Microchip Technology - PIC17C756AT-33I/L Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	33MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	50
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	902 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.23x24.23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c756at-33i-l

5.0 RESET

The PIC17CXXX differentiates between various kinds of RESET:

- Power-on Reset (POR)
- Brown-out Reset
- $\overline{\text{MCLR}}$ Reset
- WDT Reset

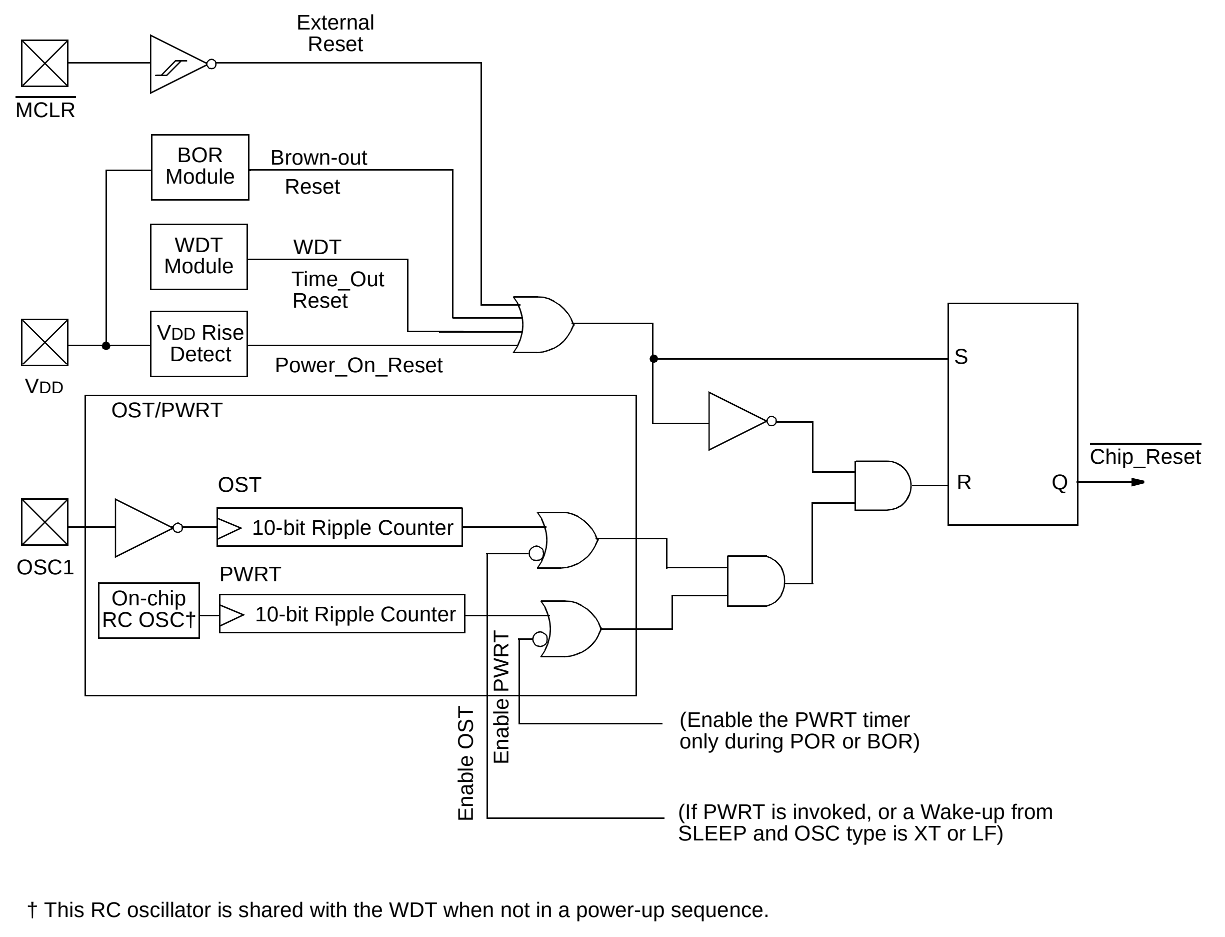
Some registers are not affected in any RESET condition, their status is unknown on POR and unchanged in any other RESET. Most other registers are forced to a “RESET state”. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are set or cleared differently in different RESET situations, as indicated in Table 5-3. These bits, in conjunction with the $\overline{\text{POR}}$ and $\overline{\text{BOR}}$ bits, are used in software to determine the nature of the RESET. See Table 5-4 for a full description of the RESET states of all registers.

When the device enters the “RESET state”, the Data Direction registers (DDR) are forced set, which will make the I/O hi-impedance inputs. The RESET state of some peripheral modules may force the I/O to other operations, such as analog inputs or the system bus.

Note: While the device is in a RESET state, the internal phase clock is held in the Q1 state. Any processor mode that allows external execution will force the RE0/ALE pin as a low output and the RE1/$\overline{\text{OE}}$ and RE2/$\overline{\text{WR}}$ pins as high outputs.

A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 5-1.

FIGURE 5-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



5.1.5 BROWN-OUT RESET (BOR)

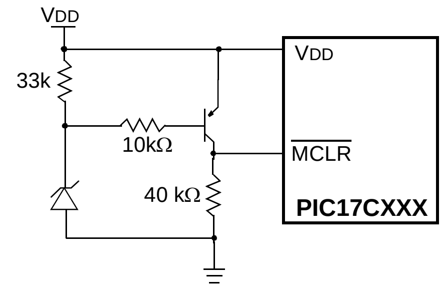
PIC17C7XX devices have on-chip Brown-out Reset circuitry. This circuitry places the device into a RESET when the device voltage falls below a trip point (BVDD). This ensures that the device does not continue program execution outside the valid operation range of the device. Brown-out Resets are typically used in AC line applications, or large battery applications, where large loads may be switched in (such as automotive).

Note:	Before using the on-chip Brown-out for a voltage supervisory function, please review the electrical specifications to ensure that they meet your requirements.

