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	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	66
Program Memory Size	16KB (8K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	678 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	84-LCC (J-Lead)
Supplier Device Package	84-PLCC (29.31x29.31)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c762t-16-l

TABLE 1-1: PIC17CXXX FAMILY OF DEVICES

Features		PIC17C42A	PIC17C43	PIC17C44	PIC17C752	PIC17C756A	PIC17C762	PIC17C766
Maximum Frequency of Operation		33 MHz	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz
Operating Voltage Range		2.5 - 6.0V	2.5 - 6.0V	2.5 - 6.0V	3.0 - 5.5V	3.0 - 5.5V	3.0 - 5.5V	3.0 - 5.5V
Program Memory (x16)	(EPROM)	2 K	4 K	8 K	8 K	16 K	8 K	16 K
	(ROM)	—	—	—	—	—	—	—
Data Memory (bytes)		232	454	454	678	902	678	902
Hardware Multiplier (8 x 8)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Timer0 (16-bit + 8-bit postscaler)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Timer1 (8-bit)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Timer2 (8-bit)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Timer3 (16-bit)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Capture inputs (16-bit)		2	2	2	4	4	4	4
PWM outputs (up to 10-bit)		2	2	2	3	3	3	3
USART/SCI		1	1	1	2	2	2	2
A/D channels (10-bit)		—	—	—	12	12	16	16
SSP (SPI/I^2C w/Master mode)		—	—	—	Yes	Yes	Yes	Yes
Power-on Reset		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Watchdog Timer		Yes	Yes	Yes	Yes	Yes	Yes	Yes
External Interrupts		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Interrupt Sources		11	11	11	18	18	18	18
Code Protect		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Brown-out Reset		—	—	—	Yes	Yes	Yes	Yes
In-Circuit Serial Programming		—	—	—	Yes	Yes	Yes	Yes
I/O Pins		33	33	33	50	50	66	66
I/O High Current Capability	Source	25 mA	25 mA	25 mA	25 mA	25 mA	25 mA	25 mA
	Sink	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)
Package Types		40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	64-pin TQFP 68-pin PLCC	64-pin TQFP 68-pin PLCC	80-pin TQFP 84-pin PLCC	80-pin TQFP 84-pin PLCC

Note 1: Pins RA2 and RA3 can sink up to 60 mA.

TABLE 3-1: PINOUT DESCRIPTIONS

Name	PIC17C75X			PIC17C76X		I/O/P Type	Buffer Type	Description
	DIP No.	PLCC No.	TQFP No.	PLCC No.	QFP No.			
OSC1/CLKIN	47	50	39	62	49	I	ST	Oscillator input in Crystal/Resonator or RC Oscillator mode. External clock input in External Clock mode.
OSC2/CLKOUT	48	51	40	63	50	O	—	Oscillator output. Connects to crystal or resonator in Crystal Oscillator mode. In RC Oscillator or External Clock modes, OSC2 pin outputs CLKOUT which has one fourth the frequency (FOSC/4) of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	15	16	7	20	9	I/P	ST	Master clear (RESET) input or Programming Voltage (VPP) input. This is the active low RESET input to the device.
								PORTA pins have individual differentiations that are listed in the following descriptions:
RA0/INT	56	60	48	72	58	I	ST	RA0 can also be selected as an external interrupt input. Interrupt can be configured to be on positive or negative edge. Input only pin.
RA1/T0CKI	41	44	33	56	43	I	ST	RA1 can also be selected as an external interrupt input and the interrupt can be configured to be on positive or negative edge. RA1 can also be selected to be the clock input to the Timer0 timer/counter. Input only pin.
RA2/$\overline{SS}$/SCL	42	45	34	57	44	I/O(2)	ST	RA2 can also be used as the slave select input for the SPI or the clock input for the I^2C bus. High voltage, high current, open drain port pin.
RA3/SDI/SDA	43	46	35	58	45	I/O(2)	ST	RA3 can also be used as the data input for the SPI or the data for the I^2C bus. High voltage, high current, open drain port pin.
RA4/RX1/DT1	40	43	32	51	38	I/O(1)	ST	RA4 can also be selected as the USART1 (SCI) Asynchronous Receive or USART1 (SCI) Synchronous Data. Output available from USART only.
RA5/TX1/CK1	39	42	31	50	37	I/O(1)	ST	RA5 can also be selected as the USART1 (SCI) Asynchronous Transmit or USART1 (SCI) Synchronous Clock. Output available from USART only.
								PORTB is a bi-directional I/O Port with software configurable weak pull-ups.
RB0/CAP1	55	59	47	71	57	I/O	ST	RB0 can also be the Capture1 input pin.
RB1/CAP2	54	58	46	70	56	I/O	ST	RB1 can also be the Capture2 input pin.
RB2/PWM1	50	54	42	66	52	I/O	ST	RB2 can also be the PWM1 output pin.
RB3/PWM2	53	57	45	69	55	I/O	ST	RB3 can also be the PWM2 output pin.
RB4/TCLK12	52	56	44	68	54	I/O	ST	RB4 can also be the external clock input to Timer1 and Timer2.
RB5/TCLK3	51	55	43	67	53	I/O	ST	RB5 can also be the external clock input to Timer3.
RB6/SCK	44	47	36	59	46	I/O	ST	RB6 can also be used as the master/slave clock for the SPI.
RB7/SDO	45	48	37	60	47	I/O	ST	RB7 can also be used as the data output for the SPI.

