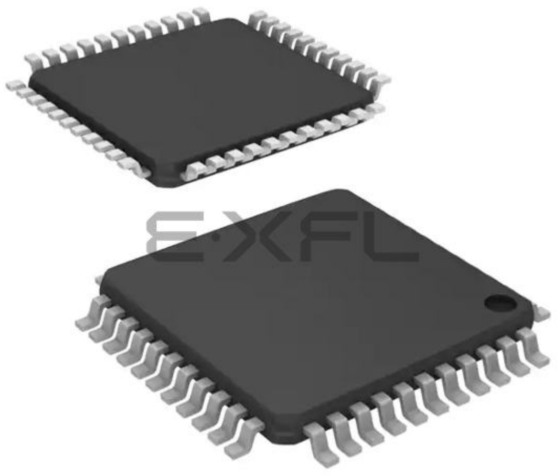


What is "Embedded - Microcontrollers"?

"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	8MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	4KB (2K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	232 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 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/pic17lc42a-08i-pt

3.1 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3, and Q4. Internally, the program counter (PC) is incremented every Q1, and the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 3-3.

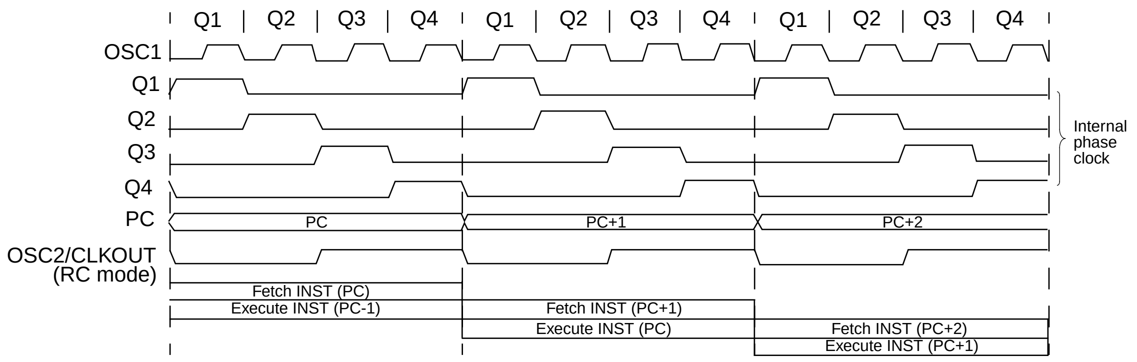
3.2 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3, and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g. `GOTO`) then two cycles are required to complete the instruction (Example 3-2).

A fetch cycle begins with the program counter incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the "Instruction Register (IR)" in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 3-3: CLOCK/INSTRUCTION CYCLE

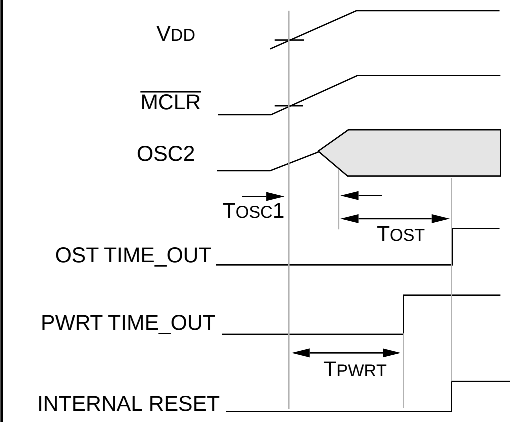


EXAMPLE 3-2: INSTRUCTION PIPELINE FLOW

	Tcy0	Tcy1	Tcy2	Tcy3	Tcy4	Tcy5
1. `MOVLW 55h`	Fetch 1	Execute 1				
2. `MOVWF PORTB`		Fetch 2	Execute 2			
3. `CALL SUB_1`			Fetch 3	Execute 3		
4. `BSF   PORTA, BIT3 (Forced NOP)`				Fetch 4	Flush	
5. `Instruction @ address SUB_1`					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

FIGURE 4-5: OSCILLATOR START-UP TIME



This figure shows in greater detail the timings involved with the oscillator start-up timer. In this example the low frequency crystal start-up time is larger than power-up time (TPWRT).
Tosc1 = time for the crystal oscillator to react to an oscillation level detectable by the Oscillator Start-up Timer (ost).
TOST = 1024TOSC.

FIGURE 4-6: USING ON-CHIP POR

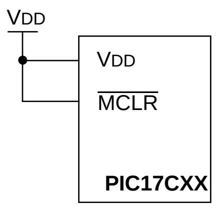
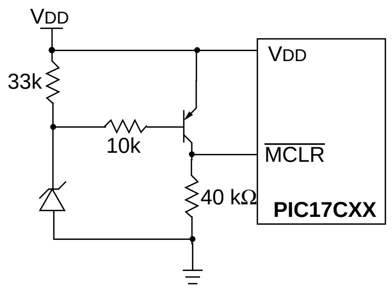
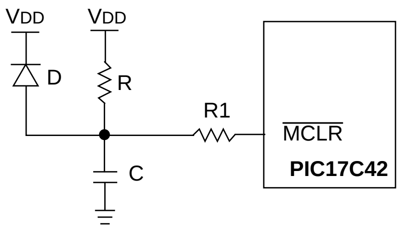


FIGURE 4-7: BROWN-OUT PROTECTION CIRCUIT 1



This circuit will activate reset when VDD goes below (Vz + 0.7V) where Vz = Zener voltage.

FIGURE 4-8: PIC17C42 EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)

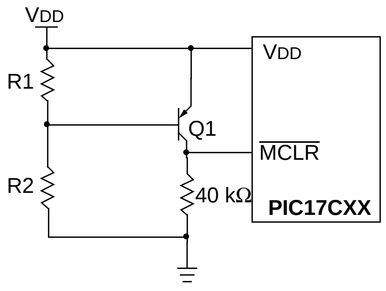


Note 1: An external Power-on Reset circuit is required only if VDD power-up time is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to ensure that the voltage drop across R does not exceed 0.2V (max. leakage current spec. on the $\overline{\text{MCLR}}$/VPP pin is 5 μA). A larger voltage drop will degrade VIH level on the $\overline{\text{MCLR}}$/VPP pin.

