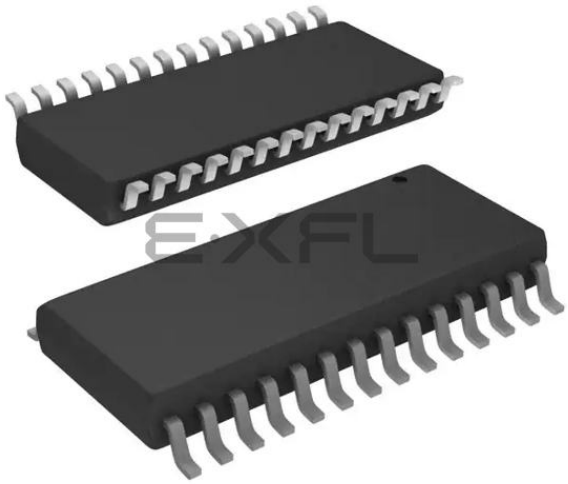


Welcome to E-XFL.COM

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	48MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	22
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2.15V ~ 3.6V
Data Converters	A/D 10x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f27j53-i-so

REGISTER 3-1: OSCTUNE: OSCILLATOR TUNING REGISTER (ACCESS F9Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTSRC	PLLEN(1)	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0
bit 7							bit 0

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

bit 7 **INTSRC:** Internal Oscillator Low-Frequency Source Select bit
1 = 31.25 kHz device clock derived from 8 MHz INTOSC source (divide-by-256 enabled)
0 = 31 kHz device clock derived directly from INTRC internal oscillator

bit 6 **PLLEN:** Frequency Multiplier Enable bit(1)
1 = 96 MHz PLL is enabled
0 = 96 MHz PLL is disabled

bit 5-0 **TUN<5:0>:** Frequency Tuning bits
011111 = Maximum frequency
011110
•
•
•
000001
000000 = Center frequency; oscillator module is running at the calibrated frequency
111111
•
•
•
100000 = Minimum frequency

Note 1: When the CFGPLLEN Configuration bit is used to enable the PLL, clearing OSCTUNE<6> will not disable the PLL.

3.3 Oscillator Settings for USB

When the PIC18F47J53 family devices are used for USB connectivity, a 6 MHz or 48 MHz clock must be provided to the USB module for operation in either Low-Speed or Full-Speed modes, respectively. This may require some forethought in selecting an oscillator frequency and programming the device.

The full range of possible oscillator configurations compatible with USB operation is shown in Table 3-5.

3.3.1 LOW-SPEED OPERATION

The USB clock for Low-Speed mode is derived from the primary oscillator or from the 96 MHz PLL. In order to operate the USB module in Low-Speed mode, a 6 MHz clock must be provided to the USB module.

See Table 3-4 and Table 3-5 for possible combinations which can be used for low-speed USB operation.

TABLE 3-4: CLOCK FOR LOW-SPEED USB

System Clock	CPDIV<1:0>	Microcontroller Clock	LS48MHZ	USB Clock
48	11	48 MHz	1	48/8 = 6 MHz
48	10	48/2 = 24 MHz	1	48/8 = 6 MHz
48	01	48/3 = 16 MHz	1	48/8 = 6 MHz
48	00	48/6 = 8 MHz	1	48/8 = 6 MHz
24	11	24 MHz	0	24/4 = 6 MHz
24	10	24/2 = 12 MHz	0	24/4 = 6 MHz
24	01	24/3 = 8 MHz	0	24/4 = 6 MHz
24	00	24/6 = 4 MHz	0	24/4 = 6 MHz

4.2.3 RC_RUN MODE

In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator; the primary clock is shut down. This mode provides the best power conservation of all the Run modes while still executing code. It works well for user applications, which are not highly timing-sensitive or do not require high-speed clocks at all times.

This mode is entered by setting the SCS<1:0> bits (OSCCON<1:0>) to '11'. When the clock source is switched to the internal oscillator block (see Figure 4-3), the primary oscillator is shut down and the OSTS bit is cleared.

On transitions from RC_RUN mode to PRI_RUN mode, the device continues to be clocked from the INTOSC block while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 4-4). When the clock switch is complete, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCS bits are not affected by the switch. The INTRC clock source will continue to run if either the WDT or the FSCM is enabled.

FIGURE 4-3: TRANSITION TIMING TO RC_RUN MODE

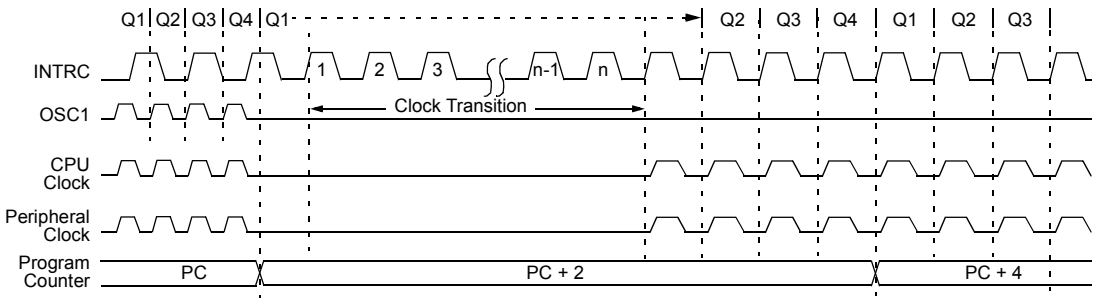
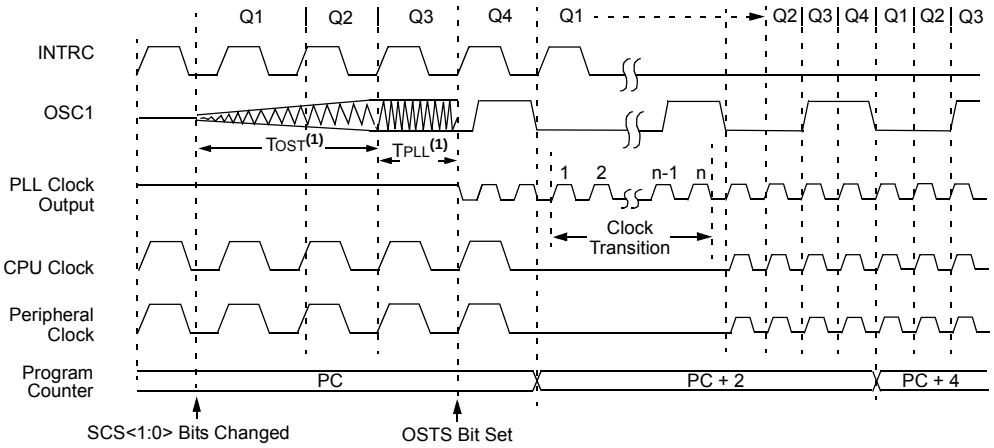


