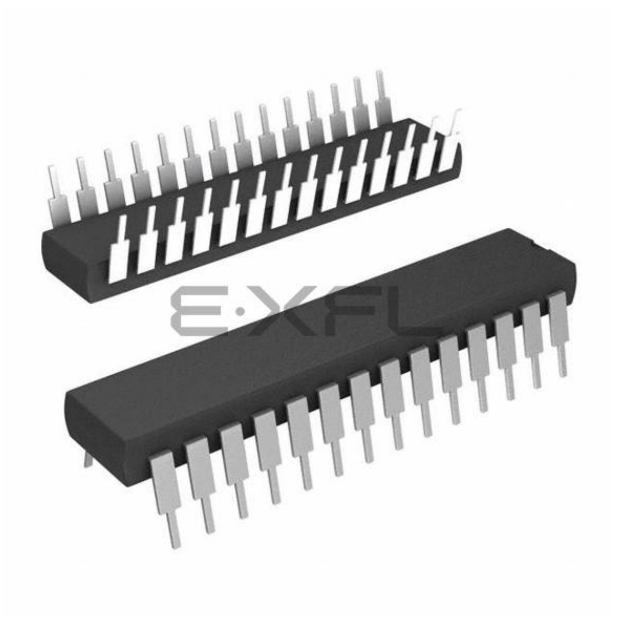


What is "Embedded - Microcontrollers"?

"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	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf24k40-i-sp

REGISTER 3-4: Configuration Word 2H (30 0003h): Supervisor

R/W-1	U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{XINST}}$	—	$\overline{\text{DEBUG}}$	STVREN	PPS1WAY	$\overline{\text{ZCD}}$	BORV<1:0>	
bit 7							bit 0

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

bit 7 **$\overline{\text{XINST}}$:** Extended Instruction Set Enable bit
1 = Extended Instruction Set and Indexed Addressing mode disabled (Legacy mode)
0 = Extended Instruction Set and Indexed Addressing mode enabled

bit 6 **Unimplemented:** Read as ‘1’

bit 5 **$\overline{\text{DEBUG}}$:** Debugger Enable bit
1 = Background debugger disabled
0 = Background debugger enabled

bit 4 **STVREN:** Stack Overflow/Underflow Reset Enable bit
1 = Stack Overflow or Underflow will cause a Reset
0 = Stack Overflow or Underflow will not cause a Reset

bit 3 **PPS1WAY:** PPSLOCKED bit One-Way Set Enable bit
1 = The PPSLOCKED bit can only be set once after an unlocking sequence is executed; once PPSLOCK is set, all future changes to PPS registers are prevented
0 = The PPSLOCKED bit can be set and cleared as needed (provided an unlocking sequence is executed)

bit 2 **$\overline{\text{ZCD}}$:** ZCD Disable bit
1 = ZCD disabled. ZCD can be enabled by setting the ZCDSEN bit of ZCDCON
0 = ZCD always enabled, ZCDMD bit is ignored

bit 1-0 **BORV<1:0>:** Brown-out Reset Voltage Selection bit[(1)]
PIC18F2xK40 device:
11 = Brown-out Reset Voltage (V_{BOR}) set to 2.45V
10 = Brown-out Reset Voltage (V_{BOR}) set to 2.45V
01 = Brown-out Reset Voltage (V_{BOR}) set to 2.7V
00 = Brown-out Reset Voltage (V_{BOR}) set to 2.85V

PIC18LF2xK40 device:
11 = Brown-out Reset Voltage (V_{BOR}) set to 1.90V
10 = Brown-out Reset Voltage (V_{BOR}) set to 2.45V
01 = Brown-out Reset Voltage (V_{BOR}) set to 2.7V
00 = Brown-out Reset Voltage (V_{BOR}) set to 2.85V

Note 1: The higher voltage setting is recommended for operation at or above 16 MHz.

4.3.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision digitally-controlled internal clock source that produces a stable clock up to 64 MHz. The HFINTOSC can be enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits in Configuration Word 1 to '110' (FOSC = 1 MHz) or '000' (FOSC = 64 MHz) to set the oscillator upon device Power-up or Reset.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See **Section 4.4 "Clock Switching"** for more information.

The HFINTOSC frequency can be selected by setting the HFFRQ<3:0> bits of the OSCFRQ register.

The NDIV<3:0> bits of the OSCCON1 register allow for division of the HFINTOSC output from a range between 1:1 and 1:512.

4.3.2.2 MFINTOSC

The module provides two (500 kHz and 31.25 kHz) constant clock outputs. These clocks are digital divisors of the HFINTOSC clock. Dynamic divider logic is used to provide constant MFINTOSC clock rates for all settings of HFINTOSC.

The MFINTOSC cannot be used to drive the system but it is used to clock certain modules such as the Timers and WWDT.

4.3.2.3 Internal Oscillator Frequency Adjustment

The internal oscillator is factory-calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register (Register 4-3).

The default value of the OSCTUNE register is 00h. The value is a 6-bit two's complement number. A value of 1Fh will provide an adjustment to the maximum frequency. A value of 20h will provide an adjustment to the minimum frequency.

When the OSCTUNE register is modified, the oscillator frequency will begin shifting to the new frequency. Code execution continues during this shift. There is no indication that the shift has occurred.

