12-Bit High-Speed, Multiple SARs A/D Converter (ADC)

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 dsPIC33/PIC24 devices. Please consult the note at the beginning of the chapter in the specific 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.

1.0 INTRODUCTION

The dsPIC33/PIC24 12-Bit High-Speed, Multiple SARs Analog-to-Digital Converter (ADC) includes the following features:

- Multiple ADC Cores:
  - Multiple single channel dedicated ADC cores (depending on the specific device implementation)
  - One shared (common) ADC core
- Configurable 6, 8, 10 or 12-Bit Resolution for each ADC Core
- Up to 3.25 Msps Conversion Rate per Channel for 12-Bit Resolution
- Up to 32 Analog Input Sources (depending on the specific device implementation)
- Single-Ended or Pseudodifferential Inputs on a per Channel Basis for All Channels
- Conversion Result can be Formatted as Unsigned or Signed Data on a per Channel Basis for All Channels
- Separate 16-Bit Conversion Result Register for each Analog Input
- Early Interrupt Generation to enable Fast Processing of Converted Data
- Multiple Integrated Digital Comparators (depending on the specific device implementation):
  - Multiple comparison options
  - Assignable to specific analog inputs
- Multiple Oversampling Filters (depending on the specific device implementation):
  - Provides increased resolution
  - Assignable to a specific analog input
- Operation during CPU Sleep and Idle modes
- Hardware Capacitive Voltage Divider (CVD) to Measure Capacitance Connected to the Input

Simplified block diagrams of the Multiple SARs 12-Bit ADC are illustrated in Figure 1-1, Figure 1-2 and Figure 1-3.

The module consists of a few independent SAR ADC cores. The analog inputs (channels) are connected through multiplexers and switches to the Sample-and-Hold (S/H) circuit of each ADC core. The core uses the channel information (output format, Measurement mode and input number) to process the analog sample. When conversion is complete, the result is stored in the result buffer for the specific analog input, and passed to the digital filter and digital comparator if they were configured to use data from this particular channel.

Figure 1-1: 12-Bit High-Speed, Multiple SARs ADC Block Diagram

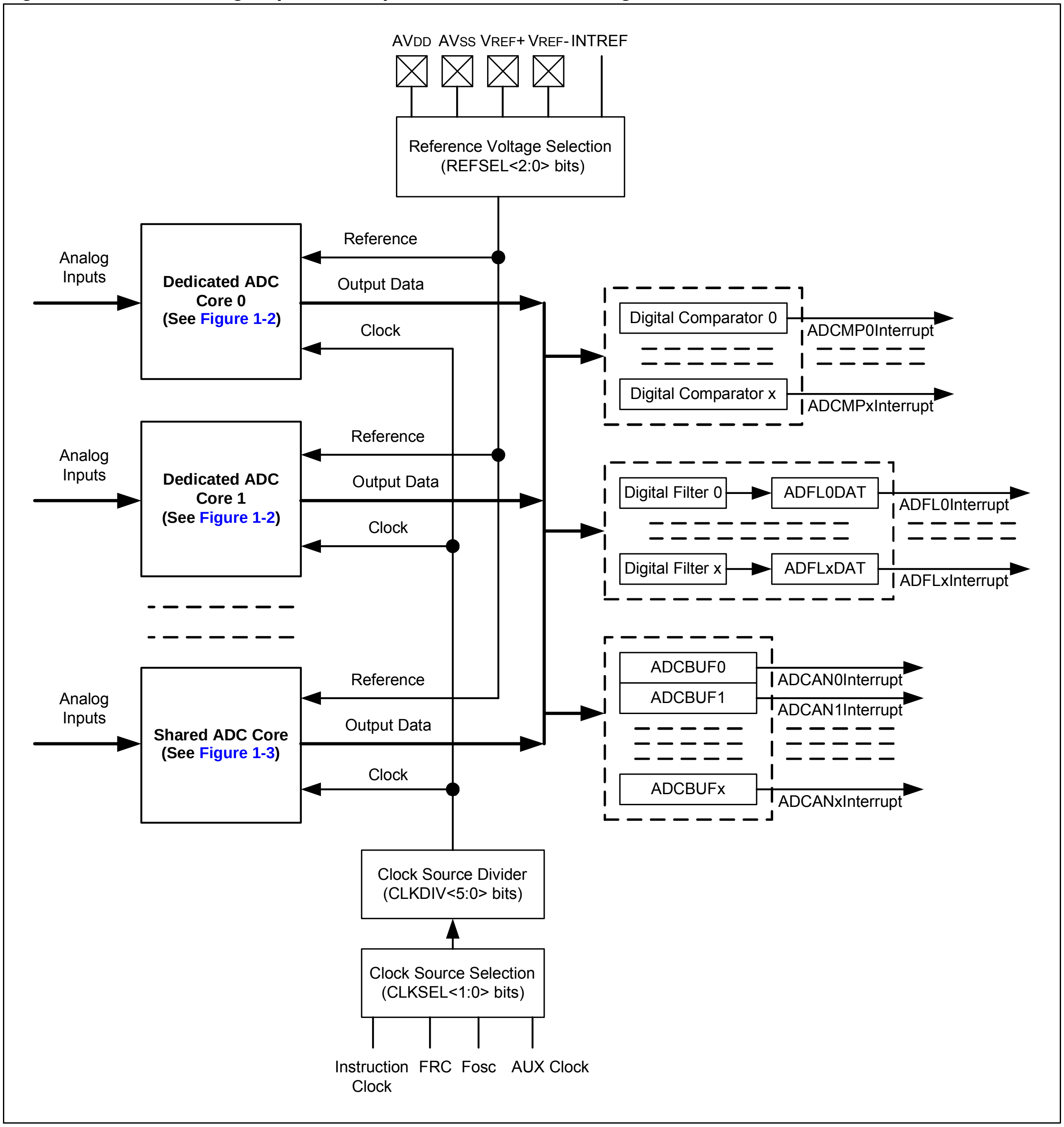


Figure 1-2: Dedicated ADC Core

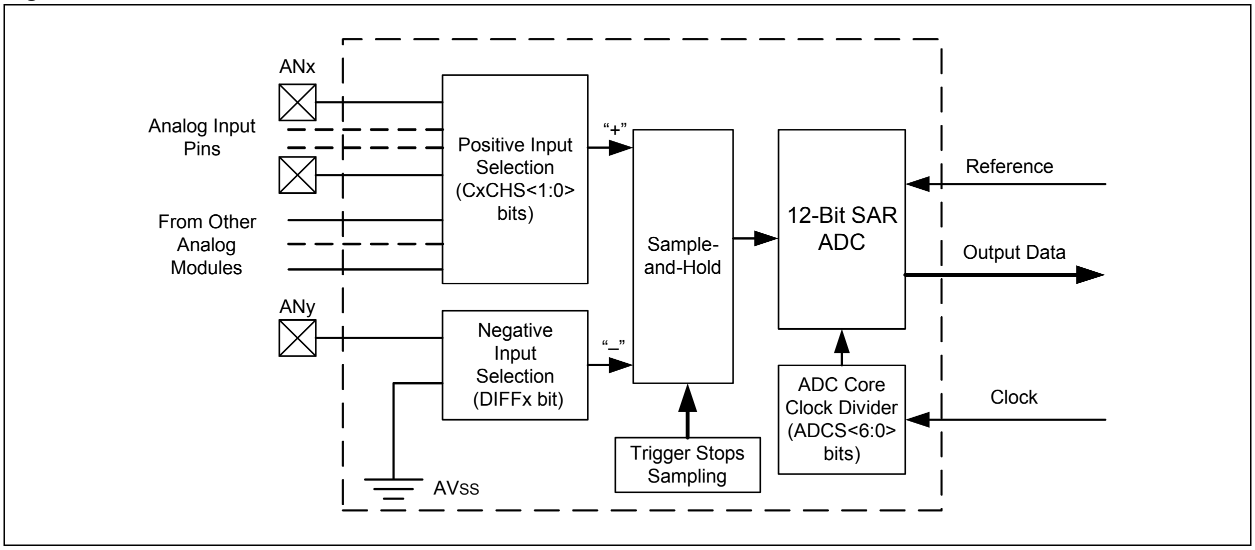
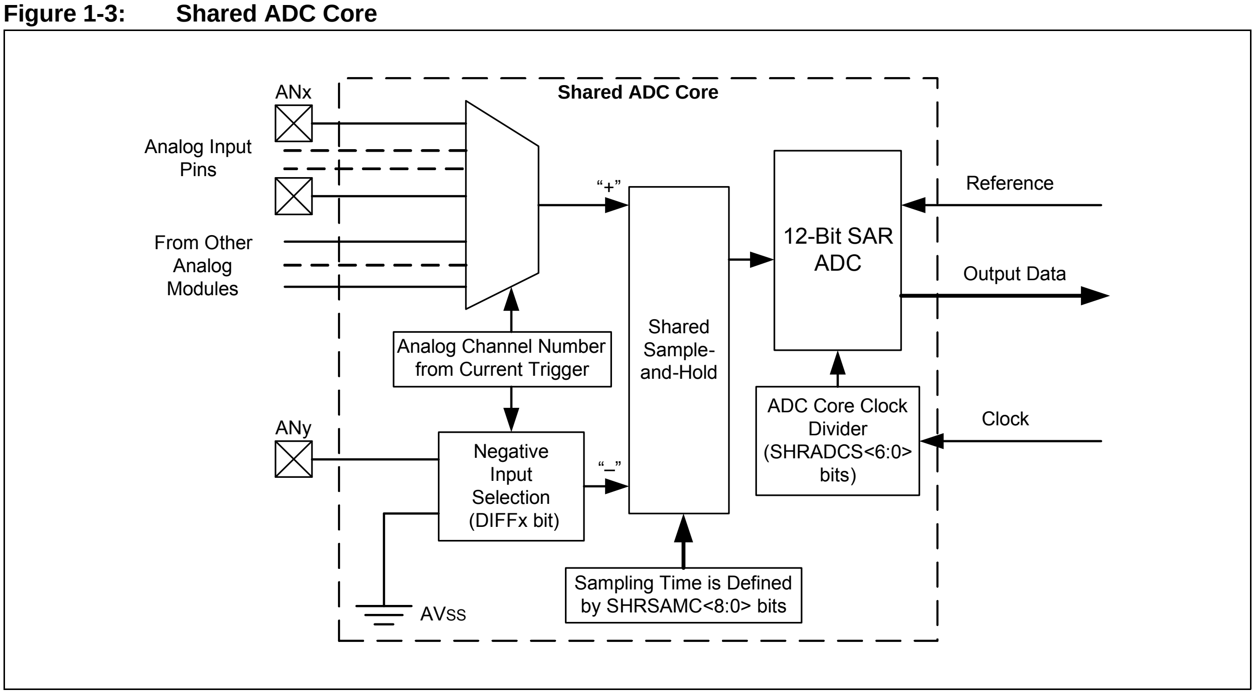


Figure 1-3: Shared ADC Core



2.0 REGISTERS

The Special Function Registers (SFRs) of the 12-Bit High-Speed, Multiple SARs ADC module are divided into two groups: control registers and data registers. A complete list of all SFRs implemented by the ADC is provided in Table 8-1.

2.1 Control Registers

The ADCON1L register (Register 2-1) contains bits to enable the module, define the module behavior in Idle mode and enable the CVD feature.

The ADCON1H register (Register 2-2) controls the output data format and the shared ADC core resolution.

The ADCON2L register (Register 2-3) controls the clock divider and early interrupt timing selection for the shared ADC core. It has bits to enable the common interrupt for the events related to the voltage reference and a bit to enable an early interrupt feature for the individual input channels.

The ADCON2H register (Register 2-4) controls the sampling time for the shared ADC core. It also provides the status bits, which indicate that the module voltage reference is ready for operation. This register allows adjusting the internal capacitance value for the CVD feature.

The ADCON3L register (Register 2-5) selects the voltage reference for all ADC cores and controls common, level and single-shot software triggers. Also, it has control bits to suspend all triggers for the module.

The ADCON3H register (Register 2-6) has bits to enable all ADC cores and select a clock source for the module. Also, this register controls the module clock source divider.

The ADCON4L register (Register 2-7) allows enabling a delay between trigger and conversion for the dedicated ADC cores, and triggers synchronization.

The ADCON4H register (Register 2-8) selects channels for the dedicated ADC cores.

The ADCON5L register (Register 2-9) controls power for all ADC cores.

The ADCON5H register (Register 2-10) has bits to enable a common interrupt for each ADC core when it is powered on and ready for operation. Also in this register, the power-on delay is specified for all ADC cores.

The ADCOREnL (where 'n' is a dedicated ADC core number) registers (Register 2-11) define a delay between trigger and conversion for each dedicated ADC core.

The ADCOREnH (where 'n' is a dedicated ADC core number) registers (Register 2-12) define resolution, early interrupt time selection and the ADC core clock divider for each dedicated ADC core.

The ADLVLTRGL and ADLVLTRGH registers (Register 2-13 and Register 2-14) have bits to select either the level-sensitive trigger or the edge-sensitive trigger for each input channel.

The ADEIEL and ADEIEH registers (Register 2-15 and Register 2-16) have bits to enable the early interrupts generation for each input channel.

The ADEISTATL and ADEISTATH registers (Register 2-17 and Register 2-18) contain the early interrupts status flags for each input channel.

The ADMOD0L, ADMOD0H, ADMOD1L and ADMOD1H registers (Register 2-19 through Register 2-22) have bits to enable the Pseudodifferential mode and signed output data format for each input channel.

The ADIEL and ADIEH registers (Register 2-23 and Register 2-24) have bits to enable the individual and common interrupts for each input channel.

The ADSTATL and ADSTATH registers (Register 2-25 and Register 2-26) contain the data ready flags for each input channel.

The ADTRIGnL and ADTRIGnH registers (Register 2-27) define a trigger source for each input channel.

The ADCAL0L, ADCAL0H, ADCAL1L and ADCAL1H registers (Register 2-28 through Register 2-31) control the calibration for each ADC core. The calibration is not required for some devices. Refer to the specific device data sheet to see if these registers are implemented.

The ADCMPnCON registers (Register 2-32) control the operation of the digital comparators, including the generation of the interrupts and the comparison criteria to be used. These registers also provide the status when a comparator event occurs. One register is provided for each digital comparator.

The ADCMPnENL and ADCMPnENH registers (Register 2-33 and Register 2-34) select which of the analog input conversion results are to be processed by the digital comparator. One pair (L and H) is provided for each digital comparator.

The ADFLnCON registers (Register 2-35) control the operation of the oversampling filters and provide status bits for the filters' operation. One register is provided for each oversampling filter. The ADCSSL and ADCSSH registers (Register 2-36 and Register 2-37) select which of the analog inputs are to be scanned/processed by the CVD.

2.2 Data Registers

The ADCBUFx registers store the output data of the Analog-to-Digital conversion. In general, there is one register provided for each of the implemented analog channels; each channel will have a corresponding numbered ADCBUFx register. Although the registers are 16 bits wide, the usage of the registers for storing the 12-bit conversion results is determined by the selected data output format. See **Section 4.10 "Conversion Result"** for more information.

The ADCMPnLO and ADCMPnHI registers store the 16-bit high and low digital comparison values for use by the digital comparators. One pair (HI and LO) is provided for each ADC comparator.

The ADFLnDAT registers contain the 16-bit output data from the oversampling filters. There is one register for each oversampling filter.

The ADCVDDAT register contains the 16-bit output data from the Capacitive Voltage Divider (CVD). This register may not be implemented on some devices. Refer to the specific device data sheet to see if the CVD feature is implemented on the device.

Register 2-1: ADCON1L: ADC Control Register 1 Low

R/W-0	U-0	R/W-0	U-0	R/W-0	U-0	U-0	U-0
ADON[1]	—	ADSIDL	—	CVDEN[2]	—	—	—
bit 15							bit 8

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
NRE[3]	—	—	—	—	—	—	—
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 15 **ADON:** ADC Enable bit[1]
1 = ADC module is enabled
0 = ADC module is off

bit 14 **Unimplemented:** Read as ‘0’

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

bit 12 **Unimplemented:** Read as ‘0’

bit 11 **CVDEN:** CVD Enable bit[2]
1 = CVD is enabled
0 = CCD is off

bit 10-8 **Unimplemented:** Read as ‘0’

bit 7 **NRE:** Noise Reduction Enable bit[3]
1 = Holds conversion process for 1 TADCORE when another core completes conversion to reduce noise between cores
0 = Noise Reduction feature is disabled

bit 6-0 **Unimplemented:** Read as ‘0’

Note 1: Set the ADON bit only after the ADC module has been configured. Changing ADC Configuration bits when ADON = 1 will result in unpredictable behavior.

2: The CVD feature is not available on all devices and the CVDEN bit may not be implemented. Refer to the device data sheet for more information.

3: The Noise Reduction feature is not available on all devices and the NRE bit may not be implemented. Refer to the device data sheet for more information.

Register 2-2: ADCON1H: ADC Control Register 1 High

r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-1	R/W-1	r-0	r-0	r-0	r-0	r-0
FORM	SHRRES1	SHRRES0	—	—	—	—	—
bit 7							bit 0

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

bit 15-8 **Reserved:** Must be written as ‘0’

bit 7 **FORM:** Fractional Data Output Format bit

1 = Fractional
0 = Integer

bit 6-5 **SHRRES<1:0>:** Shared ADC Core Resolution Selection bits

11 = 12-bit resolution
10 = 10-bit resolution
01 = 8-bit resolution
00 = 6-bit resolution

bit 4-0 **Reserved:** Must be written as ‘0’

Register 2-3: ADCON2L: ADC Control Register 2 Low

R/W-0	R/W-0	r-0	R/W-0	r-0	R/W-0	R/W-0	R/W-0
REFCIE	REFERCIE[(2)]	—	EIEN	—	SHREISEL2[(1)]	SHREISEL1[(1)]	SHREISEL0[(1)]
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SHRADCS6	SHRADCS5	SHRADCS4	SHRADCS3	SHRADCS2	SHRADCS1	SHRADCS0
bit 7							bit 0

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

bit 15 **REFCIE:** Band Gap and Reference Voltages Ready Common Interrupt Enable bit
1 = Common interrupt will be generated when band gap and reference voltage are ready
0 = Common interrupt is disabled for band gap and reference voltage ready event

bit 14 **REFERCIE:** Band Gap and Reference Voltages Error Common Interrupt Enable bit[(2)]
1 = Common interrupt will be generated when band gap or reference voltage error is detected
0 = Common interrupt is disabled for band gap and reference voltages error event

bit 13 **Reserved:** Must be written as '0'

bit 12 **EIEN:** Early Interrupts Enable bit
1 = The early interrupt feature is enabled for the input channel interrupts (when the EISTATx flag is set)
0 = The individual interrupts are generated when the conversion is done (when the ANxRDY flag is set)

bit 11 **Reserved:** Must be written as '0'

bit 10-8 **SHREISEL<2:0>:** Shared Core Early Interrupt Time Selection bits[(1)]
111 = Early interrupt is generated 8 TADCORE clocks prior to when the data is ready
110 = Early interrupt is generated 7 TADCORE clocks prior to when the data is ready
101 = Early interrupt is generated 6 TADCORE clocks prior to when the data is ready
100 = Early interrupt is generated 5 TADCORE clocks prior to when the data is ready
011 = Early interrupt is generated 4 TADCORE clocks prior to when the data is ready
010 = Early interrupt is generated 3 TADCORE clocks prior to when the data is ready
001 = Early interrupt is generated 2 TADCORE clocks prior to when the data is ready
000 = Early interrupt is generated 1 TADCORE clock prior to when the data is ready

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

bit 6-0 **SHRADCS<6:0>:** Shared ADC Core Input Clock Divider bits
These bits determine the number of TCORESRC (Source Clock Periods) for one shared TADCORE (Core Clock Period).
1111111 = 254 source clock periods
•
•
•
0000011 = 6 source clock periods
0000010 = 4 source clock periods
0000001 = 2 source clock periods
0000000 = 2 source clock periods

Note 1: For the 6-bit shared ADC core resolution (SHRRES<1:0> = 00), the SHREISEL<2:0> settings, from '100' to '111', are not valid and should not be used. For the 8-bit shared ADC core resolution (SHRRES<1:0> = 01), the SHREISEL<2:0> settings, '110' and '111', are not valid and should not be used.

2: To avoid false interrupts, the REFERCIE bit must be set only after the module is enabled (ADON = 1).

Register 2-4: ADCON2H: ADC Control Register 2 High

R/HS/HC-0	R/HS/HC-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
REFRDY	REFERR	—	CVDCAP2[(1)]	CVDCAP1[(1)]	CVDCAP0[(1)]	SHRSAMC9	SHRSAMC8
bit 15							bit 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
SHRSAMC7	SHRSAMC6	SHRSAMC5	SHRSAMC4	SHRSAMC3	SHRSAMC2	SHRSAMC1	SHRSAMC0
bit 7							bit 0

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

bit 15 **REFRDY:** Band Gap and Reference Voltages Ready Flag bit
1 = Band gap and reference voltages are ready
0 = Band gap and reference voltages are not ready

bit 14 **REFERR:** Band Gap or Reference Voltage Error Flag bit
1 = Band gap or reference voltage was interrupted after the ADC module was enabled (ADON = 1)
0 = No band gap or reference voltage error was detected

bit 13 **Reserved:** Must be written as ‘0’

bit 12-10 **CVDCAP<2:0>:** CVD Additional Capacitance Selection bits[(1)]
This capacitance is added to the shared core Sample-and-Hold Capacitance (CHOLD) when CVD is enabled.
111 = 7 * 2.5 pF = 17.5 pF
110 = 6 * 2.5 pF = 15 pF
101 = 5 * 2.5 pF = 12.5 pF
100 = 4 * 2.5 pF = 10 pF
011 = 3 * 2.5 pF = 7.5 pF
010 = 2 * 2.5 pF = 5 pF
001 = 1 * 2.5 pF = 2.5 pF
000 = 0 * 2.5 pF = 0 pF

bit 9-0 **SHRSAMC<2:0>:** Shared ADC Core Sample Time Selection bits
These bits specify the number of shared Core Clock Periods (TADCORE) for the shared ADC core sample time.
1111111111 = 1025 TADCORE
•
•
•
0000000001 = 3 TADCORE
0000000000 = 2 TADCORE

Note 1: The CVD feature is not available on all devices and the CVDCAP<2:0> bits may not be implemented. Refer to the device data sheet for more information.