Legend: I = Input only; O = Output only; I/O = Input/Output;
P = Power; — = Not Used; TTL = TTL input; ST = Schmitt Trigger input

Note 1: The output is only available by the peripheral operation.
2: Open drain input/output pin. Pin forced to input upon any device RESET.

6.4 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 clears the GLINTD bit while forcing the Program Counter (PC) to the value loaded at the Top-of-Stack.

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 the interrupt vector. There are four interrupt vectors which help 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 PIC17C7XX devices have four interrupt vectors. These vectors and their hardware priority are shown in Table 6-1. 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 6-1: 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.

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

6.5 RA0/INT Interrupt

The external interrupt on the RA0/INT pin is edge triggered. Either the rising edge if the INTEDG bit (T0STA<7>) is set, or the falling edge if the INTEDG bit is clear. When a valid edge appears on the RA0/INT pin, the INTF bit (INTSTA<4>) is set. This interrupt can be disabled by clearing the INTE control bit (INTSTA<0>). The INT interrupt can wake the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

6.6 T0CKI Interrupt

The external interrupt on the RA1/T0CKI pin is edge triggered. Either the rising edge if the T0SE bit (T0STA<6>) is set, or the falling edge if the T0SE bit is clear. When a valid edge appears on the RA1/T0CKI pin, the T0CKIF bit (INTSTA<6>) is set. This interrupt can be disabled by clearing the T0CKIE control bit (INTSTA<2>). The T0CKI interrupt can wake up the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

6.7 Peripheral Interrupt

The peripheral interrupt flag indicates that at least one of the peripheral interrupts occurred (PEIF is set). The PEIF bit is a read only bit and is a bit wise OR of all the flag bits in the PIR registers AND’d with the corresponding enable bits in the PIE registers. Some of the peripheral interrupts can wake the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

6.8 Context Saving During Interrupts

During an interrupt, only the returned PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt; e.g. WREG, ALUSTA and the BSR registers. This requires implementation in software.

Example 6-2 shows the saving and restoring of information for an Interrupt Service Routine. This is for a simple interrupt scheme, where only one interrupt may occur at a time (no interrupt nesting). The SFRs are stored in the non-banked GPR area.

Example 6-2 shows the saving and restoring of information for a more complex Interrupt Service Routine. This is useful where nesting of interrupts is required. A maximum of 6 levels can be done by this example. The BSR is stored in the non-banked GPR area, while the other registers would be stored in a particular bank. Therefore, 6 saves may be done with this routine (since there are 6 non-banked GPR registers). These routines require a dedicated indirect addressing register, FSR0, to be selected for this.

The PUSH and POP code segments could either be in each Interrupt Service Routine, or could be subroutines that were called. Depending on the application, other registers may also need to be saved.

TABLE 7-3: SPECIAL FUNCTION REGISTERS (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{\text{MCLR}}$, WDT
Bank 6											
10h	SSPADD	SSP Address Register in I^2C Slave mode. SSP Baud Rate Reload Register in I^2C Master mode								0000 0000	0000 0000
11h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
12h	SSPCON2	GCEN	AKSTAT	AKDT	AKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
13h	SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	0000 0000
14h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
15h	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
16h	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
17h	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
Bank 7											
10h	PW3DCL	DC1	DC0	TM2PW3	—	—	—	—	—	xx0- ----	uu0- ----
11h	PW3DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
12h	CA3L	Capture3 Low Byte								xxxx xxxx	uuuu uuuu
13h	CA3H	Capture3 High Byte								xxxx xxxx	uuuu uuuu
14h	CA4L	Capture4 Low Byte								xxxx xxxx	uuuu uuuu
15h	CA4H	Capture4 High Byte								xxxx xxxx	uuuu uuuu
16h	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000
17h	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
Bank 8(3)											
10h(3)	DDRH	Data Direction Register for PORTH								1111 1111	1111 1111
11h(3)	PORTH(4)	RH7/ AN15	RH6/ AN14	RH5/ AN13	RH4/ AN12	RH3	RH2	RH1	RH0	xxxx xxxx	uuuu uuuu
12h(3)	DDRJ	Data Direction Register for PORTJ								1111 1111	1111 1111
13h(3)	PORTJ(4)	RJ7	RJ6	RJ5	RJ4	RJ3	RJ2	RJ1	RJ0	xxxx xxxx	uuuu uuuu
14h(3)	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
15h(3)	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
16h(3)	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
17h(3)	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
Unbanked											
18h	PRODL	Low Byte of 16-bit Product (8 x 8 Hardware Multiply)								xxxx xxxx	uuuu uuuu
19h	PRODH	High Byte of 16-bit Product (8 x 8 Hardware Multiply)								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition.
Shaded cells are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for PC<15:8> whose contents are updated from, or transferred to, the upper byte of the program counter.