The BODEN configuration bit can disable (if clear/programmed), or enable (if set) the Brown-out Reset circuitry. If VDD falls below BVDD (typically 4.0 V, paramter #D005 in electrical specification section), for greater than parameter #35, the Brown-out situation will reset the chip. A RESET is not guaranteed to occur if VDD falls below BVDD for less than paramter #35. The chip will remain in Brown-out Reset until VDD rises above BVDD. The Power-up Timer and Oscillator Start-up Timer will then be invoked. This will keep the chip in RESET the greater of 96 ms and 1024 TOSC. If VDD drops below BVDD while the Power-up Timer/Oscillator Start-up Timer is running, the chip will go back into a Brown-out Reset. The Power-up Timer/Oscillator Start-up Timer will be initialized. Once VDD rises above BVDD, the Power-up Timer/Oscillator Start-up Timer will start their time delays. Figure 5-10 shows typical Brown-out situations.

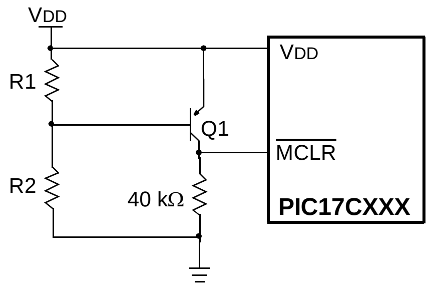
In some applications, the Brown-out Reset trip point of the device may not be at the desired level. Figure 5-8 and Figure 5-9 are two examples of external circuitry that may be implemented. Each needs to be evaluated to determine if they match the requirements of the application.

FIGURE 5-8: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 1



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

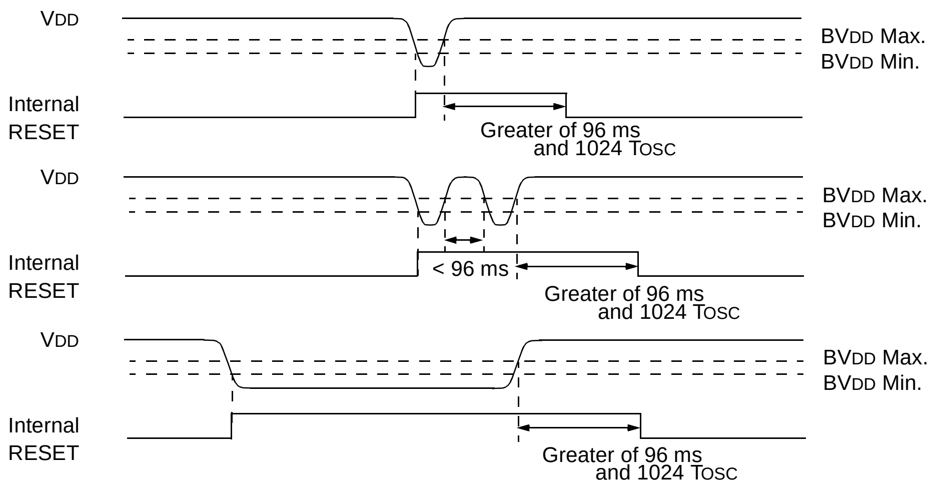
FIGURE 5-9: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 2



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

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

FIGURE 5-10: BROWN-OUT SITUATIONS



NOTES:

6.0 INTERRUPTS

PIC17C7XX devices have 18 sources of interrupt:

- External interrupt from the RA0/INT pin
- Change on RB7:RB0 pins
- TMR0 Overflow
- TMR1 Overflow
- TMR2 Overflow
- TMR3 Overflow
- USART1 Transmit buffer empty
- USART1 Receive buffer full
- USART2 Transmit buffer empty
- USART2 Receive buffer full
- SSP Interrupt
- SSP I^{2}C bus collision interrupt
- A/D conversion complete
- Capture1
- Capture2
- Capture3
- Capture4
- T0CKI edge occurred

There are six registers used in the control and status of interrupts. These are:

- CPUSTA
- INTSTA
- PIE1
- PIR1
- PIE2
- PIR2

The CPUSTA register contains the GLINTD bit. This is the Global Interrupt Disable bit. When this bit is set, all interrupts are disabled. This bit is part of the controller core functionality and is described in the Section 6.4.

When an interrupt is responded to, the GLINTD bit is automatically set to disable any further interrupts, the return address is pushed onto the stack and the PC is loaded with the interrupt vector address. There are four interrupt vectors. Each vector address is for a specific interrupt source (except the peripheral interrupts, which all vector to the same address). These sources are:

- External interrupt from the RA0/INT pin
- TMR0 Overflow
- T0CKI edge occurred
- Any peripheral interrupt

When program execution vectors to one of these interrupt vector addresses (except for the peripheral interrupts), the interrupt flag bit is automatically cleared. Vectoring to the peripheral interrupt vector address does not automatically clear the source of the interrupt. In the peripheral Interrupt Service Routine, the source(s) of the interrupt can be determined by testing the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid infinite interrupt requests.

When an interrupt condition is met, that individual interrupt flag bit will be set, regardless of the status of its corresponding mask bit or the GLINTD bit.

For external interrupt events, there will be an interrupt latency. For two-cycle instructions, the latency could be one instruction cycle longer.

The “return from interrupt” instruction, `RETFIE`, can be used to mark the end of the Interrupt Service Routine. When this instruction is executed, the stack is “POPed” and the GLINTD bit is cleared (to re-enable interrupts).

