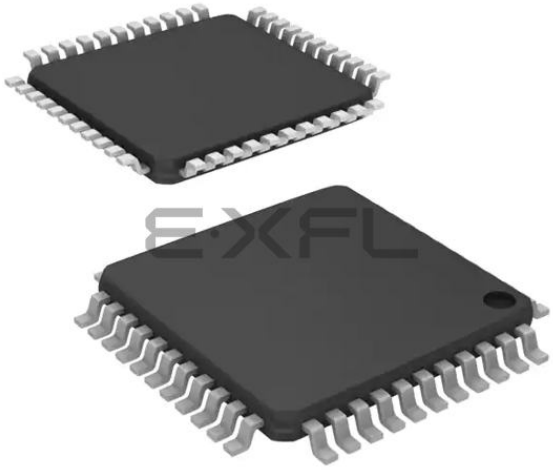


Welcome to E-XFL.COM

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	33MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	16KB (8K 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-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c44-33i-pt

NOTES:

FIGURE 3-2: PIC17CR42/42A/43/R43/44 BLOCK DIAGRAM

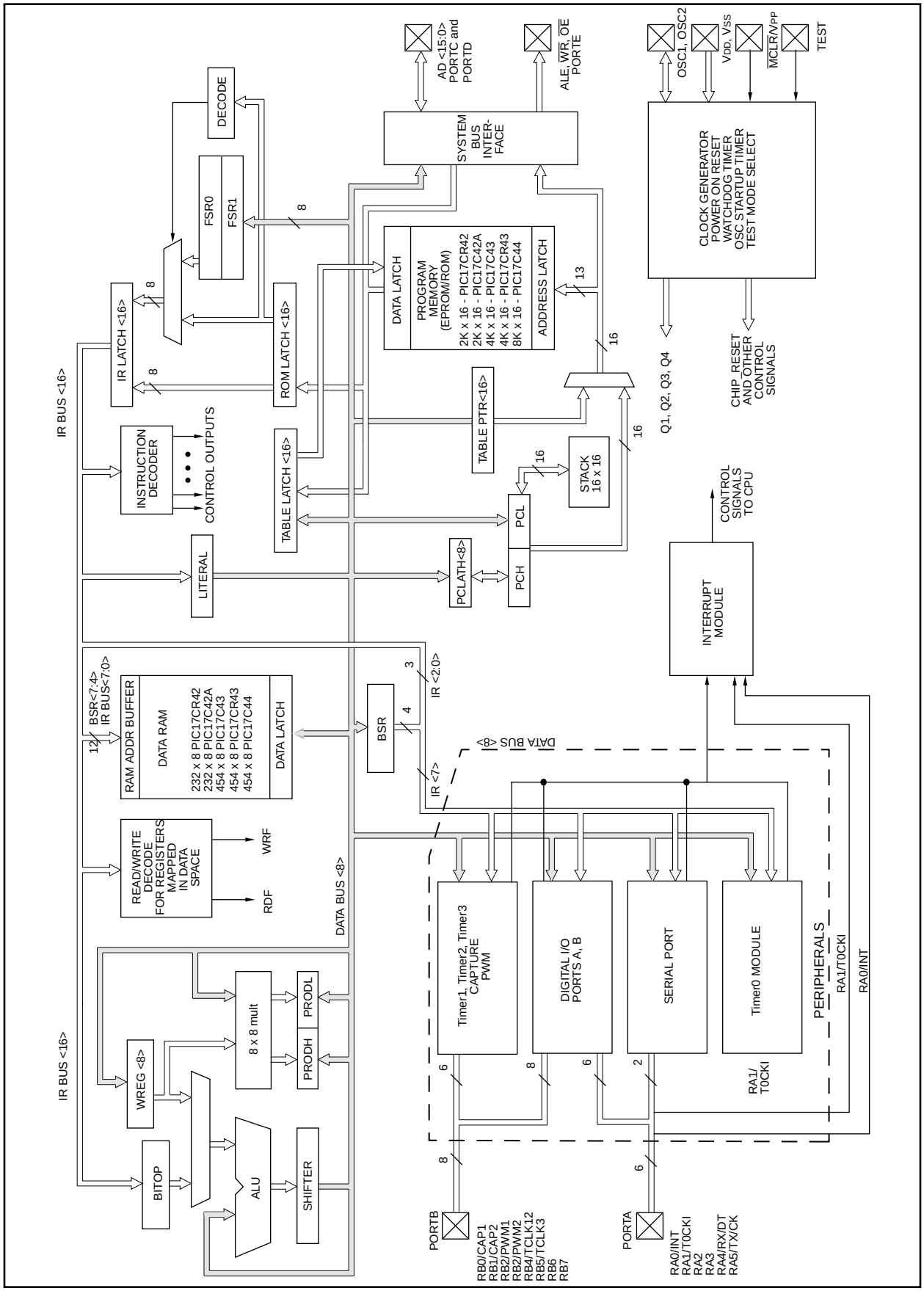


TABLE 6-1: MODE MEMORY ACCESS

Operating Mode	Internal Program Memory	Configuration Bits, Test Memory, Boot ROM
Microprocessor	No Access	No Access
Microcontroller	Access	Access
Extended Microcontroller	Access	No Access
Protected Microcontroller	Access	Access

The PIC17C4X can operate in modes where the program memory is off-chip. They are the microprocessor and extended microcontroller modes. The microprocessor mode is the default for an unprogrammed device.

Regardless of the processor mode, data memory is always on-chip.

