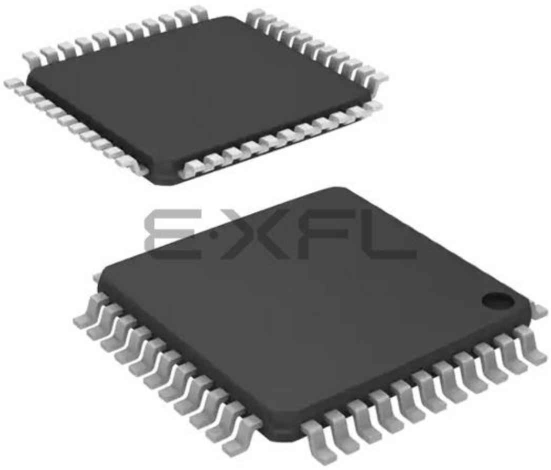


Welcome to E-XFL.COM

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	Active
Core Processor	PIC
Core Size	8-Bit
Speed	33MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	8KB (4K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	454 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c43-33i-pt

9.3 PORTC and DDRC Registers

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

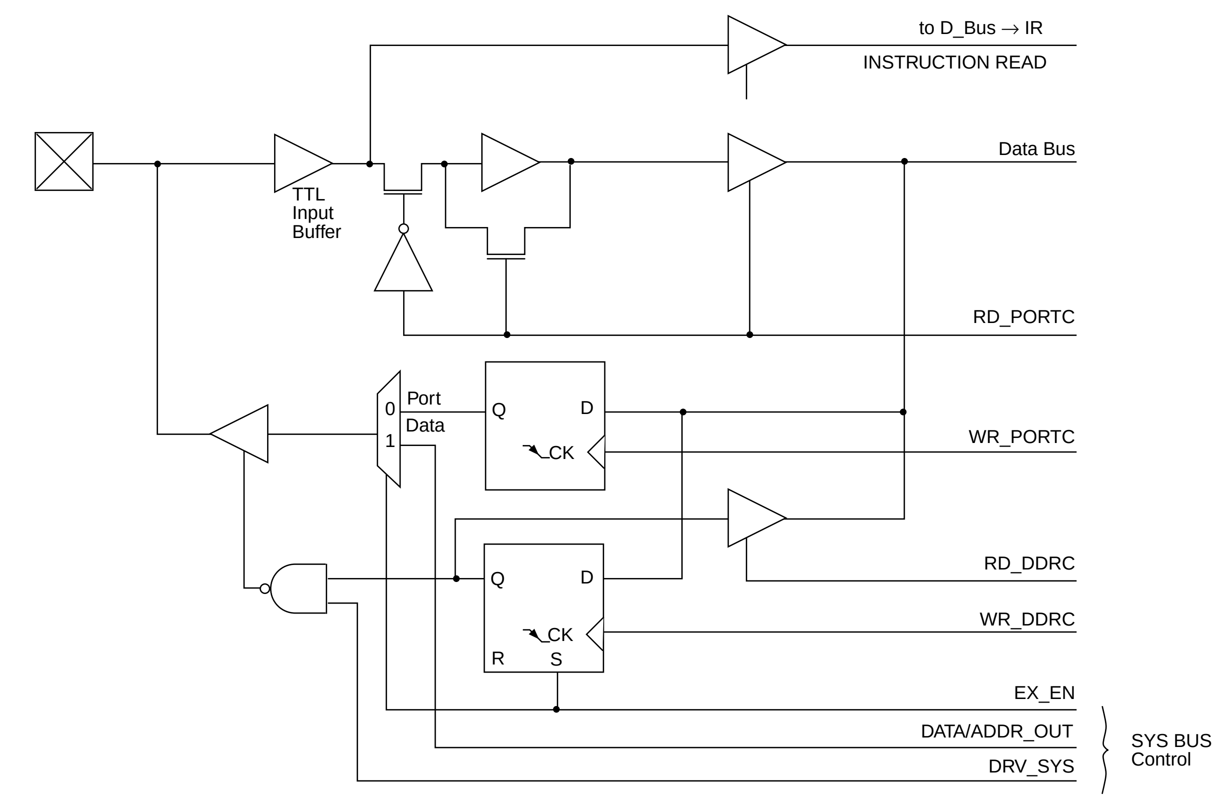
Note:	This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

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

EXAMPLE 9-2: INITIALIZING PORTC

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

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



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

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.

