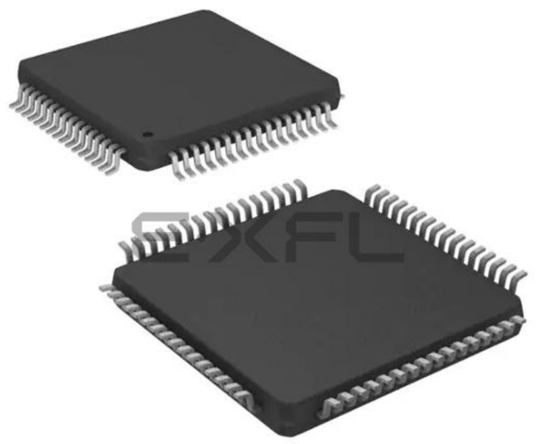
E·XFL

Microchip Technology - PIC18F67J11-I/PT 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	Active
Core Processor	PIC
Core Size	8-Bit
Speed	48MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3930 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f67j11-i-pt

6.3 Data Memory Organization

Note: The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 "Data Memory and the Extended Instruction Set"** for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. The PIC18F87J11 family implements all available banks and provide 3936 bytes of data memory available to the user. Figure 6-7 shows the data memory organization for the devices.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this section.

To ensure that commonly used registers (select SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to select SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 6.3.2 "Access Bank"** provides a detailed description of the Access RAM.

6.3.1 BANK SELECT REGISTER

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the 4 Most Significant bits of a location's address. The instruction itself includes the 8 Least Significant bits. Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the `MOVLB` instruction.

The value of the BSR indicates the bank in data memory. The 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 6-8.

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh, will end up resetting the Program Counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory map in Figure 6-7 indicates which banks are implemented.

In the core PIC18 instruction set, only the `MOVFF` instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

© 2007-2012 Microchip Technology Inc.

6.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy more than the top half of Bank 15 (F5Ah to FFFh). A list of these registers is given inTable 6-3, Table 6-4 and Table 6-5.

The SFRs can be classified into two sets: those associated with the "core" device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and Interrupt registers are described in their respective chapters, while the ALU's STATUS register is described later in this section. Registers related to the operation of the peripheral features are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as '0's

Note: Addresses, F5Ah through F5Fh, are not part of the Access Bank. These registers must always be accessed using the Bank Select Register. Addresses, F40h to F59h, are not implemented and are not accessible to the user.

TABLE 6-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18F87J11 FAMILY DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2[(1)]	FBFh	ECCP1AS	F9Fh	IPR1	F7Fh	SPBRGH1	F5Fh	PMDIN2H
FFEh	TOSH	FDEh	POSTINC2[(1)]	FBEh	ECCP1DEL	F9Eh	PIR1	F7Eh	BAUDCON1	F5Eh	PMDIN2L
FFDh	TOSL	FDDh	POSTDEC2[(1)]	FBDh	CCPR1H	F9Dh	PIE1	F7Dh	SPBRGH2	F5Dh	PMEH
FFCh	STKPTR	FDCh	PREINC2[(1)]	FBCh	CCPR1L	F9Ch	RCSTA2	F7Ch	BAUDCON2	F5Ch	PMEL
FFBh	PCLATU	FDBh	PLUSW2[(1)]	FBBh	CCP1CON	F9Bh	OSCTUNE	F7Bh	TMR3H	F5Bh	PMSTATH
FFAh	PCLATH	FDAh	FSR2H	FBAh	ECCP2AS	F9Ah	TRISJ[(2)]	F7Ah	TMR3L	F5Ah	PMSTATL
FF9h	PCL	FD9h	FSR2L	FB9h	ECCP2DEL	F99h	TRISH[(2)]	F79h	T3CON	F59h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	CCPR2H	F98h	TRISG	F78h	TMR4	F58h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	CCPR2L	F97h	TRISF	F77h	PR4[(3)]	F57h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	CCP2CON	F96h	TRISE	F76h	T4CON	F56h	—
FF5h	TABLAT	FD5h	T0CON	FB5h	ECCP3AS	F95h	TRISD	F75h	CCPR4H	F55h	—
FF4h	PRODH	FD4h	—	FB4h	ECCP3DEL	F94h	TRISC	F74h	CCPR4L	F54h	—
FF3h	PRODL	FD3h	OSCCON[(3)]	FB3h	CCPR3H	F93h	TRISB	F73h	CCP4CON	F53h	—
FF2h	INTCON	FD2h	CM1CON	FB2h	CCPR3L	F92h	TRISA	F72h	CCPR5H	F52h	—
FF1h	INTCON2	FD1h	CM2CON	FB1h	CCP3CON	F91h	LATJ[(2)]	F71h	CCPR5L	F51h	—
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRG1	F90h	LATH[(2)]	F70h	CCP5CON	F50h	—
FEFh	INDF0[(1)]	FCFh	TMR1H[(3)]	FAFh	RCREG1	F8Fh	LATG	F6Fh	SSP2BUF	F4Fh	—
FEEh	POSTINC0[(1)]	FCEh	TMR1L[(3)]	FAEh	TXREG1	F8Eh	LATF	F6Eh	SSP2ADD	F4Eh	—
FEDh	POSTDEC0[(1)]	FCDh	T1CON[(3)]	FADh	TXSTA1	F8Dh	LATE	F6Dh	SSP2STAT	F4Dh	—
FECh	PREINC0[(1)]	FCCh	TMR2[(3)]	FACh	RCSTA1	F8Ch	LATD	F6Ch	SSP2CON1	F4Ch	—
FEBh	PLUSW0[(1)]	FCBh	PR2[(3)]	FABh	SPBRG2	F8Bh	LATC	F6Bh	SSP2CON2	F4Bh	—
FEAh	FSR0H	FCAh	T2CON	FAAh	RCREG2	F8Ah	LATB	F6Ah	CMSTAT	F4Ah	—
FE9h	FSR0L	FC9h	SSP1BUF	FA9h	TXREG2	F89h	LATA	F69h	PMADDRH[(4)]	F49h	—
FE8h	WREG	FC8h	SSP1ADD	FA8h	TXSTA2	F88h	PORTJ[(2)]	F68h	PMADDRL[(4)]	F48h	—
FE7h	INDF1[(1)]	FC7h	SSP1STAT	FA7h	EECON2	F87h	PORTH[(2)]	F67h	PMDIN1H	F47h	—
FE6h	POSTINC1[(1)]	FC6h	SSP1CON1	FA6h	EECON1	F86h	PORTG	F66h	PMDIN1L	F46h	—
FE5h	POSTDEC1[(1)]	FC5h	SSP1CON2	FA5h	IPR3	F85h	PORTF	F65h	PMCONH	F45h	—
FE4h	PREINC1[(1)]	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE	F64h	PMCONL	F44h	—
FE3h	PLUSW1[(1)]	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD	F63h	PMMODEH	F43h	—
FE2h	FSR1H	FC2h	ADCON0[(3)]	FA2h	IPR2	F82h	PORTC	F62h	PMMODEL	F42h	—
FE1h	FSR1L	FC1h	ADCON1[(3)]	FA1h	PIR2	F81h	PORTB	F61h	PMDOUT2H	F41h	—
FE0h	BSR	FC0h	WDTCON	FA0h	PIE2	F80h	PORTA	F60h	PMDOUT2L	F40h	—

