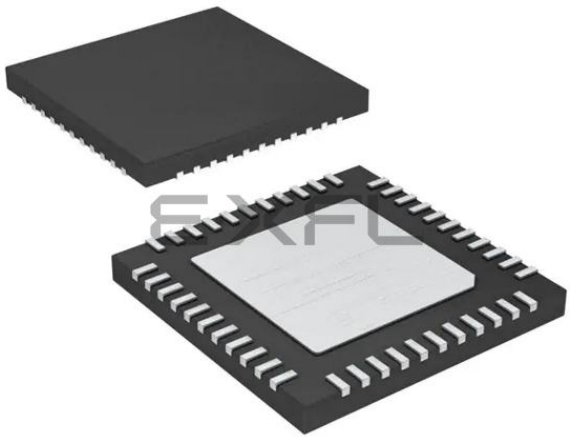
Microchip Technology - PIC18LF46J13-I/ML Datasheet



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, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	34
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 2.75V
Data Converters	A/D 13x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf46j13-i-ml

Table of Contents

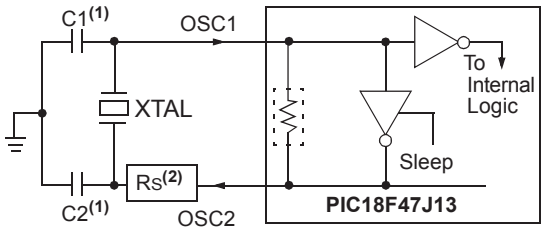
3.2.2 CRYSTAL OSCILLATOR/CERAMIC RESONATORS

In HS and HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 3-2 displays the pin connections.

The oscillator design requires the use of a parallel resonant crystal.

Note:	Use of a series resonant crystal may give a frequency out of the crystal manufacturer's specifications.

FIGURE 3-2: CRYSTAL/CERAMIC RESONATOR OPERATION (HS OR HSPLL CONFIGURATION)



Note 1: See Table 3-2 and Table 3-3 for initial values of C1 and C2.

2: RS may be required to avoid overdriving crystals with low drive level specifications.

TABLE 3-2: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Typical Capacitor Values Used:			
Mode	**Freq**	**OSC1**	**OSC2**
HS	8.0 MHz	27 pF	27 pF
	16.0 MHz	22 pF	22 pF
Capacitor values are for design guidance only. These capacitors were tested with the resonators listed below for basic start-up and operation. **These values are not optimized**. Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application. See the notes following Table 3-3 for additional information.			
Resonators Used:			
4.0 MHz			
8.0 MHz			
16.0 MHz			

TABLE 3-3: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	**C2**
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	16 MHz	18 pF	18 pF
Capacitor values are for design guidance only. These capacitors were tested with the crystals listed below for basic start-up and operation. **These values are not optimized.** Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application. See the notes following this table for additional information.			
Crystals Used:			
4 MHz			
8 MHz			
16 MHz			

Note 1:	Higher capacitance not only increases the stability of oscillator, but also increases the start-up time.
2:	Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
3:	Rs may be required to avoid overdriving crystals with low drive level specification.
4:	Always verify oscillator performance over the VDD and temperature range that is expected for the application.

An internal postscaler allows users to select a clock frequency other than that of the crystal or resonator. Frequency division is determined by the CPDIV Configuration bits. Users may select a clock frequency of the oscillator frequency, or 1/2, 1/3 or 1/6 of the frequency.

6.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action on its stored value. They are:

- POSTDEC: accesses the FSR value, then automatically decrements it by one thereafter
- POSTINC: accesses the FSR value, then automatically increments it by one thereafter
- PREINC: increments the FSR value by one, then uses it in the operation
- PLUSW: adds the signed value of the W register (range of -128 to +127) to that of the FSR and uses the new value in the operation

In this context, accessing an INDF register uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value offset by the value in the W register; neither value is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR registers.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, roll-overs of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

6.4.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains FE7h, the address of INDF1. Attempts to read the value of the INDF1, using INDF0 as an operand, will return 00h. Attempts to write to INDF1, using INDF0 as the operand, will result in a `NOP`.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses Indirect Addressing.

Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

6.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds five additional two-word commands to the existing PIC18 instruction set: `ADDFSR`, `CALLW`, `MOVSF`, `MOVSS` and `SUBFSR`. These instructions are executed as described in **Section 6.2.4 “Two-Word Instructions”**.

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 1 byte at a time. A write to program memory is executed on blocks of 64 bytes at a time or 2 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)

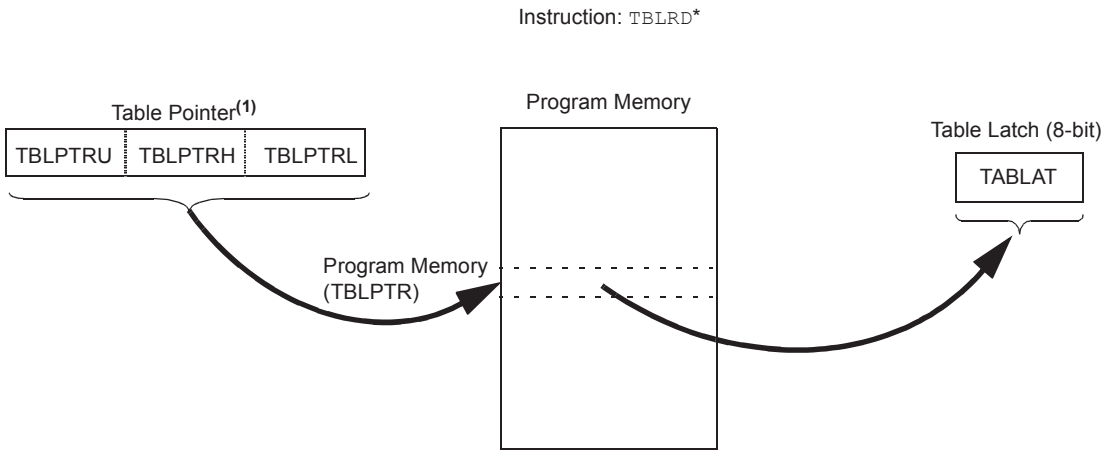
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 illustrates 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 illustrates 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: The Table Pointer register points to a byte in program memory.

REGISTER 9-12: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4 (ACCESS F8Eh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CCP10IE	CCP9IE	CCP8IE	CCP7IE	CCP6IE	CCP5IE	CCP4IE	CCP3IE
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 7-1 **CCP10IE:CCP4IE:** CCP<10:4> Interrupt Enable bits
1 = Enabled
0 = Disabled

bit 0 **CCP3IE:** ECCP3 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-13: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5 (ACCESS F91h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CM3IE	TMR8IE	TMR6IE	TMR5IE	TMR5GIE	TMR1GIE
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 **Unimplemented:** Read as '0'

bit 5 **CM3IE:** Comparator 3 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **TMR8IE:** TMR8 to PR8 Match Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **TMR5IE:** TMR5 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR5GIE:** TMR5 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR1GIE:** TMR1 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

TABLE 10-5: PORTB I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB7/CCP7/ KBI3/PGD/ RP10	RB7	0	O	DIG	LATB<7> data output.
		1	I	TTL	PORTB<7> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared.
	CCP7	1	I	ST	Capture input.
		0	O	DIG	Compare/PWM output.
	KBI3	1	O	TTL	Interrupt-on-change pin.
	PGD	x	O	DIG	Serial execution data output for ICSP and ICD operation.(2)
		x	I	ST	Serial execution data input for ICSP and ICD operation.(2)
	RP10	1	I	ST	Remappable Peripheral Pin 10 input.
		0	O	DIG	Remappable Peripheral Pin 10 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: Pins are configured as analog inputs by default on POR. Using these pins for digital inputs requires setting the appropriate bits in the ANCON1 register.