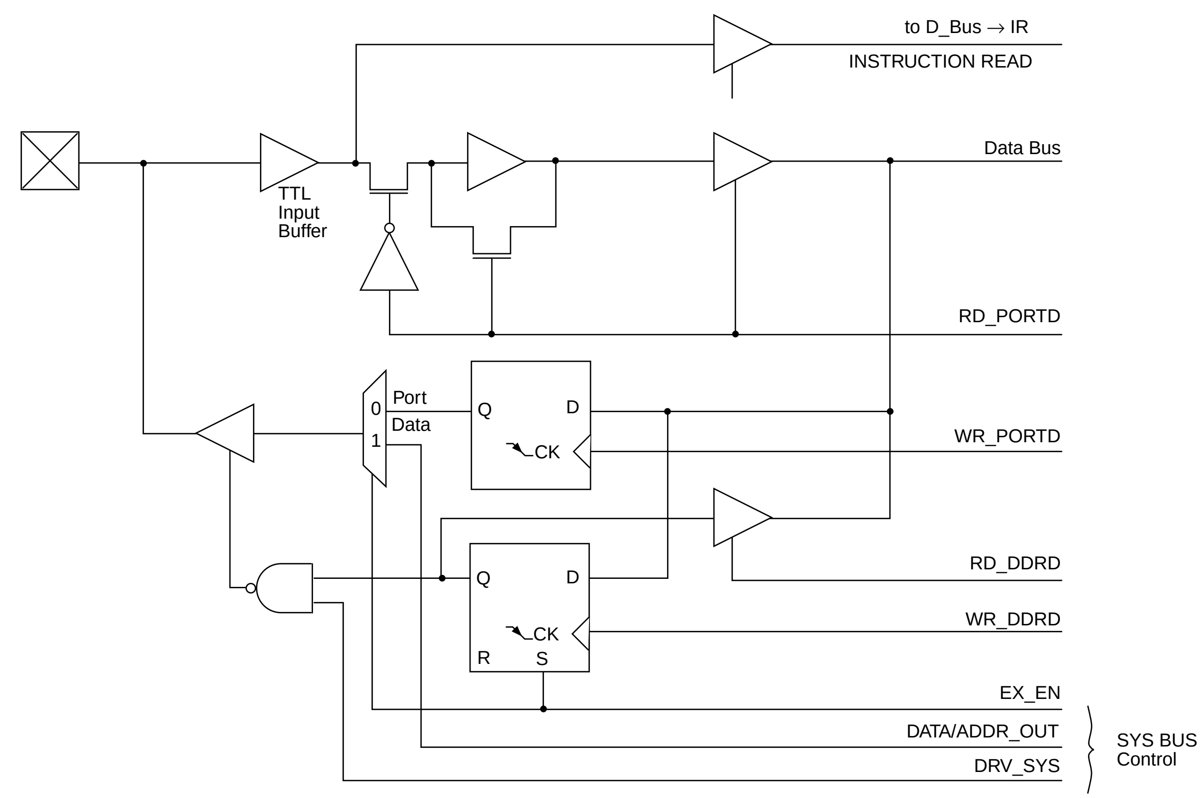
Note:	This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

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.

11.3 Read/Write Consideration for TMR0

Although TMR0 is a 16-bit timer/counter, only 8-bits at a time can be read or written during a single instruction cycle. Care must be taken during any read or write.

11.3.1 READING 16-BIT VALUE

The problem in reading the entire 16-bit value is that after reading the low (or high) byte, its value may change from FFh to 00h.

Example 11-1 shows a 16-bit read. To ensure a proper read, interrupts must be disabled during this routine.

EXAMPLE 11-1: 16-BIT READ

```
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
MOVFP   TMPLO, WREG     ;tmplo -> wreg
CPFSLT  TMR0L           ;tmr0l < wreg?
RETURN                  ;no then return
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
RETURN                  ;return
```

11.3.2 WRITING A 16-BIT VALUE TO TMR0

Since writing to either TMR0L or TMR0H will effectively inhibit increment of that half of the TMR0 in the next cycle (following write), but not inhibit increment of the other half, the user must write to TMR0L first and TMR0H next in two consecutive instructions, as shown in Example 11-2. The interrupt must be disabled. Any write to either TMR0L or TMR0H clears the prescaler.

EXAMPLE 11-2: 16-BIT WRITE

```
BSF   CPUSTA, GLINTD ; Disable interrupt
MOVFP RAM_L, TMR0L   ;
MOVFP RAM_H, TMR0H   ;
BCF   CPUSTA, GLINTD ; Done, enable interrupt
```

11.4 Prescaler Assignments

Timer0 has an 8-bit prescaler. The prescaler assignment is fully under software control; i.e., it can be changed “on the fly” during program execution. When changing the prescaler assignment, clearing the prescaler is recommended before changing assignment. The value of the prescaler is “unknown,” and assigning a value that is less then the present value makes it difficult to take this unknown time into account.

