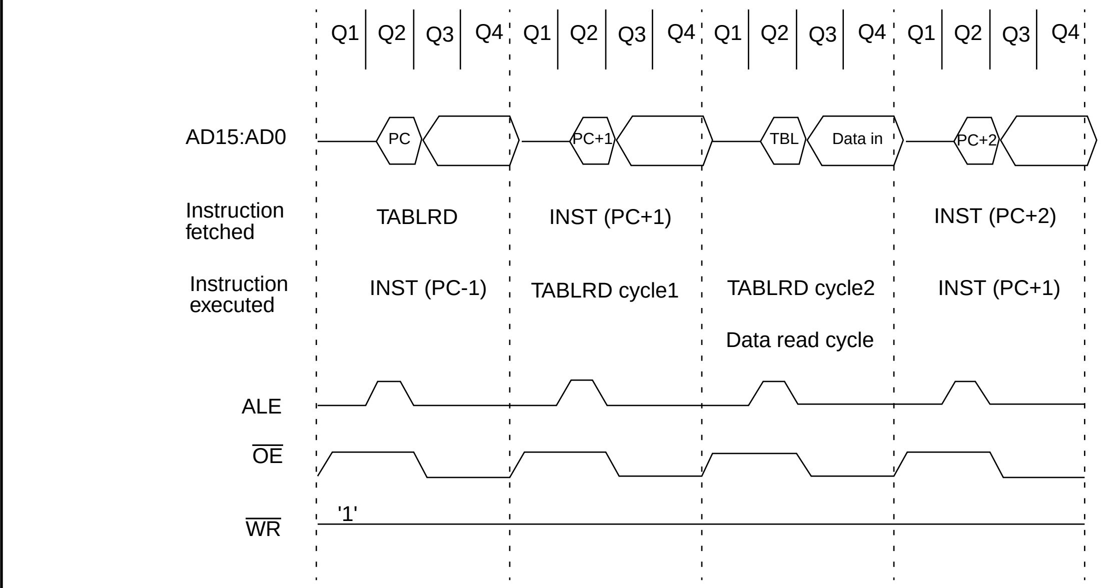
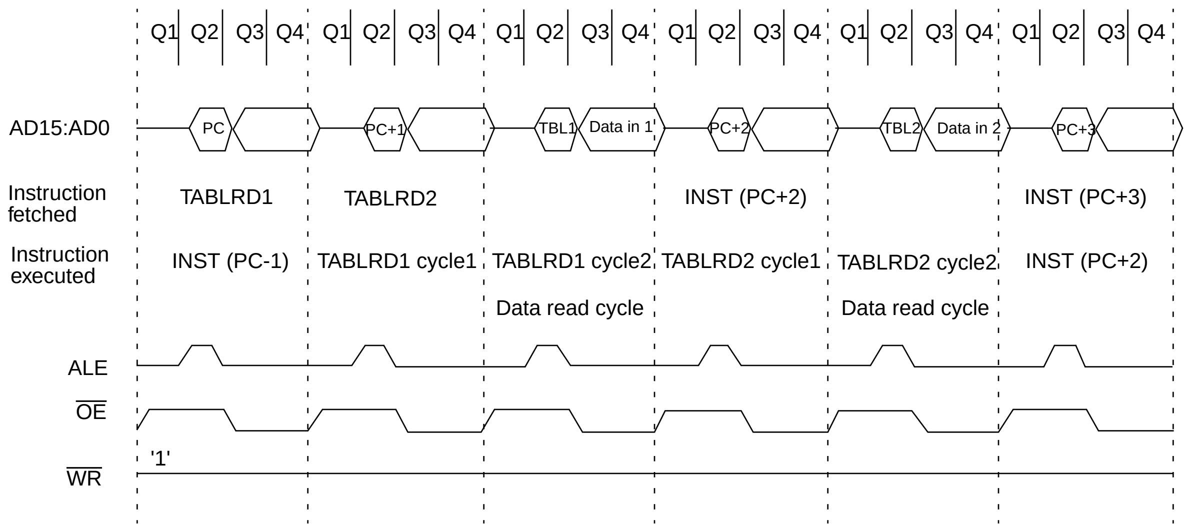


FIGURE 7-8: TABLRD TIMING (CONSECUTIVE TABLRD INSTRUCTIONS)



NOTES:

9.2 PORTB and DDRB Registers

PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is DDRB. A '1' in DDRB configures the corresponding port pin as an input. A '0' in the DDRB register configures the corresponding port pin as an output. Reading PORTB reads the status of the pins, whereas writing to it will write to the port latch.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is done by clearing the $\overline{\text{RBPU}}$ (PORTA<7>) bit. The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are enabled on any reset.

PORTB also has an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e. any RB7:RB0 pin configured as an output is excluded from the interrupt on change comparison). The input pins (of RB7:RB0) are compared with the value in the PORTB data latch. The “mismatch” outputs of RB7:RB0 are OR’ed together to generate the PORTB Interrupt Flag RBIF (PIR<7>).

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

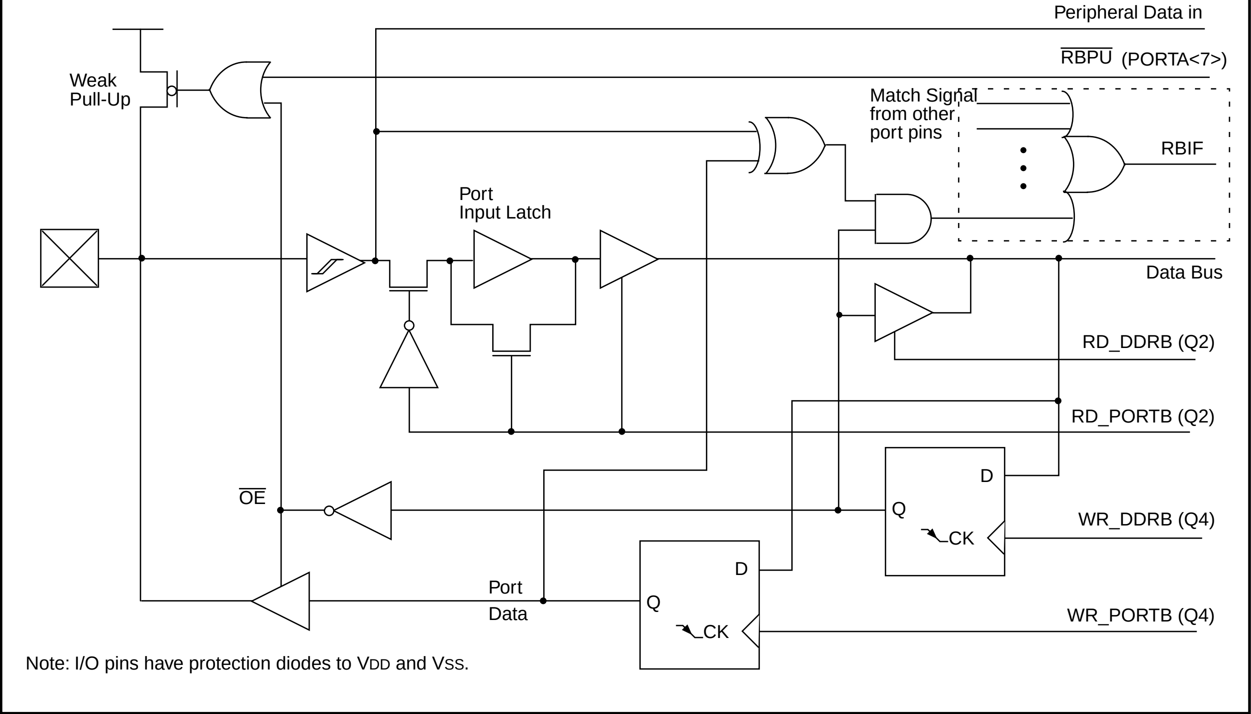
a) Read-Write PORTB (such as; `MOVPF PORTB, PORTB`). This will end mismatch condition.
b) Then, clear the RBIF bit.