FIGURE 4-4: TRANSITION TIMING FROM RC_RUN MODE TO PRI_RUN MODE



Note 1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

EXAMPLE 4-1: ULTRA LOW-POWER WAKE-UP INITIALIZATION

```
//******************************************************************************
//Configure a remappable output pin with interrupt capability
//for ULPWU function (RP21 => RD4/INT1 in this example)
//******************************************************************************
RPOR21 = 13;// ULPWU function mapped to RP21/RD4
RPINR1 = 21;// INT1 mapped to RP21 (RD4)

//***************************
//Charge the capacitor on RA0
//***************************
TRISAbits.TRISA0 = 0;
PORTAbits.RA0 = 1;
for(i = 0; i < 10000; i++) Nop();

//*********************************
//Stop Charging the capacitor on RA0
//*********************************
TRISAbits.TRISA0 = 1;

//****************************************
//Enable the Ultra Low Power Wakeup module
//and allow capacitor discharge
//****************************************
WDTCONbits.ULPEN = 1;
WDTCONbits.ULPSINK = 1;

//****************************************
//Enable Interrupt for ULPW
//****************************************
//For Sleep
//(assign the ULPOUT signal in the PPS module to a pin
//which has also been assigned an interrupt capability,
//such as INT1)
INTCON3bits.INT1IF = 0;
INTCON3bits.INT1IE = 1;

//********************
//Configure Sleep Mode
//********************
//For Sleep
OSCCONbits.IDLEN = 0;

//For Deep Sleep
OSCCONbits.IDLEN = 0;  // enable deep sleep
DSCONHbits.DSEN = 1;   // Note: must be set just before executing Sleep();
//*****************
//Enter Sleep Mode
//*****************
Sleep();
// for sleep, execution will resume here
// for deep sleep, execution will restart at reset vector (use WDTCONbits.DS to detect)
```

6.4 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 "Data Memory and the Extended Instruction Set"** for more information.

While the program memory can be addressed in only one way through the PC, information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in more detail in **Section 6.6.1 "Indexed Addressing with Literal Offset"**.

6.4.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device, or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include SLEEP, RESET and DAW.

Other instructions work in a similar way, but require an additional explicit argument in the opcode. This is known as Literal Addressing mode, because they require some literal value as an argument. Examples include ADDLW and MOVLW, which respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a 20-bit program memory address.

6.4.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their LSB. This address specifies either a register address in one of the banks of data RAM (**Section 6.3.4 "General Purpose Register File"**) or a location in the Access Bank (**Section 6.3.3 "Access Bank"**) as the data source for the instruction.

The Access RAM bit, 'a', determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 6.3.2 "Bank Select Register"**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit, 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

6.4.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as SFRs, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code using loops, such as the example of clearing an entire RAM bank in Example 6-5. It also enables users to perform Indexed Addressing and other Stack Pointer operations for program memory in data memory.

EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR    FSR0, 100h ;
NEXT    CLRF    POSTINC0   ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS   FSR0H, 1   ; All done with
                           ; Bank1?
        BRA     NEXT       ; NO, clear next
CONTINUE                   ; YES, continue
```

7.5.2 FLASH PROGRAM MEMORY WRITE SEQUENCE (WORD PRORAMMING).

The PIC18F47J53 family of devices has a feature that allows programming a single word (two bytes). This feature is enabled when the WPROG bit is set. If the memory location is already erased, the following sequence is required to enable this feature:

1. Load the Table Pointer register with the address of the data to be written. (It must be an even address.)
2. Write the 2 bytes into the holding registers by performing table writes. (Do not post-increment on the second table write.)
3. Set the WREN bit (EECON1<2>) to enable writes and the WPROG bit (EECON1<5>) to select Word Write mode.
4. Disable interrupts.
5. Write 55h to EECON2.
6. Write 0AAh to EECON2.
7. Set the WR bit; this will begin the write cycle.
8. The CPU will stall for the duration of the write for TIW (see parameter D133A).
9. Re-enable interrupts.

EXAMPLE 7-4: SINGLE-WORD WRITE TO FLASH PROGRAM MEMORY

```
                MOVLW   CODE_ADDR_UPPER         ; Load TBLPTR with the base address
                MOVWF   TBLPTRU
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW           ; The table pointer must be loaded with an even
                                                  address
                MOVWF   TBLPTRL

                MOVLW   DATA0                   ; LSB of word to be written
                MOVWF   TABLAT
                TBLWT*+
                MOVLW   DATA1                   ; MSB of word to be written
                MOVWF   TABLAT
                TBLWT*                          ; The last table write must not increment the table
                                                  pointer! The table pointer needs to point to the
                                                  MSB before starting the write operation.