FIGURE 11-4: TMR0 TIMING: WRITE HIGH OR LOW BYTE

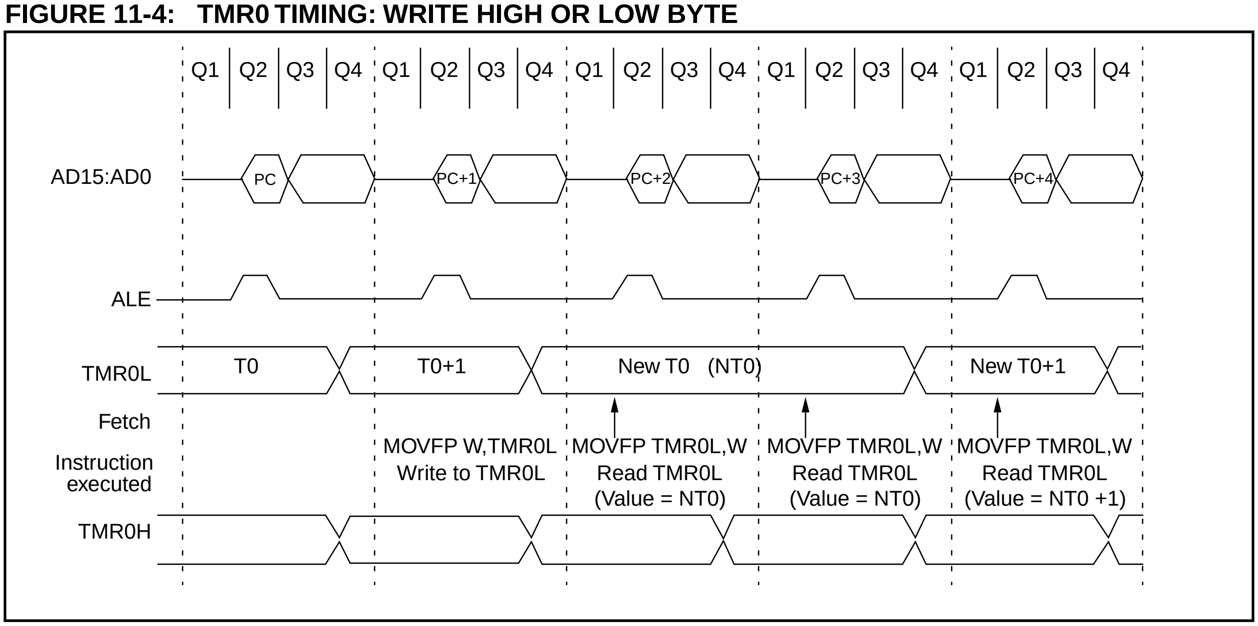


FIGURE 13-2: RCSTA REGISTER (ADDRESS: 13h, BANK 0)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0	R - 0	R - 0	R - x
SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D
bit7							bit 0

R = Readable bit
W = Writable bit
-n = Value at POR reset
(x = unknown)

bit 7: **SPEN**: Serial Port Enable bit
1 = Configures RA5/RX/DT and RA4/TX/CK pins as serial port pins
0 = Serial port disabled

bit 6: **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5: **SREN**: Single Receive Enable bit
This bit enables the reception of a single byte. After receiving the byte, this bit is automatically cleared.
Synchronous mode:
1 = Enable reception
0 = Disable reception
Note: This bit is ignored in synchronous slave reception.
Asynchronous mode:
Don’t care

bit 4: **CREN**: Continuous Receive Enable bit
This bit enables the continuous reception of serial data.
Asynchronous mode:
1 = Enable reception
0 = Disables reception
Synchronous mode:
1 = Enables continuous reception until CREN is cleared (CREN overrides SREN)
0 = Disables continuous reception

bit 3: **Unimplemented**: Read as '0'

bit 2: **FERR**: Framing Error bit
1 = Framing error (Updated by reading RCREG)
0 = No framing error

bit 1: **OERR**: Overrun Error bit
1 = Overrun (Cleared by clearing CREN)
0 = No overrun error

bit 0: **RX9D**: 9th bit of receive data (can be the software calculated parity bit)

TABLE 13-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

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
16h, Bank 0	TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	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 transmission.