2: All other pin functions are disabled when ICSP™ or ICD is enabled.

3: Only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

4: Only on 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).

TABLE 10-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
ANCON1	VBGEN	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
REFOCON	ROON	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0
CM3CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	PMPTTL
RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0

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

10.7.3.1 Input Mapping

The inputs of the PPS options are mapped on the basis of the peripheral; that is, a control register associated with a peripheral dictates the pin it will be mapped to. The RPINRx registers are used to configure peripheral input mapping (see Register 10-6 through Register 10-23). Each register contains a 5-bit field which is associated with one of the pin selectable peripherals. Programming a given peripheral's bit field with an appropriate 5-bit value maps the RPn pin with that value to that peripheral. For any given device, the valid range of values for any of the bit fields corresponds to the maximum number of Peripheral Pin Selections supported by the device.

TABLE 10-13: SELECTABLE INPUT SOURCES (MAPS INPUT TO FUNCTION)(1)

Input Name	Function Name	Register	Configuration Bits
External Interrupt 1	INT1	RPINR1	INTR1R<4:0>
External Interrupt 2	INT2	RPINR2	INTR2R<4:0>
External Interrupt 3	INT3	RPINR3	INTR3R<4:0>
Timer0 External Clock Input	T0CKI	RPINR4	T0CKR<4:0>
Timer3 External Clock Input	T3CKI	RPINR6	T3CKR<4:0>
Timer5 External Clock Input	T5CKI	RPINR15	T5CKR<4:0>
Input Capture 1	CCP1	RPINR7	IC1R<4:0>
Input Capture 2	CCP2	RPINR8	IC2R<4:0>
Input Capture 3	CCP3	RPINR9	IC3R<4:0>
Timer1 Gate Input	T1G	RPINR12	T1GR<4:0>
Timer3 Gate Input	T3G	RPINR13	T3GR<4:0>
Timer5 Gate Input	T5G	RPINR14	T5GR<4:0>
EUSART2 Asynchronous Receive/Synchronous Receive	RX2/DT2	RPINR16	RX2DT2R<4:0>
EUSART2 Asynchronous Clock Input	CK2	RPINR17	CK2R<4:0>
SPI2 Data Input	SDI2	RPINR21	SDI2R<4:0>
SPI2 Clock Input	SCK2IN	RPINR22	SCK2R<4:0>
SPI2 Slave Select Input	SS2IN	RPINR23	SS2R<4:0>
PWM Fault Input	FLT0	RPINR24	OCFAR<4:0>