3: R1 = 100Ω to 1 kΩ will limit any current flowing into $\overline{\text{MCLR}}$ from external capacitor C in the event of $\overline{\text{MCLR}}$/VPP pin breakdown due to Electrostatic Discharge (ESD) or (Electrical Overstress) EOS.

FIGURE 4-9: BROWN-OUT PROTECTION CIRCUIT 2



This brown-out circuit is less expensive, albeit less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

$$V_{DD} \bullet \frac{R1}{R1 + R2} = 0.7V$$

5.4 Interrupt Operation

Global Interrupt Disable bit, GLINTD (CPUSTA<4>), enables all unmasked interrupts (if clear) or disables all interrupts (if set). Individual interrupts can be disabled through their corresponding enable bits in the INTSTA register. Peripheral interrupts need either the global peripheral enable PEIE bit disabled, or the specific peripheral enable bit disabled. Disabling the peripherals via the global peripheral enable bit, disables all peripheral interrupts. GLINTD is set on reset (interrupts disabled).

The `RETFIE` instruction allows returning from interrupt and re-enable interrupts at the same time.

When an interrupt is responded to, the GLINTD bit is automatically set to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with interrupt vector. There are four interrupt vectors to reduce interrupt latency.

The peripheral interrupt vector has multiple interrupt sources. Once in the peripheral interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The peripheral interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid continuous interrupts.

The PIC17C4X devices have four interrupt vectors. These vectors and their hardware priority are shown in Table 5-1. If two enabled interrupts occur “at the same time”, the interrupt of the highest priority will be serviced first. This means that the vector address of that interrupt will be loaded into the program counter (PC).

TABLE 5-1: INTERRUPT VECTORS/ PRIORITIES

Address	Vector	Priority
0008h	External Interrupt on RA0/ INT pin (INTF)	1 (Highest)
0010h	TMR0 overflow interrupt (T0IF)	2
0018h	External Interrupt on T0CKI (T0CKIF)	3
0020h	Peripherals (PEIF)	4 (Lowest)

Note 1: Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GLINTD bit.

Note 2: When disabling any of the INTSTA enable bits, the GLINTD bit should be set (disabled).

Note 3: For the PIC17C42 only:
If an interrupt occurs while the Global Interrupt Disable (GLINTD) bit is being set, the GLINTD bit may unintentionally be re-enabled by the user’s Interrupt Service Routine (the `RETFIE` instruction). The events that would cause this to occur are:

1. An interrupt occurs simultaneously with an instruction that sets the GLINTD bit.
2. The program branches to the Interrupt vector and executes the Interrupt Service Routine.
3. The Interrupt Service Routine completes with the execution of the `RETFIE` instruction. This causes the GLINTD bit to be cleared (enables interrupts), and the program returns to the instruction after the one which was meant to disable interrupts.

The method to ensure that interrupts are globally disabled is:

1. Ensure that the GLINTD bit was set by the instruction, as shown in the following code:

```
LOOP   BSF    CPUSTA, GLINTD ; Disable Global
                             ; Interrupt
       BTFSS  CPUSTA, GLINTD ; Global Interrupt
                             ; Disabled?
       GOTO   LOOP           ; NO, try again
                             ; YES, continue
                             ; with program
                             ; low
```

6.2 Data Memory Organization

Data memory is partitioned into two areas. The first is the General Purpose Registers (GPR) area, while the second is the Special Function Registers (SFR) area. The SFRs control the operation of the device.

Portions of data memory are banked, this is for both areas. The GPR area is banked to allow greater than 232 bytes of general purpose RAM. SFRs are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the Bank Select Register (BSR). If an access is made to a location outside this banked region, the BSR bits are ignored. Figure 6-5 shows the data memory map organization for the PIC17C42 and Figure 6-6 for all of the other PIC17C4X devices.

Instructions `MOVPF` and `MOVFP` provide the means to move values from the peripheral area ("P") to any location in the register file ("F"), and vice-versa. The definition of the "P" range is from 0h to 1Fh, while the "F" range is 0h to FFh. The "P" range has six more locations than peripheral registers (eight locations for the PIC17C42 device) which can be used as General Purpose Registers. This can be useful in some applications where variables need to be copied to other locations in the general purpose RAM (such as saving status information during an interrupt).

The entire data memory can be accessed either directly or indirectly through file select registers FSR0 and FSR1 (Section 6.4). Indirect addressing uses the appropriate control bits of the BSR for accesses into the banked areas of data memory. The BSR is explained in greater detail in Section 6.8.

6.2.1 GENERAL PURPOSE REGISTER (GPR)

All devices have some amount of GPR area. The GPRs are 8-bits wide. When the GPR area is greater than 232, it must be banked to allow access to the additional memory space.

Only the PIC17C43 and PIC17C44 devices have banked memory in the GPR area. To facilitate switching between these banks, the `MOVLR bank` instruction has been added to the instruction set. GPRs are not initialized by a Power-on Reset and are unchanged on all other resets.

6.2.2 SPECIAL FUNCTION REGISTERS (SFR)

The SFRs are used by the CPU and peripheral functions to control the operation of the device (Figure 6-5 and Figure 6-6). These registers are static RAM.

The SFRs can be classified into two sets, those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described here, while those related to a peripheral feature are described in the section for each peripheral feature.

The peripheral registers are in the banked portion of memory, while the core registers are in the unbanked region. To facilitate switching between the peripheral banks, the `MOVLB bank` instruction has been provided.

6.7 Program Counter Module

The Program Counter (PC) is a 16-bit register. PCL, the low byte of the PC, is mapped in the data memory. PCL is readable and writable just as is any other register. PCH is the high byte of the PC and is not directly addressable. Since PCH is not mapped in data or program memory, an 8-bit register PCLATH (PC high latch) is used as a holding latch for the high byte of the PC. PCLATH is mapped into data memory. The user can read or write PCH through PCLATH.

The 16-bit wide PC is incremented after each instruction fetch during Q1 unless:

- Modified by `GOTO`, `CALL`, `LCALL`, `RETURN`, `RETLW`, or `RETFIE` instruction
- Modified by an interrupt response
- Due to destination write to PCL by an instruction

"Skips" are equivalent to a forced NOP cycle at the skipped address.

