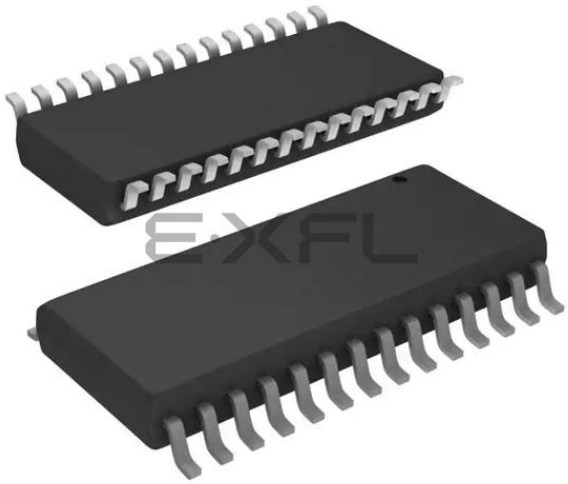


Welcome to E-XFL.COM

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	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 24x10b; D/A 1x5b
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/pic18lf24k40-i-so

4.3.2.6 Oscillator Status and Manual Enable

The Ready status of each oscillator (including the ADCRC oscillator) is displayed in OSCSTAT (Register 4-4). The oscillators (but not the PLL) may be explicitly enabled through OSCEN (Register 4-7).

4.3.2.7 HFOR and MFOR Bits

The HFOR and MFOR bits indicate that the HFINTOSC and MFINTOSC is ready. These clocks are always valid for use at all times, but only accurate after they are ready.

When a new value is loaded into the OSCFRQ register, the HFOR and MFOR bits will clear, and set again when the oscillator is ready. During pending OSCFRQ changes the MFINTOSC clock will stall at a high or a low state, until the HFINTOSC resumes operation.

4.4 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the New Oscillator Source (NOSC) bits of the OSCCON1 register. The following clock sources can be selected using the following:

- External oscillator
- Internal Oscillator Block (INTOSC)

Note:	The Clock Switch Enable bit in Configuration Word 1 can be used to enable or disable the clock switching capability. When cleared, the NOSC and NDIV bits cannot be changed by user software. When set, writing to NOSC and NDIV is allowed and would switch the clock frequency.

4.4.1 NEW OSCILLATOR SOURCE (NOSC) AND NEW DIVIDER SELECTION REQUEST (NDIV) BITS

The New Oscillator Source (NOSC) and New Divider Selection Request (NDIV) bits of the OSCCON1 register select the system clock source and frequency that are used for the CPU and peripherals.

When new values of NOSC and NDIV are written to OSCCON1, the current oscillator selection will continue to operate while waiting for the new clock source to indicate that it is stable and ready. In some cases, the newly requested source may already be in use, and is ready immediately. In the case of a divider-only change, the new and old sources are the same, so the old source will be ready immediately. The device may enter Sleep while waiting for the switch as described in **Section 4.4.2 "Clock Switch and Sleep"**.

When the new oscillator is ready, the New Oscillator Ready (NOSCR) bit of OSCCON3 is set and also the Clock Switch Interrupt Flag (CSWIF) bit of PIR1 sets. If Clock Switch Interrupts are enabled (CSWIE = 1), an interrupt will be generated at that time. The Oscillator Ready (ORDY) bit of OSCCON3 can also be polled to determine when the oscillator is ready in lieu of an interrupt.

Note:	The CSWIF interrupt will not wake the system from Sleep.

If the Clock Switch Hold (CSWHOLD) bit of OSCCON3 is clear, the oscillator switch will occur when the New Oscillator is Ready bit (NOSCR) is set, and the interrupt (if enabled) will be serviced at the new oscillator setting.

If CSWHOLD is set, the oscillator switch is suspended, while execution continues using the current (old) clock source. When the NOSCR bit is set, software should:

- Set CSWHOLD = 0 so the switch can complete, or
- Copy COSC into NOSC to abandon the switch.

If DOZE is in effect, the switch occurs on the next clock cycle, whether or not the CPU is operating during that cycle.

Changing the clock post-divider without changing the clock source (i.e., changing FOSC from 1 MHz to 2 MHz) is handled in the same manner as a clock source change, as described previously. The clock source will already be active, so the switch is relatively quick. CSWHOLD must be clear (CSWHOLD = 0) for the switch to complete.

The current COSC and CDIV are indicated in the OSCCON2 register up to the moment when the switch actually occurs, at which time OSCCON2 is updated and ORDY is set. NOSCR is cleared by hardware to indicate that the switch is complete.

7.0 PERIPHERAL MODULE DISABLE (PMD)

Sleep, Idle and Doze modes allow users to substantially reduce power consumption by slowing or stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume some amount of power. There may be cases where the application needs what these modes do not provide: the ability to allocate limited power resources to the CPU while eliminating power consumption from the peripherals.

The PIC18(L)F2x/4xK40 family addresses this requirement by allowing peripheral modules to be selectively enabled or disabled, placing them into the lowest possible power mode.

For legacy reasons, all modules are ON by default following any Reset.

