Section 16. Analog-to-Digital 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 dsPIC33E/PIC24E devices.
>
> Please consult the note at the beginning of the **"Analog-to-Digital Converter (ADC)"** chapter in the current 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

16.1   INTRODUCTION

This document describes the features and associated operational modes of the Successive Approximation (SAR) Analog-to-Digital Converter (ADC) modules available on the dsPIC33E/PIC24E families of devices.

This ADC module can be configured by the user application to function as a 10-bit, 4-channel ADC or a 12-bit, single channel ADC.

On devices with Direct Memory Access (DMA), this ADC module can be configured to use DMA or use a dedicated, 16-word memory mapped buffer instead of DMA.

An ADC module block diagram for devices without op amps is provided in Figure 16-1. The ADC module block diagram for devices with op amps is provided in Figure 16-2.

> **Note:** Op amps are not available on all devices. Refer to the **"Op Amp/Comparator"** chapter in the specific device data sheet for availability.

The following key features are common to all dsPIC33E/PIC24E devices:

- SAR conversion
- Up to 1.1 Msps conversion speed in 10-bit mode
- Up to 500 ksps conversion speed in 12-bit mode
- Up to 32 analog input pins
- External voltage reference input pins
- Four unipolar, differential Sample-and-Hold (S&H) amplifiers
- Simultaneous sampling of up to four analog input pins
- Automatic Channel Scanning mode
- Selectable conversion trigger source
- Up to 16-word conversion result buffer
- Operation during CPU Sleep and Idle modes

Additional features are available on select dsPIC33E/PIC24E devices:

- Connections for up to three internal op amps (not available on all devices)
- Connections to the Charge Time Measurement Unit (CTMU) and temperature measurement diode (not available on all devices)
- Channel selection and triggering can be controlled by the Peripheral Trigger Generator (PTG) (not available on all devices)
- Selectable Buffer Fill modes (not available on all devices)
- DMA support, including Peripheral Indirect Addressing (PIA) (not available on all devices)

> **Note:** Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet to determine the availability of these additional features.

Depending on the device variant, the ADC module may have up to 49 analog input pins, designated AN0-AN48, and four op amp outputs, designated OA1-OA3 and OA5. These analog inputs and op amp outputs are connected by multiplexers to four S&H amplifiers, designated CH0-CH3. The analog input multiplexers have two sets of control bits, designated as MUXA (CHySA/CHyNA) and MUXB (CHySB/CHyNB). These control bits select a particular analog input for conversion. The MUXA and MUXB control bits can alternatively select the analog input for conversion. Unipolar differential conversions are possible on all channels using certain input pins.

Section 16. Analog-to-Digital Converter (ADC)

Channel Scanning mode can be enabled for the CH0 S&H amplifier. Any subset of the analog inputs or op amp outputs (based on availability) can be selected by the user application. The selected inputs are converted in ascending order using CH0.

The ADC module supports simultaneous sampling using multiple S&H channels to sample the inputs at the same time, and then performs the conversion for each channel sequentially. By default, the multiple channels are sampled and converted sequentially.

For devices with DMA and with the ADC DMA Enable bit (ADDMAEN) set, the ADC module is connected to a single-word result buffer. However, multiple conversion results can be stored in a DMA RAM buffer with no CPU overhead when DMA is used with the ADC module. Each conversion result is converted to one of four 16-bit output formats when it is read from the buffer.

For devices without DMA, and for devices with DMA that have the ADC DMA Enable bit (ADDMAEN) clear, the ADC module is connected to a 16-word result buffer. The ADC result is available in four different numerical formats (see Figure 16-14).

> **Note 1:** A 'y' is used with MUXA and MUXB control bits to specify the S&H channel numbers (y = 0 or 123). Refer to **Section 16.6.2 "Alternate Input Selection Mode"** for more details.
>
> **2:** Depending on a particular device pinout, the ADC can have up to 49 analog input pins, designated AN0 through AN48, and four op amp outputs, designated OA1-OA3 and OA5. In addition, there are two analog input pins for external voltage reference connections (VREF+, VREF-). These analog inputs are shared with op amp inputs and outputs, comparator inputs, and external voltage references. When op amp/comparator functionality is enabled or an external voltage reference is used, the analog input that shares that pin is no longer available. The actual number of analog input pins and external voltage reference input configuration depends on the specific device. For more details, refer to the specific device data sheet.

Figure 16-1: **ADC Block Diagram for dsPIC33E/PIC24E Devices without Op Amps**

Note 1: V_{REF}+, V_{REF}- inputs can be multiplexed with other analog inputs. For more details, refer to the specific device data sheet.
2: Channels 1, 2 and 3 are not applicable for the 12-bit mode of operation.
3: These buffers are unavailable if DMA is available and the ADDMAEN bit is set.
4: These bits can be updated with Step commands from the PTG module (not available on all devices). Refer to the **"Peripheral Trigger Generator (PTG) Module"** chapter in the specific device data sheet for availability.

Figure 16-2:   ADC Module Block Diagram with Connection Options for ANx Pins and Op Amps

This diagram depicts all of the available ADC connection options to the four S&H amplifiers, which are designated: CH0, CH1, CH2 and CH3.

The ANx analog pins or op amp outputs are connected to the CH0-CH3 amplifiers through the multiplexers, controlled by the SFR control bits, CH0Sx, CH0Nx, CH123Sx and CH123Nx.
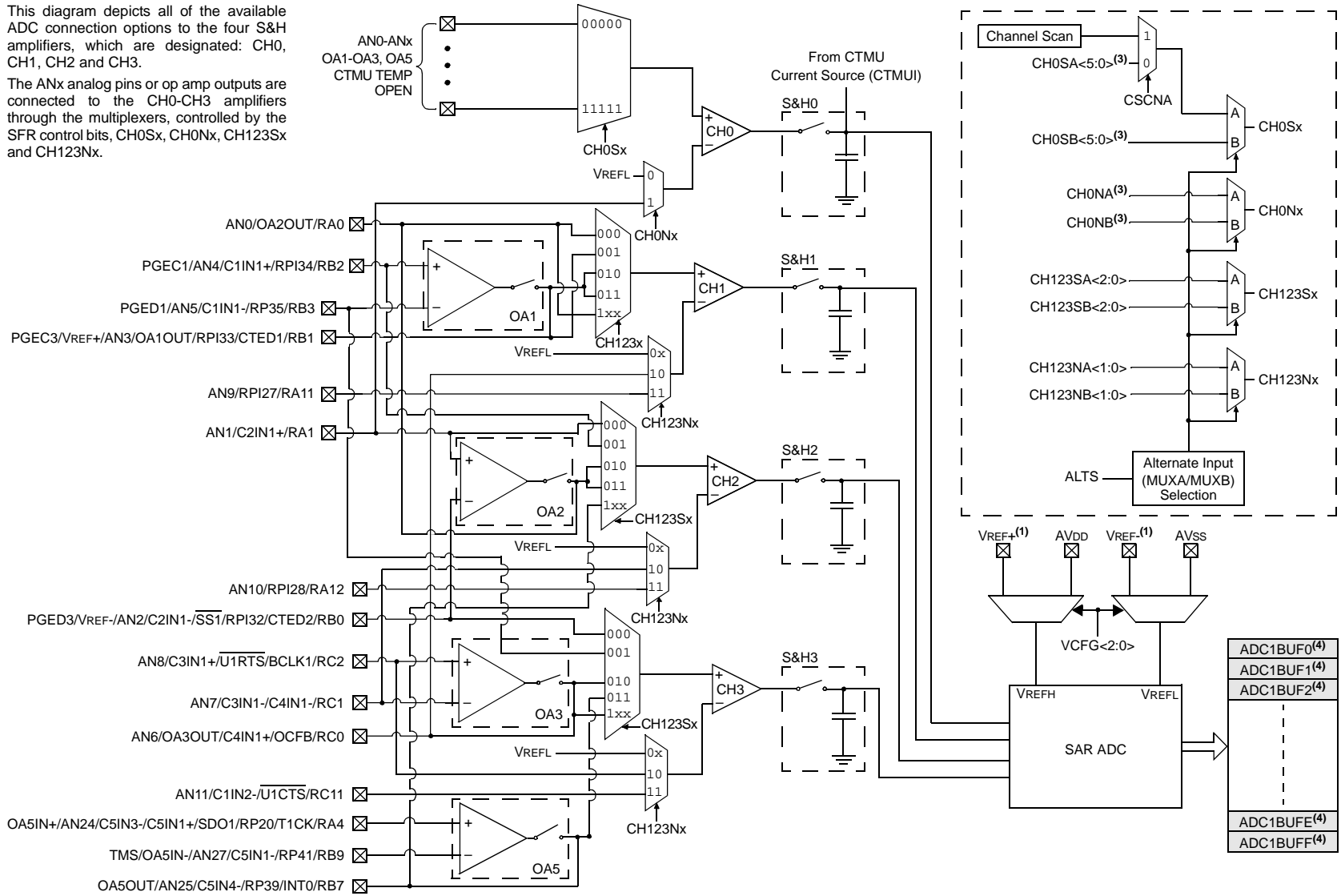
Note 1: VREF+, VREF- inputs can be multiplexed with other analog inputs.
2: Channels 1, 2 and 3 are not applicable for the 12-bit mode of operation.
3: These bits can be updated with Step commands from the PTG module. For more information, refer to the **"Peripheral Trigger Generator (PTG)"** chapter in the specific device data sheet.
4: When ADDMAEN (ADxCON4<8>) = 1 enabling DMA, only ADCxBUF0 is used.