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

This interrupt on mismatch feature, together with software configurable pull-ups on this port, allows easy interface to a key pad and make it possible for wake-up on key-depression. For an example, refer to AN552 in the *Embedded Control Handbook*.

The interrupt on change feature is recommended for wake-up on operations where PORTB is only used for the interrupt on change feature and key depression operation.

FIGURE 9-4: BLOCK DIAGRAM OF RB<7:4> AND RB<1:0> PORT PINS



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.

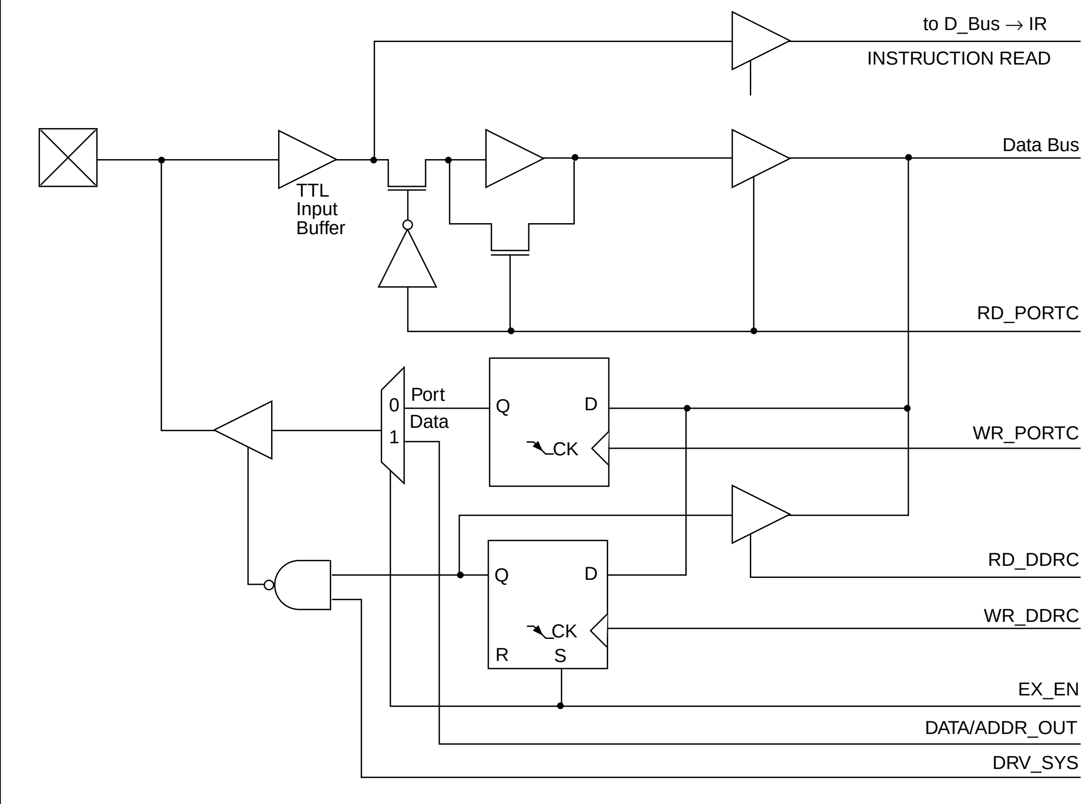
Note: This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

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.