FIGURE 6-2: MEMORY MAP IN DIFFERENT MODES

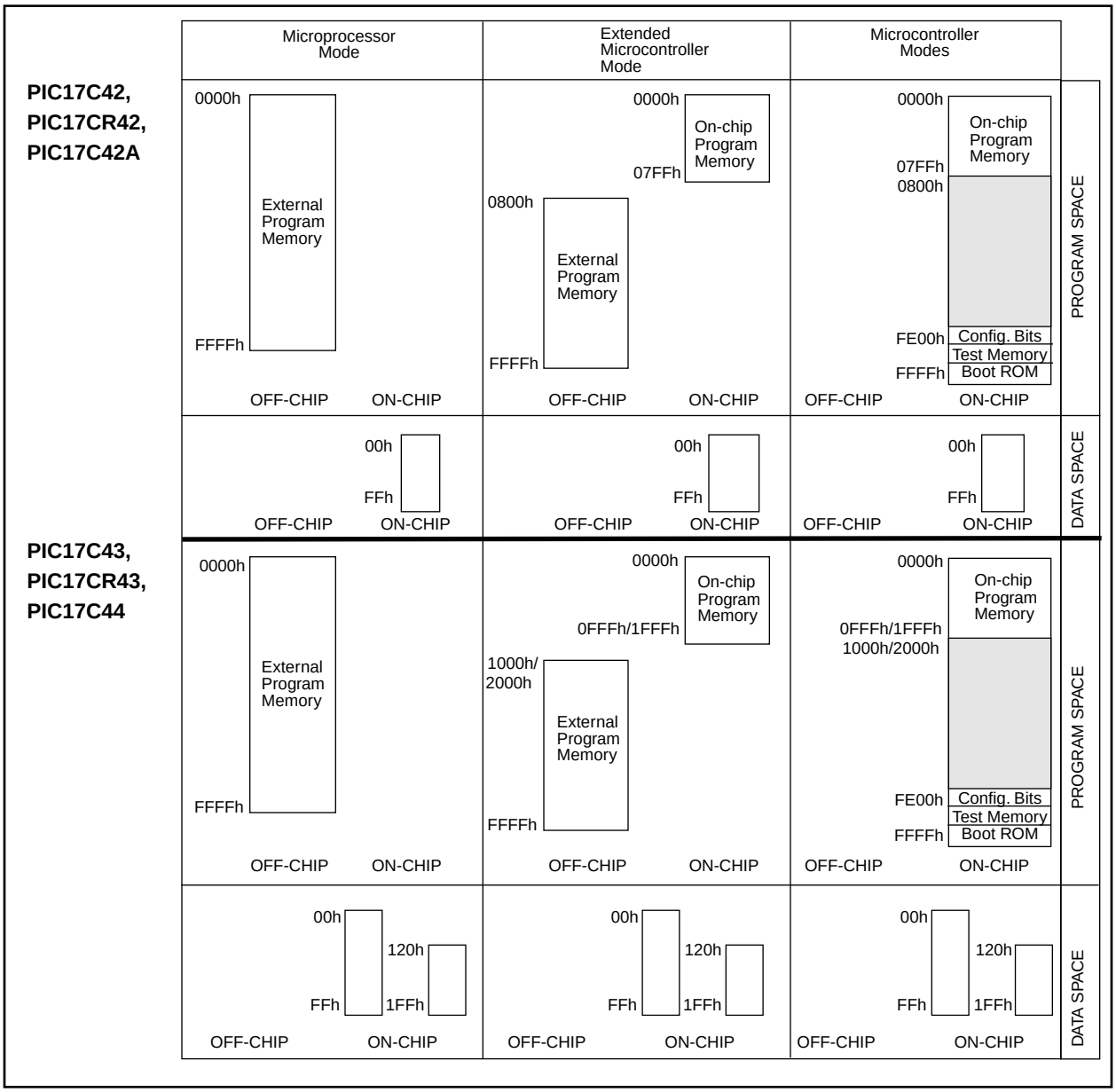
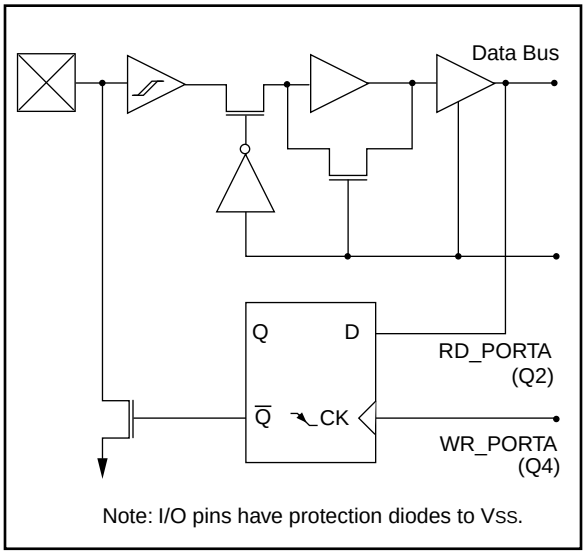
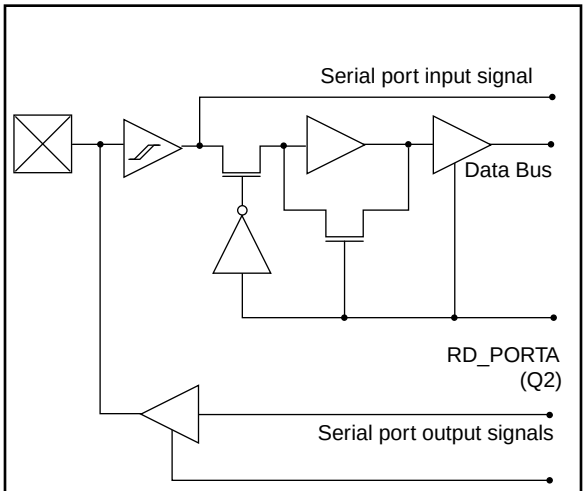


FIGURE 9-2: RA2 AND RA3 BLOCK DIAGRAM



Note: I/O pins have protection diodes to VSS.