Note 1: Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

19.4.2 FULL-BRIDGE MODE

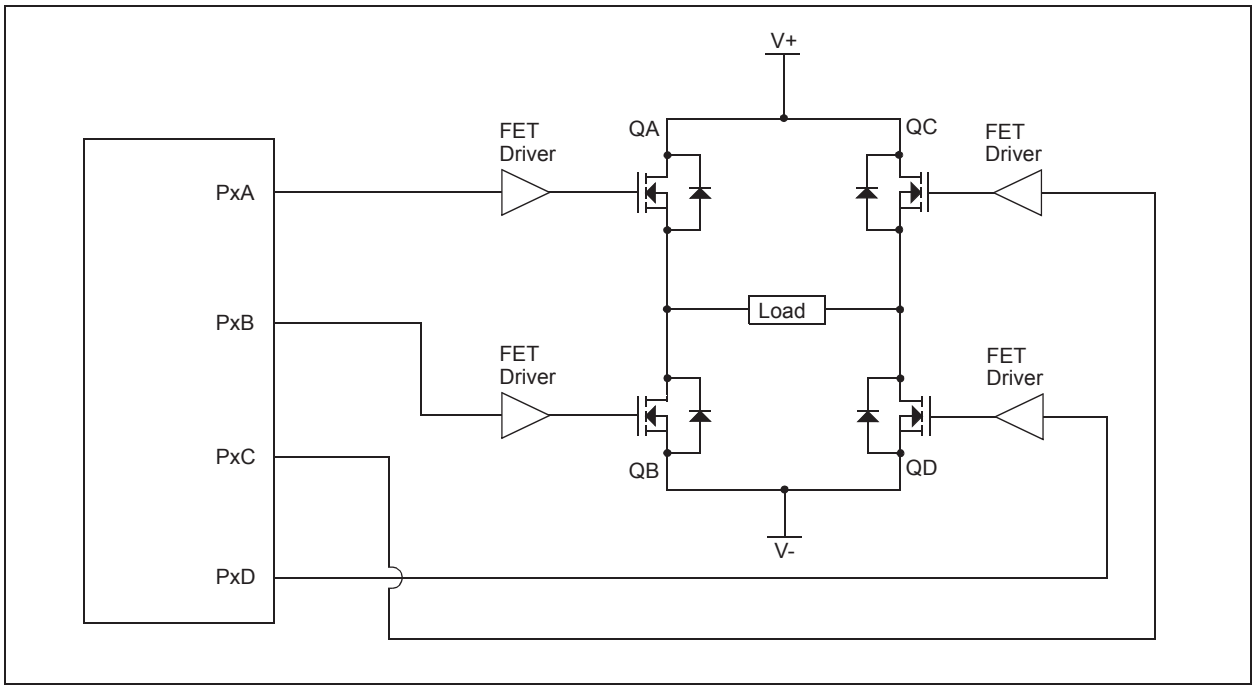
In Full-Bridge mode, all four pins are used as outputs. An example of a full-bridge application is provided in Figure 19-8.

In the Forward mode, the PxA pin is driven to its active state and the PxD pin is modulated, while the PxB and PxC pins are driven to their inactive state, as shown in Figure 19-9.

In the Reverse mode, the PxC pin is driven to its active state and the PxB pin is modulated, while the PxA and PxD pins are driven to their inactive state, as shown in Figure 19-9.

The PxA, PxB, PxC and PxD outputs are multiplexed with the port data latches. The associated TRIS bits must be cleared to configure the PxA, PxB, PxC and PxD pins as outputs.

FIGURE 19-8: EXAMPLE OF FULL-BRIDGE APPLICATION



20.3.1 REGISTERS

Each MSSP module has four registers for SPI mode operation. These are:

- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSPx Shift Register (SSPxSR) – Not directly accessible

SSPxCON1 and SSPxSTAT are the control and status registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPxSR and SSPxBUF together, create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

Note:	Because the SSPxBUF register is double-buffered, using read-modify-write instructions such as `BCF`, `COMF`, etc., will not work. Similarly, when debugging under an in-circuit debugger, performing actions that cause reads of SSPxBUF (mouse hovering, watch, etc.) can consume data that the application code was expecting to receive.

REGISTER 20-1: SSPxSTAT: MSSPx STATUS REGISTER (SPI MODE) (ACCESS 1, FC7h; 2, F73h)

R/W-1	R/W-1	R-1	R-1	R-1	R-1	R-1	R-1
SMP	CKE(1)	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
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 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at the end of data output time
0 = Input data sampled at the middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit(1)
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit
Used in I^2C mode only.

bit 4 **P:** Stop bit
Used in I^2C mode only; this bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit
Used in I^2C mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ Information bit
Used in I^2C mode only.

bit 1 **UA:** Update Address bit
Used in I^2C mode only.

bit 0 **BF:** Buffer Full Status bit
1 = Receive complete, SSPxBUF is full
0 = Receive not complete, SSPxBUF is empty

Note 1: Polarity of the clock state is set by the CKP bit (SSPxCON1<4>).