TABLE 15-2: PIC17CXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	**f,d**	ADD WREG to f	1	`0000`	`111d`	`ffff`	`ffff`	OV,C,DC,Z	
ADDWFC	**f,d**	ADD WREG and Carry bit to f	1	`0001`	`000d`	`ffff`	`ffff`	OV,C,DC,Z	
ANDWF	**f,d**	AND WREG with f	1	`0000`	`101d`	`ffff`	`ffff`	Z	
CLRF	**f,s**	Clear f, or Clear f and Clear WREG	1	`0010`	`100s`	`ffff`	`ffff`	None	3
COMF	**f,d**	Complement f	1	`0001`	`001d`	`ffff`	`ffff`	Z	
CPFSEQ	**f**	Compare f with WREG, skip if f = WREG	1 (2)	`0011`	`0001`	`ffff`	`ffff`	None	6,8
CPFSGT	**f**	Compare f with WREG, skip if f > WREG	1 (2)	`0011`	`0010`	`ffff`	`ffff`	None	2,6,8
CPFSLT	**f**	Compare f with WREG, skip if f < WREG	1 (2)	`0011`	`0000`	`ffff`	`ffff`	None	2,6,8
DAW	**f,s**	Decimal Adjust WREG Register	1	`0010`	`111s`	`ffff`	`ffff`	C	3
DECF	**f,d**	Decrement f	1	`0000`	`011d`	`ffff`	`ffff`	OV,C,DC,Z	
DECFSZ	**f,d**	Decrement f, skip if 0	1 (2)	`0001`	`011d`	`ffff`	`ffff`	None	6,8
DCFSNZ	**f,d**	Decrement f, skip if not 0	1 (2)	`0010`	`011d`	`ffff`	`ffff`	None	6,8
INCF	**f,d**	Increment f	1	`0001`	`010d`	`ffff`	`ffff`	OV,C,DC,Z	
INCFSZ	**f,d**	Increment f, skip if 0	1 (2)	`0001`	`111d`	`ffff`	`ffff`	None	6,8
INFSNZ	**f,d**	Increment f, skip if not 0	1 (2)	`0010`	`010d`	`ffff`	`ffff`	None	6,8
IORWF	**f,d**	Inclusive OR WREG with f	1	`0000`	`100d`	`ffff`	`ffff`	Z	
MOVFP	**f,p**	Move f to p	1	`011p`	`pppp`	`ffff`	`ffff`	None	
MOVPF	**p,f**	Move p to f	1	`010p`	`pppp`	`ffff`	`ffff`	Z	
MOVWF	**f**	Move WREG to f	1	`0000`	`0001`	`ffff`	`ffff`	None	
MULWF	**f**	Multiply WREG with f	1	`0011`	`0100`	`ffff`	`ffff`	None	9
NEGW	**f,s**	Negate WREG	1	`0010`	`110s`	`ffff`	`ffff`	OV,C,DC,Z	1,3
NOP	**—**	No Operation	1	`0000`	`0000`	`0000`	`0000`	None	
RLCF	**f,d**	Rotate left f through Carry	1	`0001`	`101d`	`ffff`	`ffff`	C	
RLNCF	**f,d**	Rotate left f (no carry)	1	`0010`	`001d`	`ffff`	`ffff`	None	
RRCF	**f,d**	Rotate right f through Carry	1	`0001`	`100d`	`ffff`	`ffff`	C	
RRNCF	**f,d**	Rotate right f (no carry)	1	`0010`	`000d`	`ffff`	`ffff`	None	
SETF	**f,s**	Set f	1	`0010`	`101s`	`ffff`	`ffff`	None	3
SUBWF	**f,d**	Subtract WREG from f	1	`0000`	`010d`	`ffff`	`ffff`	OV,C,DC,Z	1
SUBWFB	**f,d**	Subtract WREG from f with Borrow	1	`0000`	`001d`	`ffff`	`ffff`	OV,C,DC,Z	1
SWAPF	**f,d**	Swap f	1	`0001`	`110d`	`ffff`	`ffff`	None	
TABLRD	**t,i,f**	Table Read	2 (3)	`1010`	`10ti`	`ffff`	`ffff`	None	7

Legend: Refer to Table 15-1 for opcode field descriptions.

Note 1: 2’s Complement method.

2: Unsigned arithmetic.

3: If s = '1', only the file is affected: If s = '0', both the WREG register and the file are affected; If only the Working register (WREG) is required to be affected, then f = WREG must be specified.

4: During an `LCALL`, the contents of PCLATH are loaded into the MSB of the PC and `kkkk kkkk` is loaded into the LSB of the PC (PCL)

5: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. When writing to external program memory, it is a two-cycle instruction.

6: Two-cycle instruction when condition is true, else single cycle instruction.

7: Two-cycle instruction except for `TABLRD` to PCL (program counter low byte) in which case it takes 3 cycles.

8: A “skip” means that instruction fetched during execution of current instruction is not executed, instead an NOP is executed.

9: These instructions are not available on the PIC17C42.

RETURN — Return from Subroutine

Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC;
Status Affected:	None
Encoding:	`0000` `0000` `0000` `0010`
Description:	Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register PCL*	Execute	NOP
Forced NOP	NOP	Execute	NOP

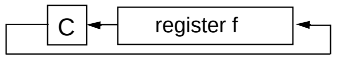
* Remember reading PCL causes PCLATH to be updated. This will be the high address of where the `RETURN` instruction is located.

Example: `RETURN`

After Interrupt
PC = TOS

RLCF — Rotate Left f through Carry

Syntax:	[*label*] RLCF f,d
Operands:	$0 \le f \le 255$ $d \in [0,1]$
Operation:	f<n> → d<n+1>;
	f<7> → C;
	C → d<0>
Status Affected:	C
Encoding:	`0001` `101d` `ffff` `ffff`
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is stored back in register 'f'.



Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write to destination

Example: `RLCF REG,0`

Before Instruction
```
REG  = 1110 0110
C    = 0
```
After Instruction
```
REG  = 1110 0110
WREG = 1100 1100
C    = 1
```

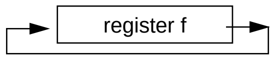
RRNCF — Rotate Right f (no carry)

Syntax:	[*label*] RRNCF f,d
Operands:	0 ≤ f ≤ 255 d ∈ [0,1]
Operation:	f<n> → d<n-1>; f<0> → d<7>
Status Affected:	None

Encoding:

`0010`	`000d`	`ffff`	`ffff`

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is placed back in register 'f'.



Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write to destination

Example 1: `RRNCF   REG, 1`

Before Instruction

WREG	=	?
REG	=	`1101 0111`

After Instruction

WREG	=	0
REG	=	`1110 1011`

Example 2: `RRNCF   REG, 0`

Before Instruction

WREG	=	?
REG	=	`1101 0111`

After Instruction

WREG	=	`1110 1011`
REG	=	`1101 0111`

SETF — Set f

Syntax:	[*label*] SETF f,s
Operands:	0 ≤ f ≤ 255 s ∈ [0,1]
Operation:	FFh → f; FFh → d
Status Affected:	None

Encoding:

`0010`	`101s`	`ffff`	`ffff`

Description: If 's' is 0, both the data memory location 'f' and WREG are set to FFh. If 's' is 1 only the data memory location 'f' is set to FFh.

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'f' and other specified register

Example1: `SETF    REG, 0`

Before Instruction

REG	=	0xDA
WREG	=	0x05

After Instruction

REG	=	0xFF
WREG	=	0xFF

Example2: `SETF    REG, 1`

Before Instruction

REG	=	0xDA
WREG	=	0x05

After Instruction

REG	=	0xFF
WREG	=	0x05

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

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

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

MPASM provides a rich directive language to support programming of the PIC16/17. Directives are helpful in making the development of your assemble source code shorter and more maintainable.

16.11 Software Simulator (MPLAB-SIM)

The MPLAB-SIM Software Simulator allows code development in a PC host environment. It allows the user to simulate the PIC16/17 series microcontrollers on an instruction level. On any given instruction, the user may examine or modify any of the data areas or provide external stimulus to any of the pins. The input/output radix can be set by the user and the execution can be performed in; single step, execute until break, or in a trace mode.