FIGURE 9-3: RA4 AND RA5 BLOCK DIAGRAM



$\overline{OE}$ = SPEN,SYNC,TXEN, $\overline{CREN}$, $\overline{SREN}$ for RA4

$\overline{OE}$ = SPEN ($\overline{SYNC}$+SYNC,$\overline{CSRC}$) for RA5

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

TABLE 9-1: PORTA FUNCTIONS

Name	Bit0	Buffer Type	Function
RA0/INT	bit0	ST	Input or external interrupt input.
RA1/T0CKI	bit1	ST	Input or clock input to the TMR0 timer/counter, and/or an external interrupt input.
RA2	bit2	ST	Input/Output. Output is open drain type.
RA3	bit3	ST	Input/Output. Output is open drain type.
RA4/RX/DT	bit4	ST	Input or USART Asynchronous Receive or USART Synchronous Data.
RA5/TX/CK	bit5	ST	Input or USART Asynchronous Transmit or USART Synchronous Clock.
$\overline{RBPU}$	bit7	—	Control bit for PORTB weak pull-ups.

Legend: ST = Schmitt Trigger input.

TABLE 9-2: REGISTERS/BITS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note1)
10h, Bank 0	PORTA	$\overline{RBPU}$	—	RA5	RA4	RA3	RA2	RA1/T0CKI	RA0/INT	0-xx xxxx	0-uu uuuu
05h, Unbanked	T0STA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
13h, Bank 0	RCSTA	SPEN	RC9	SREN	CREN	—	FERR	OERR	RC9D	0000 -00x	0000 -00u
15h, Bank 0	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u

Legend: x = unknown, u = unchanged, - = unimplemented reads as '0'. Shaded cells are not used by PORTA.

Note 1: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

9.4 PORTD and DDRD Registers

PORTD is an 8-bit bi-directional port. The corresponding data direction register is DDRD. A '1' in DDRD configures the corresponding port pin as an input. A '0' in the DDRC register configures the corresponding port pin as an output. Reading PORTD reads the status of the pins, whereas writing to it will write to the port latch. PORTD is multiplexed with the system bus. When operating as the system bus, PORTD is the high order byte of the address/data bus (AD15:AD8). The timing for the system bus is shown in the Electrical Characteristics section.

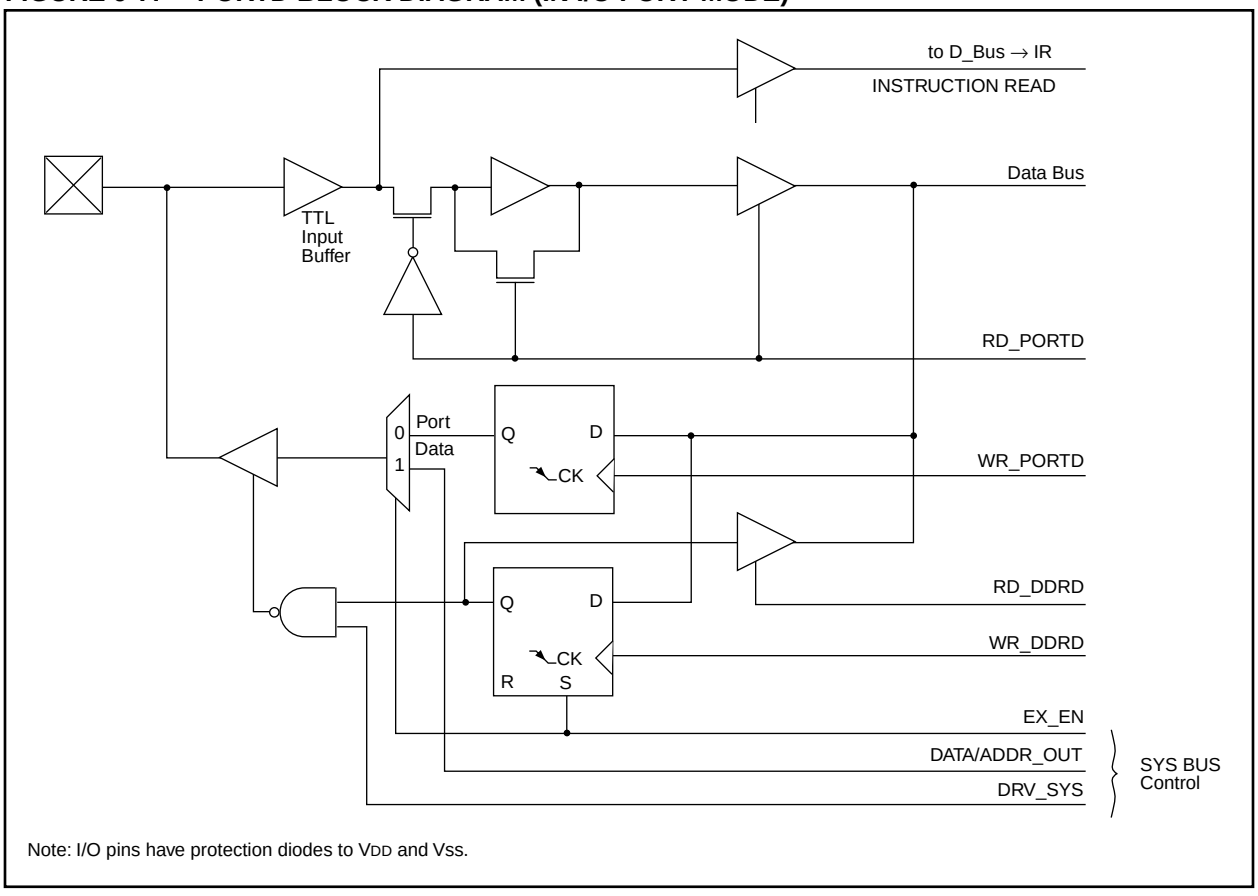
Note:	This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 9-3 shows the instruction sequence to initialize PORTD. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized.

EXAMPLE 9-3: INITIALIZING PORTD

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

FIGURE 9-7: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)



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

TABLE 9-7: PORTD FUNCTIONS

Name	Bit	Buffer Type	Function
RD0/AD8	bit0	TTL	Input/Output or system bus address/data pin.
RD1/AD9	bit1	TTL	Input/Output or system bus address/data pin.
RD2/AD10	bit2	TTL	Input/Output or system bus address/data pin.
RD3/AD11	bit3	TTL	Input/Output or system bus address/data pin.
RD4/AD12	bit4	TTL	Input/Output or system bus address/data pin.
RD5/AD13	bit5	TTL	Input/Output or system bus address/data pin.
RD6/AD14	bit6	TTL	Input/Output or system bus address/data pin.
RD7/AD15	bit7	TTL	Input/Output or system bus address/data pin.