2: The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ status bits in CPUSTA are not affected by a $\overline{\text{MCLR}}$ Reset.

3: Bank 8 and associated registers are only implemented on the PIC17C76X devices.

4: This is the value that will be in the port output latch.

5: When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

6: On any device RESET, these pins are configured as inputs.

10.1 PORTA Register

PORTA is a 6-bit wide latch. PORTA does not have a corresponding Data Direction Register (DDR). Upon a device RESET, the PORTA pins are forced to be hi-impedance inputs. For the RA4 and RA5 pins, the peripheral module controls the output. When a device RESET occurs, the peripheral module is disabled, so these pins are forced to be hi-impedance inputs.

Reading PORTA reads the status of the pins.

The RA0 pin is multiplexed with the external interrupt, INT. The RA1 pin is multiplexed with TMR0 clock input, RA2 and RA3 are multiplexed with the SSP functions, and RA4 and RA5 are multiplexed with the USART1 functions. The control of RA2, RA3, RA4 and RA5 as outputs, is automatically configured by their multiplexed peripheral module when the module is enabled.

10.1.1 USING RA2, RA3 AS OUTPUTS

The RA2 and RA3 pins are open drain outputs. To use the RA2 and/or the RA3 pin(s) as output(s), simply write to the PORTA register the desired value. A '0' will cause the pin to drive low, while a '1' will cause the pin to float (hi-impedance). An external pull-up resistor should be used to pull the pin high. Writes to the RA2 and RA3 pins will not affect the other PORTA pins.

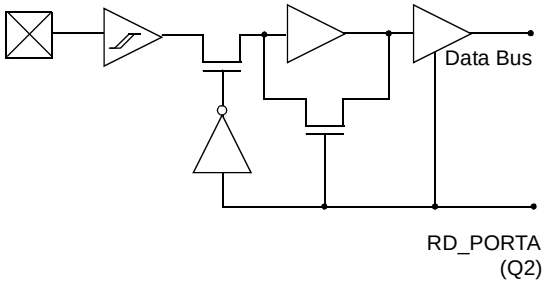
Note:	When using the RA2 or RA3 pin(s) as output(s), read-modify-write instructions (such as `BCF`, `BSF`, `BTG`) on PORTA are not recommended. Such operations read the port pins, do the desired operation, and then write this value to the data latch. This may inadvertently cause the RA2 or RA3 pins to switch from input to output (or vice-versa). To avoid this possibility, use a shadow register for PORTA. Do the bit operations on this shadow register and then move it to PORTA.

Example 10-1 shows an instruction sequence to initialize PORTA. The Bank Select Register (BSR) must be selected to Bank 0 for the port to be initialized. The following example uses the `MOVLB` instruction to load the BSR register for bank selection.

EXAMPLE 10-1: INITIALIZING PORTA

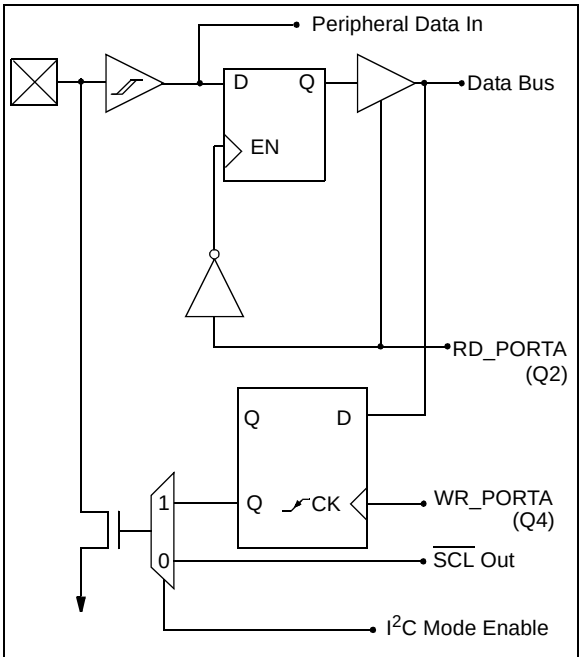
```
    MOVLB   0       ; Select Bank 0
    MOVLW   0xF3    ;
    MOVWF   PORTA   ; Initialize PORTA
                    ;   RA<3:2> are output low
                    ;   RA<5:4> and RA<1:0>
                    ;   are inputs
                    ;   (outputs floating)
```

FIGURE 10-1: RA0 AND RA1 BLOCK DIAGRAM



Note: Input pins have protection diodes to VDD and VSS.