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

FIGURE 13-9: SYNCHRONOUS TRANSMISSION

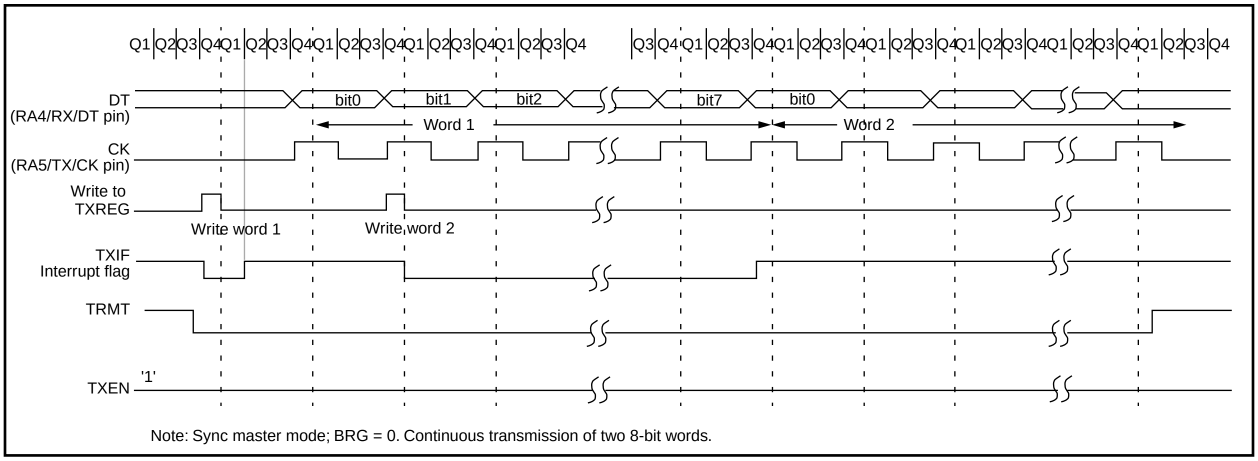
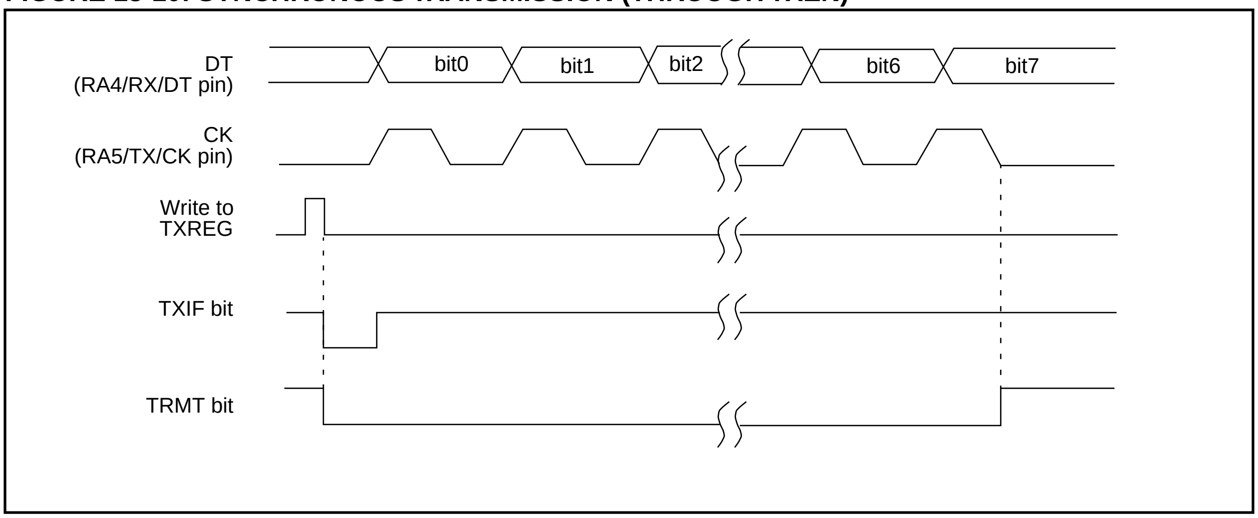


FIGURE 13-10: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



13.3.2 USART SYNCHRONOUS MASTER RECEPTION

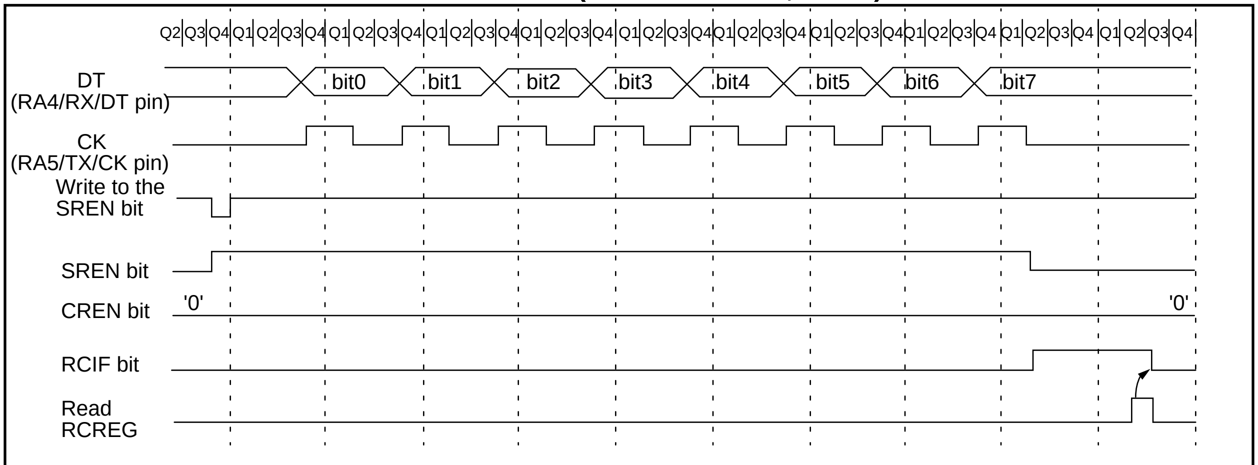
Once synchronous mode is selected, reception is enabled by setting either the SREN (RCSTA<5>) bit or the CREN (RCSTA<4>) bit. Data is sampled on the RA4/RX/DT pin on the falling edge of the clock. If SREN is set, then only a single word is received. If CREN is set, the reception is continuous until CREN is reset. If both bits are set, then CREN takes precedence. After clocking the last bit, the received data in the Receive Shift Register (RSR) is transferred to RCREG (if it is empty). If the transfer is complete, the interrupt bit RCIF (PIR<0>) is set. The actual interrupt can be enabled/disabled by setting/clearing the RCIE (PIE<0>) bit. RCIF is a read only bit which is RESET by the hardware. In this case it is reset when RCREG has been read and is empty. RCREG is a double buffered register; i.e., it is a two deep FIFO. It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting into the RSR. On the clocking of the last bit of the third byte, if RCREG is still full, then the overrun error bit OERR (RCSTA<1>) is set. The word in the RSR will be lost. RCREG can be read twice to retrieve the two bytes in the FIFO. The OERR bit has to be cleared in software. This is done by clearing the CREN bit. If OERR bit is set, transfers from RSR to RCREG are inhibited, so it is essential to clear OERR bit if it is set. The 9th receive bit is buffered the same way as the receive data. Reading the RCREG register will allow the RX9D and FERR bits to be loaded with values for the next received data; therefore, it is essential for the user to read the RCSTA register before reading RCREG in order not to lose the old FERR and RX9D information.