Note 1: This is not a physical register.
2: This register is not available on 64-pin devices.
3: This register shares the same address with another register (see Table 6-4 for alternate register).
4: The PMADDRH/L and PMDOUT1H/L register pairs share the same address. PMADDR is used in Master modes and PMDOUT1 is used in Slave modes.
5: Addresses, F40 to F59, are not implemented and are not accessible to the user.

REGISTER 6-3: WDTCON: WATCHDOG TIMER CONTROL REGISTER

R/W-0	R-x	U-0	R/W-0	U-0	U-0	U-0	U-0
REGSLP	LVDSTAT	—	ADSHR	—	—	—	SWDTEN
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	**REGSLP:** Voltage Regulator Low-Power Operation Enable bit For details of bit operation, see Register 25-9.
bit 6	**LVDSTAT:** LVD Status bit `1` = VDDCORE > 2.45V `0` = VDDCORE < 2.45V
bit 5	**Unimplemented**: Read as ‘`0`’
bit 4	**ADSHR:** Shared Address SFR Select bit `1` = Alternate SFR is selected `0` = Default (Legacy) SFR is selected
bit 3-1	**Unimplemented**: Read as ‘`0`’
bit 0	**SWDTEN:** Software Controlled Watchdog Timer Enable bit For details of bit operation, see Register 25-9.

7.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 64 bytes at a time or two bytes at a time. Program memory is erased in blocks of 1024 bytes at a time. A bulk erase operation may not be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a `NOP`.

7.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (`TBLRD`)
- Table Write (`TBLWT`)

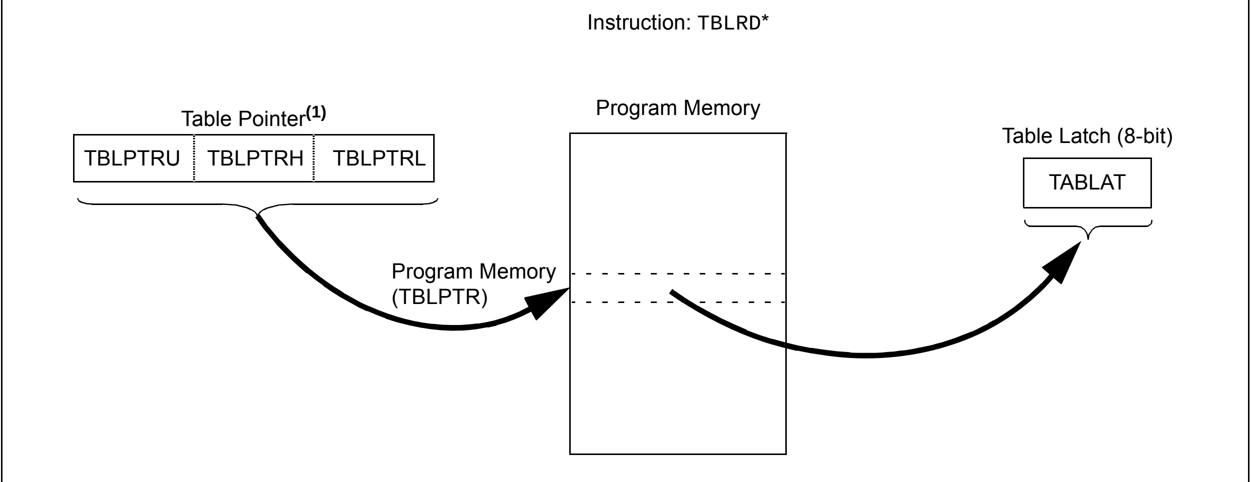
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 7.5 "Writing to Flash Program Memory"**. Figure 7-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word-aligned.