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

16.12 C Compiler (MPLAB-C)

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

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

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

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

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

16.14 MP-DriveWay™ – Application Code Generator

MP-DriveWay is an easy-to-use Windows-based Application Code Generator. With MP-DriveWay you can visually configure all the peripherals in a PIC16/17 device and, with a click of the mouse, generate all the initialization and many functional code modules in C language. The output is fully compatible with Microchip's MPLAB-C C compiler. The code produced is highly modular and allows easy integration of your own code. MP-DriveWay is intelligent enough to maintain your code through subsequent code generation.

16.15 SEEVAL® Evaluation and Programming System

The SEEVAL SEEPROM Designer's Kit supports all Microchip 2-wire and 3-wire Serial EEPROMs. The kit includes everything necessary to read, write, erase or program special features of any Microchip SEEPROM product including Smart Serials™ and secure serials. The Total Endurance™ Disk is included to aid in trade-off analysis and reliability calculations. The total kit can significantly reduce time-to-market and result in an optimized system.

16.16 TrueGauge® Intelligent Battery Management

The TrueGauge development tool supports system development with the MTA11200B TrueGauge Intelligent Battery Management IC. System design verification can be accomplished before hardware prototypes are built. User interface is graphically-oriented and measured data can be saved in a file for exporting to Microsoft Excel.

16.17 KEELOQ® Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchips HCS Secure Data Products. The HCS evaluation kit includes an LCD display to show changing codes, a decoder to decode transmissions, and a programming interface to program test transmitters.

Applicable Devices	42	R42	42A	43	R43	44

TABLE 19-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)

OSC	PIC17LCR42-08 PIC17LC42A-08 PIC17LC43-08 PIC17LCR43-08 PIC17LC44-08	PIC17CR42-16 PIC17C42A-16 PIC17C43-16 PIC17CR43-16 PIC17C44-16	PIC17CR42-25 PIC17C42A-25 PIC17C43-25 PIC17CR43-25 PIC17C44-25	PIC17CR42-33 PIC17C42A-33 PIC17C43-33 PIC17CR43-33 PIC17C44-33	JW Devices (Ceramic Windowed Devices)
RC	V_{DD}: 2.5V to 6.0V I_{DD}: 6 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 6 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 6 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 6 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 6 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.
XT	V_{DD}: 2.5V to 6.0V I_{DD}: 12 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 8 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 24 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 16 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 25 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.
EC	V_{DD}: 2.5V to 6.0V I_{DD}: 12 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 8 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 24 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 16 MHz Max	V_{DD}: 4.5V to 6.0V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 25 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.
LF	V_{DD}: 2.5V to 6.0V I_{DD}: 150 µA max. at 32 kHz I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 2 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 95 µA typ. at 32 kHz I_{PD}: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 95 µA typ. at 32 kHz I_{PD}: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	V_{DD}: 4.5V to 6.0V I_{DD}: 95 µA typ. at 32 kHz I_{PD}: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	V_{DD}: 2.5V to 6.0V I_{DD}: 150 µA max. at 32 kHz I_{PD}: 5 µA max. at 5.5V WDT disabled Freq: 2 MHz max.

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

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-11: MEMORY INTERFACE WRITE TIMING (NOT SUPPORTED IN PIC17LC4X DEVICES)

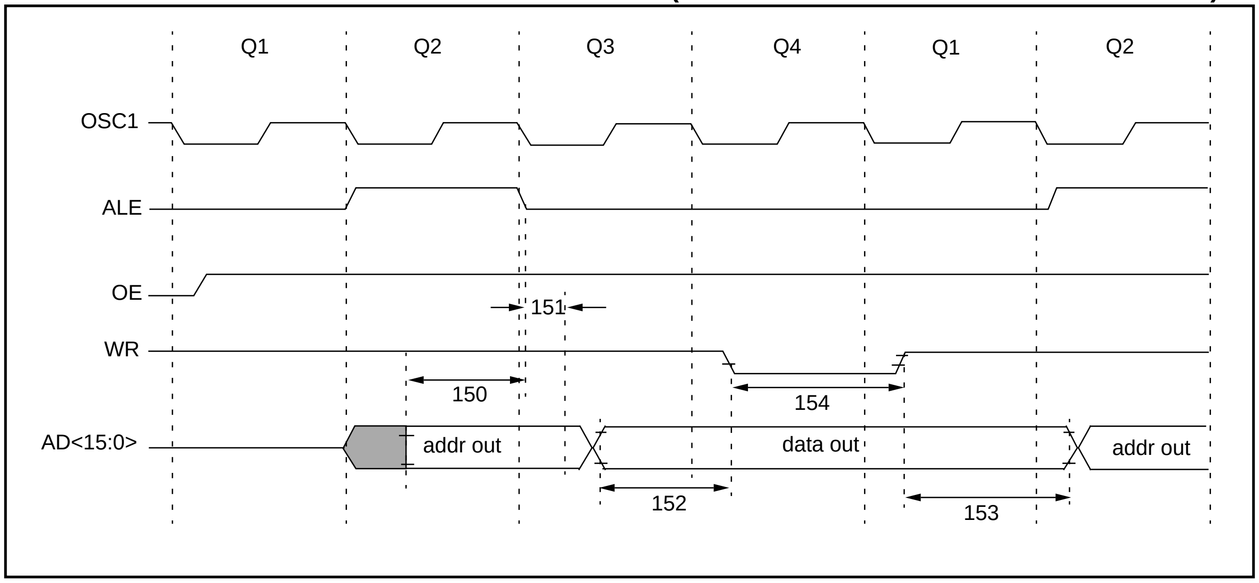


TABLE 19-11: MEMORY INTERFACE WRITE REQUIREMENTS (NOT SUPPORTED IN PIC17LC4X DEVICES)

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 10	—	—	ns	
151	TalL2adI	ALE↓ to address out invalid (address hold time)	0	—	—	ns	
152	TadV2wrL	Data out valid to $\overline{\text{WR}}$↓ (data setup time)	0.25Tcy - 40	—	—	ns	
153	TwrH2adI	$\overline{\text{WR}}$↑ to data out invalid (data hold time)	—	0.25TCY §	—	ns	
154	TwrL	$\overline{\text{WR}}$ pulse width	—	0.25TCY §	—	ns	

* These parameters are characterized but not tested.
† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
§ This specification ensured by design.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-7: TYPICAL IDD vs. FREQUENCY (EXTERNAL CLOCK 25°C)