Register 2-5: ADCON3L: ADC Control Register 3 Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/HS/HC-0	R/W-0	R/W/HC-0
REFSEL2[1]	REFSEL1[1]	REFSEL0[1]	SUSPEND	SUSPCIE	SUSPRDY	SHRSAMP	CNVRTCH
bit 15							bit 8

R/W-0	R/W/HC-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SWLCTRG	SWCTRG	CNVCHSEL5	CNVCHSEL4	CNVCHSEL3	CNVCHSEL2	CNVCHSEL1	CNVCHSEL0
bit 7							bit 0

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

bit 15-13 **REFSEL<2:0>:** ADC Reference Voltage Selection bits[1]

bit 12 **SUSPEND:** All ADC Cores Triggers Disable bit
1 = All new trigger events for all ADC cores are disabled
0 = All ADC cores can be triggered

bit 11 **SUSPCIE:** Suspend All ADC Cores Common Interrupt Enable bit
1 = Common interrupt will be generated when ADC cores triggers are suspended (SUSPEND bit = 1) and all previous conversions are finished (SUSPRDY bit becomes set)
0 = Common interrupt is not generated for suspend ADC cores event

bit 10 **SUSPRDY:** All ADC Cores Suspended Flag bit
1 = All ADC cores are suspended (SUSPEND bit = 1) and have no conversions in progress
0 = ADC cores have previous conversions in progress

bit 9 **SHRSAMP:** Shared ADC Core Sampling Direct Control bit
This bit should be used with the individual channel conversion trigger controlled by the CNVRTCH bit. It connects an analog input specified by the CNVCHSEL<5:0> bits to the shared ADC core and allows extending the sampling time. This bit is not controlled by hardware and must be cleared before the conversion starts (setting CNVRTCH to ‘1’).
1 = Shared ADC core samples an analog input specified by the CNVCHSEL<5:0> bits
0 = Sampling is controlled by the shared ADC core hardware

bit 8 **CNVRTCH:** Software Individual Channel Conversion Trigger bit
1 = Single trigger is generated for an analog input specified by the CNVCHSEL<5:0> bits; when the bit is set, it is automatically cleared by hardware on the next instruction cycle
0 = Next individual channel conversion trigger can be generated

bit 7 **SWLCTRG:** Software Level-Sensitive Common Trigger bit
1 = Triggers are continuously generated for all channels when the software level-sensitive common trigger is selected as a source in the ADTRIGnL and ADTRIGnH registers
0 = No software level-sensitive common triggers are generated

bit 6 **SWCTRG:** Software Common Trigger bit
1 = Single trigger is generated for all channels when the software common trigger is selected as a source in the ADTRIGnL and ADTRIGnH registers; when the bit is set, it is automatically cleared by hardware on the next instruction cycle
0 = Ready to generate the next software common trigger

bit 5-0 **CNVCHSEL<5:0>:** Channel Number Selection for Software Individual Channel Conversion Trigger bits
These bits define a channel to be converted when the CNVRTCH bit is set.

Note 1: Refer to the specific device data sheet for the available reference voltage source options.

Register 2-6: ADCON3H: ADC Control Register 3 High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CLKSEL1[(1)]	CLKSEL0[(1)]	CLKDIV5	CLKDIV4	CLKDIV3	CLKDIV2	CLKDIV1	CLKDIV0
bit 15							bit 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
SHREN	C6EN[(2)]	C5EN[(2)]	C4EN[(2)]	C3EN[(2)]	C2EN[(2)]	C1EN[(2)]	C0EN[(2)]
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 15-14 **CLKSEL<1:0>:** ADC Module Clock Source Selection bits[(1)]

bit 13-8 **CLKDIV<5:0>:** ADC Module Clock Source Divider bits

The divider forms a $T_{CORESRC}$ clock used by all ADC cores (shared and dedicated) from the T_{SRC} ADC module clock source selected by the CLKSEL<1:0> bits. Then, each ADC core individually divides the $T_{CORESRC}$ clock to get a core-specific T_{ADCORE} clock, using the ADCS<6:0> bits in the ADCOREnH register, or the SHRADCS<6:0> bits in the ADCON2L register.

111111 = 64 source clock periods
•
•
•
000011 = 4 source clock periods
000010 = 3 source clock periods
000001 = 2 source clock periods
000000 = 1 source clock period

bit 7 **SHREN:** Shared ADC Core Enable bit

This bit does not disable the core clock and analog bias circuitry.
1 = Shared ADC core is enabled
0 = Shared ADC core is disabled

bit 6-0 **C6EN:C0EN:** Dedicated ADC Core x Enable bits[(2)]

This bit does not disable the core clock and analog bias circuitry.
1 = Dedicated ADC Core x is enabled
0 = Dedicated ADC Core x is disabled

Note 1: Refer to the specific device data sheet for the available ADC module clock source options.

2: The number of the available dedicated ADC cores is device-specific and some CxEN bits may not be implemented. Refer to the device data sheet for more information.

Register 2-7: ADCON4L: ADC Control Register 4 Low

U-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SAMC6EN(1)	SAMC5EN(1)	SAMC4EN(1)	SAMC3EN(1)	SAMC2EN(1)	SAMC1EN(1)	SAMC0EN(1)
bit 7							bit 0

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

bit 15 **Unimplemented:** Read as ‘0’

bit 14-8 **Reserved:** Must be written as ‘0’

bit 7 **Unimplemented:** Read as ‘0’

bit 6-0 **SAMC6EN:SAMC0EN:** Dedicated ADC Core x Conversion Delay Enable bits(1)

1 = After the trigger, the conversion will be delayed and the ADC core will continue sampling during the time specified by the SAMC<9:0> bits in the ADCOREnL register

0 = After the trigger, the sampling will be stopped immediately and the conversion will be started on the next core clock cycle

Note 1: The number of available dedicated ADC cores is device-specific and some SAMCxEN bits may not be implemented. Refer to the device data sheet for more information.

Register 2-8: ADCON4H: ADC Control Register 4 High

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	C6CHS1[1]	C6CHS0[1]	C5CHS1[1]	C5CHS0[1]	C4CHS1[1]	C4CHS0[1]
bit 15							bit 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
C3CHS1[1]	C3CHS0[1]	C2CHS1[1]	C2CHS0[1]	C1CHS1[1]	C1CHS0[1]	C0CHS1[1]	C0CHS0[1]
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 15-14 **Unimplemented:** Read as ‘0’

bit 13-12 **C6CHS<1:0>:** Dedicated ADC Core 6 Input Channel Selection bits[1]

bit 11-10 **C5CHS<1:0>:** Dedicated ADC Core 5 Input Channel Selection bits[1]

bit 9-8 **C4CHS<1:0>:** Dedicated ADC Core 4 Input Channel Selection bits[1]

bit 7-6 **C3CHS<1:0>:** Dedicated ADC Core 3 Input Channel Selection bits[1]

bit 5-4 **C2CHS<1:0>:** Dedicated ADC Core 2 Input Channel Selection bits[1]

bit 3-2 **C1CHS<1:0>:** Dedicated ADC Core 1 Input Channel Selection bits[1]

bit 1-0 **C0CHS<1:0>:** Dedicated ADC Core 0 Input Channel Selection bits[1]

Note 1: The number of available dedicated ADC cores and input channel options for each ADC core are device-specific. Some CxCHS<1:0> bits may not be implemented. Refer to the device data sheet for the available ADC cores and their input channel options.

Register 2-9: ADCON5L: ADC Control Register 5 Low

R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0
SHRRDY	C6RDY(1)	C5RDY(1)	C4RDY(1)	C3RDY(1)	C2RDY(1)	C1RDY(1)	C0RDY(1)
bit 15							bit 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
SHRPWR	C6PWR(1)	C5PWR(1)	C4PWR(1)	C3PWR(1)	C2PWR(1)	C1PWR(1)	C0PWR(1)
bit 7							bit 0

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

bit 15 **SHRRDY:** Shared ADC Core Ready Flag bit
1 = ADC core is powered and ready for operation
0 = ADC core is not ready for operation

bit 14-8 **C6RDY:C0RDY:** Dedicated ADC Core x Ready Flag bits(1)
1 = ADC core is powered and ready for operation
0 = ADC core is not ready for operation

bit 7 **SHRPWR:** Shared ADC Core Power Enable bit
1 = ADC core is powered
0 = ADC core is off

bit 6-0 **C6PWR:C0PWR:** Dedicated ADC Core x Power Enable bits(1)
1 = ADC core is powered
0 = ADC core is off

Note 1: The number of available dedicated ADC cores is device-specific. Some CxRDY and CxPWR bits may not be implemented. Refer to the device data sheet for the available ADC cores.

Register 2-10: ADCON5H: ADC Control Register 5 High

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	WARMTIME3	WARMTIME2	WARMTIME1	WARMTIME0
bit 15							bit 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
SHRCIE	C6CIE(1)	C5CIE(1)	C4CIE(1)	C3CIE(1)	C2CIE(1)	C1CIE(1)	C0CIE(1)
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 15-12 **Unimplemented:** Read as '0'

bit 11-8 **WARMTIME<3:0>:** ADC Cores Power-up Delay bits

These bits determine the power-up delay in the number of the Core Clock Source periods ($T_{CORESRC}$) for all ADC cores.

1111 = 32768 source clock periods
1110 = 16384 source clock periods
1101 = 8192 source clock periods
1100 = 4096 source clock periods
1011 = 2048 source clock periods
1010 = 1024 source clock periods
1001 = 512 source clock periods
1000 = 256 source clock periods
0111 = 128 source clock periods
0110 = 64 source clock periods
0101 = 32 source clock periods
0000-0100 = 16 source clock periods

bit 7 **SHRCIE:** Shared ADC Core Ready Common Interrupt Enable bit

1 = Common interrupt will be generated when the ADC core is powered and ready for operation
0 = Common interrupt is disabled for the ADC core ready event

bit 6-0 **C6CIE:C0CIE:** Dedicated ADC Core x Ready Common Interrupt Enable bits(1)

1 = Common interrupt will be generated when the ADC core is powered and ready for operation
0 = Common interrupt is disabled for the ADC core ready event

Note 1: The number of available dedicated ADC cores is device-specific. Some CxCIE bits may not be implemented. Refer to the device data sheet for the available ADC cores.

Register 2-11: ADCOREnL: Dedicated ADC Core n Control Register Low[1]

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	SAMC<9:8>	
bit 15							bit 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
SAMC<7:0>							
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 15-10 **Unimplemented:** Read as '0'

bit 9-0 **SAMC<9:0>:** Dedicated ADC Core n Conversion Delay Selection bits

These bits determine the time between the trigger event and the start of conversion in the number of the Core Clock periods (TADCORE). During this time, the ADC core continues sampling. This feature is enabled by the SAMCxEN bits in the ADCON4L register.

1111111111 = 1025 TADCORE
•
•
•
0000000001 = 3 TADCORE
0000000000 = 2 TADCORE

Note 1: The number of available dedicated ADC cores is device-specific. Refer to the device data sheet for the available ADC cores.

Register 2-12: ADCOREnH: Dedicated ADC Core n Control Register High[1]

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
—	—	—	EISEL2[2]	EISEL1[2]	EISEL0[2]	RES1	RES0
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
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 15-13 **Unimplemented:** Read as '0'

bit 12-10 **EISEL<2:0>:** ADC Core n Early Interrupt Time Selection bits[2]

111 = Early interrupt is generated 8 TADCORE clocks prior to when the data is ready
110 = Early interrupt is generated 7 TADCORE clocks prior to when the data is ready
101 = Early interrupt is generated 6 TADCORE clocks prior to when the data is ready
100 = Early interrupt is generated 5 TADCORE clocks prior to when the data is ready
011 = Early interrupt is generated 4 TADCORE clocks prior to when the data is ready
010 = Early interrupt is generated 3 TADCORE clocks prior to when the data is ready
001 = Early interrupt is generated 2 TADCORE clocks prior to when the data is ready
000 = Early interrupt is generated 1 TADCORE clock prior to when the data is ready

bit 9-8 **RES<1:0>:** ADC Core n Resolution Selection bits

11 = 12-bit resolution
10 = 10-bit resolution
01 = 8-bit resolution
00 = 6-bit resolution

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

bit 6-0 **ADCS<6:0>:** ADC Core x Input Clock Divider bits

These bits determine the number of Core Clock Source periods (TCORESRC) for one Core Clock period (TADCORE).

1111111 = 254 source clock periods
•
•
•
0000011 = 6 source clock periods
0000010 = 4 source clock periods
0000001 = 2 source clock periods
0000000 = 2 source clock periods

Note 1: The number of available dedicated ADC cores is device-specific. Refer to the device data sheet for the available ADC cores.

2: For the 6-bit ADC core resolution (RES<1:0> = 00), the EISEL<2:0> settings, from '100' to '111', are not valid and should not be used. For the 8-bit ADC core resolution (RES<1:0> = 01), the EISEL<2:0> settings, '110' and '111', are not valid and should not be used.

Register 2-13: ADLVLTRGL: ADC Level-Sensitive Trigger Control Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
LVLEN<15:8>[(1)]							
bit 15							bit 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
LVLEN<7:0>[(1)]							
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 15-0 **LVLEN<15:0>:** Level Trigger Enable bits[(1)]
1 = Input channel trigger is level-sensitive
0 = Input channel trigger is edge-sensitive

Note 1: The number of available ADC channels is device-specific. Some LVLENx bits may not be implemented. Refer to the device data sheet for the available ADC cores and their input channel options.

Register 2-14: ADLVLTRGH: ADC Level-Sensitive Trigger Control Register High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
LVLEN<31:24>[(1)]							
bit 15							bit 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
LVLEN<23:16>[(1)]							
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 15-0 **LVLEN<31:16>:** Level Trigger Enable bits[(1)]
1 = Input channel trigger is level-sensitive
0 = Input channel trigger is edge-sensitive

Note 1: The number of available ADC channels is device-specific. Some LVLENx bits may not be implemented. Refer to the device data sheet for the available ADC cores and their input channel options.

Register 2-15: ADEIEL: ADC Early Interrupt Enable Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EIE<15:8>[1]							
bit 15							bit 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
EIE<7:0>[1]							
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 15-0 **EIE<15:0>:** Early Interrupt Enable for Corresponding Analog Inputs bits[1]
`1` = Early interrupt is enabled for the channel
`0` = Early interrupt is disabled for the channel

Note 1: The available channels are device-specific. Some EIEx bits may not be implemented. Refer to the device data sheet for the available channel information.

Register 2-16: ADEIEH: ADC Early Interrupt Enable Register High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EIE<31:24>[1]							
bit 15							bit 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
EIE<23:16>[1]							
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 15-0 **EIE<31:16>:** Early Interrupt Enable for Corresponding Analog Inputs bits[1]
`1` = Early interrupt is enabled for the channel
`0` = Early interrupt is disabled for the channel

Note 1: The available channels are device-specific. Some EIEx bits may not be implemented. Refer to the device data sheet for the available channel information.

Register 2-17: ADEISTATL: ADC Early Interrupt Status Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EISTAT<15:8>[1]							
bit 15							bit 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
EISTAT<7:0>[1]							
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 15-0 **EISTAT<15:0>:** Early Interrupt Status for Corresponding Analog Inputs bits[1]
`1` = Early interrupt was generated
`0` = Early interrupt was not generated since the last ADCBUFx read

Note 1: The available channels are device-specific. Some EISTATx bits may not be implemented. Refer to the device data sheet for the available channel information.

Register 2-18: ADEISTATH: ADC Early Interrupt Status Register High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EISTAT<31:24>[1]							
bit 15							bit 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
EISTAT<23:16>[1]							
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 15-0 **EISTAT<31:16>:** Early Interrupt Status for Corresponding Analog Inputs bits[1]
`1` = Early interrupt was generated
`0` = Early interrupt was not generated since the last ADCBUFx read

Note 1: The available channels are device-specific. Some EISTATx bits may not be implemented. Refer to the device data sheet for the available channel information.

Register 2-19: ADMOD0L: ADC Input Mode Control Register 0 Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DIFF7[1]	SIGN7[1]	DIFF6[1]	SIGN6[1]	DIFF5[1]	SIGN5[1]	DIFF4[1]	SIGN4[1]
bit 15							bit 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
DIFF3[1]	SIGN3[1]	DIFF2[1]	SIGN2[1]	DIFF1[1]	SIGN1[1]	DIFF0[1]	SIGN0[1]
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 (odd) **DIFF<7:0>:** Differential-Mode for Corresponding Analog Inputs bits[1]
1 = Channel is differential
0 = Channel is single-ended

bit (even) **SIGN<7:0>:** Output Data Sign for Corresponding Analog Inputs bits[1]
1 = Channel output data is signed
0 = Channel output data is unsigned

Note 1: The available input channels are device-specific. Some channels may not be implemented. Also, not all channels may support the Differential-mode. Refer to the device data sheet for the available SIGNx and DIFFx bits.

Register 2-20: ADMOD0H: ADC Input Mode Control Register 0 High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DIFF15[1]	SIGN15[1]	DIFF14[1]	SIGN14[1]	DIFF13[1]	SIGN13[1]	DIFF12[1]	SIGN12[1]
bit 15							bit 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
DIFF11[1]	SIGN11[1]	DIFF10[1]	SIGN10[1]	DIFF9[1]	SIGN9[1]	DIFF8[1]	SIGN8[1]
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 (odd) **DIFF<15:8>:** Differential-Mode for Corresponding Analog Inputs bits[1]
1 = Channel is differential
0 = Channel is single-ended

bit (even) **SIGN<15:8>:** Output Data Sign for Corresponding Analog Inputs bits[1]
1 = Channel output data is signed
0 = Channel output data is unsigned

Note 1: The available input channels are device-specific. Some channels may not be implemented. Also, not all channels may support the Differential-mode. Refer to the device data sheet for the available SIGNx and DIFFx bits.

Register 2-21: ADMOD1L: ADC Input Mode Control Register 1 Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DIFF23(1)	SIGN23(1)	DIFF22(1)	SIGN22(1)	DIFF21(1)	SIGN21(1)	DIFF20(1)	SIGN20(1)
bit 15							bit 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
DIFF19(1)	SIGN19(1)	DIFF18(1)	SIGN18(1)	DIFF17(1)	SIGN17(1)	DIFF16(1)	SIGN16(1)
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 (odd) **DIFF<23:16>:** Differential-Mode for Corresponding Analog Inputs bits(1)
1 = Channel is differential
0 = Channel is single-ended

bit (even) **SIGN<23:16>:** Output Data Sign for Corresponding Analog Inputs bits(1)
1 = Channel output data is signed
0 = Channel output data is unsigned

Note 1: The available input channels are device-specific. Some channels may not be implemented. Also, not all channels may support the Differential-mode. Refer to the device data sheet for the available SIGNx and DIFFx bits.

Register 2-22: ADMOD1H: ADC Input Mode Control Register 1 High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DIFF31(1)	SIGN31(1)	DIFF30(1)	SIGN30(1)	DIFF29(1)	SIGN29(1)	DIFF28(1)	SIGN28(1)
bit 15							bit 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
DIFF27(1)	SIGN27(1)	DIFF26(1)	SIGN26(1)	DIFF25(1)	SIGN25(1)	DIFF24(1)	SIGN24(1)
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 (odd) **DIFF<31:24>:** Differential-Mode for Corresponding Analog Inputs bits(1)
1 = Channel is differential
0 = Channel is single-ended

bit (even) **SIGN<31:24>:** Output Data Sign for Corresponding Analog Inputs bits(1)
1 = Channel output data is signed
0 = Channel output data is unsigned

Note 1: The available input channels are device-specific. Some channels may not be implemented. Also, not all channels may support the Differential-mode. Refer to the device data sheet for the available SIGNx and DIFFx bits.

Register 2-23: ADIEL: ADC Interrupt Enable Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IE<15:8>[1]							
bit 15							bit 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
IE<7:0>[1]							
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 15-0 **IE<15:0>:** Interrupt Enable bits[1]
`1` = Individual and common interrupts are enabled for the corresponding channel
`0` = Individual and common interrupts are disabled for the corresponding channel

Note 1: The available channels are device-specific. Some IEx bits may not be implemented. Refer to the device data sheet for the available channels.

Register 2-24: ADIEH: ADC Interrupt Enable Register High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IE<31:24>[1]							
bit 15							bit 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
IE<23:16>[1]							
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 15-0 **IE<31:16>:** Interrupt Enable bits[1]
`1` = Individual and common interrupts are enabled for the corresponding channel
`0` = Individual and common interrupts are disabled for the corresponding channel

Note 1: The available channels are device-specific. Some IEx bits may not be implemented. Refer to the device data sheet for the available channels.

Register 2-25: ADSTATL: ADC Data Ready Status Register Low

R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0
AN15RDY[(1)]	AN14RDY[(1)]	AN13RDY[(1)]	AN12RDY[(1)]	AN11RDY[(1)]	AN10RDY[(1)]	AN9RDY[(1)]	AN8RDY[(1)]
bit 15							bit 8

R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0
AN7RDY[(1)]	AN6RDY[(1)]	AN5RDY[(1)]	AN4RDY[(1)]	AN3RDY[(1)]	AN2RDY[(1)]	AN1RDY[(1)]	AN0RDY[(1)]
bit 7							bit 0

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

bit 15-0 **AN15RDY:AN0RDY:** Data Ready Status for Corresponding Analog Inputs bits[(1)]
1 = Channel conversion result is ready in the corresponding ADCBUFx register
0 = Channel conversion result is not ready

Note 1: The available channels are device-specific. Some ANxRDY bits may not be implemented. Refer to the device data sheet for the available channels.