Figure 6-11 and Figure 6-12 show the operation of the program counter for various situations.

FIGURE 6-11: PROGRAM COUNTER OPERATION

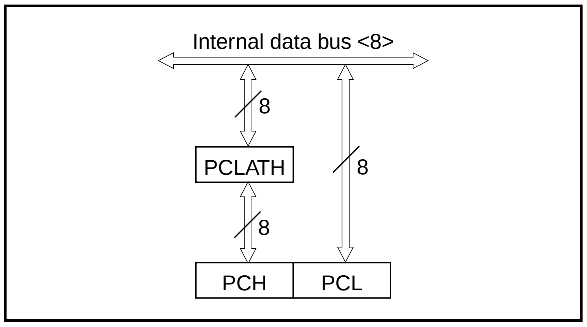
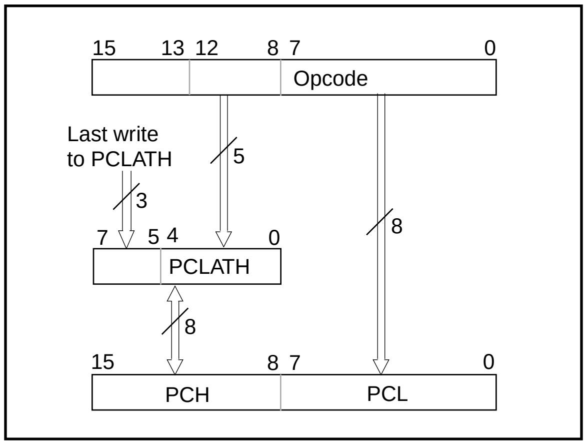


FIGURE 6-12: PROGRAM COUNTER USING THE CALL AND GOTO INSTRUCTIONS



Using Figure 6-11, the operations of the PC and PCLATH for different instructions are as follows:

a) `LCALL` instructions:

An 8-bit destination address is provided in the instruction (opcode). PCLATH is unchanged.

PCLATH $\rightarrow$ PCH

Opcode<7:0> $\rightarrow$ PCL

b) Read instructions on PCL:

Any instruction that reads PCL.

PCL $\rightarrow$ data bus $\rightarrow$ ALU or destination

PCH $\rightarrow$ PCLATH

c) Write instructions on PCL:

Any instruction that writes to PCL.

8-bit data $\rightarrow$ data bus $\rightarrow$ PCL

PCLATH $\rightarrow$ PCH

d) Read-Modify-Write instructions on PCL:

Any instruction that does a read-write-modify operation on PCL, such as `ADDWF PCL`.

Read: PCL $\rightarrow$ data bus $\rightarrow$ ALU

Write: 8-bit result $\rightarrow$ data bus $\rightarrow$ PCL

PCLATH $\rightarrow$ PCH

e) `RETURN` instruction:

PCH $\rightarrow$ PCLATH

Stack<MRU> $\rightarrow$ PC<15:0>

Using Figure 6-12, the operation of the PC and PCLATH for `GOTO` and `CALL` instructions is a follows:

`CALL`, `GOTO` instructions:

A 13-bit destination address is provided in the instruction (opcode).

Opcode<12:0> $\rightarrow$ PC <12:0>

PC<15:13> $\rightarrow$ PCLATH<7:5>

Opcode<12:8> $\rightarrow$ PCLATH <4:0>

The read-modify-write only affects the PCL with the result. PCH is loaded with the value in the PCLATH. For example, `ADDWF PCL` will result in a jump within the current page. If PC = 03F0h, WREG = 30h and PCLATH = 03h before instruction, PC = 0320h after the instruction. To accomplish a true 16-bit computed jump, the user needs to compute the 16-bit destination address, write the high byte to PCLATH and then write the low value to PCL.

The following PC related operations do not change PCLATH:

a) `LCALL`, `RETLW`, and `RETFIE` instructions.

b) Interrupt vector is forced onto the PC.

c) Read-modify-write instructions on PCL (e.g. `BSF PCL`).

NOTES:

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)