PROGRAM_MEMORY
                BSF     EECON1, WPROG           ; enable single word write
                BSF     EECON1, WREN            ; enable write to memory
                BCF     INTCON, GIE             ; disable interrupts
                MOVLW   55h
    Required    MOVWF   EECON2                  ; write 55h
    Sequence    MOVLW   0AAh
                MOVWF   EECON2                  ; write AAh
                BSF     EECON1, WR              ; start program (CPU stall)
                BSF     INTCON, GIE             ; re-enable interrupts
                BCF     EECON1, WPROG           ; disable single word write
                BCF     EECON1, WREN            ; disable write to memory
```

FIGURE 9-1: PIC18F47J53 FAMILY INTERRUPT LOGIC

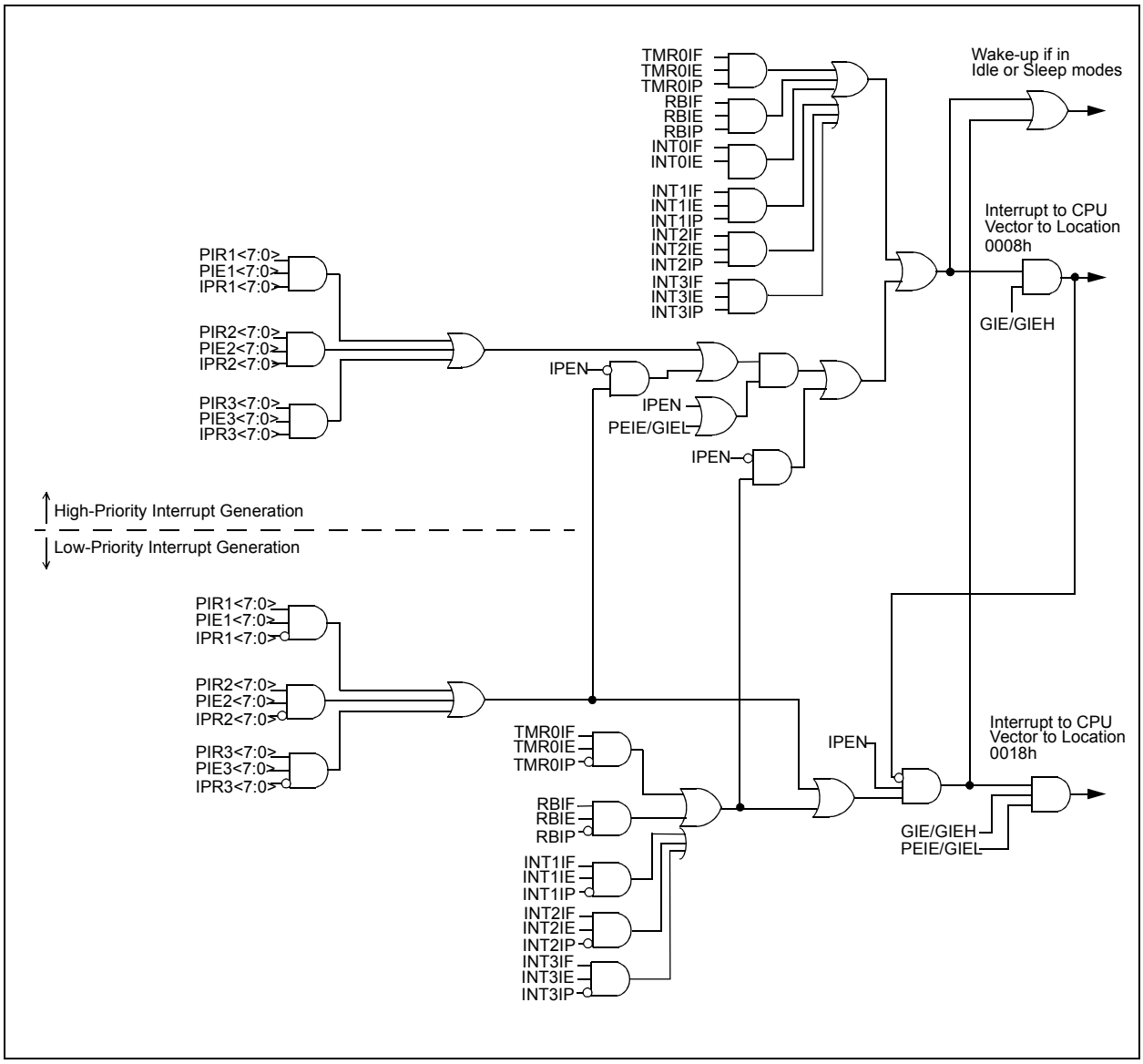


FIGURE 11-18: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE

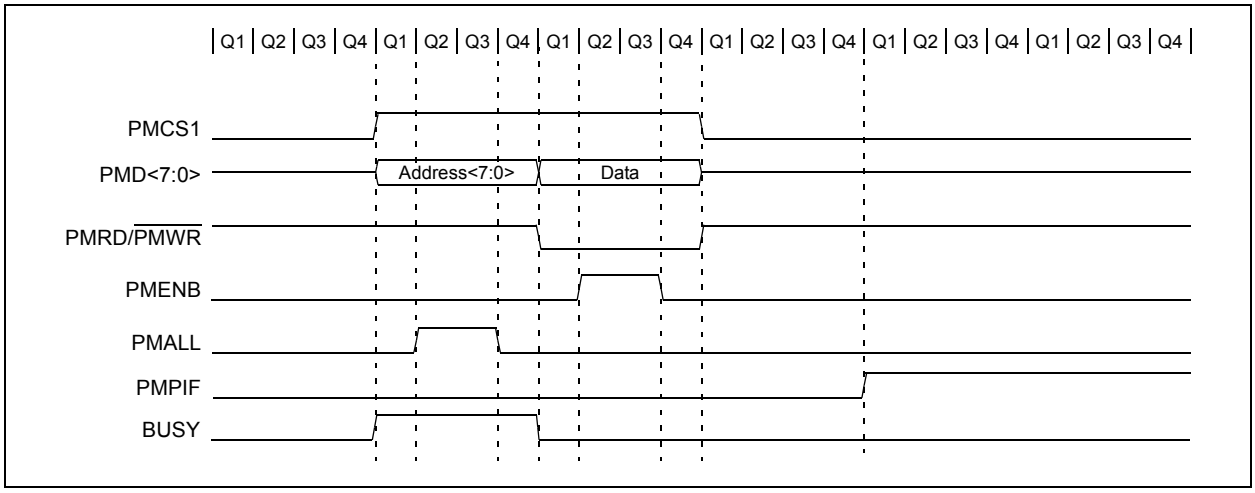


FIGURE 11-19: READ TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS

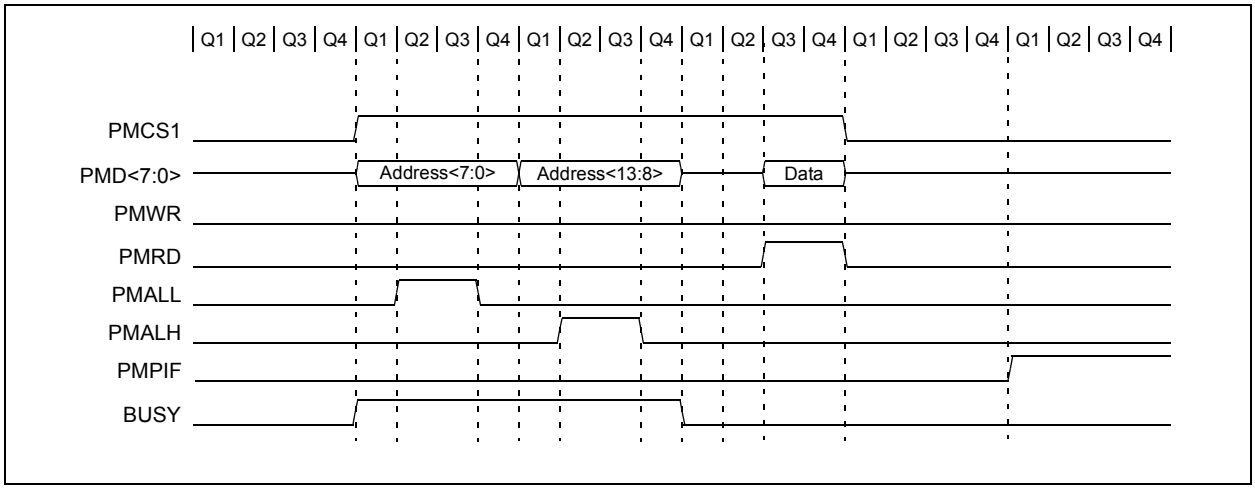


FIGURE 11-20: WRITE TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS

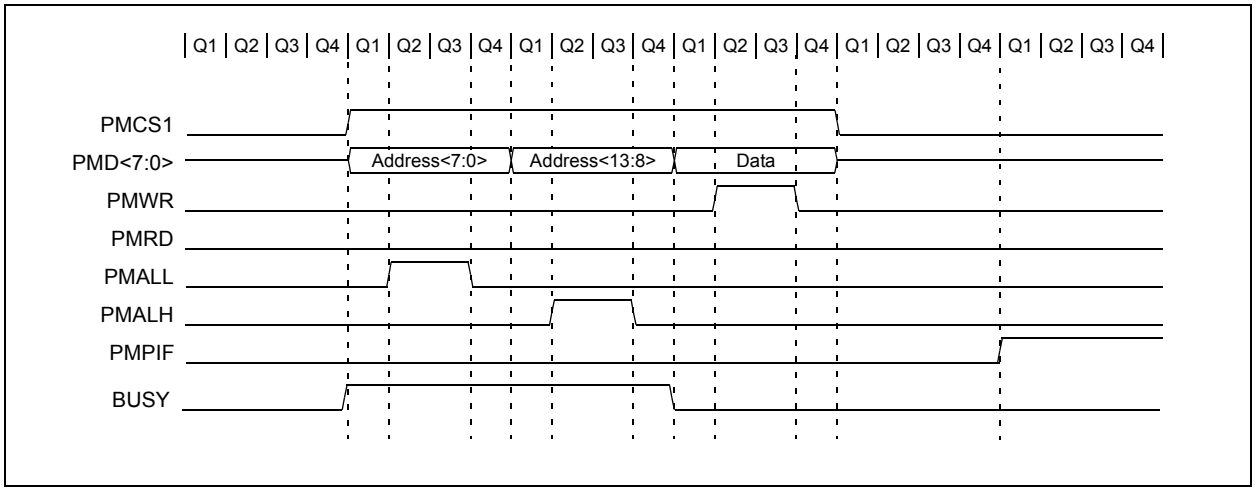


FIGURE 13-7: TIMER1 GATE SINGLE PULSE AND TOGGLE COMBINED MODE

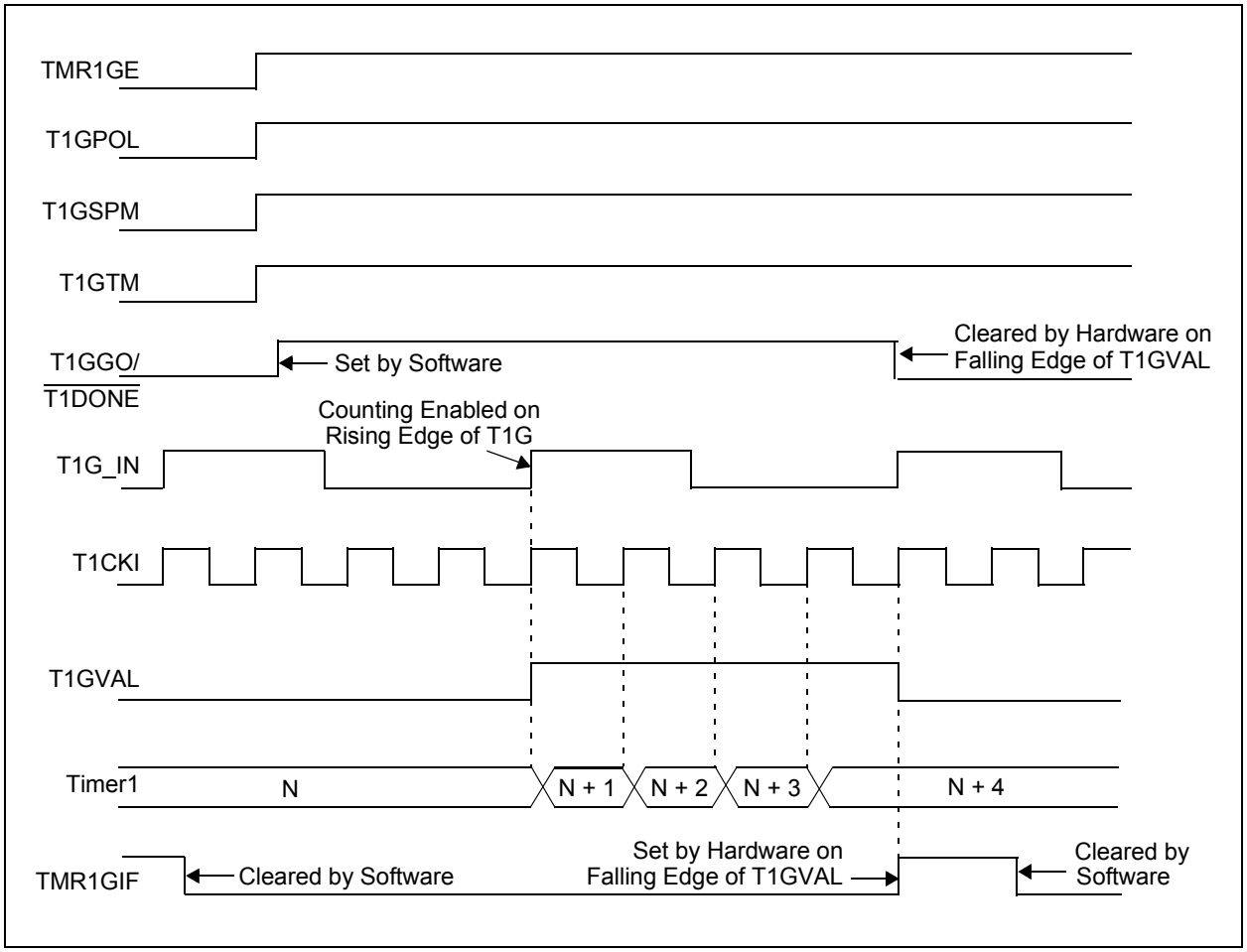


TABLE 13-5: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(1)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(1)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
TMR1L	Timer1 Register Low Byte							