FIGURE 7-1: TABLE READ OPERATION



Note 1: Table Pointer register points to a byte in program memory.

7.4 Erasing Flash Program Memory

The minimum erase block is 512 words or 1024 bytes. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the microcontroller itself, a block of 1024 bytes of program memory is erased. The Most Significant 12 bits of the TBLPTR<21:10> point to the block being erased. TBLPTR<9:0> are ignored.

The EECON1 register commands the erase operation. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation. For protection, the write initiate sequence for EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load Table Pointer register with address of row being erased.
2. Set the WREN and FREE bits (EECON1<2,4>) to enable the erase operation.
3. Disable interrupts.
4. Write H'55' to EECON2.
5. Write H'AA' to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase for TIW (see Parameter D133A).
8. Re-enable interrupts.

EXAMPLE 7-2: ERASING A FLASH PROGRAM MEMORY ROW

```
                      MOVLW   CODE_ADDR_UPPER        ; load TBLPTR with the base
                      MOVWF   TBLPTRU                ; address of the memory block
                      MOVLW   CODE_ADDR_HIGH
                      MOVWF   TBLPTRH
                      MOVLW   CODE_ADDR_LOW
                      MOVWF   TBLPTRL
         ERASE_ROW
                      BSF     EECON1, WREN           ; enable write to memory
                      BSF     EECON1, FREE           ; enable Row Erase operation
                      BCF     INTCON, GIE            ; disable interrupts
     Required         MOVLW   H'55'
     Sequence         MOVWF   EECON2                 ; write H'55'
                      MOVLW   H'AA'
                      MOVWF   EECON2                 ; write H'AA'
                      BSF     EECON1, WR             ; start erase (CPU stall)
                      BSF     INTCON, GIE            ; re-enable interrupts
```

© 2007-2012 Microchip Technology Inc.

8.6.2 16-BIT WORD WRITE MODE

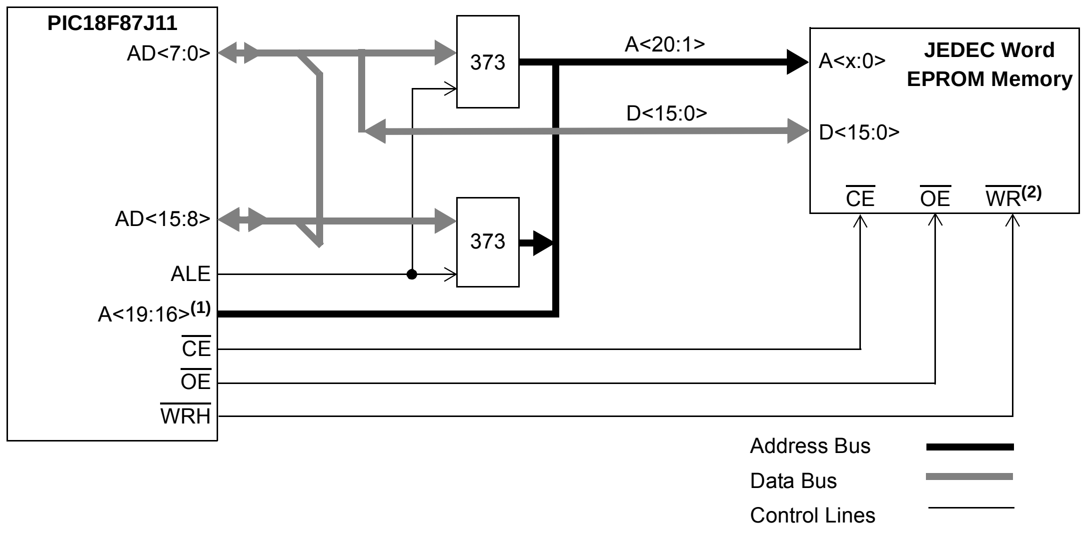
Figure 8-2 shows an example of 16-Bit Word Write mode for PIC18F87J11 family devices. This mode is used for word-wide memories which include some of the EPROM and Flash type memories. This mode allows opcode fetches and table reads from all forms of 16-bit memory and table writes to any type of word-wide external memories. This method makes a distinction between `TBLWT` cycles to even or odd addresses.

During a `TBLWT` cycle to an even address (TBLPTR<0> = `0`), the TABLAT data is transferred to a holding latch and the external address data bus is tri-stated for the data portion of the bus cycle. No write signals are activated.