FIGURE 6-1: INTERRUPT LOGIC

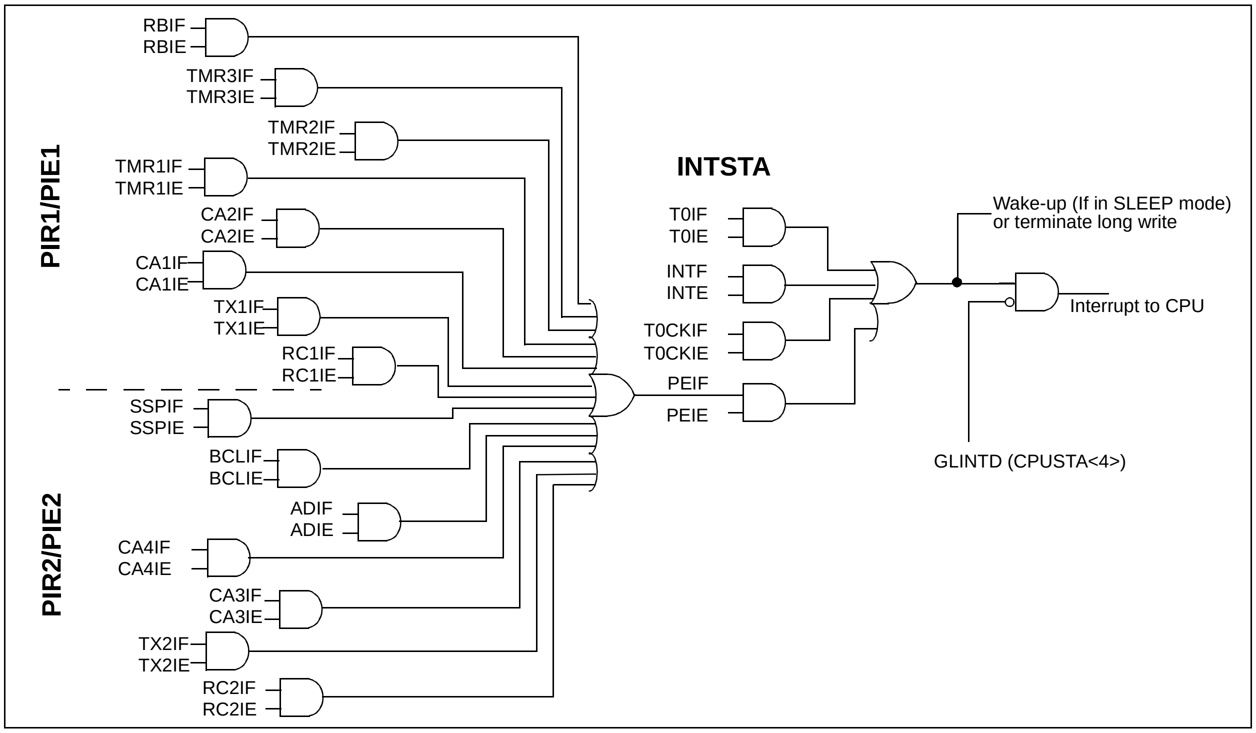


TABLE 7-1: MODE MEMORY ACCESS

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

The PIC17C7XX can operate in modes where the program memory is off-chip. They are the Microprocessor and Extended Microcontroller modes. The Microprocessor mode is the default for an unprogrammed device.