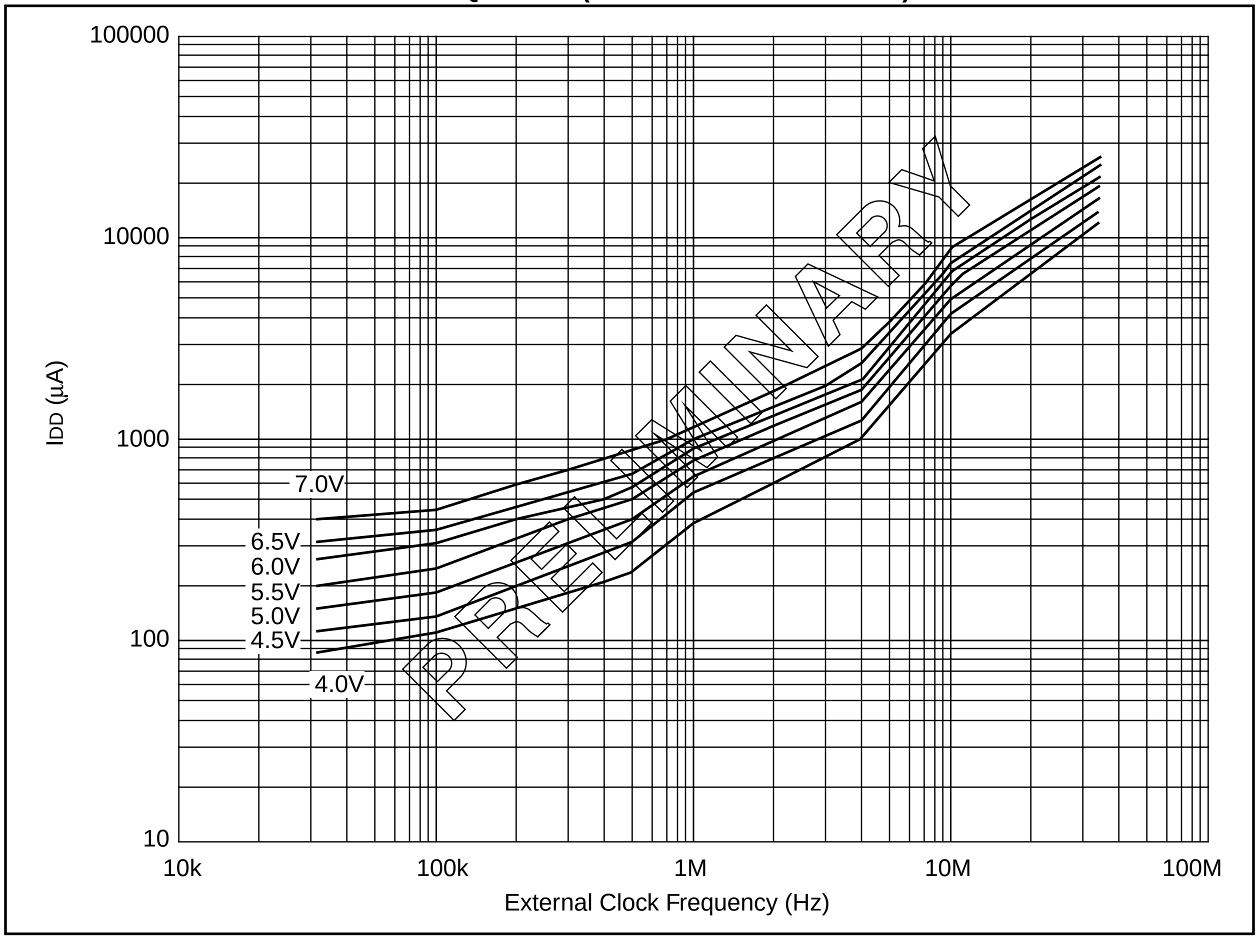
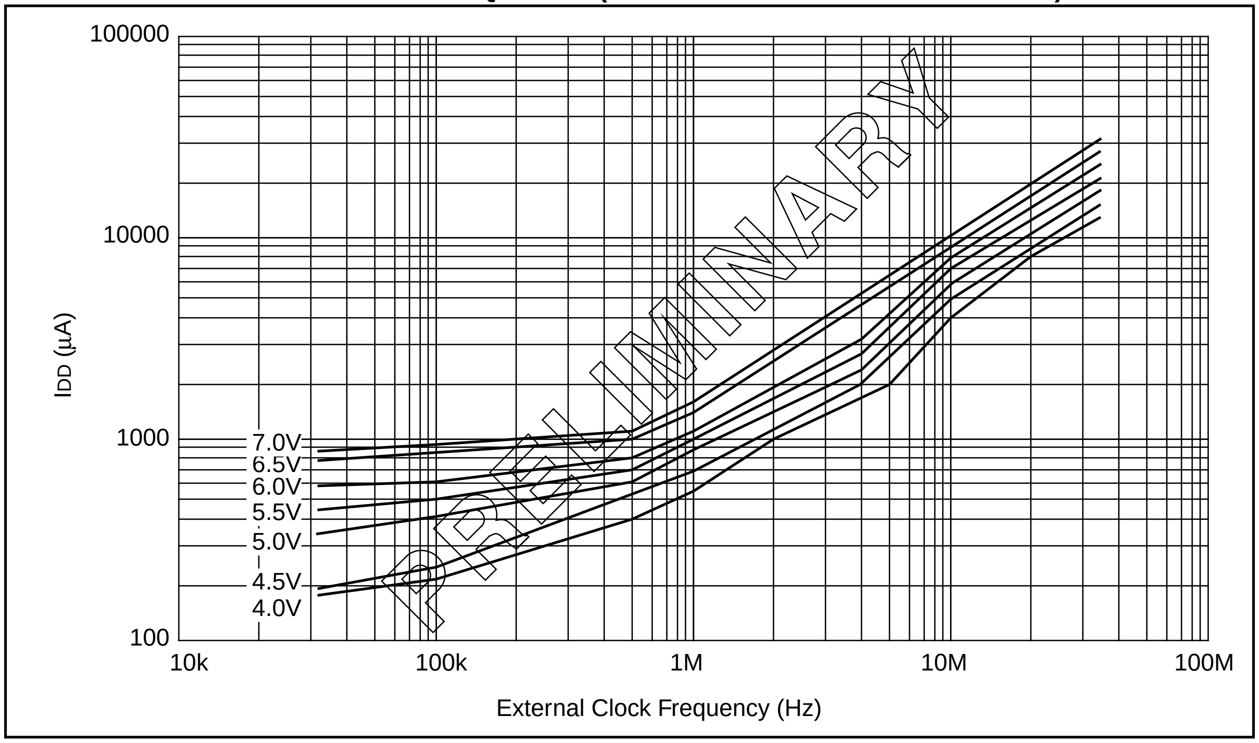