16.2   CONTROL REGISTERS

The ADC module has nine Control and Status registers:

- **ADxCON1: ADCx Control Register 1**
- **ADxCON2: ADCx Control Register 2**
- **ADxCON3: ADCx Control Register 3**
- **ADxCON4: ADCx Control Register 4**
- **ADxCHS123: ADCx Input Channel 1, 2, 3 Select Register**
- **ADxCHS0: ADCx Input Channel 0 Select Register**
- **ADxCSSH: ADCx Input Scan Select Register High**
- **ADxCSSL: ADCx Input Scan Select Register Low**
- **ANSELy: Analog/Digital Pin Selection Register**

The ADxCON1, ADxCON2 and ADxCON3 registers control the operation of the ADC module. For devices with DMA, the ADxCON4 register sets up the number of conversion results stored in a DMA buffer for each analog input in the Scatter/Gather mode. The ADxCHS123 and ADxCHS0 registers select the input pins to be connected to the S&H amplifiers. The ADCSSH/L registers select inputs to be sequentially scanned. The ANSELy register specifies the input collection of device pins used as analog inputs. Along with the Data Direction register (TRISx) in the Parallel I/O Port module, ANSELy registers control the operation of the ADC pins.

16.2.1   ADC Result Buffer

For devices with DMA and with the ADC DMA Enable bit (ADDMAEN) set, the ADC module contains a single-word result buffer, ADC1BUF0. For devices without DMA, and for devices with DMA that have the ADC DMA Enable bit (ADDMAEN) clear, the ADC module contains a 16-word dual port RAM to buffer the results. The 16 buffer locations are referred to as ADC1BUF0, ADC1BUF1, ADC1BUF2, ..., ADC1BUFE and ADC1BUFF.

> **Note:**   After a device Reset, the ADC Buffer register(s) will contain unknown data.

Register 16-1:   ADxCON1: ADCx Control Register 1

R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
ADON	—	ADSIDL	ADDMABM[1]	—	AD12B[1]	FORM<1:0>	
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, HC, HS	R/C-0, HC, HS
SSRC<2:0>			SSRCG	SIMSAM	ASAM[2]	SAMP	DONE[2]
bit 7							bit 0

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

bit 15   **ADON:** ADC Operating Mode bit

1 = ADC module is operating
0 = ADC 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   **ADDMABM:** DMA Buffer Build Mode bit[1]

1 = DMA buffers are written in the order of conversion; the module provides an address to the DMA channel that is the same as the address used for the non-DMA stand-alone buffer
0 = DMA buffers are written in Scatter/Gather mode; the module provides a Scatter/Gather mode address to the DMA channel, based on the index of the analog input and the size of the DMA buffer

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

bit 10   **AD12B:** ADC 10-Bit or 12-Bit Operation Mode bit[1]

1 = 12-bit, 1-channel ADC operation
0 = 10-bit, 4-channel ADC operation

bit 9-8   **FORM<1:0>:** Data Output Format bits

For 10-Bit Operation:
11 = Signed fractional (D$_{OUT}$ = sddd dddd dd00 0000, where s = sign, d = data)
10 = Fractional (D$_{OUT}$ = dddd dddd dd00 0000)
01 = Signed integer (D$_{OUT}$ = ssss sssd dddd dddd, where s = sign, d = data)
00 = Integer (D$_{OUT}$ = 0000 00dd dddd dddd)

For 12-Bit Operation:
11 = Signed fractional (D$_{OUT}$ = sddd dddd dddd 0000, where s = sign, d = data)
10 = Fractional (D$_{OUT}$ = dddd dddd dddd 0000)
01 = Signed Integer (D$_{OUT}$ = ssss sddd dddd dddd, where s = sign, d = data)
00 = Integer (D$_{OUT}$ = 0000 dddd dddd dddd)

bit 7-5   **SSRC<2:0>:** Sample Clock Source Select bits

These settings vary by device. Refer to the ADxCON1 register in the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

bit 4   **SSRCG:** Sample Clock Source Group bit

These settings vary by device. Refer to the ADxCON1 register in the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

Note 1: This bit or setting is not available on all devices. Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

2: Do not clear the DONE bit in software if ADC Sample Auto-Start is enabled (ASAM = 1).

Register 16-1:   ADxCON1: ADCx Control Register 1 (Continued)

bit 3   **SIMSAM:** Simultaneous Sample Select bit (only applicable when CHPS<1:0> = `01` or `1x`)

In 12-bit mode (AD21B = `1`), SIMSAM is unimplemented and is read as '`0`'.

`1` = Samples CH0, CH1, CH2, CH3 simultaneously (when CHPS<1:0> = `1x`); or samples CH0 and CH1 simultaneously (when CHPS<1:0> = `01`)
`0` = Samples multiple channels individually in sequence

bit 2   **ASAM:** ADC Sample Auto-Start bit[2]

`1` = Sampling begins immediately after last conversion; SAMP bit is auto-set
`0` = Sampling begins when SAMP bit is set

bit 1   **SAMP:** ADC Sample Enable bit

`1` = ADC Sample-and-Hold amplifiers are sampling
`0` = ADC Sample-and-Hold amplifiers are holding

If ASAM = `0`, software can write '1' to begin sampling. Automatically set by hardware if ASAM = `1`.
If SSRC<2:0> = `000` and SSRCG = `0`, software can write '0' to end sampling and start conversion. If SSRC<2:0> $\neq$ `000`, automatically cleared by hardware to end sampling and start conversion.

bit 0   **DONE:** ADC Conversion Status bit[2]

`1` = ADC conversion cycle has completed
`0` = ADC conversion has not started or is in progress

Automatically set by hardware when Analog-to-Digital conversion is complete. Software can write '0' to clear the DONE status (software not allowed to write '1'). Clearing this bit does NOT affect any operation in progress. Automatically cleared by hardware at the start of a new conversion.

Note 1:   This bit or setting is not available on all devices. Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

2:   Do not clear the DONE bit in software if ADC Sample Auto-Start is enabled (ASAM = `1`).

Register 16-2:   ADxCON2: ADCx Control Register 2

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
VCFG<2:0>			—	—	CSCNA	CHPS<1:0>	
bit 15							bit 8

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	SMPI<4:0>[1,2,3]					BUFM	ALTS
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   **VCFG<2:0>:** ADC Converter Voltage Reference Configuration bits

	V$_{REFH}$	V$_{REFL}$
`000`	AV$_{DD}$	AV$_{SS}$
`001`	External V$_{REF}$+	AV$_{SS}$
`010`	AV$_{DD}$	External V$_{REF}$-
`011`	External V$_{REF}$+	External V$_{REF}$-
`1xx`	AV$_{DD}$	AV$_{SS}$

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

bit 10   **CSCNA:** Input Scan Select bit

`1` = Scans inputs for CH0+ during Sample A bit
`0` = Does not scan inputs

bit 9-8   **CHPS<1:0>:** Channel Select bits

When AD12B = `1`, CHPS<1:0> is: U-0 (Unimplemented: Read as '`0`')

`1x` = Converts CH0, CH1, CH2 and CH3
`01` = Converts CH0 and CH1
`00` = Converts CH0

bit 7   **BUFS:** Buffer Fill Status bit (only valid when BUFM = 1)

`1` = ADC is currently filling the second half of the buffer; the user application should access data in the first half of the buffer
`0` = ADC is currently filling the first half of the buffer; the user application should access data in the second half of the buffer

Note 1:   For devices with DMA and with the ADC DMA Enable bit (ADDMAEN) set, the SMPI<4:0> bits are referred to as the "Increment Rate for DMA Address Select bits".

2:   For devices without DMA, and for devices with DMA that have the ADC DMA Enable bit (ADDMAEN) clear, the SMPI<4:0> bits are referred to as the "Number of Samples per Interrupt Select bits".

3:   For ADC2, the sample and conversion operation bits are only four bits (SMPI<3:0>), which provide an ADC interrupt (for devices without DMA), and incrementation of the DMA address (for devices with DMA) at the completion of up to16 sample and conversion operations.