FIGURE 10-2: RA2 BLOCK DIAGRAM



Note: I/O pin has protection diodes to VSS.

13.1.2 TIMER1 AND TIMER2 IN 16-BIT MODE

To select 16-bit mode, set the T16 bit. In this mode, TMR2 and TMR1 are concatenated to form a 16-bit timer (TMR2:TMR1). The 16-bit timer increments until it matches the 16-bit period register (PR2:PR1). On the following timer clock, the timer value is reset to 0h, and the TMR1IF bit is set.

When selecting the clock source for the 16-bit timer, the TMR1CS bit controls the entire 16-bit timer and TMR2CS is a “don’t care”, however, ensure that TMR2ON is set (allows TMR2 to increment). When TMR1CS is clear, the timer increments once every instruction cycle (FOSC/4). When TMR1CS is set, the timer increments on every falling edge of the RB4/TCLK12 pin. For the 16-bit timer to increment, both TMR1ON and TMR2ON bits must be set (Table 13-2).

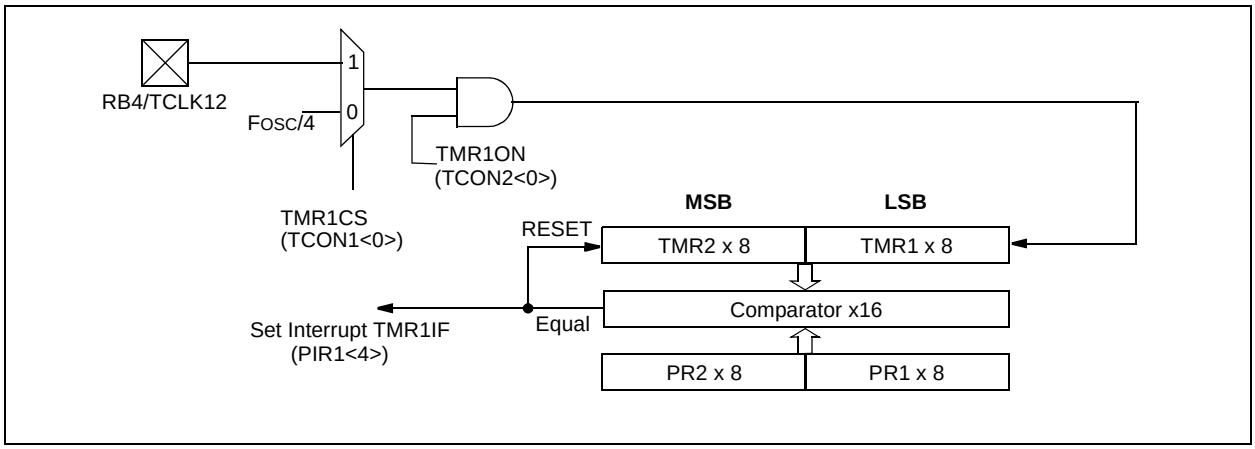
13.1.2.1 External Clock Input for TMR2:TMR1

When TMR1CS is set, the 16-bit TMR2:TMR1 increments on the falling edge of clock input TCLK12. The input on the RB4/TCLK12 pin is sampled and synchronized by the internal phase clocks twice every instruction cycle. This causes a delay from the time a falling edge appears on RB4/TCLK12 to the time TMR2:TMR1 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section.

TABLE 13-2: TURNING ON 16-BIT TIMER

T16	TMR2ON	TMR1ON	Result
1	1	1	16-bit timer (TMR2:TMR1) ON
1	0	1	Only TMR1 increments
1	x	0	16-bit timer OFF
0	1	1	Timers in 8-bit mode

FIGURE 13-2: TMR2 AND TMR1 IN 16-BIT TIMER/COUNTER MODE



14.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. Table 14-2 shows the formula for computation of the baud rate for different USART modes. These only apply when the USART is in Synchronous Master mode (internal clock) and Asynchronous mode.

Given the desired baud rate and Fosc, the nearest integer value between 0 and 255 can be calculated using the formula below. The error in baud rate can then be determined.

TABLE 14-2: BAUD RATE FORMULA

SYNC	Mode	Baud Rate
0	Asynchronous	FOSC/(64(X+1))
1	Synchronous	FOSC/(4(X+1))

X = value in SPBRG (0 to 255)

Example 14-1 shows the calculation of the baud rate error for the following conditions:

FOSC = 16 MHz
Desired Baud Rate = 9600
SYNC = 0

EXAMPLE 14-1: CALCULATING BAUD RATE ERROR

Desired Baud Rate = FOSC / (64 (X + 1))

9600 = 16000000 /(64 (X + 1))

X = 25.042 → 25

Calculated Baud Rate = 16000000 / (64 (25 + 1))

= 9615

$$\text{Error} = \frac{\text{(Calculated Baud Rate - Desired Baud Rate)}}{\text{Desired Baud Rate}}$$

= (9615 - 9600) / 9600

= 0.16%

Writing a new value to the SPBRG, causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

Effects of Reset

After any device RESET, the SPBRG register is cleared. The SPBRG register will need to be loaded with the desired value after each RESET.