REGISTER 20-2: SSPxCON1: MSSPx CONTROL REGISTER 1 (SPI MODE) (1, ACCESS FC6h; 2, F72h)

R/W-0	R/C-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV(1)	SSPEN(2)	CKP	SSPM3(3)	SSPM2(3)	SSPM1(3)	SSPM0(3)
bit 7							bit 0

Legend:	C = Clearable bit		
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 **WCOL:** Write Collision Detect bit
- 1 = The SSPxBUF 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(1)

SPI Slave mode:
- 1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. The user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
- 0 = No overflow

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(2)
- 1 = Enables serial port and configures SCKx, SDOx, SDIx and $\overline{SSx}$ as serial port pins
- 0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
- 1 = Idle state for clock is a high level
- 0 = Idle state for clock is a low level

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(3)
- 0101 = SPI Slave mode, clock = SCKx pin; $\overline{SSx}$ pin control disabled, $\overline{SSx}$ can be used as I/O pin
- 0100 = SPI Slave mode, clock = SCKx pin; $\overline{SSx}$ pin control enabled
- 0011 = SPI Master mode, clock = TMR2 output/2
- 0010 = SPI Master mode, clock = FOSC/64
- 0001 = SPI Master mode, clock = FOSC/16
- 1010 = SPI Master mode, clock = FOSC/8
- 0000 = SPI Master mode, clock = FOSC/4

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.

2: When enabled, this pin must be properly configured as input or output.

3: Bit combinations not specifically listed here are either reserved or implemented in I^2C mode only.

20.4 SPI DMA MODULE

The SPI DMA module contains control logic to allow the MSSP2 module to perform SPI direct memory access transfers. This enables the module to quickly transmit or receive large amounts of data with relatively little CPU intervention. When the SPI DMA module is used, MSSP2 can directly read and write to general purpose SRAM. When the SPI DMA module is not enabled, MSSP2 functions normally, but without DMA capability.

The SPI DMA module is composed of control logic, a Destination Receive Address Pointer, a Transmit Source Address Pointer, an interrupt manager and a Byte Count register for setting the size of each DMA transfer. The DMA module may be used with all SPI Master and Slave modes, and supports both half-duplex and full-duplex transfers.

20.4.1 I/O PIN CONSIDERATIONS

When enabled, the SPI DMA module uses the MSSP2 module. All SPI input and output signals, related to MSSP2, are routed through the Peripheral Pin Select module. The appropriate initialization procedure, as described in **Section 20.4.6 "Using the SPI DMA Module"**, will need to be followed prior to using the SPI DMA module. The output pins assigned to the SDO2 and SCK2 functions can optionally be configured as open-drain outputs, such as for level shifting operations mentioned in the same section.

20.4.2 RAM TO RAM COPY OPERATIONS

Although the SPI DMA module is primarily intended to be used for SPI communication purposes, the module can also be used to perform RAM to RAM copy operations. To do this, configure the module for Full-Duplex Master mode operation, but assign the SDO2 output and SDI2 input functions onto the same RPn pin in the PPS module. Also assign SCK2 out and SCK2 in onto the same RPn pin (a different pin than used for SDO2 and SDI2). This will allow the module to operate in Loopback mode, providing RAM copy capability.

20.4.3 IDLE AND SLEEP CONSIDERATIONS

The SPI DMA module remains fully functional when the microcontroller is in Idle mode.

During normal Sleep, the SPI DMA module is not functional and should not be used. To avoid corrupting a transfer, user firmware should be careful to make certain that pending DMA operations are complete by polling the DMAEN bit in the DMACON1 register prior to putting the microcontroller into Sleep.

In SPI Slave modes, the MSSP2 module is capable of transmitting and/or receiving one byte of data while in Sleep mode. This allows the SSP2IF flag in the PIR3 register to be used as a wake-up source. When the DMAEN bit is cleared, the SPI DMA module is effectively disabled, and the MSSP2 module functions normally, but without DMA capabilities. If the DMAEN bit is clear prior to entering Sleep, it is still possible to use the SSP2IF as a wake-up source without any data loss.