TMR1H	Timer1 Register High Byte							
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	RD16	TMR1ON
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	$\overline{\text{T1GGO}}$/$\overline{\text{T1DONE}}$	T1GVAL	T1GSS1	T1GSS0
OSCCON2	—	SOSCRUN	—	SOSCDRV	SOSCGO	PRISD	—	—

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: These bits are only available in 44-pin devices.

REGISTER 15-1: TxCON: TIMER3/5 CONTROL REGISTER (ACCESS F79h, BANKED F22h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMRxCS1	TMRxCS0	TxCKPS1	TxCKPS0	TxOSCEN	$\overline{\text{TxSYNC}}$	RD16	TMRxON
bit 7							bit 0

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

bit 7-6 **TMRxCS<1:0>:** Clock Source Select bits
10 = Clock source is the pin or TxCKI input pin
01 = Clock source is the system clock (FOSC)(1)
00 = Clock source is the instruction clock (FOSC/4)

bit 5-4 **TxCKPS<1:0>**: Timerx Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **TxOSCEN**: Timer Oscillator Enable bit
1 = T1OSC/SOSC oscillator used as clock source
0 = TxCKI digital input pin used as clock source

bit 2 **TxSYNC:** External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)
When TMRxCS1:TMRxCS0 = 10:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMRxCS1:TMRxCS0 = 0x:
This bit is ignored; Timer3 uses the internal clock.

bit 1 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of timer in one 16-bit operation
0 = Enables register read/write of timer in two eight-bit operations