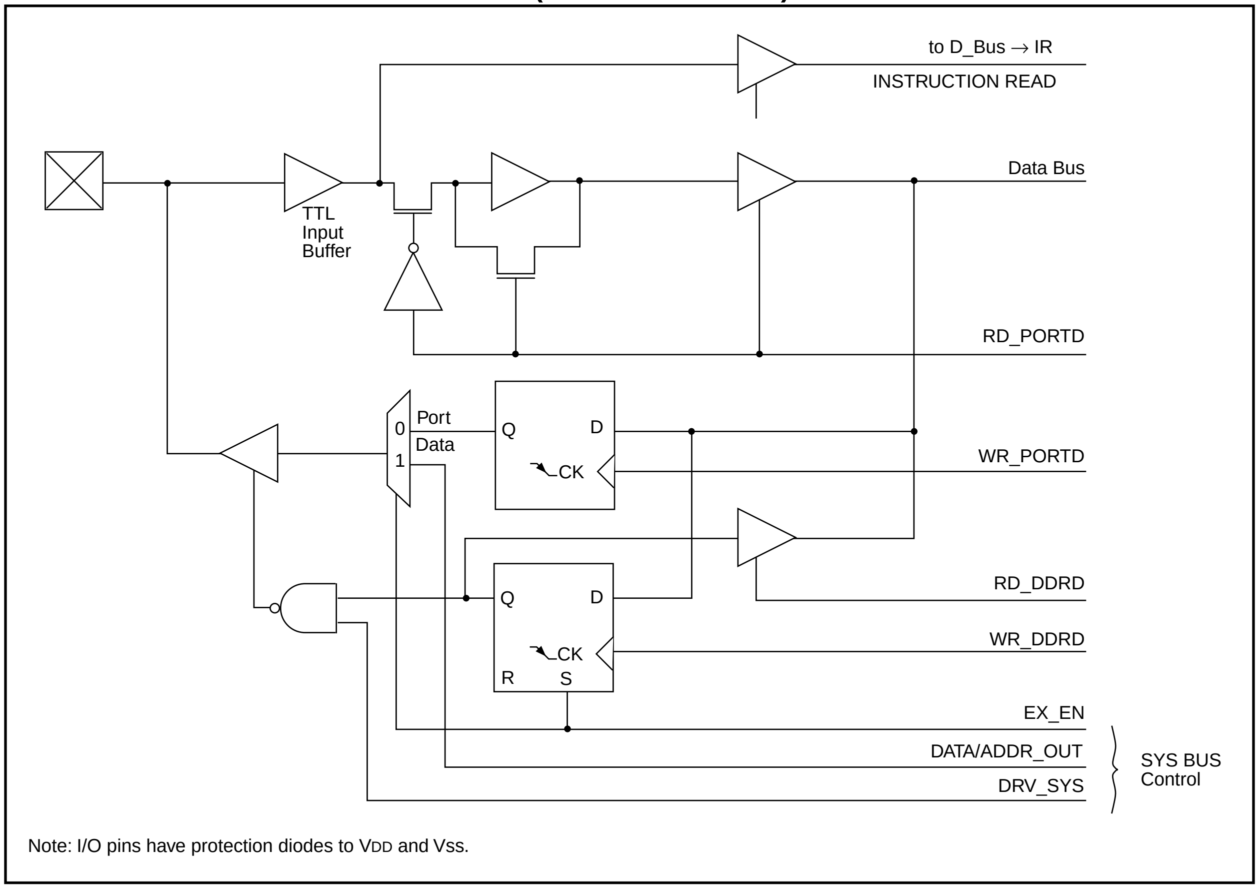


TABLE 13-3: BAUD RATES FOR SYNCHRONOUS MODE

BAUD RATE (K)	FOSC = 33 MHz KBAUD	%ERROR	SPBRG value (decimal)	FOSC = 25 MHz KBAUD	%ERROR	SPBRG value (decimal)	FOSC = 20 MHz KBAUD	%ERROR	SPBRG value (decimal)	FOSC = 16 MHz KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	NA	—	—	NA	—	—	NA	—	—
19.2	NA	—	—	NA	—	—	19.53	+1.73	255	19.23	+0.16	207
76.8	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64	76.92	+0.16	51
96	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51	95.24	-0.79	41
300	294.64	-1.79	27	297.62	-0.79	20	294.1	-1.96	16	307.69	+2.56	12
500	485.29	-2.94	16	480.77	-3.85	12	500	0	9	500	0	7
HIGH	8250	—	0	6250	—	0	5000	—	0	4000	—	0
LOW	32.22	—	255	24.41	—	255	19.53	—	255	15.625	—	255

BAUD RATE (K)	FOSC = 10 MHz KBAUD	%ERROR	SPBRG value (decimal)	FOSC = 7.159 MHz KBAUD	%ERROR	SPBRG value (decimal)	FOSC = 5.068 MHz KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—
9.6	9.766	+1.73	255	9.622	+0.23	185	9.6	0	131
19.2	19.23	+0.16	129	19.24	+0.23	92	19.2	0	65
76.8	75.76	-1.36	32	77.82	+1.32	22	79.2	+3.13	15
96	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	312.5	+4.17	7	298.3	-0.57	5	316.8	+5.60	3
500	500	0	4	NA	—	—	NA	—	—
HIGH	2500	—	0	1789.8	—	0	1267	—	0
LOW	9.766	—	255	6.991	—	255	4.950	—	255

BAUD RATE (K)	FOSC = 3.579 MHz KBAUD	%ERROR	SPBRG value (decimal)	FOSC = 1 MHz KBAUD	%ERROR	SPBRG value (decimal)	FOSC = 32.768 kHz KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	—	—	NA	—	—	0.303	+1.14	26
1.2	NA	—	—	1.202	+0.16	207	1.170	-2.48	6
2.4	NA	—	—	2.404	+0.16	103	NA	—	—
9.6	9.622	+0.23	92	9.615	+0.16	25	NA	—	—
19.2	19.04	-0.83	46	19.24	+0.16	12	NA	—	—
76.8	74.57	-2.90	11	83.34	+8.51	2	NA	—	—
96	99.43	_3.57	8	NA	—	—	NA	—	—
300	298.3	-0.57	2	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	894.9	—	0	250	—	0	8.192	—	0
LOW	3.496	—	255	0.976	—	255	0.032	—	255

13.2 USART Asynchronous Mode

In this mode, the USART uses standard nonreturn-to-zero (NRZ) format (one start bit, eight or nine data bits, and one stop bit). The most common data format is 8-bits. An on-chip dedicated 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART's transmitter and receiver are functionally independent but use the same data format and baud rate. The baud rate generator produces a clock x64 of the bit shift rate. Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

The asynchronous mode is selected by clearing the SYNC bit (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

13.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 13-3. The heart of the transmitter is the transmit shift register (TSR). The shift register obtains its data from the read/write transmit buffer (TXREG). TXREG is loaded with data in software. The TSR is not loaded until the stop bit has been transmitted from the previous load. As soon as the stop bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one TCY at the end of the current BRG cycle), the TXREG is empty and an interrupt bit, TXIF (PIR<1>) is set. This interrupt can be enabled or disabled by the TXIE bit (PIE<1>). TXIF will be set regardless of TXIE and cannot be reset in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of the TXREG, the TRMT (TXSTA<1>) bit shows the status of the TSR. TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR is empty.

Note: The TSR is not mapped in data memory, so it is not available to the user.

Transmission is enabled by setting the TXEN (TXSTA<5>) bit. The actual transmission will not occur until TXREG has been loaded with data and the baud rate generator (BRG) has produced a shift clock (Figure 13-5). The transmission can also be started by first loading TXREG and then setting TXEN. Normally when transmission is first started, the TSR is empty, so a transfer to TXREG will result in an immediate transfer to TSR resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 13-6). Clearing TXEN during a transmission will cause the transmission to be aborted. This will reset the transmitter and the RA5/TX/CK pin will revert to hi-impedance.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG. This is because a data write to TXREG can result in an immediate transfer of the data to the TSR (if the TSR is empty).