7.1 Disabling a Module

Disabling a module has the following effects:

- All clock and control inputs to the module are suspended; there are no logic transitions, and the module will not function.
- The module is held in Reset.
- Any SFR becomes “unimplemented”
  - Writing is disabled
  - Reading returns 00h
- I/O functionality is prioritized as per **Section 15.1, I/O Priorities**
- All associated Input Selection registers are also disabled

7.2 Enabling a Module

When the PMD register bit is cleared, the module is re-enabled and will be in its Reset state (Power-on Reset). SFR data will reflect the POR Reset values.

Depending on the module, it may take up to one full instruction cycle for the module to become active. There should be no interaction with the module (e.g., writing to registers) for at least one instruction after it has been re-enabled.

7.3 Effects of a Reset

Following any Reset, each control bit is set to ‘0’, enabling all modules.

7.4 System Clock Disable

Setting SYSCMD (PMD0, Register 7-1) disables the system clock (FOSC) distribution network to the peripherals. Not all peripherals make use of SYSCLK, so not all peripherals are affected. Refer to the specific peripheral description to see if it will be affected by this bit.

10.1.1 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 10.2.3.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

10.1.2 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, or as a 35-word by 21-bit RAM with a 6-bit Stack Pointer in ICD mode. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack (TOS) Special File registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits in the PCON0 register indicate if the stack is full or has overflowed or has underflowed.

10.1.2.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 10-1). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the Global Interrupt Enable (GIE) bits while accessing the stack to prevent inadvertent stack corruption.

TABLE 10-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F24/25K40 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
F5Fh	ADPCH	F31h	FVRCON	F03h	—	ED5h	WDTPSH	EA7h	T3CKIPPS
F5Eh	ADPRE	F30h	HLVDCON1	F02h	—	ED4h	WDTPSL	EA6h	T1GPPS
F5Dh	ADCAP	F2Fh	HLVDCON0	F01h	—	ED3h	WDTCON1	EA5h	T1CKIPPS
F5Ch	ADACQ	F2Eh	—	F00h	—	ED2h	WDTCON0	EA4h	T0CKIPPS
F5Bh	ADCON3	F2Dh	WPUE	EFFh	—	ED1h	PIR7	EA3h	INT2PPS
F5Ah	ADCON2	F2Ch	—	EFEh	RC7PPS	ED0h	PIR6	EA2h	INT1PPS
F59h	ADCON1	F2Bh	—	EFDh	RC6PPS	ECFh	PIR5	EA1h	INT0PPS
F58h	ADREF	F2Ah	INLVLE	EFCh	RC5PPS	ECEh	PIR4	EA0h	PPSLOCK
F57h	ADCLK	F29h	IOCEP	EFBh	RC4PPS	ECDh	PIR3		
F56h	ADACT	F28h	IOCEN	EFAh	RC3PPS	ECCh	PIR2		
F55h	MDCARH	F27h	IOCEF	EF9h	RC2PPS	ECBh	PIR1		
F54h	MDCARL	F26h	—	EF8h	RC1PPS	ECAh	PIR0		
F53h	MDSRC	F25h	—	EF7h	RC0PPS	EC9h	PIE7		
F52h	MDCON1	F24h	—	EF6h	RB7PPS	EC8h	PIE6		
F51h	MDCON0	F23h	—	EF5h	RB6PPS	EC7h	PIE5		
F50h	SCANDTRIG	F22h	—	EF4h	RB5PPS	EC6h	PIE4		
F4Fh	SCANCON0	F21h	ANSELC	EF3h	RB4PPS	EC5h	PIE3		
F4Eh	SCANHADRU	F20h	WPUC	EF2h	RB3PPS	EC4h	PIE2		
F4Dh	SCANHADRH	F1Fh	ODCONC	EF1h	RB2PPS	EC3h	PIE1		
F4Ch	SCANHADRL	F1Eh	SLRCONC	EF0h	RB1PPS	EC2h	PIE0		
F4Bh	SCANLADRU	F1Dh	INLVLC	EEFh	RB0PPS	EC1h	IPR7		
F4Ah	SCANLADRH	F1Ch	IOCCP	EEEh	RA7PPS	EC0h	IPR6		
F49h	SCANLADRL	F1Bh	IOCCN	EEDh	RA6PPS	EBFh	IPR5		
F48h	CWG1STR	F1Ah	IOCCF	EECh	RA5PPS	EBEh	IPR4		
F47h	CWG1AS1	F19h	ANSELB	EEBh	RA4PPS	EBDh	IPR3		
F46h	CWG1AS0	F18h	WPUB	EEAh	RA3PPS	EBCh	IPR2		
F45h	CWG1CON1	F17h	ODCONB	EE9h	RA2PPS	EBBh	IPR1		
F44h	CWG1CON0	F16h	SLRCONB	EE8h	RA1PPS	EBAh	IPR0		
F43h	CWG1DBF	F15h	INLVLB	EE7h	RA0PPS	EB9h	SSP1SSPPS		
F42h	CWG1DBR	F14h	IOCBP	EE6h	PMD5	EB8h	SSP1DATPPS		
F41h	CWG1ISM	F13h	IOCBN	EE5h	PMD4	EB7h	SSP1CLKPPS		
F40h	CWG1CLKCON	F12h	IOCBF	EE4h	PMD3	EB6h	TX1PPS		
F3Fh	CLKRCLK	F11h	ANSELA	EE3h	PMD2	EB5h	RX1PPS		
F3Eh	CLKRCON	F10h	WPUA	EE2h	PMD1	EB4h	MDSRCPPS		
F3Dh	CMOUT	F0Fh	ODCONA	EE1h	PMD0	EB3h	MDCARHPPS		
F3Ch	CM1PCH	F0Eh	SLRCONA	EE0h	BORCON	EB2h	MDCARLPPS		
F3Bh	CM1NCH	F0Dh	INLVLA	EDFh	VREGCON(1)	EB1h	CWGINPPS		
F3Ah	CM1CON1	F0Ch	IOCAP	EDEh	OSCFRQ	EB0h	CCP2PPS		
F39h	CM1CON0	F0Bh	IOCAN	EDDh	OSCTUNE	EAFh	CCP1PPS		
F38h	CM2PCH	F0Ah	IOCAF	EDCh	OSCEN	EAEh	ADACTPPS		
F37h	CM2NCH	F09h	—	EDBh	OSCSTAT	EADh	T6INPPS		
F36h	CM2CON1	F08h	—	EDAh	OSCCON3	EACh	T4INPPS		
F35h	CM2CON0	F07h	—	ED9h	OSCCON2	EABh	T2INPPS		
F34h	DAC1CON1	F06h	—	ED8h	OSCCON1	EAAh	T5GPPS		
F33h	DAC1CON0	F05h	—	ED7h	CPUDOZE	EA9h	T5CKIPPS		
F32h	ZCDCON	F04h	—	ED6h	WDTTMR	EA8h	T3GPPS		