bit 0 **TMRxON:** Timer On bit
1 = Enables Timer
0 = Stops Timer

Note 1: The FOSC clock source should not be selected if the timer will be used with the ECCP capture/compare features.

REGISTER 19-2: CCPTMRS0: ECCP1/2/3 TIMER SELECT 0 REGISTER (BANKED F52h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C3TSEL1	C3TSEL0	C2TSEL2	C2TSEL1	C2TSEL0	C1TSEL2	C1TSEL1	C1TSEL0
bit 7							bit 0

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

bit 7-6 **C3TSEL<1:0>:** ECCP3 Timer Selection bits
- 00 = ECCP3 is based off of TMR1/TMR2
- 01 = ECCP3 is based off of TMR3/TMR4
- 10 = ECCP3 is based off of TMR3/TMR6
- 11 = ECCP3 is based off of TMR3/TMR8

bit 5-3 **C2TSEL<2:0>:** ECCP2 Timer Selection bits
- 000 = ECCP2 is based off of TMR1/TMR2
- 001 = ECCP2 is based off of TMR3/TMR4
- 010 = ECCP2 is based off of TMR3/TMR6
- 011 = ECCP2 is based off of TMR3/TMR8
- 1xx = Reserved; do not use

bit 2-0 **C1TSEL<2:0>:** ECCP1 Timer Selection bits
- 000 = ECCP1 is based off of TMR1/TMR2
- 001 = ECCP1 is based off of TMR3/TMR4
- 010 = ECCP1 is based off of TMR3/TMR6
- 011 = ECCP1 is based off of TMR3/TMR8
- 1xx = Reserved; do not use

In addition to the expanded range of modes available through the CCPxCON and ECCPxAS registers, the ECCP modules have two additional registers associated with Enhanced PWM operation and auto-shutdown features. They are:

- ECCPxDEL – Enhanced PWM Control
- PSTRxCON – Pulse Steering Control

EXAMPLE 21-1: CALCULATING BAUD RATE ERROR