Steps to follow when setting up a Synchronous Master Reception:

1. Initialize the SPBRG register for the appropriate baud rate. See Section 13.1 for details.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
3. If interrupts are desired, then set the RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. If a single reception is required, set bit SREN. For continuous reception set bit CREN.
6. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading RCREG.
9. If any error occurred, clear the error by clearing CREN.

Note:	To terminate a reception, either clear the SREN and CREN bits, or the SPEN bit. This will reset the receive logic, so that it will be in the proper state when receive is re-enabled.

FIGURE 13-11: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates SYNC master mode with SREN = 1.

TABLE 13-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

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
16h, Bank 0	TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	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 slave transmission.

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

TABLE 13-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE 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, Bank1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
13h, Bank0	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank0	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
17h, Bank1	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, Bank0	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 slave reception.

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

BSF **Bit Set f**

Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Encoding:	`1000` `0bbb` `ffff` `ffff`
Description:	Bit 'b' in register 'f' is set.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `BSF FLAG_REG, 7`

Before Instruction
FLAG_REG= 0x0A

After Instruction
FLAG_REG= 0x8A

BTFSC **Bit Test, skip if Clear**

Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Encoding:	`1001` `1bbb` `ffff` `ffff`
Description:	If bit 'b' in register 'f' is 0 then the next instruction is skipped. If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and a `NOP` is executed instead, 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    BTFSC   FLAG,1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (`HERE`)

After Instruction
If FLAG<1> = 0;
PC = address (`TRUE`)
If FLAG<1> = 1;
PC = address (`FALSE`)

DECF — Decrement f

Syntax: [*label*] DECF f,d

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

Operation: $(f) - 1 \rightarrow$ (dest)

Status Affected: OV, C, DC, Z

Encoding:

`0000`	`011d`	`ffff`	`ffff`

Description: Decrement 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: `DECF CNT, 1`

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

DECFSZ — Decrement f, skip if 0

Syntax: [*label*] DECFSZ f,d

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

Operation: $(f) - 1 \rightarrow$ (dest);
skip if result = 0

Status Affected: None

Encoding:

`0001`	`011d`	`ffff`	`ffff`

Description: The contents of register 'f' are decremented. 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 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

Example:

```
HERE      DECFSZ   CNT, 1
          GOTO     LOOP
CONTINUE
```

Before Instruction
PC = Address (`HERE`)

After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address (`CONTINUE`)
If CNT ≠ 0;
PC = Address (`HERE+1`)

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 $\rightarrow$ (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) $\rightarrow$ (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

RLNCF — Rotate Left f (no carry)