During a `TBLWT` cycle to an odd address (TBLPTR<0> = `1`), the TABLAT data is presented on the upper byte of the AD<15:0> bus. The contents of the holding latch are presented on the lower byte of the AD<15:0> bus.

The $\overline{\text{WRH}}$ signal is strobed for each write cycle; the $\overline{\text{WRL}}$ pin is unused. The signal on the BA0 pin indicates the LSb of the TBLPTR, but it is left unconnected. Instead, the $\overline{\text{UB}}$ and $\overline{\text{LB}}$ signals are active to select both bytes. The obvious limitation to this method is that the table write must be done in pairs on a specific word boundary to correctly write a word location.

FIGURE 8-2: 16-BIT WORD WRITE MODE EXAMPLE



Note 1: Upper order address lines are used only for 20-bit address widths.

2: This signal only applies to table writes. See **Section 7.1 "Table Reads and Table Writes"**.

© 2007-2012 Microchip Technology Inc.

10.5 RCON Register

The RCON register contains bits used to determine the cause of the last Reset or wake-up from Idle or Sleep modes. RCON also contains the bit that enables interrupt priorities (IPEN).

REGISTER 10-13: RCON: RESET CONTROL REGISTER

R/W-0	U-0	R/W-1	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
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	**IPEN:** Interrupt Priority Enable bit `1` = Enable priority levels on interrupts `0` = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
bit 6	**Unimplemented:** Read as ‘`0`’
bit 5	**$\overline{\text{CM}}$:** Configuration Mismatch Flag bit For details of bit operation, see Register 5-1.
bit 4	**$\overline{\text{RI}}$:** RESET Instruction Flag bit For details of bit operation, see Register 5-1.
bit 3	**$\overline{\text{TO}}$:** Watchdog Timer Time-out Flag bit For details of bit operation, see Register 5-1.
bit 2	**$\overline{\text{PD}}$:** Power-Down Detection Flag bit For details of bit operation, see Register 5-1.
bit 1	**$\overline{\text{POR}}$:** Power-on Reset Status bit For details of bit operation, see Register 5-1.
bit 0	**$\overline{\text{BOR}}$:** Brown-out Reset Status bit For details of bit operation, see Register 5-1.

TABLE 11-20: PORTJ FUNCTIONS

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RJ0/ALE	RJ0	`0`	O	DIG	LATJ<0> data output.
		`1`	I	ST	PORTJ<0> data input.
	ALE	x	O	DIG	External Memory Interface address latch enable control output; takes priority over digital I/O.
RJ1/$\overline{\text{OE}}$	RJ1	`0`	O	DIG	LATJ<1> data output.
		`1`	I	ST	PORTJ<1> data input.
	$\overline{\text{OE}}$	x	O	DIG	External Memory Interface output enable control output; takes priority over digital I/O.
RJ2/$\overline{\text{WRL}}$	RJ2	`0`	O	DIG	LATJ<2> data output.
		`1`	I	ST	PORTJ<2> data input.
	$\overline{\text{WRL}}$	x	O	DIG	External Memory Bus write low byte control; takes priority over digital I/O.
RJ3/$\overline{\text{WRH}}$	RJ3	`0`	O	DIG	LATJ<3> data output.
		`1`	I	ST	PORTJ<3> data input.
	$\overline{\text{WRH}}$	x	O	DIG	External Memory Interface write high byte control output; takes priority over digital I/O.
RJ4/BA0	RJ4	`0`	O	DIG	LATJ<4> data output.
		`1`	I	ST	PORTJ<4> data input.
	BA0	x	O	DIG	External Memory Interface Byte Address 0 control output; takes priority over digital I/O.
RJ5/$\overline{\text{CE}}$	RJ5	`0`	O	DIG	LATJ<5> data output.
		`1`	I	ST	PORTJ<5> data input.
	$\overline{\text{CE}}$	x	O	DIG	External Memory Interface chip enable control output; takes priority over digital I/O.
RJ6/$\overline{\text{LB}}$	RJ6	`0`	O	DIG	LATJ<6> data output.
		`1`	I	ST	PORTJ<6> data input.
	$\overline{\text{LB}}$	x	O	DIG	External Memory Interface lower byte enable control output; takes priority over digital I/O.
RJ7/$\overline{\text{UB}}$	RJ7	`0`	O	DIG	LATJ<7> data output.
		`1`	I	ST	PORTJ<7> data input.
	$\overline{\text{UB}}$	x	O	DIG	External Memory Interface upper byte enable control output; takes priority over digital I/O.

Legend: O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input,
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

TABLE 11-21: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
PORTJ[1]	RJ7	RJ6	RJ5	RJ4	RJ3	RJ2	RJ1	RJ0	65
LATJ[1]	LATJ7	LATJ6	LATJ5	LATJ4	LATJ3	LATJ2	LATJ1	LATJ0	64
TRISJ[1]	TRISJ7	TRISJ6	TRISJ5	TRISJ4	TRISJ3	TRISJ2	TRISJ1	TRISJ0	64
PORTG	RDPU	REPU	RJPU[1]	RG4	RG3	RG2	RG1	RG0	65

Legend: Shaded cells are not used by PORTJ.