Register 16-2: ADxCON2: ADCx Control Register 2 (Continued)

bit 6-2   **SMPI<4:0>:** Sample and Conversion Operation bits[1,2,3]

For Devices with DMA and with the ADC DMA Enable bit (ADDMAEN) Set:

x1111 = Increments the DMA address after completion of every 16th sample/conversion operation
x1110 = Increments the DMA address after completion of every 15th sample/conversion operation
- •
- •
- •

x0001 = Increments the DMA address after completion of every 2nd sample/conversion operation
x0000 = Increments the DMA address after completion of every sample/conversion operation

For Devices without DMA and for Devices with DMA that have the ADC DMA Enable bit (ADDMAEN) Clear:

11111 = ADC interrupt is generated at the completion of every 32nd sample/conversion operation
11110 = ADC interrupt is generated at the completion of every 31st sample/conversion operation
- •
- •
- •

00001 = ADC interrupt is generated at the completion of every 2nd sample/conversion operation
00000 = ADC interrupt is generated at the completion of every sample/conversion operation

bit 1   **BUFM:** Buffer Fill Mode Select bit

1 = Starts buffer filling the first half of the buffer on the first interrupt and the second half of the buffer on the next interrupt
0 = Always starts filling the buffer from the Start address

bit 0   **ALTS:** Alternate Input Sample Mode Select bit

1 = Uses channel input selects for Sample MUXA on first sample and Sample MUXB on next sample
0 = Always uses channel input selects for Sample MUXA

Note 1: For devices with DMA and with the ADC DMA Enable bit (ADDMAEN) set, the SMPI<4:0> bits are referred to as the "Increment Rate for DMA Address Select bits".

2: For devices without DMA, and for devices with DMA that have the ADC DMA Enable bit (ADDMAEN) clear, the SMPI<4:0> bits are referred to as the "Number of Samples per Interrupt Select bits".

3: For ADC2, the sample and conversion operation bits are only four bits (SMPI<3:0>), which provide an ADC interrupt (for devices without DMA), and incrementation of the DMA address (for devices with DMA) at the completion of up to16 sample and conversion operations.

Register 16-3:   ADxCON3: ADCx Control Register 3

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	—	SAMC<4:0>[1,2]				
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
ADCS<7:0>[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   **ADRC:** ADC Conversion Clock Source bit
                1 = ADC internal RC clock
                0 = Clock derived from system clock

bit 14-13   **Unimplemented:** Read as '0'

bit 12-8   **SAMC<4:0>:** Auto-Sample Time bits[1,2]
                11111 = 31 T_{AD}
                •
                •
                •
                00001 = 1 T_{AD}
                00000 = 0 T_{AD}

bit 7-0   **ADCS<7:0>:** ADC Conversion Clock Select bits[3]
                11111111 = $T_{CY} \bullet (ADCS<7:0> + 1) = 256 \bullet T_{CY} = T_{AD}$
                •
                •
                •
                00000010 = $T_{CY} \bullet (ADCS<7:0> + 1) = 3 \bullet T_{CY} = T_{AD}$
                00000001 = $T_{CY} \bullet (ADCS<7:0> + 1) = 2 \bullet T_{CY} = T_{AD}$
                00000000 = $T_{CY} \bullet (ADCS<7:0> + 1) = 1 \bullet T_{CY} = T_{AD}$

Note 1: These bits are only used when the SSRC<2:0> bits (ADxCON1<7:5>) = 111 and SSRCG = 0.

2: If SSRC<2:0> = 111 and SSRCG = 0, the SAMC<4:0> bits should be set to at least '11111' when using one S&H channel or using simultaneous sampling. When using multiple S&H channels with sequential sampling, the SAMCx bits should be set to '00000' for the fastest possible conversion rate.

3: These bits are not used if the ADRC bit (ADxCON3<15>) = 1.

Register 16-4: ADxCON4: ADCx Control Register 4

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	ADDMAEN[1]
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	DMABL<2:0>		
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 15-9 **Unimplemented:** Read as '0'

bit 8 **ADDMAEN:** ADC DMA Enable bit[1]

1 = Conversion results stored in ADCxBUF0 register for transfer to RAM using DMA
0 = Conversion results stored in ADCxBUF0 through ADCxBUFF registers; DMA is not used

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **DMABL<2:0>:** Selects Number of DMA Buffer Locations per Analog Input bits

111 = Allocates 128 words of buffer to each analog input
110 = Allocates 64 words of buffer to each analog input
101 = Allocates 32 words of buffer to each analog input
100 = Allocates 16 words of buffer to each analog input
011 = Allocates 8 words of buffer to each analog input
010 = Allocates 4 words of buffer to each analog input
001 = Allocates 2 words of buffer to each analog input
000 = Allocates 1 word of buffer to each analog input

Note 1: If this bit is cleared to disable DMA, the DMABL<2:0> and ADDMABM bits have no effect.

Note: This register is not available in all devices. Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

Register 16-5:   ADxCHS123: ADCx Input Channel 1, 2, 3 Select Register

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH123SB<2:1>		CH123NB<1:0>		CH123SB0
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
—	—	—	CH123SA<2:1>		CH123NA<1:0>		CH123SA0
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 12-11   **CH123SB<2:1>:** Channels 1, 2, 3 Positive Input Select for Sample B bits

bit 10-9   **CH123NB<1:0>:** Channels 1, 2, 3 Negative Input Select for Sample B bits

bit 8   **CH123SB0:** Channels 1, 2, 3 Positive Input Select for Sample B bit

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

bit 4-3   **CH123SA<2:1>:** Channels 1, 2, 3 Positive Input Select for Sample A bits

bit 2-1   **CH123NA<1:0>:** Channels 1, 2, 3 Negative Input Select for Sample A bits

bit 0   **CH123SA0:** Channels 1, 2, 3 Positive Input Select for Sample A bit

> **Note:**   The bit settings in this register vary by device. Refer to the ADxCHS123 register in the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

Register 16-6:   ADxCHS0: ADCx Input Channel 0 Select Register

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NB	—			CH0SB<5:0>[1]			
bit 15							bit 8

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NA	—			CH0SA<5:0>[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   **CH0NB:** Channel 0 Negative Input Select for Sample B bit

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

bit 13-8   **CH0SB<5:0>:** Channel 0 Positive Input Select for Sample B bits[1]

bit 7   **CH0NA:** Channel 0 Negative Input Select for Sample A bit

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

bit 5-0   **CH0SA<5:0>:** Channel 0 Positive Input Select for Sample A bits[1]

Note 1:   These bits have no effect when the CSCNA bit (ADxCON2<10>) = 1.

Note: The bit settings in this register vary by device. Refer to the ADxCHS0 register in the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

Register 16-7:   ADxCSSH: ADCx Input Scan Select 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
CSS31	CSS30	CSS29	CSS28	CSS27	CSS26	CSS25	CSS24
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
CSS23	CSS22	CSS21	CSS20	CSS19	CSS18	CSS17	CSS16
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>:** ADC Input Scan Selection bits
1 = Selects ANx for input scan
0 = Skips ANx for input scan

> **Note:**   Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability of channel scan selections.

Register 16-8:   ADxCSSL: ADCx Input Scan Select 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
CSS15	CSS14	CSS13	CSS12	CSS11	CSS10	CSS9	CSS8
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
CSS7	CSS6	CSS5	CSS4	CSS3	CSS2	CSS1	CSS0
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>:** ADC Input Scan Selection bits
1 = Selects ANx for input scan
0 = Skips ANx for input scan

> **Note:**   Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability of channel scan selections.

Register 16-9: **ANSELy: Analog/Digital Pin Selection Register**

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANSy15	ANSy14	ANSy13	ANSy12	ANSy11	ANSy10	ANSy9	ANSy8
bit 15							bit 8

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANSy7	ANSy6	ANSy5	ANSy4	ANSy3	ANSy2	ANSy1	ANSy0
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   **ANSy<15:0>:** Analog/Digital Pin Selection bits

       1 = Pin is configured as an analog input
       0 = Pin is configured as a digital I/O pin

Note: Refer to the **"I/O Ports"** chapter in the specific device data sheet for availability of I/O ports. The 'y' in ANSELy refers to PORTA, PORTB, PORTC, etc.


16.3 OVERVIEW OF SAMPLE AND CONVERSION SEQUENCE

Figure 16-3 illustrates the three-step process of the Analog-to-Digital conversion:

1. The input voltage signal is connected to the sample capacitor.
2. The sample capacitor is disconnected from the input.
3. The stored voltage is converted to equivalent digital bits.

The two distinct phases, sample and convert, are independently controlled.