```
For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, and
8-bit BRG:
Desired Baud Rate = Fosc/(64 ([SPBRGHx:SPBRGx] + 1))
Solving for SPBRGHx:SPBRGx:
    X  =  ((Fosc/Desired Baud Rate)/64) – 1
       =  ((16000000/9600)/64) – 1
       =  [25.042] = 25
Calculated Baud Rate=16000000/(64 (25 + 1))
       =  9615
Error  =  (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
       =  (9615 – 9600)/9600 = 0.16%
```

TABLE 21-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator High Byte							
SPBRGx	EUSARTx Baud Rate Generator Low Byte							

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the BRG.

23.2.3 USB STATUS REGISTER (USTAT)

The USB Status register reports the transaction status within the SIE. When the SIE issues a USB transfer complete interrupt, USTAT should be read to determine the status of the transfer. USTAT contains the transfer endpoint number, direction and Ping-Pong Buffer Pointer value (if used).

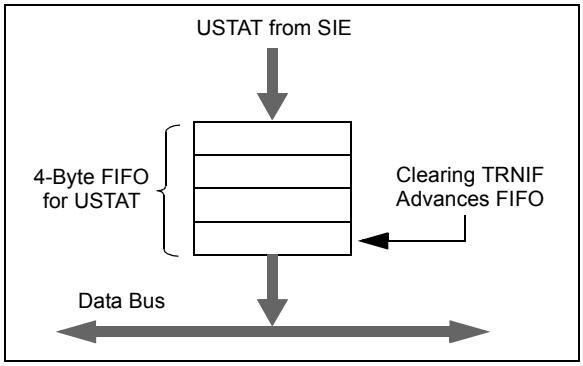
Note:	The data in the USB Status register is valid only when the TRNIF interrupt flag is asserted.

The USTAT register is actually a read window into a 4-byte status FIFO, maintained by the SIE. It allows the microcontroller to process one transfer while the SIE processes additional endpoints (Figure 23-3). When the SIE completes using a buffer for reading or writing data, it updates the USTAT register. If another USB transfer is performed before a transaction complete interrupt is serviced, the SIE will store the status of the next transfer into the status FIFO.

Clearing the transfer complete flag bit, TRNIF, causes the SIE to advance the FIFO. If the next data in the FIFO holding register is valid, the SIE will reassert the interrupt within 5 TCY of clearing TRNIF. If no additional data is present, TRNIF will remain clear; USTAT data will no longer be reliable.

Note:	If an endpoint request is received while the USTAT FIFO is full, the SIE will automatically issue a NAK back to the host.

FIGURE 23-3: USTAT FIFO



REGISTER 23-3: USTAT: USB STATUS REGISTER (ACCESS F64h)

U-0	R-x	R-x	R-x	R-x	R-x	R-x	U-0
—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI(1)	—
bit 7							bit 0

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

bit 7 **Unimplemented:** Read as ‘0’

bit 6-3 **ENDP<3:0>:** Encoded Number of Last Endpoint Activity bits
(represents the number of the BDT updated by the last USB transfer)
1111 = Endpoint 15
1110 = Endpoint 14
.
.
.
0001 = Endpoint 1
0000 = Endpoint 0

bit 2 **DIR:** Last BD Direction Indicator bit
1 = The last transaction was an IN token
0 = The last transaction was an OUT or SETUP token

bit 1 **PPBI:** Ping-Pong BD Pointer Indicator bit(1)
1 = The last transaction was to the Odd BD bank
0 = The last transaction was to the Even BD bank

bit 0 **Unimplemented:** Read as ‘0’

Note 1: This bit is only valid for endpoints with available Even and Odd BD registers.

REGISTER 23-5: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), CPU MODE

R/W-x	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
UOWN(1)	DTS(2)	—(3)	—(3)	DTSEN	BSTALL	BC9	BC8
bit 7							bit 0

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

bit 7 **UOWN:** USB Own bit(1)
- 0 = The microcontroller core owns the BD and its corresponding buffer

bit 6 **DTS:** Data Toggle Synchronization bit(2)
- 1 = Data 1 packet
- 0 = Data 0 packet

bit 5-4 **Unimplemented:** These bits should always be programmed to '0'(3)

bit 3 **DTSEN:** Data Toggle Synchronization Enable bit
- 1 = Data toggle synchronization is enabled; data packets with incorrect Sync value will be ignored except for a SETUP transaction, which is accepted even if the data toggle bits do not match
- 0 = No data toggle synchronization is performed

bit 2 **BSTALL:** Buffer Stall Enable bit
- 1 = Buffer stall enabled; STALL handshake issued if a token is received that would use the BD in the given location (UOWN bit remains set, BD value is unchanged)
- 0 = Buffer stall disabled

bit 1-0 **BC<9:8>:** Byte Count 9 and 8 bits

The byte count bits represent the number of bytes that will be transmitted for an IN token or received during an OUT token. Together with BC<7:0>, the valid byte counts are 0-1023.

Note 1: This bit must be initialized by the user to the desired value prior to enabling the USB module.

2: This bit is ignored unless DTSEN = 1.

3: If these bits are set, USB communication may not work. Hence, these bits should always be maintained as '0'.

23.5.1 USB INTERRUPT STATUS REGISTER (UIR)

The USB Interrupt Status register (Register 23-7) contains the flag bits for each of the USB status interrupt sources. Each of these sources has a corresponding interrupt enable bit in the UIE register. All of the USB status flags are ORed together to generate the USBIF interrupt flag for the microcontroller's interrupt funnel.

Once an interrupt bit has been set by the SIE, it must be cleared in software by writing a '0'. The flag bits can also be set in software, which can aid in firmware debugging.

When the USB module is in the Low-Power Suspend mode (UCON<1> = 1), the SIE does not get clocked. When in this state, the SIE cannot process packets, and therefore, cannot detect new interrupt conditions other than the Activity Detect Interrupt, ACTVIF. The ACTVIF bit is typically used by USB firmware to detect when the microcontroller should bring the USB module out of the Low-Power Suspend mode (UCON<1> = 0).

REGISTER 23-7: UIR: USB INTERRUPT STATUS REGISTER (ACCESS F62h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0
—	SOFIF	STALLIF	IDLEIF(1)	TRNIF(2)	ACTVIF(3)	UERRIF(4)	URSTIF
bit 7							bit 0

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

bit 7 **Unimplemented:** Read as '0'

bit 6 **SOFIF:** Start-Of-Frame Token Interrupt bit
1 = A Start-Of-Frame token is received by the SIE
0 = No Start-Of-Frame token is received by the SIE

bit 5 **STALLIF:** A STALL Handshake Interrupt bit
1 = A STALL handshake was sent by the SIE
0 = A STALL handshake has not been sent

bit 4 **IDLEIF:** Idle Detect Interrupt bit(1)
1 = Idle condition is detected (constant Idle state of 3 ms or more)
0 = No Idle condition is detected

bit 3 **TRNIF:** Transaction Complete Interrupt bit(2)