Steps to follow when setting up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are desired, then set the TXIE bit.
4. If 9-bit transmission is desired, then set the TX9 bit.
5. Load data to the TXREG register.
6. If 9-bit transmission is selected, the ninth bit should be loaded in TX9D.
7. Enable the transmission by setting TXEN (starts transmission).

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 opposite 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.

TABLE 13-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

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)
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	`0000 0010`	`0000 0010`
13h, Bank 0	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	`0000 -00x`	`0000 -00u`
14h, Bank 0	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	`xxxx xxxx`	`uuuu uuuu`
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	`0000 0000`	`0000 0000`
15h, Bank 0	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	`0000 --1x`	`0000 --1u`
17h, Bank 0	SPBRG	Baud rate generator register								`xxxx xxxx`	`uuuu uuuu`

Legend: x = unknown, u = unchanged, - = unimplemented read as a '0', shaded cells are not used for synchronous master reception.

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

14.2.4 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator can be used or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used: one with series resonance, or one with parallel resonance.

Figure 14-5 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180-degree phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides the negative feedback for stability. The 10 kΩ potentiometer biases the 74AS04 in the linear region. This could be used for external oscillator designs.

FIGURE 14-5: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT

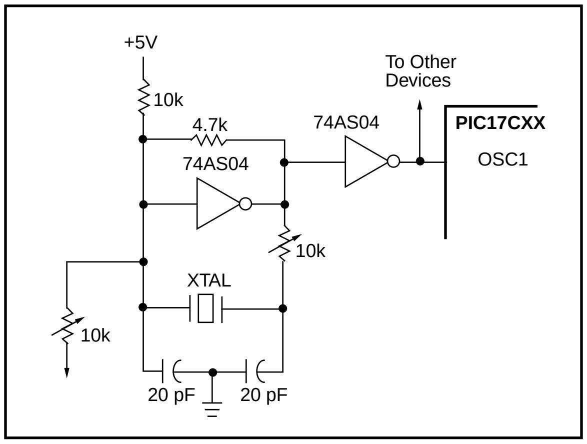
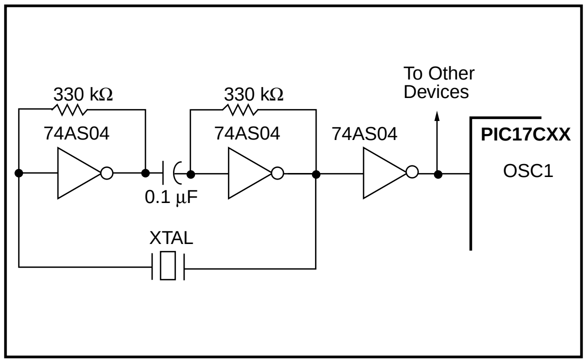


Figure 14-6 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330 kΩ resistors provide the negative feedback to bias the inverters in their linear region.

FIGURE 14-6: EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT



14.2.5 RC OSCILLATOR

For timing insensitive applications, the RC device option offers additional cost savings. RC oscillator frequency is a function of the supply voltage, the resistor (Rext) and capacitor (Cext) values, and the operating temperature. In addition to this, oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect oscillation frequency, especially for low Cext values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 14-6 shows how the R/C combination is connected to the PIC17CXX. For Rext values below 2.2 kΩ, the oscillator operation may become unstable, or stop completely. For very high Rext values (e.g. 1 MΩ), the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend to keep Rext between 3 kΩ and 100 kΩ.

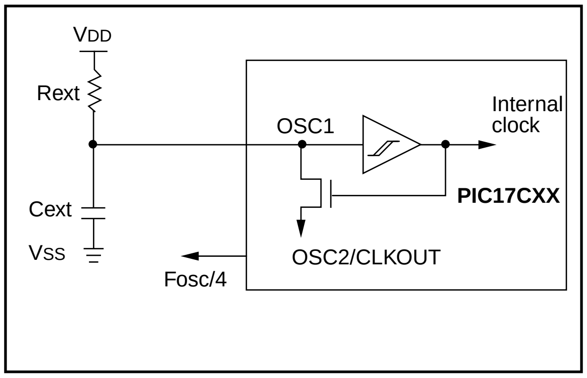
Although the oscillator will operate with no external capacitor (Cext = 0 pF), we recommend using values above 20 pF for noise and stability reasons. With little or no external capacitance, oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance or package lead frame capacitance.

See Section 18.0 for RC frequency variation from part to part due to normal process variation. The variation is larger for larger R (since leakage current variation will affect RC frequency more for large R) and for smaller C (since variation of input capacitance will affect RC frequency more).

See Section 18.0 for variation of oscillator frequency due to VDD for given Rext/Cext values as well as frequency variation due to operating temperature for given R, C, and VDD values.

The oscillator frequency, divided by 4, is available on the OSC2/CLKOUT pin, and can be used for test purposes or to synchronize other logic (see Figure 3-2 for waveform).

FIGURE 14-7: RC OSCILLATOR MODE



TLWT **Table Latch Write**

Syntax:	[*label*] TLWT t,f
Operands:	0 ≤ f ≤ 255 t ∈ [0,1]
Operation:	If t = 0, f → TBLATL; If t = 1, f → TBLATH
Status Affected:	None

Encoding:

`1010`	`01tx`	`ffff`	`ffff`

Description: Data from file register 'f' is written into the 16-bit table latch (TBLAT).

If t = 1; high byte is written

If t = 0; low byte is written

This instruction is used in conjunction with `TABLWT` to transfer data from data memory to program memory.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register TBLATH or TBLATL

Example: `TLWT t, RAM`

Before Instruction

```
t      =   0
RAM    =   0xB7
TBLAT  =   0x0000   (TBLATH = 0x00)
                    (TBLATL = 0x00)
```

After Instruction

```
RAM    =   0xB7
TBLAT  =   0x00B7   (TBLATH = 0x00)
                    (TBLATL = 0xB7)
```

Before Instruction

```
t      =   1
RAM    =   0xB7
TBLAT  =   0x0000   (TBLATH = 0x00)
                    (TBLATL = 0x00)
```

After Instruction

```
RAM    =   0xB7
TBLAT  =   0xB700   (TBLATH = 0xB7)
                    (TBLATL = 0x00)
```

TSTFSZ **Test f, skip if 0**

Syntax:	[*label*] TSTFSZ f
Operands:	0 ≤ f ≤ 255
Operation:	skip if f = 0
Status Affected:	None

Encoding:

`0011`	`0011`	`ffff`	`ffff`

Description: If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and an NOP is executed making this a two-cycle instruction.