Note 1: Not available on LF parts

10.6 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 10.7 "Data Memory and the Extended Instruction Set"** for more information.

While the program memory can be addressed in only one way – through the program counter – 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 greater detail in **Section 10.7.1 "Indexed Addressing with Literal Offset"**.

10.6.1 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.

10.6.2 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 Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 10.4.3 "General Purpose Register File"**) or a location in the Access Bank (**Section 10.4.2 "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 10.4.1 "Bank Select Register (BSR)"**) 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.

10.6.3 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 which are to be read or written. Since the FSRs are themselves located in RAM as Special File Registers, 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 10-5.

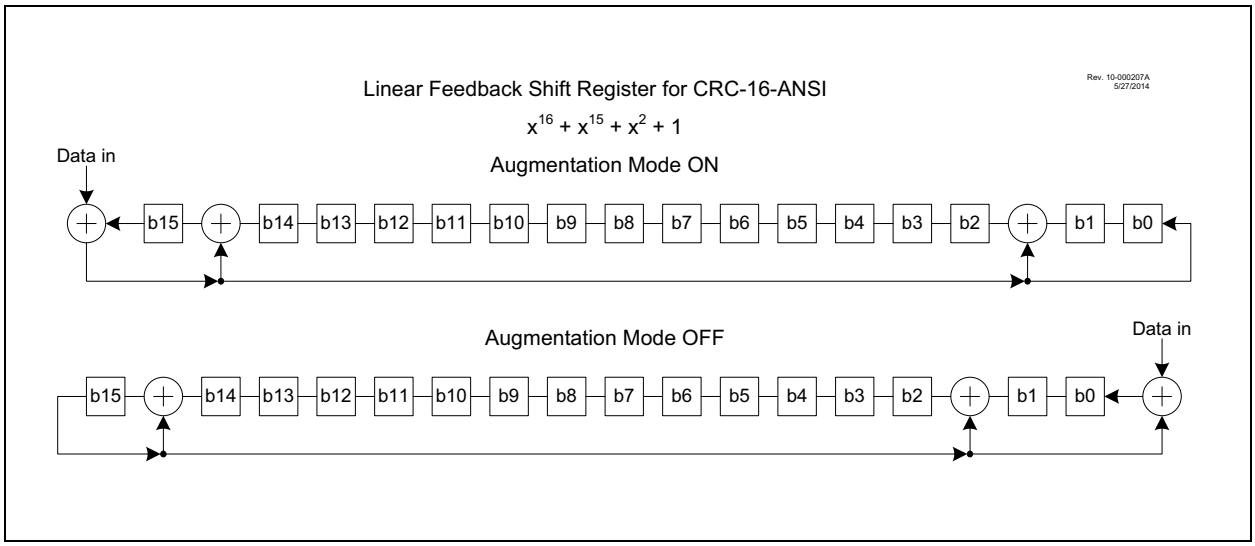
EXAMPLE 10-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
         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
```

13.4 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the PLEN<3:0> bits. For an n-bit accumulator, PLEN = n-1 and the corresponding polynomial is n+1 bits. Therefore, the accumulator can be any size up to 16 bits with a corresponding polynomial up to 17 bits. The MSb and LSb of the polynomial are always '1' which is forced by hardware. All polynomial bits between the MSb and LSb are specified by the CRCXOR registers. For example, when using CRC16-ANSI, the polynomial is defined as $X^{16}+X^{15}+X^2+1$. The X^{16} and X^0 = 1 terms are the MSb and LSb controlled by hardware. The X^{15} and X^2 terms are specified by setting the corresponding CRCXOR<15:0> bits with the value of 0x8004. The actual value is 0x8005 because the hardware sets the LSb to 1. However, the LSb of the CRCXORL register is unimplemented and always reads as '0'. Refer to Example 13-1.

EXAMPLE 13-2: CRC LFSR EXAMPLE



13.5 CRC Data Sources

Data can be input to the CRC module in two ways:

- User data using the CRCDATA registers (CRCDATH and CRCDATL)
- Flash using the Program Memory Scanner

To set the number of bits of data, up to 16 bits, the DLEN bits of CRCCON1 must be set accordingly. Only data bits in CRCDATA registers up to DLEN will be used, other data bits in CRCDATA registers will be ignored.

Data is moved into the CRCSHIFT as an intermediate to calculate the check value located in the CRCACC registers.

The SHIFTM bit is used to determine the bit order of the data being shifted into the accumulator. If SHIFTM is not set, the data will be shifted in MSb first (Big Endian). The value of DLEN will determine the MSb. If SHIFTM bit is set, the data will be shifted into the accumulator in reversed order, LSb first (Little Endian).

The CRC module can be seeded with an initial value by setting the CRCACC<15:0> registers to the appropriate value before beginning the CRC.

13.5.1 CRC FROM USER DATA

To use the CRC module on data input from the user, the user must write the data to the CRCDAT registers. The data from the CRCDAT registers will be latched into the shift registers on any write to the CRCDATL register.

13.5.2 CRC FROM FLASH

To use the CRC module on data located in Flash memory, the user can initialize the Program Memory Scanner as defined in **Section 13.9, Program Memory Scan Configuration**.

REGISTER 14-14: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE
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 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **TMR5IE:** TMR5 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **TMR3IE:** TMR3 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

TABLE 17-1: PPS INPUT REGISTER DETAILS

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	Input Available from Selected PORTx		
Interrupt 0	INT0PPS	RB0	5'b0 1000	A	B	—
Interrupt 1	INT1PPS	RB1	5'b0 1001	A	B	—
Interrupt 2	INT2PPS	RB2	5'b0 1010	A	B	—
Timer0 Clock	T0CKIPPS	RA4	5'b0 0100	A	B	—
Timer1 Clock	T1CKIPPS	RC0	5'b1 0000	A	—	C
Timer1 Gate	T1GPPS	RB5	5'b0 1101	—	B	C
Timer3 Clock	T3CKIPPS	RC0	5'b1 0000	—	B	C
Timer3 Gate	T3GPPS	RC0	5'b1 0000	A	—	C
Timer5 Clock	T5CKIPPS	RC2	5'b1 0010	A	—	C
Timer5 Gate	T5GPPS	RB4	5'b0 1100	—	B	C