TABLE 14-3: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

	Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{\text{MCLR}}$, WDT
USART1	13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
	15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
	17h, Bank 0	SPBRG1	Baud Rate Generator Register								0000 0000	0000 0000
USART2	13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
	15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
	17h, Bank 4	SPBRG2	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Baud Rate Generator.

TABLE 14-5: BAUD RATES FOR ASYNCHRONOUS MODE

BAUD RATE (K)	FOSC = 33 MHz			FOSC = 25 MHz			FOSC = 20 MHz			FOSC = 16 MHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	1.221	+1.73	255	1.202	+0.16	207
2.4	2.398	-0.07	214	2.396	0.14	162	2.404	+0.16	129	2.404	+0.16	103
9.6	9.548	-0.54	53	9.53	-0.76	40	9.469	-1.36	32	9.615	+0.16	25
19.2	19.09	-0.54	26	19.53	+1.73	19	19.53	+1.73	15	19.23	+0.16	12
76.8	73.66	-4.09	6	78.13	+1.73	4	78.13	+1.73	3	83.33	+8.51	2
96	103.12	+7.42	4	97.65	+1.73	3	104.2	+8.51	2	NA	—	—
300	257.81	-14.06	1	390.63	+30.21	0	312.5	+4.17	0	NA	—	—
500	515.62	+3.13	0	NA	—	—	NA	—	—	NA	—	—
HIGH	515.62	—	0	—	—	0	312.5	—	0	250	—	0
LOW	2.014	—	255	1.53	—	255	1.221	—	255	0.977	—	255

BAUD RATE (K)	FOSC = 10 MHz			FOSC = 7.159 MHz			FOSC = 5.068 MHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	NA	—	—	NA	—	—	0.31	+3.13	255
1.2	1.202	+0.16	129	1.203	_0.23	92	1.2	0	65
2.4	2.404	+0.16	64	2.380	-0.83	46	2.4	0	32
9.6	9.766	+1.73	15	9.322	-2.90	11	9.9	-3.13	7
19.2	19.53	+1.73	7	18.64	-2.90	5	19.8	+3.13	3
76.8	78.13	+1.73	1	NA	—	—	79.2	+3.13	0
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	156.3	—	0	111.9	—	0	79.2	—	0
LOW	0.610	—	255	0.437	—	255	0.309	—	255

BAUD RATE (K)	FOSC = 3.579 MHz			FOSC = 1 MHz			FOSC = 32.768 kHz		
	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)	KBAUD	%ERROR	SPBRG VALUE (DECIMAL)
0.3	0.301	+0.23	185	0.300	+0.16	51	0.256	-14.67	1
1.2	1.190	-0.83	46	1.202	+0.16	12	NA	—	—
2.4	2.432	+1.32	22	2.232	-6.99	6	NA	—	—
9.6	9.322	-2.90	5	NA	—	—	NA	—	—
19.2	18.64	-2.90	2	NA	—	—	NA	—	—
76.8	NA	—	—	NA	—	—	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—
HIGH	55.93	—	0	15.63	—	0	0.512	—	0
LOW	0.218	—	255	0.061	—	255	0.002	—	255