Note 1: Unimplemented on 64-pin devices, read as '`0`'.

NOTES:

© 2007-2012 Microchip Technology Inc.

REGISTER 12-6: PMEL: PARALLEL PORT ENABLE LOW BYTE REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTEN7	PTEN6	PTEN5	PTEN4	PTEN3	PTEN2	PTEN1	PTEN0
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 7-2 **PTEN<7:2>:** PMP Address Port Enable bits
1 = PMA<7:2> function as PMP address lines
0 = PMA<7:2> function as port I/O

bit 1-0 **PTEN<1:0>:** PMALH/PMALL Strobe Enable bits
1 = PMA1 and PMA0 function as either PMA<1:0> or PMALH and PMALL
0 = PMA1 and PMA0 pads function as port I/O

REGISTER 12-7: PMSTATH: PARALLEL PORT STATUS HIGH BYTE REGISTER

R-0	R/W-0	U-0	U-0	R-0	R-0	R-0	R-0
IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F
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 **IBF:** Input Buffer Full Status bit
1 = All writable input buffer registers are full
0 = Some or all of the writable input buffer registers are empty

bit 6 **IBOV:** Input Buffer Overflow Status bit
1 = A write attempt to a full input byte register occurred (must be cleared in software)
0 = No overflow occurred

bit 5-4 **Unimplemented:** Read as ‘0’

bit 3-0 **IB3F:IB0F:** Input Buffer Status Full bits
1 = Input buffer contains data that has not been read (reading buffer will clear this bit)
0 = Input buffer does not contain any unread data

© 2007-2012 Microchip Technology Inc.

NOTES:

20.3.7 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCKx. When the last bit is latched, the SSPxIF interrupt flag bit is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCKx pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device can be configured to wake-up from Sleep.

20.3.8 SLAVE SELECT SYNCHRONIZATION

The $\overline{\text{SSx}}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with the $\overline{\text{SSx}}$ pin control enabled (SSPxCON1<3:0> = 04h). When the $\overline{\text{SSx}}$ pin is low, transmission and reception are enabled and the SDOx pin is driven. When the $\overline{\text{SSx}}$ pin goes high, the SDOx pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

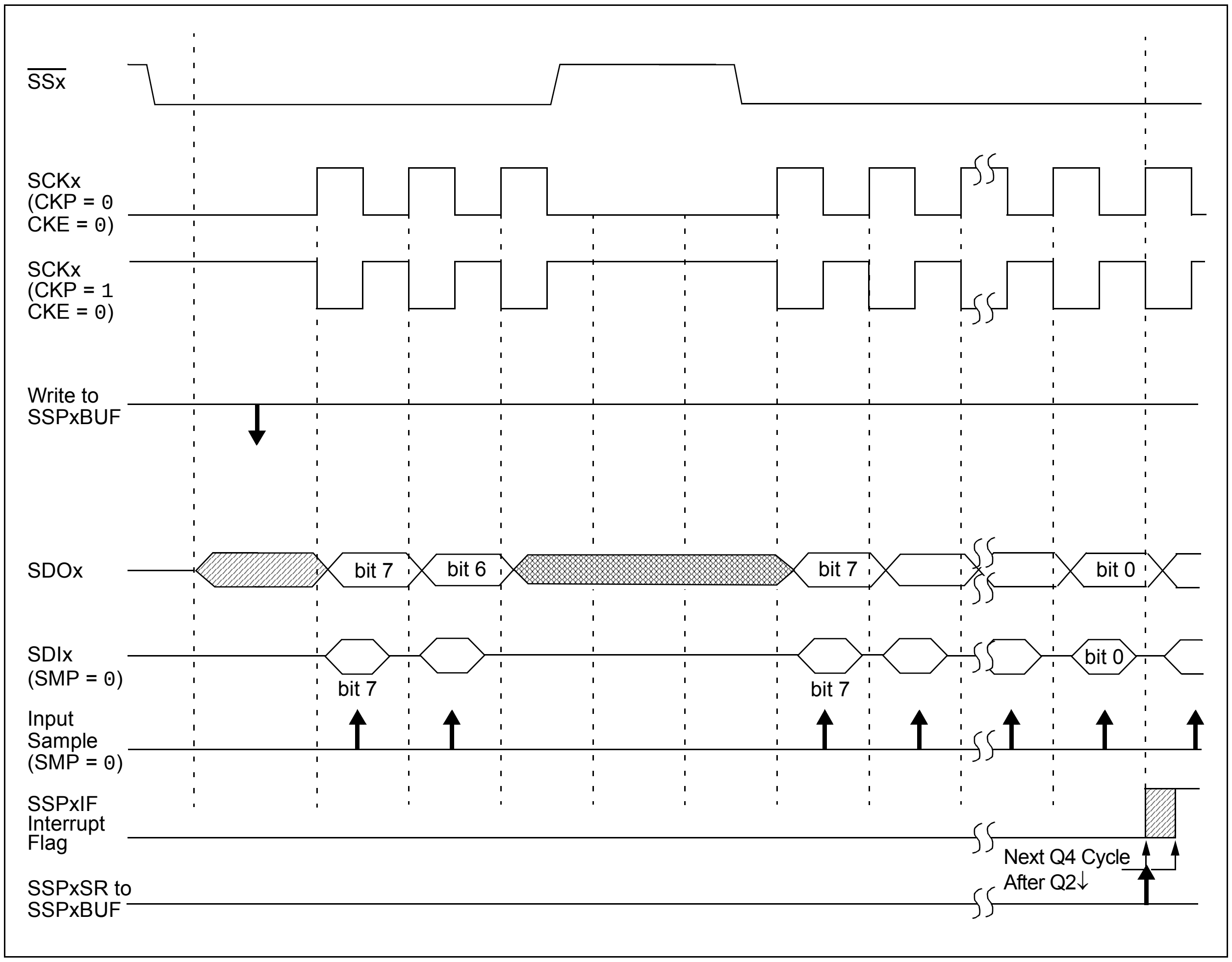
Note 1: When the SPI is in Slave mode, with the $\overline{\text{SSx}}$ pin control enabled, (SSPxCON1<3:0> = `0100`), the SPI module will reset if the $\overline{\text{SSx}}$ pin is set to VDD.