Timer2 Clock	T2INPPS	RC3	5'b1 0011	A	—	C
Timer4 Clock	T4INPPS	RC5	5'b1 0101	—	B	C
Timer6 Clock	T6INPPS	RB7	5'b0 1111	—	B	C
CCP1	CCP1PPS	RC2	5'b1 0010	—	B	C
CCP2	CCP2PPS	RC1	5'b1 0001	—	B	C
CWG	CWG1PPS	RB0	5'b0 1000	—	B	C
DSM Carrier Low	MDCARLPPS	RA3	5'b0 0011	A	—	C
DSM Carrier High	MDCARHPPS	RA4	5'b0 0100	A	—	C
DSM Source	MDSRCPPS	RA5	5'b0 0101	A	—	C
ADC Conversion Trigger	ADACTPPS	RB4	5'b0 1100	—	B	C
MSSP1 Clock	SSP1CLKPPS	RC3	5'b1 0011	—	B	C
MSSP1 Data	SSP1DATPPS	RC4	5'b1 0100	—	B	C
MSSP1 Slave Select	SSP1SSPPS	RA5	5'b0 0101	A	—	C
EUSART1 Receive	RX1PPS	RC7	5'b1 0111	—	B	C
EUSART1 Transmit	TX1PPS	RC6	5'b1 0110	—	B	C

REGISTER 17-2: RxyPPS: PIN Rxy OUTPUT SOURCE SELECTION REGISTER

U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	RxyPPS<4:0>				
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **RxyPPS<4:0>:** Pin Rxy Output Source Selection bits

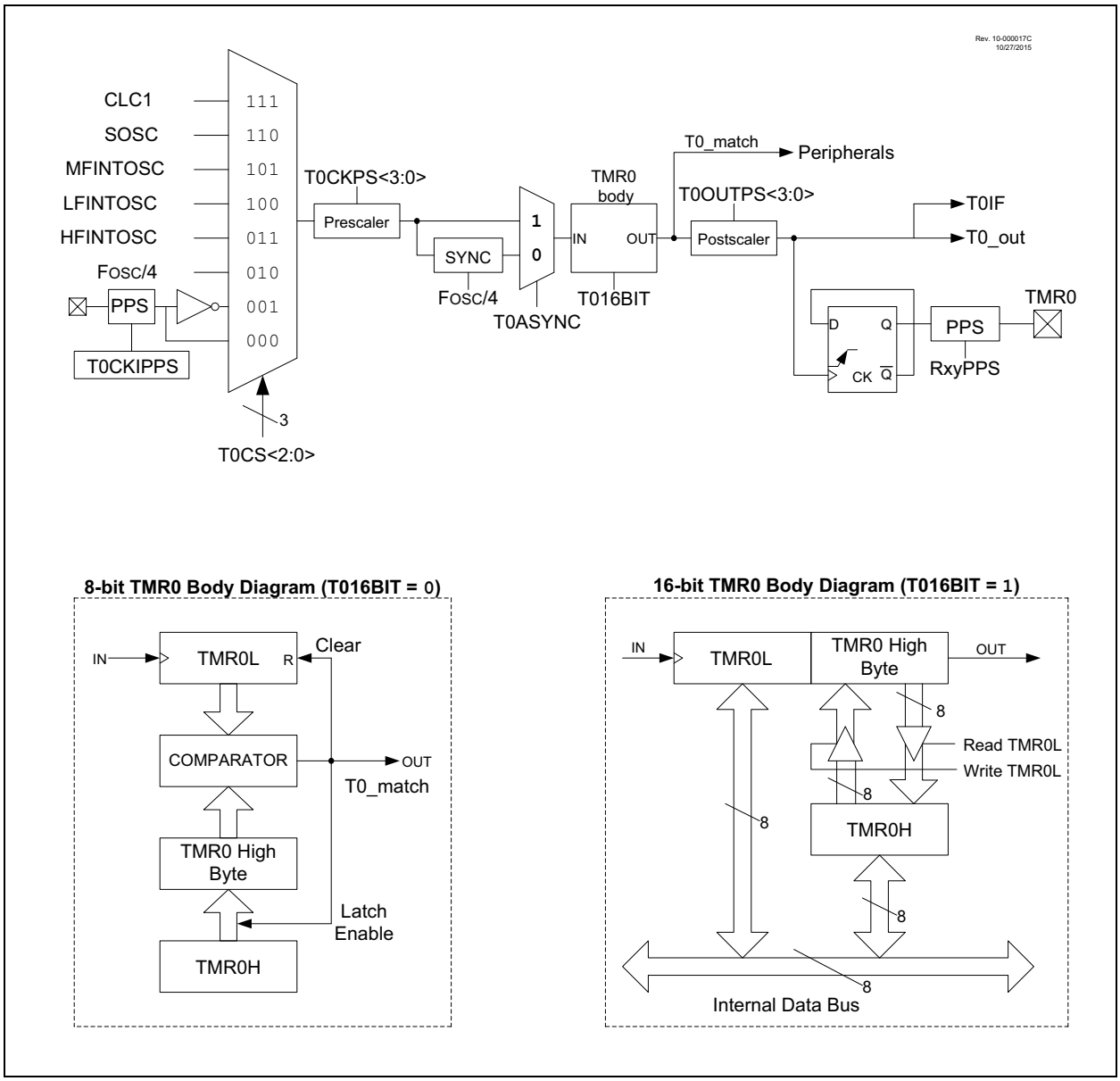
RxyPPS<4:0>	**Pin Rxy Output Source**	**Device Configuration**		
		PIC18(L)F24/25K40		
0x13	ADGRDB	A	—	C
0x12	ADGRDA	A	—	C
0x11	DSM	A	—	C
0x10	CLKR	—	B	C
0x0F	TMR0	—	B	C
0x0E	MSSP1 (SDO/SDA)	—	B	C
0x0D	MSSP1 (SCK/SCL)	—	B	C
0x0C	CMP2	A	—	C
0x0B	CMP1	A	—	C
0x0A	EUSART1 (RX)	—	B	C
0x09	EUSART1 (TX)	—	B	C
0x08	PWM4	A	—	C
0x07	PWM3	A	—	C
0x06	CCP2	—	B	C
0x05	CCP1	—	B	C
0x04	CWG1D	—	B	C
0x03	CWG1C	—	B	C
0x02	CWG1B	—	B	C
0x01	CWG1A	—	B	C
0x00	LATxy	A	B	C

18.0 TIMER0 MODULE

Timer0 module is an 8/16-bit timer/counter with the following features:

- 16-bit timer/counter
- 8-bit timer/counter with programmable period
- Synchronous or asynchronous operation
- Selectable clock sources
- Programmable prescaler
- Programmable postscaler
- Operation during Sleep mode
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals

FIGURE 18-1: BLOCK DIAGRAM OF TIMER0



REGISTER 19-2: TxGCON: TIMERx GATE CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R-x	U-0	U-0
GE	GPOL	GTM	GSPM	GGO/$\overline{\text{DONE}}$	GVAL	—	—
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 **GE:** Timerx Gate Enable bit

If TMRxON = 1:
1 = Timerx counting is controlled by the Timerx gate function
0 = Timerx is always counting
If TMRxON = 0:
This bit is ignored