Figure 16-3: Sample Conversion Sequence
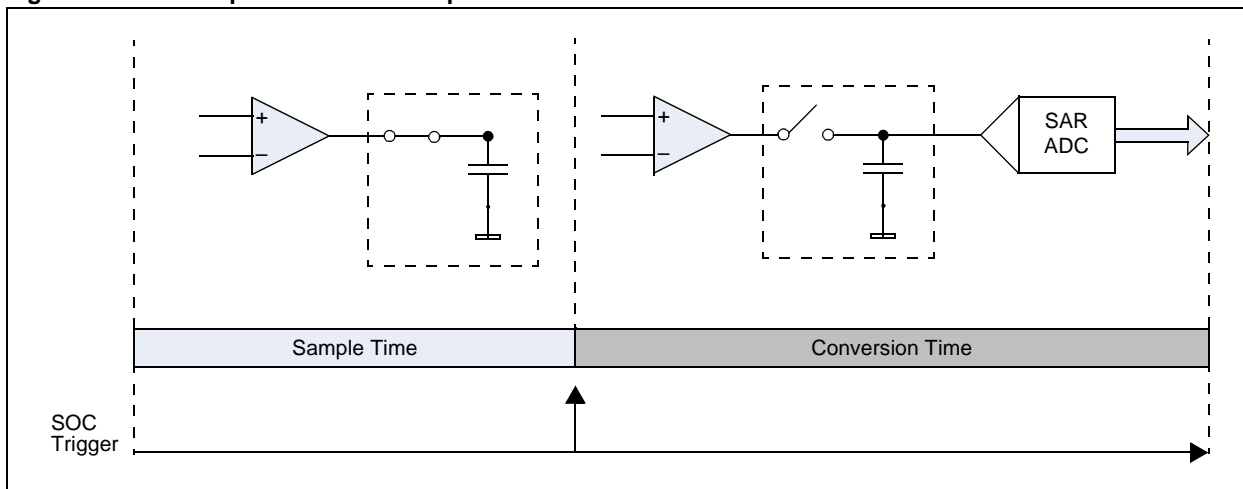
16.3.1 Sample Time

Sample time is when the selected analog input is connected to the sample capacitor. There is a minimum sample time to ensure that the S&H amplifier provides a desired accuracy for the Analog-to-Digital conversion (see **Section 16.13 "Analog-to-Digital Sampling Requirements"**).

> **Note:** The ADC module requires a finite number of Analog-to-Digital clock cycles to start conversion after receiving a conversion trigger or ending the sampling process. For more details, refer to the T_{PCS} parameter in the **"Electrical Characteristics"** chapter of the specific device data sheet.

The sampling phase can be set up to start automatically upon conversion or by manually setting the Sample bit (SAMP) in the ADC Control Register 1 (ADxCON1<1>). The sampling phase is controlled by the Auto-Sample bit (ASAM) in the ADC Control Register 1 (ADxCON1<2>). Table 16-1 lists the options selected by the specific bit configuration.

Table 16-1: Start of Sampling Selection

ASAM	Start of Sampling Selection
0	Manual Sampling
1	Automatic Sampling

If automatic sampling is enabled, the Sampling Time (T_{SMP}) taken by the ADC module is equal to the number of T_{AD} cycles defined by the SAMC<4:0> bits (ADxCON3<12:8>), as shown in Equation 16-1.

Equation 16-1: Sampling Time Calculation

$$T_{SMP} = SAMC\langle4{:}0\rangle \cdot T_{AD}$$

If manual sampling is desired, the user software must provide sufficient time to ensure adequate sampling time.

16.3.2 Conversion Time

The Start of Conversion (SOC) trigger ends the sampling time and begins an Analog-to-Digital conversion. During the conversion period, the sample capacitor is disconnected from the multiplexer and the stored voltage is converted to equivalent digital bits. The conversion times for 10-bit and 12-bit modes are shown in Equation 16-2 and Equation 16-3. The sum of the sample time and the Analog-to-Digital conversion time provides the total conversion time.

For correct Analog-to-Digital conversion, the Analog-to-Digital Conversion Clock (TAD) must be selected to ensure a minimum TAD time. Refer to the **"Electrical Characteristics"** chapter of the specific device data sheet for the minimum TAD specifications for 10-bit and 12-bit modes.

Equation 16-2: 10-Bit ADC Conversion Time

$$T_{CONV} = 12 \cdot T_{AD}$$

Where:
T_{CONV} = Conversion Time
T_{AD} = ADC Clock Period

Equation 16-3: 12-Bit ADC Conversion Time

$$T_{CONV} = 14 \cdot T_{AD}$$

Where:
T_{CONV} = Conversion Time
T_{AD} = ADC Clock Period

The SOC can be triggered by a variety of hardware sources or controlled manually in user software. The trigger source to initiate conversion is selected by the SOC Trigger Source Select bits (SSRC<2:0>) in the ADCx Control Register 1 (ADxCON1<7:5>). The Sample Clock Source Group bit, SSRCG (ADxCON1<4>), selects between the two groups. The SSRCx bits provide different sample clock sources based on the group selected.

Note:	Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for the available SOC trigger sources.

Table 16-2 lists the sample conversion sequence with different sample and conversion phase selections.

Table 16-2: Sample Conversion Sequence Selection

ASAM	SSRCG	SSRC<2:0>	Description
0	0	000	Manual Sample and Manual Conversion Sequence
0	0	111	Manual Sample and Automatic Conversion Sequence
0	0 or 1	001 010 011 100	Manual Sample and Triggered Conversion Sequence
	1	000 111	
1	0	000	Automatic Sample and Manual Conversion Sequence
1	0	111	Automatic Sample and Automatic Conversion Sequence
1	0 or 1	001 010 011 100	Automatic Sample and Triggered Conversion Sequence
	1	000 111	

16.3.3 Manual Sample and Manual Conversion Sequence

In the Manual Sample and Manual Conversion Sequence, setting the Sample bit (SAMP) in the ADCx Control Register 1 (ADxCON1<1>) initiates sampling, and clearing the SAMP bit terminates sampling and starts the conversion (see Figure 16-4). The user application must time the setting and clearing of the SAMP bit to ensure adequate sampling time for the input signal. Example 16-1 shows a code sequence for Manual Sample and Manual Conversion.