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	NOP

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:

```
HERE    TSTFSZ  CNT
NZERO     :
ZERO    :
```

Before Instruction

PC = Address(`HERE`)

After Instruction

```
If CNT      =   0x00,
    PC      =   Address (ZERO)
If CNT      ≠   0x00,
    PC      =   Address (NZERO)
```

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 18-13: WDT TIMER TIME-OUT PERIOD vs. VDD

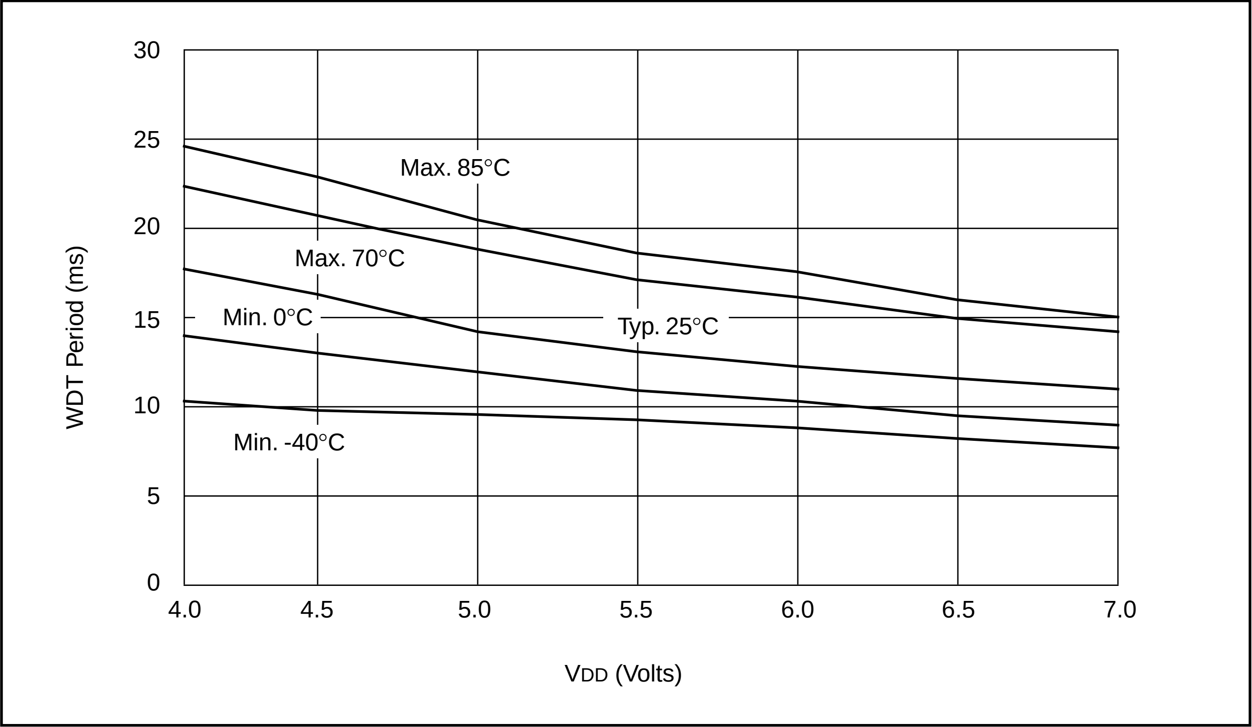
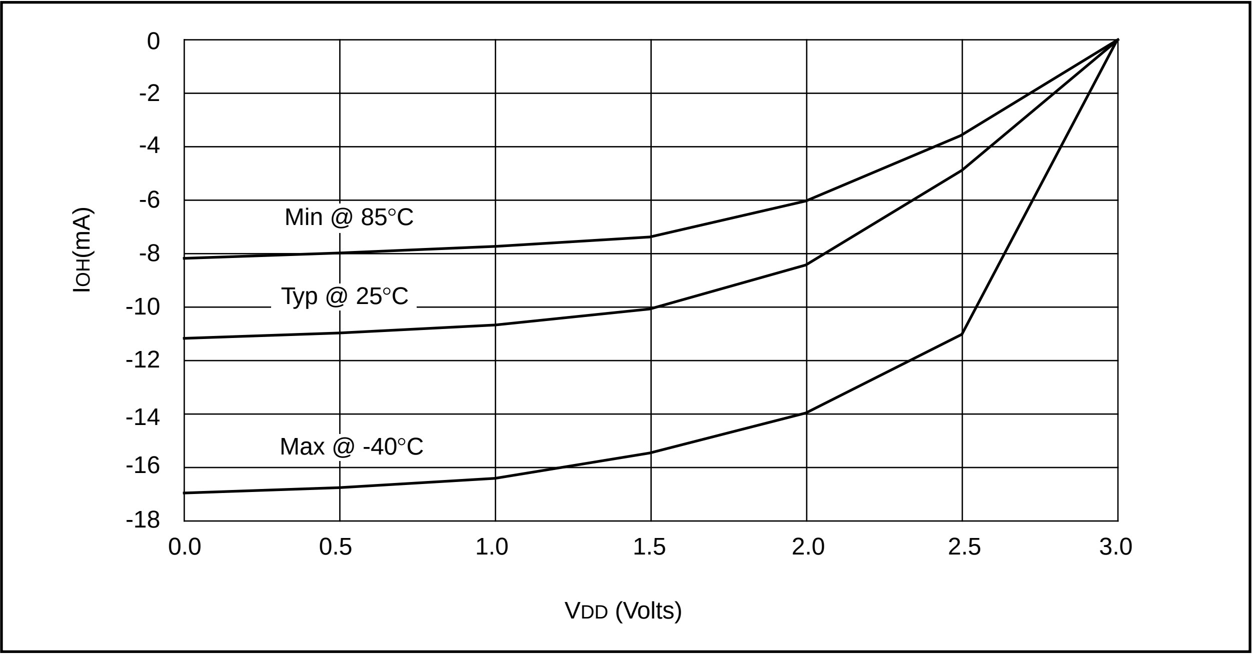