bit 6 **GPOL:** Timerx Gate Polarity bit

1 = Timerx gate is active-high (Timerx counts when gate is high)
0 = Timerx gate is active-low (Timerx counts when gate is low)

bit 5 **GTM:** Timerx Gate Toggle Mode bit

1 = Timerx Gate Toggle mode is enabled
0 = Timerx Gate Toggle mode is disabled and Toggle flip-flop is cleared
Timerx Gate Flip Flop Toggles on every rising edge

bit 4 **GSPM:** Timerx Gate Single Pulse Mode bit

1 = Timerx Gate Single Pulse mode is enabled and is controlling Timerx gate)
0 = Timerx Gate Single Pulse mode is disabled

bit 3 **GGO/$\overline{\text{DONE}}$:** Timerx Gate Single Pulse Acquisition Status bit

1 = Timerx Gate Single Pulse Acquisition is ready, waiting for an edge
0 = Timerx Gate Single Pulse Acquisition has completed or has not been started.
This bit is automatically cleared when TxGSPM is cleared.

bit 2 **GVAL:** Timerx Gate Current State bit

Indicates the current state of the Timerx gate that could be provided to TMRxH:TMRxL
Unaffected by Timerx Gate Enable (TMRxGE)

bit 1-0 **Unimplemented:** Read as '0'

FIGURE 24-16: SHUTDOWN FUNCTIONALITY, AUTO-RESTART ENABLED (REN = 1, LSAC = 01, LSBD = 01)

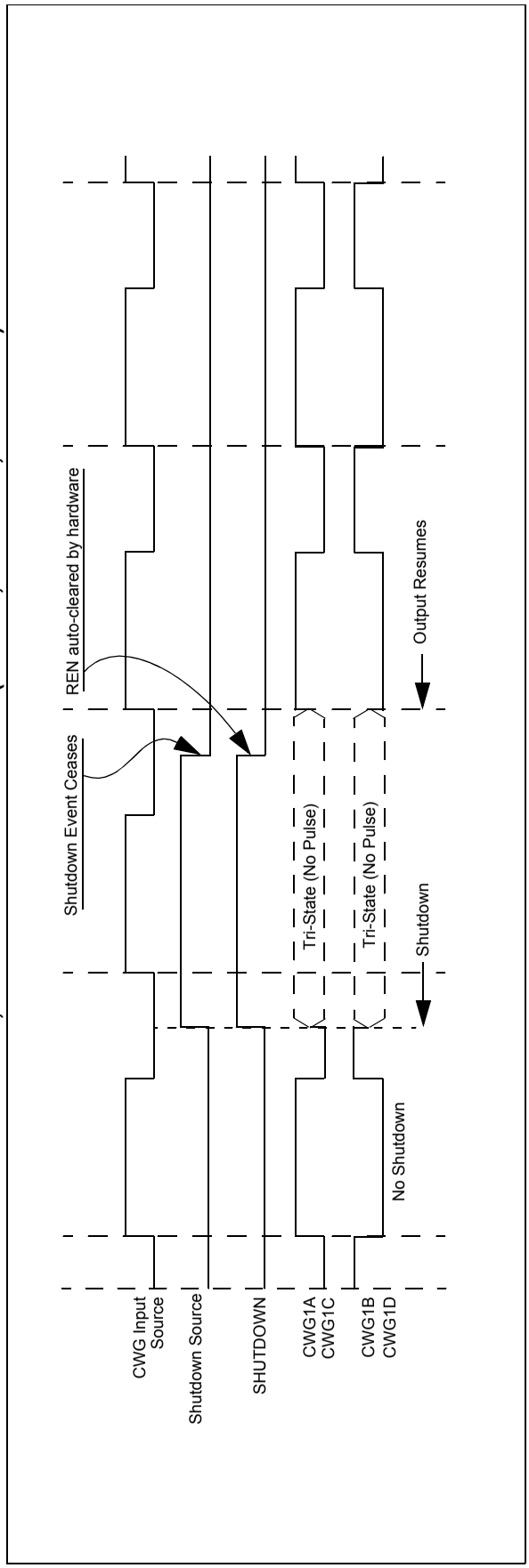
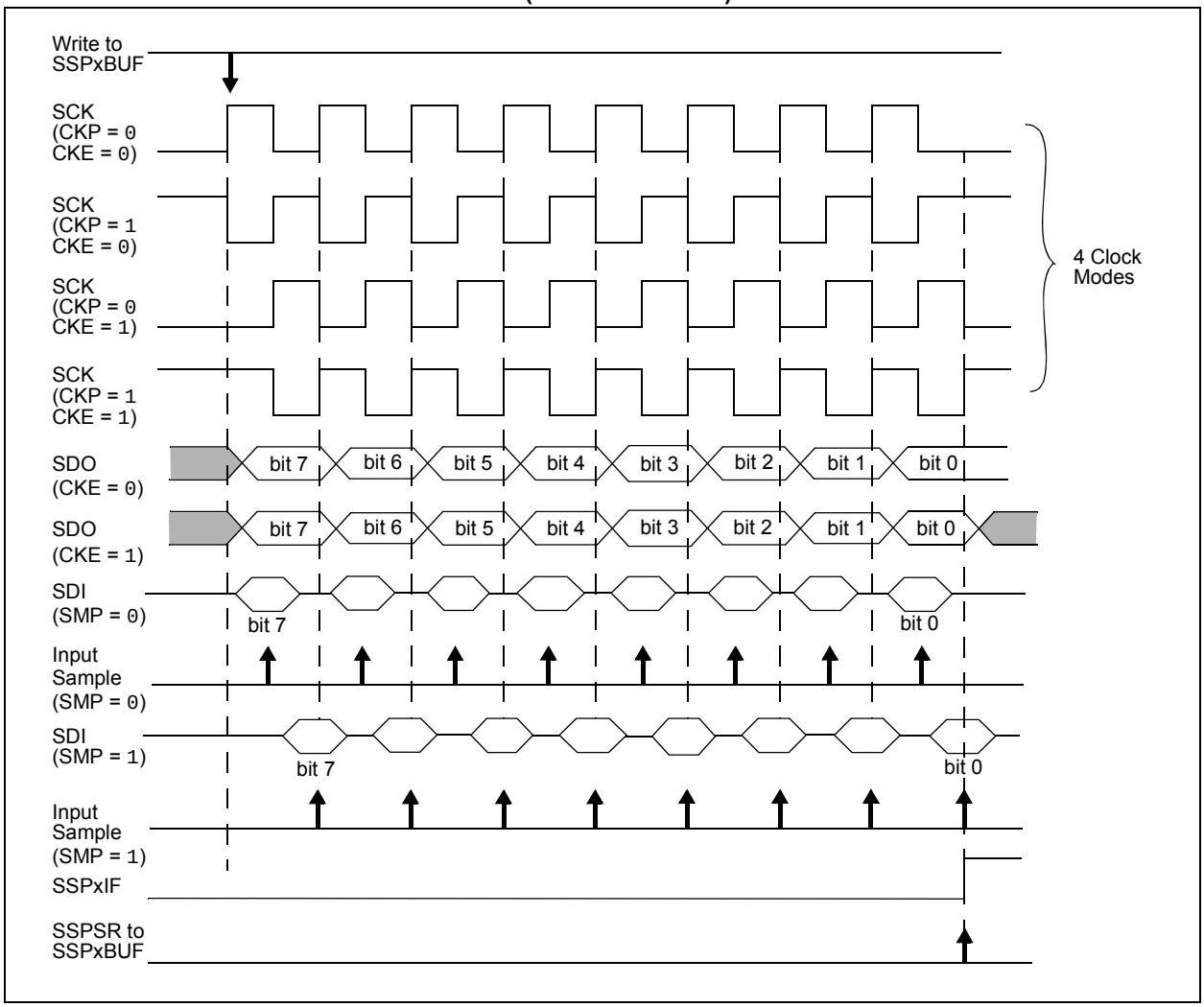


FIGURE 26-4: SPI MODE WAVEFORM (MASTER MODE)



26.5.2 SPI SLAVE MODE

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

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit of the SSPxCON1 register.

While in Slave mode, the external clock is supplied by the external clock source on the SCK 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. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

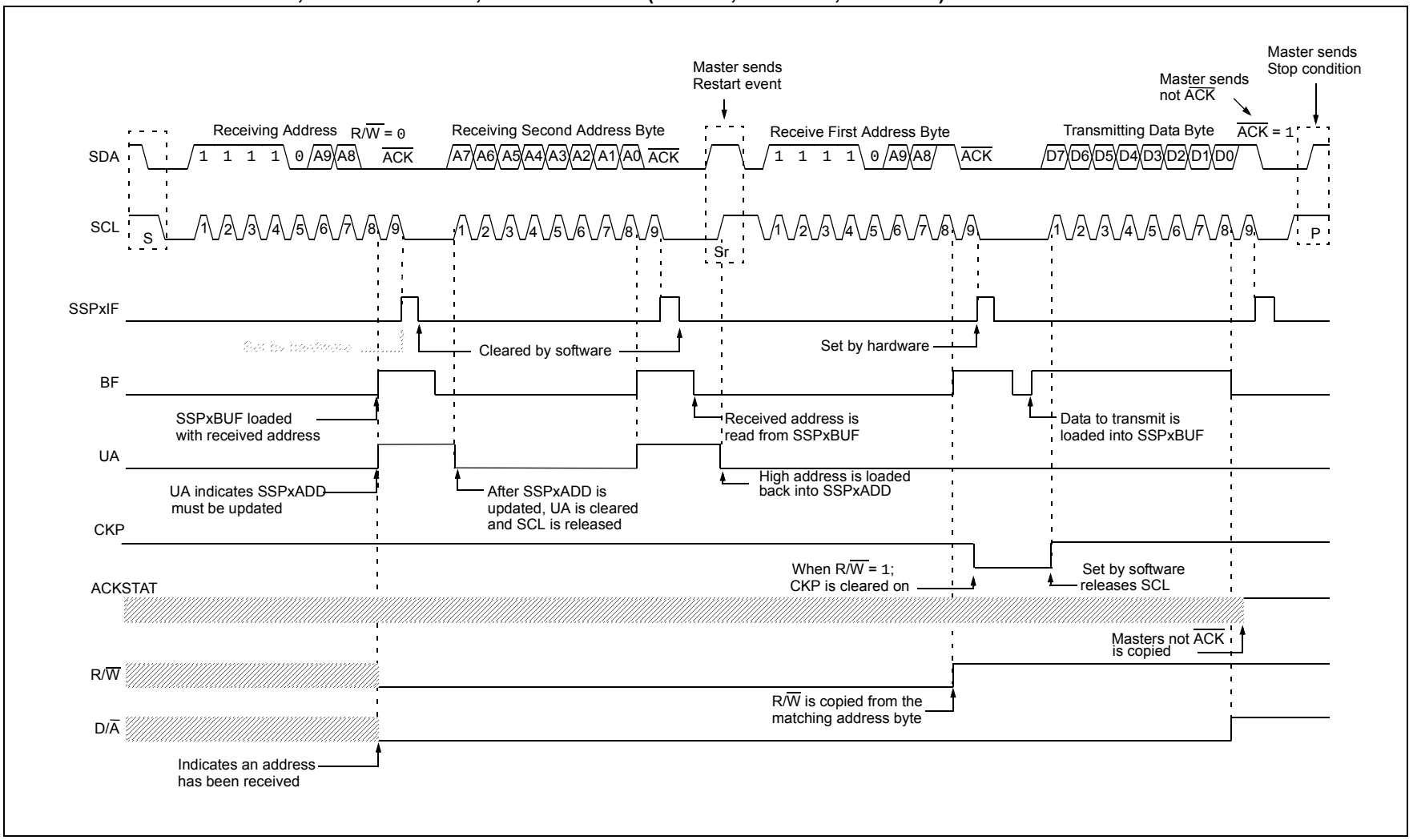
26.5.3 DAISY-CHAIN CONFIGURATION

The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device.

Figure 26-5 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPxCON3 register will enable writes to the SSPxBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

FIGURE 26-22: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)



27.5.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 27.5.1.5 "Synchronous Master Reception"**), with the following exceptions:

- Sleep