OSCTUNE **does not affect** the LFINTOSC frequency. Operation of features that depend on the LFINTOSC clock source frequency, such as the Power-up Timer (PWRT), WWDT, Fail-Safe Clock Monitor (FSCM) and peripherals, are *not* affected by the change in frequency.

4.3.2.4 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a factory-calibrated 31 kHz internal clock source.

The LFINTOSC is the frequency for the Power-up Timer (PWRT), Windowed Watchdog Timer (WWDT) and Fail-Safe Clock Monitor (FSCM).

The LFINTOSC is enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits of Configuration Word 1 to enable LFINTOSC.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See **Section 4.4, Clock Switching** for more information.

4.3.2.5 ADCRC

The ADCRC is an oscillator dedicated to the ADC^2 module. The ADCRC oscillator can be manually enabled using the ADOEN bit of the OSCEN register. The ADCRC runs at a fixed frequency of 600 kHz. ADCRC is automatically enabled if it is selected as the clock source for the ADC^2 module.

7.5 Register Definitions: Peripheral Module Disable

REGISTER 7-1: PMD0: PMD CONTROL REGISTER 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	R/W-0/0	R/W-0/0
SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
7							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	q = Value depends on condition

bit 7	**SYSCMD:** Disable Peripheral System Clock Network bit(1) See description in **Section 7.4 “System Clock Disable”**. `1` = System clock network disabled (FOSC) `0` = System clock network enabled
bit 6	**FVRMD:** Disable Fixed Voltage Reference bit `1` = FVR module disabled `0` = FVR module enabled
bit 5	**HLVDMD:** Disable Low-Voltage Detect bit `1` = HLVD module disabled `0` = HLVD module enabled
bit 4	**CRCMD:** Disable CRC Engine bit `1` = CRC module disabled `0` = CRC module enabled
bit 3	**SCANMD:** Disable NVM Memory Scanner bit(2) `1` = NVM Memory Scan module disabled `0` = NVM Memory Scan module enabled
bit 2	**NVMMD:** NVM Module Disable bit(3) `1` = All Memory reading and writing is disabled; NVMCON registers cannot be written `0` = NVM module enabled
bit 1	**CLKRMD:** Disable Clock Reference bit `1` = CLKR module disabled `0` = CLKR module enabled
bit 0	**IOCMD:** Disable Interrupt-on-Change bit, All Ports `1` = IOC module(s) disabled `0` = IOC module(s) enabled

Note 1: Clearing the SYSCMD bit disables the system clock (FOSC) to peripherals, however peripherals clocked by FOSC/4 are not affected.

2: Subject to SCANE bit in CONFIG4H.

3: When enabling NVM, a delay of up to 1 μs may be required before accessing data.

10.4.2 ACCESS BANK

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 10-4).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the ‘a’ parameter in the instruction). When ‘a’ is equal to ‘1’, the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When ‘a’ is ‘0’, however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. For 8-bit addresses of 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in **Section 10.7.3 “Mapping the Access Bank in Indexed Literal Offset Mode”**.

10.4.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM, which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

10.4.4 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 the top portion of Bank 15 (F38h to FFFh). A list of these registers is given in Table 10-3 and Table 10-4.

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 a peripheral feature 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.

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F24/25K40 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
EF2h	RB3PPS	—	—	—	RB3PPS<4:0>					`---00000`
EF1h	RB2PPS	—	—	—	RB2PPS<4:0>					`---00000`
EF0h	RB1PPS	—	—	—	RB1PPS<4:0>					`---00000`
EEFh	RB0PPS	—	—	—	RB0PPS<4:0>					`---00000`
EEEh	RA7PPS	—	—	—	RA7PPS<4:0>					`---00000`
EEDh	RA6PPS	—	—	—	RA6PPS<4:0>					`---00000`
EECh	RA5PPS	—	—	—	RA5PPS<4:0>					`---00000`
EEBh	RA4PPS	—	—	—	RA4PPS<4:0>					`---00000`
EEAh	RA3PPS	—	—	—	RA3PPS<4:0>					`---00000`
EE9h	RA2PPS	—	—	—	RA2PPS<4:0>					`---00000`
EE8h	RA1PPS	—	—	—	RA1PPS<4:0>					`---00000`
EE7h	RA0PPS	—	—	—	RA0PPS<4:0>					`---00000`
EE6h	PMD5	—	—	—	—	—	—	—	DSMMD	`-------0`
EE5h	PMD4	—	UART1MD	—	MSSP1MD	—	—	—	CWG1MD	`-0-0---0`
EE4h	PMD3	—	—	—	—	PWM4MD	PWM3MD	CCP2MD	CCP1MD	`----0000`
EE3h	PMD2	—	DACMD	ADCMD	—	—	CMP2MD	CMP1MD	ZCDMD	`-00--000`
EE2h	PMD1	—	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD	`-0000000`
EE1h	PMD0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	`00x00000`
EE0h	BORCON	SBOREN	—	—	—	—	—	—	BORRDY	`1------q`
EDFh	VREGCON[(1)]	—	—	—	—	—	—	VREGPM	Reserved	`------01`
EDEh	OSCFRQ	—	—	—	—	HFFRQ<3:0>				`----1111`
EDDh	OSCTUNE	—	—	HFTUN<5:0>						`--100000`
EDCh	OSCEN	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN	—	—	`000000--`
EDBh	OSCSTAT	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR	—	PLLR	`qqqqqq-q`
EDAh	OSCCON3	CSWHOLD	SOSCPWR	—	ORDY	NOSCR	—	—	—	`00-00---`
ED9h	OSCCON2	—	COSC<2:0>			CDIV<3:0>				`-qqqqqqq`

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not available on LF devices.

REGISTER 11-2: NVMCON2: NONVOLATILE MEMORY CONTROL 2 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
NVMCON2<7:0>							
bit 7							bit 0

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

bit 7-0	**NVMCON2<7:0>:** Refer to **Section 11.1.4 “NVM Unlock Sequence”**.

Note 1: This register always reads zeros, regardless of data written.

Register 11-3: NVMADRL: Data EEPROM Memory Address Low

R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0
NVMADR<7:0>							
bit 7							bit 0

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

bit 7-0	**NVMADR<7:0>:** EEPROM Read Address bits

REGISTER 11-4: NVMADRH: DATA EEPROM MEMORY ADDRESS HIGH(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-x/u	R/W-x/u
—	—	—	—	—	—	NVMADR<9:8>	
bit 7							bit 0

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

bit 7-2	**Unimplemented:** Read as ‘0’
bit 1-0	**NVMADR<9:8>:** EEPROM Read Address bits

Note 1: The NVMADRH register is not implemented on PIC18(L)F24/25K40.

14.8 Register Definitions: Interrupt Control

REGISTER 14-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1
GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG
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 **GIE/GIEH:** Global Interrupt Enable bit
If IPEN = 1:
1 = Enables all unmasked interrupts and cleared by hardware for high-priority interrupts only
0 = Disables all interrupts
If IPEN = 0:
1 = Enables all unmasked interrupts and cleared by hardware for all interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
If IPEN = 1:
1 = Enables all low-priority interrupts and cleared by hardware for low-priority interrupts only
0 = Disables all low-priority interrupts
If IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts

bit 4-3 **Unimplemented**: Read as ‘0’

bit 2 **INT2EDG:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge of INT2 pin
0 = Interrupt on falling edge of INT2 pin

bit 1 **INT1EDG:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge of INT1 pin
0 = Interrupt on falling edge of INT1 pin

bit 0 **INT0EDG:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge of INT0 pin
0 = Interrupt on falling edge of INT0 pin

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

21.5.2 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for standard PWM operation:

1. Use the desired output pin RxyPPS control to select CCPx as the source and disable the CCPx pin output driver by setting the associated TRIS bit.
2. Load the PR2 register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register, and the CCPRxH register with the PWM duty cycle value and configure the FMT bit of the CCPxCON register to set the proper register alignment.
5. Configure and start Timer2:
    - Clear the TMR2IF interrupt flag bit of the PIR4 register. See Note below.
    - Select the timer clock source to be as FOSC/4 using the TxCLKCON register. This is required for correct operation of the PWM module.
    - Configure the T2CKPS bits of the T2CON register with the Timer prescale value.
    - Enable the Timer by setting the ON bit of the T2CON register.
6. Enable PWM output pin:
    - Wait until the Timer overflows and the TMR2IF bit of the PIR4 register is set. See Note below.
    - Enable the CCPx pin output driver by clearing the associated TRIS bit.

Note:	In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

21.5.3 TIMER2 TIMER RESOURCE

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

21.5.4 PWM PERIOD

The PWM period is specified by the PR2 register of Timer2. The PWM period can be calculated using the formula of Equation 21-1.

EQUATION 21-1: PWM PERIOD

$$PWM\ Period = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

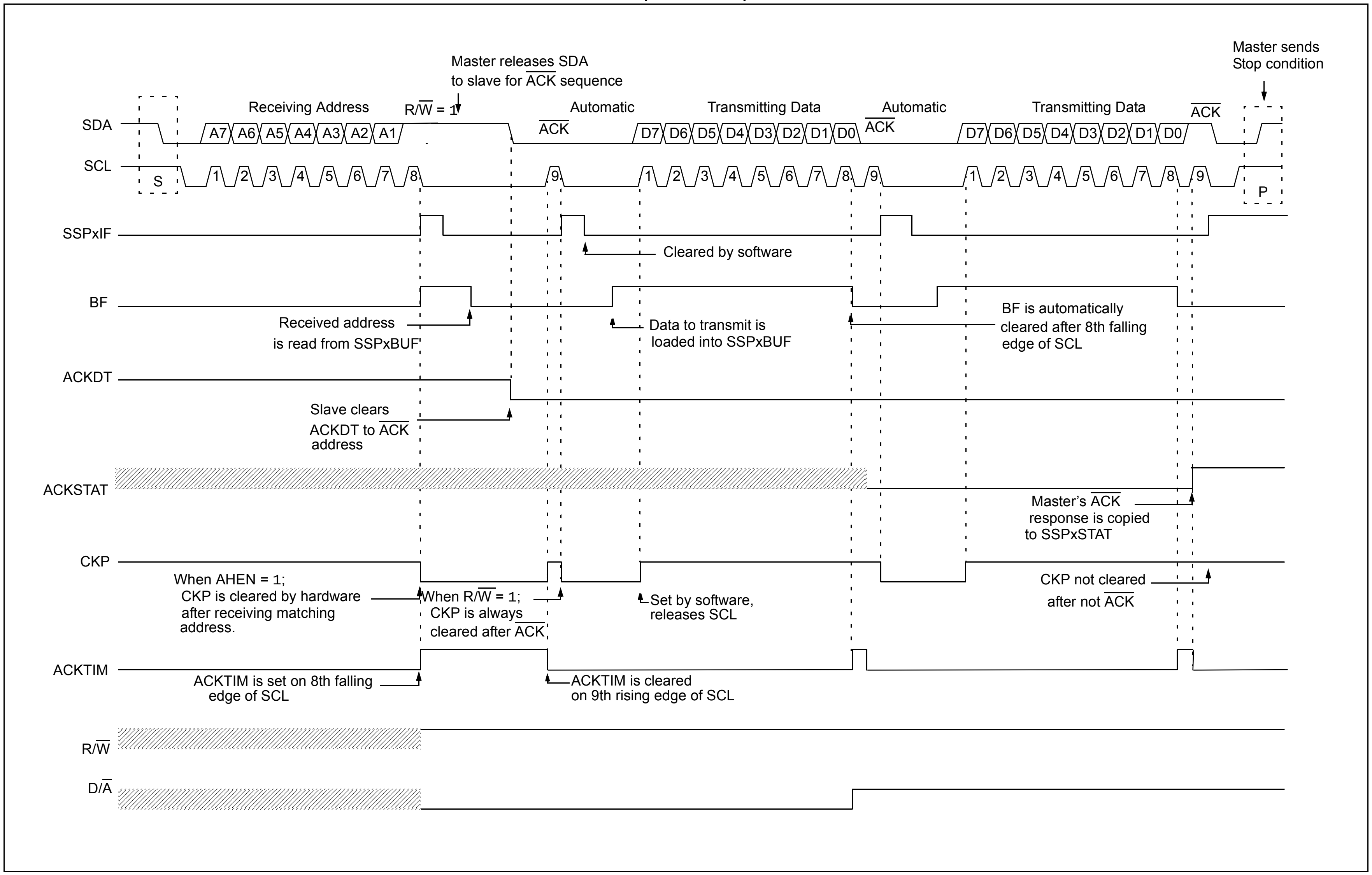
Note 1: TOSC = 1/FOSC

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is transferred from the CCPRxL/H register pair into a 10-bit buffer.

Note:	The Timer postscaler (see **Section 20.4 “Timer2 Interrupt”**) is not used in the determination of the PWM frequency.

FIGURE 26-19: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)



26.10.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPxCON2 register. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode (Figure 26-30).

26.10.8.1 WCOL Status Flag

If the user writes the SSPxBUF when an Acknowledge sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

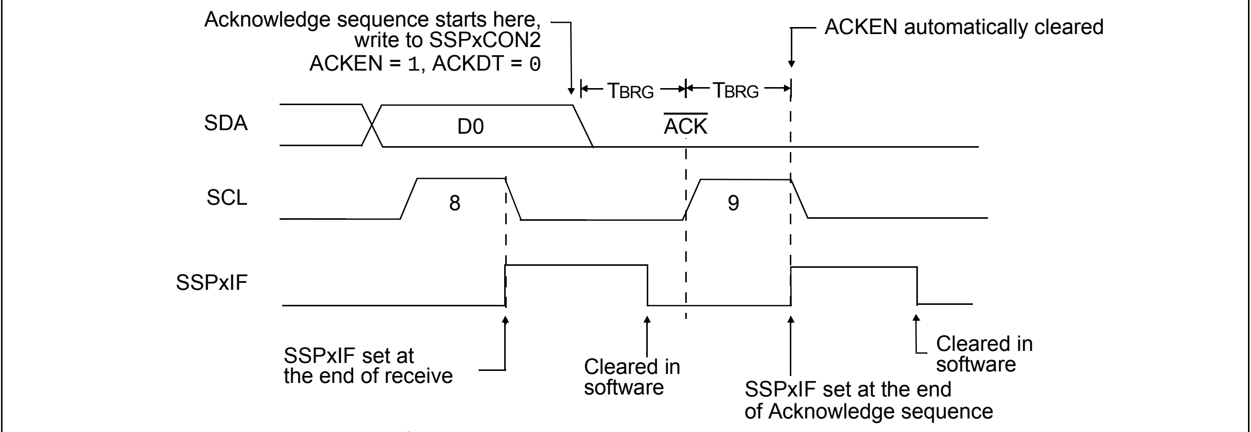
26.10.9 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPxCON2 register. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit of the SSPxSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPxIF bit is set (Figure 26-31).

26.10.9.1 WCOL Status Flag

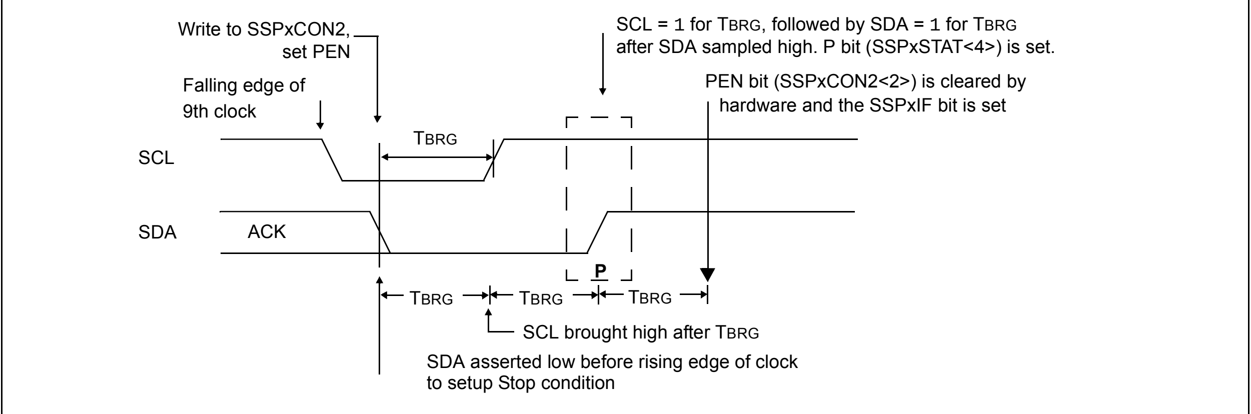
If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 26-30: ACKNOWLEDGE SEQUENCE WAVEFORM


Acknowledge sequence starts here, write to SSPxCON2 ACKEN = 1, ACKDT = 0
ACKEN automatically cleared
TBRG
TBRG
SDA
D0
ACK
SCL
8
9
SSPxIF
SSPxIF set at the end of receive
Cleared in software
SSPxIF set at the end of Acknowledge sequence
Cleared in software

Note: TBRG = one Baud Rate Generator period.

FIGURE 26-31: STOP CONDITION RECEIVE OR TRANSMIT MODE


Write to SSPxCON2, set PEN
Falling edge of 9th clock
SCL = 1 for TBRG, followed by SDA = 1 for TBRG after SDA sampled high. P bit (SSPxSTAT<4>) is set.
PEN bit (SSPxCON2<2>) is cleared by hardware and the SSPxIF bit is set
TBRG
SCL
SDA
ACK
P
TBRG
TBRG
TBRG
SCL brought high after TBRG
SDA asserted low before rising edge of clock to setup Stop condition

Note: TBRG = one Baud Rate Generator period.

27.2.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error Status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCxSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCxREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCxSTA register which resets the EUSART. Clearing the CREN bit of the RCxSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

Note: If all receive characters in the receive FIFO have framing errors, repeated reads of the RCxREG will not clear the FERR bit.

27.2.2.5 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCxSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCxSTA register or by resetting the EUSART by clearing the SPEN bit of the RCxSTA register.

27.2.2.6 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCxSTA register is set the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCxSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCxREG.

27.2.2.7 Address Detection

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems. Address detection is enabled by setting the ADDEN bit of the RCxSTA register.

Address detection requires 9-bit character reception. When address detection is enabled, only characters with the ninth data bit set will be transferred to the receive FIFO buffer, thereby setting the RCxIF interrupt bit. All other characters will be ignored.

Upon receiving an address character, user software determines if the address matches its own. Upon address match, user software must disable address detection by clearing the ADDEN bit before the next Stop bit occurs. When user software detects the end of the message, determined by the message protocol used, software places the receiver back into the Address Detection mode by setting the ADDEN bit.

FIGURE 31-1: ADC^2 BLOCK DIAGRAM

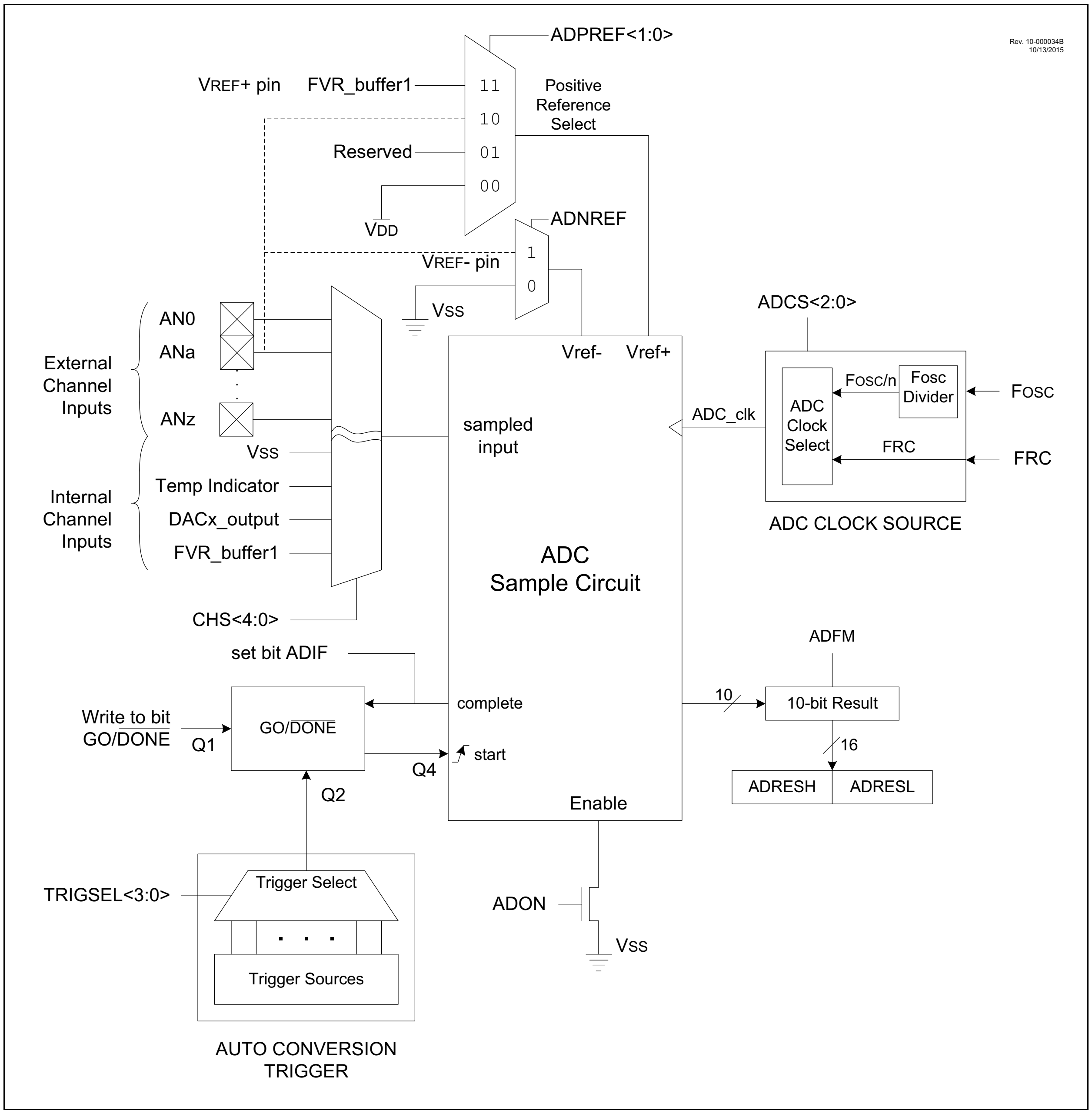
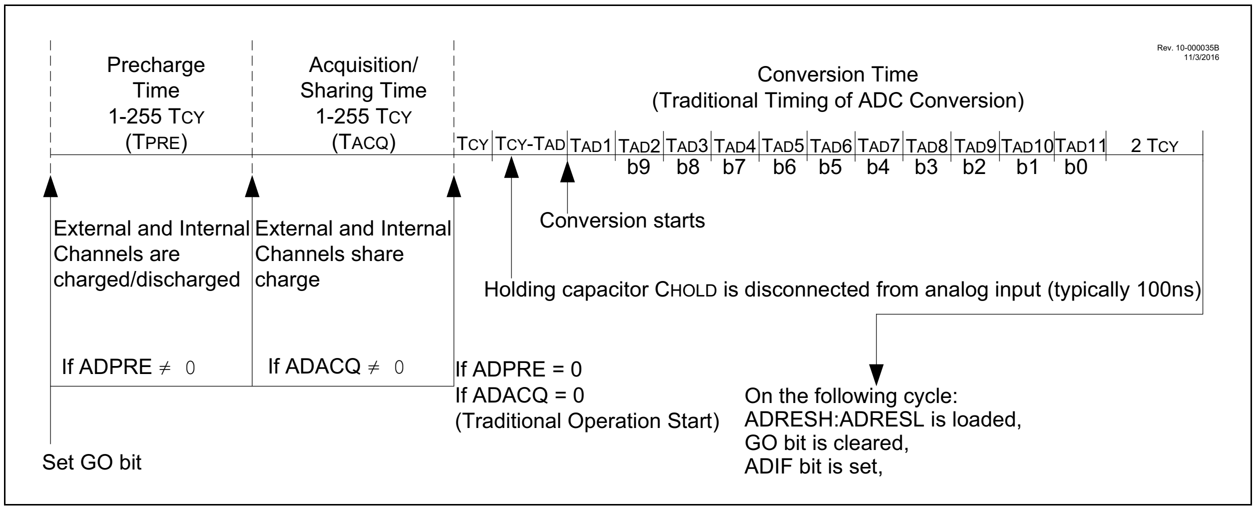


TABLE 31-1: ADC CLOCK PERIOD (TAD) VS. DEVICE OPERATING FREQUENCIES[1,4]

ADC Clock Period (TAD)		Device Frequency (FOSC)						
ADC Clock Source	**ADCS<5:0>**	**64 MHz**	**32 MHz**	**20 MHz**	**16 MHz**	**8 MHz**	**4 MHz**	**1 MHz**
FOSC/2	`000000`	31.25 ns[2]	62.5 ns[2]	100 ns[2]	125 ns[2]	250 ns[2]	500 ns[2]	2.0 µs
FOSC/4	`000001`	62.5 ns[2]	125 ns[2]	200 ns[2]	250 ns[2]	500 ns[2]	1.0 µs	4.0 µs
FOSC/6	`000010`	125 ns[2]	187.5 ns[2]	300 ns[2]	375 ns[2]	750 ns[2]	1.5 µs	6.0 µs
FOSC/8	`000011`	187.5 ns[2]	250 ns[2]	400 ns[2]	500 ns[2]	1.0 µs	2.0 µs	8.0 µs[3]
...	...	...	...	...	...	...	...	...
FOSC/16	`000100`	250 ns[2]	500 ns[2]	800 ns[2]	1.0 µs	2.0 µs	4.0 µs	16.0 µs[3]
...	...	...	...	...	...	...	...	...
FOSC/128	`111111`	2.0 µs	4.0 µs	6.4 µs	8.0 µs	16.0 µs[3]	32.0 µs[2]	128.0 µs[2]
FRC	ADCS(ADCON0<4>) = `1`	1.0-6.0 µs	1.0-6.0 µs	1.0-6.0 µs	1.0-6.0 µs	1.0-6.0 µs	1.0-6.0 µs	1.0-6.0 µs

Legend: Shaded cells are outside of recommended range.
Note 1: See TAD parameter for FRC source typical TAD value.
2: These values violate the required TAD time.
3: Outside the recommended TAD time.
4: The ADC clock period (TAD) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the FRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

FIGURE 31-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES



31.2.6 ADC CONVERSION PROCEDURE (BASIC MODE)

This is an example procedure for using the ADC to perform an Analog-to-Digital conversion:

1. Configure Port:
   - Disable pin output driver (Refer to the TRISx register)
   - Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   - Select ADC conversion clock
   - Configure voltage reference
   - Select ADC input channel (precharge+acquisition)
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt (PEIE bit)
   - Enable global interrupt (GIE bit)(1)
4. If ADACQ = 0, software must wait the required acquisition time(2).
5. Start conversion by setting the ADGO bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the ADGO bit
   - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

2: Refer to **Section 31.3 “ADC Acquisition Requirements”**.

EXAMPLE 31-1: ADC CONVERSION

```
;This code block configures the ADC
;for polling, VDD and VSS references, FRC
;oscillator and AN0 input.
;
;Conversion start & polling for completion
;are included.
;
BANKSEL   ADCON1       ;
MOVLW     B’11110000’  ;Right justify,
                       ;FRC oscillator
MOVWF     ADCON1       ;Vdd and Vss Vref
BANKSEL   TRISA        ;
BSF       TRISA,0      ;Set RA0 to input
BANKSEL   ANSEL        ;
BSF       ANSEL,0      ;Set RA0 to analog
BANKSEL   ADCON0       ;
MOVLW     B’00000001’  ;Select channel AN0
MOVWF     ADCON0       ;Turn ADC On
CALL      SampleTime   ;Acquisiton delay
BSF       ADCON0,ADGO  ;Start conversion
BTFSC     ADCON0,ADGO  ;Is conversion done?
GOTO      $-1          ;No, test again
BANKSEL   ADRESH       ;
MOVF      ADRESH,W     ;Read upper 2 bits
MOVWF     RESULTHI     ;store in GPR space
BANKSEL   ADRESL       ;
MOVF      ADRESL,W     ;Read lower 8 bits
MOVWF     RESULTLO     ;Store in GPR space
```

REGISTER 31-27: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ADERR<7:0>							
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 7-0 **ADERR<7:0>**: ADC Setpoint Error LSB. Lower byte of ADC Setpoint Error calculation is determined by ADCALC bits of ADCON3, see Register 23-1 for more details.

REGISTER 31-28: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ADLTH<15:8>							
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 7-0 **ADLTH<15:8>**: ADC Lower Threshold MSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 31-29: ADLTHL: ADC LOWER THRESHOLD LOW BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ADLTH<7:0>							
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 7-0 **ADLTH<7:0>**: ADC Lower Threshold LSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 31-30: ADUTHH: ADC UPPER THRESHOLD HIGH BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ADUTH<15:8>							
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 7-0	**ADUTH<15:8>**: ADC Upper Threshold MSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 31-31: ADUTHL: ADC UPPER THRESHOLD LOW BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ADUTH<7:0>							
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 7-0	**ADUTH<7:0>**: ADC Upper Threshold LSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

BZ — Branch if Zero

Syntax:	BZ n
Operands:	-128 ≤ n ≤ 127
Operation:	if ZERO bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None

Encoding:

`1110`	`0000`	`nnnn`	`nnnn`

Description:	If the ZERO 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 BZ Jump`

Before Instruction
- PC = address (`HERE`)

After Instruction
- If ZERO = 1;
- PC = address (`Jump`)
- If ZERO = 0;
- PC = address (`HERE + 2`)

CALL — Subroutine Call

Syntax:	CALL k {,s}
Operands:	0 ≤ k ≤ 1048575 s ∈ [0,1]
Operation:	(PC) + 4 → TOS, k → PC<20:1>, if s = `1` (W) → WS, (Status) → STATUSS, (BSR) → BSRS
Status Affected:	None

Encoding:

1st word (k<7:0>)	`1110`	`110s`	k_7`kkk`	`kkkk`$_0$
2nd word(k<19:8>)	`1111`	k_{19}`kkk`	`kkkk`	`kkkk`$_8$

Description:	Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = `1`, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = `0`, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. `CALL` is a 2-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE, 1`

Before Instruction
- PC = address (`HERE`)

After Instruction
- PC = address (`THERE`)
- TOS = address (`HERE + 4`)
- WS = W
- BSRS = BSR
- STATUSS = Status

MULLW — Multiply literal with W

Syntax:	MULLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) x k → PRODH:PRODL
Status Affected:	None
Encoding:	`0000` `1101` `kkkk` `kkkk`
Description:	An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged. None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: `MULLW 0C4h`

Before Instruction

W	=	E2h
PRODH	=	?
PRODL	=	?

After Instruction

W	=	E2h
PRODH	=	ADh
PRODL	=	08h

MULWF — Multiply W with f

Syntax:	MULWF f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) x (f) → PRODH:PRODL
Status Affected:	None
Encoding:	`0000` `001a` `ffff` `ffff`
Description:	An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged. None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. 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 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 registers PRODH: PRODL

Example: `MULWF REG, 1`

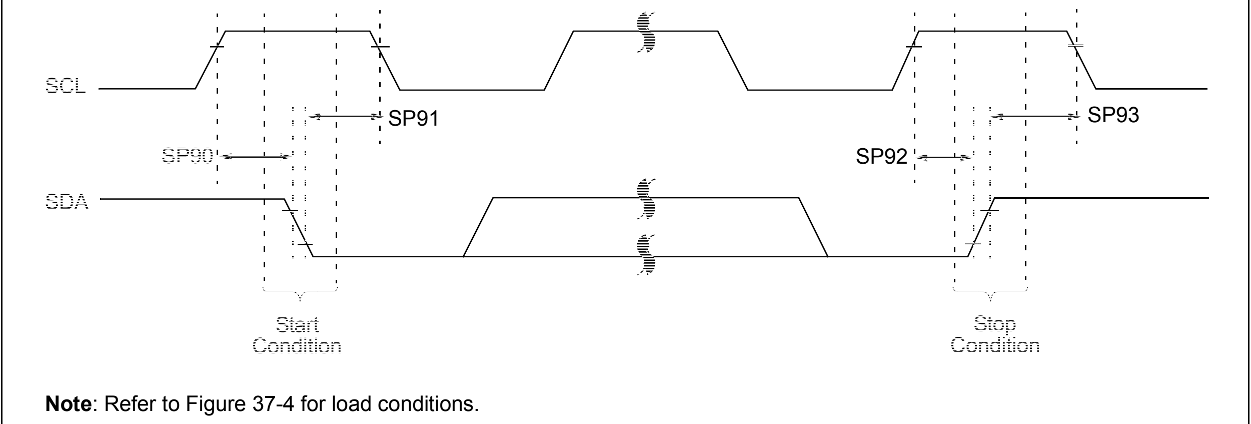
Before Instruction

W	=	C4h
REG	=	B5h
PRODH	=	?
PRODL	=	?

After Instruction

W	=	C4h
REG	=	B5h
PRODH	=	8Ah
PRODL	=	94h

FIGURE 37-20: I^{2}C BUS START/STOP BITS TIMING



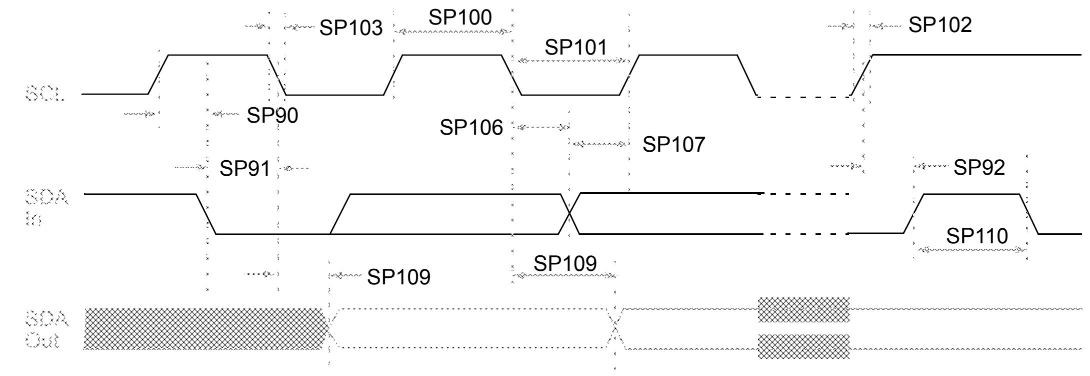
Note: Refer to Figure 37-4 for load conditions.

TABLE 37-24: I^{2}C BUS START/STOP BITS REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)								
Param No.	**Symbol**	**Characteristic**		**Min.**	**Typ**	**Max.**	**Units**	**Conditions**
SP90*	TSU:STA	Start condition	100 kHz mode	4700	—	—	ns	Only relevant for Repeated Start condition
		Setup time	400 kHz mode	600	—	—		
SP91*	THD:STA	Start condition	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
		Hold time	400 kHz mode	600	—	—		
SP92*	TSU:STO	Stop condition	100 kHz mode	4700	—	—	ns	
		Setup time	400 kHz mode	600	—	—		
SP93	THD:STO	Stop condition	100 kHz mode	4000	—	—	ns	
		Hold time	400 kHz mode	600	—	—		

* These parameters are characterized but not tested.

FIGURE 37-21: I^{2}C BUS DATA TIMING



Note: Refer to Figure 37-4 for load conditions.

38.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

Graphs and tables are not available at this time.