Register 2-26: ADSTATH: ADC Data Ready Status Register High

R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0
AN31RDY[(1)]	AN30RDY[(1)]	AN29RDY[(1)]	AN28RDY[(1)]	AN27RDY[(1)]	AN26RDY[(1)]	AN25RDY[(1)]	AN24RDY[(1)]
bit 15							bit 8

R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0
AN23RDY[(1)]	AN22RDY[(1)]	AN21RDY[(1)]	AN20RDY[(1)]	AN19RDY[(1)]	AN18RDY[(1)]	AN17RDY[(1)]	AN16RDY[(1)]
bit 7							bit 0

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

bit 15-0 **AN31RDY:AN16RDY:** Data Ready Status for Corresponding Analog Inputs bits[(1)]
1 = Channel conversion result is ready in the corresponding ADCBUFx register
0 = Channel conversion result is not ready

Note 1: The available channels are device-specific. Some ANxRDY bits may not be implemented. Refer to the device data sheet for the available channels.

Register 2-27: ADTRIGnL and ADTRIGnH: ADC Channel Trigger n Selection Registers Low and High (where n is a register number from 0 to 7)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	TRGSRC(x+1)<4:0>[(1)]				
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	TRGSRCx<4:0>[(1)]				
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 15-13 **Unimplemented:** Read as ‘0’

bit 12-8 **TRGSRC(x+1)<4:0>:** Trigger Source Selection for Corresponding Analog Input (x+1) bits[(1)]

11111 = External trigger pin
00100-11110 = Other trigger options specific for the device; refer to the device data sheet for more information
00011 = Reserved
00010 = Common level-sensitive software trigger
00001 = Common software trigger
00000 = No trigger is enabled

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **TRGSRCx<4:0>:** Trigger Source Selection for Corresponding Analog Input x bits[(1)]

11111 = External trigger pin
00100-11110 = Other trigger options specific for the device; refer to the device data sheet for more information
00011 = Reserved
00010 = Common level-sensitive software trigger
00001 = Common software trigger
00000 = No trigger is enabled

Note 1: The available channels are device-specific. Some TRGSRCx<4:0> bits may not be implemented. Refer to the device data sheet for the available channel information.

Register 2-28: ADCAL0L: ADC Calibration Register 0 Low[2]

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CAL1RDY[1]	—	—	—	—	CAL1DIFF[1]	CAL1EN[1]	CAL1RUN[1]
bit 15							bit 8

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CAL0RDY[1]	—	—	—	—	CAL0DIFF[1]	CAL0EN[1]	CAL0RUN[1]
bit 7							bit 0

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

bit 15 **CAL1RDY:** Dedicated ADC Core 1 Calibration Status Flag bit[1]
1 = Dedicated ADC core calibration is finished
0 = Dedicated ADC core calibration is in progress

bit 14-12 **Unimplemented:** Read as ‘0’

bit 11 **Reserved:** Must be written as ‘0’

bit 10 **CAL1DIFF:** Dedicated ADC Core 1 Differential-Mode Calibration bit[1]
1 = Dedicated ADC core will be calibrated in Differential Input mode
0 = Dedicated ADC core will be calibrated in Single-Ended Input mode

bit 9 **CAL1EN:** Dedicated ADC Core 1 Calibration Enable bit[1]
1 = Dedicated ADC core calibration bits (CAL1RDY, CAL1DIFF and CAL1RUN) can be accessed by software
0 = Dedicated ADC core calibration bits are disabled

bit 8 **CAL1RUN:** Dedicated ADC Core 1 Calibration Start bit[1]
1 = If this bit is set by software, the dedicated ADC core calibration cycle is started; this bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

bit 7 **CAL0RDY:** Dedicated ADC Core 0 Calibration Status Flag bit[1]
1 = Dedicated ADC core calibration is finished
0 = Dedicated ADC core calibration is in progress

bit 6-4 **Unimplemented:** Read as ‘0’

bit 3 **Reserved:** Must be written as ‘0’

bit 2 **CAL0DIFF:** Dedicated ADC Core 0 Differential-Mode Calibration bit[1]
1 = Dedicated ADC core will be calibrated in Differential Input mode
0 = Dedicated ADC core will be calibrated in Single-Ended Input mode

bit 1 **CAL0EN:** Dedicated ADC Core 0 Calibration Enable bit[1]
1 = Dedicated ADC core calibration bits (CAL0RDY, CAL0DIFF and CAL0RUN) can be accessed by software
0 = Dedicated ADC core calibration bits are disabled

bit 0 **CAL0RUN:** Dedicated ADC Core 0 Calibration Start Bit[1]
1 = If this bit is set by software, the dedicated ADC core calibration cycle is started; this bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

Note 1: The available dedicated ADC cores number is device-specific. Some CALxRDY, CALxDIFF, CALxEN and CALxRUN bits may not be implemented. Refer to the device data sheet for the available ADC cores.
2: The calibration is not required for some devices. Refer to the specific device data sheet to see if this register is implemented.

Register 2-29: ADCAL0H: ADC Calibration Register 0 High[2]

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CAL3RDY[1]	—	—	—	—	CAL3DIFF[1]	CAL3EN[1]	CAL3RUN[1]
bit 15							bit 8

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CAL2RDY[1]	—	—	—	—	CAL2DIFF[1]	CAL2EN[1]	CAL2RUN[1]
bit 7							bit 0

Legend:	HC = Hardware Clearable bit	HS = Hardware Settable bit	r = Reserved bit
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **CAL3RDY:** Dedicated ADC Core 3 Calibration Status Flag bit[1]
1 = Dedicated ADC core calibration is finished
0 = Dedicated ADC core calibration is in progress

bit 14-12 **Unimplemented:** Read as '0'

bit 11 **Reserved:** Must be written as '0'

bit 10 **CAL3DIFF:** Dedicated ADC Core 3 Differential-Mode Calibration bit[1]
1 = Dedicated ADC core will be calibrated in Differential Input mode
0 = Dedicated ADC core will be calibrated in Single-Ended Input mode

bit 9 **CAL3EN:** Dedicated ADC Core 3 Calibration Enable bit[1]
1 = Dedicated ADC core calibration bits (CAL3RDY, CAL3DIFF and CAL3RUN) can be accessed by software.
0 = Dedicated ADC core calibration bits are disabled

bit 8 **CAL3RUN:** Dedicated ADC Core 3 Calibration Start bit[1]
1 = If this bit is set by software, the dedicated ADC core calibration cycle is started; this bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

bit 7 **CAL2RDY:** Dedicated ADC Core 2 Calibration Status Flag bit[1]
1 = Dedicated ADC core calibration is finished
0 = Dedicated ADC core calibration is in progress

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **Reserved:** Must be written as '0'

bit 2 **CAL2DIFF:** Dedicated ADC Core 2 Differential-Mode Calibration bit[1]
1 = Dedicated ADC core will be calibrated in Differential Input mode
0 = Dedicated ADC core will be calibrated in Single-Ended Input mode

bit 1 **CAL2EN:** Dedicated ADC Core 2 Calibration Enable bit[1]
1 = Dedicated ADC core calibration bits (CAL2RDY, CAL2DIFF and CAL2RUN) can be accessed by software
0 = Dedicated ADC core calibration bits are disabled

bit 0 **CAL2RUN:** Dedicated ADC Core 2 Calibration Start bit[1]
1 = If this bit is set by software, the dedicated ADC core calibration cycle is started; this bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

Note 1: The available dedicated ADC cores number is device-specific. Some CALxRDY, CALxDIFF, CALxEN and CALxRUN bits may not be implemented. Refer to the device data sheet for the available ADC cores.
2: The calibration is not required for some devices. Refer to the specific device data sheet to see if this register is implemented.

Register 2-30: ADCAL1L: ADC Calibration Register 1 Low[2]

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CAL5RDY[1]	—	—	—	—	CAL5DIFF[1]	CAL5EN[1]	CAL5RUN[1]
bit 15							bit 8

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CAL4RDY[1]	—	—	—	—	CAL4DIFF[1]	CAL4EN[1]	CAL4RUN[1]
bit 7							bit 0

Legend:	HC = Hardware Clearable bit	HS = Hardware Settable bit	r = Reserved bit
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **CAL5RDY:** Dedicated ADC Core 5 Calibration Status Flag bit[1]
1 = Dedicated ADC core calibration is finished
0 = Dedicated ADC core calibration is in progress

bit 14-12 **Unimplemented:** Read as '0'

bit 11 **Reserved:** Must be written as '0'

bit 10 **CAL5DIFF:** Dedicated ADC Core 5 Differential-Mode Calibration bit[1]
1 = Dedicated ADC core will be calibrated in differential input mode
0 = Dedicated ADC core will be calibrated in single-ended input mode

bit 9 **CAL5EN:** Dedicated ADC Core 5 Calibration Enable bit[1]
1 = Dedicated ADC core calibration bits (CAL5RDY, CAL5DIFF and CAL5RUN) can be accessed by software
0 = Dedicated ADC core calibration bits are disabled

bit 8 **CAL5RUN:** Dedicated ADC Core 5 Calibration Start bit[1]
1 = If this bit is set by software, the dedicated ADC core calibration cycle is started; this bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

bit 7 **CAL4RDY:** Dedicated ADC Core 4 Calibration Status Flag bit[1]
1 = Dedicated ADC core calibration is finished
0 = Dedicated ADC core calibration is in progress

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **Reserved:** Must be written as '0'

bit 2 **CAL4DIFF:** Dedicated ADC Core 4 Differential-Mode Calibration bit[1]
1 = Dedicated ADC core will be calibrated in Differential Input mode
0 = Dedicated ADC core will be calibrated in Single-Ended Input mode

bit 1 **CAL4EN:** Dedicated ADC Core 4 Calibration Enable bit[1]
1 = Dedicated ADC core calibration bits (CAL4RDY, CAL4DIFF and CAL4RUN) can be accessed by software
0 = Dedicated ADC core calibration bits are disabled

bit 0 **CAL4RUN:** Dedicated ADC Core 4 Calibration Start bit[1]
1 = If this bit is set by software, the dedicated ADC core calibration cycle is started. This bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

Note 1: The available dedicated ADC cores number is device-specific. Some CALxRDY, CALxDIFF, CALxEN and CALxRUN bits may not be implemented. Refer to the device data sheet for the available ADC cores.

2: The calibration is not required for some devices. Refer to the specific device data sheet to see if this register is implemented.

Register 2-31: ADCAL1H: ADC Calibration Register 1 High[2]

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CSHRRDY	—	—	—	—	CSHRDIFF	CSHREN	CSHRRUN
bit 15							bit 8

R/HC/HS-0	U-0	U-0	U-0	r-0	R/W-0	R/W-0	R/W-0
CAL6RDY[1]	—	—	—	—	CAL6DIFF[1]	CAL6EN[1]	CAL6RUN[1]
bit 7							bit 0

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

bit 15 **CSHRRDY:** Shared ADC Core Calibration Status Flag bit
1 = Shared ADC core calibration is finished
0 = Shared ADC core calibration is in progress

bit 14-12 **Unimplemented:** Read as ‘0’

bit 11 **Reserved:** Must be written as ‘0’

bit 10 **CSHRDIFF:** Shared ADC Core Differential-Mode Calibration bit
1 = Shared ADC core will be calibrated in Differential Input mode
0 = Shared ADC core will be calibrated in Single-Ended Input mode

bit 9 **CSHREN:** Shared ADC Core Calibration Enable bit
1 = Shared ADC core calibration bits (CSHRRDY, CSHRDIFF and CSHRRUN) can be accessed by software
0 = Shared ADC core calibration bits are disabled

bit 8 **CSHRRUN:** Shared ADC Core Calibration Start bit
1 = If this bit is set by software, the shared ADC core calibration cycle is started; this bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

bit 7 **CAL6RDY:** Dedicated ADC Core 6 Calibration Status Flag bit[1]
1 = Dedicated ADC core calibration is finished
0 = Dedicated ADC core calibration is in progress

bit 6-4 **Unimplemented:** Read as ‘0’

bit 3 **Reserved:** Must be written as ‘0’

bit 2 **CAL6DIFF:** Dedicated ADC Core 6 Differential-Mode Calibration bit[1]
1 = Dedicated ADC core will be calibrated in Differential Input mode
0 = Dedicated ADC core will be calibrated in Single-Ended Input mode

bit 1 **CAL6EN:** Dedicated ADC Core 6 Calibration Enable bit[1]
1 = Dedicated ADC core calibration bits (CAL6RDY, CAL6DIFF and CAL6RUN) can be accessed by software
0 = Dedicated ADC core calibration bits are disabled

bit 0 **CAL6RUN:** Dedicated ADC Core 6 Calibration Start bit[1]
1 = If this bit is set by software, the dedicated ADC core calibration cycle is started; this bit is cleared by hardware automatically
0 = Software can start the next calibration cycle

Note 1: The available dedicated ADC cores number is device-specific. Some CALxRDY, CALxDIFF, CALxEN and CALxRUN bits may not be implemented. Refer to the device data sheet for the available ADC cores.

2: The calibration is not required for some devices. Refer to the specific device data sheet to see if this register is implemented.

Register 2-32: ADCMPnCON: ADC Digital Comparator n Control Register[(1)]

U-0	U-0	U-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0	R/HC/HS-0
—	—	—	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0
bit 15							bit 8

R/W-0	R/W-0	R/HC/HS-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMPEN	IE	STAT	BTWN	HIHI	HILO	LOHI	LOLO
bit 7							bit 0

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

bit 15-13 **Unimplemented:** Read as ‘0’

bit 12-8 **CHNL<4:0>:** Input Channel Number bits
These bits identify the analog input that caused the comparator event.
11111 = AN31
•
•
•
00001 = AN1
00000 = AN0

bit 7 **CMPEN:** Digital Comparator Enable bit
1 = Comparator is enabled
0 = Comparator is disabled and the STAT status bit is cleared

bit 6 **IE:** Comparator Interrupts Enable bit
1 = individual and common interrupts will be generated if the comparator detects a comparison event
0 = Individual and common interrupts will not be generated for the comparator

bit 5 **STAT:** Comparator Event Status bit
This bit is cleared by hardware when the channel number is read from the CHNL<4:0> bits.
1 = A comparison event has been detected since the last read of the CHNL<4:0> bits
0 = A comparison event has not been detected since the last read of the CHNL<4:0> bits

bit 4 **BTWN:** Between Low/High Comparator Event bit
1 = Generates a comparator event when ADCMPnLO ≤ ADCBUFx < ADCMPnHI
0 = Does not generate a digital comparator event when ADCMPnLO ≤ ADCBUFx < ADCMPnHI

bit 3 **HIHI:** High/High Comparator Event bit
1 = Generates a digital comparator event when ADCBUFx ≥ ADCMPnHI
0 = Does not generate a digital comparator event when ADCBUFx ≥ ADCMPnHI

bit 2 **HILO:** High/Low Comparator Event bit
1 = Generates a digital comparator event when ADCBUFx < ADCMPnHI
0 = Does not generate a digital comparator event when ADCBUFx < ADCMPnHI

bit 1 **LOHI:** Low/High Comparator Event bit
1 = Generates a digital comparator event when ADCBUFx ≥ ADCMPnLO
0 = Does not generate a digital comparator event when ADCBUFx ≥ ADCMPnLO

bit 0 **LOLO:** Low/Low Comparator Event bit
1 = Generates a digital comparator event when ADCBUFx < ADCMPnLO
0 = Does not generate a digital comparator event when ADCBUFx < ADCMPnLO

Note 1: The available digital comparators number is device-specific. Refer to the device data sheet for the available digital comparators.

Register 2-33: ADCMPnENL: ADC Digital Comparator n Channel Enable Register Low(1)

R/W/0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMPEN<15:8>(2)							
bit 15							bit 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
CMPEN<7:0>(2)							
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 15-0 **CMPEN<15:0>:** Comparator Enable for Corresponding Input Channels bits(2)
`1` = Conversion result for corresponding channel is used by the comparator
`0` = Conversion result for corresponding channel is not used by the comparator

Note 1: The available digital comparators number is device-specific. Refer to the device data sheet for the available digital comparators.
2: The available channels are device-specific. Some CMPENx bits may not be implemented. Refer to the device data sheet for the available channels.

Register 2-34: ADCMPnENH: ADC Digital Comparator n Channel Enable Register High(1)

R/W/0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CMPEN<31:24>(2)							
bit 15							bit 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
CMPEN<23:16>(2)							
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 15-0 **CMPEN<31:16>:** Comparator Enable for Corresponding Input Channels bits(2)
`1` = Conversion result for corresponding channel is used by the comparator
`0` = Conversion result for corresponding channel is not used by the comparator

Note 1: The available digital comparators number is device-specific. Refer to the device data sheet for the available digital comparators.
2: The available channels are device-specific. Some CMPENx bits may not be implemented. Refer to the device data sheet for the available channels.

Register 2-35: ADFLnCON: ADC Digital Filter n Control Register[1]

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/HC/HS-0
FLEN	MODE1	MODE0	OVRSAM2	OVRSAM1	OVRSAM0	IE	RDY
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLCHSEL4	FLCHSEL3	FLCHSEL2	FLCHSEL1	FLCHSEL0
bit 7							bit 0

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

bit 15 **FLEN:** Filter Enable bit
1 = Filter is enabled
0 = Filter is disabled and the RDY bit is cleared

bit 14-13 **MODE<1:0>:** Filter Mode bits
11 = Averaging mode
10 = Reserved
01 = Reserved
00 = Oversampling mode

bit 12-10 **OVRSAM<2:0>:** Filter Averaging/Oversampling Ratio bits
If MODE<1:0> = 00:
111 = 128x (16-bit result in the ADFLnDAT register)
110 = 32x (15-bit result in the ADFLnDAT register)
101 = 8x (14-bit result in the ADFLnDAT register)
100 = 2x (13-bit result in the ADFLnDAT register)
011 = 256x (16-bit result in the ADFLnDAT register)
010 = 64x (15-bit result in the ADFLnDAT register)
001 = 16x (14-bit result in the ADFLnDAT register)
000 = 4x (13-bit result in the ADFLnDAT register)
If MODE<1:0> = 11 (12-bit result in the ADFLnDAT register):
111 = 256x
110 = 128x
101 = 64x
100 = 32x
011 = 16x
010 = 8x
001 = 4x
000 = 2x

bit 9 **IE:** Filter Interrupts Enable bit
1 = Individual and common interrupts will be generated when the filter result is ready
0 = Individual and common interrupts will not be generated for the filter

bit 8 **RDY:** Oversampling Filter Data Ready Flag bit
This bit is cleared by hardware when the result is read from the ADFLnDAT register.
1 = Data in the ADFLnDAT register is ready
0 = The ADFLnDAT register has been read and new data in the ADFLnDAT register is not ready

bit 7-5 **Unimplemented:** Read as ‘0’

Note 1: The available oversampling filter number is device-specific. Refer to the device data sheet for the available oversampling filters.

Register 2-35: ADFLnCON: ADC Digital Filter n Control Register[1] (Continued)

bit 4-0 **FLCHSEL<4:0>:** Oversampling Filter Input Channel Selection bits

`11111` = AN31
•
•
•
`00001` = AN1
`00000` = AN0

Note 1: The available oversampling filter number is device-specific. Refer to the device data sheet for the available oversampling filters.

Register 2-36: ADCSSL: CVD Scan Select Register Low[1]

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>[2]							
bit 15							bit 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<7:0>[2]							
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 15-0 **CSS<15:0>:** CVD Scan Enable for Corresponding Analog Inputs bits[2]
`1` = The shared core analog input is included in the CVD scan
`0` = The shared core analog input is not scanned by CVD

Note 1: This register is not available if the CVD feature is not implemented. Refer to the device data sheet to see if CVD is available.

2: The available channels for the ADC shared core are device-specific. Some CSSx bits may not be implemented. Refer to the device data sheet for the available channels information.

Register 2-37: ADCSSH: CVD Scan Select Register High[1]

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>[2]							
bit 15							bit 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<23:16>[2]							
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 15-0 **CSS<31:16>:** CVD Scan Enable for Corresponding Analog Inputs bits[2]
`1` = The shared core analog input is included in the CVD scan
`0` = The shared core analog input is not scanned by CVD

Note 1: This register is not available if the CVD feature is not implemented. Refer to the device data sheet to see if CVD is available.

2: The available channels for the ADC shared core are device-specific. Some CSSx bits may not be implemented. Refer to the device data sheet for the available channels information.

3.0 CONVERSION SEQUENCE

Analog-to-Digital conversion using the 12-Bit High-Speed, Multiple SARs ADC involves the following three steps:

1. Sampling of the input signal.
2. Capture of the input signal (holding) and transfer to the converter.
3. Conversion of the analog signal to its digital representation.

Sampling of the input signal involves charging of the capacitor in the Sample-and-Hold (S/H) circuit. The sampling time must be adequate so that the capacitor charges to a value equal to the input voltage. At the appropriate time, the input is disconnected from the capacitor, and subsequently, the analog voltage is transferred to the converter. The converter then digitizes the analog signal and provides the result.

The converter requires a clock source and a reference voltage. The clock and reference voltage sources are selectable, as well as the clock prescaling.

4.0 ADC OPERATION

4.1 SAR ADC Cores

The number of dedicated SAR ADC cores available is device-specific. For more information, refer to the specific device data sheet.