- CREN bit is always set, therefore the receiver is never idle
- SREN bit, which is a "don't care" in Slave mode

A character may be received while in Sleep mode by setting the CREN bit prior to entering Sleep. Once the word is received, the RSR register will transfer the data to the RCxREG register. If the RCxIE enable bit is set, the interrupt generated will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will branch to the interrupt vector.

27.5.2.4 Synchronous Slave Reception Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for both the CKx and DTx pins (if applicable).
3. If interrupts are desired, set the RCxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Set the CREN bit to enable reception.
6. The RCxIF bit will be set when reception is complete. An interrupt will be generated if the RCxIE bit was set.
7. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCxSTA register.
8. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCxREG register.
9. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

TABLE 27-10: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BAUDxCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	389
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	166
PIE3	—	—	RC1IE	TX1IE	—	—	BCL1IE	SSP1IE	178
PIR3	—	—	RC1IF	TX1IF	—	—	BCL1IF	SSP1IF	170
IPR3	—	—	RC1IP	TX1IP	—	—	BCL1IP	SSP1IP	186
RCxREG	EUSART Receive Data Register								393*
RCxSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	388
RxyPPS	—	—	—	RxyPPS<4:0>					213
RXxPPS	—	—	—	RXPPS<4:0>					211
TXxSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	387

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous slave reception.
* Page provides register information.

TABLE 35-2: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	k, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	k	Go to address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

CLRF **Clear f**

Syntax: CLRF f {,a}

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

Operation: 000h → f
1 → 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 $f \le 95$ (5Fh). See **Section 35.2.3 "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 → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