Figure 16-4:   Manual Sample and Manual Conversion Sequence

Note	1:	Sampling starts by setting the SAMP bit (ADxCON1<1>) in software.
	2:	Conversion starts by clearing the SAMP bit in software.
	3:	Conversion is complete.
	4:	Sampling starts by setting the SAMP bit in software.
	5:	Conversion starts by clearing the SAMP bit in software.
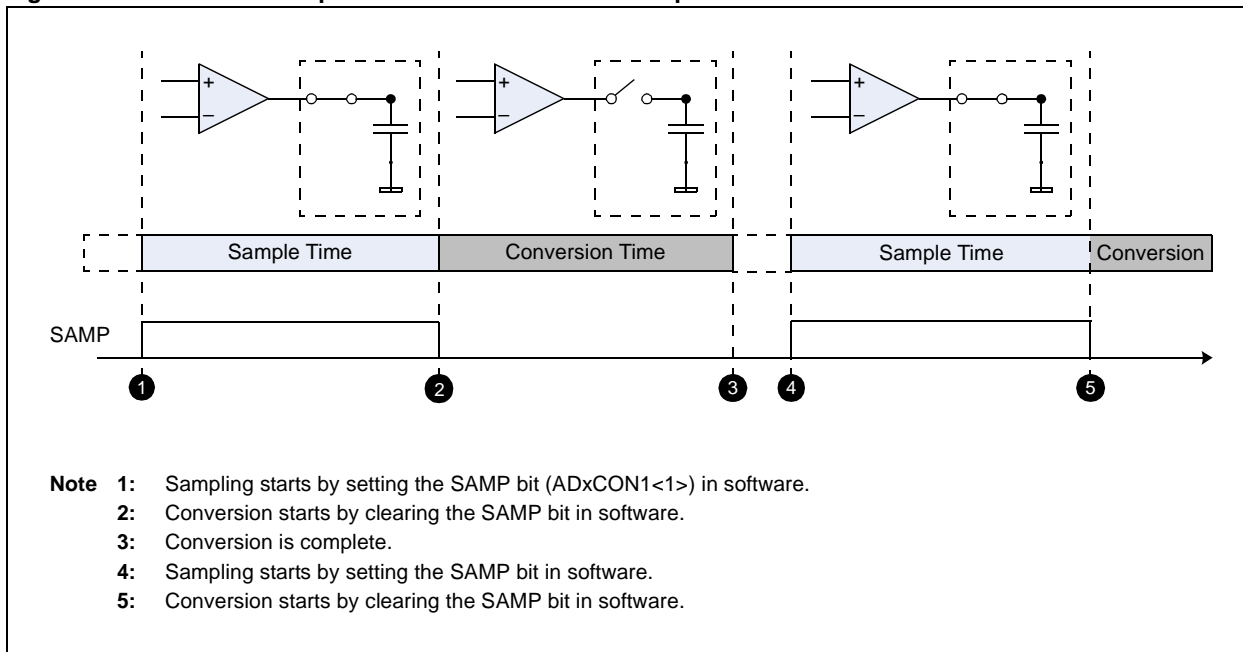
Example 16-1:   Code Sequence for Manual Sample and Manual Conversion

```
#include <p33Exxxx.h>

/*****************************CONFIGURATION**************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void Delay_us(unsigned int);
int  ADCValue, i;

int  main(void)
{
    // Configure the device PLL to obtain 40 MIPS operation. The crystal frequency is 8 MHz.
    // Divide 8 MHz by 2, multiply by 40 and divide by 2. This results in Fosc of 80 MHz.
    // The CPU clock frequency is Fcy = Fosc/2 = 40 MHz.
    PLLFBD = 38;                      /* M  = 40 */
    CLKDIVbits.PLLPOST = 0;           /* N1 = 2  */
    CLKDIVbits.PLLPRE = 0;            /* N2 = 2  */
    OSCTUN = 0;

    /* Initiate Clock Switch to Primary Oscillator with PLL (NOSC = 0x3) */
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);               /* Wait for PLL lock at 40 MIPS */

    initAdc1();

    while(1)
    {
      AD1CON1bits.SAMP = 1;          // Start sampling
      Delay_us(10);                  // Wait for sampling time (10 us)
      AD1CON1bits.SAMP = 0;          // Start the conversion
      while (!AD1CON1bits.DONE);     // Wait for the conversion to complete
      ADCValue = ADC1BUF0;           // Read the ADC conversion result
    }
}

void initAdc1(void)
{
    /* Set port configuration */
    ANSELA = ANSELB = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
    ANSELBbits.ANSB5 = 1;            // Ensure AN5/RB5 is analog

    /* Initialize and enable ADC module */
    AD1CON1 = 0x0000;
    AD1CON2 = 0x0000;
    AD1CON3 = 0x000F;
    AD1CON4 = 0x0000;
    AD1CHS0 = 0x0005;
    AD1CHS123 = 0x0000;
    AD1CSSH = 0x0000;
    AD1CSSL = 0x0000;
    AD1CON1bits.ADON = 1;
    Delay_us(20);
}

void Delay_us(unsigned int delay)
{
    for (i = 0; i < delay; i++)
    {
      __asm__ volatile ("repeat #39");
      __asm__ volatile ("nop");
    }
}
```
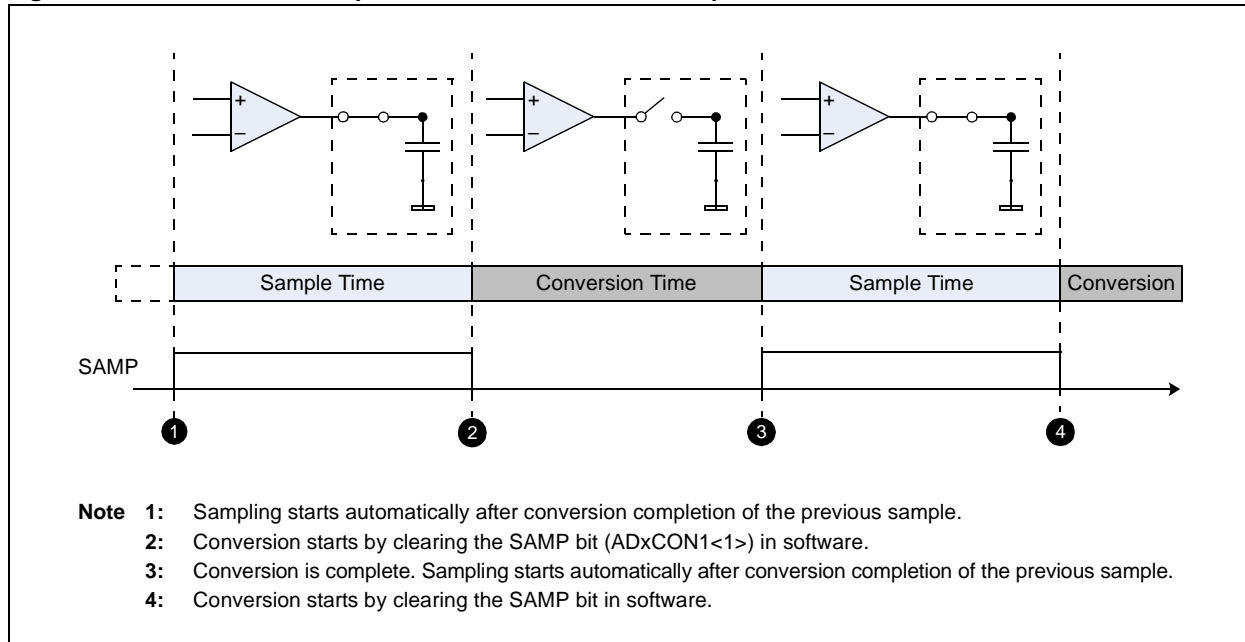
Note: Due to the internal delay within the ADC module, the SAMP bit (ADxCON1<1>) will read as '0' to the user software. This change occurs in a small interval of time after the conversion has started. In general, the time interval is 2 T_{CY}.

16.3.4 Automatic Sample and Manual Conversion Sequence

In the Automatic Sample and Manual Conversion Sequence, sampling starts automatically after conversion of the previous sample. The user application must allocate sufficient time for sampling before clearing the SAMP bit (ADxCON1<1>). Clearing the SAMP bit initiates the conversion (see Figure 16-5).

Figure 16-5:   Automatic Sample and Manual Conversion Sequence

Note 1: Sampling starts automatically after conversion completion of the previous sample.

   2: Conversion starts by clearing the SAMP bit (ADxCON1<1>) in software.

   3: Conversion is complete. Sampling starts automatically after conversion completion of the previous sample.

   4: Conversion starts by clearing the SAMP bit in software.

Example 16-2: Code Sequence for Automatic Sample and Manual Conversion

```c
#include <p33Exxxx.h>

/*************************CONFIGURATION*************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void Delay_us(unsigned int);
int  ADCValue, i, j;

int  main(void)
{
    // Configure the device PLL to obtain 40 MIPS operation. The crystal frequency is 8 MHz.
    // Divide 8 MHz by 2, multiply by 40 and divide by 2. This results in Fosc of 80 MHz.
    // The CPU clock frequency is Fcy = Fosc/2 = 40 MHz.
    PLLFBD = 38;                        /* M  = 40 */
    CLKDIVbits.PLLPOST = 0;             /* N1 = 2  */
    CLKDIVbits.PLLPRE = 0;             /* N2 = 2  */
    OSCTUN = 0;

    /* Initiate Clock Switch to Primary Oscillator with PLL (NOSC = 0x3) */
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);                 /* Wait for PLL lock at 40 MIPS */

    initAdc1();

    while(1)
    {
      Delay_us(100);                    // Sample for 100 us
      AD1CON1bits.SAMP = 0;             // Start the conversion
      while (!AD1CON1bits.DONE);        // Wait for the conversion to complete
      AD1CON1bits.DONE = 0;             // Clear conversion done status bit
      ADCValue = ADC1BUF0;             // Read the ADC conversion result
    }
}

void initAdc1(void)
{
    /* Set port configuration */
    ANSELA = ANSELB = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
    ANSELBbits.ANSB5 = 1;            // Ensure AN5/RB5 is analog

    /* Initialize and enable ADC module */
    AD1CON1 = 0x0004;
    AD1CON2 = 0x0000;
    AD1CON3 = 0x000F;
    AD1CON4 = 0x0000;
    AD1CHS0 = 0x0005;
    AD1CHS123 = 0x0000;
    AD1CSSH = 0x0000;
    AD1CSSL = 0x0000;
    AD1CON1bits.ADON = 1;
    Delay_us(20);
}

void Delay_us(unsigned int delay)
{
    for (i = 0; i < delay; i++)
    {
      __asm__ volatile ("repeat #39");
      __asm__ volatile ("nop");
    }
}
```
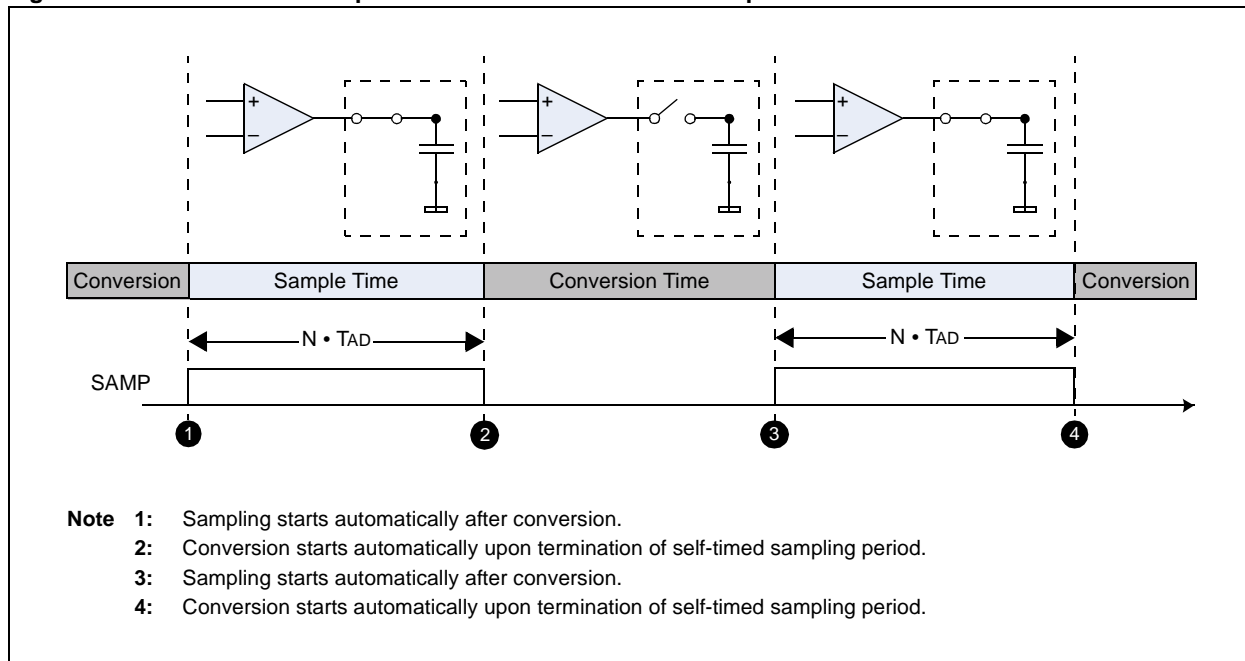
16.3.5   Automatic Sample and Automatic Conversion Sequence