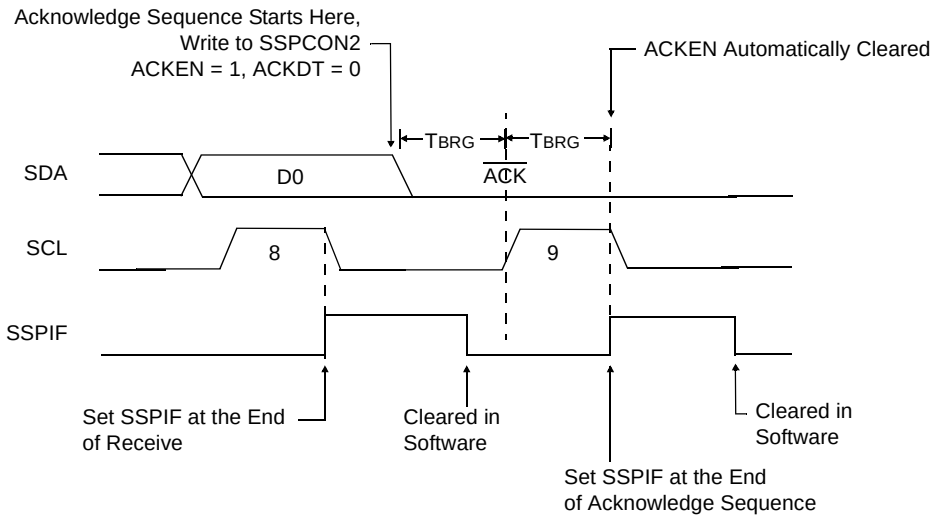
15.2.13 ACKNOWLEDGE SEQUENCE TIMING

An acknowledge sequence is enabled by setting the acknowledge sequence enable bit, ACKEN (SSPCON2<4>). When this bit is set, the SCL pin is pulled low and the contents of the acknowledge data bit is presented on the SDA pin. If the user wishes to generate an acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an acknowledge sequence. The baud rate generator then counts for one rollover period (TBRG), and the SCL pin is de-asserted (pulled high). When the SCL pin is sampled high (clock arbitration), the baud rate generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the baud rate generator is turned off and the SSP module then goes into IDLE mode (Figure 15-29).

15.2.13.1 WCOL Status Flag

If the user writes the SSPBUF when an acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 15-29: ACKNOWLEDGE SEQUENCE WAVEFORM

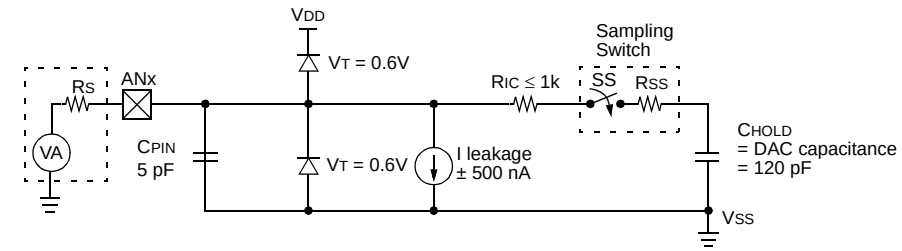


Note: TBRG = one baud rate generator period.

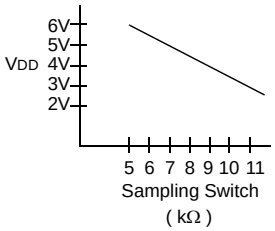
EXAMPLE 15-2: INTERFACING TO A 24LC01B SERIAL EEPROM (USING MPLAB C17)

```
// Writes the byte data to 24LC01B at the specified address
void ByteWrite(static unsigned char address, static unsigned char data)
{
    StartI2C();                      // Send start bit
    IdleI2C();                       // Wait for idle condition
    WriteI2C(CONTROL);               // Send control byte
    IdleI2C();                       // Wait for idle condition
    if (!SSPCON2bits.ACKSTAT)        // If 24LC01B ACKs
    {
        WriteI2C(address);           // Send control byte
        IdleI2C();                   // Wait for idle condition

        if (!SSPCON2bits.ACKSTAT)    // If 24LC01B ACKs
            WriteI2C(data);          // Send data
    }
    IdleI2C();                       // Wait for idle condition
    StopI2C();                       // Send stop bit
    IdleI2C();                       // Wait for idle condition
    return;
}


// Reads a byte of data from 24LC01B at the specified address
unsigned char ByteRead(static unsigned char address)
{
    StartI2C();                      // Send start bit
    IdleI2C();                       // Wait for idle condition
    WriteI2C(CONTROL);               // Send control byte
    IdleI2C();                       // Wait for idle condition
    if (!SSPCON2bits.ACKSTAT)        // If the 24LC01B ACKs
    {
        WriteI2C(address);           // Send address
        IdleI2C();                   // Wait for idle condition
        if (!SSPCON2bits.ACKSTAT)    // If the 24LC01B ACKs
        {
            RestartI2C();            // Send restart
            IdleI2C();               // Wait for idle condition
            WriteI2C(CONTROL+1);     // Send control byte with R/W set
            IdleI2C();               // Wait for idle condition
            if (!SSPCON2bits.ACKSTAT)    // If the 24LC01B ACKs
            {
                getcI2C();           // Read a byte of data from 24LC01B
                IdleI2C();           // Wait for idle condition
                NotAckI2C();         // Send a NACK to 24LC01B
                IdleI2C();           // Wait for idle condition
                StopI2C();           // Send stop bit
                IdleI2C();           // Wait for idle condition
             }
        }
    }
    return(SSPBUF);
}
```

FIGURE 16-3: ANALOG INPUT MODEL



Legend		
	C_{PIN}	= input capacitance
	V_T	= threshold voltage
	I leakage	= leakage current at the pin due to various junctions
	R_{IC}	= interconnect resistance
	SS	= sampling switch
	C_{HOLD}	= sample/hold capacitance (from DAC)



17.4.2 MINIMIZING CURRENT CONSUMPTION

To minimize current consumption, all I/O pins should be either at VDD, or VSS, with no external circuitry drawing current from the I/O pin. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should be at VDD or VSS. The contributions from on-chip pull-ups on PORTB should also be considered and disabled, when possible.

17.5 Code Protection

The code in the program memory can be protected by selecting the microcontroller in Code Protected mode (PM2:PM0 = '000').