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

FIGURE 7-2: MEMORY MAP IN DIFFERENT MODES

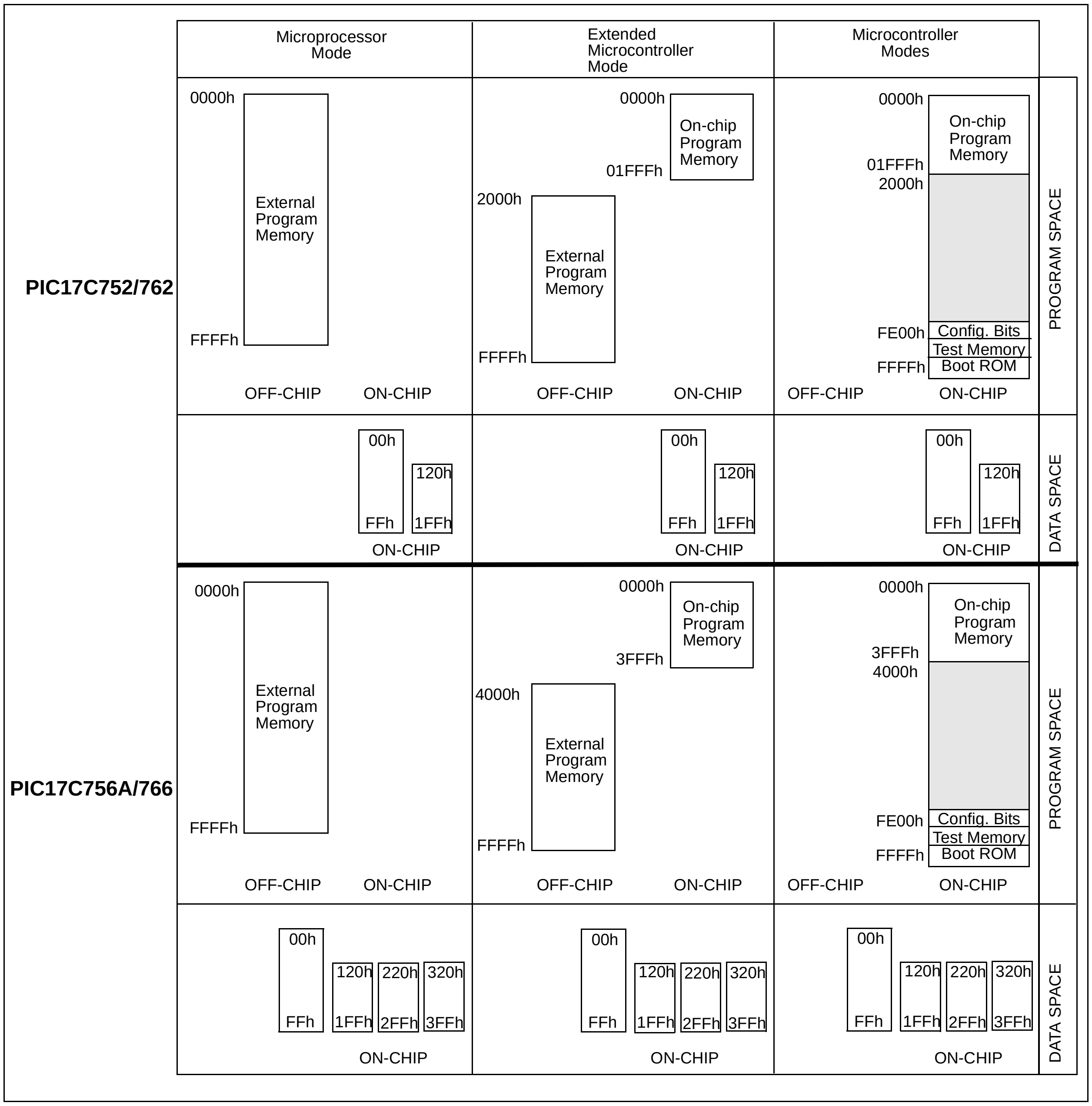


FIGURE 7-5: PIC17C7XX REGISTER FILE MAP

Addr	Unbanked
00h	INDF0
01h	FSR0
02h	PCL
03h	PCLATH
04h	ALUSTA
05h	T0STA
06h	CPUSTA
07h	INTSTA
08h	INDF1
09h	FSR1
0Ah	WREG
0Bh	TMR0L
0Ch	TMR0H
0Dh	TBLPTRL
0Eh	TBLPTRH
0Fh	BSR

	Bank 0	Bank 1[1]	Bank 2[1]	Bank 3[1]	Bank 4[1]	Bank 5[1]	Bank 6[1]	Bank 7[1]	Bank 8[1,4]
10h	PORTA	DDRC	TMR1	PW1DCL	PIR2	DDRF	SSPADD	PW3DCL	DDRH
11h	DDRB	PORTC	TMR2	PW2DCL	PIE2	PORTF	SSPCON1	PW3DCH	PORTH
12h	PORTB	DDRD	TMR3L	PW1DCH	—	DDRG	SSPCON2	CA3L	DDRJ
13h	RCSTA1	PORTD	TMR3H	PW2DCH	RCSTA2	PORTG	SSPSTAT	CA3H	PORTJ
14h	RCREG1	DDRE	PR1	CA2L	RCREG2	ADCON0	SSPBUF	CA4L	—
15h	TXSTA1	PORTE	PR2	CA2H	TXSTA2	ADCON1	—	CA4H	—
16h	TXREG1	PIR1	PR3L/CA1L	TCON1	TXREG2	ADRESL	—	TCON3	—
17h	SPBRG1	PIE1	PR3H/CA1H	TCON2	SPBRG2	ADRESH	—	—	—

	Unbanked
18h	PRODL
19h	PRODH
1Ah 1Fh	General Purpose RAM

	Bank 0[2]	Bank 1[2]	Bank 2[2]	Bank 3[2,3]
20h FFh	General Purpose RAM	General Purpose RAM	General Purpose RAM	General Purpose RAM

Note 1: SFR file locations 10h - 17h are banked. The lower nibble of the BSR specifies the bank. All unbanked SFRs ignore the Bank Select Register (BSR) bits.

2: General Purpose Registers (GPR) locations 20h - FFh, 120h - 1FFh, 220h - 2FFh, and 320h - 3FFh are banked. The upper nibble of the BSR specifies this bank. All other GPRs ignore the Bank Select Register (BSR) bits.

3: RAM bank 3 is not implemented on the PIC17C752 and the PIC17C762. Reading any unimplemented register reads '0's.

4: Bank 8 is only implemented on the PIC17C76X devices.

TABLE 7-3: SPECIAL FUNCTION REGISTERS

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
Unbanked											
00h	INDF0	Uses contents of FSR0 to address Data Memory (not a physical register)								`---- ----`	`---- ----`
01h	FSR0	Indirect Data Memory Address Pointer 0								`xxxx xxxx`	`uuuu uuuu`
02h	PCL	Low order 8-bits of PC								`0000 0000`	`0000 0000`
03h[1]	PCLATH	Holding Register for upper 8-bits of PC								`0000 0000`	`uuuu uuuu`
04h	ALUSTA	FS3	FS2	FS1	FS0	OV	Z	DC	C	`1111 xxxx`	`1111 uuuu`
05h	T0STA	INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—	`0000 000-`	`0000 000-`
06h[2]	CPUSTA	—	—	STKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	`--11 11qq`	`--11 qquu`
07h	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	`0000 0000`	`0000 0000`
08h	INDF1	Uses contents of FSR1 to address Data Memory (not a physical register)								`---- ----`	`---- ----`
09h	FSR1	Indirect Data Memory Address Pointer 1								`xxxx xxxx`	`uuuu uuuu`
0Ah	WREG	Working Register								`xxxx xxxx`	`uuuu uuuu`
0Bh	TMR0L	TMR0 Register; Low Byte								`xxxx xxxx`	`uuuu uuuu`
0Ch	TMR0H	TMR0 Register; High Byte								`xxxx xxxx`	`uuuu uuuu`
0Dh	TBLPTRL	Low Byte of Program Memory Table Pointer								`0000 0000`	`0000 0000`
0Eh	TBLPTRH	High Byte of Program Memory Table Pointer								`0000 0000`	`0000 0000`
0Fh	BSR	Bank Select Register								`0000 0000`	`0000 0000`
Bank 0											
10h	PORTA[4,6]	RBPU	—	RA5/TX1/ CK1	RA4/RX1/ DT1	RA3/SDI/ SDA	RA2/$\overline{\text{SS}}$/ SCL	RA1/T0CKI	RA0/INT	`0-xx 11xx`	`0-uu 11uu`
11h	DDRB	Data Direction Register for PORTB								`1111 1111`	`1111 1111`
12h	PORTB[4]	RB7/ SDO	RB6/ SCK	RB5/ TCLK3	RB4/ TCLK12	RB3/ PWM2	RB2/ PWM1	RB1/ CAP2	RB0/ CAP1	`xxxx xxxx`	`uuuu uuuu`
13h	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	`0000 -00x`	`0000 -00u`
14h	RCREG1	Serial Port Receive Register								`xxxx xxxx`	`uuuu uuuu`
15h	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	`0000 --1x`	`0000 --1u`
16h	TXREG1	Serial Port Transmit Register (for USART1)								`xxxx xxxx`	`uuuu uuuu`
17h	SPBRG1	Baud Rate Generator Register (for USART1)								`0000 0000`	`0000 0000`
Bank 1											
10h	DDRC[5]	Data Direction Register for PORTC								`1111 1111`	`1111 1111`
11h	PORTC[4,5]	RC7/AD7	RC6/AD6	RC5/AD5	RC4/AD4	RC3/AD3	RC2/AD2	RC1/AD1	RC0/AD0	`xxxx xxxx`	`uuuu uuuu`
12h	DDRD[5]	Data Direction Register for PORTD								`1111 1111`	`1111 1111`
13h	PORTD[4,5]	RD7/ AD15	RD6/ AD14	RD5/ AD13	RD4/ AD12	RD3/ AD11	RD2/ AD10	RD1/AD9	RD0/AD8	`xxxx xxxx`	`uuuu uuuu`
14h	DDRE[5]	Data Direction Register for PORTE								`---- 1111`	`---- 1111`
15h	PORTE[4,5]	—	—	—	—	RE3/ CAP4	RE2/$\overline{\text{WR}}$	RE1/$\overline{\text{OE}}$	RE0/ALE	`---- xxxx`	`---- uuuu`
16h	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	`x000 0010`	`u000 0010`
17h	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	`0000 0000`	`0000 0000`