FIGURE 20-8: MAXIMUM IDD vs. FREQUENCY (EXTERNAL CLOCK 125°C TO -40°C)



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-11: TYPICAL IPD vs. VDD WATCHDOG ENABLED 25°C

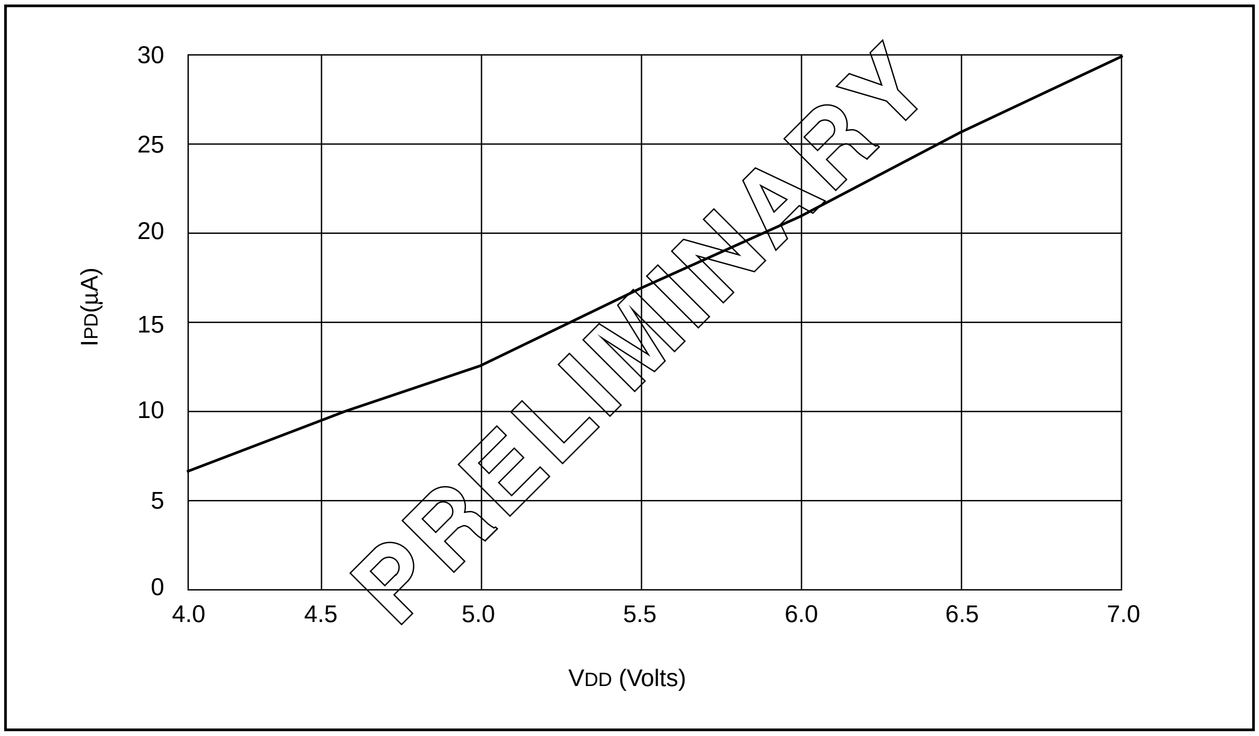
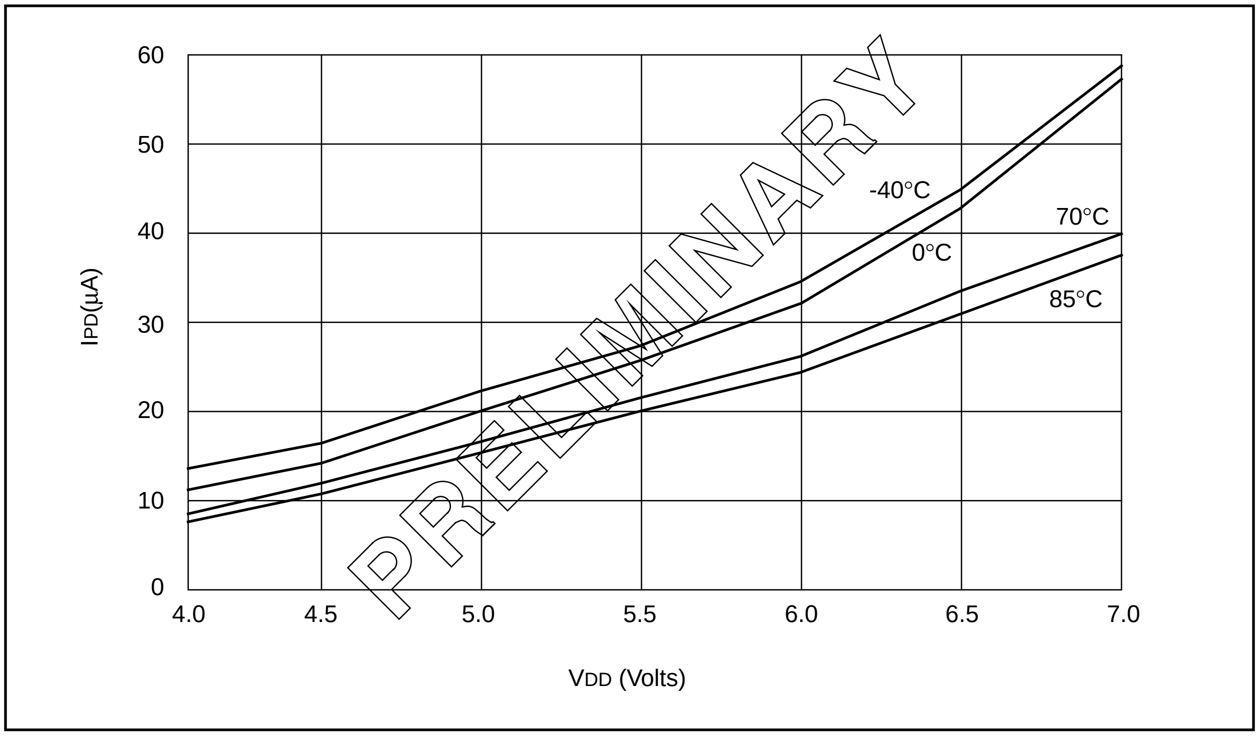


FIGURE 20-12: MAXIMUM IPD vs. VDD WATCHDOG ENABLED



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-17: I_{OL} VS. V_{OL}, V_{DD} = 5V