2: If the SPI is used in Slave mode, with CKE set, then the $\overline{\text{SSx}}$ pin control must be enabled.

When the SPI module resets, the bit counter is forced to '`0`'. This can be done by either forcing the $\overline{\text{SSx}}$ pin to a high level or clearing the SSPEN bit.

To emulate two-wire communication, the SDOx pin can be connected to the SDIx pin. When the SPI needs to operate as a receiver, the SDOx pin can be configured as an input; this disables transmissions from the SDOx. The SDIx can always be left as an input (SDI function) since it cannot create a bus conflict.

FIGURE 20-4: SLAVE SYNCHRONIZATION WAVEFORM



© 2007-2012 Microchip Technology Inc.

20.4.14 SLEEP OPERATION

While in Sleep mode, the I^2C module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

20.4.15 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

20.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit (SSPxSTAT<4>) is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the MSSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed in hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

20.4.17 MULTI -MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '1' on SDAx, by letting SDAx float high, and another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin = 0, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF, and reset the I^2C port to its Idle state (Figure 20-27).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I^2C bus is free, the user can resume communication by asserting a Start condition.

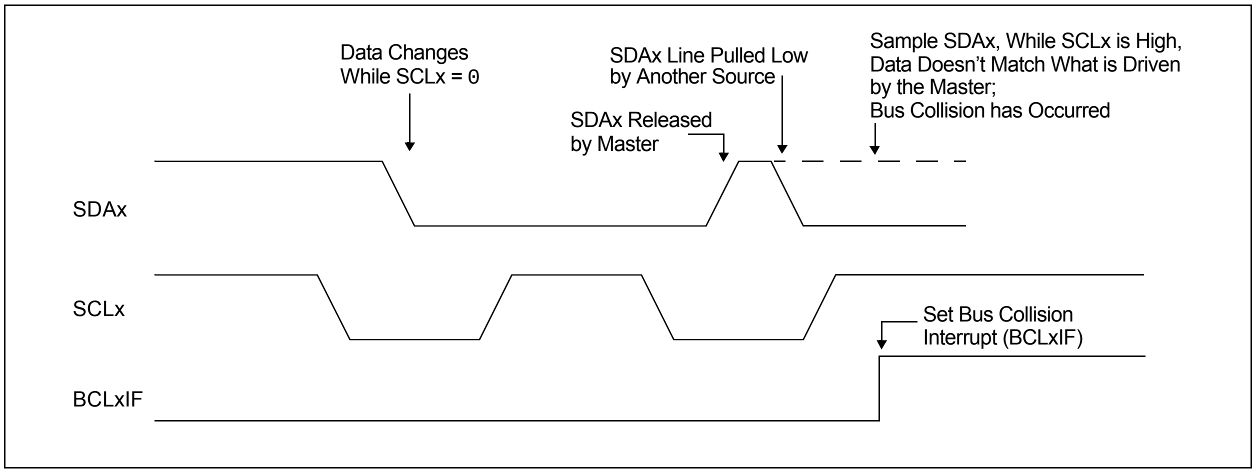
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the I^2C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set.

A write to the SSPxBUF will start the transmission of data at the first data bit regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I^2C bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is Idle and the S and P bits are cleared.

FIGURE 20-27: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE


Data Changes While SCLx = 0
SDAx Line Pulled Low by Another Source
Sample SDAx, While SCLx is High, Data Doesn't Match What is Driven by the Master; Bus Collision has Occurred
SDAx Released by Master
SDAx
SCLx
Set Bus Collision Interrupt (BCLxIF)
BCLxIF

© 2007-2012 Microchip Technology Inc.

CLRF — Clear f

Syntax:	CLRF f {,a}
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	$000h \rightarrow f$, $1 \rightarrow Z$
Status Affected:	Z
Encoding:	`0110` | `101a` | `ffff` | `ffff`
Description:	Clears the contents of the specified register. If 'a' is '`0`', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 26.2.3 "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: `CLRF FLAG_REG,1`

Before Instruction

FLAG_REG	=	5Ah

After Instruction

FLAG_REG	=	00h

CLRWDT — Clear Watchdog Timer

Syntax:	CLRWDT
Operands:	None
Operation:	$000h \rightarrow$ WDT, $000h \rightarrow$ WDT postscaler, $1 \rightarrow \overline{TO}$, $1 \rightarrow \overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	`0000` | `0000` | `0000` | `0100`
Description:	`CLRWDT` instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: `CLRWDT`

Before Instruction

WDT Counter	=	?