Syntax:	[*label*] RLNCF f,d
Operands:	0 ≤ f ≤ 255 d ∈ [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 ≤ f ≤ 255 d ∈ [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
```

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

Status Affected: None

Encoding:

1010	11ti	ffff	ffff

Description:

1. Load value in 'f' into 16-bit table latch (TBLAT)
   If t = 0: load into low byte;
   If t = 1: load into high byte
2. The contents of TBLAT is 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 $\overline{MCLR}$/VPP pin must be at the programming voltage for successful programming of internal memory.
If $\overline{MCLR}$/VPP = VDD
the programming sequence of internal memory will be executed, but will not be successful (although the internal memory location may be disturbed)

3. The TBLPTR can be automatically incremented
   If i = 0; TBLPTR is not incremented
   If i = 1; 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'	Execute	Write register TBLATH or TBLATL

Applicable Devices	42	R42	42A	43	R43	44

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

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

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, AND POWER-UP TIMER TIMING

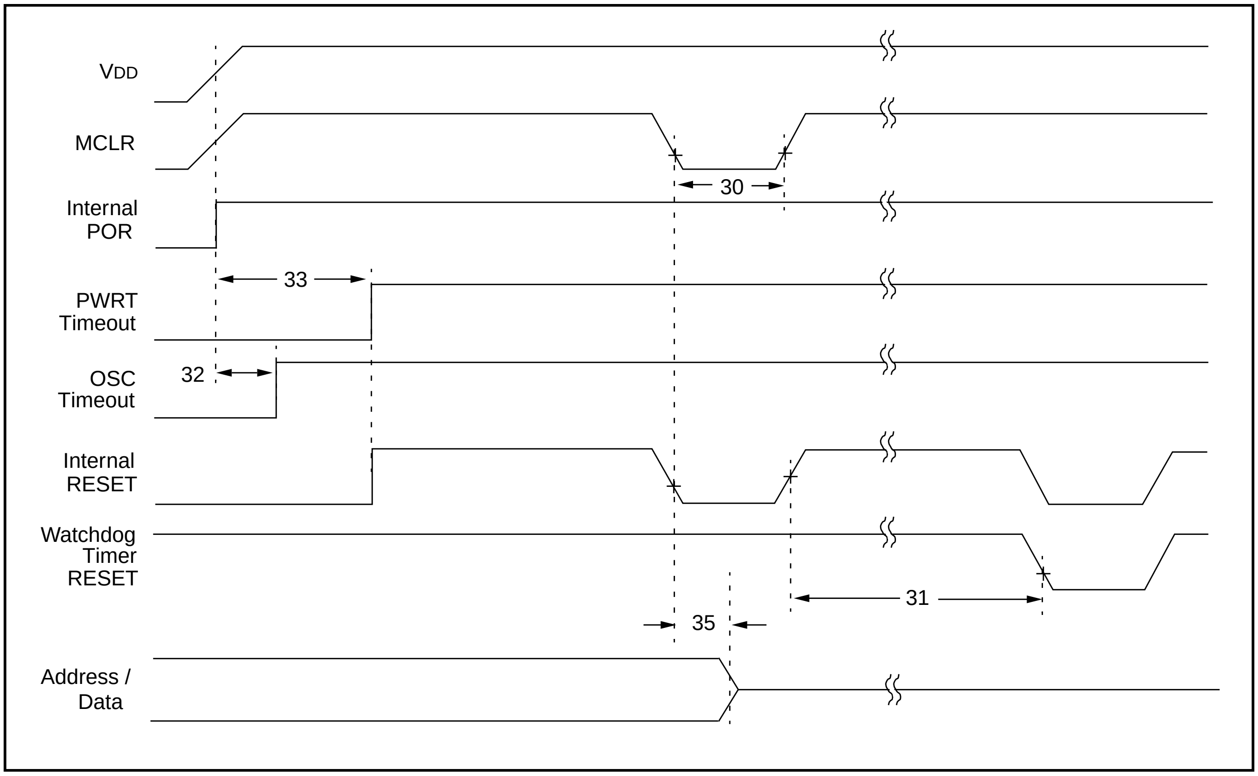


TABLE 19-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER REQUIREMENTS

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)		100 *	—	—	ns	VDD = 5V
31	Twdt	Watchdog Timer Time-out Period (Prescale = 1)		5 *	12	25 *	ms	VDD = 5V
32	Tost	Oscillation Start-up Timer Period		—	1024TOSC§	—	ms	TOSC = OSC1 period
33	Tpwrt	Power-up Timer Period		40 *	96	200 *	ms	VDD = 5V
35	TmcL2adl	$\overline{\text{MCLR}}$ to System Interface bus (AD15:AD0>) invalid	PIC17CR42/42A/43/R43/44	—	—	100 *	ns	
			PIC17LCR42/42A/43/R43/44	—	—	120 *	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.

‡ These parameters are for design guidance only and are not tested, nor characterized.