Legend: TTL = TTL input.

TABLE 9-8: REGISTERS/BITS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note1)
13h, Bank 1	PORTD	RD7/ AD15	RD6/ AD14	RD5/ AD13	RD4/ AD12	RD3/ AD11	RD2/ AD10	RD1/ AD9	RD0/ AD8	xxxx xxxx	uuuu uuuu
12h, Bank 1	DDRD	Data direction register for PORTD								1111 1111	1111 1111

Legend: x = unknown, u = unchanged.

Note 1: Other (non power-up) resets include: external reset through $\overline{\text{MCLR}}$ and the Watchdog Timer Reset.

10.0 OVERVIEW OF TIMER RESOURCES

The PIC17C4X has four timer modules. Each module can generate an interrupt to indicate that an event has occurred. These timers are called:

- Timer0 - 16-bit timer with programmable 8-bit prescaler
- Timer1 - 8-bit timer
- Timer2 - 8-bit timer
- Timer3 - 16-bit timer

For enhanced time-base functionality, two input Captures and two Pulse Width Modulation (PWM) outputs are possible. The PWMs use the TMR1 and TMR2 resources and the input Captures use the TMR3 resource.

10.1 Timer0 Overview

The Timer0 module is a simple 16-bit overflow counter. The clock source can be either the internal system clock (Fosc/4) or an external clock.

The Timer0 module also has a programmable prescaler option. The PS3:PS0 bits (T0STA<4:1>) determine the prescaler value. TMR0 can increment at the following rates: 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:128, 1:256.

When TImer0's clock source is an external clock, the Timer0 module can be selected to increment on either the rising or falling edge.

Synchronization of the external clock occurs after the prescaler. When the prescaler is used, the external clock frequency may be higher then the device's frequency. The maximum frequency is 50 MHz, given the high and low time requirements of the clock.

10.2 Timer1 Overview

The TImer0 module is an 8-bit timer/counter with an 8-bit period register (PR1). When the TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set, and an interrupt will be generated when enabled. In counter mode, the clock comes from the RB4/TCLK12 pin, which can also be selected to be the clock for the Timer2 module.

TMR1 can be concatenated to TMR2 to form a 16-bit timer. The TMR1 register is the LSB and TMR2 is the MSB. When in the 16-bit timer mode, there is a corresponding 16-bit period register (PR2:PR1). When the TMR2:TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set, and an interrupt will be generated when enabled.

10.3 Timer2 Overview

The TMR2 module is an 8-bit timer/counter with an 8-bit period register (PR2). When the TMR2 value rolls over from the period match value to 0h, the TMR2IF flag is set, and an interrupt will be generated when enabled. In counter mode, the clock comes from the RB4/TCLK12 pin, which can also be selected to be the clock for the TMR1 module.

TMR1 can be concatenated to TMR2 to form a 16-bit timer. The TMR2 register is the MSB and TMR1 is the LSB. When in the 16-bit timer mode, there is a corresponding 16-bit period register (PR2:PR1). When the TMR2:TMR1 value rolls over from the period match value to 0h, the TMR1IF flag is set, and an interrupt will be generated when enabled.

10.4 Timer3 Overview

The TImer3 module is a 16-bit timer/counter with a 16-bit period register. When the TMR3H:TMR3L value rolls over to 0h, the TMR3IF bit is set and an interrupt will be generated when enabled. In counter mode, the clock comes from the RB5/TCLK3 pin.

When operating in the dual capture mode, the period registers become the second 16-bit capture register.

10.5 Role of the Timer/Counters

The timer modules are general purpose, but have dedicated resources associated with them. TImer1 and Timer2 are the time-bases for the two Pulse Width Modulation (PWM) outputs, while Timer3 is the time-base for the two input captures.

ADDLW **ADD Literal to WREG**

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) + k → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

1011	0001	kkkk	kkkk

Description: The contents of WREG are added to the 8-bit literal 'k' and the result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDLW 0x15`

Before Instruction
WREG = 0x10

After Instruction
WREG = 0x25

ADDWF **ADD WREG to f**

Syntax: [*label*] ADDWF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (WREG) + (f) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	111d	ffff	ffff

Description: Add WREG to register 'f'. If 'd' is 0 the result is stored 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: `ADDWF REG, 0`

Before Instruction
WREG = 0x17
REG = 0xC2

After Instruction
WREG = 0xD9
REG = 0xC2

INFSNZ **Increment f, skip if not 0**

Syntax: [*label*] INFSNZ f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (f) + 1 → (dest), skip if not 0

Status Affected: None

Encoding:

0010	010d	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is placed back in register 'f'.
If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:
```
HERE    INFSNZ  REG, 1
ZERO
NZERO
```

Before Instruction
REG = REG

After Instruction
REG = REG + 1
If REG = 1;
PC = Address (ZERO)
If REG = 0;
PC = Address (NZERO)

IORLW **Inclusive OR Literal with WREG**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .OR. (k) → (WREG)

Status Affected: Z

Encoding:

1011	0011	kkkk	kkkk

Description: The contents of WREG are OR'ed with the eight bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
IORLW   0x35
```

Before Instruction
WREG = 0x9A

After Instruction
WREG = 0xBF

MOVFP **Move f to p**

Syntax: [*label*] MOVFP f,p

Operands: $0 \le f \le 255$
$0 \le p \le 31$

Operation: (f) → (p)

Status Affected: None

Encoding:

011p	pppp	ffff	ffff

Description: Move data from data memory location 'f' to data memory location 'p'. Location 'f' can be anywhere in the 256 word data space (00h to FFh) while 'p' can be 00h to 1Fh.

Either 'p' or 'f' can be WREG (a useful special situation).

MOVFP is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). Both 'f' and 'p' can be indirectly addressed.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'p'

Example: MOVFP REG1, REG2

Before Instruction
REG1 = 0x33,
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

MOVLB **Move Literal to low nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \le k \le 15$

Operation: k → (BSR<3:0>)

Status Affected: None

Encoding:

1011	1000	uuuu	kkkk

Description: The four bit literal 'k' is loaded in the Bank Select Register (BSR). Only the low 4-bits of the Bank Select Register are affected. The upper 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 'u:k'	Execute	Write literal 'k' to BSR<3:0>

Example: MOVLB 0x5

Before Instruction
BSR register = 0x22

After Instruction
BSR register = 0x25

Note: For the PIC17C42, only the low four bits of the BSR register are physically implemented. The upper nibble is read as '0'.

RLNCF **Rotate Left f (no carry)**

Syntax: [*label*] RLNCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

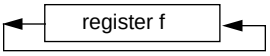
Operation: f<n> → d<n+1>;
f<7> → d<0>

Status Affected: None

Encoding:

0010	001d	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. 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: RLNCF REG, 1

Before Instruction
C = 0
REG = 1110 1011

After Instruction
C =
REG = 1101 0111

RRCF **Rotate Right f through Carry**

Syntax: [*label*] RRCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

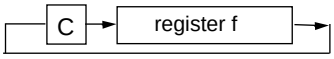
Operation: f<n> → d<n-1>;
f<0> → C;
C → d<7>

Status Affected: C

Encoding:

0001	100d	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in 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: RRCF REG1,0

Before Instruction
REG1 = 1110 0110
C = 0

After Instruction
REG1 = 1110 0110
WREG = 0111 0011
C = 0

Applicable Devices	42	R42	42A	43	R43	44

17.3 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS

T			
F	Frequency	T	Time

Lowercase symbols (pp) and their meanings:

pp			
ad	Address/Data	ost	Oscillator Start-up Timer
al	ALE	pwrt	Power-up Timer
cc	Capture1 and Capture2	rb	PORTB