16.3.5.1   CLOCKED CONVERSION TRIGGER

The auto-conversion method provides a more automated process to sample and convert the analog inputs, as shown in Figure 16-6. The sampling period is self-timed and the conversion starts automatically upon termination of a self-timed sampling period. The Auto-Sample Time bits (SAMC<4:0>) in the ADxCON3 register (ADxCON3<12:8>) select 0 to 31 ADC clock cycles (T_{AD}) for the sampling period. Refer to the **"Electrical Characteristics"** chapter of the specific device data sheet for a minimum recommended sampling time (SAMCx bits value).

The SSRCG bit is set to '0' and the SSRC<2:0> bits are set to '111' to choose the internal counter as the sample clock source, which ends sampling and starts conversion.

Figure 16-6:   Automatic Sample and Automatic Conversion Sequence

Note   1:   Sampling starts automatically after conversion.
    2:   Conversion starts automatically upon termination of self-timed sampling period.
    3:   Sampling starts automatically after conversion.
    4:   Conversion starts automatically upon termination of self-timed sampling period.
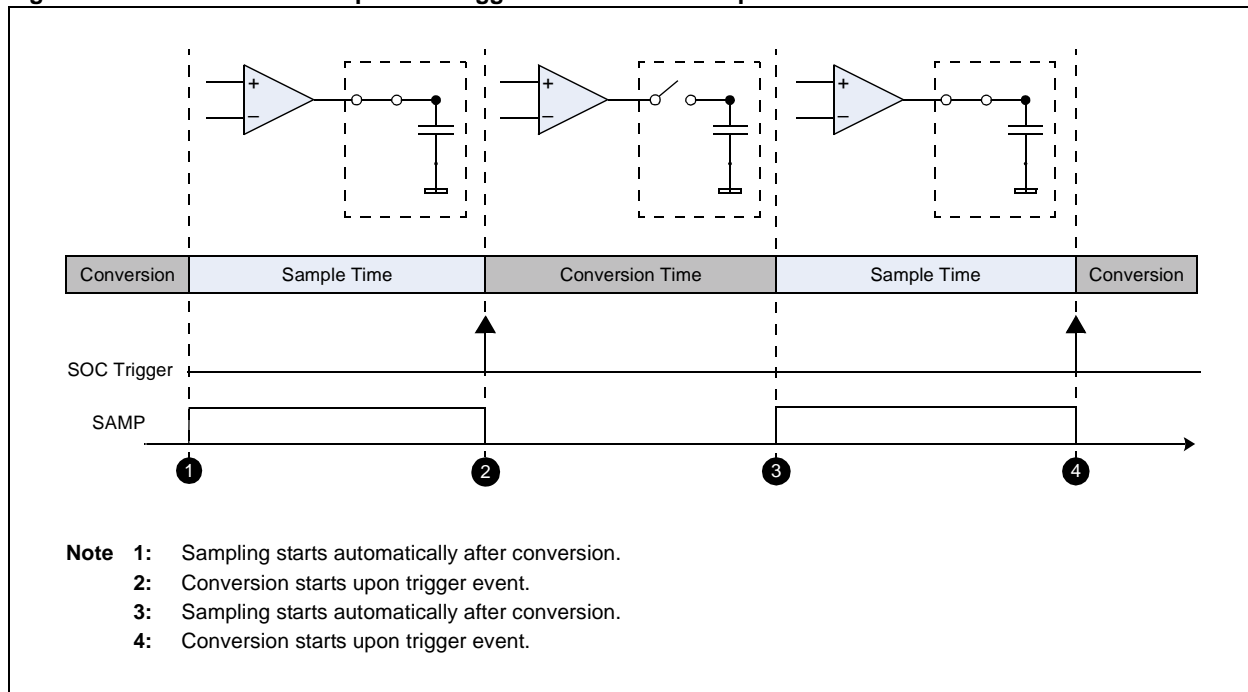
16.3.5.2 EXTERNAL CONVERSION TRIGGER

In an Automatic Sample and Triggered Conversion Sequence, sampling starts automatically after conversion and the conversion starts upon a trigger event from the selected peripheral, as shown in Figure 16-7. This enables ADC conversion to be synchronized with the internal or external events. The external conversion trigger is selected by configuring the SSRC<2:0> bits as shown in Table 16-2. Refer to **Section 16.4.8 "Conversion Trigger Sources"** for various external conversion trigger sources.

The ASAM bit must not be modified while the ADC is turned on. If automatic sampling is desired, the ASAM bit must be set before turning the module on. The ADC module takes some amount of time to stabilize (see the T$_{DPU}$ parameter in the specific device data sheet). If automatic sampling is enabled, there is no assurance that the initial ADC results are correct until the ADC module stabilizes. It may be necessary to discard the first few ADC results depending on the Analog-to-Digital clock speed.

Figure 16-7:   Automatic Sample and Triggered Conversion Sequence

Note   **1:**   Sampling starts automatically after conversion.
       2:   Conversion starts upon trigger event.
       3:   Sampling starts automatically after conversion.
       4:   Conversion starts upon trigger event.

16.3.6 Multi-Channel Sample Conversion Sequence

Multi-channel Analog-to-Digital Converters typically convert each input channel sequentially using an input multiplexer. Simultaneously sampling multiple signals ensures that the snapshot of the analog inputs occurs at precisely the same time for all inputs, as shown in Figure 16-8.

Certain applications require simultaneous sampling, especially when phase information exists between different channels. Sequential sampling takes a snapshot of each analog input just before conversion starts on that input, as shown in Figure 16-8. The sampling of multiple inputs is not correlated. For example, motor control and power monitoring requires voltage and current measurements, and the phase angle between them.

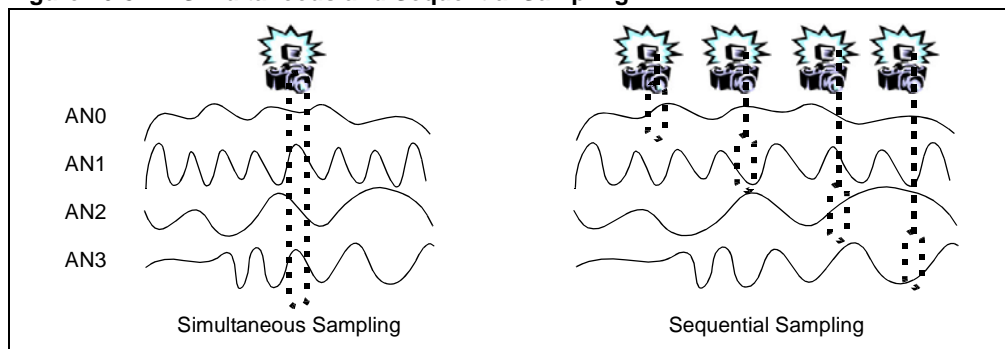**Figure 16-8:   Simultaneous and Sequential Sampling**

Figure 16-9 and Figure 16-10 illustrate that the ADC module supports simultaneous sampling using two S&H or four S&H channels to sample the inputs at the same time, and then performs the conversion for each channel sequentially.