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.

7.2.2.3 TMR0 Status/Control Register (T0STA)

This register contains various control bits. Bit7 (INTEDG) is used to control the edge upon which a signal on the RA0/INT pin will set the RA0/INT interrupt flag. The other bits configure Timer0, it's prescaler and clock source.

REGISTER 7-3: T0STA REGISTER (ADDRESS: 05h, UNBANKED)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—
bit 7							bit 0

bit 7 **INTEDG**: RA0/INT Pin Interrupt Edge Select bit
This bit selects the edge upon which the interrupt is detected.
1 = Rising edge of RA0/INT pin generates interrupt
0 = Falling edge of RA0/INT pin generates interrupt

bit 6 **T0SE**: Timer0 External Clock Input Edge Select bit
This bit selects the edge upon which TMR0 will increment.

When T0CS = 0 (External Clock):
1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or sets the T0CKIF bit
0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or sets a T0CKIF bit

When T0CS = 1 (Internal Clock):
Don't care

bit 5 **T0CS**: Timer0 Clock Source Select bit
This bit selects the clock source for Timer0.
1 = Internal instruction clock cycle (TCY)
0 = External clock input on the T0CKI pin

bit 4-1 **T0PS3:T0PS0**: Timer0 Prescale Selection bits
These bits select the prescale value for Timer0.

T0PS3:T0PS0	Prescale Value
0000	1:1
0001	1:2
0010	1:4
0011	1:8
0100	1:16
0101	1:32
0110	1:64
0111	1:128
1xxx	1:256

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Example 9-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 9-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers, RES3:RES0.

EQUATION 9-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned} RES3{:}RES0 &= ARG1H{:}ARG1L \bullet ARG2H{:}ARG2L \\ &= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\ &\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\ &\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\ &\quad (ARG1L \bullet ARG2L) \end{aligned}$$

EXAMPLE 9-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L        ; ARG1L * ARG2L ->
                         ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H        ; ARG1H * ARG2H ->
                         ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H        ; ARG1L * ARG2H ->
                         ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F      ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F      ;
   CLRF     WREG, F      ;
   ADDWFC   RES3, F      ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L        ; ARG1H * ARG2L ->
                         ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F      ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F      ;
   CLRF     WREG, F      ;
   ADDWFC   RES3, F      ;
```

10.9 PORTJ and DDRJ Registers (PIC17C76X only)

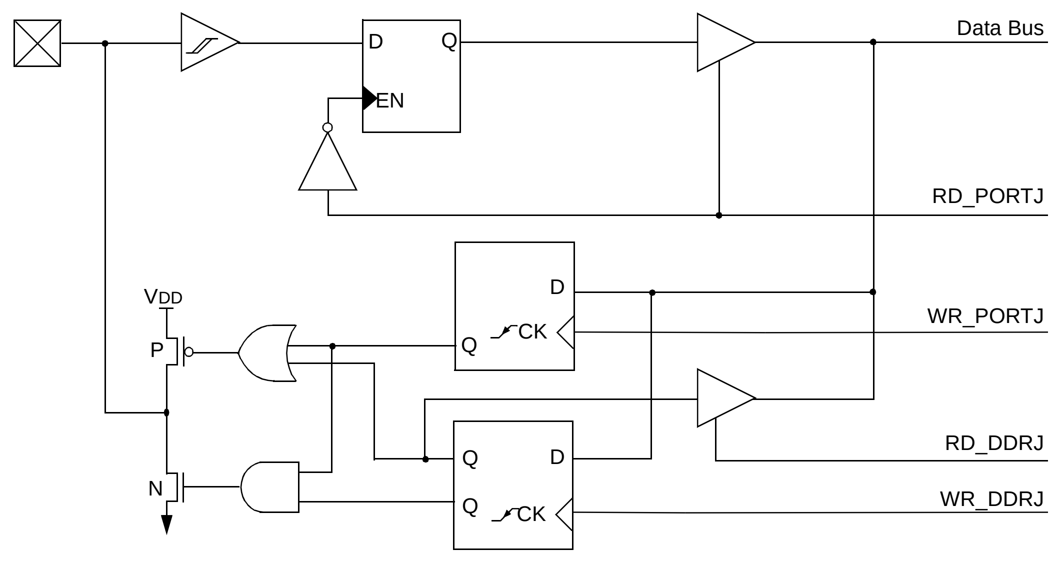
PORTJ is an 8-bit wide, bi-directional port. The corresponding data direction register is DDRJ. A '1' in DDRJ configures the corresponding port pin as an input. A '0' in the DDRJ register configures the corresponding port pin as an output. Reading PORTJ reads the status of the pins, whereas writing to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

PORTJ is a general purpose I/O port.

EXAMPLE 10-9: INITIALIZING PORTJ