ck	CLKOUT or clock	rd	$\overline{RD}$
dt	Data in	rw	$\overline{RD}$ or $\overline{WR}$
in	INT pin	t0	T0CKI
io	I/O port	t123	TCLK12 and TCLK3
mc	$\overline{MCLR}$	wdt	Watchdog Timer
oe	$\overline{OE}$	wr	$\overline{WR}$
os	OSC1		

Uppercase symbols and their meanings:

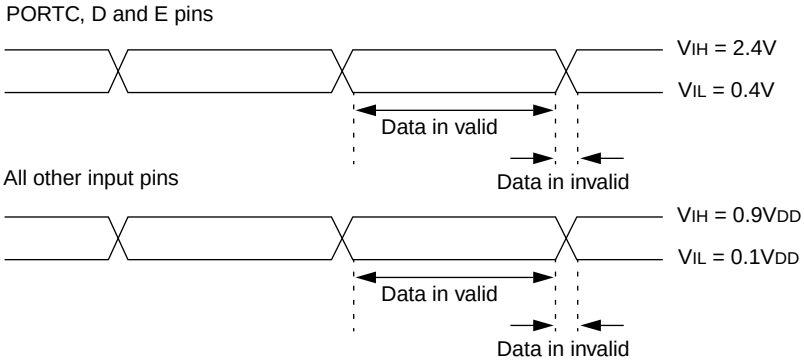
S			
D	Driven	L	Low
E	Edge	P	Period
F	Fall	R	Rise
H	High	V	Valid
I	Invalid (Hi-impedance)	Z	Hi-impedance

Applicable Devices	42	R42	42A	43	R43	44

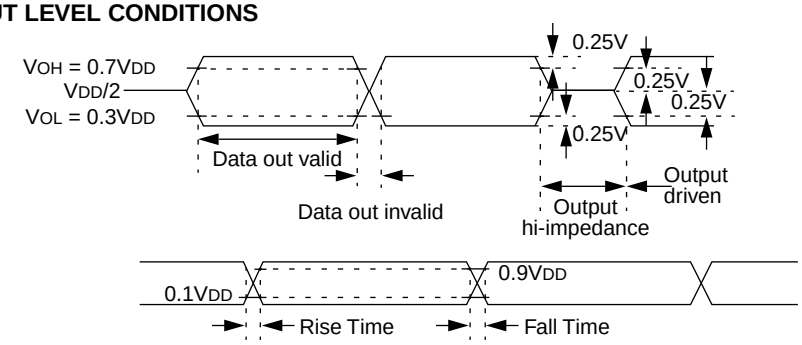
FIGURE 17-1: PARAMETER MEASUREMENT INFORMATION

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

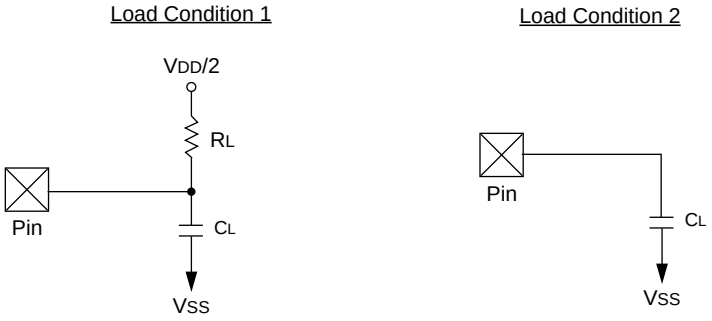
INPUT LEVEL CONDITIONS



OUTPUT LEVEL CONDITIONS



LOAD CONDITIONS



R_L = 464
$C_L \leq$ 50 pF

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 18-2: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD

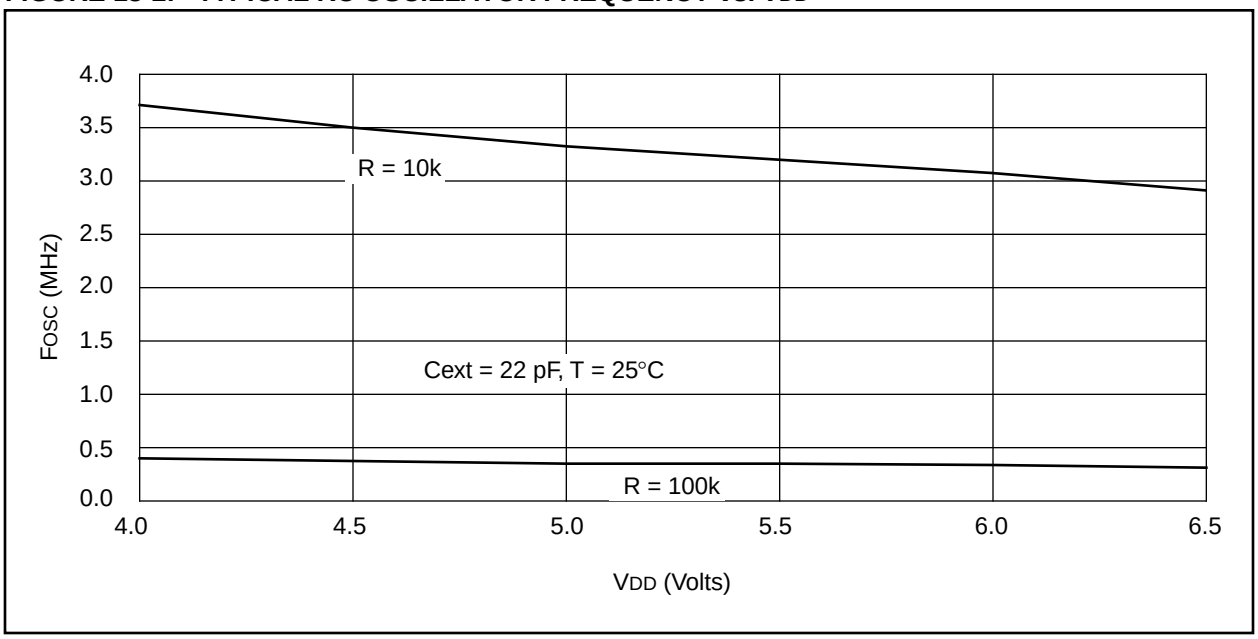
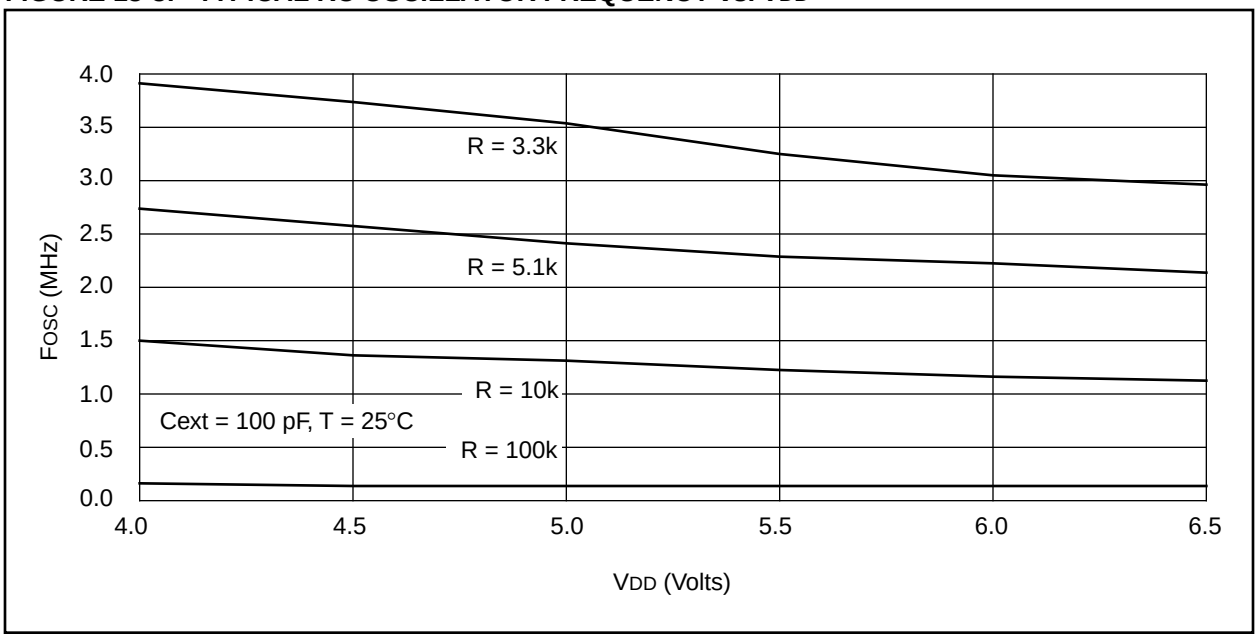


FIGURE 18-3: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD



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	VDD: 2.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.	VDD: 4.5V to 6.0V IDD: 6 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 4 MHz max.
XT	VDD: 2.5V to 6.0V IDD: 12 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 8 MHz max.	VDD: 4.5V to 6.0V IDD: 24 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 16 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 25 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.
EC	VDD: 2.5V to 6.0V IDD: 12 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 8 MHz max.	VDD: 4.5V to 6.0V IDD: 24 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 16 MHz Max	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 25 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.	VDD: 4.5V to 6.0V IDD: 38 mA max. IPD: 5 µA max. at 5.5V WDT disabled Freq: 33 MHz max.
LF	VDD: 2.5V to 6.0V IDD: 150 µA max. at 32 kHz IPD: 5 µA max. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 4.5V to 6.0V IDD: 95 µA typ. at 32 kHz IPD: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 4.5V to 6.0V IDD: 95 µA typ. at 32 kHz IPD: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 4.5V to 6.0V IDD: 95 µA typ. at 32 kHz IPD: < 1 µA typ. at 5.5V WDT disabled Freq: 2 MHz max.	VDD: 2.5V to 6.0V IDD: 150 µA max. at 32 kHz IPD: 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

19.4 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase symbols (pp) and their meanings:

pp			
ad	Address/Data	ost	Oscillator Start-Up Timer
al	ALE	pwrt	Power-Up Timer
cc	Capture1 and Capture2	rb	PORTB
ck	CLKOUT or clock	rd	$\overline{RD}$
dt	Data in	rw	$\overline{RD}$ or $\overline{WR}$
in	INT pin	t0	T0CKI
io	I/O port	t123	TCLK12 and TCLK3