The Simultaneous Sampling mode is selected by setting the Simultaneous Sampling bit (SIMSAM) in the ADCx Control Register 1 (ADxCON1<3>). By default, the channels are sampled and converted sequentially. Table 16-3 lists the options selected by a specific bit configuration. The CHPS<1:0> bits determine the channels to be sampled, either sequentially or simultaneously.

Table 16-3:   Start of Sampling Selection

SIMSAM	Sampling Mode
0	Sequential Sampling
1	Simultaneous Sampling

Figure 16-9:   2-Channel Simultaneous Sampling (ASAM = 1)
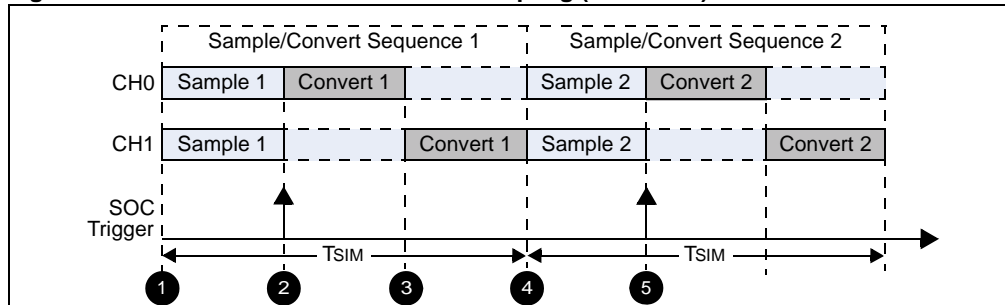
Note 1: CH0-CH1 input multiplexer selects the analog input for sampling. The selected analog input connects to the sample capacitor.

   2: On a SOC trigger, CH0-CH1 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0 is converted to equivalent digital bits.

   3: The analog voltage captured in CH1 is converted to equivalent digital bits.

   4: CH0-CH1 input multiplexer selects the next analog input for sampling. The selected analog input connects to the sample capacitor.

   5: On a SOC trigger, CH0-CH1 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0 is converted to equivalent digital bits.

For simultaneous sampling, the total time taken to sample and convert the channels is shown in Equation 16-4.

Equation 16-4: Channel Sample and Conversion Total Time, Simultaneous Sampling Selected

$$T_{SIM} = T_{SMP} + (M \bullet T_{CONV})$$

Where:

T_{SIM} = Total Time to Sample and Convert multiple channels with simultaneous sampling

T_{SMP} = Sampling Time (see Equation 16-1)

T_{CONV} = Conversion Time (see Equation 16-2)

M = Number of Channels selected by the CHPS<1:0> bits

Figure 16-10: 4-Channel Simultaneous Sampling

Note 1: CH0-CH3 input multiplexer selects the analog input for sampling. The selected analog input connects to the sample capacitor.

2: On a SOC trigger, CH0-CH3 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0 is converted to equivalent digital bits.

3: The analog voltage captured in CH1 is converted to equivalent digital bits.

4: The analog voltage captured in CH2 is converted to equivalent digital bits.

5: The analog voltage captured in CH3 is converted to equivalent digital bits.

6: CH0-CH3 input multiplexer selects the next analog input for sampling. The selected analog input connects to the sample capacitor.

7: On a SOC trigger, CH0-CH3 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0 is converted to equivalent digital bits.
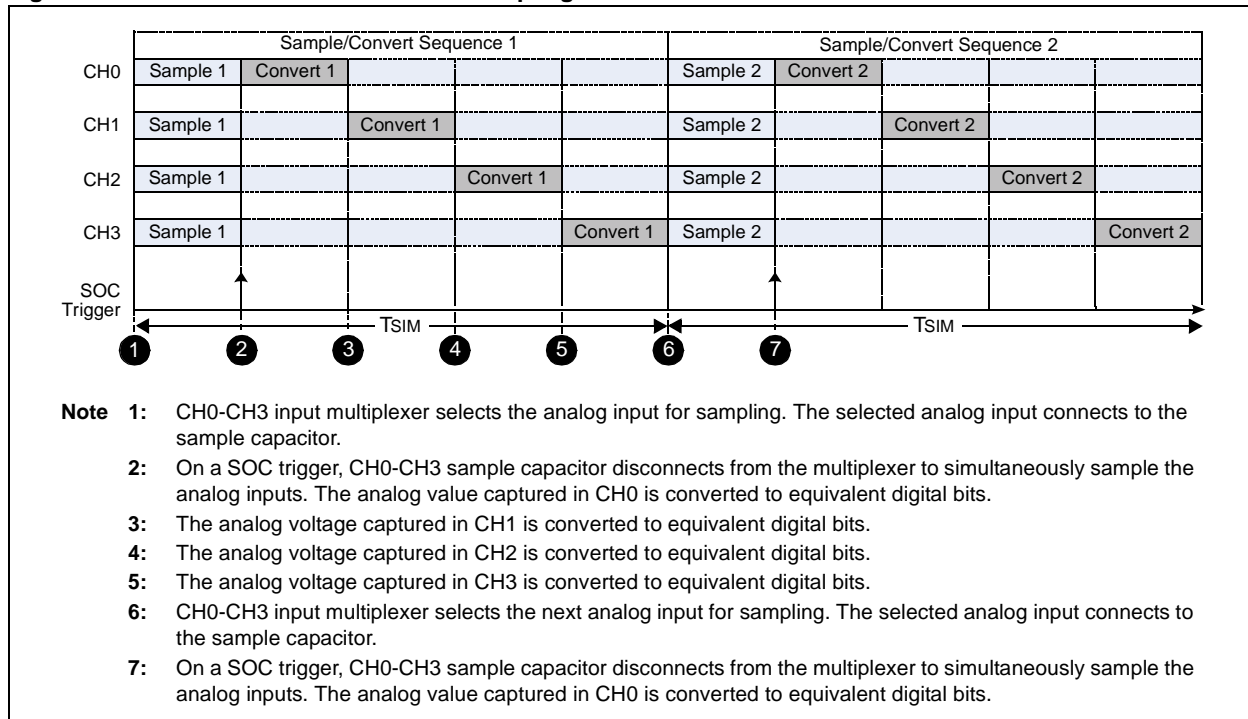
Figure 16-11 and Figure 16-12 illustrate that, by default, multiple channels are sampled and converted sequentially.

For sequential sampling, the total time taken to sample and convert channels is shown in Equation 16-5.

Equation 16-5: Channel Sample and Conversion Total Time, Sequential Sampling Selected

When $T_{SMP} < T_{CONV}$,

$$T_{SEQ} = M \bullet T_{CONV} \qquad \text{(if M > 1)}$$

$$T_{SEQ} = T_{SMP} + T_{CONV} \qquad \text{(if M = 1)}$$

Where:

T_{SEQ} = Total Time to Sample and Convert multiple channels with sequential sampling

T_{CONV} = Conversion Time (see Equation 16-2)

T_{SMP} = Sampling Time (see Equation 16-1)