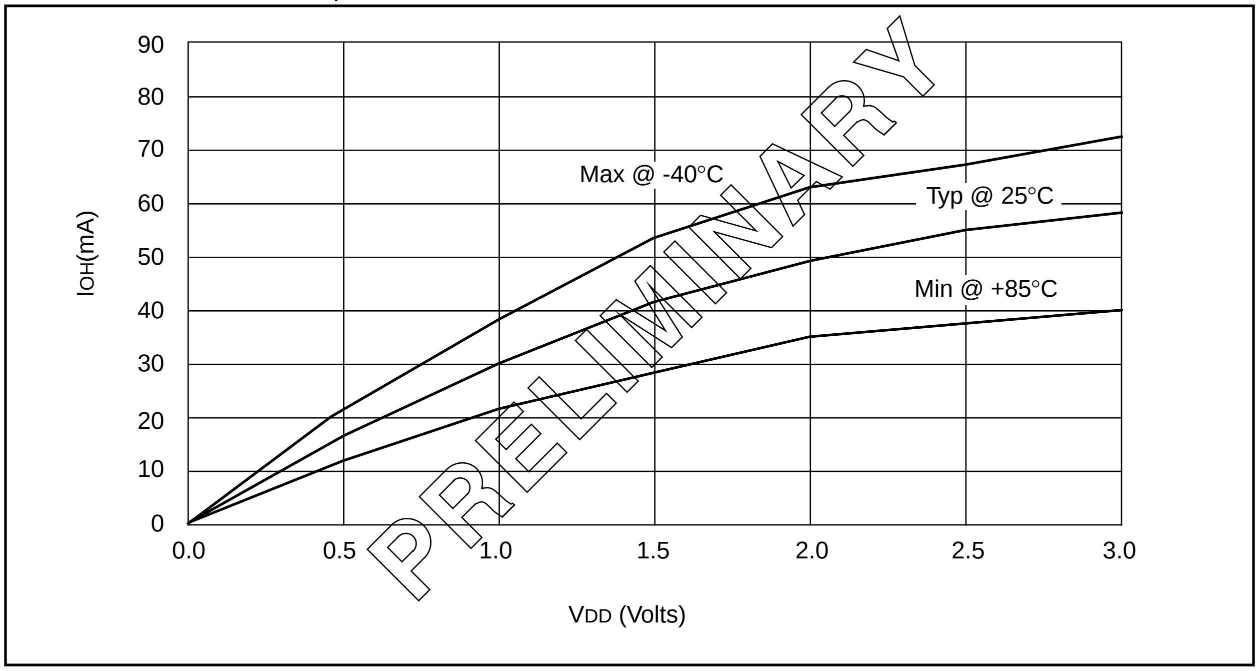
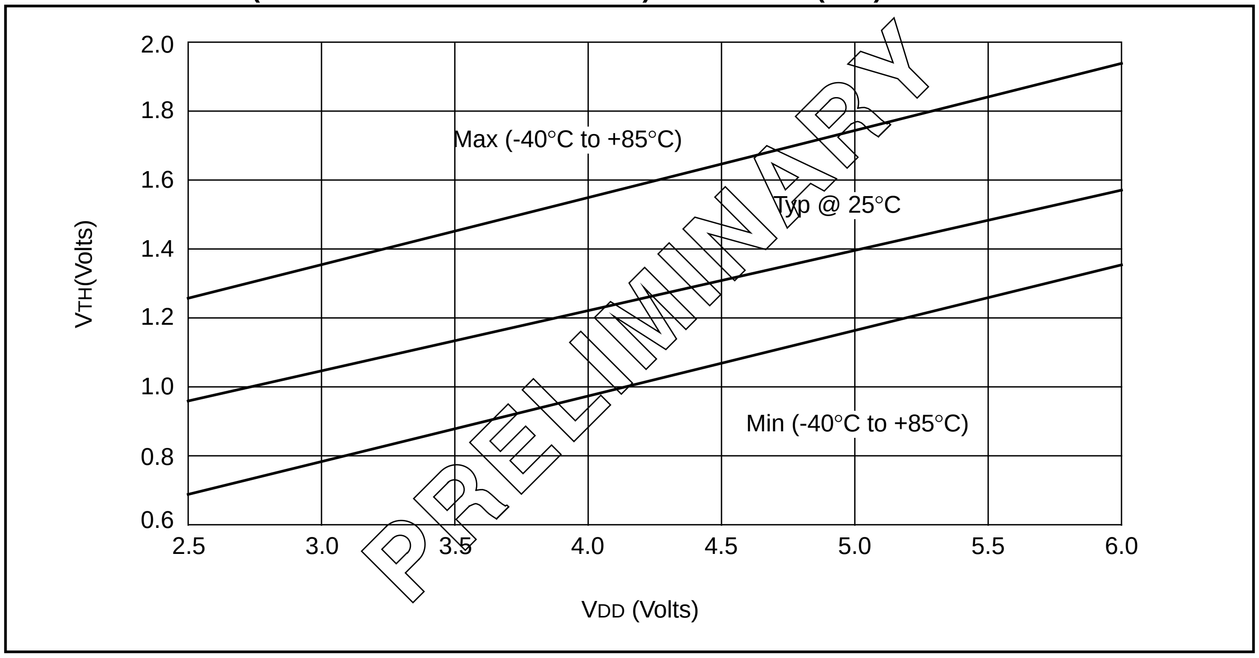
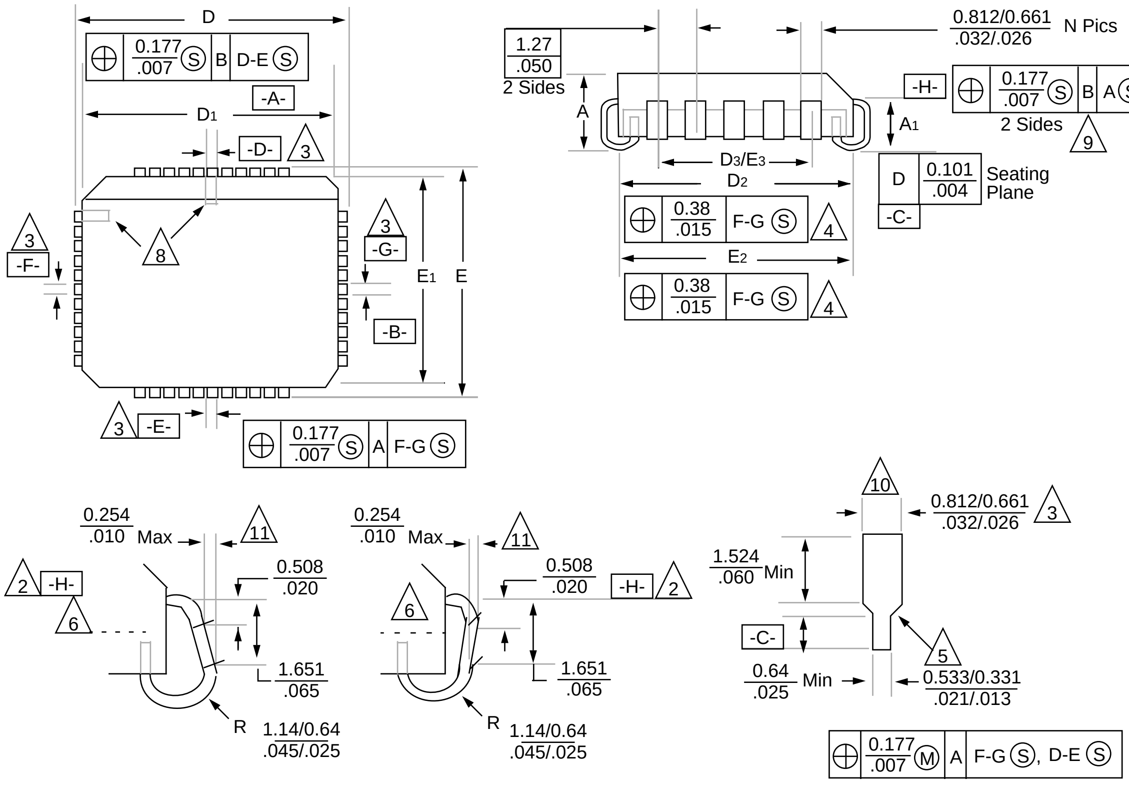