mc	$\overline{MCLR}$	wdt	Watchdog Timer
oe	$\overline{OE}$	wr	$\overline{WR}$
os	OSC1		

Uppercase symbols and their meanings:

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

Applicable Devices	42	R42	42A	43	R43	44

20.0 PIC17CR42/42A/43/R43/44 DC AND AC CHARACTERISTICS

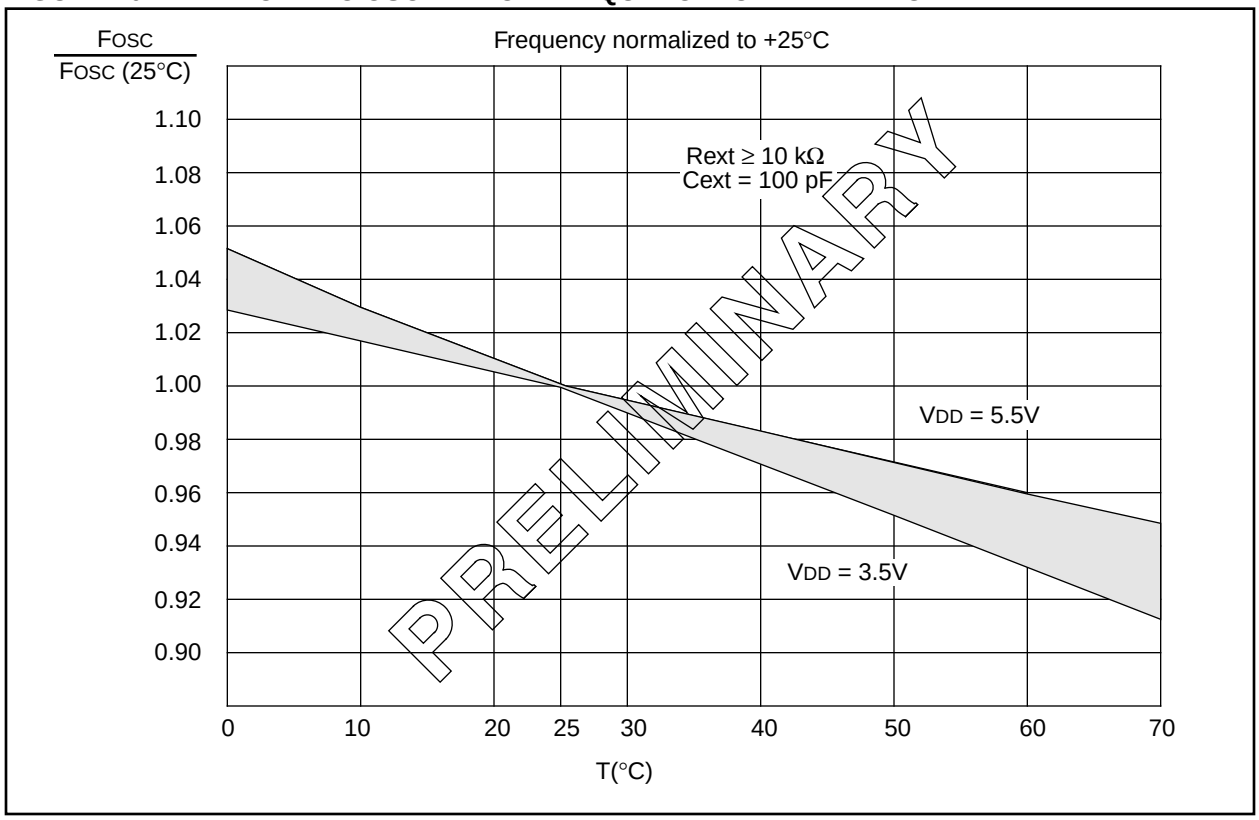
The graphs and tables provided in this section are for design guidance and are not tested nor guaranteed. In some graphs or tables the data presented is outside specified operating range (e.g. outside specified VDD range). This is for information only and devices are ensured to operate properly only within the specified range.

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

TABLE 20-1: PIN CAPACITANCE PER PACKAGE TYPE

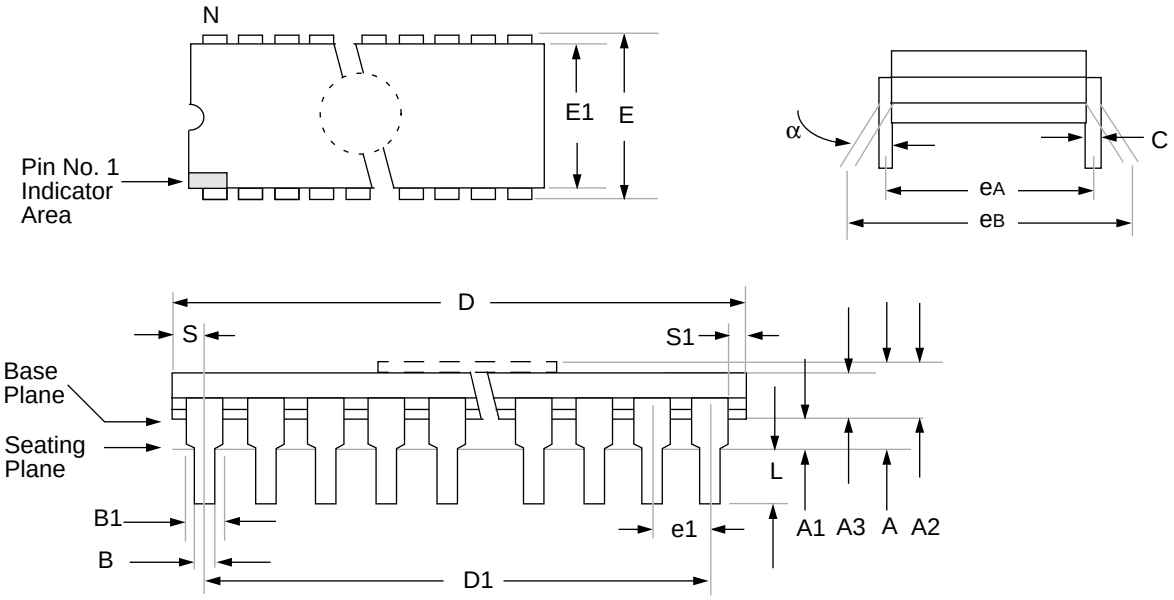
Pin Name	Typical Capacitance (pF)			
	40-pin DIP	44-pin PLCC	44-pin MQFP	44-pin TQFP
All pins, except $\overline{MCLR}$, VDD, and VSS	10	10	10	10
$\overline{MCLR}$ pin	20	20	20	20

FIGURE 20-1: TYPICAL RC OSCILLATOR FREQUENCY vs. TEMPERATURE



21.0 PACKAGING INFORMATION

21.1 40-Lead Ceramic CERDIP Dual In-line, and CERDIP Dual In-line with Window (600 mil)



Package Group: Ceramic CERDIP Dual In-Line (CDP)						
	Millimeters			Inches		
Symbol	Min	Max	Notes	Min	Max	Notes
α	0°	10°		0°	10°	
A	4.318	5.715		0.170	0.225	
A1	0.381	1.778		0.015	0.070	
A2	3.810	4.699		0.150	0.185	
A3	3.810	4.445		0.150	0.175	
B	0.355	0.585		0.014	0.023	
B1	1.270	1.651	**Typical**	0.050	0.065	**Typical**
C	0.203	0.381	**Typical**	0.008	0.015	**Typical**
D	51.435	52.705		2.025	2.075	
D1	48.260	48.260	**Reference**	1.900	1.900	**Reference**
E	15.240	15.875		0.600	0.625	
E1	12.954	15.240		0.510	0.600	
e1	2.540	2.540	**Reference**	0.100	0.100	**Reference**
eA	14.986	16.002	**Typical**	0.590	0.630	**Typical**
eB	15.240	18.034		0.600	0.710	
L	3.175	3.810		0.125	0.150	
N	40	40		40	40	
S	1.016	2.286		0.040	0.090	
S1	0.381	1.778		0.015	0.070	

APPENDIX A: MODIFICATIONS

The following is the list of modifications over the PIC16CXX microcontroller family:

1. Instruction word length is increased to 16-bit. This allows larger page sizes both in program memory (8 Kwords verses 2 Kwords) and register file (256 bytes versus 128 bytes).
2. Four modes of operation: microcontroller, protected microcontroller, extended microcontroller, and microprocessor.
3. 22 new instructions.
   The MOVF, TRIS and OPTION instructions have been removed.
4. 4 new instructions for transferring data between data memory and program memory. This can be used to "self program" the EPROM program memory.
5. Single cycle data memory to data memory transfers possible (MOVPF and MOVFP instructions). These instructions do not affect the Working register (WREG).
6. W register (WREG) is now directly addressable.