After Instruction

WDT Counter	=	00h
WDT Postscaler	=	0
$\overline{TO}$	=	1
$\overline{PD}$	=	1

DAW — Decimal Adjust W Register

Syntax:	DAW
Operands:	None
Operation:	If [W<3:0> > 9] or [DC = 1] then, (W<3:0>) + 6 → W<3:0>; else, (W<3:0>) → W<3:0> If [W<7:4> > 9] or [C = 1] then, (W<7:4>) + 6 → W<7:4>, C = 1; else, (W<7:4>) → W<7:4>
Status Affected:	C
Encoding:	0000 0000 0000 0111
Description:	`DAW` adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example 1: `DAW`

```
Before Instruction
    W    =    A5h
    C    =    0
    DC   =    0
After Instruction
    W    =    05h
    C    =    1
    DC   =    0
```

Example 2:

```
Before Instruction
    W    =    CEh
    C    =    0
    DC   =    0
After Instruction
    W    =    34h
    C    =    1
    DC   =    0
```

DECF — Decrement f

Syntax:	DECF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – 1 → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	0000 01da ffff ffff
Description:	Decrement register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’. If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank. If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 “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: `DECF    CNT,   1, 0`

```
Before Instruction
    CNT   =    01h
    Z     =    0
After Instruction
    CNT   =    00h
    Z     =    1
```

© 2007-2012 Microchip Technology Inc.

SUBFSR	Subtract Literal from FSR
Syntax:	SUBFSR f, k
Operands:	0 £ k £ 63 f Î [0, 1, 2]
Operation:	FSRf – k ® FSRf
Status Affected:	None
Encoding:	`1110` `1001` `ffkk` `kkkk`
Description:	The 6-bit literal ‘k’ is subtracted from the contents of the FSR specified by ‘f’.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

Example: `SUBFSR 2, 23h`

Before Instruction

FSR2	=	03FFh

After Instruction

FSR2	=	03DCh

SUBULNK	Subtract Literal from FSR2 and Return
Syntax:	SUBULNK k
Operands:	0 £ k £ 63
Operation:	FSR2 – k ® FSR2, (TOS) → PC
Status Affected:	None
Encoding:	`1110` `1001` `11kk` `kkkk`
Description:	The 6-bit literal ‘k’ is subtracted from the contents of the FSR2. A `RETURN` is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a `NOP` is performed during the second cycle. This may be thought of as a special case of the `SUBFSR` instruction, where f = 3 (binary ‘`11`’); it operates only on FSR2.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination
No Operation	No Operation	No Operation	No Operation

Example: SUBULNK 23h

Before Instruction

FSR2	=	03FFh
PC	=	0100h

After Instruction

FSR2	=	03DCh
PC	=	(TOS)

27.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

27.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, preprocessor, and one-step driver, and can run on multiple platforms.

27.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

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, absolute LST files that contain source lines and generated machine code and COFF files 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

27.5 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

27.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

© 2007-2012 Microchip Technology Inc.

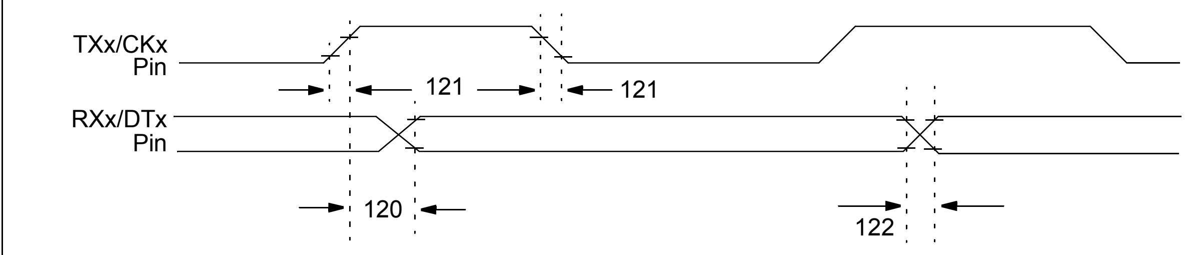
28.1 DC Characteristics: Supply Voltage PIC18F87J11 Family (Industrial)

PIC18F87J11 Family (Industrial)			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
D001	VDD	**Supply Voltage**	VDDCORE 2.0	— —	3.6 3.6	V V	ENVREG tied to VSS ENVREG tied to VDD
D001B	VDDCORE	**External Supply for Microcontroller Core**	2.0	—	2.7	V	ENVREG tied to VSS
D001C	AVDD	**Analog Supply Voltage**	VDD – 0.3	—	VDD + 0.3	V	
D001D	AVSS	**Analog Ground Potential**	VSS – 0.3	—	VSS + 0.3	V	
D002	VDR	**RAM Data Retention Voltage**(1)	1.5	—	—	V	
D003	VPOR	**VDD Power-on Reset Voltage**	—	—	0.7	V	See **Section 5.3 “Power-on Reset (POR)”** for details
D004	SVDD	**VDD Rise Rate** to Ensure Internal Power-on Reset Signal	0.05	—	—	V/ms	See **Section 5.3 “Power-on Reset (POR)”** for details
D005	VBOR	**Brown-out Reset Voltage**	1.75(2)	2.0	2.4	V	

Note 1: This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

2: When the Brown-out Reset is enabled, the part will continue to operate until the BOR occurs. This is valid, although VDD may be below the minimum VDD voltage.