§ This specification ensured by design.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-12: MEMORY INTERFACE READ TIMING (NOT SUPPORTED IN PIC17LC4X DEVICES)

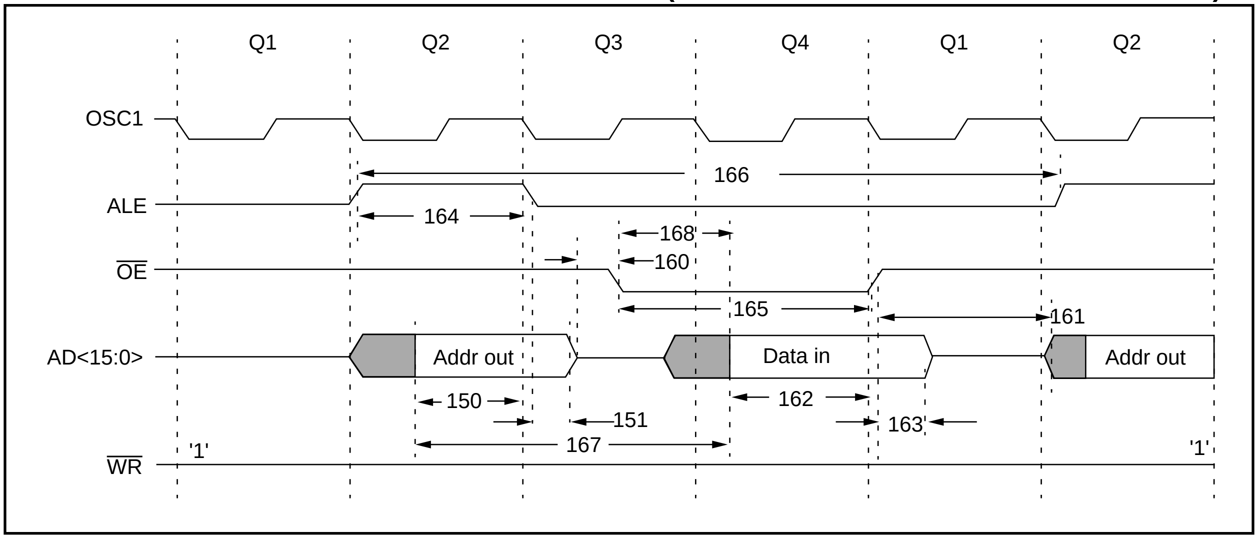


TABLE 19-12: MEMORY INTERFACE READ REQUIREMENTS (NOT SUPPORTED IN PIC17LC4X DEVICES)

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD15:AD0 (address) valid to ALE↓ (address setup time)	0.25Tcy - 10	—	—	ns	
151	TalL2adI	ALE↓ to address out invalid (address hold time)	5*	—	—	ns	
160	TadZ2oeL	AD15:AD0 hi-impedance to $\overline{OE}$↓	0*	—	—	ns	
161	ToeH2adD	$\overline{OE}$↑ to AD15:AD0 driven	0.25Tcy - 15	—	—	ns	
162	TadV2oeH	Data in valid before $\overline{OE}$↑ (data setup time)	35	—	—	ns	
163	ToeH2adI	$\overline{OE}$↑to data in invalid (data hold time)	0	—	—	ns	
164	TalH	ALE pulse width	—	0.25TCY §	—	ns	
165	ToeL	$\overline{OE}$ pulse width	0.5Tcy - 35 §	—	—	ns	
166	TalH2alH	ALE↑ to ALE↑(cycle time)	—	TCY §	—	ns	
167	Tacc	Address access time	—	—	0.75TCY - 30	ns	
168	Toe	Output enable access time ($\overline{OE}$ low to Data Valid)	—	—	0.5TCY - 45	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.

NOTES:

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-5: TRANSCONDUCTANCE (gm) OF LF OSCILLATOR vs. VDD

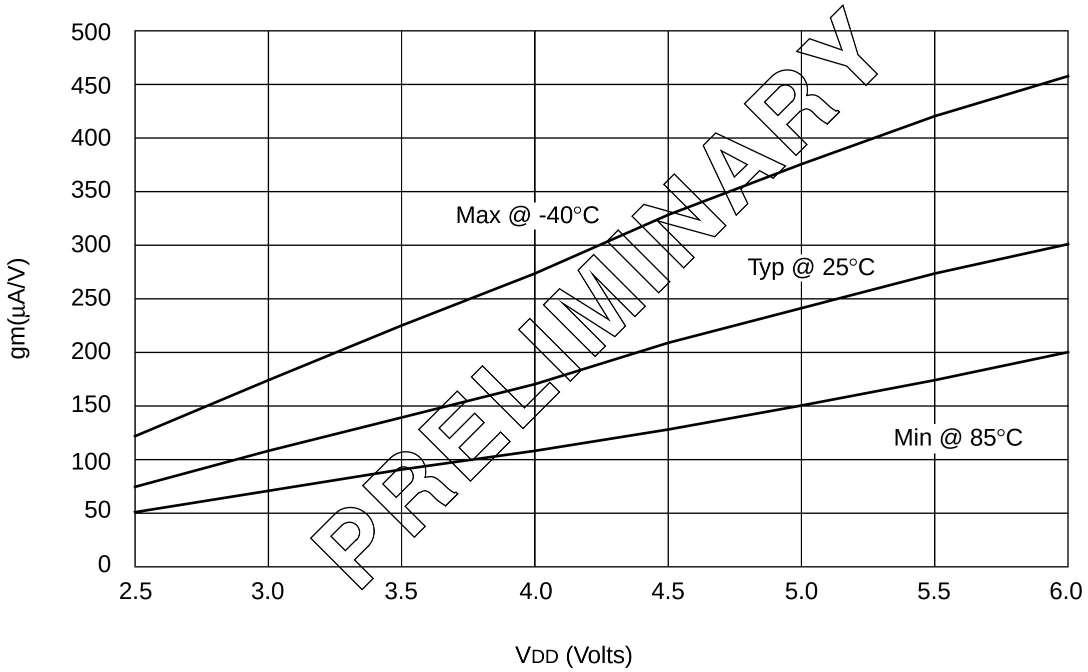
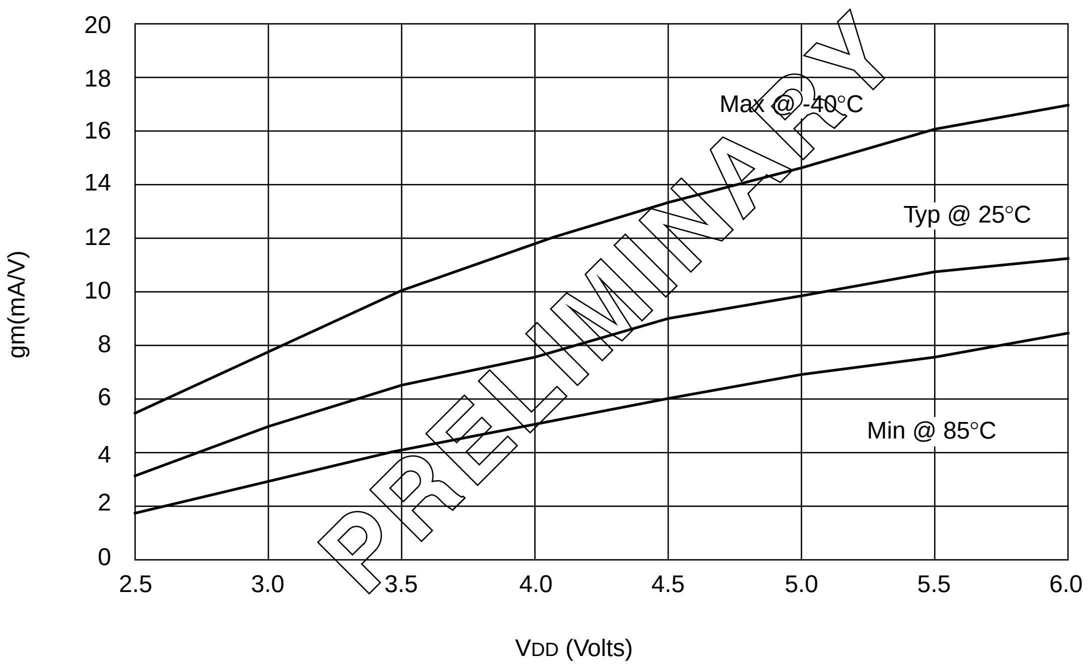
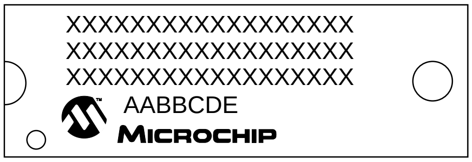


FIGURE 20-6: TRANSCONDUCTANCE (gm) OF XT OSCILLATOR vs. VDD

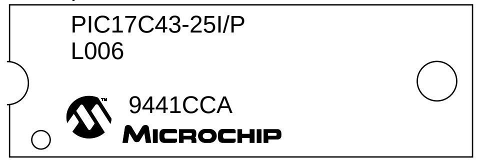


21.6 Package Marking Information

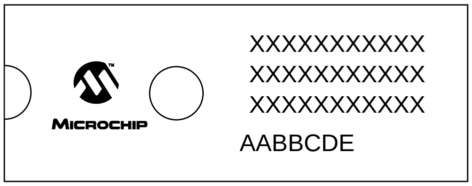
40-Lead PDIP/CERDIP



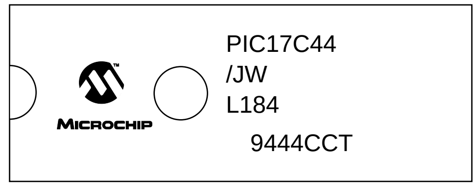
Example



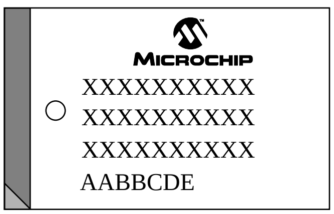
40 Lead CERDIP Windowed



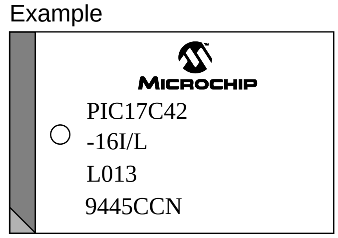
Example



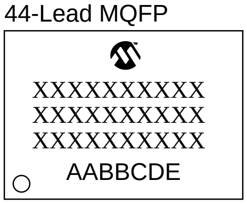
44-Lead PLCC



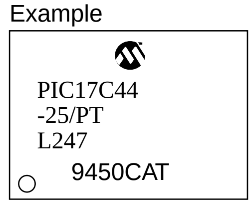
Example



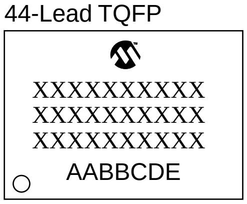
44-Lead MQFP



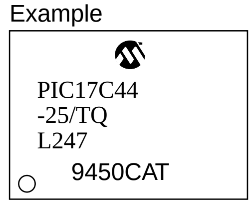