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

RCALL **Relative Call**

Syntax: RCALL n

Operands: $-1024 \le n \le 1023$

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1101	1nnn	nnnn	nnnn

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' 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 a 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

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

Example: HERE RCALL Jump

Before Instruction
PC = Address (HERE)
After Instruction
PC = Address (Jump)
TOS = Address (HERE + 2)

RESET **Reset**

Syntax: RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset by software.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: RESET

After Instruction
Registers = Reset Value
Flags* = Reset Value

TABLE 37-25: I²C BUS DATA REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Symbol**	**Characteristic**		**Min.**	**Max.**	**Units**	**Conditions**
SP100*	THIGH	Clock high time	100 kHz mode	4.0	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	μs	Device must operate at a minimum of 10 MHz
			SSP module	1.5TCY	—		
SP101*	TLOW	Clock low time	100 kHz mode	4.7	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	μs	Device must operate at a minimum of 10 MHz
			SSP module	1.5TCY	—		
SP102*	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1CB	300	ns	CB is specified to be from 10-400 pF
SP103*	TF	SDA and SCL fall time	100 kHz mode	—	250	ns	
			400 kHz mode	20 + 0.1CB	250	ns	CB is specified to be from 10-400 pF
SP106*	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
SP107*	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
SP109*	TAA	Output valid from clock	100 kHz mode	—	3500	ns	**(Note 1)**
			400 kHz mode	—	—	ns	
SP110*	TBUF	Bus free time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
SP111	CB	Bus capacitive loading		—	400	pF	

* These parameters are characterized but not tested.

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

2: A Fast mode (400 kHz) I²C bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement TSU:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification), before the SCL line is released.

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