```
MOVLB   8          ; Select Bank 8
CLRF    PORTJ, F   ; Initialize PORTJ data
                   ; latches before setting
                   ; the data direction
                   ; register
MOVLW    0xCF      ; Value used to initialize
                   ; data direction
MOVWF    DDRJ      ; Set RJ<3:0> as inputs
                   ; RJ<5:4> as outputs
                   ; RJ<7:6> as inputs
```

FIGURE 10-19: PORTJ BLOCK DIAGRAM



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

10.10 I/O Programming Considerations

10.10.1 BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read, followed by a write operation. For example, the `BCF` and `BSF` instructions read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a `BSF` operation on bit5 of PORTB, will cause all eight bits of PORTB to be read into the CPU. Then the `BSF` operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bi-directional I/O pin (e.g. bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and rewritten to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched into output mode later on, the content of the data latch may now be unknown.

Reading a port reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (`BCF, BSF`, `BTG`, etc.) on a port, the value of the port pins is read, the desired operation is performed with this value and the value is then written to the port latch.

Example 10-10 shows the possible effect of two sequential read-modify-write instructions on an I/O port.

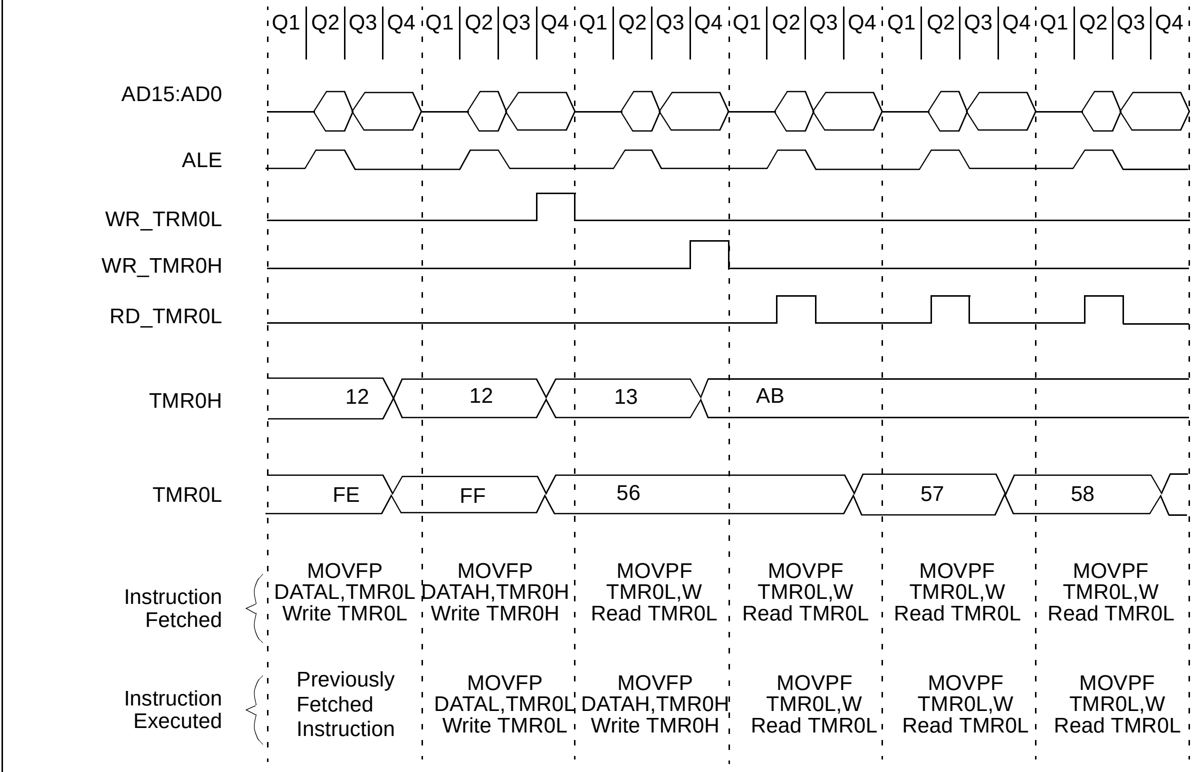
EXAMPLE 10-10: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

```
; Initial PORT settings: PORTB<7:4> Inputs
;                        PORTB<3:0> Outputs
; PORTB<7:6> have pull-ups and are
; not connected to other circuitry
;
;                      PORT latch  PORT pins
;                      ----------  ---------
;
   BCF   PORTB, 7   ; 01pp pppp   11pp pppp
   BCF   PORTB, 6   ; 10pp pppp   11pp pppp