In this mode, instructions that are in the on-chip program memory space, can continue to read or write the program memory. An instruction that is executed outside of the internal program memory range will be inhibited from writing to, or reading from, program memory.

Note:	Microchip does not recommend code protecting windowed devices.

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

18.0 INSTRUCTION SET SUMMARY

The PIC17CXXX instruction set consists of 58 instructions. Each instruction is a 16-bit word divided into an OPCODE and one or more operands. The opcode specifies the instruction type, while the operand(s) further specify the operation of the instruction. The PIC17CXXX instruction set can be grouped into three types:

- byte-oriented
- bit-oriented
- literal and control operations

These formats are shown in Figure 18-1.

Table 18-1 shows the field descriptions for the opcodes. These descriptions are useful for understanding the opcodes in Table 18-2 and in each specific instruction descriptions.

For **byte-oriented instructions**, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' = '0', the result is placed in the WREG register. If 'd' = '1', the result is placed in the file register specified by the instruction.

For **bit-oriented instructions**, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

For **literal and control operations**, 'k' represents an 8- or 13-bit constant or literal value.

The instruction set is highly orthogonal and is grouped into:

- byte-oriented operations
- bit-oriented operations
- literal and control operations

All instructions are executed within one single instruction cycle, unless:

- a conditional test is true
- the program counter is changed as a result of an instruction
- a table read or a table write instruction is executed (in this case, the execution takes two instruction cycles with the second cycle executed as a NOP)

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 25 MHz, the normal instruction execution time is 160 ns. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 320 ns.

TABLE 18-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (00h to FFh)
p	Peripheral register file address (00h to 1Fh)
i	Table pointer control i = '0' (do not change) i = '1' (increment after instruction execution)
t	Table byte select t = '0' (perform operation on lower byte) t = '1' (perform operation on upper byte literal field, constant data)
WREG	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= '0' or '1') The assembler will generate code with x = '0'. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select 0 = store result in WREG 1 = store result in file register f Default is d = '1'
u	Unused, encoded as '0'
s	Destination select 0 = store result in file register f and in the WREG 1 = store result in file register f Default is s = '1'
label	Label name
C,DC, Z,OV	ALU status bits Carry, Digit Carry, Zero, Overflow
GLINTD	Global Interrupt Disable bit (CPUSTA<4>)
TBLPTR	Table Pointer (16-bit)
TBLAT	Table Latch (16-bit) consists of high byte (TBLATH) and low byte (TBLATL)
TBLATL	Table Latch low byte
TBLATH	Table Latch high byte
TOS	Top-of-Stack
PC	Program Counter
BSR	Bank Select Register
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
dest	Destination either the WREG register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

RRNCF **Rotate Right f (no carry)**

Syntax: [*label*] RRNCF f,d

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

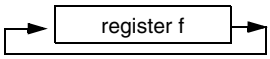
Operation: f<n> → d<n-1>;
f<0> → d<7>

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'	Process Data	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: $0 \le f \le 255$
$s \in [0,1]$

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'	Process Data	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

SLEEP **Enter SLEEP mode**

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT;
0 → WDT postscaler;
1 → $\overline{TO}$;
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0011

Description: The power-down status bit ($\overline{PD}$) is cleared. The time-out status bit ($\overline{TO}$) is set. Watchdog Timer and its post-scaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to sleep

Example: `SLEEP`

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?

After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared

SUBLW **Subtract WREG from Literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k – (WREG) → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

1011	0010	kkkk	kkkk

Description: WREG is subtracted from 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'	Process Data	Write to WREG

Example 1: `SUBLW 0x02`

Before Instruction
WREG = 1
C = ?

After Instruction
WREG = 1
C = 1 ; result is positive
Z = 0

Example 2:

Before Instruction
WREG = 2
C = ?

After Instruction
WREG = 0
C = 1 ; result is zero
Z = 1

Example 3:

Before Instruction
WREG = 3
C = ?

After Instruction
WREG = FF ; (2's complement)
C = 0 ; result is negative
Z = 0

TABLRD **Table Read**

Example1: `TABLRD 1, 1, REG ;`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

After Instruction (table write completion)

REG	=	0xAA
TBLATH	=	0x12
TBLATL	=	0x34
TBLPTR	=	0xA357
MEMORY(TBLPTR)	=	0x5678

Example2: `TABLRD 0, 0, REG ;`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

After Instruction (table write completion)

REG	=	0x55
TBLATH	=	0x12
TBLATL	=	0x34
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

TABLWT **Table Write**

Syntax: [*label*] TABLWT t,i,f

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

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH;
TBLAT → Prog Mem (TBLPTR);
If i = 1,
TBLPTR + 1 → TBLPTR
If i = 0,
TBLPTR is unchanged

Status Affected: None

Encoding:

1010	11ti	ffff	ffff

Description:

1. Load value in 'f' into 16-bit table latch (TBLAT)
   If t = 1: load into high byte;
   If t = 0: load into low byte
2. The contents of TBLAT are written to the program memory location pointed to by TBLPTR.
   If TBLPTR points to external program memory location, then the instruction takes two-cycle.
   If TBLPTR points to an internal EPROM location, then the instruction is terminated when an interrupt is received.