FIGURE 18-14: IOH vs. VOH, VDD = 3V



Applicable Devices	42	R42	42A	43	R43	44

19.0 PIC17CR42/42A/43/R43/44 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias	-55 to +125˚C
Storage temperature	-65˚C to +150˚C
Voltage on V_{DD} with respect to V_{SS}	0 to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} (Note 2)	-0.6V to +14V
Voltage on RA2 and RA3 with respect to V_{SS}	-0.6V to +14V
Voltage on all other pins with respect to V_{SS}	-0.6V to V_{DD} + 0.6V
Total power dissipation (Note 1)	1.0W
Maximum current out of V_{SS} pin(s) - total	250 mA
Maximum current into V_{DD} pin(s) - total	200 mA
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{DD}$)	±20 mA
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{DD}$)	±20 mA
Maximum output current sunk by any I/O pin (except RA2 and RA3)	35 mA
Maximum output current sunk by RA2 or RA3 pins	60 mA
Maximum output current sourced by any I/O pin	20 mA
Maximum current sunk by PORTA and PORTB (combined)	150 mA
Maximum current sourced by PORTA and PORTB (combined)	100 mA
Maximum current sunk by PORTC, PORTD and PORTE (combined)	150 mA
Maximum current sourced by PORTC, PORTD and PORTE (combined)	100 mA

Note 1: Power dissipation is calculated as follows: Pdis = V_{DD} x {I_{DD} - $\sum$ I_{OH}} + $\sum$ {(V_{DD}-V_{OH}) x I_{OH}} + $\sum$(V_{OL} x I_{OL})

Note 2: Voltage spikes below V_{SS} at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin rather than pulling this pin directly to V_{SS}.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Applicable Devices	42	R42	42A	43	R43	44

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

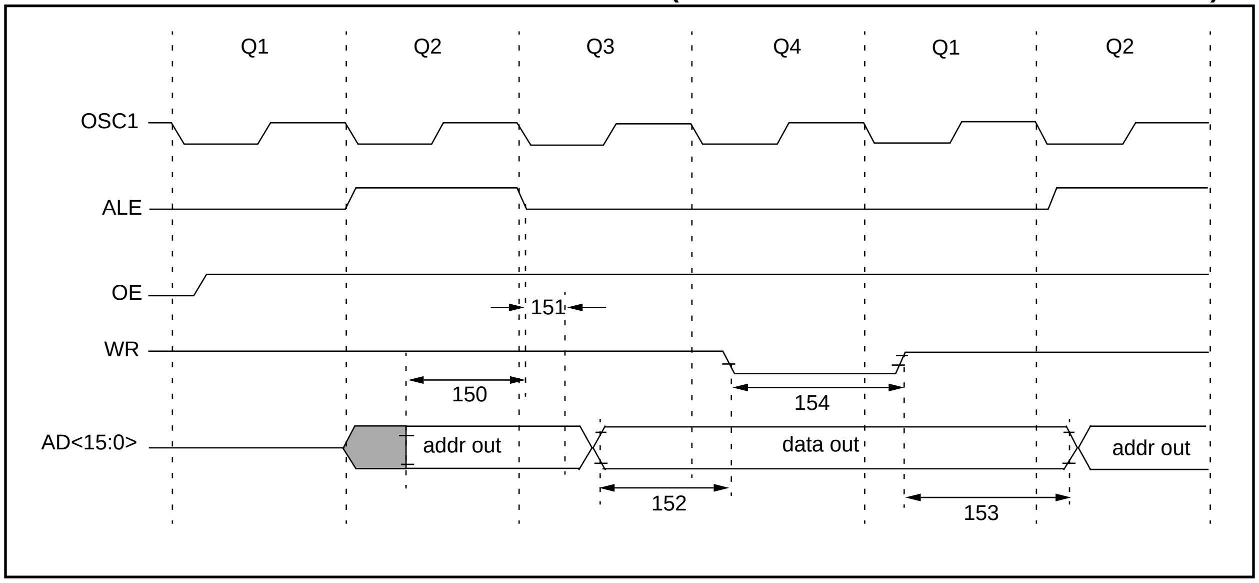


TABLE 19-11: MEMORY INTERFACE WRITE REQUIREMENTS (NOT SUPPORTED IN PIC17LC4X DEVICES)