   BCF   DDRB, 7    ; 10pp pppp   11pp pppp
   BCF   DDRB, 6    ; 10pp pppp   10pp pppp
;
; Note that the user may have expected the
; pin values to be 00pp pppp. The 2nd BCF
; caused RB7 to be latched as the pin value
; (High).
```

Note:	A pin actively outputting a Low or High should not be driven from external devices, in order to change the level on this pin (i.e., “wired-or”, “wired-and”). The resulting high output currents may damage the device.

FIGURE 12-4: TMR0 READ/WRITE IN TIMER MODE



Note: In this example, old TMR0 value is 12FEh, new value of AB56h is written.

TABLE 12-1: REGISTERS/BITS ASSOCIATED WITH TIMER0

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
05h, Unbanked	T0STA	INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—	0000 000-	0000 000-
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	--11 11qq	--11 qquu
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
0Bh, Unbanked	TMR0L	TMR0 Register; Low Byte								xxxx xxxx	uuuu uuuu
0Ch, Unbanked	TMR0H	TMR0 Register; High Byte								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as a '0', q = value depends on condition. Shaded cells are not used by Timer0.

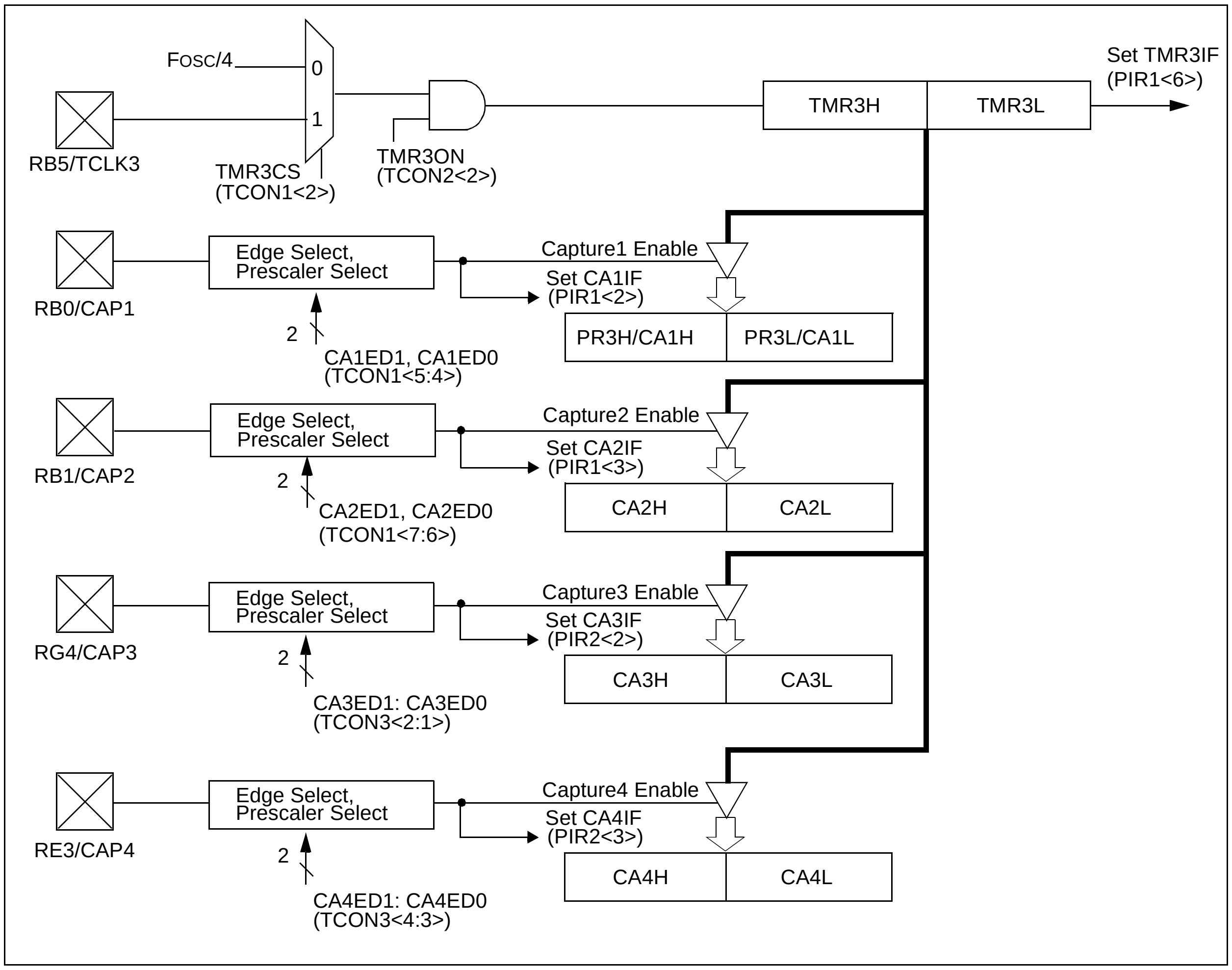
13.2.2 FOUR CAPTURE MODE

This mode is selected by setting bit CA1/$\overline{\text{PR3}}$. A block diagram is shown in Figure 13-6. In this mode, TMR3 runs without a period register and increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt Flag (TMR3IF) is set on this rollover. The TMR3IF bit must be cleared in software.

Registers PR3H/CA1H and PR3L/CA1L make a 16-bit capture register (Capture1). It captures events on pin RB0/CAP1. Capture mode is configured by the CA1ED1 and CA1ED0 bits. Capture1 Interrupt Flag bit (CA1IF) is set upon detection of the capture event. The corresponding interrupt mask bit is CA1IE. The Capture1 Overflow Status bit is CA1OVF.

All the captures operate in the same manner. Refer to Section 13.2.1 for the operation of capture.

FIGURE 13-6: TIMER3 WITH FOUR CAPTURES BLOCK DIAGRAM

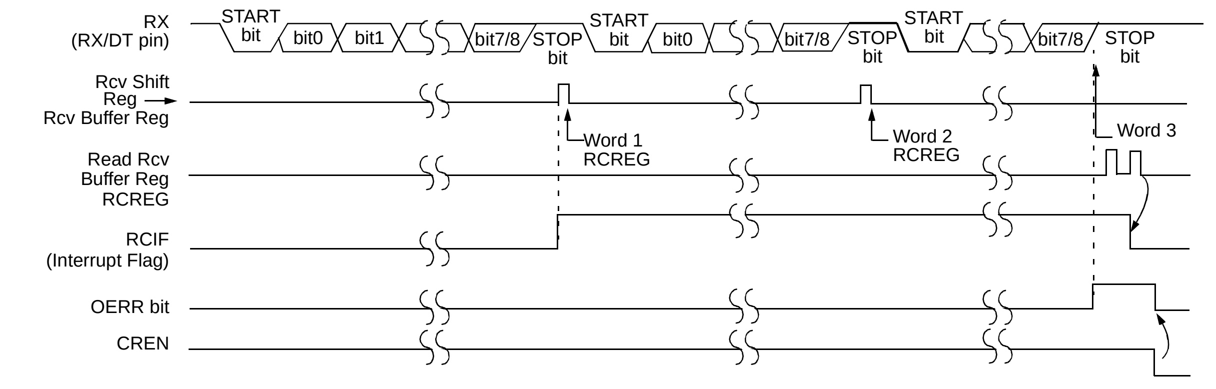


Steps to follow when setting up an Asynchronous Reception:

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 RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. Enable the reception by setting the CREN bit.
6. The RCIF bit will be set when reception completes and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and FERR bit to determine if any error occurred during reception.
8. Read RCREG for the 8-bit received data.
9. If an overrun error occurred, clear the error by clearing the OERR bit.

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 14-7: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

TABLE 14-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

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
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	`x000 0010`	`u000 0010`
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	`0000 0000`	`0000 0000`
13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	`0000 -00x`	`0000 -00u`
14h, Bank 0	RCREG1	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	`xxxx xxxx`	`uuuu uuuu`
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`
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	`000- 0010`	`000- 0010`
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	`000- 0000`	`000- 0000`
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	`0000 -00x`	`0000 -00u`
14h, Bank 4	RCREG2	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	`xxxx xxxx`	`uuuu uuuu`
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 a '0'. Shaded cells are not used for asynchronous reception.

REGISTER 15-2: SSPCON1: SYNC SERIAL PORT CONTROL REGISTER1 (ADDRESS 11h, BANK 6)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	$\overline{\text{SSPM1}}$	$\overline{\text{SSPM0}}$
bit 7							bit 0

bit 7 **WCOL**: Write Collision Detect bit

Master mode:
1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started
0 = No collision

Slave mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV**: Receive Overflow Indicator bit

In SPI mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set, since each new reception (and transmission) is initiated by writing to the SSPBUF register. (Must be cleared in software.)
0 = No overflow

In I^2C mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a “don’t care” in Transmit mode. (Must be cleared in software.)
0 = No overflow

bit 5 **SSPEN**: Synchronous Serial Port Enable bit
In both modes, when enabled, these pins must be properly configured as input or output.

In SPI mode:
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{\text{SS}}$ as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

In I^2C mode:
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

Note: In SPI mode, these pins must be properly configured as input or output.

bit 4 **CKP**: Clock Polarity Select bit

In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

In I^2C Slave mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In I^2C Master mode:
Unused in this mode

bit 3-0 **SSPM3:SSPM0**: Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = FOSC/4
0001 = SPI Master mode, clock = FOSC/16
0010 = SPI Master mode, clock = FOSC/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control disabled, $\overline{\text{SS}}$ can be used as I/O pin
0110 = I^2C Slave mode, 7-bit address
0111 = I^2C Slave mode, 10-bit address
1000 = I^2C Master mode, clock = FOSC / (4 * (SSPADD+1))
1xx1 = Reserved
1x1x = Reserved

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR Reset	’1’ = Bit is set	’0’ = Bit is cleared	x = Bit is unknown

DCFSNZ — Decrement f, skip if not 0

Syntax:	[*label*] DCFSNZ f,d
Operands:	$0 \le f \le 255$ $d \in [0,1]$
Operation:	(f) – 1 → (dest); skip if not 0
Status Affected:	None

Encoding:

`0010`	`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 not 0, the next instruction, which is already fetched is discarded and a `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'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE     DCFSNZ   TEMP, 1
ZERO     :
NZERO    :
```