Neither MSSP2 nor the SPI DMA module will provide any functionality in Deep Sleep. Upon exiting from Deep Sleep, all of the I/O pins, MSSP2 and SPI DMA related registers will need to be fully re-initialized before the SPI DMA module can be used again.

20.4.4 REGISTERS

The SPI DMA engine is enabled and controlled by the following Special Function Registers:

- DMACON1
- DMACON2
- TXADDRH
- TXADDRL
- RXADDRH
- RXADDRL
- DMABCH
- DMABCL

20.5.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPxCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCLx pin to be held low at the end of each data receive sequence.

20.5.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the ACK sequence, if the BF bit is set, the CKP bit in the SSPxCON1 register is automatically cleared, forcing the SCLx output to be held low. The CKP bit being cleared to '0' will assert the SCLx line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and read the contents of the SSPxBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 20-15).

Note 1: If the user reads the contents of the SSPxBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

20.5.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = 1)

In 10-Bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/W bit cleared to '0'. The release of the clock line occurs upon updating SSPxADD. Clock stretching will occur on each data receive sequence, as described in 7-bit mode.

Note: If the user polls the UA bit and clears it by updating the SSPxADD register before the falling edge of the ninth clock occurs, and if the user has not cleared the BF bit by reading the SSPxBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

20.5.4.3 Clock Stretching for 7-Bit Slave Transmit Mode

The 7-Bit Slave Transmit mode implements clock stretching by clearing the CKP bit, after the falling edge of the ninth clock, if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's Interrupt Service Routine (ISR) must set the CKP bit before transmission is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and load the contents of the SSPxBUF before the master device can initiate another transmit sequence (see Figure 20-10).

Note 1: If the user loads the contents of SSPxBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit.

20.5.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high-order bits of the 10-bit address and the R/W bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 20-13).

21.1 Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit (BAUDCONx<3>) selects 16-bit mode.

The SPBRGHx:SPBRGx register pair controls the period of a free-running timer. In Asynchronous mode, the BRGH (TXSTAx<2>) and BRG16 (BAUDCONx<3>) bits also control the baud rate. In Synchronous mode, BRGH is ignored.

Table 21-1 provides the formula for computation of the baud rate for different EUSART modes, which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGHx:SPBRGx registers can be calculated using the formulas in Table 21-1. From this, the error in baud rate can be determined. An example calculation is provided in Example 21-1. Typical baud rates and error values for the various Asynchronous modes are provided in Table 21-2. It may be advantageous to use the high baud rate (BRGH = 1) or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGHx:SPBRGx registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

When operated in the Synchronous mode, SPBRGH:SPBRG values of 0000h and 0001h are not supported. In the Asynchronous mode, all BRG values may be used.

21.1.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRGx register pair.

21.1.2 SAMPLING

The data on the RXx pin (either RC7/CCP10/PMA4/RX1/DT1/RP18 or RPn/RX2/DT2) is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RXx pin.

TABLE 21-1: BAUD RATE FORMULAS
TABLE 21-4:

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	FOSC/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	FOSC/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	
1	0	x	8-bit/Synchronous	FOSC/[4 (n + 1)]
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGHx:SPBRGx register pair

26.4 Measuring Capacitance with the CTMU

There are two separate methods of measuring capacitance with the CTMU. The first is the absolute method, in which the actual capacitance value is desired. The second is the relative method, in which the actual capacitance is not needed, rather an indication of a change in capacitance is required.

26.4.1 ABSOLUTE CAPACITANCE MEASUREMENT

For absolute capacitance measurements, both the current and capacitance calibration steps found in **Section 26.3 "Calibrating the CTMU Module"** should be followed. Capacitance measurements are then performed using the following steps:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Set EDG1STAT.
4. Wait for a fixed delay, T.
5. Clear EDG1STAT.
6. Perform an A/D conversion.
7. Calculate the total capacitance, $C_{TOTAL} = (I * T)/V$, where I is known from the current source measurement step (see **Section 26.3.1 "Current Source Calibration"**), T is a fixed delay and V is measured by performing an A/D conversion.
8. Subtract the stray and A/D capacitance (C_{OFFSET} from **Section 26.3.2 "Capacitance Calibration"**) from C_{TOTAL} to determine the measured capacitance.