1 = Processing of pending transaction is complete; read the USTAT register for endpoint information
0 = Processing of pending transaction is not complete or no transaction is pending

bit 2 **ACTVIF:** Bus Activity Detect Interrupt bit(3)
1 = Activity on the D+/D- lines was detected
0 = No activity detected on the D+/D- lines

bit 1 **UERRIF:** USB Error Condition Interrupt bit(4)
1 = An unmasked error condition has occurred
0 = No unmasked error condition has occurred.

bit 0 **URSTIF:** USB Reset Interrupt bit
1 = Valid USB Reset occurred; 00h is loaded into the UADDR register
0 = No USB Reset has occurred

Note 1: Once an Idle state is detected, the user may want to place the USB module in Suspend mode.
2: Clearing this bit will cause the USTAT FIFO to advance (valid only for IN, OUT and SETUP tokens).
3: This bit is typically unmasked only following the detection of a UIDLE interrupt event.
4: Only error conditions enabled through the UEIE register will set this bit. This bit is a status bit only and cannot be set or cleared by the user.

23.5.2 USB INTERRUPT ENABLE REGISTER (UIE)

The USB Interrupt Enable (UIE) register (Register 23-8) contains the enable bits for the USB status interrupt sources. Setting any of these bits will enable the respective interrupt source in the UIR register.

The values in this register only affect the propagation of an interrupt condition to the microcontroller's interrupt logic. The flag bits are still set by their interrupt conditions, allowing them to be polled and serviced without actually generating an interrupt.

REGISTER 23-8: UIE: USB INTERRUPT ENABLE REGISTER (BANKED F36h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE
bit 7							bit 0

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

bit 7 **Unimplemented:** Read as '0'

bit 6 **SOFIE:** Start-Of-Frame Token Interrupt Enable bit
1 = Start-Of-Frame token interrupt is enabled
0 = Start-Of-Frame token interrupt is disabled

bit 5 **STALLIE:** STALL Handshake Interrupt Enable bit
1 = STALL interrupt is enabled
0 = STALL interrupt is disabled

bit 4 **IDLEIE:** Idle Detect Interrupt Enable bit
1 = Idle detect interrupt is enabled
0 = Idle detect interrupt is disabled

bit 3 **TRNIE:** Transaction Complete Interrupt Enable bit
1 = Transaction interrupt is enabled
0 = Transaction interrupt is disabled

bit 2 **ACTVIE:** Bus Activity Detect Interrupt Enable bit
1 = Bus activity detect interrupt is enabled
0 = Bus activity detect interrupt is disabled

bit 1 **UERRIE:** USB Error Interrupt Enable bit
1 = USB error interrupt is enabled
0 = USB error interrupt is disabled

bit 0 **URSTIE:** USB Reset Interrupt Enable bit
1 = USB Reset interrupt is enabled
0 = USB Reset interrupt is disabled

27.9 Registers

There are three control registers for the CTMU:

- CTMUCONH
- CTMUCONL
- CTMUICON

The CTMUCONH and CTMUCONL registers (Register 27-1 and Register 27-2) contain control bits for configuring the CTMU module edge source selection, edge source polarity selection, edge sequencing, A/D trigger, analog circuit capacitor discharge and enables. The CTMUICON register (Register 27-3) has bits for selecting the current source range and current source trim.

REGISTER 27-1: CTMUCONH: CTMU CONTROL REGISTER HIGH (ACCESS FB3h)

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG
bit 7							bit 0

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

bit 7 **CTMUEN:** CTMU Enable bit
1 = Module is enabled
0 = Module is disabled

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **CTMUSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 4 **TGEN:** Time Generation Enable bit
1 = Enables edge delay generation
0 = Disables edge delay generation

bit 3 **EDGEN:** Edge Enable bit
1 = Edges are not blocked
0 = Edges are blocked

bit 2 **EDGSEQEN:** Edge Sequence Enable bit
1 = Edge 1 event must occur before Edge 2 event can occur
0 = No edge sequence is needed

bit 1 **IDISSEN:** Analog Current Source Control bit
1 = Analog current source output is grounded
0 = Analog current source output is not grounded

bit 0 **CTTRIG:** CTMU Special Event Trigger bit
1 = CTMU Special Event Trigger is enabled
0 = CTMU Special Event Trigger is disabled

28.2 Watchdog Timer (WDT)

PIC18F47J53 family devices have both a conventional WDT circuit and a dedicated, Deep Sleep capable Watchdog Timer. When enabled, the conventional WDT operates in normal Run, Idle and Sleep modes. This data sheet section describes the conventional WDT circuit.

The dedicated, Deep Sleep capable WDT can only be enabled in Deep Sleep mode. This timer is described in **Section 4.6.4 "Deep Sleep Watchdog Timer (DSWDT)"**.

The conventional WDT is driven by the INTRC oscillator. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by the WDTPS bits in Configuration Register 2H. Available periods range from about 4 ms to 135 seconds (2.25 minutes depending on voltage, temperature and WDT postscaler). The WDT and postscaler are cleared whenever a SLEEP or CLRWDT instruction is executed, or a clock failure (primary or Timer1 oscillator) has occurred.

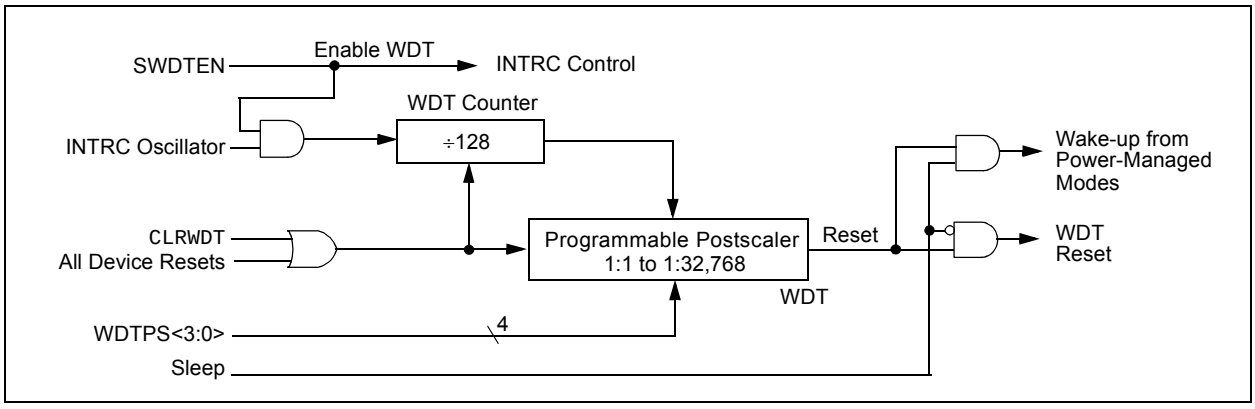
Note 1:	The CLRWDT and SLEEP instructions clear the WDT and postscaler counts when executed.
2:	When a CLRWDT instruction is executed, the postscaler count will be cleared.

28.2.1 CONTROL REGISTER

The WDTCON register (Register 28-11) is a readable and writable register. The SWDTEN bit enables or disables WDT operation. This allows software to override the WDTEN Configuration bit and enable the WDT only if it has been disabled by the Configuration bit.

LVDSTAT is a read-only status bit that is continuously updated and provides information about the current level of V_{DDCORE}. This bit is only valid when the on-chip voltage regulator is enabled.

FIGURE 28-1: WDT BLOCK DIAGRAM



RRNCF **Rotate Right f (No Carry)**

Syntax: RRNCF f {,d {,a}}

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

Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

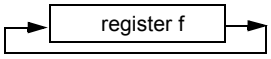
Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



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, 0
Before Instruction
    REG     =   1101 0111
After Instruction
    REG     =   1110 1011
```