Before Instruction

TEMP_VALUE	=	?

After Instruction

TEMP_VALUE	=	TEMP_VALUE - 1,
If TEMP_VALUE	=	0;
PC	=	Address (`ZERO`)
If TEMP_VALUE	≠	0;
PC	=	Address (`NZERO`)

GOTO — Unconditional Branch

Syntax:	[*label*] GOTO k
Operands:	$0 \le k \le 8191$
Operation:	k → PC<12:0>; k<12:8> → PCLATH<4:0>, PC<15:13> → PCLATH<7:5>
Status Affected:	None

Encoding:

`110k`	`kkkk`	`kkkk`	`kkkk`

Description: `GOTO` allows an unconditional branch anywhere within an 8K page boundary. The thirteen-bit immediate value is loaded into PC bits <12:0>. Then the upper eight bits of PC are loaded into PCLATH. `GOTO` is always a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to PC
No operation	No operation	No operation	No operation

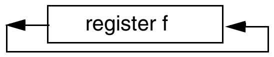
Example: `GOTO THERE`

After Instruction

PC = Address (`THERE`)

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'	Process Data	Write to destination

Example: `RLNCF REG, 1`

Before Instruction

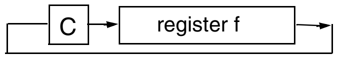
C	=	`0`
REG	=	`1110 1011`

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'	Process Data	Write to destination

Example: `RRCF REG1,0`

Before Instruction

REG1	=	`1110 0110`
C	=	`0`

After Instruction

REG1	=	`1110 0110`
WREG	=	`0111 0011`
C	=	`0`

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.

20.3 Timing Parameter Symbology

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

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

T			
F	Frequency	T	Time

Lowercase symbols (pp) and their meanings:

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

Uppercase symbols and their meanings:

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

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	X	/XX	XXX
Device	Temperature Range	Package	Pattern

Device	PIC17C756: Standard VDD range PIC17C756T: (Tape and Reel) PIC17LC756: Extended VDD range		
Temperature Range	- I	= =	0°C to +70°C -40°C to +85°C
Package	CL PT L	= = =	Windowed LCC TQFP PLCC
Pattern	QTP, SQTP, ROM Code (factory specified) or Special Requirements . Blamk for OTP and Windowed devices.		

Examples:

a) PIC17C756 – 16L Commercial Temp., PLCC package, 16 MHz, normal VDD limits

b) PIC17LC756–08/PT Commercial Temp., TQFP package, 8MHz, extended VDD limits

c) PIC17C756–33I/PT Industrial Temp., TQFP package, 33 MHz, normal VDD limits

* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type.

Sales and Support

Data Sheets
Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System
Register on our web site (www.microchip.com/cn) to receive the most current information on our products.