7. A PC high latch register (PCLATH) is extended to 8-bits. The PCLATCH register is now both readable and writable.
8. Data memory paging is redefined slightly.
9. DDR registers replaces function of TRIS registers.
10. Multiple Interrupt vectors added. This can decrease the latency for servicing the interrupt.
11. Stack size is increased to 16 deep.
12. BSR register for data memory paging.
13. Wake up from SLEEP operates slightly differently.
14. The Oscillator Start-Up Timer (OST) and Power-Up Timer (PWRT) operate in parallel and not in series.
15. PORTB interrupt on change feature works on all eight port pins.
16. TMR0 is 16-bit plus 8-bit prescaler.
17. Second indirect addressing register added (FSR1 and FSR2). Configuration bits can select the FSR registers to auto-increment, auto-decrement, remain unchanged after an indirect address.
18. Hardware multiplier added (8 x 8 $\rightarrow$ 16-bit) (PIC17C43 and PIC17C44 only).
19. Peripheral modules operate slightly differently.
20. Oscillator modes slightly redefined.
21. Control/Status bits and registers have been placed in different registers and the control bit for globally enabling interrupts has inverse polarity.
22. Addition of a test mode pin.
23. In-circuit serial programming is not implemented.

APPENDIX B: COMPATIBILITY

To convert code written for PIC16CXX to PIC17CXX, the user should take the following steps:

1. Remove any TRIS and OPTION instructions, and implement the equivalent code.
2. Separate the interrupt service routine into its four vectors.
3. Replace:
   ```
   MOVF    REG1, W
   ```
   with:
   ```
   MOVFP   REG1, WREG
   ```
4. Replace:
   ```
   MOVF    REG1, W
   MOVWF   REG2
   ```
   with:
   ```
   MOVPF   REG1, REG2 ; Addr(REG1)<20h
   ```
   or
   ```
   MOVFP   REG1, REG2 ; Addr(REG2)<20h
   ```

   Note: If REG1 and REG2 are both at addresses greater then 20h, two instructions are required.
   ```
   MOVFP   REG1, WREG ;
   MOVPF   WREG, REG2 ;
   ```
5. Ensure that all bit names and register names are updated to new data memory map location.
6. Verify data memory banking.
7. Verify mode of operation for indirect addressing.
8. Verify peripheral routines for compatibility.
9. Weak pull-ups are enabled on reset.

To convert code from the PIC17C42 to all the other PIC17C4X devices, the user should take the following steps.

1. If the hardware multiply is to be used, ensure that any variables at address 18h and 19h are moved to another address.
2. Ensure that the upper nibble of the BSR was not written with a non-zero value. This may cause unexpected operation since the RAM bank is no longer 0.
3. The disabling of global interrupts has been enhanced so there is no additional testing of the GLINTD bit after a `BSF CPUSTA, GLINTD` instruction.

NOTES: