Section 25. 12-Bit Analog-to-Digital Converter (ADC) with Threshold Detect

HIGHLIGHTS

This section of the manual contains the following major topics:

Note: This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all PIC32 devices.

Please consult the note at the beginning of the **"Analog-to-Digital Converter"** chapter in the device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: http://www.microchip.com.

25.1 INTRODUCTION

The PIC32 12-Bit A/D Converter with Threshold Detect includes the following features:

- Successive Approximation Register (SAR) conversion
- Conversion speeds of up to 200 ksps
- User-selectable resolution of 10 bits is available
- Up to 32 analog inputs (internal and external)
- External voltage reference input pins
- Unipolar differential Sample-and-Hold Amplifier (SHA)
- Automated threshold scan and compare operation to pre-evaluate conversion results
- Selectable conversion trigger source
- Fixed-length configurable conversion result buffer
- Eight options for result alignment and encoding
- Configurable interrupt generation
- Operation during CPU Sleep and Idle modes

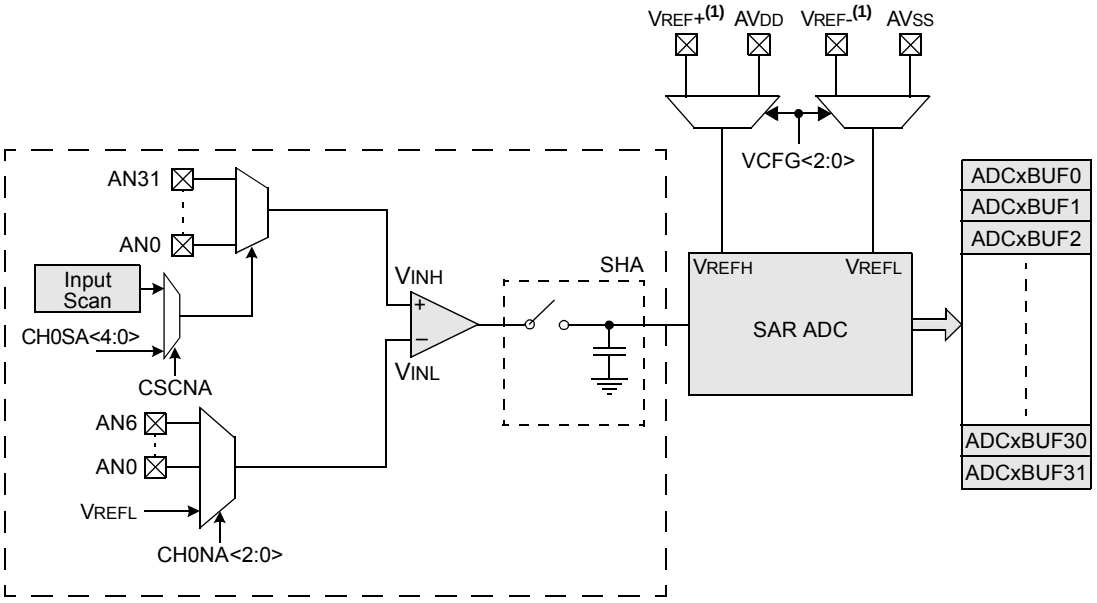
Figure 25-1 illustrates a block diagram of the 12-bit ADC. The 12-bit ADC can have up to 32 analog input pins, AN0 through AN31. In addition, there are two analog input pins for external voltage reference connections. These voltage reference inputs may be shared with other analog input pins and may be common to other analog module references. The actual number of analog input pins and the external voltage reference input configuration will depend on the specific PIC32 device. Refer to the specific device data sheet for more information.

The analog inputs are connected through a multiplexer to the SHA. Unipolar differential conversions are possible on all inputs (see Figure 25-1).

The Automatic Input Scan mode sequentially converts multiple analog inputs. A special control register specifies which inputs will be included in the scanning sequence.

The 12-bit ADC is connected to a result buffer. The 12-bit result is converted to one of eight output formats in either 32-bit or 16-bit word widths.

Figure 25-1: ADC Block Diagram



Note 1: VREF+ and VREF- inputs can be multiplexed with other analog inputs.

25.2 CONTROL REGISTERS

The ADC module has the following Special Function Registers (SFRs):

- **ADxCON1: ADCx Control Register 1**
- **ADxCON2: ADCx Control Register 2**
- **ADxCON3: ADCx Control Register 3**
- **ADxCON5: ADCx Control Register 5**

  The ADxCON1, ADxCON2, ADxCON3 and ADxCON5 registers control the operation of the ADC module.
- **ADxCHS: ADCx Input Select Register**

  The ADxCHS register selects the input pins to be connected to the SHA.
- **ADxCSS: ADCx Input Scan Select Register(1,2)**

  The ADxCSS register selects inputs to be sequentially scanned.
- **ADxCHIT: ADCx Compare Hit Register(1)**

  The ADxCHIT register indicates the analog inputs meeting specified comparison requirements.

Table 25-1 provides a summary of all ADC module-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register. All unimplemented registers and/or bits within a register read as zero.

Table 25-1: ADC SFR Summary

Register Name(1)		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
ADxCON1	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ON	r	SIDL	—	—	FORM<2:0>		
	7:0	SSRC<3:0>				MODE12	ASAM	SAMP	DONE
ADxCON2	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	VCFG<2:0>			OFFCAL	BUFREGEN	CSCNA	—	—
	7:0	BUFS	SMPI<4:0>					BUFM	—
ADxCON3	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ADRC	EXTSAM	—	SAMC<4:0>				
	7:0	ADCS<7:0>							
ADxCON5	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ASEN	LPEN	—	BGREQ	—	—	ASINT<1:0>	
	7:0	—	—	—	—	WM<1:0>		CM<1:0>	
ADxCHS	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	CH0NA<2:0>			CH0SA<4:0>				
ADxCSS	31:24	CSS<31:24>							
	23:16	CSS<23:16>							
	15:8	CSS<15:8>							
	7:0	CSS<7:0>							

Legend: r = Reserved bit; — = unimplemented, read as '0'.

Note 1: All registers have associated Clear, Set and Invert registers at an offset of 0x4, 0x8 and 0xC bytes, respectively. These registers have the same name with CLR, SET or INV appended to the end of the register name (e.g., ADxCON1CLR). Writing a '1' to any bit position in these registers will clear, set or invert valid bits in the associated register. Reads from these registers should be ignored.

Table 25-1: ADC SFR Summary (Continued)

Register Name(1)		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
ADxCHIT	31:24	CHH<31:24>							
	23:16	CHH<23:16>							
	15:8	CHH<15:8>							
	7:0	CHH<7:0>							
ADCxBUF0	31:0	ADCx Result Word 0							
ADCxBUF1	31:0	ADCx Result Word 1							
ADCxBUF2	31:0	ADCx Result Word 2							
ADCxBUF3	31:0	ADCx Result Word 3							
ADCxBUF4	31:0	ADCx Result Word 4							
ADCxBUF5	31:0	ADCx Result Word 5							
ADCxBUF6	31:0	ADCx Result Word 6							
ADCxBUF7	31:0	ADCx Result Word 7							
ADCxBUF8	31:0	ADCx Result Word 8							
ADCxBUF9	31:0	ADCx Result Word 9							
ADCxBUF10	31:0	ADCx Result Word 10							
ADCxBUF11	31:0	ADCx Result Word 11							
ADCxBUF12	31:0	ADCx Result Word 12							
ADCxBUF13	31:0	ADCx Result Word 13							
ADCxBUF14	31:0	ADCx Result Word 14							
ADCxBUF15	31:0	ADCx Result Word 15							
ADCxBUF16	31:0	ADCx Result Word 16							
ADCxBUF17	31:0	ADCx Result Word 17							
ADCxBUF18	31:0	ADCx Result Word 18							
ADCxBUF19	31:0	ADCx Result Word 19							
ADCxBUF20	31:0	ADCx Result Word 20							
ADCxBUF12	31:0	ADCx Result Word 21							
ADCxBUF22	31:0	ADCx Result Word 22							
ADCxBUF23	31:0	ADCx Result Word 23							
ADCxBUF24	31:0	ADCx Result Word 24							
ADCxBUF25	31:0	ADCx Result Word 25							
ADCxBUF26	31:0	ADCx Result Word 26							
ADCxBUF27	31:0	ADCx Result Word 27							
ADCxBUF28	31:0	ADCx Result Word 28							
ADCxBUF29	31:0	ADCx Result Word 29							
ADCxBUF30	31:0	ADCx Result Word 30							
ADCxBUF31	31:0	ADCx Result Word 31							

Legend: r = Reserved bit; — = unimplemented, read as '0'.

Note 1: All registers have associated Clear, Set and Invert registers at an offset of 0x4, 0x8 and 0xC bytes, respectively. These registers have the same name with CLR, SET or INV appended to the end of the register name (e.g., ADxCON1CLR). Writing a '1' to any bit position in these registers will clear, set or invert valid bits in the associated register. Reads from these registers should be ignored.

Register 25-1: ADxCON1: ADCx Control Register 1

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	r-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
	ON	—	SIDL	—	—	FORM<2:0>		
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0, HSC	R/C-0, HSC
	SSRC<3:0>(1)				MODE12	ASAM	SAMP(3)	DONE(2)

Legend:	r = Reserved bit	HSC = Hardware Settable/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 C = Clearable bit

bit 31-16 **Unimplemented:** Read as '0'

bit 15 **ON:** ADC Operating Mode bit
1 = ADC module is operating
0 = ADC is off

bit 14 **Reserved:** Used by the debugger

bit 13 **SIDL:** ADC Stop in Idle Mode bit
1 = Discontinues module operation when device enters Idle mode
0 = Continues module operation in Idle mode

bit 12-11 **Unimplemented:** Read as '0'

bit 10-8 **FORM<2:0>:** Data Output Format bits

For 12-Bit Operation (MODE12 bit = 1):
111 = Signed fractional 32-bit (DOUT = sddd dddd dddd 0000 0000 0000 0000)
110 = Fractional 32-bit (DOUT = dddd dddd dddd 0000 0000 0000 0000 0000)
101 = Signed Integer 32-bit (DOUT = ssss ssss ssss ssss ssss sddd dddd dddd)
100 = Integer 32-bit (DOUT = 0000 0000 0000 0000 0000 dddd dddd dddd)
011 = Signed fractional 16-bit (DOUT = 0000 0000 0000 0000 sddd dddd dddd 0000)
010 = Fractional 16-bit (DOUT = 0000 0000 0000 0000 dddd dddd dddd 0000)
001 = Signed integer 16-bit (DOUT = 0000 0000 0000 0000 ssss sddd dddd dddd)
000 = Integer 16-bit (DOUT = 0000 0000 0000 0000 0000 dddd dddd dddd)

For 10-Bit Operation (MODE12 bit = 0):
111 = Signed fractional 32-bit (DOUT = sddd dddd dd00 0000 0000 0000 0000)
110 = Fractional 32-bit (DOUT = dddd dddd dd00 0000 0000 0000 0000 0000)
101 = Signed integer 32-bit (DOUT = ssss ssss ssss ssss ssss sssd dddd dddd)
100 = Integer 32-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)
011 = Signed fractional 16-bit (DOUT = 0000 0000 0000 0000 sddd dddd dd00 0000)
010 = Fractional 16-bit (DOUT = 0000 0000 0000 0000 dddd dddd dd00 0000)
001 = Signed integer 16-bit (DOUT = 0000 0000 0000 0000 ssss sssd dddd dddd)
000 = Integer 16-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)

Note 1: The trigger options listed in this document are often used, but the trigger sources are device-specific and can be different from device to device. Refer to the particular device data sheet for the trigger sources available.

2: The DONE bit is not persistent in Automatic modes. It is cleared by hardware at the beginning of the next sample.

3: The SAMP bit is cleared and cannot be written if the ADC is disabled (ON bit = 0).

Register 25-1: ADxCON1: ADCx Control Register 1 (Continued)

bit 7-4 **SSRC<3:0>:** Conversion Trigger Source Select bits(1)
1111-1101 = Reserved
1100 = CLC2 module event ends sampling and starts conversion
1011 = CLC1 module event ends sampling and starts conversion
1010 = SCCP3 module event ends sampling and starts conversion
1001 = SCCP2 module event ends sampling and starts conversion
1000 = MCCP1 module event ends sampling and starts conversion
0111 = Internal counter ends sampling and starts conversion (auto-convert)
0110 = Timer1 period match ends sampling and starts conversion (can trigger during Sleep mode)
0101 = Timer1 period match ends sampling and starts conversion (will not trigger during Sleep mode)
0100-0010 = Reserved
0001 = Active transition on INT0 pin ends sampling and starts conversion
0000 = Clearing the SAMP bit ends sampling and starts conversion

bit 3 **MODE12:** 12-Bit Operation Mode bit
1 = 12-bit ADC operation
0 = 10-bit ADC operation

bit 2 **ASAM:** ADC Sample Auto-Start bit
1 = Sampling begins immediately after last conversion completes; SAMP bit is automatically set
0 = Sampling begins when SAMP bit is set

bit 1 **SAMP:** ADC Sample Enable bit(3)
1 = The ADC Sample-and-Hold Amplifier (SHA) is sampling
0 = The ADC Sample-and-Hold Amplifier is holding

bit 0 **DONE:** Analog-to-Digital Conversion Status bit(2)
1 = Analog-to-Digital conversion is done
0 = Analog-to-Digital conversion is not done or has not started
Clearing this bit will not affect any operation in progress.

Note 1: The trigger options listed in this document are often used, but the trigger sources are device-specific and can be different from device to device. Refer to the particular device data sheet for the trigger sources available.

2: The DONE bit is not persistent in Automatic modes. It is cleared by hardware at the beginning of the next sample.

3: The SAMP bit is cleared and cannot be written if the ADC is disabled (ON bit = 0).

Register 25-2: ADxCON2: ADCx Control Register 2

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
	VCFG<2:0>			OFFCAL	BUFREGEN(2)	CSCNA	—	—
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
	BUFS	SMPI<4:0>(1)					BUFM	—

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 31-16 **Unimplemented:** Read as '0'

bit 15-13 **VCFG<2:0>:** Voltage Reference Configuration bits

VCFG<2:0>	VREFH	VREFL
000	AVDD	AVSS
001	AVDD	External VREF- Pin
010	External VREF+ Pin	AVSS
011	External VREF+ Pin	External VREF- Pin
1xx	Unimplemented; do not use	

bit 12 **OFFCAL:** Input Offset Calibration Mode Select bit

1 = Enables Offset Calibration mode: VINH and VINL of the SHA are connected to VREFL
0 = Disables Offset Calibration mode: The inputs to the SHA are controlled by ADxCHS or ADxCSS

bit 11 **BUFREGEN:** ADC Buffer Register Enable bit(2)

1 = Conversion result is loaded into the buffer location determined by the converted analog input
0 = ADC result buffer is treated as a FIFO

bit 10 **CSCNA:** Scan Mode bit

1 = Scans inputs
0 = Does not scan inputs

bit 9-8 **Unimplemented:** Read as '0'

bit 7 **BUFS:** ADC Buffer Fill Status bit

Only valid when BUFM = 1 (ADC buffers split into 2 x N/2-word buffers).
1 = ADC is currently filling second buffer (from N/2 to (N-1)), user should access data in the first buffer (from 0 to (N/2-1))
0 = ADC is currently filling first buffer (from 0 to (N/2-1)), user should access data in the second buffer (from N/2 to (N-1))

Note 1: The number of conversions per interrupt is limited by the number of implemented ADC Result registers (ADCxBUFn). Refer to the specific device data sheet for the available options.

2: This bit only takes effect when the auto-scan feature is enabled (ASEN (ADxCON5<15>) = 1).

Register 25-2: ADxCON2: ADCx Control Register 2 (Continued)

bit 6-2 **SMPI<4:0>:** Sample/Convert Sequences Per Interrupt Selection bits(1)

11111 = Interrupts at the completion of conversion for each 32nd sample/convert sequence
11110 = Interrupts at the completion of conversion for each 31st sample/convert sequence
•
•
•
00001 = Interrupts at the completion of conversion for each 2nd sample/convert sequence
00000 = Interrupts at the completion of conversion for each sample/convert sequence

bit 1 **BUFM:** ADC Result Buffer Mode Select bit

1 = The ADC Result registers, ADCxBUFn, are divided into 2 equal buffers
0 = The ADC Result registers, ADCxBUFn, are configured as one buffer

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

Note 1: The number of conversions per interrupt is limited by the number of implemented ADC Result registers (ADCxBUFn). Refer to the specific device data sheet for the available options.

2: This bit only takes effect when the auto-scan feature is enabled (ASEN (ADxCON5<15>) = 1).

Register 25-3: ADxCON3: ADCx Control Register 3

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADRC	EXTSAM	—	SAMC<4:0>				
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADCS<7:0>							

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 31-16 **Unimplemented:** Read as '0'

bit 15 **ADRC:** ADC Conversion Clock Source (TSRC) bit
1 = Clock derived from Fast RC (FRC) oscillator
0 = Clock derived from Peripheral Bus Clock (TPB)

bit 14 **EXTSAM:** Extended Sampling Time bit
1 = ADC is still sampling after SAMP bit = 0
0 = ADC stops sampling when SAMP bit = 0

bit 13 **Unimplemented:** Read as '0'

bit 12-8 **SAMC<4:0>:** Auto-Sample Time bits
11111 = 31 TAD
•
•
•
00001 = 1 TAD
00000 = 0 TAD (not allowed)

bit 7-0 **ADCS<7:0>:** ADC Conversion Clock Select bits
11111111 = 2 • TSRC • ADCS<7:0> = 510 • TSRC = TAD
•
•
•
00000001 = 2 • TSRC • ADCS<7:0> = 2 • TSRC = TAD
00000000 = 1 • TSRC = TAD
Where TSRC is a period of clock selected by the ADRC bit (ADxCON3<15>).

Register 25-4: ADxCON5: ADCx Control Register 5

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	R/W-0
	ASEN(3)	LPEN	—	BGREQ(1)	—	—	ASINT<1:0>(2)	
7:0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	WM<1:0>		CM<1:0>	

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 31-16 **Unimplemented:** Read as '0'

bit 15 **ASEN:** Auto-Scan Enable bit(3)
- 1 = Auto-scan is enabled
- 0 = Auto-scan is disabled

bit 14 **LPEN:** Low-Power Enable bit
- 1 = Low power is enabled after scan
- 0 = Full power is enabled after scan

bit 13 **Unimplemented:** Read as '0'

bit 12 **BGREQ:** Band Gap Request bit(1)
- 1 = Band gap is enabled when the ADC is enabled
- 0 = Band gap is not enabled by the ADC

bit 11-10 **Unimplemented:** Read as '0'

bit 9-8 **ASINT<1:0>:** Auto-Scan (Threshold Detect) Interrupt Mode bits(2)
- 11 = Interrupt after Threshold Detect sequence has completed and a valid compare has occurred
- 10 = Interrupt after valid compare has occurred
- 01 = Interrupt after Threshold Detect sequence has completed
- 00 = No interrupt

bit 7-4 **Unimplemented:** Read as '0'

bit 3-2 **WM<1:0>:** Write Mode bits
- 11 = Reserved
- 10 = Auto-compare only (conversion results are not saved, but interrupts are generated when a valid match occurs, as defined by the CM<1:0> and ASINT<1:0> bits)
- 01 = Convert and save (conversion results saved to the ADCxBUFn registers when a match occurs, as defined by the CM<1:0> bits)
- 00 = Threshold (Comparison) mode is disabled, legacy operation (conversion data saved to the ADCxBUFn registers)

Note 1: This bit is not implemented in all devices.

2: The ASINT<1:0> bits setting only takes effect when ASEN = 1 (ADxCON5<15>) and WM<1:0> (ADxCON5<3:2>) = 10 or 01. Otherwise, the interrupt generation is governed by the SMPI<4:0> bits (ADxCON2<6:2>).

3: When auto-scan is enabled (ASEN (ADxCON5<15>) = 1), the CSCNA (ADxCON2<10>) and SMPI<4:0> (ADxCON2<6:2>) bits are ignored.

Register 25-4: ADxCON5: ADCx Control Register 5 (Continued)

bit 1-0 **CM<1:0>:** Compare Mode bits

11 = Outside Window mode (valid match occurs if the conversion result is outside of the window defined by the corresponding buffer pair)

10 = Inside Window mode (valid match occurs if the conversion result is inside the window defined by the corresponding buffer pair)

01 = Greater Than mode (valid match occurs if the result is greater than the value in the corresponding buffer register)

00 = Less Than mode (valid match occurs if the result is less than the value in the corresponding buffer register)

Note 1: This bit is not implemented in all devices.

2: The ASINT<1:0> bits setting only takes effect when ASEN = 1 (ADxCON5<15>) and WM<1:0> (ADxCON5<3:2>) = 10 or 01. Otherwise, the interrupt generation is governed by the SMPI<4:0> bits (ADxCON2<6:2>).

3: When auto-scan is enabled (ASEN (ADxCON5<15>) = 1), the CSCNA (ADxCON2<10>) and SMPI<4:0> (ADxCON2<6:2>) bits are ignored.

Register 25-5: ADxCHS: ADCx Input Select Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CH0NA<2:0>(1)			CH0SA<4:0>(1,2)				

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 31-8 **Unimplemented:** Read as '0'

bit 7-5 **CH0NA<2:0>:** Negative Input Select bits(1)
- 111 = Negative input is AN6
- 110 = Negative input is AN5
- 101 = Negative input is AN4
- 100 = Negative input is AN3
- 011 = Negative input is AN2
- 010 = Negative input is AN1
- 001 = Negative input is AN0
- 000 = Negative input is VREFL

bit 4-0 **CH0SA<4:0>:** Positive Input Select bits(1,2)
- 11111 = Positive input is AN31
- 11110 = Positive input is AN30
- •
- •
- •
- 00001 = Positive input is AN1
- 00000 = Positive input is AN0

Note 1: The input selection options are device-specific and can be different from device to device. Refer to the particular device data sheet for the available options.

2: The CH0SA<4:0> positive input selection is only used when CSCNA (ADxCON2<10>) = 0 and ASEN (ADxCON5<15>) = 0. The ADxCSS bits specify the positive inputs when CSCNA = 1 or ASEN = 1.

Register 25-6: ADxCSS: ADCx Input Scan Select Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSS<7:0>							

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 31-0 **CSS<31:0>:** ADC Input Pin Scan Selection bits
1 = Adds ANx to positive input scan list
0 = Skips ANx for input scan

Note 1: The actual number of bits available depends on which analog inputs are implemented on a specific device. Refer to the device data sheet for details. Unimplemented bits are read as '0'.
2: This register is only applicable when CSCNA (ADxCON2<10>) = 1 or ASEN (ADxCON5<15>) = 1.

Register 25-7: ADxCHIT: ADCx Compare Hit Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CHH<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CHH<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CHH<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CHH<7:0>							

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 31-0 **CHH<31:0>:** ADC Compare Hit bits
If CM<1:0> = 11:
1 = ADC Result Buffer n has been written with data or a match has occurred
0 = ADC Result Buffer n has not been written with data and no match has occurred.
For All Other Values of CM<1:0>:
1 = A match has occurred on ANx
0 = No match has occurred on ANx

Note 1: The actual number of bits available depends on which analog inputs are implemented on a specific device. Refer to the device data sheet for details. Unimplemented bits are read as '0'.

25.3 ADC TERMINOLOGY AND CONVERSION SEQUENCE

The ADC module operation consists of two steps: acquisition and conversion (see Figure 25-2). During acquisition, the selected analog input pin is connected to the Sample-and-Hold Amplifier (SHA). After the signal has been collected for a sufficient period, the sample voltage is equivalent to the input and the pin is disconnected from the SHA to provide a stable input voltage for the conversion process. Then, the analog sample voltage is converted to a binary representation.

An overview of the ADC module is presented in Figure 25-1. The 12-bit ADC has a single SHA. The SHA is connected to the analog input pins through the analog input multiplexer. The analog input multiplexer is controlled by the ADxCHS register. The ADC can also optionally scan through a series of analog inputs.

For single-ended measurements, the VINL input should be configured so that it connects to VREFL. The converted binary ADC output then directly specifies where, within the VREFH and VREFL voltage range, that the VINH signal is located.

For differential measurements, the converted binary value represents the VINH – VINL voltage difference. The ADC requires the differential voltage to be unipolar and the VINH – VINL voltage difference must always be positive for correct conversion codes at the output.

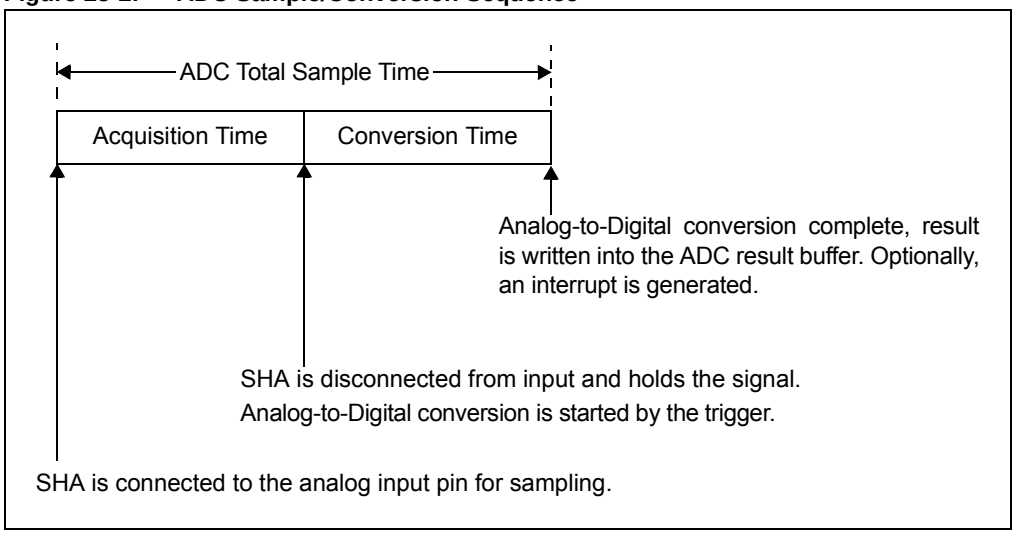
Acquisition time can be controlled manually or automatically. The acquisition may be started by setting the SAMP bit (ADxCON1<1>) and ended by clearing the SAMP bit in the software. The acquisition may be started automatically by the ADC hardware and ended automatically by a conversion trigger source. This auto-sample function is controlled by the ASAM bit (ADxCON1<2>). When the ASAM bit is set, the SHA is reconnected to the analog input pin at the end of the conversion in the sample/convert sequence. The ADC also has a mode when the conversions are started by hardware continuously. In this mode, the acquisition is set by the SAMC<4:0> bits (ADxCON3<12:8>).

Conversion time is the time required for the ADC to convert the voltage held by the SHA. The ADC requires one ADC clock cycle (TAD) to convert each bit of the result, plus two additional clock cycles. Therefore, a total of 14 TAD cycles are required to perform a complete 12-bit conversion. When the conversion time is complete, the result is written into one of the 32 ADC Result registers (ADCxBUF0 through ADCxBUF31). There are multiple input clock options for the ADC that are used to create the TAD clock.

The sum of the acquisition time and the Analog-to-Digital conversion time provides the total sample time (refer to Figure 25-2).

The sampling process can be performed once, periodically or based on a trigger, as defined by the module configuration.

Figure 25-2: ADC Sample/Conversion Sequence



The conversion trigger ends the sampling time and begins an Analog-to-Digital conversion. The conversion trigger source is selected by the SSRC<3:0> bits (ADxCON1<7:4>). The conversion trigger can be taken from a variety of hardware sources or can be controlled in software by clearing the SAMP bit. One of the conversion trigger sources is an auto-conversion timer. The time between auto-conversions is defined by a counter and the ADC clock period. The Auto-Sample mode and auto-conversion trigger can be used together to provide endless automatic conversions without software intervention.

An interrupt may be generated at the end of each sample sequence, or multiple sample sequences, as determined by the value of the SMPI<4:0> bits (ADxCON2<6:2>). The number of sample sequences between interrupts can vary and is limited by the number of implemented result buffers (ADCxBUFn). The user should note that the Analog-to-Digital conversion buffer holds the results of a single conversion sequence. The next sequence starts filling the buffer from the top, even if the number of samples in the previous sequence was less than the number of implemented ADCxBUFn registers. The total number of conversion results between interrupts is the SMPI<4:0> value. The total number of conversions between interrupts should not exceed the physical buffer depth.

25.4 ADC MODULE OPERATION

25.4.1 Analog Port Pins

The ANSELx and TRISx registers control the operation of the ADC port pins. ANSELx specifies the configuration of device pins to be used as analog inputs. A pin is configured as an analog input when the corresponding bit in the ANSELx register = 1. When the bit = 0, the pin is set to a digital input. When configured for analog input, the associated port I/O digital input buffer is disabled, so it does not consume current. The ANSELx registers are set at Reset, causing the ADC input pins to be configured for analog inputs by default.

The TRISx registers control the digital function of the port pins. The port pins that are desired as analog inputs must have their corresponding TRISx bit set, specifying the pin as an input. If the I/O pin associated with an ADC input is configured as an output (i.e, the TRISx bit is cleared), the port's digital output level (V_{OH} or V_{OL}) will be converted. After a device Reset, all of the TRISx bits are set.

Since the analog inputs employ Electrostatic Discharge (ESD) protection, they have diodes to V_{DD} and V_{SS}. This requires that the analog input must be between V_{DD} and V_{SS}. If the input voltage exceeds this range by greater than 0.3V (in either direction), one of the diodes becomes forward-biased and it may damage the device if the input current specification is exceeded.

An external RC filter is sometimes added for anti-aliasing of the input signal. The R component should be selected to ensure that the acquisition time requirements are satisfied. Any external components connected (through high-impedance) to an analog input pin (capacitor, Zener diode, etc.) should have very little leakage current at the pin.

Note:	When reading a PORTx register that shares pins with the ADC, any pin configured as an analog input reads as a '0' when the PORTx latch is read. Analog levels on any pin that is defined as a digital input, but while not configured as an analog input, may cause the input buffer to consume current that is out of the device's specification.

25.4.2 Analog Inputs

The ADxCHS register is used to select which analog input pin is connected for the conversion. The input analog multiplexer has two inputs, referred to as the positive and the negative input. The positive input is controlled by the CH0SA<4:0> bits (ADxCHS<4:0>) and the negative input is controlled by the CH0NA<2:0> bits (ADxCHS<7:5>).

The positive input can be selected from any one of the available analog input pins. The negative input can be selected as the ADC negative reference or an analog input from AN0 to AN6. The use of an analog input pin as the negative input allows the ADC to be used in Unipolar Differential mode.

Some devices include an option to measure a voltage of the device band gap circuit. To enable this feature, the BGREQ bit (ADxCON5<12>) must be set.

25.4.3 ADC Result Format

The data in the ADCxBUFn Result registers can be read as one of eight formats. The format is controlled by the FORM<2:0> bits (ADxCON1<10:8>). The user can select from integer, signed integer, fractional or signed fractional as a 16-bit or 32-bit result. Figure 25-3 and Figure 25-4 illustrate how a result is formatted.

Figure 25-3: ADC Output Data Formats, 10-Bit Mode (MODE12 Bit in ADxCON1 Register = 0)

Conversion Result:

d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Read to Bus:

Integer (FORM<2:0> = 000 or 100)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Signed Integer (FORM<2:0> = 001)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S	S	S	S	S	S	S	d08	d07	d06	d05	d04	d03	d02	d01	d00

Fractional (FORM<2:0> = 010)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0

Signed Fractional (FORM<2:0> = 011)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0

Figure 25-3: ADC Output Data Formats, 10-Bit Mode (MODE12 Bit in ADxCON1 Register = 0) (Continued)

Conversion Result:

d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Read to Bus:

Signed Integer (FORM<2:0> = 101)

S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	d08	d07	d06	d05	d04	d03	d02	d01	d00

Fractional (FORM<2:0> = 110)

d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signed Fractional (FORM<2:0> = 111)

S	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-4: ADC Output Data Formats, 12-Bit Mode (MODE12 Bit in ADxCON1 Register = 1)

Conversion Result:

d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Read to Bus:

Integer (FORM<2:0> = 000 or 100)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Signed Integer (FORM<2:0> = 001)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S	S	S	S	S	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Fractional (FORM<2:0> = 010)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0	0	0	0	0

Signed Fractional (FORM<2:0> = 011)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0	0	0	0	0

Figure 25-4: ADC Output Data Formats, 12-Bit Mode (MODE12 Bit in ADxCON1 Register = 1) (Continued)

Conversion Result:

d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Read to Bus:

Signed Integer (FORM<2:0> = 101)

S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Fractional (FORM<2:0> = 110)

d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signed Fractional (FORM<2:0> = 111)

S	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.4 Number of Conversions per Interrupt

The SMPI<4:0> bits (ADxCON2<6:2>) select how many Analog-to-Digital conversions will take place before a CPU interrupt is generated. The maximum number of the conversions per interrupt is limited by the number of implemented ADCxBUFn registers, N. The SMPI<4:0> bits also define the number of locations that will be written in the result buffer, starting with ADCxBUF0 (ADCxBUF0 or ADCxBUF(N/2) for Dual Buffer mode). This can vary from one sample to N samples (one to N/2 samples for Dual Buffer mode). After the interrupt is generated, the sampling sequence restarts, with the result of the first sample being written to the first buffer location.

For example, if the SMPI<4:0> bits = 00000, the conversion results will always be written to ADCxBUF0. In this example, no other buffer locations will be used.

For example, if the SMPI<4:0> bits = 01110, 15 samples would be converted and stored in buffer locations, ADCxBUF0 through ADCxBUF14. An interrupt will be generated after ADCxBUF14 is written. After this, the next sample will be written to ADCxBUF0 again. In this example, buffers from ADCxBUF15 and above will not be used.

The data in the result registers will be overwritten by the next sampling sequence. The data in the result buffer must be read before the completion of the next sample, which targets the same buffer. The Buffer Fill mode can be used to increase the effective time software has to read a series of result buffers before hardware overwrites the same buffers with subsequent samples. Refer to **Section 25.4.5 "Buffer Fill Mode"**.

For the Split Buffer mode (BUFM bit (ADxCON2<1>) = 1), the number of conversions per interrupt must not exceed N/2 (SPMI<4:0> bits (ADxCON2<6:2>) < N/2).

25.4.5 Buffer Fill Mode

The ADC Output Result registers can be used as a single buffer or can be divided into 2 buffers. This option is controlled by the BUFM bit (ADxCON2<1>). The sizes of the buffers depend on the number of implemented result registers, N. When the BUFM bit = 0, a single, up to N words, buffer is used. This single buffer begins from the ADC1BUF0 register and has a size defined by the SMPI<4:0> bits (ADxCON2<6:2>). If the BUFM bit = 1, the ADC has two buffers, up to N/2 words each. The first buffer begins from the ADCxBUF0 register and the second buffer begins from the ADCxBUF(N/2) register. Both buffers have the same size, specified by the SMPI<4:0> bits.

When the Two Buffers mode is used (BUFM bit = 1), then the BUFS bit (ADxCON2<7>) indicates which buffer the ADC is currently filling. If the BUFS bit = 0, the ADC is filling the first buffer, starting from ADCxBUF0, and the application software should read conversion values from the second buffer, starting from ADCxBUF(N/2). If the BUFS bit = 1, the situation is reversed and the application software should read conversion values from the buffer starting from ADCxBUF0.

When the ADC Result Buffer Mode Select bit, BUFM (ADxCON2<1>), is '0', the complete, up to N word buffer is used for all conversion sequences. Conversion results will be written sequentially in the buffer, starting at ADCxBUF0, until the number of samples as defined by the SMPI<4:0> bits (ADxCON2<6:2>) is reached. Then, the process is repeated and the next conversion result will be written to ADCxBUF0 again. If the ADC interrupt is enabled, an interrupt will be generated when the conversion result is written into the last buffer location.

When the BUFM bit is '1', the ADC Result registers will be split into two N/2 word groups. Conversion results will be written sequentially into the first buffer, starting at ADCxBUF0, until the number of samples as defined by the SMPI<4:0> bits is reached. During this time, the BUFS bit (ADxCON2<7>) will be cleared. The ADC interrupt will be generated when the buffer is filled completely.

After the ADC interrupt flag is set, the following result will be written sequentially to the second buffer, starting at ADCxBUF(N/2). The next conversion result will be written to the second buffer, starting at ADCxBUF(N/2), until the number of samples as defined by the SMPI<4:0> bits is reached. During this time, the BUFS bit will be set. The ADC interrupt will be generated when the second buffer is filled completely. The process then restarts with BUFS = 0 and the results being written to the first buffer.

The decision of which Buffer Fill mode to use will depend upon how much time is available to move the buffer contents, after the Analog-to-Digital interrupt and the interrupt latency, as determined by the application. If the processor can unload a full buffer within the time it takes to sample and convert one analog input, the BUFM bit can be '0' and up to N conversions may be done per interrupt. In case there are back-to-back continuous conversions, the processor will have one acquisition and conversion period before the first buffer location is overwritten.

For back-to-back conversions, if the processor cannot unload the buffer within the sample and conversion time, set the BUFM bit = 1 to prevent overwriting result data. For example, if the SMPI<4:0> bits = 00111, eight conversions will be written, loaded into the first buffer, following which, an interrupt will occur. The next eight conversions will be written to the second buffer. Therefore, the processor will have the entire time between interrupts to read the eight conversions out of the buffer.

25.4.6 Trigger Source

The ADC module may use different sources as a conversion trigger. The selection of the conversion trigger source is controlled by the SSRC<3:0> bits (ADxCON1<7:4>).

25.4.6.1 SOFTWARE TRIGGER

The ADC can be configured to end sampling and start a conversion when the SAMP bit (ADxCON1<1>) is cleared (= 0). This option is selected when the SSRC<3:0> bits are set to '0000'. To generate the software trigger, the time between setting and clearing the SAMP bit must be more than one ADC clock cycle (TAD). If this requirement is not met, the trigger will not be generated.

25.4.6.2 EXTERNAL TRIGGER

The ADC trigger can be generated using an event from another module. For example, the Timer1 period match event can trigger the ADC if the SSRC<3:0> bits are set to '0101'.

There is a special option when the SSRC<3:0> bits equal '0110'. For this selection, the trigger can be generated by Timer1 when the device is in Sleep mode.

25.4.6.3 EXTERNAL INT0 PIN TRIGGER

To configure the ADC to begin a conversion on an active transition on the INT0 pin, the SSRC<3:0> bits should be set to '0001'. The INT0 pin may be programmed for either a rising edge input or a falling edge input to trigger the conversion process.

25.4.6.4 AUTO-CONVERSION TRIGGER

The ADC is configured to automatically perform sampling and conversion when the Conversion Trigger Source Select bits, SSRC<3:0>, are set to '0111'. In this mode, setting of the SAMP bit starts the sampling. The input sampling time is defined by the SAMC<4:0> bits (ADxCON3<12:8>). When sampling is finished, the SAMP bit is cleared automatically and the conversion begins.

25.4.7 Sampling Mode

Sampling can be started by software or automatically by hardware when the previous conversion is complete. This sampling option is controlled by the ASAM bit (ADxCON1<2>).

Clearing the ASAM bit (ADxCON1<2>) disables the Auto-Sample mode. Acquisition will begin when the SAMP bit (ADxCON1<1>) is set by software. Acquisition will not resume until the SAMP bit is once again set.

Setting the ASAM bit (ADxCON1<2>) enables Auto-Sample mode. In this mode, the sampling will start automatically after the previous conversion has been completed. Clearing the ASAM bit (ADxCON1<2>) while in Auto-Sample mode will not terminate an ongoing acquire/convert sequence. However, sampling will not automatically resume after the current sample is converted.

When the SAMP bit becomes zero, the analog input is disconnected from the Sample-and-Hold circuit. This behavior can be changed using the EXTSAM bit (ADxCON3<14>). When this bit is set, the ADC still continues to sample the input signal when the SAMP bit is zero.

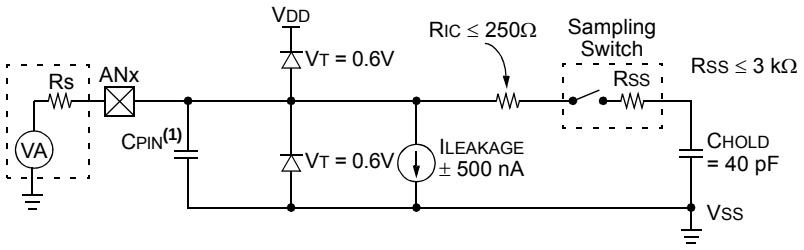
When the auto conversion trigger is selected (SSRC<3:0> bits are set to '0111') or the auto scan feature is enabled (ASEN bit in ADxCON5 register is set), the analog input sampling time is defined by SAMC<4:0> bits (ADxCON3<12:8>).

25.4.8 Sampling Time Requirements

The analog input model of the 12-bit ADC module is shown in Figure 25-5. The total acquisition time for the Analog-to-Digital conversion is a function of the internal amplifier settling time and the holding capacitor charge time.

For the ADC module to meet its specified accuracy, the Charge Sample-and-Hold Capacitor (CHOLD) must be allowed to fully charge to the voltage level on the analog input pin. The Analog Output Source Resistance (RS), the Interconnect Resistance (RIC) and the internal Sampling Switch Resistance (RSS) impedance combine to directly affect the time required to charge the CHOLD. The combined impedance of the analog sources must therefore be small enough to fully charge the holding capacitor within the chosen sample time. After the analog input is selected (changed), this acquisition function must be completed prior to starting the conversion.

Figure 25-5: 10-Bit ADC Analog Input Model



Note 1: The CPIN value depends on the device package and is not tested. The effect of the CPIN is negligible if Rs ≤ 5 kΩ.

Legend:

CPIN = Input Capacitance

RSS = Sampling Switch Resistance

RS = Source Resistance

ILEAKAGE = Leakage Current at the Pin due to Various Junctions

VT = Threshold Voltage

RIC = Interconnect Resistance

CHOLD = Sample-and-Hold Capacitor

25.4.9 Voltage Reference Source

The VCFG<2:0> bits (ADxCON2<15:13>) select the voltage reference for Analog-to-Digital conversions. The higher voltage reference (VREFH) and the lower voltage reference (VREFL) may be the internal AVDD and AVSS voltage rails, or the VREF+ and VREF- input pins.

The external voltage reference pins may be shared with some ADC inputs on low pin count devices. The voltages applied to the external reference pins must meet certain specifications. Refer to the **"Electrical Characteristics"** section in the specific device data sheet for more information.

25.4.10 ADC Clock Source and Prescaler

The ADC module can use the Fast RC (FRC) oscillator or the Peripheral Bus Clock (TPB) as a conversion clock source (with period, TSRC). The clock source is controlled by the ADRC bit (ADxCON3<15>).

When the ADRC bit = 1, the Fast 8 MHz RC oscillator is used (TSRC = 1/8 MHz).

Note:	The Fast RC oscillator provides the ADC operation in Sleep mode.

When the ADRC bit = 0, the Peripheral Bus Clock is used for the ADC (TSRC = TPB). For the correct operation, the ADC conversion clock (TAD) must meet the minimum TAD time requirement (see the specific device data sheet for the minimum TAD time). The ADC conversion clock (TAD) can be set using the prescaler specified by the ADCS<7:0> bits (ADxCON3<7:0>).

Equation 25-1 gives the TAD value as a function of the ADCS<7:0> bits and the source clock period, TSRC.

Equation 25-1: ADC Conversion Clock Period

$$T_{AD} = 2 \bullet T_{SRC} \bullet (ADCS<7:0> + 1)$$

$$ADCS<7:0> = \left(\frac{T_{AD}}{2 \bullet T_{SRC}}\right) - 1$$

25.4.11 Scan Mode

The ADC module has a feature to scan several inputs. The CSCNA bit (ADxCON2<10>) enables the multiplexer input to be scanned across a selected number of analog inputs.

When Scan mode is enabled, the positive input is controlled by the contents of the ADxCSS register. Each bit in the ADxCSS register corresponds to an analog input. Bit 0 corresponds to AN0, bit 1 corresponds to AN1, and so on. If a particular bit in the ADxCSS register is '1', the corresponding input is part of the scan sequence. The input is always scanned from the lower numbered input to the higher numbered input, starting at the lowest order bit that is enabled in ADxCSS. For scanning, one trigger must be generated for each channel included in the scan list (ADxCSS register). When Scan mode is enabled, the CH0SA<4:0> bits (ADxCHS<4:0>) are ignored. The number of samples per interrupt (defined by the SMPI<4:0> bits) must be set equal to the number of scanned inputs.

Note:	If the number of scanned inputs selected is greater than the number of samples taken per interrupt (defined by the SMPI<4:0> bits), the higher numbered inputs will not be sampled. The ADxCSS register specifies only the input of the positive analog input. The CH0NA bits (ADxCHS<7:5>) select the input of the negative analog input during scanning.

When the CSCNA bit = 0, Scan mode is disabled and the positive input to the multiplexer is controlled by the CH0SA<4:0> bits.

25.4.12 Auto-Scan Mode

The ADC module Auto-Scan mode feature allows the scanning of a few inputs. Unlike the regular Scan mode controlled by the CSNA bit (ADxCON2<10>), the auto-scan requires only one initial trigger to convert all channels. The scan start is triggered by the trigger source programmed by the SSRC<4:0> bits (ADxCON1<7:4>). The ASEN bit (ADxCON5<15>) controls the overall operation of auto-scan; setting this bit enables the functionality.

For the Auto-Scan mode, an interrupt is generated after the scan sequence completion or the interrupt generation is controlled by the ASINT<1:0> bits (ADxCON5<9:8>) if the Threshold Detect feature is enabled (refer to **Section 25.4.13 "Threshold Detect Operation"** for details).

The ASEN bit enables the multiplexer input to be scanned across a selected number of analog inputs. When Auto-Scan mode is enabled, the positive input is controlled by the contents of the ADxCSS register. Each bit in the ADxCSS register corresponds to an analog input. Bit 0 corresponds to AN0, bit 1 corresponds to AN1, and so on. If a particular bit in the ADxCSS register is '1', the corresponding input is part of the scan sequence. The input is always scanned from the lower numbered input to the higher numbered input, starting at the lowest order bit that is enabled in ADxCSS. When Auto-Scan mode is enabled, the CH0SA<4:0> (ADxCHS<4:0>), CSCNA (ADxCON2<10>) and SMPI<4:0>(ADxCON2<6:2>) bits are ignored. When the ASEN bit = 0, Auto-Scan mode is disabled and the positive input to the multiplexer is controlled by the CH0SA<4:0> bits, or by scan logic, if it is enabled by the CSCNA bit.

For the Auto-Scan mode, the result of the conversion can be stored in the ADCxBUFn registers by the channel number (Channel Indexed Buffer mode). This mode is controlled by the BUFREGEN bit (ADxCON2<11>). If this bit is set, the FIFO mode is disabled and each channel has its own corresponding ADC Result Buffer register. The result for the AN0 input will be stored in the ADCxBUF0 register, the result for the AN5 input will be stored in the ADCxBUF5 register, and so on.

The LPEN bit (ADxCON5<14>) allows the auto-scan feature to function in Sleep or Idle mode. Setting LPEN configures the ADC to enter into Low-Power mode between triggers after the scan is finished. If the Threshold Detect feature is enabled and the LPEN bit = 1, the ADC can operate in Sleep or Idle mode and wake the CPU (generate an interrupt) only when the specified threshold conditions are met (refer to **Section 25.4.13 "Threshold Detect Operation"** for details).

25.4.13 Threshold Detect Operation

Threshold Detect is an extension of the auto-scan feature described in the previous section. To use the Threshold Detect, the Auto-Scan mode must be enabled by setting the ASEN bit (ADxCON5<15>). In addition to being able to repeatedly sample a predefined sequence of analog channels, Threshold Detect allows the user to define match conditions based on the conversion results and generates an interrupt based on these conditions. During normal operation, this can potentially reduce the amount of CPU time spent on processing ADC interrupts. For low-power applications, this can allow the CPU to remain inactive for longer periods, waking only when specific analog conditions are met.

When selected by the user, Threshold Detect changes the operation of the ADC results buffer by making it a read/write array for both conversion results and comparison (threshold) values. It also brings into play the ADxCHIT register, which is used to indicate match conditions.

25.4.13.1 OPERATING MODES

The operation of Threshold Detect is controlled by the ADxCON5 register (Register 25-4). The Threshold Detect is enabled if the WM<1:0> bits (ADxCON5<3:2>) are set to a non-zero value ('01' or '10').

The channels for comparison are selected using the ADxCSS register. Setting a particular bit in either register includes the corresponding channel in an automatic sequential scan. One or more channels may be selected. After the channels have been selected, set the ASEN bit to enable a scan of the designated channels. The scan itself is triggered by the trigger source programmed by the SSRC<3:0> bits.

The Compare Mode bits, CM<1:0> (ADxCON5<1:0>), select the type of comparison to be performed. Four types are available:

- The result of the current conversion is greater than a reference threshold
- The result of the current conversion is less than a reference threshold
- The result of the current conversion is between two predefined thresholds ("Inside Window")
- The result of the current conversion is outside of the predefined thresholds ("Outside Window")

The Write Mode bits, WM<1:0> (ADxCON5<3:2>), enable the Threshold Detect feature and determine the disposition of the conversion. Three options are available:

- Discard the conversion after the comparison has been performed
- Store the conversion after the comparison has been performed
- Store the conversion without comparison

25.4.13.1.1 Buffer Operation and Comparisons

For Buffer Write modes that involve storing conversions (WM<1:0> = 0x), the BUFM and BUFREGEN bits control how the buffer functions (as a channel indexed, single FIFO or split FIFO buffer). However, when the compare and store option is selected (WM<1:0> = 01), using a FIFO mode may overwrite the buffers of other channels and cause unpredictable comparison results. For that reason, always use Channel Indexed Buffer mode (BUFREGEN = 1) when using the compare and store option.

25.4.13.1.2 Buffer Operation in Windowed Comparisons (Channel Mirroring)

The use of windowed comparisons changes the available options for the results buffer. To accommodate the storage of two threshold values, the buffer of N registers is automatically split into two N/2 buffers, similar to Split FIFO mode. Buffer addresses in each half are paired with the lowest address in one buffer, matched to the buffer address in the upper half. (For example, in a 16-word buffer, ADCxBUF0 is paired with ADCxBUF8, ADCxBUF1 is paired with ADCxBUF9, and so on.) This pairing is referred to as "channel mirroring".

Mirroring can obviously be applied only to the lower ADC channels; for most devices, this corresponds to the lower half of the external analog channels. This does not mean that those buffer locations cannot be used for other purposes. However, storing any other data in a particular buffer location, where channel mirroring is being used, may result in misleading comparison evaluations.

25.4.13.2 SETTING COMPARISON THRESHOLDS

The comparison thresholds for Threshold Detect are set by writing the desired values to an appropriate location in the ADC results buffer. This can only be done when the module is deactivated (ADxCON1<15> = 0).

The location of the threshold is determined by the comparison type. For simple greater than, and less than, comparisons, the value is written to the buffer location corresponding to the input channel to be monitored. For example, if AN0 is to be monitored for a voltage over a certain level, the ceiling threshold is stored in ADCxBUF0.

The location of the thresholds for windowed comparisons are written to two addresses. The lower value is written to the address corresponding to the monitored channel. The upper value is stored in the corresponding mirrored address in the upper half of the buffer. To expand on the previous example with a 16-word buffer, if the conversion on AN0 is to be a windowed comparison, the floor threshold is stored in ADCxBUF0, while the ceiling threshold is stored in ADCxBUF8.

25.4.13.3 COMPARE HIT REGISTERS

To determine if a particular event has occurred, the ADC module uses the ADxCHIT register to record match events. This register maps its individual bits sequentially to each of the (up to) 32 analog channels. If a particular channel in a device is not implemented, the corresponding Compare Hit bit (CHHn) is not implemented.

Each bit serves as an event semaphore for its corresponding channel. When the programmed event occurs on that channel, the bit becomes set and stays set until it is cleared by the application. It is the user's responsibility to clear the bits after the application has evaluated them.

Depending on the event, more than one compare hit bit may be set. The significance of a set bit must be interpreted by the application in the context of the Compare mode selected. Particular examples are covered in **Section 25.4.13.5 "Comparison Mode Examples"**.

25.4.13.4 THRESHOLD DETECT INTERRUPTS

The ADC module can generate an interrupt and set the ADxIF flag based on Threshold Detect operation. This is based on completion of a Threshold Detect sequence and/or the occurrence of a valid comparison. When Threshold Detect is enabled (ASEN = `1` and WM<1:0> ≠ `00`), ADC module interrupt generation is governed by the ASINT<1:0> bits (ADxCON5<9:8>) and the SMPI<4:0> bits (ADxCON2<6:2>) are ignored.

The Threshold Detect interrupt is configured by the ASINT<1:0> bits. Options include interrupt after a scan sequence, interrupt after a scan sequence with a valid match, interrupt after a valid match (without waiting for the sequence to end) or no interrupt.

25.4.13.5 COMPARISON MODE EXAMPLES

The following examples show the effect of valid comparisons on the results buffer and the Compare Hit registers. In each figure, changes within the registers are indicated in bold.

For the sake of simplicity, the examples assume a device with only 16 ADC result buffers (from ADC1BUF0 to ADC1BUF15). Devices with a larger number of results buffers will function in a similar fashion.

Note:	When using any comparison mode, always use channel indexed buffer storage (BUFREGEN = `1`). Otherwise, the threshold values for other channels may be overwritten, resulting in unpredictable comparisons.

25.4.13.5.1 Simple Comparisons (Greater and Less Than Results)

When the Compare Mode bits, CM<1:0> (ADxCON5<1:0>), are programmed as '0x', the converter compares the sampled value to see if it is greater than (CM<1:0> = 01), or less than (CM<1:0> = 00), the threshold value in the buffer location. If the condition is met, both of the following occur:

- The Compare Hit bit (CHHn) for the corresponding channel is set.
- If the Write Mode bits, WM<1:0> (ADxCON5<3:2>), are programmed to '01', the converted value is written to the buffer, replacing the threshold value. If WM<1:0> = 10, the converted value is discarded.

The changes to the 16-word result buffer and the Compare Hit register are shown in Figure 25-6. Note that they are the same for both types of simple comparison.

Figure 25-6: Simple Comparison Operations (Greater Than and Less Than)

Before Conversion and Comparison

ADCxBUF15	—
ADCxBUF14	—
ADCxBUF13	—
ADCxBUF12	—
ADCxBUF11	—
ADCxBUF10	—
ADCxBUF9	—
ADCxBUF8	—
ADCxBUF7	—
ADCxBUF6	—
ADCxBUF5	—
ADCxBUF4	—
ADCxBUF3	—
ADCxBUF2	Threshold Value
ADCxBUF1	—
ADCxBUF0	—

ADxCHIT = 0x00000000

After Conversion and Comparison

	Compare Only (WM<1:0> = 10)	**Compare and Store (WM<1:0> = 01)**
ADCxBUF15	—	—
ADCxBUF14	—	—
ADCxBUF13	—	—
ADCxBUF12	—	—
ADCxBUF11	—	—
ADCxBUF10	—	—
ADCxBUF9	—	—
ADCxBUF8	—	—
ADCxBUF7	—	—
ADCxBUF6	—	—
ADCxBUF5	—	—
ADCxBUF4	—	—
ADCxBUF3	—	—
ADCxBUF2	Threshold Value	**Conversion Value**
ADCxBUF1	—	—
ADCxBUF0	—	—

ADxCHIT = 0x00000004

25.4.13.5.2 Inside Window Comparison

When the Compare Mode bits, CM<1:0>, are programmed as '10', the converter compares the sampled value to see if it falls between the threshold values in the buffer and mirrored channel location. Since the value in the mirrored channel location is always the greater value of the two thresholds, the condition is met when:

Threshold 2 > Converted Value > Threshold 1

In this case, both of the following occur:

- The Compare Hit bit (CHHn) for the corresponding channel is set; the Compare Hit bit for the mirrored channel remains cleared.
- If the Write Mode bits, WM<1:0>, are programmed to '01', the converted value is written to the buffer, replacing the lower threshold value. If WM<1:0> = 10, the converted value is discarded.

The changes to the 16-word result buffer and the Compare Hit register are shown in Figure 25-7.

Figure 25-7: Inside Window Comparison Operation

Before Conversion and Comparison

ADCxBUF15	—
ADCxBUF14	—
ADCxBUF13	—
ADCxBUF12	—
ADCxBUF11	—
ADCxBUF10	Threshold 2
ADCxBUF9	—
ADCxBUF8	—
ADCxBUF7	—
ADCxBUF6	—
ADCxBUF5	—
ADCxBUF4	—
ADCxBUF3	—
ADCxBUF2	Threshold 1
ADCxBUF1	—
ADCxBUF0	—

ADxCHIT = 0x00000000

After Conversion and Comparison

	Compare Only (WM<1:0> = 10)	**Compare and Store (WM<1:0> = 01)**
ADCxBUF15	—	—
ADCxBUF14	—	—
ADCxBUF13	—	—
ADCxBUF12	—	—
ADCxBUF11	—	—
ADCxBUF10	Threshold 2	Threshold 2
ADCxBUF9	—	—
ADCxBUF8	—	—
ADCxBUF7	—	—
ADCxBUF6	—	—
ADCxBUF5	—	—
ADCxBUF4	—	—
ADCxBUF3	—	—
ADCxBUF2	Threshold 1	**Conversion Value**
ADCxBUF1	—	—
ADCxBUF0	—	—

ADxCHIT = 0x00000004

25.4.13.5.3 Outside Window Comparison

When the Compare Mode bits, CM<1:0>, are programmed as '11', the converter compares the sampled value to see if it falls outside of the threshold values in the buffer and mirrored channel location. Again, since the value in the mirrored channel location is always the greater value of the two thresholds, the condition is met when either:

$$\textit{Converted Value} > \textit{Threshold 2}$$

or

$$\textit{Threshold 1} > \textit{Converted Value}$$

In these cases, the following occurs:

- The Compare Hit bit (CHHn) for the corresponding channel is set.
- If the converted value is greater than Threshold 2, the CHHn bit for the mirrored channel is also set. If it is less than Threshold 1, the mirrored channel bit remains '0'.
- If the Write Mode bits, WM<1:0>, are programmed to '01':
  - If the converted value is above Threshold 2, the converted value is written to the mirrored channel address, replacing the upper threshold value.
  - If the converted value is below Threshold 1, the converted value is written to the channel address, replacing the lower threshold value.
- If WM<1:0> = 10, the converted value is discarded.

The changes to the 16-word result buffer and the Compare Hit register are shown in Figure 25-8 (over the upper threshold) and Figure 25-9 (under the lower threshold).

Note that when a Windowed Comparison mode is selected and channel mirroring is enabled, nothing prevents a conversion from another operation from being stored in the mirrored channel location. In the previous examples of windowed operation, if AN10 is included in a Threshold Detect operation, a conversion on AN10 might be tested against the upper threshold for AN2, stored in that location. This could result in the threshold value being overwritten and/or the CHH10 bit being set.

For this reason, users must always carefully consider the allocation and use of the upper analog channels (both external and internal) when using Windowed Compare modes. Wherever possible, exclude the upper analog channels for Threshold Detect operations, and convert and test those channels in a separate routine.

Figure 25-8: Outside Window Comparison Operation (over Threshold 2)

Before Conversion and Comparison

ADCxBUF15	—
ADCxBUF14	—
ADCxBUF13	—
ADCxBUF12	—
ADCxBUF11	—
ADCxBUF10	Threshold 2
ADCxBUF9	—
ADCxBUF8	—
ADCxBUF7	—
ADCxBUF6	—
ADCxBUF5	—
ADCxBUF4	—
ADCxBUF3	—
ADCxBUF2	Threshold 1
ADCxBUF1	—
ADCxBUF0	—

ADxCHIT = 0x00000000

After Conversion and Comparison

	Compare Only (WM<1:0> = 10)	**Compare and Store (WM<1:0> = 01)**
ADCxBUF15	—	—
ADCxBUF14	—	—
ADCxBUF13	—	—
ADCxBUF12	—	—
ADCxBUF11	—	—
ADCxBUF10	Threshold 2	**Conversion Value**
ADCxBUF9	—	—
ADCxBUF8	—	—
ADCxBUF7	—	—
ADCxBUF6	—	—
ADCxBUF5	—	—
ADCxBUF4	—	—
ADCxBUF3	—	—
ADCxBUF2	Threshold 1	Threshold 1
ADCxBUF1	—	—
ADCxBUF0	—	—

ADxCHIT = 0x00000044

Figure 25-9: Outside Window Comparison Operation (under Threshold 1)

Before Conversion and Comparison

ADCxBUF15	—
ADCxBUF14	—
ADCxBUF13	—
ADCxBUF12	—
ADCxBUF11	—
ADCxBUF10	Threshold 2
ADCxBUF9	—
ADCxBUF8	—
ADCxBUF7	—
ADCxBUF6	—
ADCxBUF5	—
ADCxBUF4	—
ADCxBUF3	—
ADCxBUF2	Threshold 1
ADCxBUF1	—
ADCxBUF0	—

ADxCHIT = 0x00000000

After Conversion and Comparison

	Compare Only (WM<1:0> = 10)	**Compare and Store (WM<1:0> = 01)**
ADCxBUF15	—	—
ADCxBUF14	—	—
ADCxBUF13	—	—
ADCxBUF12	—	—
ADCxBUF11	—	—
ADCxBUF10	Threshold 2	Threshold 2
ADCxBUF9	—	—
ADCxBUF8	—	—
ADCxBUF7	—	—
ADCxBUF6	—	—
ADCxBUF5	—	—
ADCxBUF4	—	—
ADCxBUF3	—	—
ADCxBUF2	Threshold 1	**Conversion Value**
ADCxBUF1	—	—
ADCxBUF0	—	—

ADxCHIT = 0x00000004

25.4.14 Turning On the ADC

When the ON bit (ADxCON1<15>) is '1', the ADC module is in Active mode and is fully powered and functional.

When ON is '0', the ADC module is disabled. The digital and analog portions of the circuit are turned off for maximum current savings. Clearing the ON bit during a conversion will abort the current conversion. The ADC Result registers will not be updated. That is, the corresponding result buffer location will continue to contain the value of the last completed conversion (or the last value written to the buffer).

In order to return to the Active mode from the Off mode, the user software must wait for the analog stages to stabilize. Refer to the **"Electrical Characteristics"** section in the specific device data sheet for the stabilization time.

Note:	Writing to any ADC control bits other than the ON (ADxCON1<15>), SAMP (ADxCON1<1>) and DONE (ADxCON1<0>) bits is not recommended while the ADC module is running.

25.4.15 Offset Calibration

The ADC module provides a method of measuring the internal offset error. After this offset error is measured, it can be subtracted in software from the result of an Analog-to-Digital conversion. Use the following steps to perform an offset measurement:

1. Configure the ADC in the same manner as it will be used in the application.
2. Set the OFFCAL bit (ADxCON2<12>). This overrides the input selections and connects the SHA inputs to AVSS.
3. Enable the ADC and perform a single conversion. The value that is written to the ADC result buffer is the internal offset error.
4. Clear the OFFCAL bit to return the ADC to normal operation.

Note:	Only positive ADC offsets can be measured with this method.

25.4.16 DONE Bit Operation

The DONE bit (ADxCON1<0>) is set when a conversion sequence is complete. In Manual mode, the DONE bit is persistent; it remains set until it is cleared by software. The DONE bit can be polled to determine when the conversion has completed.

In all Automatic Sample modes (ASAM bit = 1), the DONE bit is not persistent. It is set at the end of a conversion sequence and cleared by hardware when the next acquisition is started. Polling the DONE bit is not recommended when operating the ADC in Automatic Sample modes. The ADC interrupt flag bit is latched after a conversion sequence is completed and can therefore be polled.

25.5 APPLICATION EXAMPLES

The following configuration examples show the ADC operation in different sampling and buffering configurations. In each example, it is assumed that the device has one ADC with a 16-word buffer.

25.5.1 Software Conversion Control

When the SSRC<3:0> bits (ADxCON1<7:4>) = 0000, the conversion trigger is under software control. Clearing the SAMP bit (ADxCON1<1>) starts the conversion sequence.

Example 25-1 is an example where setting the SAMP bit initiates sampling, and clearing the SAMP bit terminates sampling and starts conversion. The user software must time the setting and clearing of the SAMP bit for at least one TAD cycle to ensure that the ADC will detect the SAMP bit state transition.

Example 25-1: Conversion Code Using Software Controlled Sampling and Trigger

```
#include   <xc.h>

unsigned   long   ADCResult;

void       ADCTest()
{
// Set RB2/AN4 as an analog input
ANSELBbits.ANSB2 = 1;
TRISBbits.TRISB2 = 1;
// Select RB2/AN4 input
AD1CHSbits.CH0SA = 4;
// Clock from peripheral clock TPB
AD1CON3bits.ADRC = 0;
// ADC clock TAD = 8 peripheral clock TPB
AD1CON3bits.ADCS = 8;
// Set software trigger
AD1CON1bits.SSRC = 0;
// Turn on the ADC
AD1CON1bits.ON = 1;

// Repeat continuously
while (1)
{
    // Start sampling...
    AD1CON1bits.SAMP = 1;
    // Delay at least 1 TAD
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    // Reset ADC interrupt flag
    IFS0CLR = _IFS0_AD1IF_MASK;
    // Start converting ...
    AD1CON1bits.SAMP = 0;
    // Wait for the result
    while (IFS0bits.AD1IF == 0);
    // Get ADC value
    ADCResult = ADC1BUF0;
}
}
```

25.5.2 Automatic Acquisition

Example 25-2 is an example in which setting the ASAM bit (ADxCON1<2>) initiates automatic acquisition, and clearing the SAMP bit (ADxCON1<1>) terminates sampling and starts conversion. After the conversion completes, the module will automatically return to an acquisition state. The SAMP bit is automatically set at the start of the acquisition interval. The user software must time the clearing of the SAMP bit for at least one TAD cycle to ensure that the ADC will detect the SAMP bit state transition.

Example 25-2: Conversion Code Using Automatic Sampling and Software Trigger

```
#include   <xc.h>

unsigned   long   ADCResult;

void       ADCTest()
{
// Set RB15/AN10 as an analog input
ANSELBbits.ANSB15 = 1;
TRISBbits.TRISB15 = 1;
// Select RB15/AN10 input
AD1CHSbits.CH0SA = 10;
// Clock from peripheral clock TPB
AD1CON3bits.ADRC = 0;
// ADC clock TAD = 8 peripheral clock TPB
AD1CON3bits.ADCS = 8;
// Enable sampling automatically after each conversion
AD1CON1bits.ASAM = 1;
// Set software trigger
AD1CON1bits.SSRC = 0;
// Turn on the ADC
AD1CON1bits.ON = 1;
// Start the first sampling cycle
AD1CON1bits.SAMP = 1;

// Repeat continuously
while (1)
{
    // Delay at least 1 TAD
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    Nop(); Nop(); Nop(); Nop();
    // Reset ADC interrupt flag
    IFS0CLR = _IFS0_AD1IF_MASK;
    // Start converting ...
    AD1CON1bits.SAMP = 0;
    // Wait for the result
    while (IFS0bits.AD1IF == 0);
    // Get ADC value
    ADCResult = ADC1BUF0;
}
}
```

25.5.3 Auto-Conversion Trigger

When the SSRC<3:0> bits (ADxCON1<7:4>) = 0111, the conversion trigger is under ADC clock control. The SAMC<4:0> bits (ADxCON3<12:8>) select the number of TAD clock cycles between the start of acquisition and the start of conversion. This trigger option provides the fastest conversion rates on multiple channels. After the start of acquisition, the module will count a number of TAD clocks specified by the SAMC<4:0> bits.

Equation 25-2: Clocked Conversion Trigger Time

$$TSMP = SAMC<4:0> * TAD$$

The SAMCx bits must always be programmed for at least one clock cycle. See Example 25-3 for a code example.

Example 25-3: Conversion Code Using Software Sampling Start and Auto-Conversion Trigger

```
#include   <xc.h>

unsigned   long   ADCResult;

void       ADCTest()
{
// Set RB12/AN7 as an analog input
ANSELBbits.ANSB12 = 1;
TRISBbits.TRISB12 = 1;
// Select RB12/AN7 input
AD1CHSbits.CH0SA = 7;
// Clock from peripheral clock TPB
AD1CON3bits.ADRC = 0;
// ADC clock TAD = 8 peripheral clock TPB
AD1CON3bits.ADCS = 8;
// Set sample time to 5 TAD
AD1CON3bits.SAMC = 5;
// Set auto conversion trigger
AD1CON1bits.SSRC = 7;
// Turn on the ADC
AD1CON1bits.ON = 1;

// Repeat continuously
while (1)
{
    // Reset ADC interrupt flag
    IFS0CLR = _IFS0_AD1IF_MASK;
    // Start sampling and after 5 TAD go to conversion
    AD1CON1bits.SAMP = 1;
    // Wait for the result
    while (IFS0bits.AD1IF == 0);
    // Get ADC value
    ADCResult = ADC1BUF0;
}
}
```

25.5.4 Free-Running Sample/Conversion Sequence

As shown in Example 25-4, using the Auto-Conversion Trigger mode (SSRC<3:0> = 0111) in combination with the Automatic Sampling Start mode (ASAM = 1), allows the ADC module to schedule acquisition/conversion sequences with no intervention by the user or other device resources. This Automatic mode allows continuous data collection after module initialization.

Example 25-4: Conversion Code Using Auto-Sampling and Auto-Conversion Start

```
#include   <xc.h>

unsigned   long   ADCResult;
void       ADCTest()
{
// Set RB12/AN7 as an analog input
ANSELBbits.ANSB12 = 1;
TRISBbits.TRISB12 = 1;
// Select RB12/AN7 input
AD1CHSbits.CH0SA = 7;
// Clock from peripheral clock TPB
AD1CON3bits.ADRC = 0;
// ADC clock TAD = 8 peripheral clock TPB
AD1CON3bits.ADCS = 8;
// Set sample time to 5 TAD
AD1CON3bits.SAMC = 5;
// Enable sampling automatically after each conversion
AD1CON1bits.ASAM = 1;
// Set auto conversion trigger
AD1CON1bits.SSRC = 7;
// Turn on the ADC
AD1CON1bits.ON = 1;
// Start the first sampling cycle
AD1CON1bits.SAMP = 1;

// Repeat continuously
while (1)
{
    // Reset ADC interrupt flag
    IFS0CLR = _IFS0_AD1IF_MASK;
    // Wait for the result
    while (IFS0bits.AD1IF == 0);
    // Get ADC value
    ADCResult = ADC1BUF0;
}
}
```

25.5.5 Analog-to-Digital Conversions While Scanning Through Analog Inputs

Example 25-5 illustrates a typical setup where all available analog inputs are sampled and converted. Setting the CSCNA bit (ADxCON2<10>) specifies scanning of the ADC inputs.

Initially, the AN4 input is acquired and converted. The result is stored in the ADC buffer. Then the AN7 and AN10 inputs are acquired and converted.

Example 25-5: Conversion Code Using Inputs Scanning

```
#include   <xc.h>

unsigned   long   ADCResultAN4;
unsigned   long   ADCResultAN7;
unsigned   long   ADCResultAN10;

void       ADCTest()
{
// Set RB2/AN4 as an analog input
ANSELBbits.ANSB2 = 1;
TRISBbits.TRISB2 = 1;
// Set RB12/AN7 as an analog input
ANSELBbits.ANSB12 = 1;
TRISBbits.TRISB12 = 1;
// Set RB15/AN10 as an analog input
ANSELBbits.ANSB15 = 1;
TRISBbits.TRISB15 = 1;
// Add 3 inputs above to the scan list
AD1CSS = (1<<4)|(1<<7)|(1<<10);
// Enable scan
AD1CON2bits.CSCNA = 1;
// Set number conversion per interrupt to the number of inputs (3)
AD1CON2bits.SMPI = 3-1;
// Clock from peripheral clock TPB
AD1CON3bits.ADRC = 0;
// ADC clock TAD = 8 peripheral clock TPB
AD1CON3bits.ADCS = 8;
// Enable sampling automatically after each conversion
AD1CON1bits.ASAM = 1;
// Set external trigger (from TIMER 1)
AD1CON1bits.SSRC = 5;
// Turn on the ADC
AD1CON1bits.ON = 1;
// Start the first sampling cycle
AD1CON1bits.SAMP = 1;
// Reset ADC interrupt flag
IFS0CLR = _IFS0_AD1IF_MASK;
// Set TIMER 1 period
PR1 = 0x2000;
// Enable TIMER 1
T1CONbits.ON = 1;

// Repeat continuously
while  (1)
{
    // Wait for the result
    while (IFS0bits.AD1IF == 0);
    // Reset ADC interrupt flag
    IFS0CLR = _IFS0_AD1IF_MASK;

    // Get ADC values
    ADCResultAN4 = ADC1BUF0;
    ADCResultAN7 = ADC1BUF1;
    ADCResultAN10 = ADC1BUF2;
}
}
```

25.5.6 Using Dual 8-Word Buffers

Example 25-6 demonstrates using dual 8-word buffers and alternating the buffer fill. Setting the BUFM bit (ADxCON2<1>) enables dual 8-word buffers. The BUFM bit setting does not affect other operational parameters. First, the conversion sequence starts filling the buffer at ADC1BUF0. After the first interrupt occurs, the buffer begins to fill at ADC1BUF8. The BUFS bit (ADxCON2<7>) is alternately set and cleared after each interrupt to show which buffer is being filled. In this example, three analog inputs are sampled and an interrupt occurs after every third sample.

Example 25-6: Conversion Code Using Dual Buffers

```
#include   <xc.h>

unsigned   long    ADCResultAN4;
unsigned   long    ADCResultAN7;
unsigned   long    ADCResultAN10;

void       ADCTest()
{
// Set RB2/AN4 as an analog input
ANSELBbits.ANSB2 = 1;
TRISBbits.TRISB2 = 1;
// Set RB12/AN7 as an analog input
ANSELBbits.ANSB12 = 1;
TRISBbits.TRISB12 = 1;
// Set RB15/AN10 as an analog input
ANSELBbits.ANSB15 = 1;
TRISBbits.TRISB15 = 1;
// Add 3 inputs above to the scan list
AD1CSS = (1<<4)|(1<<10)|(1<<12);
// Enable scan
AD1CON2bits.CSCNA = 1;
// Set number conversion per interrupt to the number of inputs (3)
AD1CON2bits.SMPI = 3-1;
// Slit buffer in 2
AD1CON2bits.BUFM = 1;
// Clock from peripheral clock TPB
AD1CON3bits.ADRC = 0;
// ADC clock TAD = 8 peripheral clock TPB
AD1CON3bits.ADCS = 8;
// Enable sampling automatically after each conversion
AD1CON1bits.ASAM = 1;
// Set external trigger (from TIMER 1)
AD1CON1bits.SSRC = 5;
// Turn on the ADC
AD1CON1bits.ON = 1;
// Start the first sampling cycle
AD1CON1bits.SAMP = 1;
// Reset ADC interrupt flag
IFS0CLR = _IFS0_AD1IF_MASK;
// Set TIMER1 period
PR1 = 0x2000;
// Enable TIMER1
T1CONbits.ON = 1;
// Repeat continuously
while  (1)
{
```

Example 25-6: Conversion Code Using Dual Buffers (Continued)

```
 // Wait for the result
    while (IFS0bits.AD1IF == 0);
    // Reset ADC interrupt flag
    IFS0CLR = _IFS0_AD1IF_MASK;
    // Read current buffer number
    if(AD1CON2bits.BUFS == 1)
    {
        // ADC is currently filling Buffers 8-10, read data from 0-2
        ADCResultAN4 = ADC1BUF0;
        ADCResultAN7 = ADC1BUF1;
        ADCResultAN10 = ADC1BUF2;
    }else{
        // ADC is currently filling Buffers 0-2, read data from 8-10
        ADCResultAN4 = ADC1BUF8;
        ADCResultAN7 = ADC1BUF9;
        ADCResultAN10 = ADC1BUF10;
    }
}
}
```

25.5.7 Analog-to-Digital Conversions in Sleep Mode Using Threshold Detect

Example 25-7 demonstrates the ADC operation in Sleep mode. The Auto-Scan with the Threshold Detect feature is used to wake-up the device if the voltage level on any of the three inputs exceeds the programmed threshold.

Example 25-7: Code Using Threshold Detection in Sleep Mode

```
#include   <xc.h>

void       ADCTest()
{
// Set RB2/AN4 as an analog input
ANSELBbits.ANSB2 = 1;
TRISBbits.TRISB2 = 1;
// Set RB12/AN7 as an analog input
ANSELBbits.ANSB12 = 1;
TRISBbits.TRISB12 = 1;
// Set RB15/AN10 as an analog input
ANSELBbits.ANSB15 = 1;
TRISBbits.TRISB15 = 1;
// Add 3 inputs above to the scan list
AD1CSS = (1<<4)|(1<<7)|(1<<10);
// Enable auto scan
AD1CON5bits.ASEN = 1;
// Low power mode after conversion sequence
AD1CON5bits.LPEN = 1;
// Enable Threshold Detect, discard the conversion result
AD1CON5bits.WM = 2;
// Set compare mode to "Greater Than"
AD1CON5bits.CM = 1;
// Generate interrupt to wake up only if a valid compare has occurred
AD1CON5bits.ASINT = 2;
// Set thresholds for all inputs
ADC1BUF4 = 3072;
ADC1BUF10 = 3072;
ADC1BUF12 = 3072;
// Clock from Fast RC oscillator to operate in Sleep
AD1CON3bits.ADRC = 1;
// ADC clock TAD = 8 RC oscillator cycles = 1uS
AD1CON3bits.ADCS = 8;
// Set sample time to 5 TAD
AD1CON3bits.SAMC = 5;
// Enable sampling automatically after each conversion
AD1CON1bits.ASAM = 1;
// Set external trigger (from TIMER 1 in Sleep mode)
AD1CON1bits.SSRC = 6;
// Turn on the ADC
AD1CON1bits.ON = 1;
// Start the first sampling cycle
AD1CON1bits.SAMP = 1;
// Enable ADC interrupt
IPC3bits.AD1IP = 3;
IEC0bits.AD1IE = 1;
// Initialize timer to generate triggers in Sleep mode
// Set TIMER 1 period
PR1 = 0x1000;
// Set LPRC as a clock source (active in Sleep mode)
T1CONbits.TECS = 2;
T1CONbits.TCS = 1;
// Enable TIMER 1
T1CONbits.ON = 1;
```

Example 25-7: Code Using Threshold Detection in Sleep Mode (Continued)

```
// Repeat continuously
while  (1)
{
    // Enter Sleep mode
    SYSKEY = 0xAA996655;
    SYSKEY = 0x556699AA;
    OSCCONbits.SLPEN = 1;
    asm    volatile("wait");

    // Here device will wake up if conversion result
    // on AN4, AN7 or AN10 will be greater than 3072,
    // toggle RB7 to indicate
    TRISBbits.TRISB7 = 0;
    LATBbits.LATB7 ^= 1;

}
}

void __attribute__ ((vector(_ADC_VECTOR), interrupt(IPL3SOFT)))
ADCSoftwareHandler(void)
{
    // Reset ADC interrupt flag
    IFS0CLR = _IFS0_AD1IF_MASK;
}
```

25.6 INTERRUPTS

The ADC module has a dedicated interrupt bit, ADxIF bit, and a corresponding interrupt enable/mask bit, ADxIE bit. These bits are used to determine the source of an interrupt and to enable or disable an individual interrupt source.

The ADxIF bit is set when the condition set by the Samples Per Interrupt bits, SMPI<4:0> (ADxCON2<6:2>), is met. The ADxIF bit will then be set without regard to the state of the corresponding ADxIE bit. The ADxIF bit can be polled by software if desired.

The ADxIE bit controls the interrupt generation. If the ADxIE bit is set, the CPU will be interrupted whenever an event defined by SMPI<4:0> occurs and the corresponding ADxIF bit will be set (subject to the priority and subpriority as outlined below).

It is the responsibility of the routine that services a particular interrupt to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of the ADC interrupt can be set independently through the ADxIP<2:0> bits. This priority defines the priority group that the interrupt source will be assigned to. The priority groups range from a value of 7, the highest priority, to a value of 0, which does not generate an interrupt. An interrupt being serviced will be preempted by an interrupt in a higher priority group.

The subpriority bits allow setting the relative priority of an interrupt source within a priority group. The values of the subpriority, ADxIS<1:0> bits, range from 3, the highest priority, to 0 the lowest priority. An interrupt with the same priority group, but having a higher subpriority value, will preempt a lower subpriority interrupt that is in progress.

The priority group and subpriority bits allow more than one interrupt source to share the same priority and subpriority. If simultaneous interrupts occur in this configuration, the subpriority of the interrupt sources within a priority/group determine the interrupt generated. If multiple interrupts of the same priority and subpriority are pending, the natural priority based on the vector numbers of the interrupt sources is used to determine which interrupt is serviced first. The lower vector number has a higher priority of interrupt.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The users code at this vector address should perform any operations required, such as reading the converted samples, clearing the interrupt flag and then exiting. Refer to the **"Interrupt Controller"** section in the device data sheet for vector address table details and for more information on interrupts.

25.7 OPERATION DURING SLEEP AND IDLE MODES

Sleep and Idle modes are useful for minimizing conversion noise because the digital activity of the CPU, buses and other peripherals is minimized.

When the device enters Sleep mode, all clock sources to the module are shut down and stay at logic '0'.

If Sleep occurs in the middle of a conversion, the conversion may be aborted unless the ADC module is clocked from its internal RC oscillator. If the converter resumes a partially completed conversion on exiting from Sleep mode, the converted results should be discarded as the Sample-and-Hold capacitor will not maintain the acquired voltage.

ADC register contents are not affected by the device entering or leaving Sleep mode with the exception of the SAMP bit (ADxCON1<1>). After leaving Sleep mode, the SAMP bit will be cleared.

The ADC module can operate during Sleep mode only if the ADC clock source is set to the Fast RC (FRC) oscillator (ADRC (ADxCON3<15>) = 1). The operation in Sleep reduces the digital switching noise from the conversion. When the conversion is completed, the DONE bit (ADxCON1<0>) will be set and the result loaded into the ADC result buffer.

If the ADC interrupt is enabled (ADxIE bit = 1), the device will wake-up from Sleep when the ADC interrupt occurs. Program execution will resume at the ADC Interrupt Service Routine (ISR) if the ADC interrupt is greater than the current CPU priority. Otherwise, execution will continue from the instruction after the WAIT instruction that placed the device in Sleep mode.

Note:	For the ADC module to operate in Sleep mode, the ADC clock source must be set to the Fast RC oscillator (ADRC bit = 1).

25.7.1 ADC Operation During CPU IDLE Mode

For the ADC, the SIDL bit (ADxCON1<13>) specifies whether the module will stop on Idle or continue on Idle. If the SIDL bit = 0, the ADC module will continue normal operation when the device enters Idle mode. If the ADC interrupt is enabled (ADxIE bit = 1), the device will wake-up from Idle mode when the ADC interrupt occurs. Program execution will resume at the ADC ISR if the ADC interrupt is greater than the current CPU priority. Otherwise, execution will continue from the instruction after the WAIT instruction that placed the device in Idle mode.

If the SIDL bit = 1, the ADC module will stop in Idle mode and internal analog biasing circuitry will be shut down, reducing Idle current. If the device enters Idle mode in the middle of a conversion, the conversion is aborted. The converter will not resume a partially completed conversion on exiting from Idle mode.

25.8 EFFECT OF RESET

Following a Reset, all of the ADC Control registers (ADxCON1, ADxCON2, ADxCON3, ADxCON5, ADxCHIT, ADxCHS, ANSELx and ADxCSS) are reset to a value of '0x00000000'. This disables the ADC module and sets the analog input pins to Analog Input mode. Any conversion that was in progress will terminate and the result will not be written to the result buffer.

The values in the ADCxBUFn registers are initialized to zero during a Reset.

25.9 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32 device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the 12-Bit Analog-to-Digital Converter (ADC) with Threshold Detect module are:

Title	Application Note #
Using the Analog-to-Digital (A/D) Converter	AN546
Four-Channel Digital Voltmeter with Display and Keyboard	AN557
Understanding A/D Converter Performance Specifications	AN693

Note: Please visit the Microchip web site (www.microchip.com) for additional Application Notes and code examples for the PIC32 family of devices.

25.10 REVISION HISTORY

Revision A (September 2015)

This is the initial version of the document.

Revision B (May 2016)

Changed the title for Figure 25-1.

Updated bit values for ADCS<7:0> in Figure 25-3.

This revision also includes grammatical corrections throughout the document.

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-0642-6

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

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

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

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

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

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 - 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: 86-21-5407-5533
Fax: 86-21-5407-5066

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: 886-7-213-7828

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

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

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Karlsruhe
Tel: 49-721-625370

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

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

Italy - Venice
Tel: 39-049-7625286

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

Poland - Warsaw
Tel: 48-22-3325737

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

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

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

07/14/15