FIGURE 20-18: V_{TH} (INPUT THRESHOLD VOLTAGE) OF I/O PINS (TTL) VS. V_{DD}



21.3 44-Lead Plastic Leaded Chip Carrier (Square)



Package Group: Plastic Leaded Chip Carrier (PLCC)						
	Millimeters			Inches		
Symbol	Min	Max	Notes	Min	Max	Notes
A	4.191	4.572		0.165	0.180	
A1	2.413	2.921		0.095	0.115	
D	17.399	17.653		0.685	0.695	
D1	16.510	16.663		0.650	0.656	
D2	15.494	16.002		0.610	0.630	
D3	12.700	12.700	Reference	0.500	0.500	Reference
E	17.399	17.653		0.685	0.695	
E1	16.510	16.663		0.650	0.656	
E2	15.494	16.002		0.610	0.630	
E3	12.700	12.700	Reference	0.500	0.500	Reference
N	44	44		44	44	
CP	–	0.102		–	0.004	
LT	0.203	0.381		0.008	0.015	

E.5 PIC16C7X Family of Devices

	Clock	Memory		Peripherals					Features				
	Maximum Frequency of Operation (MHz)	EPROM Program Memory (x14 words)	Data Memory (bytes)	Timer Module(s)	Capture/Compare/PWM Module(s)	Serial Port(s) (SPI/I²C, USART)	Parallel Slave Port	A/D Converter (8-bit) Channels	Interrupt Sources	I/O Pins	Voltage Range (Volts)	In-Circuit Serial Programming	Brown-out Reset	Packages
PIC16C710	20	512	36	TMR0	—	—	—	4	4	13	3.0-6.0	Yes	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C71	20	1K	36	TMR0	—	—	—	4	4	13	3.0-6.0	Yes	—	18-pin DIP, SOIC
PIC16C711	20	1K	68	TMR0	—	—	—	4	4	13	3.0-6.0	Yes	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C72	20	2K	128	TMR0, TMR1, TMR2	1	SPI/I²C	—	5	8	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC, SSOP
PIC16C73	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I²C, USART	—	5	11	22	3.0-6.0	Yes	—	28-pin SDIP, SOIC
PIC16C73A(1)	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I²C, USART	—	5	11	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC
PIC16C74	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I²C, USART	Yes	8	12	33	3.0-6.0	Yes	—	40-pin DIP; 44-pin PLCC, MQFP
PIC16C74A(1)	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I²C, USART	Yes	8	12	33	2.5-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP

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

All PIC16C7X Family devices use serial programming with clock pin RB6 and data pin RB7.

Note 1: Please contact your local sales office for availability of these devices.

APPENDIX F: ERRATA FOR PIC17C42 SILICON

The PIC17C42 devices that you have received have the following anomalies. At present there is no intention for future revisions to the present PIC17C42 silicon. If these cause issues for the application, it is recommended that you select the PIC17C42A device.

Note:	New designs should use the PIC17C42A.

1. When the Oscillator Start-Up Timer (OST) is enabled (in LF or XT oscillator modes), any interrupt that wakes the processor may cause a WDT reset. This occurs when the WDT is greater than or equal to 50% time-out period when the `SLEEP` instruction is executed. This will not occur in either the EC or RC oscillator modes.

   Work-arounds

a) Always ensure that the `CLRWDT` instruction is executed before the WDT increments past 50% of the WDT period. This will keep the “false” WDT reset from occurring.

b) When using the WDT as a normal timer (WDT disabled), ensure that the WDT is less than or equal to 50% time-out period when the `SLEEP` instruction is executed. This can be done by monitoring the $\overline{TO}$ bit for changing state from set to clear. Example 1 shows putting the PIC17C42 to sleep.

EXAMPLE F-1: PIC17C42 TO SLEEP

```
      BTFSS   CPUSTA, TO  ; TO = 0?
      CLRWDT              ; YES, WDT = 0
LOOP  BTFSC   CPUSTA, TO  ; WDT rollover?
      GOTO    LOOP        ; NO, Wait
      SLEEP               ; YES, goto Sleep
```

2. When the clock source of Timer1 or Timer2 is selected to external clock, the overflow interrupt flag will be set twice, once when the timer equals the period, and again when the timer value is reset to 0h. If the latency to clear TMRxIF is greater than the time to the next clock pulse, no problems will be noticed. If the latency is less than the time to the next timer clock pulse, the interrupt will be serviced twice.

   Work-arounds

a) Ensure that the timer has rolled over to 0h before clearing the flag bit.

b) Clear the timer in software. Clearing the timer in software causes the period to be one count less than expected.

Design considerations

The device must not be operated outside of the specified voltage range. An external reset circuit must be used to ensure the device is in reset when a brown-out occurs or the V_{DD} rise time is too long. Failure to ensure that the device is in reset when device voltage is out of specification may cause the device to lock-up and ignore the $\overline{MCLR}$ pin.