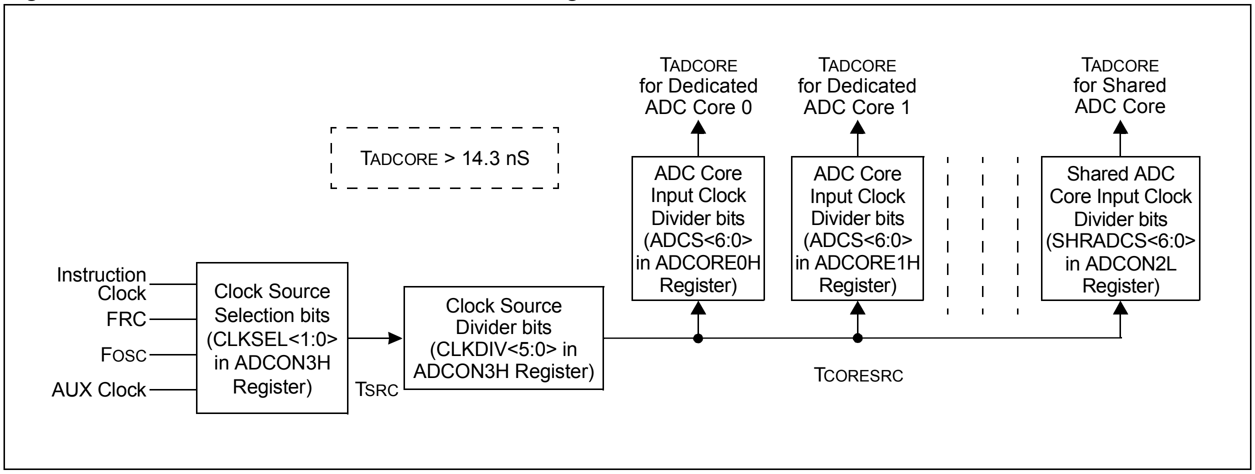
The module may implement up to eight independent SAR ADC cores. It allows sampling signals simultaneously from multiple analog inputs. The seven first SAR ADC cores (0 through 6) are referred to as dedicated, since each has a single dedicated analog channel. Each channel is connected and the ADC core samples (follows) the input signal voltage continuously. The channel for the dedicated ADC core is disconnected only when the conversion is started. The dedicated independent ADC cores allow an application to sample the associated analog channels simultaneously and convert them in a single “snap shot”.

The last SAR ADC core is referred to as a shared core, as it is shared among the analog inputs that are not associated with the dedicated ADC cores. For this core, the analog channel to be sampled, and the sampling process, are controlled by the ADC module. When the conversion is not in progress, all inputs are disconnected from the shared ADC core. By the trigger event, the ADC connects the analog input defined in the trigger and samples the input signal during the specified amount of time. After sampling is completed, the analog input is disconnected again and the conversion is performed.

4.2 ADC Clock

The ADC module has different options for the clock source. The source can be selected using the CLKSEL<1:0> bits in the ADCON3H register. The selected source has a period, TSRC, and is divided by the ratio specified by the CLKDIV<5:0> bits in the ADCON3H register. After this divider, the result clock with the period, TCORESRC, goes to each SAR ADC core. Each ADC core has its own clock divider that is configured with the ADCS<6:0> bits in the corresponding ADCOREnH register for the dedicated core, and with the SHRADCS<6:0> bits in the ADCON2L register for the shared ADC core. After the divisions, each ADC core can have a different clock period, TADCORE. The maximum operation clock frequency for each SAR ADC core is limited by 70 MHz. Thus, the clock settings must be selected to provide a Core Clock period, TADCORE, more than 14.3 nS. The module clock path diagram is shown in Figure 4-1.

Figure 4-1: ADC Module Clock Path Block Diagram



4.3 ADC Resolution

Each SAR ADC core resolution can be set individually using the RES<1:0> bits in the ADCOREnH register for the corresponding dedicated core, and using the SHRRES<1:0> bits in the ADCON1H register for the shared core.

Depending on the setting, the resolution of the ADC is either 12 bits, 10 bits, 8 bits or 6 bits. By default after Reset, all cores are configured for 12-bit resolution.

4.4 Sampling and Conversion Timing

The conversion time for all ADC cores depends on the resolution selected by the RES<1:0> or SHRRES<1:0> bits. The time required for the conversion is defined by Equation 4-1.

Equation 4-1: Conversion Time

$$Conversion\ Time = 8 \cdot T_{CORESRC} + (Bit\ Resolution + 2.5) \cdot T_{ADCORE}$$

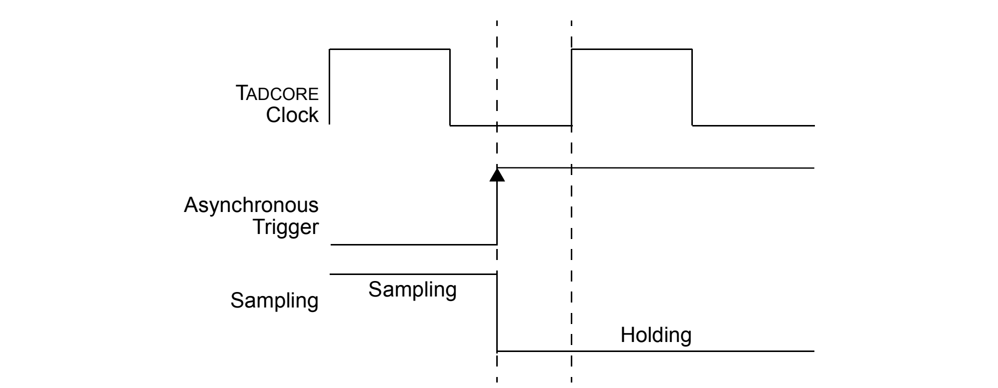
For example, if an ADC core is configured for 12-bit resolution, the conversion time for this core will be: $8 \cdot T_{CORESRC} + 14.5 \cdot T_{ADCORE}$.

If a few ADC cores have the conversion results at the same time, the core with the low priority will wait an additional TCORESRC cycle for each high-priority core to store its result.

Additionally, the conversion time can be extended even further if the Noise Reduction feature is enabled. If the NRE bit in the ADCON1L register is set, the end of conversion time is adjusted to reduce the noise between ADC cores. Depending on the number of cores converting and the priority of the input, a few additional TADs may be inserted, making the conversion time slightly less deterministic. The Noise Reduction feature is not available on all devices and the NRE bit may not be implemented. Refer to the device data sheet for more information.

With multiple dedicated SAR ADC cores, several analog signals can be captured simultaneously. Each dedicated core continuously tracks the input signal in Sample mode until an asynchronous trigger event occurs. The trigger event causes the dedicated core to immediately stop sampling and enter the holding state. It is important to note that, by default, the trigger event which ends sampling occurs asynchronously to the ADC core clock. While the S/H circuits enter the hold state immediately, the asynchronous trigger must be synchronized to the ADC clock, consuming up to one ADC clock edge before the conversion request is issued to the SAR (Figure 4-2).

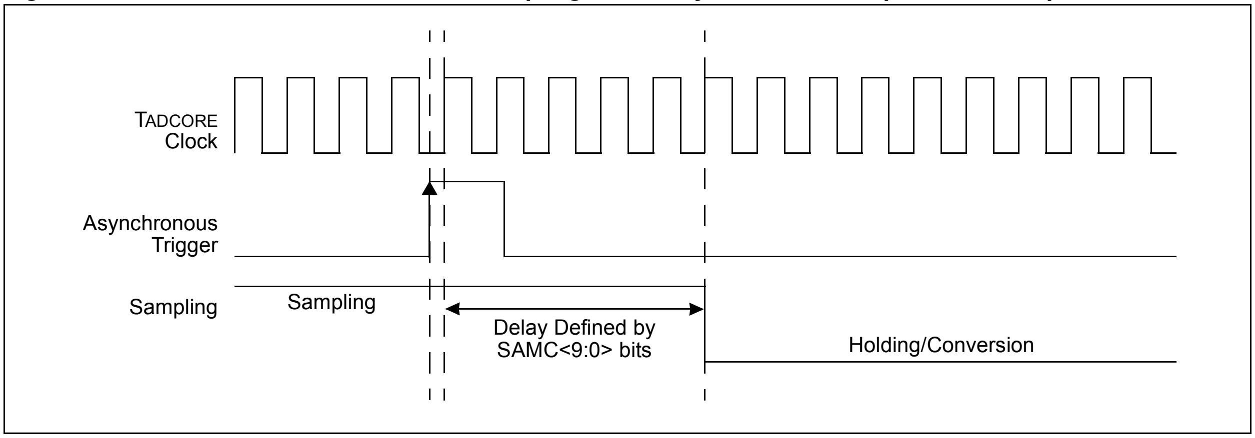
Figure 4-2: Dedicated SAR ADC Core Sampling



If using a periodic trigger source with a dedicated core, the total sampling time is determined by the trigger rate. The trigger rate must not violate the necessary sampling time. See **Section 4.4.1 "Sampling Time Requirements"** for more information.

Another dedicated SAR ADC core sampling option is a delayed conversion. This feature is controlled by the SAMCxEN bits in the ADCON4L register. When SAMCxEN = 1, the delay is inserted between the trigger event and the conversion start (Figure 4-3). During this time, the core continues to sample the input signal. The delay time is defined by the SAMC<9:0> bits in the corresponding Dedicated ADC Core n Control Register Low, ADCOREnL. This delay should be used with the digital filters or level-sensitive triggers to ensure the minimum sampling time. Regardless of the SAMCxEN bit, the SAMC<9:0> bits in the ADCOREnL register limit the time between triggers. If the time between triggers will be less than the time specified in the SAMC<9:0> bits, then the trigger will be delayed.

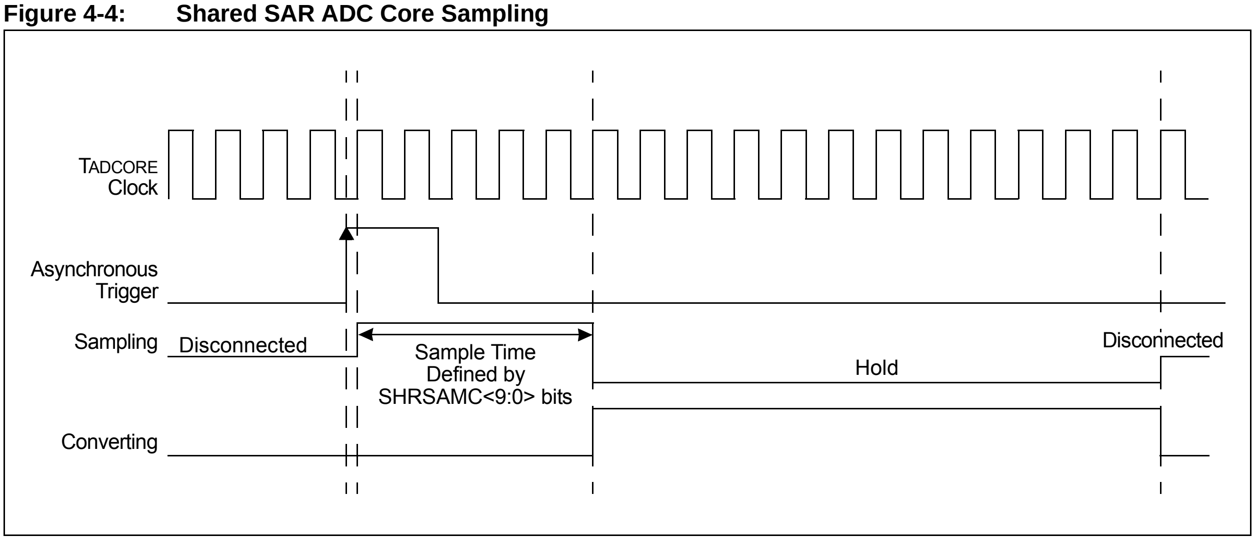
Figure 4-3: Dedicated SAR ADC Core Sampling with Delayed Conversion (SAMCxEN = 1)



Unlike the dedicated ADC core, the trigger event of a shared ADC core starts the sampling process using a sample time specified by the SHRSAMC<9:0> bits in the ADCON2H register. Once the signal has sampled the specified number of ADC Core Clocks (TADCORE), the S/H enters the hold state and the conversion request is issued (shown in Figure 4-4).

When periodically triggering a single input of the shared ADC core, the trigger rate should not be more than the sample time plus the conversion time. There is no assurance that a conversion request for the shared ADC core will be serviced immediately. Conversion requests for the shared core are serviced in the order of priority.

Figure 4-4: Shared SAR ADC Core Sampling

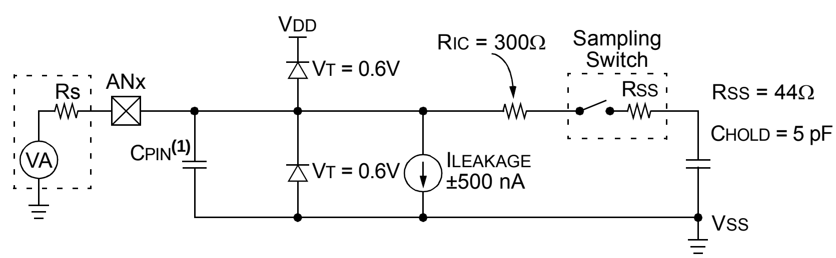


4.4.1 SAMPLING TIME REQUIREMENTS

The analog input model of the 12-Bit High-Speed, Multiple SARs ADC is illustrated in Figure 4-5. The total acquisition time for the Analog-to-Digital conversion is a function of the Holding Capacitor (CHOLD) charge time.

For the ADC module to meet its specified accuracy, the Holding Capacitor (CHOLD) must be allowed to fully charge to the voltage level on the analog input pin. The analog output Source Impedance (RS), the Interconnect Impedance (RIC) and the internal Sampling Switch Impedance (RSS) 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 selected sample time. The internal Holding Capacitor will be in the discharged state prior to each sample operation.

Figure 4-5: 12-Bit ADC Analog Input Model



Legend:

CPIN	= Input Capacitance	VT	= Threshold Voltage
RSS	= Sampling Switch Resistance	RIC	= Interconnect Resistance
RS	= Source Resistance	CHOLD	= Sample/Hold Capacitance
ILEAKAGE	= Leakage Current at the Pin due to Various Junctions		

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.

4.5 Voltage Reference

The voltage reference options vary by device. Refer to the device data sheet for the specific options.

The ADC Reference Voltage Selection bits, REFSEL<2:0> in the ADCON3L register, select the voltage reference for all SAR ADC cores. Also, the ADC module depends on the internal band gap circuit voltage. When the voltage reference and the band gap are ready for operation, the REFRDY bit in ADCON2H is set. The voltage reference source cannot be changed when the module is enabled (ADON (ADCON1L<15>) = 1). If the ADC reference or AVDD power was changed or interrupted, the REFERR bit in the ADCON2H register is set. After a voltage reference Fault event is detected, the ADC module must be recalibrated. The voltage reference ready and voltage reference Fault events, indicated by the REFRDY and REFERR bits, can generate common interrupts if the corresponding REFCIE and REFERCIE bits are set in the ADCON2L register. To avoid false Fault interrupts, the REFERCIE bit must be set only after the module is enabled (ADON = 1).

4.6 Analog Input Channels

The number of input channels available is device-specific. For more information, refer to the specific device data sheet.

4.6.1 CONFIGURING ANALOG PORT PINS

The ANSELx registers for the I/O ports, associated with the analog inputs, are used to configure the corresponding pins as analog pins. A pin is configured as an analog input when the corresponding ANSELx bit = `1`. When the ANSELx bit = `0`, the pin is set to digital control. When configured for analog input, the associated port I/O digital input buffer is disabled so that it does not consume current. The ANSELx registers are set when the device comes out of Reset, causing the ADC input pins to be configured as analog inputs by default.

The TRISx registers control the digital function of the port pins. The port pin that is required as an analog input must have its corresponding bit set in the specific TRISx register, configuring the pin as an input. If the I/O pin associated with an ADC input is configured as an output by clearing the TRISx bit, the port's digital output level will be converted. After a device Reset, all of the TRISx bits are set. For more information on port pin configuration, refer to the **"I/O Ports"** chapter of the specific device data sheet.

> **Note:** The PORT register bit reads as '`0`' if its corresponding pin is configured as an analog input.

4.6.2 SINGLE-ENDED AND PSEUDODIFFERENTIAL INPUT OPTIONS AND OUTPUT RESULT FORMAT

The A/D Converter comprises both single-ended and pseudodifferential channels. The input voltage on any analog pin should not be less than the analog ground level, AV_{SS}, and should not exceed the analog power voltage, AV_{DD} (for either Single-Ended mode or Pseudodifferential mode). The pseudodifferential channel has inverting and non-inverting inputs. The analog pin used in Pseudo-differential mode for the inverting input is fixed for each ADC core. Refer to the device-specific data sheet for the inverting input number for the particular ADC core. For the correct operation in the Pseudodifferential mode, one input (inverting or non-inverting) is allowed to swing from V_{R-} to V_{R+}, while another input is restricted to around $(V_{R+} + V_{R-})/2 \pm 150$ mV (where V_{R+} and V_{R-} are positive and negative reference voltages).

The Single-Ended mode or Pseudodifferential mode for each ADC input channel is defined by the DIFFx bit in the ADMODnL or ADMODnH registers. If the corresponding DIFFx bit is set, the channel is differential. When this DIFFx bit is zero, the channel is single-ended. Also, the data output format can be set individually for each channel using the SIGNx bit in the ADMODnL or ADMODnH registers. If the SIGNx bit is set, the conversion result for the input is written as a signed value into the corresponding result buffer, ADCBUFx. If the SIGNx bit is cleared, the result of the conversion is unsigned. Table 4-1 details the input configuration options.

Table 4-1: Input Configuration

Input		Input Voltage		Output Code (for FORM bit (ADCON1H<7>) = `0`)
Pseudodifferential Mode Bit (DIFFx)	Signed Result Bit (SIGNx)	V_{INP} = Voltage on Non-Inverting Input, V_{INN} = Voltage on Inverting Input, V_{R+} = Positive Reference Voltage, V_{R-} = Negative Reference Voltage		
`1`	`1`	Minimum	$V_{INP} \le V_{R-}$; $V_{INN} = (V_{R+} + V_{R-})/2$	-1024
		Maximum	$V_{INP} \ge V_{R+}$; $V_{INN} = (V_{R+} + V_{R-})/2$	+1023
`1`	`0`	Minimum	$V_{INP} \le V_{R-}$; $V_{INN} = (V_{R+} + V_{R-})/2$	+1024
		Maximum	$V_{INP} \ge V_{R+}$; $V_{INN} = (V_{R+} + V_{R-})/2$	+3071
`0`	`1`	Minimum	$V_{INP} \le V_{R-}$	-2048
		Maximum	$V_{INP} \ge V_{R+}$	+2047
`0`	`0`	Minimum	$V_{INP} \le V_{R-}$	0
		Maximum	$V_{INP} \ge V_{R+}$	+4095

4.6.3 SELECTING ANALOG INPUT FOR DEDICATED ADC CORE

To provide greater flexibility to the dedicated ADC cores, several inputs are provided for each. These may include the analog pins and the outputs of other analog modules, such as amplifiers. The input options are chosen using the CxCHS<1:0> bits in the ADCON4H register.

The input options are device-specific. For more information, refer to the device data sheet. The CxCHS<1:0> bits physically connect different analog inputs to the dedicated ADC core, but they do not change the trigger source. The dedicated ADC core accepts only triggers from the channel corresponding to the CxCHS<1:0> = 00 option, regardless of the value written into the CxCHS<1:0> bits.

> **Note:** The CxCHS<1:0> bits do not change the trigger source channel for the dedicated core. The trigger source is defined by the default channel assigned to the core. Usually, for dedicated Core 0, the trigger is always selected by the TRGSRC0<4:0> bits; for dedicated Core 1, it is always selected by the TRGSCR1<4:0> bits and so on. Refer to the device data sheet for the details of the trigger options. Also, regardless of the CxCHS<1:0> bits value, the conversion result for dedicated cores is stored in ADCBUF0 for Core 0, in ADCBUF1 for Core 1 and so on.

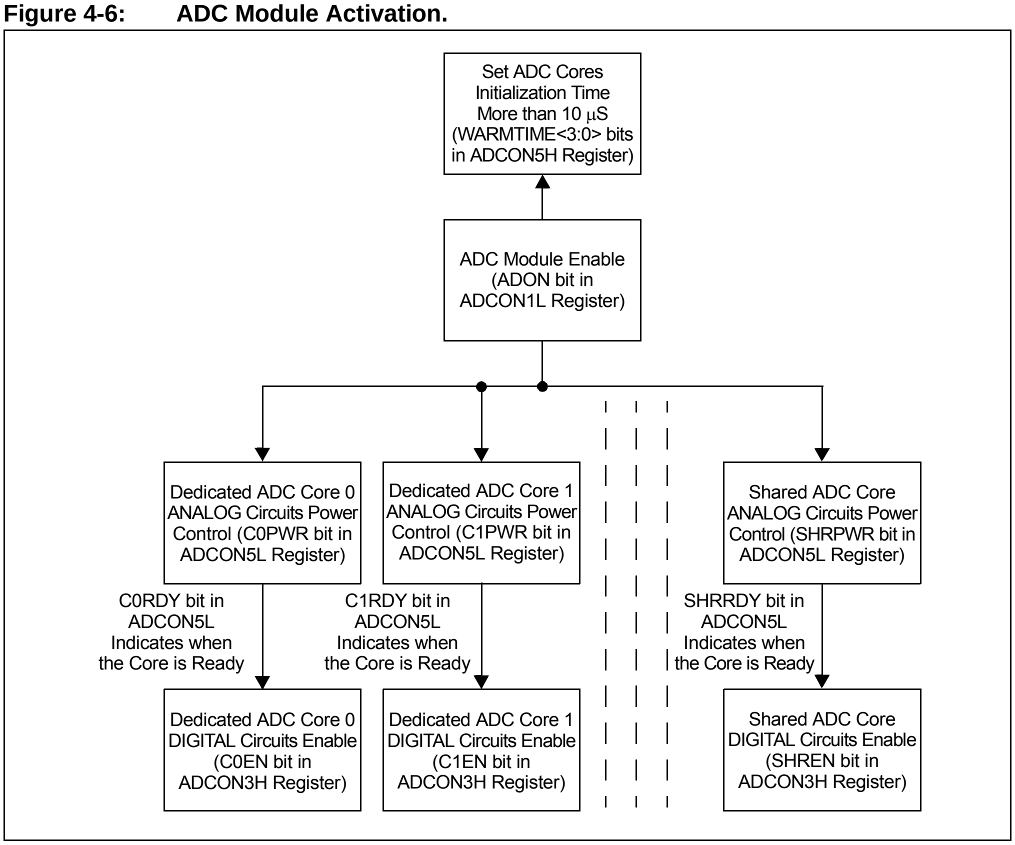
4.6.4 INPUT PRIORITY

To resolve simultaneous requests for input conversions between different channels, the shared ADC core uses a natural order priority scheme. The priority scheme is fixed and is defined by the input channel number, with the channel with lowest number receiving the highest priority. In other words, input conversions occur in ascending order of the analog channel number, with the lowest channel number being converted first.

4.7 Enabling the ADC

The ADC module has several levels of activation (see Figure 4-6).

Figure 4-6: ADC Module Activation.



The ADON bit in the ADCON1L register enables the ADC module as a whole. Then, each SAR ADC core should be switched on individually. The CxPWR and SHRPWR bits in the ADCON5L register control the analog circuits of the dedicated and shared ADC cores accordingly. When the CxPWR or SHRPWR bits are set, the power-on delay is inserted to stabilize the analog circuits. The delay value is defined by the WARMTIME<3:0> bits in the ADCON5H register. After the delay, the corresponding CxRDY or SHRRDY bit is set in the ADCON5L register to indicate that the ADC core is ready for operation. At this point, the ADC cores do not receive the triggers. To enable the ADC core digital circuit, the CxEN or SHREN bit must be set in the ADCON3H register for the dedicated and shared ADC cores accordingly. When the CxEN or SHREN bits are set, the corresponding SAR ADC core is fully operational. The SAR ADC cores require 10 μS initialization time. The WARMTIME<3:0> bits must be configured to provide enough time for the initialization.

To enable the ADC module, the following steps should be performed:

1. Set the WARMTIME<3:0> bits in the ADCON5H register to provide at least 10 μS for the ADC cores initialization.
2. Set the ADON bit in the ADCON1L register.
3. Set the CxPWR and/or SHRPWR bits in the ADCON5L register for the selected ADC cores.
4. Poll the CxRDY and/or SHRRDY bits in the ADCON5L register for the selected ADC cores until they are set.
5. Set the CxEN and/or SHREN bits in the ADCON3H register for the selected ADC cores.

The analog circuit ready events, indicated by the CxRDY and SHRRDY bits, can be used to generate the common ADC interrupt. To do this, the corresponding CxCIE and/or SHRCIE bits must be set in the ADCON5H register.

4.8 Calibration

On some devices, whenever the module is enabled (refer to "**Section 4.7 "Enabling the ADC"**"), the calibration of all SAR ADC cores must be performed. Refer to the specific device data sheet to see if the calibration is required and the Calibration registers, ADCALnL and ADCALnH, are implemented. The ADCALnL and ADCALnH registers control the calibration process for all cores. Two calibration procedures must be done for each core. One procedure is used for Single-Ended Input mode when the CALxDIFF bits for the dedicated cores, and/or the CSHRDIFF bit for the shared core, are cleared. The second procedure is used for the Differential-mode when the CALxDIFF and/or CSHRDIFF bits are set. The differential calibration can be skipped if only Single-Ended mode is used for the core. Also, the single-ended calibration is not required if the core uses differential inputs only. To enter into the Calibration mode, the CALxEN bits and/or CSHREN bit (for dedicated cores and shared core accordingly) should be set. Then, the CALxRUN and/or CSHRRUN bits should be set to execute the calibration. These bits are cleared by hardware to allow the next calibration cycle. Also, when the CALxRUN and/or CSHRRUN bits are set, the corresponding CALxRDY and/or CSHRRDY bits are cleared. The application should poll the CALxRDY and CSHRRDY bits to detect the end of the calibration process. After calibration, the CALxEN and/or CSHREN bits must be cleared in order to resume normal operation on all SAR ADC cores.

To perform the SAR ADC cores' calibration, the following steps should be done:

1. Set the CALxEN and/or CSHREN bits.
2. Clear the CALxDIFF and/or CSHRDIFF bits.
3. Set the CALxRUN and/or CSHRRUN bits.
4. Poll the CALxRDY and/or CSHRRDY bits until they are set.
5. Set the CALxDIFF and/or CSHRDIFF bits.
6. Set the CALxRUN and/or CSHRRUN bits.
7. Poll the CALxRDY and CSHRRDY bits until they are set.
8. Clear the CALxEN and/or CSHREN bits.

After the ADC module is enabled, the recalibration is required if the following ADC options are changed:

- Voltage Reference (REFSEL<2:0> bits)
- Clock (CLKSEL<1:0>, CLKDIV<5:0>, SHRADCS<6:0> and ADCS<6:0> bits)

4.9 Triggering

There are four methods to initiate a conversion request:

- Individual Input Triggers
- Common Software Trigger
- Common Level-Sensitive Software Trigger
- Individual Input Software Trigger (one-shot trigger)

4.9.1 INDIVIDUAL INPUT TRIGGER

The application can independently specify an individual conversion trigger source for each analog input using the TRGSRCn<4:0> bits in the ADTRIGnL or ADTRIGnH register. Typical trigger sources may include general purpose timers, output compare modules, PWM generators, comparators, external pins, common software triggers and the common level-sensitive software trigger. Each trigger source, selected by the TRGSRCn<4:0> bits, can be set as edge-sensitive or level-sensitive using the LVLENx bit in the ADLVLTRGL or ADLVLTRGH register for the corresponding channel. If the LVLENx bit is set, the ADC core will be triggered continuously as long as the trigger signal is asserted. If the LVLENx bit is cleared, the ADC core will be triggered just once for the transition of the trigger signal. When the level-sensitive triggering is used for the dedicated SAR ADC core, the corresponding SAMCxEN bit in the ADCON4L register should be set to give enough sampling time between conversions. The sampling time is defined by the corresponding SAMC<9:0> bits in the ADCOREnL register.

4.9.2 CHANNEL SCAN

If it is necessary to scan several analog channels, this can be done by selecting the same trigger source for each channel using the TRGSRCn<4:0> bits in the ADTRIGnL or ADTRIGnH register.

4.9.3 COMMON SOFTWARE TRIGGER

The conversion of any analog input can be triggered by the Software Common Trigger bit, SWCTRG (ADCON3L<6>). This option is selected when the corresponding channel bits, TRGSRCn<4:0>, are set to '00001' in the ADTRIGnL or ADTRIGnH register. The associated analog inputs will be triggered when the SWCTRG bit is set by the software. This bit is automatically cleared by hardware, allowing the software to trigger another conversion if needed.

4.9.4 COMMON LEVEL-SENSITIVE SOFTWARE TRIGGER

The conversion of any analog input can be triggered by the Software Level-Sensitive Common Trigger bit, SWLCTRG (ADCON3L<7>). To select this option, the TRGSRCn<4:0> bits must be set to '00010' in the ADTRIGnL or ADTRIGnH register for the corresponding input channel. The LVLENx bit must be set in the ADLVLTRGL or ADLVLTRGH register as well. When the SWLCTRG bit is set, the associated analog input will be triggered continuously until the SWLCTRG bit is cleared by the software. When the level-sensitive triggering is used for the dedicated SAR ADC core, the corresponding SAMCxEN bit in the ADCON4L register should be set to give enough sampling time between the conversions. The sampling time is defined by the corresponding SAMC<9:0> bits in the ADCOREnL register.

4.9.5 INDIVIDUAL CHANNEL SOFTWARE TRIGGER

The application can explicitly request a single conversion of any selected analog input, at any time during program execution, without changing the trigger source configuration of the ADC. The input to be converted should be specified by the CNVCHSEL<5:0> bits in the ADCON3L register. The CNVRTCH bit in the ADCON3L register is used to trigger the conversion. This bit is automatically cleared by hardware, allowing the application to trigger another conversion if needed.

4.9.6 TRIGGER SUSPENSION

The ADC module has a special SUSPEND bit in the ADCON3L register to suspend all triggers for all ADC cores. When the SUSPEND bit is set, all future triggers for all SAR ADC cores will be disabled. However, this bit does not cancel the ‘delayed by priority’ triggers and the triggers that are currently in progress. After the triggers are suspended, the software should poll the SUSPRDY bit in the ADCON3L register to make sure that all pending triggers have been serviced. When the SUSPEND and SUSPRDY bits are set, there are no triggers for all ADC cores. The SUSPRDY bit event can generate the common ADC interrupt if the SUSPCIE bit is set in the ADCON3L register.

4.10 Conversion Result

The module contains the data output registers for each analog input to store the A/D results, called the ADCBUFx (where 'x' is the number of the analog channel). The buffers are read-only. When data is written into a data register, the associated ANxRDY bit in the ADSTATL or ADSTATH register is set. When an ANxRDY bit is set, an interrupt request is generated. When a specific data register is read, the associated ANxRDY bit is immediately cleared. If a buffer location has not been read by the software, and the ADC needs to overwrite that location with a new conversion result, the previous data is lost.

4.10.1 FORMAT OF THE ADC RESULT

Writes of data from the A/D Converter into the ADC Result registers pass through the data formatter. The resulting data is formatted into a 16-bit word.

The data in the ADC Result register can be read in any of the four supported data formats. The user can select from unsigned integer, signed integer, unsigned fractional or signed fractional format. Integer data is right justified and fractional data is left justified.

- The integer/fractional data format selection is specified globally for all ADC inputs using the FORM bit in the ADCON1H register.
- The signed/unsigned data format selection can be independently specified for each individual input channel using the DIFFx and SIGNx bits in the ADMODnL or ADMODnH register, as described in **Section 4.6.2 "Single-Ended and Pseudodifferential Input Options and Output Result Format"**.
- Output data format depends on the ADC core resolution, specified by the RES<1:0> bits in the ADCOREnH register for the dedicated core and by the SHRRES<1:0> bits in the ADCON1H register for the shared core.

Figure 4-7, Figure 4-8, Figure 4-9 and Figure 4-10 illustrate how a result is formatted.

Figure 4-7: ADC Output Data Formats (12-Bit Resolution)

Analog-to-Digital Conversion Result

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

Unsigned Integer (SIGNx = 0, FORM = 0)

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

Signed Integer (SIGNx = 1, FORM = 0)

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

Fractional (SIGNx = 0, FORM = 1)

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

Signed Fractional (SIGNx = 1, FORM = 1)

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

Figure 4-8: ADC Output Data Formats (10-Bit Resolution)

Analog-to-Digital Conversion Result

d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Unsigned Integer (SIGNx = 0, FORM = 0)

0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	1	0

Signed Integer (SIGNx = 1, FORM = 0)

S	S	S	S	S	d08	d07	d06	d05	d04	d03	d02	d01	d00	1	0

Fractional (SIGNx = 0, FORM = 1)

d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	1	0	0	0	0	0

Signed Fractional (SIGNx = 1, FORM = 1)

S	d08	d07	d06	d05	d04	d03	d02	d01	d00	1	0	0	0	0	0

Figure 4-9: ADC Output Data Formats (8-Bit Resolution)

Analog-to-Digital Conversion Result

d07	d06	d05	d04	d03	d02	d01	d00

Unsigned Integer (SIGNx = 0, FORM = 0)

0	0	0	0	d07	d06	d05	d04	d03	d02	d01	d00	1	0	0	0

Signed Integer (SIGNx = 1, FORM = 0)

S	S	S	S	S	d06	d05	d04	d03	d02	d01	d00	1	0	0	0

Fractional (SIGNx = 0, FORM = 1)

d07	d06	d05	d04	d03	d02	d01	d00	1	0	0	0	0	0	0	0

Signed Fractional (SIGNx = 1, FORM = 1)

S	d06	d05	d04	d03	d02	d01	d00	1	0	0	0	0	0	0	0

Figure 4-10: ADC Output Data Formats (6-Bit Resolution)

Analog-to-Digital Conversion Result

d05	d04	d03	d02	d01	d00

Unsigned Integer (SIGNx = 0, FORM = 0)

0	0	0	0	d05	d04	d03	d02	d01	d00	1	0	0	0	0	0

Signed Integer (SIGNx = 1, FORM = 0)

S	S	S	S	S	d04	d03	d02	d01	d00	1	0	0	0	0	0

Fractional (SIGNx = 0, FORM = 1)

d05	d04	d03	d02	d01	d00	1	0	0	0	0	0	0	0	0	0

Signed Fractional (SIGNx = 1, FORM = 1)

S	d04	d03	d02	d01	d00	1	0	0	0	0	0	0	0	0	0

4.11 Digital Comparator

The ADC module features multiple digital comparators that can be used to monitor selected analog input conversion results and generate an interrupt when a conversion result is within, or not within, the user-specified limits. The comparison occurs automatically once the conversion is complete. The digital comparator is enabled by setting the Digital Comparator Enable bit, CMPEN, in the ADCMPnCON register.

An interrupt is generated when the Analog-to-Digital conversion result is in between or higher, or lower than the high and low limit values specified by the ADCMPnLO and ADCMPnHI registers. The CMPENx bits in the ADCMPnENL/H registers are used to specify which analog inputs are monitored by the digital comparator. The limit values written to ADCMPnLO and ADCMPnHI must match the data format selected by the FORM bit in the ADCON1H register, and the DIFFx and SIGNx bits in the ADMODnL/H registers.

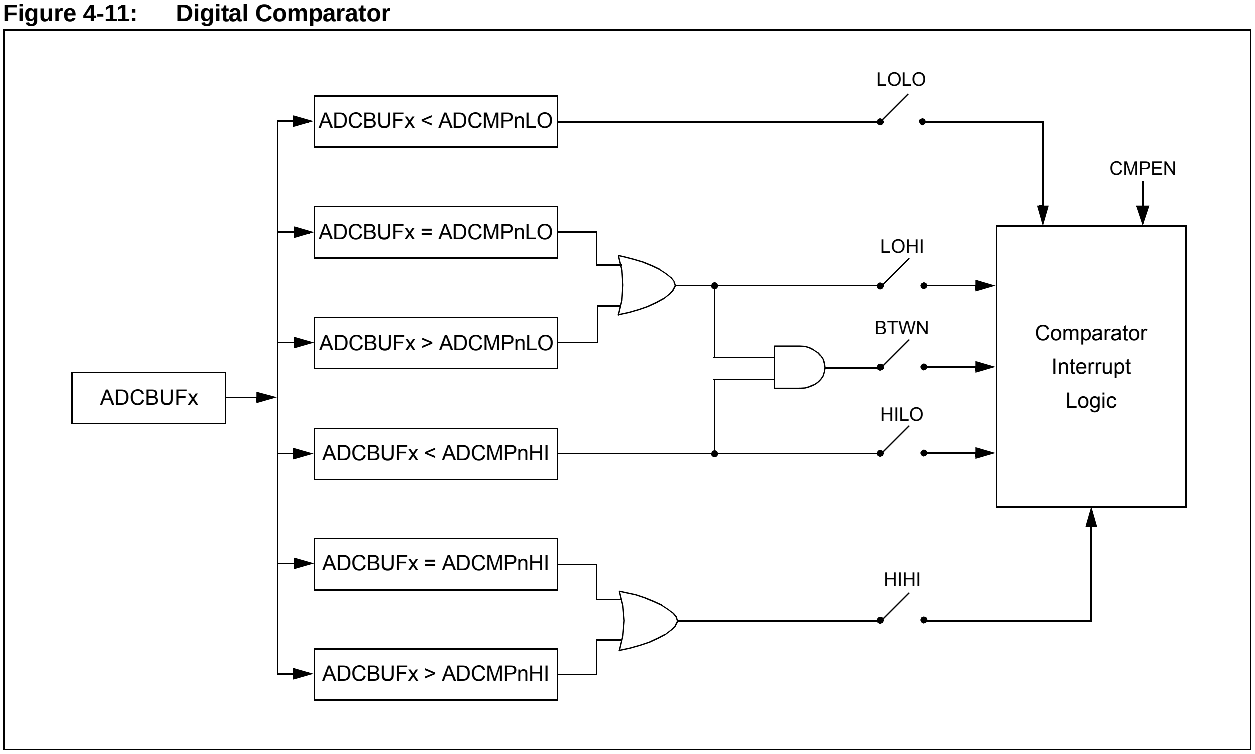
The ADCMPnCON register specifies the comparison conditions that will generate an interrupt:

- When BTWN = 1, an event is generated when ADCMPnLO ≤ ADCBUFx < ADCMPnHI
- When HIHI = 1, an event is generated when ADCBUFx ≥ ADCMPnHI
- When HILO = 1, an event is generated when ADCBUFx < ADCMPnHI
- When LOHI = 1, an event is generated when ADCBUFx ≥ ADCMPnLO
- When LOLO = 1, an event is generated when ADCBUFx < ADCMPnLO

The comparator event generation is illustrated in Figure 4-11. When the ADC generates a conversion result, the digital comparator compares the ADC result for the selected channels with the high and low limit values (depending on the selected comparison criteria in the ADCMPnCON register). If a comparator event occurs, the Comparator Event Status bit, STAT, is set and the Input Channel Number bits, CHNL<4:0>, are automatically updated in the ADCMPnCON register so that the application knows which analog input generated the event. Reading the CHNL<4:0> bits clears the STAT flag. The comparator can generate individual and common interrupts if the IE bit in the ADCMPnCON register is set.

Note: The application must format the values contained in the ADCMPnLO and ADCMPnHI registers to match the converted data format as either signed or unsigned, and fractional or integer.

Figure 4-11: Digital Comparator



4.12 Oversampling Digital Filter

The ADC module may support multiple oversampling digital filters. The filter consists of an accumulator and a decimator (down-sampler), which function together as a low-pass filter. By sampling an analog input at a higher than required sample rate, and then processing the data through the filter, the effective resolution of the ADC module can be increased at the expense of decreased conversion throughput. For example, using 4x oversampling yields one extra bit of resolution, 16x oversampling yields two extra bits of resolution, 64x oversampling provides three extra bits of resolution and 256x oversampling provides four extra bits of resolution.

To perform a conversion using the oversampling digital filter:

1. Select the oversampling ratio with the OVRSAM<2:0> bits in the ADFLnCON register.
2. Set the sample time for subsequent conversions:
   - For dedicated ADC core inputs, enable a delay between the trigger and the conversion start by setting the corresponding SAMCxEN bit in the ADCON4L register, and select the sample time of the recurring conversions using the SAMC<9:0> bits in the ADCOREnL register
   - For shared ADC core inputs, select the sample time using the SHRSAMC<9:0> bits in the ADCON2H register
3. Select the specific analog input to be oversampled by configuring the FLCHSEL<4:0> bits in the ADFLnCON register.
4. Select the Averaging or Oversampling mode using the MODE<1:0> bits in the ADFLnCON register.
5. Enable the oversampling filter by setting the FLEN bit.

Once the oversampling digital filter is configured, it waits for an input channel trigger to initiate the oversampling process. The trigger causes the accumulator to be cleared and initiates the first conversion. After this initial trigger from the input channel, all of the next triggers are generated by the filter itself automatically. Once each conversion request has been processed, sampling is initiated based on the values of the SAMC<9:0> or SHRSAMC<9:0> bits for dedicated, or shared ADC core, respectively. This process continues until the required number of samples (4, 8, 16, 32, 64, 128 or 256) has been converted. When the converted samples have been summed, the output is transferred to the ADFLnDAT register and the RDY bit in the ADFLnCON register is set. Reading the ADFLnDAT register clears the RDY flag. The filter will generate individual and common interrupts if the IE bit in the ADFLnCON register is set. The filter does not support the fractional data format; if the filter is used, the FORM bit in the ADCON1H register must be ‘0’.

Figure 4-12 illustrates 4x oversampling on an input for the dedicated ADC core. Prior to the trigger, the ADC core is tracking the input signal. The trigger starts the oversampling process. When the sampling delay defined by the SAMC<9:0> bits has elapsed, the conversion is started. Then, a new sampling/conversion sequence occurs. After each sample is converted, it is added to the accumulator. The sequence repeats until the number of samples specified by the OVRSAM<2:0> bits field has been accumulated. When the last sample has been converted, its value is added to the accumulator. The result is formatted as defined by the MODE<1:0> bits and then stored in the ADFLnDAT register. Reading the ADFLnDAT register clears the RDY flag.

Figure 4-12: 4x Oversampling of an Analog Input for Dedicated ADC Core

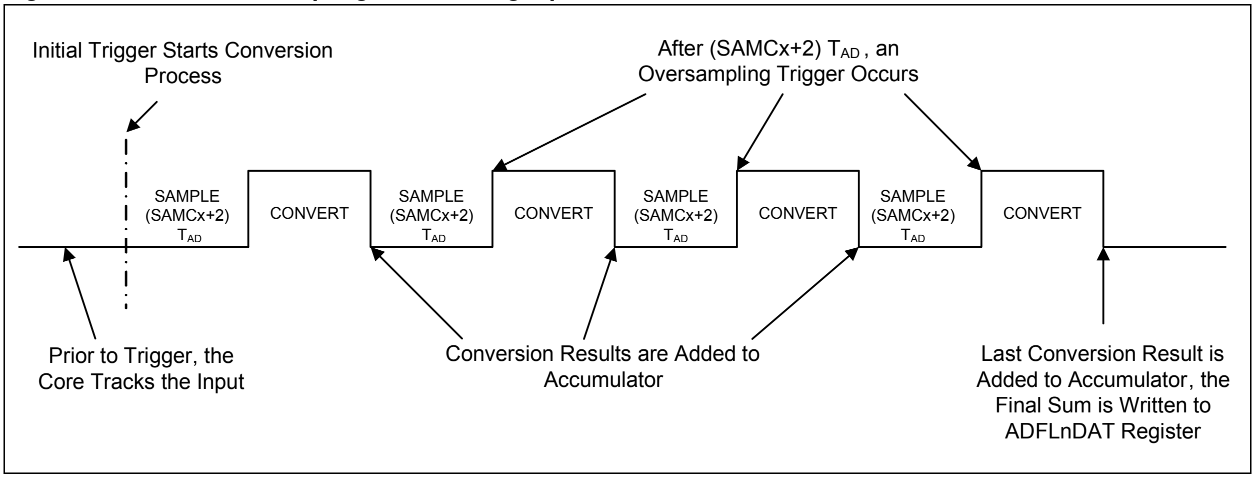
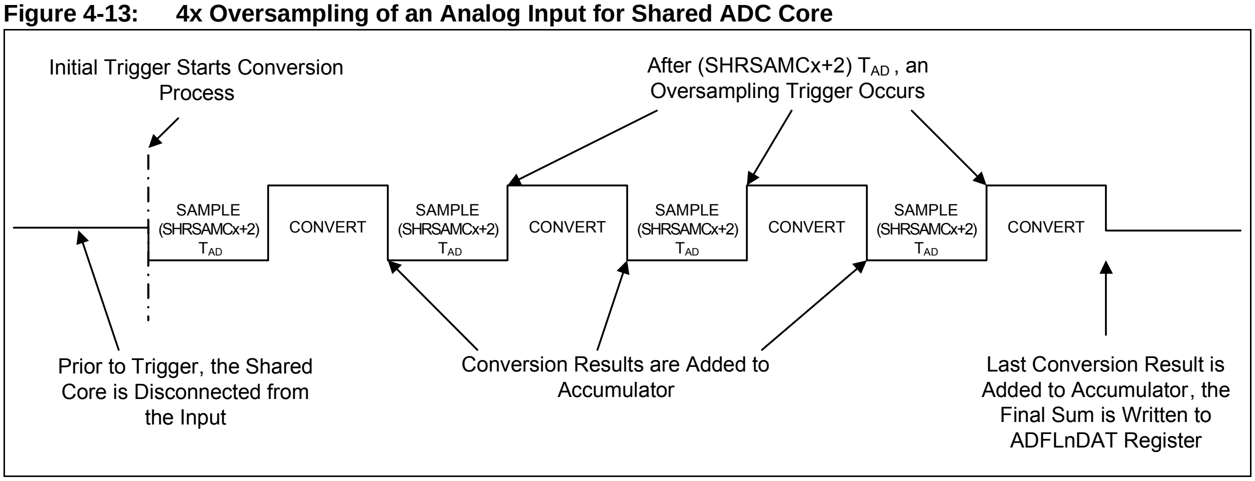


Figure 4-13 illustrates 4x oversampling using an input for the shared ADC core. The input channel trigger initiates sampling for the length of time defined by the SHRSAMC<9:0> bits.

Figure 4-13: 4x Oversampling of an Analog Input for Shared ADC Core



4.13 Interrupts

The ADC module can generate individual interrupts for a variety of sources. Also, the common interrupt is called for all ADC events. An early interrupt feature is available to compensate for interrupt servicing latency. After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The CPU then begins executing code at the vector address. The application at this vector address should perform the required operations, such as processing the data results, clearing the interrupt flags and then exiting. The interrupt controller flags cannot be cleared until the corresponding event flags are cleared or the corresponding interrupts are disabled in the ADC module.

4.13.1 INDIVIDUAL INTERRUPTS

Many events generated by the ADC have their own unique interrupt vectors. This can significantly optimize the servicing of multiple ADC events by keeping each Interrupt Service Routine (ISR) focused on efficiently handling a specific event.

Individual interrupts are generated from the following events:

- Individual Input Data Ready Event: Upon completion of a conversion from an analog input source (ANx), the corresponding ANxRDY bit associated with that input becomes set in the ADSTATL/H registers. Each of the ANxRDY bits is associated with its own ADCANxIF interrupt flag at the device level. To clear the ADC Interrupt Controller Flag, ADCANxIF, the ANxRDY bit must be cleared first by reading the ADCBUFx register. To enable the individual input channel interrupt, the corresponding IEx bit must be set in the ADIEL or ADIEH register.
- Digital Comparator Event: When a conversion's comparison criteria is met for the enabled digital comparator, the STAT bit becomes set in the ADCMPnCON register. Each of the digital comparators is capable of generating its own device-level interrupt, controlled by the DCMPxIF flag, when the corresponding STAT bit is set. To clear the DCMPxIF interrupt controller flag, the STAT bit must be cleared first by reading the CHNL<4:0>bits in the ADCMPnCON register. To enable the individual comparator interrupt, the corresponding IE bit must be set in the ADCMPnCON register.
- Oversampling Filter Data Ready Event: When an oversampling filter has completed the accumulation/decimation process and has stored the result, the RDY bit becomes set in the ADFLnCON register. Each of the oversampling filters is capable of generating its own device-level interrupt, controlled by the ADFLTRxIF flag, when the corresponding RDY bit is set. To clear the ADFLTRxIF interrupt controller flag, the RDY bit must be cleared first by reading the ADFLnDAT register. To enable the individual filter interrupt, the corresponding IE bit must be set in the ADFLnCON register.

As with other interrupts, the corresponding interrupt controller enable bits (ADCANxIE, DCMPxIE or ADFLTRxIE) must be set in order for the application to vector to the ISR when their interrupt flags are set.

4.13.2 COMMON INTERRUPT

One common interrupt is used for all ADC events. Any interrupt event enabled in the ADC module sets the ADCIF flag in the interrupt controller. This common ADC Interrupt Flag (ADCIF) stays set and cannot be cleared until all status flags enabled in the ADC module are cleared or the corresponding interrupts are disabled in the ADC module. The common interrupt is generated:

- For each individual input data ready interrupt event when the corresponding IEx bit is set in the ADIEL or ADIEH register.
- For each digital comparator interrupt event when the corresponding IE bit is set in the ADCMPnCON register.
- For each oversampling filter interrupt event when the corresponding IE bit is set in the ADFLnCON register.
- For each ADC core power ready event when the corresponding CxCIE or SHRCIE bit is set in the ADCON5H register. The power ready events flags (CxRDY and SHRRDY in the ADCON5L register) stay set when the used ADC cores are active. Thus, after the event is detected, the corresponding interrupts must be disabled (CxCIE = 0 and SHRCIE = 0) to clear the ADCIF flag.
- When the SUSPCIE bit is set in the ADCON3L register for the event when the ADC triggers are suspended. The All ADC Cores Suspended flag (SUSPRDY in the ADCON3L register) stays set until the ADC triggers are resumed by clearing the SUSPEND bit in the ADCON3L register. Thus, after the event is detected, the corresponding interrupt must be disabled (SUSPCIE = 0) to clear the ADCIF flag.
- When the REFCIE and REFERCIE bits in the ADCON2L register are set for the reference voltage ready, and the reference voltage Fault events, respectively. The event flags (REFRDY and REFERR in the ADCON2H register) can stay set until the ADC is disabled by the ADON bit in the ADCON1L register. Thus, after the events are detected, the corresponding interrupts must be disabled (REFCIE = 0 and REFERCIE = 0) to clear the ADCIF flag.

As with other interrupts, the corresponding ADC Interrupt Enable bit, ADCIE, must be set in order for the application to vector to the common ISR.

4.13.3 EARLY INTERRUPTS

The early interrupt can improve the throughput of a system by overlapping the completion of the ADC conversion with the processor overhead associated with an interrupt. To use this feature, the EIEN bit in the ADCON2L register should be set. If this bit is set, common and individual interrupts for all cores (input channels) will be executed prior to completion of the conversion. When the early interrupt for the input channel is generated, the corresponding EISTATx bit is set in the ADEISTATL or ADEISTATH register. The EISTATx flag is cleared when the corresponding ADCBUFx register is read. The interrupt is enabled for each input channel, individually, using the EIEx bit in the ADEIEL or ADEIEH register.

Even though the input is still in the conversion process, the application software can use the “head start” to begin execution of the entry into the ISR.

The value stored in the EISEL<2:0> bits in the ADCOREnH register for the dedicated ADC cores, and the SHREISEL<2:0> bits in the ADCON2L register for the shared ADC core inputs, determines the number of TADCORE clock cycles that the ADC core early interrupt will execute prior to the completion of the conversion.

Early interrupts can reduce the latency from the moment the analog input was triggered, until the point in time when the application software can use the data. On a conversion request, the early interrupt option allows the corresponding analog input interrupt to be processed without any latency (zero latency).

Note 1: The early interrupts are enabled for all cores (input channels) at the same time (EIEN bit = `1`). It is not possible to select the early interrupt feature for each channel individually, but the early interrupt latency (timing) can be set separately for each SAR ADC core (EISEL<2:0> bits for dedicated cores and SHREISEL<2:0> bits for the shared core).

2: When early interrupts are enabled, the settings in the ADIEL and ADIEH registers have no effect. To enable interrupts, the ADEIEL and ADEIEH registers should be used instead.

3: For the 6-bit ADC core resolution (RES<1:0> or SHRRES<1:0> = `00`), the EISEL<2:0> or SHREISEL<2:0> settings, from ‘`100`’ (5 TADCORE) to ‘`111`’ (8 TADCORE), are not valid and should not be used. For the 8-bit ADC core resolution (RES<1:0> or SHRRES<1:0> = `01`), the EISEL<2:0> or SHREISEL<2:0> settings, ‘`110`’ (7 TADCORE) and ‘`111`’ (8 TADCORE), are not valid and should not be used.

4.14 Capacitive Voltage Divider (CVD)

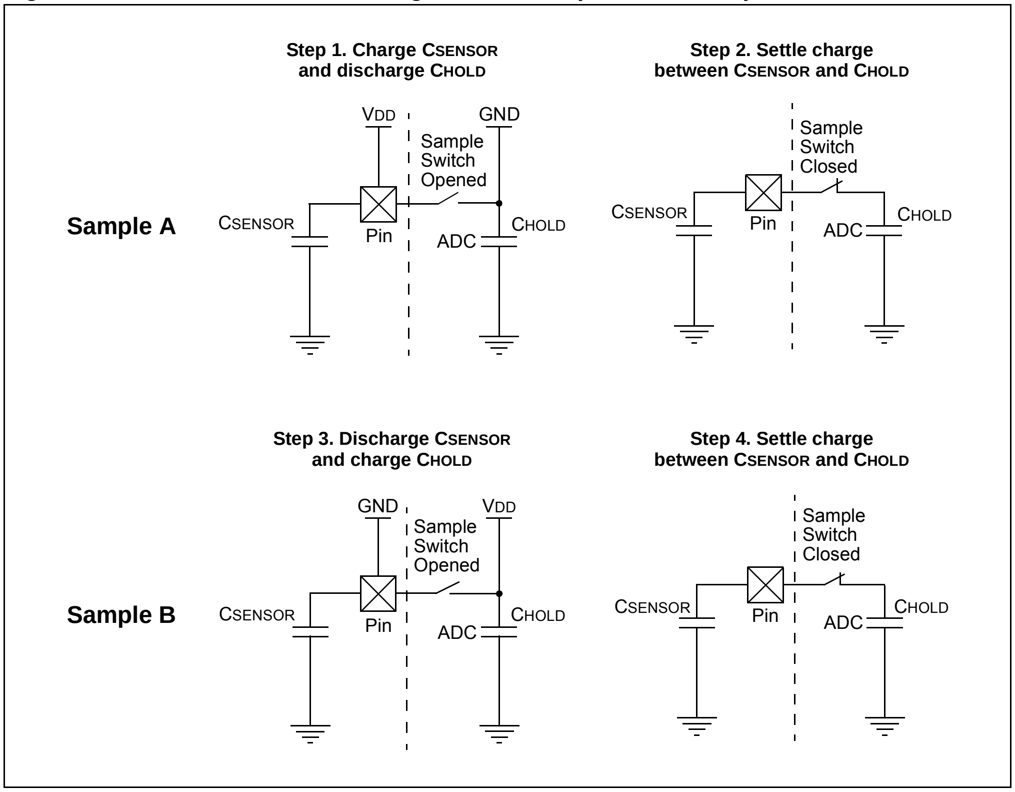
The shared core of the ADC can include a hardware for the Capacitive Voltage Divider (CVD) algorithm support. This feature is not available on all devices. Refer to the specific device data sheet to see if the CVD feature is implemented. The CVD allows measuring a capacitance connected to the ADC input. It can be used to detect an event of touch in touch sensor applications.

To add an analog input to the CVD scan list, the corresponding bit in the ADCSSL or ADCSSH register must be set. The CVD feature can be enabled by the CVDEN bit in the ADCON1L register only after the ADC is enabled and calibrated. When the CVDEN bit is set, the CVD hardware automatically runs the CVD sequence on all inputs selected in the ADCSSL and ADCSSH registers.

The CVD measurement algorithm consists of two phases, Sample A and Sample B:

- **Sample A:** The Capacitive Sensor, C_{SENSOR}, is connected to the I/O power (V_{DD}, charged). The internal ADC Sample-and-Hold Capacitor, C_{HOLD}, is connected to ground (GND, discharged), as shown in Figure 4-14, Step 1. Then, both capacitors are connected in parallel for half of the sampling time, set by the SHRSAMC<9:0> bits in the ADCCON2H register; as shown in Figure 4-14, Step 2. After the sampling, the charge between capacitors is settled. The resulting voltage is proportional to a ratio of the C_{SENSOR} and C_{SENSOR} + C_{HOLD} capacitances. Then, this voltage is converted by the ADC shared core, the corresponding channel ADxRDY flag is set in the ADSTATL or ADSTAH register, the result is stored in the ADCBUFx register and an individual channel interrupt is generated.
- **Sample B:** The Capacitive Sensor, C_{SENSOR}, is connected to ground (GND, discharged). The internal ADC Sample-and-Hold Capacitor, C_{HOLD}, is connected to the I/O power (V_{DD}, charged), as shown in Figure 4-14, Step 3. Then, both capacitors are connected in parallel for half of the sampling time, set by the SHRSAMC<9:0> bits in the ADCCON2H register; as shown in Figure 4-14, Step 4. After sampling, the charge between capacitors is settled. The resulting voltage is proportional to a ratio of the C_{HOLD} and C_{SENSOR} + C_{HOLD} capacitances. Then, this voltage is converted by the ADC shared core, the corresponding channel ADxRDY flag is set in the ADSTATL or ADSTAH register, the result is stored in the ADCBUFx register and an individual channel interrupt is generated.

Figure 4-14: CVD Connection Diagrams for Sample A and Sample B



Equation 4-2 shows the relationship between the CVD result and the capacitances of CSENSOR and CHOLD:

Equation 4-2: CVD Result Calculation for 12-Bit Resolution

$$Sample\ A - Sample\ B = 4095 \times \frac{V_{DD}}{V_{R+} - V_{R-}} \times \frac{C_{SENSOR} - C_{HOLD}}{C_{SENSOR} + C_{HOLD}}$$

Where:
V_{R+} = Positive Reference Voltage
V_{R-} = Negative Reference Voltage

To ensure maximum sensitivity, CSENSOR should have a similar value as CHOLD, which can be done by adding an additional capacitance to CHOLD using the CVDCAP<2:0> bits in the ADCCON2H register.

When CVD is enabled (CVDEN bit is set), the Digital Comparator 0 (if it is implemented on the device) is automatically linked to the CVD hardware, and it can be used to generate a comparator event and interrupt. The Digital Comparator 0 operates as described in **Section 4.11 "Digital Comparator"** with just one difference: in CVD mode, the channel selections in the ADCMP0ENL and ADCMP0ENH registers are ignored; the Digital Comparator 0 monitors only the Sample A – Sample B value for the current CVD channel. The analog input that caused the event is stored in the CHNL<4:0> bits in the ADCMP0CON register. Also, if an event is detected, then the Sample A – Sample B value is stored in the ADCVDDAT register in signed format.

5.0 APPLICATION EXAMPLES

To use the 12-Bit High-Speed, Multiple SARs A/D Converter:

1. Configure the I/O pins to be used as analog inputs by setting the corresponding bits in the ANSELx and TRISx registers.
2. Select the common ADC clock source and configure the prescaler using the CLKSEL<1:0> and CLKDIV<5:0> bits in the ADCON3H register.
3. Select the clock period for each ADC core using the ADCS<6:0> bits in the ADCOREnH registers, and the SHRADCS<6:0> bits in the ADCON2L register, for the dedicated and shared cores, respectively.
4. Configure the ADC reference sources using the REFSEL<2:0> bits in the ADCON3L register.
5. Select the result resolution for each ADC core using the RES<1:0> bits in the ADCOREnH registers, and the SHRRES<1:0> bits in the ADCON1H register, for the dedicated and shared cores, respectively.
6. Configure the data output format for integer or fractional using the FORM bit in the ADCON1H register.
7. Select single or differential input configuration, and output format for each input channel, using the DIFFx and SIGNx bits in the ADMODnL or ADMODnH registers.
8. If the shared ADC core is used, configure the Shared ADC Core Sample Time Selection bits, SHRSAMC<8:0>, in the ADCON2H register.
9. Configure and enable the ADC interrupts.
10. Set the ADON bit in the ADCON1L register to enable the module and set the WARMTIME<3:0> bits in the ADCON5H register to provide at least 10 µS for the initialization.
11. Turn on the module power:
    a) Set the CxPWR and SHRPWR bits in the ADCON5L register.
    b) Poll the CxRDY or SHRRDY bits in the ADCON5L register until they are set.
    c) Set the CxEN or SHREN bits in the ADCON3H register.
12. Calibrate all ADC cores using the ADCALnL and ADCALnH registers:
    a) Set the CALxEN and CSHREN bits.
    b) Clear the CALxDIFF and CSHRDIFF bits.
    c) Set the CALxRUN and CSHRRUN bits.
    d) Poll the CALxRDY and CSHRRDY bits until they are set.
    e) Set the CALxDIFF and CSHRDIFF bits.
    f) Set the CALxRUN and CSHRRUN bits.
    g) Poll the CALxRDY and CSHRRDY bits until they are set.
    h) Clear the CALxEN and CSHREN bits.
13. Set the trigger source for each analog input in the corresponding ADTRIGnL or ADTRIGnH registers.

The following sections provide some typical examples for using the various features of the ADC.

5.1 Turning On and Calibrating ADC Cores

Example 5-1 shows the procedure to enable and calibrate 2 dedicated, and 1 shared, ADC cores. The calibration step may not be required for some devices. Refer to the specific device data sheet to see if the calibration is required and the Calibration registers, ADCALnL and ADCALnH, are implemented.

Example 5-1: ADC Turn-On and Calibration Procedure

```
void EnableAndCalibrate()
{
// Set initialization time to maximum
ADCON5Hbits.WARMTIME = 15;

// Turn on ADC module
ADCON1Lbits.ADON = 1;

// Turn on analog power for dedicated core 0
ADCON5Lbits.C0PWR = 1;
// Wait when the core 0 is ready for operation
while(ADCON5Lbits.C0RDY == 0);
// Turn on digital power to enable triggers to the core 0
ADCON3Hbits.C0EN = 1;

// Turn on analog power for dedicated core 1
ADCON5Lbits.C1PWR = 1;
// Wait when the core 1 is ready for operation
while(ADCON5Lbits.C1RDY == 0);
// Turn on digital power to enable triggers to the core 1
ADCON3Hbits.C1EN = 1;

// Turn on analog power for shared core
ADCON5Lbits.SHRPWR = 1;
// Wait when the shared core is ready for operation
while(ADCON5Lbits.SHRRDY == 0);
// Turn on digital power to enable triggers to the shared core
ADCON3Hbits.SHREN = 1;

// Enable calibration for the dedicated core 0
ADCAL0Lbits.CAL0EN = 1;
// Single-ended input calibration
ADCAL0Lbits.CAL0DIFF = 0;
// Start calibration
ADCAL0Lbits.CAL0RUN = 1;
// Poll for the calibration end
while(ADCAL0Lbits.CAL0RDY == 0);
// Differential input calibration
ADCAL0Lbits.CAL0DIFF = 1;
// Start calibration
ADCAL0Lbits.CAL0RUN = 1;
// Poll for the calibration end
while(ADCAL0Lbits.CAL0RDY == 0);
// End the core 0 calibration
ADCAL0Lbits.CAL0EN = 0;
```

Example 5-1: ADC Turn-On and Calibration Procedure (Continued)

```
// Enable calibration for the dedicated core 1
ADCAL0Lbits.CAL1EN = 1;
// Single-ended input calibration
ADCAL0Lbits.CAL1DIFF = 0;
// Start calibration
ADCAL0Lbits.CAL1RUN = 1;
// Poll for the calibration end
while(ADCAL0Lbits.CAL1RDY == 0);
// Differential input calibration
ADCAL0Lbits.CAL1DIFF = 1;
// Start calibration
ADCAL0Lbits.CAL1RUN = 1;
// Poll for the calibration end
while(ADCAL0Lbits.CAL1RDY == 0);
// End the core 1 calibration
ADCAL0Lbits.CAL1EN = 0;

// Enable calibration for the shared core
ADCAL1Hbits.CSHREN = 1;
// Single-ended input calibration
ADCAL1Hbits.CSHRDIFF = 0;
// Start calibration
ADCAL1Hbits.CSHRRUN = 1;
// Poll for the calibration end
while(ADCAL1Hbits.CSHRRDY == 0);
// Differential input calibration
ADCAL1Hbits.CSHRDIFF = 1;
// Start calibration
ADCAL1Hbits.CSHRRUN = 1;
// Poll for the calibration end
while(ADCAL1Hbits.CSHRRDY == 0);
// End the shared core calibration
ADCAL1Hbits.CSHREN = 0;
}
```

5.2 Basic Conversion Sequence

A basic sequence for initialization and conversion is shown in Example 5-2. This example demonstrates the simultaneous sampling and conversion from two dedicated ADC cores. All inputs are triggered from a single source (Timer2). The individual input ISRs are used to store the result of the conversion.