Example 2:
```
RRNCF   REG, 0, 0
Before Instruction
    W       =   ?
    REG     =   1101 0111
After Instruction
    W       =   1110 1011
    REG     =   1101 0111
```

SETF **Set f**

Syntax: SETF f {,a}

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

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example:
```
SETF    REG,1
Before Instruction
    REG     =   5Ah
After Instruction
    REG     =   FFh
```

31.2 DC Characteristics: Power-Down and Supply Current PIC18F47J53 Family (Industrial)

PIC18LF47J53 Family		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F47J53 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param. No.	**Device**	**Typ.**	**Max.**	**Units**	**Conditions**		
	Power-Down Current (IPD)(1) – Sleep mode						
	PIC18LFXXJ53	0.1	1.6	µA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	**Sleep mode**, REGSLP = 1
		0.2	1.6	µA	+25°C		
		0.8	7.0	µA	+60°C		
		2.1	11.5	µA	+85°C		
	PIC18LFXXJ53	0.2	2.0	µA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		0.5	2.0	µA	+25°C		
		1.4	9.0	µA	+60°C		
		3.2	15.0	µA	+85°C		
	PIC18FXXJ53	3.0	6.0	µA	-40°C	VDD = 2.15V Vddcore = 10 µF Capacitor	
		3.8	6.0	µA	+25°C		
		4.7	9.0	µA	+60°C		
		6.4	18.5	µA	+85°C		
	PIC18FXXJ53	3.3	9.0	µA	-40°C	VDD = 3.3V Vddcore = 10 µF Capacitor	
		4.2	9.0	µA	+25°C		
		5.5	12.0	µA	+60°C		
		7.8	22.0	µA	+85°C		
	Power-Down Current (IPD)(1) – Deep Sleep mode						
	PIC18FXXJ53	2	25	nA	-40°C	VDD = 2.15V, VDDCORE = 10 µF Capacitor	**Deep Sleep** mode
		9	100	nA	+25°C		
		72	250	nA	+60°C		
		0.26	1.0	µA	+85°C		
	PIC18FXXJ53	17	50	nA	-40°C	VDD = 3.3V, VDDCORE = 10 µF Capacitor	
		53	150	nA	+25°C		
		186	400	nA	+60°C		
		0.50	2.0	µA	+85°C		

Note 1: 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 high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption. All features that add delta current are disabled (USB module, WDT, etc.). 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/VSS;
$\overline{MCLR}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

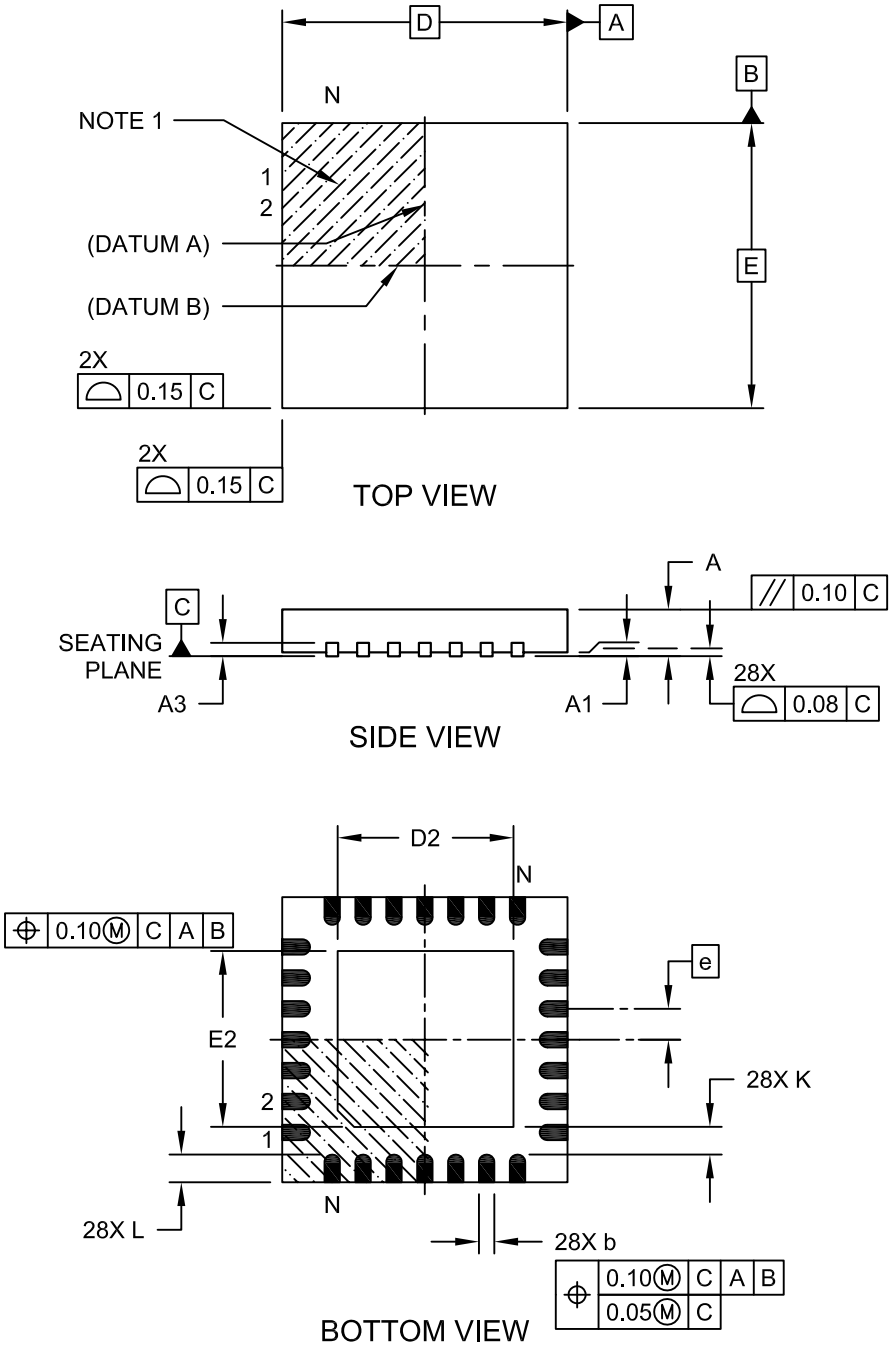
4: This is the module differential current when the USB module is enabled and clocked at 48 MHz, but with no USB cable attached. When the USB cable is attached or data is being transmitted, the current consumption may be much higher (see **Section 23.6.4 "USB Transceiver Current Consumption"**). During USB Suspend mode (USBEN = 1, SUSPND = 1, bus in Idle state), the USB module current will be dominated by the D+ or D- pull-up resistor. The integrated pull-up resistors use "resistor switching" according to the `resistor_ecn` supplement to the USB 2.0 Specifications, and therefore, may be as low as 900Ω during Idle conditions.

32.2 Package Details

The following sections give the technical details of the packages.

28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN] With 0.55 mm Terminal Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-105C Sheet 1 of 2