Note: The $\overline{MCLR}/V_{PP}$ pin must be at the programming voltage for successful programming of internal memory.
If $\overline{MCLR}/V_{PP} = V_{DD}$
the programming sequence of internal memory will be interrupted. A short write will occur (2 T_{CY}). The internal memory location will not be affected.

3. The TBLPTR can be automatically incremented
   If i = 1; TBLPTR is not incremented
   If i = 0; TBLPTR is incremented

Words: 1

Cycles: 2 (many if write is to on-chip EPROM program memory)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register TBLATH or TBLATL
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, $\overline{WR}$ goes low)

19.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

19.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or ‘C’)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

19.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PIC MCU’s.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

19.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI ‘C’ compilers for Microchip’s PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

20.1 DC Characteristics

PIC17LC7XX-08 (Commercial, Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial				
PIC17C7XX-16 (Commercial, Industrial, Extended) **PIC17C7XX-33** (Commercial, Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +125°C for extended -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial				
Param. No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D001	VDD	**Supply Voltage**					
		PIC17LC7XX	3.0	—	5.5	V	
D001		PIC17C7XX-33 PIC17C7XX-16	4.5 VBOR	— —	5.5 5.5	V V	(BOR enabled) **(Note 5)**
D002	VDR	**RAM Data Retention Voltage (Note 1)**	1.5	—	—	V	Device in SLEEP mode
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	Vss	—	V	See section on Power-on Reset for details
D004	SVDD	**VDD Rise Rate** to ensure proper operation					
		PIC17LCXX	0.010	—	—	V/ms	See section on Power-on Reset for details
D004		PIC17CXX	0.085	—	—	V/ms	See section on Power-on Reset for details
D005	VBOR	**Brown-out Reset** voltage trip point	3.65	—	4.35	V	
D006	VPORTP	**Power-on Reset** trip point	—	2.2	—	V	VDD = VPORTP

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD or VSS, T0CKI = VDD, MCLR = VDD; WDT disabled.

Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.

For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as: VDD/(2 • R).

For capacitive loads, the current can be estimated (for an individual I/O pin) as (CL • VDD) • f

CL = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.

The capacitive currents are most significant when the device is configured for external execution (includes Extended Microcontroller mode).

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula IR = VDD/2REXT (mA) with REXT in kOhm.

5: This is the voltage where the device enters the Brown-out Reset. When BOR is enabled, the device (-16) will operate correctly to this trip point.

PIC17LC7XX-08 (Commercial, Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial				
PIC17C7XX-16 (Commercial, Industrial, Extended) **PIC17C7XX-33** (Commercial, Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +125°C for extended -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial				
Param. No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D010	IDD	**Supply Current (Note 2)**					
		PIC17LC7XX	—	3	6	mA	FOSC = 4 MHz **(Note 4)**
D010		PIC17C7XX	—	3	6	mA	FOSC = 4 MHz **(Note 4)**
D011		PIC17LC7XX	—	5	10	mA	FOSC = 8 MHz
D011		PIC17C7XX	—	5	10	mA	FOSC = 8 MHz
D012			—	9	18	mA	FOSC = 16 MHz
D014		PIC17LC7XX	—	85	150	µA	FOSC = 32 kHz, (EC osc configuration)
D015		PIC17C7XX	—	15	30	mA	FOSC = 33 MHz
D021	IPD	**Power-down Current (Note 3)**					
		PIC17LC7XX	—	<1	5	µA	VDD = 3.0V, WDT disabled
D021 (commercial, industrial)		PIC17C7XX	—	<1	20	µA	VDD = 5.5V, WDT disabled
D021A (extended)			—	2	20	µA	VDD = 5.5V, WDT disabled
		Module Differential Current					
D023	ΔIBOR	BOR circuitry	–	75	150	µA	VDD = 4.5V, BODEN enabled
D024	ΔIWDT	Watchdog Timer	–	10	35	µA	VDD = 5.5V
D026	ΔIAD	A/D converter	–	1	–	µA	VDD = 5.5V, A/D not converting

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD or VSS, T0CKI = VDD, MCLR = VDD; WDT disabled.

Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.

For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as: VDD/(2 • R).

For capacitive loads, the current can be estimated (for an individual I/O pin) as (CL • VDD) • f

CL = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.

The capacitive currents are most significant when the device is configured for external execution (includes Extended Microcontroller mode).

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula IR = VDD/2REXT (mA) with REXT in kOhm.

5: This is the voltage where the device enters the Brown-out Reset. When BOR is enabled, the device (-16) will operate correctly to this trip point.

INDEX