Example



44-Lead TQFP



Example



Legend:	MM...M	Microchip part number information
	XX...X	Customer specific information*
	AA	Year code (last 2 digits of calendar year)
	BB	Week code (week of January 1 is week '01')
	C	Facility code of the plant at which wafer is manufactured C = Chandler, Arizona, U.S.A., S = Tempe, Arizona, U.S.A.
	D	Mask revision number
	E	Assembly code of the plant or country of origin in which part was assembled

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

* Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

APPENDIX C: WHAT'S NEW

The structure of the document has been made consistent with other data sheets. This ensures that important topics are covered across all PIC16/17 families. Here is an overview of new features.

Added the following devices:

PIC17CR42

PIC17C42A

PIC17CR43

A 33 MHz option is now available.

APPENDIX D: WHAT'S CHANGED

To make software more portable across the different PIC16/17 families, the name of several registers and control bits have been changed. This allows control bits that have the same function, to have the same name (regardless of processor family). Care must still be taken, since they may not be at the same special function register address. The following shows the register and bit names that have been changed:

Old Name	New Name
TX8/9	TX9
RC8/9	RX9
RCD8	RX9D
TXD8	TX9D

Instruction DECFSNZ corrected to DCFSNZ

Instruction INCFSNZ corrected to INFSNZ

Enhanced discussion on PWM to include equation for determining bits of PWM resolution.

Section 13.2.2 and 13.3.2 have had the description of updating the FERR and RX9 bits enhanced.

The location of configuration bit PM2 was changed (Figure 6-1 and Figure 14-1).

Enhanced description of the operation of the INTSTA register.

Added note to discussion of interrupt operation.

Tightened electrical spec D110.

Corrected steps for setting up USART Asynchronous Reception.

L

M