Example 5-2: Simultaneous Sample and Conversion for Dedicated ADC Cores

```
// These variables will keep the conversion result.
volatile unsigned short dataAN0;
volatile unsigned short dataAN1;

int     main()
{
// ADC INITIALIZATION
// Configure the I/O pins to be used as analog inputs.
ANSELAbits.ANSA0 = 1; TRISAbits.TRISA0 = 1;  // AN0/RA0 connected the dedicated core 0
ANSELAbits.ANSA1 = 1; TRISAbits.TRISA1 = 1;  // AN1/RA1 connected the dedicated core 1

// Configure the common ADC clock.
ADCON3Hbits.CLKSEL = 2;                      // clock from FRC oscillator
ADCON3Hbits.CLKDIV = 0;                      // no clock divider (1:1)
// Configure the cores' ADC clock.
ADCORE0Hbits.ADCS = 0;                       // clock divider (1:2)
ADCORE1Hbits.ADCS = 0;                       // clock divider (1:2)

// Configure the ADC reference sources.
ADCON3Lbits.REFSEL = 0;                      // AVdd as voltage reference
// Configure the integer of fractional output format.
ADCON1Hbits.FORM = 0;                        // integer format

// Select single-ended input configuration and unsigned output format.
ADMOD0Lbits.SIGN0 = 0;                       // AN0/RA0
ADMOD0Lbits.DIFF0 = 0;                       // AN0/RA0
ADMOD0Lbits.SIGN1 = 0;                       // AN1/RA1
ADMOD0Lbits.DIFF1 = 0;                       // AN1/RA1

// Enable and calibrate the module.
EnableAndCalibrate();                        // See Example 5-1

// Configure and enable ADC interrupts.
ADIELbits.IE0 = 1;                           // enable interrupt for AN0
ADIELbits.IE1 = 1;                           // enable interrupt for AN1
_ADCAN0IF = 0;                               // clear interrupt flag for AN0
_ADCAN0IE = 1;                               // enable interrupt for AN0
_ADCAN1IF = 0;                               // clear interrupt flag for AN1
_ADCAN1IE = 1;                               // enable interrupt for AN1
```

Example 5-2: Simultaneous Sample and Conversion for Dedicated ADC Cores (Continued)

```
// Set same trigger source for all inputs to sample signals simultaneously.
ADTRIG0Lbits.TRGSRC0 = 13;                    // timer 2 for AN0
ADTRIG0Lbits.TRGSRC1 = 13;                    // timer 2 for AN1

// TIMER 2 INITIALIZATION (TIMER IS USED AS A TRIGGER SOURCE FOR ALL CHANNELS).
T2CONbits.TCS = 0;                            // clock from peripheral clock
T2CONbits.TCKPS = 0;                          // 1:1 prescale
PR2 = 0x8000;                                 // rollover every 0x8000 clocks
T2CONbits.TON = 1;                            // start timer to generate ADC triggers
while(1);
return 1;
}

// ADC AN0 ISR
void __attribute__((interrupt, no_auto_psv)) _ADCAN0Interrupt(void)
{
dataAN0 = ADCBUF0;                            // read conversion result
_ADCAN0IF = 0;                                // clear interrupt flag
}

// ADC AN1 ISR
void __attribute__((interrupt, no_auto_psv)) _ADCAN1Interrupt(void)
{
dataAN1 = ADCBUF1;                            // read conversion result
_ADCAN1IF = 0;                                // clear interrupt flag
}
```

5.3 Digital Comparator

The digital comparator allows monitoring selected analog input conversion results. For the digital comparator module, the recommended initialization sequence is:

1. Set high and low thresholds in the ADCMPnLO and ADCMPnHI registers in the format selected for the output data in the ADCBUFx registers.
2. Select analog inputs for the comparison by setting the corresponding bits in the ADCMPnENL/H registers.
3. Set the comparator event rules using the BTWN, HIHI, HILO, LOHI and LOLO bits in the ADCMPnCON register.
4. Configure and enable the ADC digital comparator interrupt.
5. Set the CMPEN bit in the ADCMPnCON register to enable the comparator.

Example 5-3 demonstrates using the digital comparator to detect voltages for two analog inputs that are outside the thresholds defined in ADCMPnLO and ADCMPnHI.

Example 5-3: Using the Digital Comparator

```
// This variable will contain the analog input number
// which has generated an event for the digital comparator.
volatile unsigned short comparatorChannelNumber;

// These variables will keep the conversion result.
volatile unsigned short dataAN0;
volatile unsigned short dataAN1;

int     main()
{
// ADC INITIALIZATION
// Configure the I/O pins to be used as analog inputs.
ANSELAbits.ANSA0 = 1; TRISAbits.TRISA0 = 1;  // AN0/RA0 connected the dedicated core 0
ANSELAbits.ANSA1 = 1; TRISAbits.TRISA1 = 1;  // AN1/RA1 connected the dedicated core 1

// Configure the common ADC clock.
ADCON3Hbits.CLKSEL = 2;                      // clock from FRC oscillator
ADCON3Hbits.CLKDIV = 0;                      // no clock divider (1:1)
// Configure the cores’ ADC clock.
ADCORE0Hbits.ADCS = 0;                       // clock divider (1:2)
ADCORE1Hbits.ADCS = 0;                       // clock divider (1:2)

// Configure the ADC reference sources.
ADCON3Lbits.REFSEL = 0;                      // AVdd as voltage reference
// Configure the integer of fractional output format.
ADCON1Hbits.FORM = 0;                        // integer format

// Select single-ended input configuration and unsigned output format.
ADMOD0Lbits.SIGN0 = 0;                       // AN0/RA0
ADMOD0Lbits.DIFF0 = 0;                       // AN0/RA0
ADMOD0Lbits.SIGN1 = 0;                       // AN1/RA1
ADMOD0Lbits.DIFF1 = 0;                       // AN1/RA1

// Enable and calibrate the module.
EnableAndCalibrate();                        // See Example 5-1
// Configure and enable ADC interrupts.
ADIELbits.IE0 = 1;                           // enable interrupt for AN0
ADIELbits.IE1 = 1;                           // enable interrupt for AN1
_ADCAN0IF = 0;                               // clear interrupt flag for AN0
_ADCAN0IE = 1;                               // enable interrupt for AN0
_ADCAN1IF = 0;                               // clear interrupt flag for AN1
_ADCAN1IE = 1;                               // enable interrupt for AN1
```

Example 5-3: Using the Digital Comparator (Continued)

```
// Set same trigger source for all inputs to sample signals simultaneously.
ADTRIG0Lbits.TRGSRC0 = 13;                     // timer 2 for AN0
ADTRIG0Lbits.TRGSRC1 = 13;                     // timer 2 for AN1

// TIMER 2 INITIALIZATION (TIMER IS USED AS A TRIGGER SOURCE FOR ALL CHANNELS).
T2CONbits.TCS = 0;                             // clock from peripheral clock
T2CONbits.TCKPS = 0;                           // 1:1 prescale
PR2 = 0x8000;                                  // rollover every 0x8000 clocks
T2CONbits.TON = 1;                             // start timer to generate ADC triggers

// DIGITAL COMPARATOR INITIALIZATION.
// Set high and low thresholds.
ADCMP0LO = 1024;
ADCMP0HI = 4096-1024;
// Select analog inputs for the comparison.
ADCMP0ENLbits.CMPEN0 = 1;                      // AN0
ADCMP0ENLbits.CMPEN1 = 1;                      // AN1
// Set the comparator event rule.
ADCMP0CONbits.LOLO = 1;                        // Generate interrupt if input level is outside window
ADCMP0CONbits.HIHI = 1;                        // specified by ADCMP0LO and ADCMP0HI.
// Enable the ADC Digital Comparator interrupt.
ADCMP0CONbits.IE = 1;
_ADCMP0IF = 0;
_ADCMP0IE = 1;
// Enable the comparator.
ADCMP0CONbits.CMPEN = 1;

while(1);
return 1;
}
// ADC COMPARATOR ISR
// If the conversion result for AN0 or AN1 is less than ADCMP0LO or more than ADCMP0HI,
// the interrupt is generated
void __attribute__((interrupt, no_auto_psv)) _ADCMP0Interrupt(void)
{
comparatorChannelNumber = ADCMP0CONbits.CHNL;   // read the channel number
                                               // that generated interrupt
_ADCMP0IF = 0;                                 // clear interrupt flag
}
// ADC AN0 ISR
void __attribute__((interrupt, no_auto_psv)) _ADCAN0Interrupt(void)
{
dataAN0 = ADCBUF0;                             // read conversion result
_ADCAN0IF = 0;                                 // clear interrupt flag
}
// ADC AN1 ISR
void __attribute__((interrupt, no_auto_psv)) _ADCAN1Interrupt(void)
{
dataAN1 = ADCBUF1;                             // read conversion result
_ADCAN1IF = 0;                                 // clear interrupt flag
}
```

5.4 Oversampling Filter

The oversampling filter can help to increase the resolution or to filter the noise. The general initialization procedure for the oversampling filter is:

1. Select the analog input for the filter using the FLCHSEL<4:0> bits in the ADFLnCON register.
2. Set the Filter mode using the MODE<1:0> bits in the ADFLnCON register.
3. Set the desired oversampling factor using the OVRSAM<2:0> bits in the ADFLnCON register.
4. Set the correct sampling time using the SAMC<9:0> bits in the ADCOREnL register for the dedicated ADC core and the SHRSAMC<9:0> bits in the ADCON2H register for the shared core.
5. Configure and enable the ADC oversampling filter interrupt.
6. Set the FLEN bit in the ADFLnCON register to enable the filter.

Example 5-4 demonstrates the filtering for the analog input connected to the dedicated ADC core.

Example 5-4: Using the Oversampling Filter

```
// This variable will contain the output data from the oversampling filter.
volatile unsigned short filterData;

int     main()
{
// ADC INITIALIZATION
// Configure the I/O pins to be used as analog inputs.
ANSELAbits.ANSA0 = 1; TRISAbits.TRISA0 = 1;  // AN0/RA0 connected the dedicated core 0

// Configure the common ADC clock.
ADCON3Hbits.CLKSEL = 2;                      // clock from FRC oscillator
ADCON3Hbits.CLKDIV = 0;                      // no clock divider (1:1)
// Configure the cores' ADC clock.
ADCORE0Hbits.ADCS = 0;                       // clock divider (1:2)

// Configure the ADC reference sources.
ADCON3Lbits.REFSEL = 0;                      // AVdd as voltage reference
// Configure the integer of fractional output format.
ADCON1Hbits.FORM = 0;                        // integer format

// Select single-ended input configuration and unsigned output format.
ADMOD0Lbits.SIGN0 = 0;                       // AN0/RA0
ADMOD0Lbits.DIFF0 = 0;                       // AN0/RA1

// Enable and calibrate the module.
EnableAndCalibrate();                        // See Example 5-1

// Set software common trigger as AN0 input trigger source.
ADTRIG0Lbits.TRGSRC0 = 1;

// OVERSAMPLING FILTER INITIALIZATION.
ADFL0CONbits.FLCHSEL = 0;                    // Select the AN0 input for the filter.
ADFL0CONbits.MODE = 3;                       // Averaging, 12-bit result.
ADFL0CONbits.OVRSAM = 6;                     // 128X
```

Example 5-4: Using the Oversampling Filter (Continued)

```
// Enable delay between trigger and the conversion start (SAMC bits).
ADCON4Lbits.SAMC0EN = 1;
// Set sampling time (10x Tad)
ADCORE0Lbits.SAMC = 10;

// Enable the filter.
ADFL0CONbits.FLEN = 1;

while(1)
{
// Generate Software Common Trigger
ADCON3Lbits.SWCTRG = 1;

// Wait for the filter result is ready.
while(ADFL0CONbits.RDY == 0);

// Read result (it will clear RDY bit)
filterData = ADFL0DAT;

}

return 1;
}
```

5.5 Early Interrupts

The early interrupt can improve the throughput of a system by overlapping the completion of the ADC conversion with the processor overhead associated with an interrupt. The code in Example 5-5 shows the early interrupts usage. In this example, interrupts for the Dedicated Core 0 and the shared core are generated before the conversion completion (1 TAD cycle for the dedicated core and 4 TAD cycles for the shared core).

Example 5-5: Using Early Interrupts

```
// These variables will keep the conversion result.
volatile unsigned short dataAN0;             // dedicated core
volatile unsigned short dataAN2;             // shared core

int     main()
{
// ADC INITIALIZATION
// Configure the I/O pins to be used as analog inputs.
ANSELAbits.ANSA0 = 1; TRISAbits.TRISA0 = 1;  // AN0/RA0 connected the dedicated core 0
ANSELAbits.ANSA2 = 1; TRISAbits.TRISA2 = 1;  // AN2/RA2 connected the shared core
// Configure the common ADC clock.
ADCON3Hbits.CLKSEL = 2;                      // clock from FRC oscillator
ADCON3Hbits.CLKDIV = 0;                      // no clock divider (1:1)
// Configure the cores' ADC clock.
ADCORE0Hbits.ADCS = 0;                       // dedicated core clock divider (1:2)
ADCON2Lbits.SHRADCS = 0;                     // shared core clock divider (1:2)
// Configure sample time for shared core.
ADCON2Hbits.SHRSAMC = 10;                    // 12 TAD sample time
// Configure the ADC reference sources.
ADCON3Lbits.REFSEL = 0;                      // AVdd as voltage reference
// Configure the integer of fractional output format.
ADCON1Hbits.FORM = 0;                        // integer format
// Select single-ended input configuration and unsigned output format.
ADMOD0Lbits.SIGN0 = 0;                       // AN0/RA0
ADMOD0Lbits.DIFF0 = 0;                       // AN0/RA0
ADMOD0Lbits.SIGN2 = 0;                       // AN2/RA2
// Enable and calibrate the module.
EnableAndCalibrate();                        // See Example 5-1
// Configure and enable early ADC interrupts.
ADCORE0Hbits.EISEL = 0;                      // early interrupt is generated 1 TADCORE clock prior
                                             // to when the data is ready
ADCORE1Hbits.EISEL = 3;                      // early interrupt is generated 4 TADCORE clocks prior
                                             // to when the data is ready
ADCON2Lbits.EIEN = 1;                        // enable early interrupts for ALL inputs
ADEIELbits.EIEN0 = 1;                        // enable interrupt for AN0
ADEIELbits.EIEN2 = 1;                        // enable interrupt for AN2
_ADCAN0IF = 0;                               // clear interrupt flag for AN0
_ADCAN0IE = 1;                               // enable interrupt for AN0
_ADCAN2IF = 0;                               // clear interrupt flag for AN2
_ADCAN2IE = 1;                               // enable interrupt for AN2
```

Example 5-5: Using Early Interrupts (Continued)

```
// Set same trigger source for all inputs to sample signals simultaneously.
ADTRIG0Lbits.TRGSRC0 = 13;                     // timer 2 for AN0
ADTRIG0Hbits.TRGSRC2 = 13;                     // timer 2 for AN2
// TIMER 2 INITIALIZATION (TIMER IS USED AS A TRIGGER SOURCE FOR ALL CHANNELS).
T2CONbits.TCS = 0;                             // clock from peripheral clock
T2CONbits.TCKPS = 0;                           // 1:1 prescale
PR2 = 0x8000;                                  // rollover every 0x8000 clocks
T2CONbits.TON = 1;                             // start timer to generate ADC triggers

while(1);

return 1;
}

// ADC AN0 ISR (DEDICATED CORE)
void __attribute__((interrupt, no_auto_psv)) _ADCAN0Interrupt(void)
{
dataAN0 = ADCBUF0;                             // read conversion result
_ADCAN0IF = 0;                                 // clear interrupt flag
}

// ADC AN2 ISR (SHARED CORE)
void __attribute__((interrupt, no_auto_psv)) _ADCAN2Interrupt(void)
{
dataAN2 = ADCBUF2;                             // read conversion result
_ADCAN2IF = 0;                                 // clear interrupt flag
}
```

5.6 Capacitive Voltage Divider (CVD)

The CVD allows measuring a capacitance connected to the ADC input. It can be used to detect an event of touch in touch sensor applications. The sensor capacitance is bigger when the sensor is touched. The code in Example 5-6 shows the CVD feature usage. In this example, CVD scans 2 inputs and Digital Comparator 0 detects when the capacitance is changed on these inputs (capacitive touch sensor is pressed).

Example 5-6: Using CVD

```
// These variables will keep the CVD result.
volatile short dataAN5CVDSampleA;
volatile short dataAN5CVDSampleB;

volatile short dataAN6CVDSampleA;
volatile short dataAN6CVDSampleB;
volatile short dataCVDResult;

// These variable will contain the CVD input number
// for the touched sensor.
volatile short comparatorChannelNumber;

int main()
{
// ADC INITIALIZATION
// Configure the I/O pins to be used as analog inputs.
// Only shared core inputs support CVD.
ANSELBbits.ANSB10 = 1;  TRISBbits.TRISB10 = 1;  // AN5/RB10 connected the shared dedicated core 0
ANSELBbits.ANSB1 = 1;   TRISBbits.TRISB1 = 1;   // AN6/RB1 connected the dedicated core 1

// Configure the common ADC clock.
ADCON3Hbits.CLKSEL = 2;                         // clock from FRC oscillator (7.3728 MHz)
ADCON3Hbits.CLKDIV = 0;                         // no clock divider (1:1)
// Configure the cores' ADC clock.
ADCORE0Hbits.ADCS = 0;                          // clock divider (1:2)
ADCORE1Hbits.ADCS = 0;                          // clock divider (1:2)
ADCON2Lbits.SHRADCS = 0;                        // clock divider (1:2) for shared core
// Configure the ADC reference sources.
ADCON3Lbits.REFSEL = 0;                         // AVdd as voltage reference
// Configure the integer of fractional output format.
ADCON1Hbits.FORM = 0;                           // integer format

// Select single-ended input configuration and unsigned output format.
ADMOD0Lbits.SIGN5 = 0;                          // AN5/RB10
ADMOD0Lbits.SIGN6 = 0;                          // AN6/RB1

// Add AN5 and AN6 to CVD scan list
ADCSSLbits.CSS5 = 1;                            // AN5
ADCSSLbits.CSS6 = 1;                            // AN6

// Disable all triggers for CVD inputs.
// Conversions for these channels will be triggered by CVD.
ADTRIG1Lbits.TRGSRC5 = 0;                       // AN5
ADTRIG1Hbits.TRGSRC6 = 0;                       // AN6

// Select CVD settle time Tsettle = 128*Tad/2 = 128*(2/FRC)/2 = 17uS
ADCON2Hbits.SHRSAMC = 128-2;

// Enable and calibrate the module.
EnableAndCalibrate();                           // See Example 5-1

// Configure and enable ADC interrupts.
ADIELbits.IE5 = 1;                              // enable interrupt for AN5
ADIELbits.IE6 = 1;                              // enable interrupt for AN6
```

Example 5-6: Using CVD (Continued)

```
_ADCAN5IF = 0;                                   // clear interrupt flag for AN5
_ADCAN5IE = 1;                                   // enable interrupt for AN5
_ADCAN6IF = 0;                                   // clear interrupt flag for AN6
_ADCAN6IE = 1;                                   // enable interrupt for AN6

// DIGITAL COMPARATOR INITIALIZATION.
// Set high threshold.
ADCMP0HI = 2048;
// Set the comparator event rule.
ADCMP0CONbits.HIHI = 1;                          // generate event when ADCVDDAT > ADCMP0HI
// Enable the ADC Digital Comparator interrupt.
ADCMP0CONbits.IE = 1;
_ADCMP0IF = 0;
_ADCMP0IE = 1;
// Enable the comparator.
ADCMP0CONbits.CMPEN = 1;

// Scan CVD inputs.
ADCON1Lbits.CVDEN = 1;

while(1);
return 1;
}

// ADC AN5 ISR
void __attribute__((interrupt, no_auto_psv)) _ADCAN5Interrupt(void)
{
static short sampleA = 0;

sampleA ^= 1;                                    // toggle SampleA flag
if(sampleA){
dataAN5CVDSampleA = ADCBUF5;                     // read CVD SampleA result for AN5
}else{
dataAN5CVDSampleB = ADCBUF5;                     // read CVD SampleB result for AN5
}

_ADCAN5IF = 0;                                   // clear interrupt flag
}

// ADC AN6 ISR
void __attribute__((interrupt, no_auto_psv)) _ADCAN6Interrupt(void)
{
static  short sampleA = 0;

sampleA ^= 1;                                    // toggle SampleA flag
if(sampleA){
dataAN6CVDSampleA = ADCBUF6;                     // read CVD SampleA result for AN6
}else{
dataAN6CVDSampleB = ADCBUF6;                     // read CVD SampleB result for AN6
}

_ADCAN6IF = 0;                                   // clear interrupt flag
}

// ADC COMPARATOR ISR
// If the conversion result for AN5 or AN6 is greater than ADCMP0HI,
// the interrupt is generated
void __attribute__((interrupt, no_auto_psv)) _ADCMP0Interrupt(void)
{
comparatorChannelNumber = ADCMP0CONbits.CHNL;    // read the channel number
// that generated interrupt
dataCVDResult = ADCVDDAT;                        // read SampleA - SampleB result
_ADCMP0IF = 0;                                   // clear interrupt flag
}
```

6.0 OPERATION DURING POWER-SAVING MODES

The power-saving modes, Sleep and Idle, are useful for reducing the conversion noise by minimizing the digital activity of the CPU, buses and other peripherals.

6.1 Sleep Mode

When a device enters Sleep mode, the system oscillator (FOSC) and all components that operate from it are halted. This includes the ADC when FOSC is selected for the clock source. When Sleep mode is invoked during a conversion with FOSC as the clock source, the conversion is aborted. The converter will not resume a partially completed conversion on exiting from Sleep mode. The ADC register contents are not affected by the device entering or leaving Sleep mode.

The ADC module can operate during Sleep mode if the ADC clock source is active during Sleep mode. The FRC oscillator is a logical choice for operation in Sleep mode. ADC operation during Sleep mode reduces the digital switching noise from the rest of the microcontroller during the conversion process.