26.4.2 CAPACITIVE TOUCH SENSE USING RELATIVE CHARGE MEASUREMENT

An application may not require precise capacitance measurements. For example, when detecting a valid press of a capacitance-based touch sense button, detecting a relative change of capacitance is of interest. In this type of application, when the touch sense pad is not being pressed, the total capacitance is the capacitance of the combination of the board traces, the A/D Converter, etc. A larger voltage will be measured by the A/D Converter. When the touch sense pad is pressed, the total capacitance is larger due to the addition of the capacitance of the human body, therefore, a smaller voltage will be measured by the A/D Converter.

Detecting capacitance changes is easily accomplished with the CTMU using these steps:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT.
3. Wait for a fixed delay.
4. Clear EDG1STAT.
5. Perform an A/D conversion.

The voltage measured by performing the A/D conversion is an indication of the relative capacitance. Note that in this case, no calibration of the current source or circuit capacitance measurement is needed. See Example 26-4 for a sample software routine for a capacitive touch switch.

REGISTER 27-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)

U-1	U-1	U-1	U-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
—	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Program the corresponding Flash Configuration bit to '1'

bit 3-0 **WDTPS<3:0>:** Watchdog Timer Postscale Select bits

1111 = 1:32,768
1110 = 1:16,384
1101 = 1:8,192
1100 = 1:4,096
1011 = 1:2,048
1010 = 1:1,024
1001 = 1:512
1000 = 1:256
0111 = 1:128
0110 = 1:64
0101 = 1:32
0100 = 1:16
0011 = 1:8
0010 = 1:4
0001 = 1:2
0000 = 1:1

BCF **Bit Clear f**

Syntax: BCF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared.

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 28.2.3 "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: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

BN **Branch if Negative**

Syntax: BN n

Operands: $-128 \le n \le 127$

Operation: if Negative bit is '1',
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the Negative bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BN Jump`

Before Instruction
PC = address (HERE)
After Instruction
If Negative = 1;
PC = address (Jump)
If Negative = 0;
PC = address (HERE + 2)

POP **Pop Top of Return Stack**

Syntax: POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

0000	0000	0000	0110

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:

```
POP
GOTO    NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

PUSH **Push Top of Return Stack**

Syntax: PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding:

0000	0000	0000	0101

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:

```
PUSH
```

Before Instruction
TOS = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah

29.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

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

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. 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. MPLAB XC Compiler uses the assembler to produce its object 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 X IDE compatibility

29.3 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 X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

29.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. 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

29.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC 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 X IDE compatibility

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](1)	-	X	/XX	XXX
Device	Tape and Reel Option		Temperature Range	Package	Pattern

Device(3,4):	PIC18F26J13, PIC18F27J13, PIC18F46J13 and PIC18F47J13 VDD range 2.15V to 3.6V PIC18LF26J13, PIC18LF27J13, PIC18LF46J13 and PIC18LF47J13 VDD range 2.0V to 3.6V
Temperature Range:	I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended)
Package(2):	ML = QFN PT = TQFP (Thin Quad Flatpack) SO = SOIC SP = Skinny Plastic DIP SS = SSOP
Pattern:	QTP, SQTP, Code or Special Requirements (blank otherwise)

Examples:

a) PIC18F47J13-I/PT 301 = Industrial temp., TQFP package, QTP pattern #301.

b) PIC18F47J13T-I/PT = Tape and reel, Industrial temp., TQFP package.

Note 1: Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

2: Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

3: F = Standard Voltage Range with internal 2.5V core voltage regulator
LF= Wide Voltage Range with no internal core voltage regulator

4: T = In tape and reel TQFP packages only

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820