FIGURE 28-22: EUSARTx SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

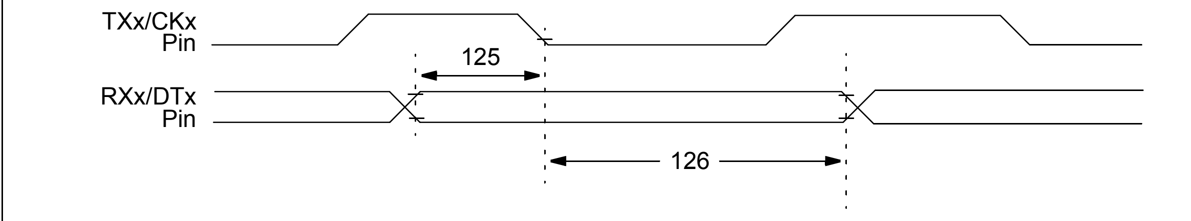


Note: Refer to Figure 28-3 for load conditions.

TABLE 28-28: EUSARTx SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	TCKH2DTV	<u>SYNC XMIT (MASTER and SLAVE)</u> Clock High to Data Out Valid	—	40	ns	
121	TCKRF	Clock Out Rise Time and Fall Time (Master mode)	—	20	ns	
122	TDTRF	Data Out Rise Time and Fall Time	—	20	ns	

FIGURE 28-23: EUSARTx SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 28-3 for load conditions.

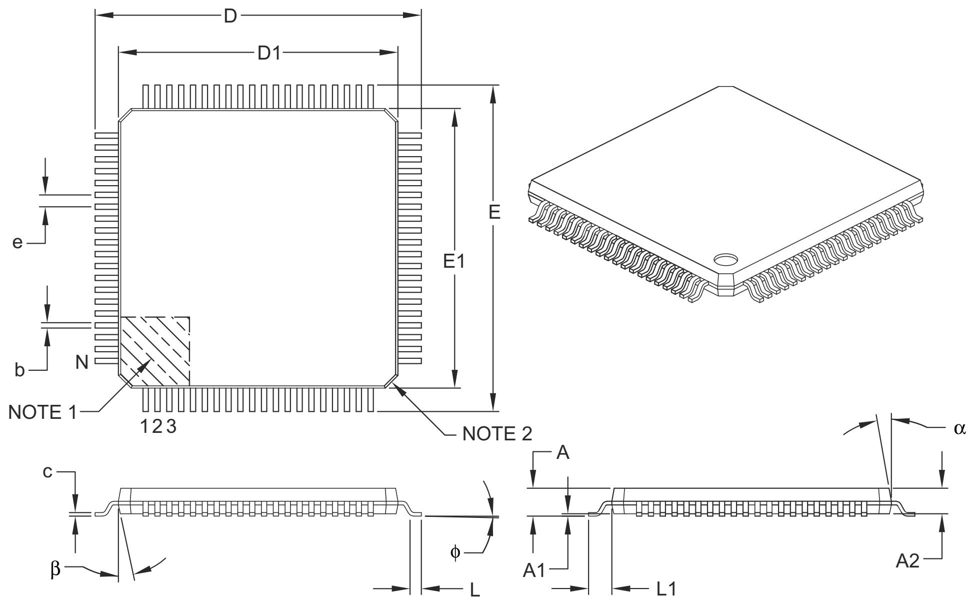
TABLE 28-29: EUSARTx SYNCHRONOUS RECEIVE REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TDTV2CKL	<u>SYNC RCV (MASTER and SLAVE)</u> Data Hold Before CKx ↓ (DTx hold time)	10	—	ns	
126	TCKL2DTL	Data Hold After CKx ↓ (DTx hold time)	15	—	ns	

© 2007-2012 Microchip Technology Inc.

80-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 mm Body, 2.00 mm [TQFP]

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



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Leads	N	80		
Lead Pitch	e	0.50 BSC		
Overall Height	A	–	–	1.20
Molded Package Thickness	A2	0.95	1.00	1.05
Standoff	A1	0.05	–	0.15
Foot Length	L	0.45	0.60	0.75
Footprint	L1	1.00 REF		
Foot Angle	φ	0°	3.5°	7°
Overall Width	E	14.00 BSC		
Overall Length	D	14.00 BSC		
Molded Package Width	E1	12.00 BSC		
Molded Package Length	D1	12.00 BSC		
Lead Thickness	c	0.09	–	0.20
Lead Width	b	0.17	0.22	0.27
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-092B

© 2007-2012 Microchip Technology Inc.