If any of the ADC interrupts are enabled, the device will wake from Sleep mode when the ADC interrupt occurs. The 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 `PWRSAV` instruction that placed the device in Sleep mode.

For operation during Sleep mode, the application must use a conversion trigger source that ensures that the A/D conversion will take place in Sleep mode. For example, the external trigger pin option (TRGSRCn<4:0> = `11111`) can be used for performing sampling and conversion while the device is in Sleep mode.

6.2 Operation During Idle Mode

For the ADC, the Stop in Idle Mode bit, ADSIDL, in the ADCON1L register, specifies whether the ADC module will stop on Idle or continue on Idle. If ADSIDL = `0`, the ADC module will continue normal operation when the device enters Idle mode. If any of the ADC interrupts are enabled, the device will wake-up from Idle mode when the ADC interrupt occurs. The 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 `PWRSAV` instruction that placed the device in Idle mode.

If ADSIDL = `1`, the ADC module will stop in Idle mode. If the device enters Idle mode during a conversion, the conversion is aborted. The converter will not resume a partially completed conversion on exiting from Idle mode.

7.0 EFFECTS OF RESET

Following any Reset event, all the ADC control and status registers are reset to their default values with the control bits in a non-active state. This disables the ADC module and sets the analog input pins to Analog Input mode. Any conversion that was in progress will be terminated and the result will not be written to the result buffer. The values in the ADCBUFx registers are initialized to 0000h during a device Reset.

8.0 REGISTER MAP

Table 8-1: 12-Bit High-Speed, Multiple SARs A/D Converter Register Map[1]

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
ADCON1L	ADON	—	ADSIDL	—	CVDEN	—	—	—	NRE	—	—	—	—	—	—	—	0000
ADCON1H	r	r	r	r	r	r	r	r	FORM	SHRRES1	SHRRES0	r	r	r	r	r	0060
ADCON2L	REFCIE	REFERCIE	r	EIEN	r	SHREISEL2	SHREISEL1	SHREISEL0	—	SHRADCS6	SHRADCS5	SHRADCS4	SHRADCS3	SHRADCS2	SHRADCS1	SHRADCS0	0000
ADCON2H	REFRDY	REFERR	r	CVDCAP2	CVDCAP1	CVDCAP0	SHRSAMC9	SHRSAMC8	SHRSAMC7	SHRSAMC6	SHRSAMC5	SHRSAMC4	SHRSAMC3	SHRSAMC2	SHRSAMC1	SHRSAMC0	0000
ADCON3L	RFSEL2	RFSEL1	RFSEL0	SUSPEND	SUSPCIE	SUSPRDY	SHRSAMP	CNVRTCH	SWLCTRG	SWCTRG	CNVCHSEL5	CNVCHSEL4	CNVCHSEL3	CNVCHSEL2	CNVCHSEL1	CNVCHSEL0	0000
ADCON3H	CLKSEL1	CLKSEL0	CLKDIV5	CLKDIV4	CLKDIV3	CLKDIV2	CLKDIV1	CLKDIV0	SHREN	C6EN	C5EN	C4EN	C3EN	C2EN	C1EN	C0EN	0000
ADCON4L	—	r	r	r	r	r	r	r	—	SAMC6EN	SAMC5EN	SAMC4EN	SAMC3EN	SAMC2EN	SAMC1EN	SAMC0EN	0000
ADCON4H	—	—	C6CHS1	C6CHS0	C5CHS1	C5CHS0	C4CHS1	C4CHS0	C3CHS1	C3CHS0	C2CHS1	C2CHS0	C1CHS1	C1CHS0	C0CHS1	C0CHS0	0000
ADMOD0L	DIFF7	SIGN7	DIFF6	SIGN6	DIFF5	SIGN5	DIFF4	SIGN4	DIFF3	SIGN3	DIFF2	SIGN2	DIFF1	SIGN1	DIFF0	SIGN0	0000
ADMOD0H	DIFF15	SIGN15	DIFF14	SIGN14	DIFF13	SIGN13	DIFF12	SIGN12	DIFF11	SIGN11	DIFF10	SIGN10	DIFF9	SIGN9	DIFF8	SIGN8	0000
ADMOD1L	DIFF23	SIGN23	DIFF22	SIGN22	DIFF21	SIGN21	DIFF20	SIGN20	DIFF19	SIGN19	DIFF18	SIGN18	DIFF17	SIGN17	DIFF16	SIGN16	0000
ADMOD1H	DIFF31	SIGN31	DIFF30	SIGN30	DIFF29	SIGN29	DIFF28	SIGN28	DIFF27	SIGN27	DIFF26	SIGN26	DIFF25	SIGN25	DIFF24	SIGN24	0000
ADIEL	IE<15:0>																0000
ADIEH	IE<31:16>																0000
ADCSSL	CSS<15:0>																0000
ADCSSH	CSS<31:16>																0000
ADSTATL	AN15RDY	AN14RDY	AN13RDY	AN12RDY	AN11RDY	AN10RDY	AN9RDY	AN8RDY	AN7RDY	AN6RDY	AN5RDY	AN4RDY	AN3RDY	AN2RDY	AN1RDY	AN0RDY	0000
ADSTATH	AN31RDY	AN30RDY	AN29RDY	AN28RDY	AN27RDY	AN26RDY	AN25RDY	AN24RDY	AN23RDY	AN22RDY	AN21RDY	AN20RDY	AN19RDY	AN18RDY	AN17RDY	AN16RDY	0000
ADCMP0ENL	CMPEN<15:0>																0000
ADCMP0ENH	CMPEN<31:16>																0000
ADCMP0LO	ADC CMPLO Register																0000
ADCMP0HI	ADC CMPHI Register																0000
ADCMP1ENL	CMPEN<15:0>																0000
ADCMP1ENH	CMPEN<31:16>																0000
ADCMP1LO	ADC CMPLO Register																0000
ADCMP1HI	ADC CMPHI Register																0000
ADCMP2ENL	CMPEN<15:0>																0000
ADCMP2ENH	CMPEN<31:16>																0000
ADCMP2LO	ADC CMPLO Register																0000
ADCMP2HI	ADC CMPHI Register																0000
ADCMP3ENL	CMPEN<15:0>																0000
ADCMP3ENH	CMPEN<31:16>																0000
ADCMP3LO	ADC CMPLO Register																0000
ADCMP3HI	ADC CMPHI Register																0000

Legend: — = unimplemented, read as '0'; r = reserved, must be written as '0'. Reset values are shown in hexadecimal.

Note 1: Not all registers are implemented in all devices. Refer to the device data sheet for device-specific register maps and bit implementation.

Table 8-1: 12-Bit High-Speed, Multiple SARs A/D Converter Register Map[1] (Continued)

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
ADCMP4ENL	CMPEN<15:0>																0000
ADCMP4ENH	CMPEN<31:16>																0000
ADCMP4LO	ADC CMPLO Register																0000
ADCMP4HI	ADC CMPHI Register																0000
ADCMP5ENL	CMPEN<15:0>																0000
ADCMP5ENH	CMPEN<31:16>																0000
ADCMP5LO	ADC CMPLO Register																0000
ADCMP5HI	ADC CMPHI Register																0000
ADFL0DAT	ADC FLDATA Register																0000
ADFL0CON	FLEN	MODE1	MODE0	OVRSAM2	OVRSAM1	OVRSAM0	IE	RDY	—	—	—	FLCHSEL4	FLCHSEL3	FLCHSEL2	FLCHSEL1	FLCHSEL0	0000
ADFL1DAT	ADC FLDATA Register																0000
ADFL1CON	FLEN	MODE1	MODE0	OVRSAM2	OVRSAM1	OVRSAM0	IE	RDY	—	—	—	FLCHSEL4	FLCHSEL3	FLCHSEL2	FLCHSEL1	FLCHSEL0	0000
ADFL2DAT	ADC FLDATA Register																0000
ADFL2CON	FLEN	MODE1	MODE0	OVRSAM2	OVRSAM1	OVRSAM0	IE	RDY	—	—	—	FLCHSEL4	FLCHSEL3	FLCHSEL2	FLCHSEL1	FLCHSEL0	0000
ADFL3DAT	ADC FLDATA Register																0000
ADFL3CON	FLEN	MODE1	MODE0	OVRSAM2	OVRSAM1	OVRSAM0	IE	RDY	—	—	—	FLCHSEL4	FLCHSEL3	FLCHSEL2	FLCHSEL1	FLCHSEL0	0000
ADFL4DAT	ADC FLDATA Register																0000
ADFL4CON	FLEN	MODE1	MODE0	OVRSAM2	OVRSAM1	OVRSAM0	IE	RDY	—	—	—	FLCHSEL4	FLCHSEL3	FLCHSEL2	FLCHSEL1	FLCHSEL0	0000
ADFL5DAT	ADC FLDATA Register																0000
ADFL5CON	FLEN	MODE1	MODE0	OVRSAM2	OVRSAM1	OVRSAM0	IE	RDY	—	—	—	FLCHSEL4	FLCHSEL3	FLCHSEL2	FLCHSEL1	FLCHSEL0	0000
ADTRIG0L	—	—	—	TRGSRC14	TRGSRC13	TRGSRC12	TRGSRC11	TRGSRC10	—	—	—	TRGSRC04	TRGSRC03	TRGSRC02	TRGSRC01	TRGSRC00	0000
ADTRIG0H	—	—	—	TRGSRC34	TRGSRC33	TRGSRC32	TRGSRC31	TRGSRC30	—	—	—	TRGSRC24	TRGSRC23	TRGSRC22	TRGSRC21	TRGSRC20	0000
ADTRIG1L	—	—	—	TRGSRC54	TRGSRC53	TRGSRC52	TRGSRC51	TRGSRC50	—	—	—	TRGSRC44	TRGSRC43	TRGSRC42	TRGSRC41	TRGSRC40	0000
ADTRIG1H	—	—	—	TRGSRC74	TRGSRC73	TRGSRC72	TRGSRC71	TRGSRC70	—	—	—	TRGSRC64	TRGSRC63	TRGSRC62	TRGSRC61	TRGSRC60	0000
ADTRIG2L	—	—	—	TRGSRC94	TRGSRC93	TRGSRC92	TRGSRC91	TRGSRC90	—	—	—	TRGSRC84	TRGSRC83	TRGSRC82	TRGSRC81	TRGSRC80	0000
ADTRIG2H	—	—	—	TRGSRC114	TRGSRC113	TRGSRC112	TRGSRC111	TRGSRC110	—	—	—	TRGSRC104	TRGSRC103	TRGSRC102	TRGSRC101	TRGSRC100	0000
ADTRIG3L	—	—	—	TRGSRC134	TRGSRC133	TRGSRC132	TRGSRC131	TRGSRC130	—	—	—	TRGSRC124	TRGSRC123	TRGSRC122	TRGSRC121	TRGSRC120	0000
ADTRIG3H	—	—	—	TRGSRC154	TRGSRC153	TRGSRC152	TRGSRC151	TRGSRC150	—	—	—	TRGSRC144	TRGSRC143	TRGSRC142	TRGSRC141	TRGSRC140	0000
ADTRIG4L	—	—	—	TRGSRC174	TRGSRC173	TRGSRC172	TRGSRC171	TRGSRC170	—	—	—	TRGSRC164	TRGSRC163	TRGSRC162	TRGSRC161	TRGSRC160	0000
ADTRIG4H	—	—	—	TRGSRC194	TRGSRC193	TRGSRC192	TRGSRC191	TRGSRC190	—	—	—	TRGSRC184	TRGSRC183	TRGSRC182	TRGSRC181	TRGSRC180	0000
ADTRIG5L	—	—	—	TRGSRC214	TRGSRC213	TRGSRC212	TRGSRC211	TRGSRC210	—	—	—	TRGSRC204	TRGSRC203	TRGSRC202	TRGSRC201	TRGSRC200	0000
ADTRIG5H	—	—	—	TRGSRC234	TRGSRC233	TRGSRC232	TRGSRC231	TRGSRC230	—	—	—	TRGSRC224	TRGSRC223	TRGSRC222	TRGSRC221	TRGSRC220	0000
ADTRIG6L	—	—	—	TRGSRC254	TRGSRC253	TRGSRC252	TRGSRC251	TRGSRC250	—	—	—	TRGSRC244	TRGSRC243	TRGSRC242	TRGSRC241	TRGSRC240	0000
ADTRIG6H	—	—	—	TRGSRC274	TRGSRC273	TRGSRC272	TRGSRC271	TRGSRC270	—	—	—	TRGSRC264	TRGSRC263	TRGSRC262	TRGSRC261	TRGSRC260	0000

Legend: — = unimplemented, read as ‘0’; r = reserved, must be written as ‘0’. Reset values are shown in hexadecimal.

Note 1: Not all registers are implemented in all devices. Refer to the device data sheet for device-specific register maps and bit implementation.

Table 8-1: 12-Bit High-Speed, Multiple SARs A/D Converter Register Map[1] (Continued)

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
ADTRIG7L	—	—	—	TRGSRC294	TRGSRC293	TRGSRC292	TRGSRC291	TRGSRC290	—	—	—	TRGSRC284	TRGSRC283	TRGSRC282	TRGSRC281	TRGSRC280	0000
ADTRIG7H	—	—	—	TRGSRC314	TRGSRC313	TRGSRC312	TRGSRC311	TRGSRC310	—	—	—	TRGSRC304	TRGSRC303	TRGSRC302	TRGSRC301	TRGSRC300	0000
ADCMP0CON	—	—	—	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0	CMPEN	IE	STAT	BTWN	HIHI	HILO	LOHI	LOLO	0000
ADCVDDAT	ADCVDDAT<15:0>																0000
ADCMP1CON	—	—	—	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0	CMPEN	IE	STAT	BTWN	HIHI	HILO	LOHI	LOLO	0000
ADCMP2CON	—	—	—	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0	CMPEN	IE	STAT	BTWN	HIHI	HILO	LOHI	LOLO	0000
ADCMP3CON	—	—	—	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0	CMPEN	IE	STAT	BTWN	HIHI	HILO	LOHI	LOLO	0000
ADCMP4CON	—	—	—	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0	CMPEN	IE	STAT	BTWN	HIHI	HILO	LOHI	LOLO	0000
ADCMP5CON	—	—	—	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0	CMPEN	IE	STAT	BTWN	HIHI	HILO	LOHI	LOLO	0000
ADLVLTRGL	LVLEN<15:0>																0000
ADLVLTRGH	LVLEN<31:16>																0000
ADCORE0L	—	—	—	—	—	—	—	SAMC<9:0>									0000
ADCORE0H	—	—	—	—	—	—	RES1	RES0	—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0300
ADCORE1L	—	—	—	—	—	—	—	SAMC<9:0>									0000
ADCORE1H	—	—	—	EISEL2	EISEL1	EISEL0	RES1	RES0	—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0300
ADCORE2L	—	—	—	—	—	—	—	SAMC<9:0>									0000
ADCORE2H	—	—	—	EISEL2	EISEL1	EISEL0	RES1	RES0	—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0300
ADCORE3L	—	—	—	—	—	—	—	SAMC<9:0>									0000
ADCORE3H	—	—	—	EISEL2	EISEL1	EISEL0	RES1	RES0	—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0300
ADCORE4L	—	—	—	—	—	—	—	SAMC<9:0>									0000
ADCORE4H	—	—	—	EISEL2	EISEL1	EISEL0	RES1	RES0	—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0300
ADCORE5L	—	—	—	—	—	—	—	SAMC<9:0>									0000
ADCORE5H	—	—	—	EISEL2	EISEL1	EISEL0	RES1	RES0	—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0300
ADCORE6L	—	—	—	—	—	—	—	SAMC<9:0>									0000
ADCORE6H	—	—	—	EISEL2	EISEL1	EISEL0	RES1	RES0	—	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0300
ADEIEL	EIE<15:0>																0000
ADEIEH	EIE<31:16>																0000
ADEISTATL	EISTAT<15:0>																0000
ADEISTATH	EISTAT<31:16>																0000
ADCON5L	SHRRDY	C6RDY	C5RDY	C4RDY	C3RDY	C2RDY	C1RDY	C0RDY	SHRPWR	C6PWR	C5PWR	C4PWR	C3PWR	C2PWR	C1PWR	C0PWR	0000
ADCON5H	—	—	—	—	WARMTIME3	WARMTIME2	WARMTIME1	WARMTIME0	SHRCIE	C6CIE	C5CIE	C4CIE	C3CIE	C2CIE	C1CIE	C0CIE	0000
ADCAL0L	CAL1RDY	—	—	—	r	CAL1DIFF	CAL1EN	CAL1RUN	CAL0RDY	—	—	—	r	CAL0DIFF	CAL0EN	CAL0RUN	0000
ADCAL0H	CAL3RDY	—	—	—	r	CAL3DIFF	CAL3EN	CAL3RUN	CAL2RDY	—	—	—	r	CAL2DIFF	CAL2EN	CAL2RUN	0000
ADCAL1L	CAL5RDY	—	—	—	r	CAL5DIFF	CAL5EN	CAL5RUN	CAL4RDY	—	—	—	r	CAL4DIFF	CAL4EN	CAL4RUN	0000
ADCAL1H	CSHRRDY	—	—	—	r	CSHRDIFF	CSHREN	CSHRRUN	CAL6RDY	—	—	—	r	CAL6DIFF	CAL6EN	CAL6RUN	0000

Legend: — = unimplemented, read as ‘0’; r = reserved, must be written as ‘0’. Reset values are shown in hexadecimal.

Note 1: Not all registers are implemented in all devices. Refer to the device data sheet for device-specific register maps and bit implementation.

Table 8-1: 12-Bit High-Speed, Multiple SARs A/D Converter Register Map[1] (Continued)

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
ADCBUF0	DATA0<15:0>																0000
ADCBUF1	DATA1<15:0>																0000
ADCBUF2	DATA2<15:0>																0000
ADCBUF3	DATA3<15:0>																0000
ADCBUF4	DATA4<15:0>																0000
ADCBUF5	DATA5<15:0>																0000
ADCBUF6	DATA6<15:0>																0000
ADCBUF7	DATA7<15:0>																0000
ADCBUF8	DATA8<15:0>																0000
ADCBUF9	DATA9<15:0>																0000
ADCBUF10	DATA10<15:0>																0000
ADCBUF11	DATA11<15:0>																0000
ADCBUF12	DATA12<15:0>																0000
ADCBUF13	DATA13<15:0>																0000
ADCBUF14	DATA14<15:0>																0000
ADCBUF15	DATA15<15:0>																0000
ADCBUF16	DATA16<15:0>																0000
ADCBUF17	DATA17<15:0>																0000
ADCBUF18	DATA18<15:0>																0000
ADCBUF19	DATA19<15:0>																0000
ADCBUF20	DATA20<15:0>																0000
ADCBUF21	DATA21<15:0>																0000
ADCBUF22	DATA22<15:0>																0000
ADCBUF23	DATA23<15:0>																0000
ADCBUF24	DATA24<15:0>																0000
ADCBUF25	DATA25<15:0>																0000
ADCBUF26	DATA26<15:0>																0000
ADCBUF27	DATA27<15:0>																0000
ADCBUF28	DATA28<15:0>																0000
ADCBUF29	DATA29<15:0>																0000
ADCBUF30	DATA30<15:0>																0000
ADCBUF31	DATA31<15:0>																0000

Legend: — = unimplemented, read as '0'; r = reserved, must be written as '0'. Reset values are shown in hexadecimal.

Note 1: Not all registers are implemented in all devices. Refer to the device data sheet for device-specific register maps and bit implementation.

9.0 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 dsPIC33/PIC24 device families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the 12-Bit High-Speed, Multiple SARs A/D Converter (ADC) module are:

Title	Application Note #
No related application notes at this time.	

Note:	Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the dsPIC33/PIC24 families of devices.

10.0 REVISION HISTORY

Revision A (November 2014)

This is the initial revision of this document.

Revision B (May 2015)

Updated **Section 4.4 "Sampling and Conversion Timing"** and added Equation 4-1.

Revision C (September 2015)

Changed all occurrences of Differential Inputs to Pseudodifferential Inputs. Removed all references to CALxSKIP bits.

Updated Figure 1-2, Figure 4-5, Figure 4-6 and Figure 4-10.

Updated Register 2-1, Register 2-2, Register 2-3, Register 2-4, Register 2-7, Register 2-27, Register 2-28, Register 2-29, Register 2-30 and Register 2-31.

Updated Table 4-1 and Table 8-1.

This revision also includes numerous grammatical corrections throughout the document.

Revision D (January 2016)

Updated Figure 4-12 and Figure 4-13.

Updated **Section 4.6.3 "Selecting Analog Input for Dedicated ADC Core"** and **Section 4.13.3 "Early Interrupts"**, and added **Section 5.5 "Early Interrupts"**.

Revision E (January 2017)

Updated Register 2-5, Register 2-7, Register 2-27, Register 2-31 and Register 2-35.

Removed **Asynchronous** from the Figure 4-2 title.

Removed Figure 4-3 and added new paragraph before Figure 4-4.

Added additional content to Note in **Section 4.6.3 "Selecting Analog Input for Dedicated ADC Core"**.

Updated the steps to enable the ADC module in **Section 4.7 "Enabling the ADC"**.

Updated Table 8-1.

Revision F (June 2017)

Updated **Section 1.0 "Introduction"**, **Section 2.1 "Control Registers"**, **Section 2.2 "Data Registers"** and **Section 4.4 "Sampling and Conversion Timing"**. Added **Section 4.14 "Capacitive Voltage Divider (CVD)"** and **Section 5.6 "Capacitive Voltage Divider (CVD)"**.

Updated Register 2-1 and Register 2-4. Added Register 2-36 and Register 2-37.

Added Figure 4-14.

Added Equation 4-2.

Added Example 5-6.

Updated Table 8-1.

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: 978-1-5224-1830-6

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

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

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

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

EUROPE

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

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

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

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

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

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

Germany - Karlsruhe
Tel: 49-721-625370

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

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

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

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

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

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

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

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

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

11/07/16