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

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

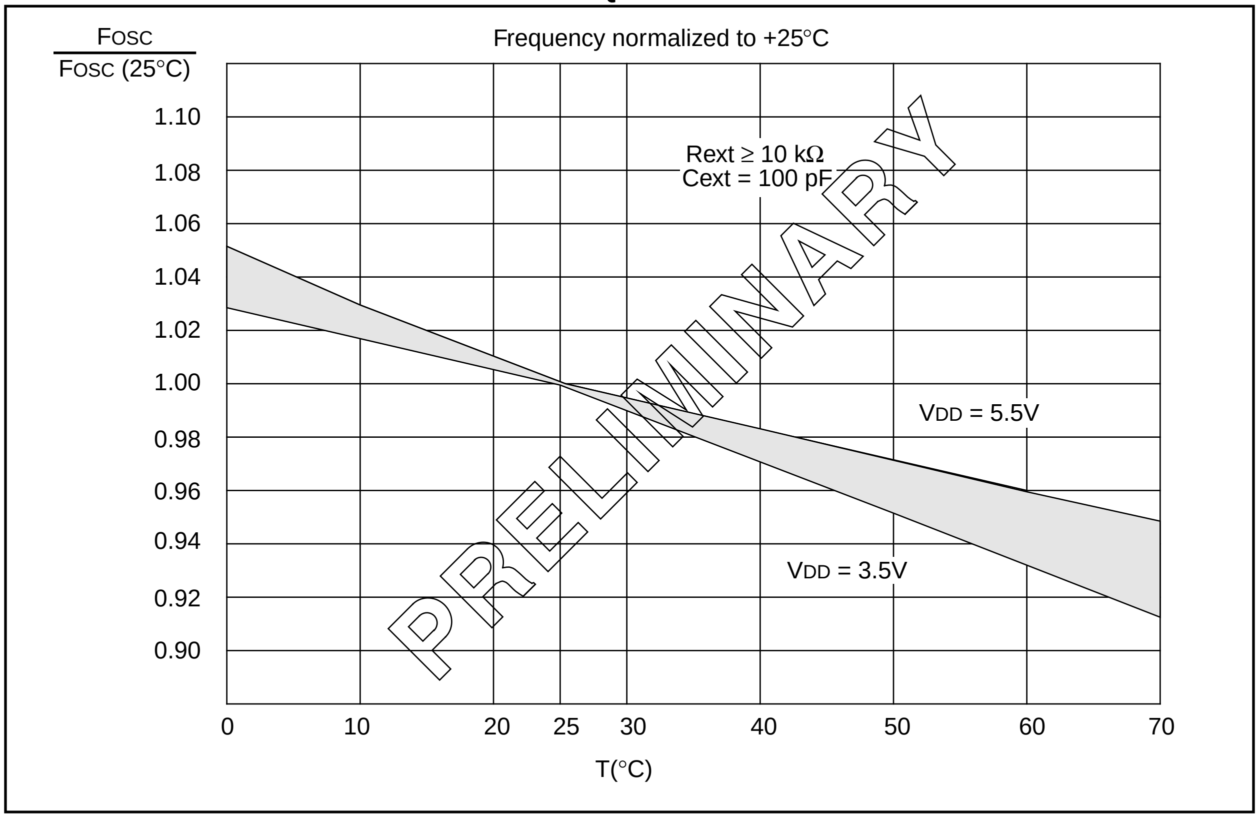
The graphs and tables provided in this section are for design guidance and are not tested nor guaranteed. In some graphs or tables the data presented is outside specified operating range (e.g. outside specified VDD range). This is for information only and devices are ensured to operate properly only 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 $\overline{\text{MCLR}}$, VDD, and VSS	10	10	10	10
$\overline{\text{MCLR}}$ pin	20	20	20	20

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



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-4: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD

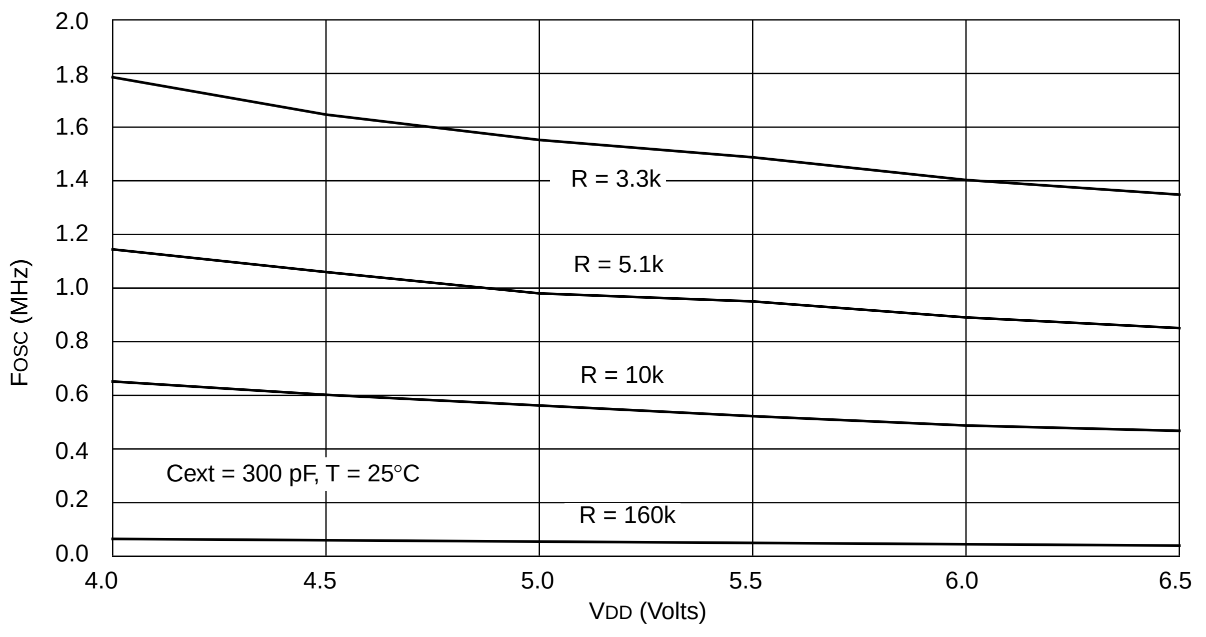


TABLE 20-2: RC OSCILLATOR FREQUENCIES

Cext	Rext	Average Fosc @ 5V, 25°C	
22 pF	10k	3.33 MHz	± 12%
	100k	353 kHz	± 13%
100 pF	3.3k	3.54 MHz	± 10%
	5.1k	2.43 MHz	± 14%
	10k	1.30 MHz	± 17%
	100k	129 kHz	± 10%
300 pF	3.3k	1.54 MHz	± 14%
	5.1k	980 kHz	± 12%
	10k	564 kHz	± 16%
	160k	35 kHz	± 18%

E.3 PIC16CXXX Family of Devices

	Clock	Memory		Peripherals			Features				
	Maximum Frequency of Operation (MHz)	Program Memory (x14 words) EPROM	Data Memory (bytes)	Timer Module(s)	Comparator(s)	Internal Reference Voltage	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Brown-out Reset	Packages
PIC16C554	20	512	80	TMR0	—	—	3	13	2.5-6.0	—	18-pin DIP, SOIC; 20-pin SSOP
PIC16C556	20	1K	80	TMR0	—	—	3	13	2.5-6.0	—	18-pin DIP, SOIC; 20-pin SSOP
PIC16C558	20	2K	128	TMR0	—	—	3	13	2.5-6.0	—	18-pin DIP, SOIC; 20-pin SSOP
PIC16C620	20	512	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C621	20	1K	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C622	20	2K	128	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.
All PIC16C6XXX Family devices use serial programming with clock pin RB6 and data pin RB7.

INDEX