M = Number of Channels selected by the CHPS<1:0> bits

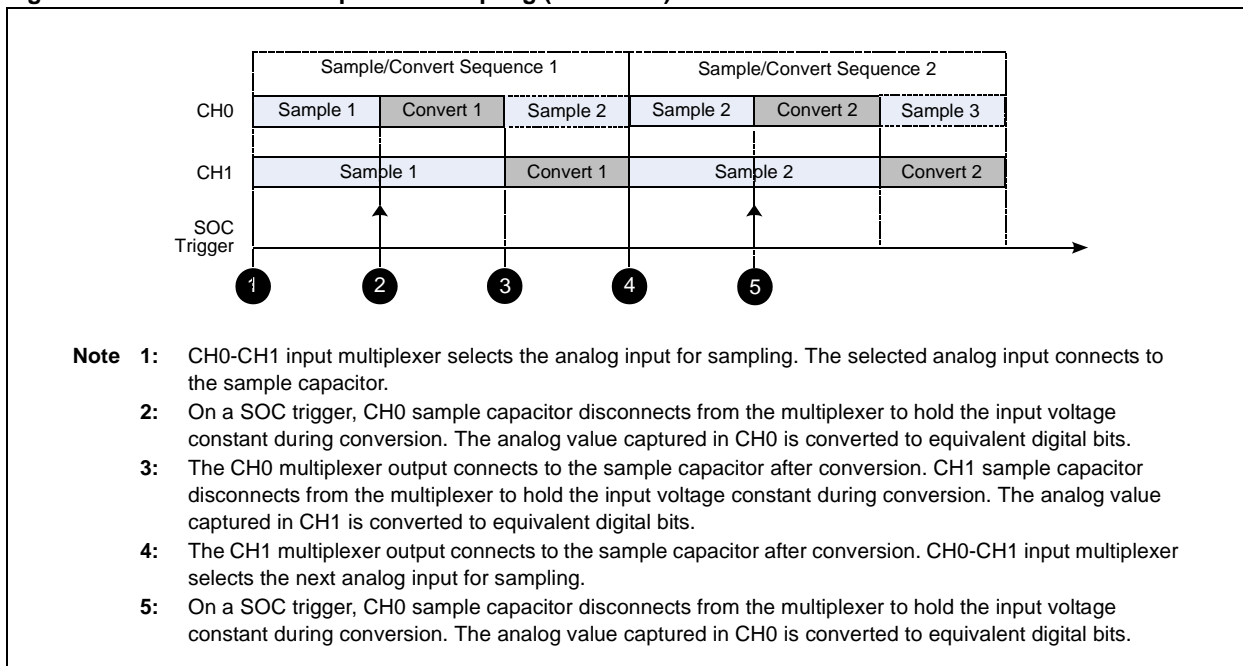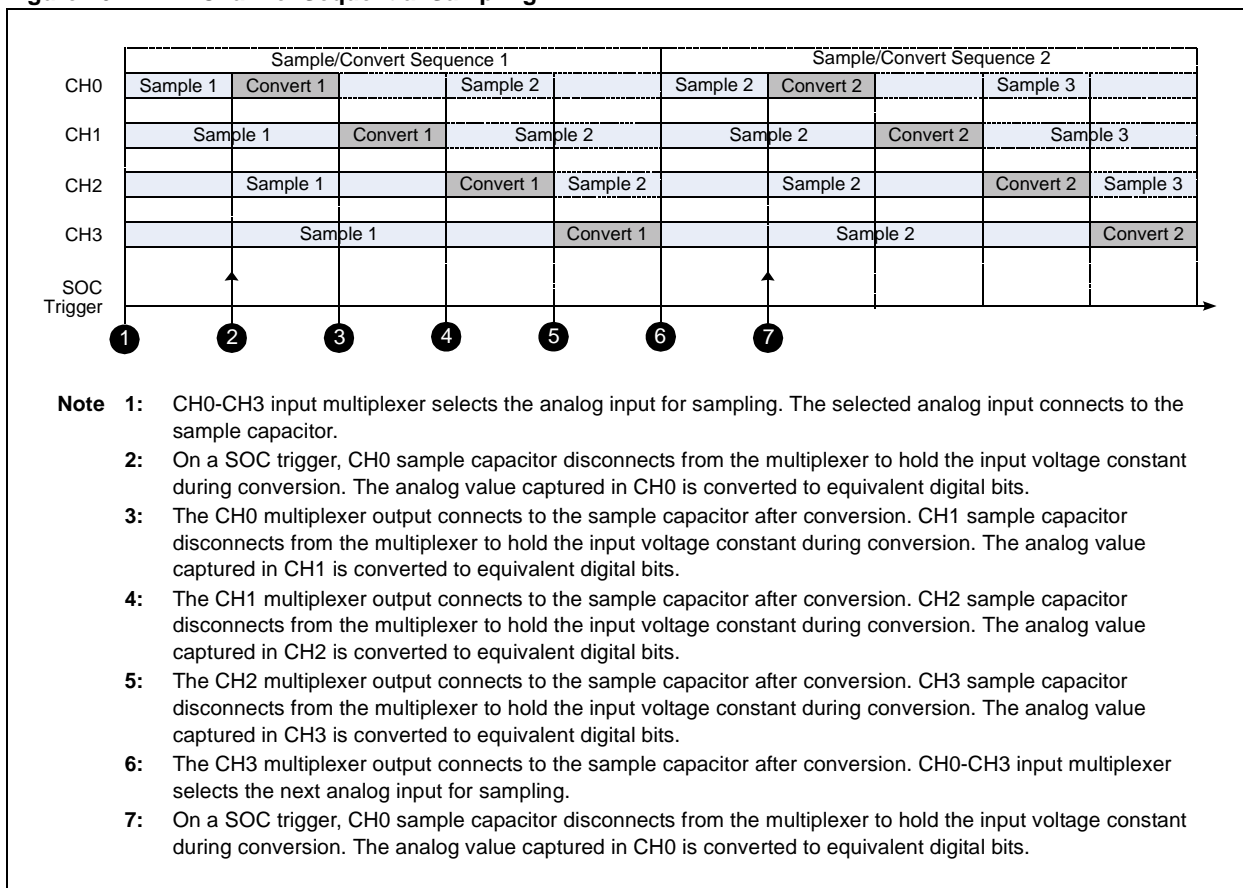**Figure 16-11: 2-Channel Sequential Sampling (ASAM = 1)**

Note 1: CH0-CH1 input multiplexer selects the analog input for sampling. The selected analog input connects to the sample capacitor.

2: On a SOC trigger, CH0 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH0 is converted to equivalent digital bits.

3: The CH0 multiplexer output connects to the sample capacitor after conversion. CH1 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH1 is converted to equivalent digital bits.

4: The CH1 multiplexer output connects to the sample capacitor after conversion. CH0-CH1 input multiplexer selects the next analog input for sampling.

5: On a SOC trigger, CH0 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH0 is converted to equivalent digital bits.

Figure 16-12: 4-Channel Sequential Sampling

Note 1: CH0-CH3 input multiplexer selects the analog input for sampling. The selected analog input connects to the sample capacitor.

2: On a SOC trigger, CH0 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH0 is converted to equivalent digital bits.

3: The CH0 multiplexer output connects to the sample capacitor after conversion. CH1 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH1 is converted to equivalent digital bits.

4: The CH1 multiplexer output connects to the sample capacitor after conversion. CH2 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH2 is converted to equivalent digital bits.

5: The CH2 multiplexer output connects to the sample capacitor after conversion. CH3 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH3 is converted to equivalent digital bits.

6: The CH3 multiplexer output connects to the sample capacitor after conversion. CH0-CH3 input multiplexer selects the next analog input for sampling.

7: On a SOC trigger, CH0 sample capacitor disconnects from the multiplexer to hold the input voltage constant during conversion. The analog value captured in CH0 is converted to equivalent digital bits.

16.4 ADC CONFIGURATION

16.4.1 Disabling the Use of DMA with the ADC Module

When the ADDMAEN bit (ADxCON4<8>) is '1' (default), the ADC module can use DMA to transfer conversion results from the ADCxBUF0 register to DMA RAM.

When the ADDMAEN bit is '0', the DMA cannot be used with the ADC module and the DMABL<2:0> and ADDMABM bits have no effect. Additionally, the conversion results are stored in the ADCxBUF0-ADCxBUFF registers.

> **Note:** The ADDMAEN bit is only available on devices with DMA. Refer to the specific device data sheet for availability.

16.4.2 ADC Operational Mode Selection

The 12-Bit ADC Operation Mode bit (AD12B) in the ADCx Control Register 1 (ADxCON1<10>) enables the ADC module to function as either a 10-bit, 4-channel ADC (default configuration) or a 12-bit, single channel ADC. Table 16-4 lists the options selected by different bit settings.

> **Note:** The ADC module must be disabled before the AD12B bit is modified.

Table 16-4: ADC Operational Mode

AD12B	Channel Selection
0	10-Bit, 4-Channel ADC
1	12-Bit, Single Channel ADC

16.4.3 ADC Channel Selection

In 10-bit mode (AD12B = 0), the user application can select 1-Channel (CH0), 2-Channel (CH0, CH1) or 4-Channel mode (CH0-CH3) using the Channel Select bits (CHPS<1:0>) in the ADCx Control Register 2 (ADxCON2<9:8>). In 12-bit mode, the user application can only use CH0. Table 16-5 lists the number of channels selected for the different bit settings.

> **Note:** ADC2 can operate only in 10-bit mode.

Table 16-5: 10-Bit ADC Channel Selection

CHPS<1:0>	Channel Selection
00	CH0
01	Dual Channel (CH0, CH1)
1x	Multi-Channel (CH0-CH3)

16.4.4 Voltage Reference Selection

The voltage references for Analog-to-Digital conversions are selected using the Voltage Reference Configuration bits (VCFG<2:0>) in the ADCx Control Register 2 (ADxCON2<15:13>). Table 16-6 lists the voltage reference selection for different bit settings.The Voltage Reference High (VREFH) and the Voltage Reference Low (VREFL) to the ADC module can be supplied from the internal AVDD and AVSS voltage rails or the external VREF+ and VREF- input pins. The external voltage reference pins can be shared with the AN0 and AN1 inputs on low pin count devices. The ADC module can still perform conversions on these pins when they are shared with the VREF+ and VREF- input pins. The voltages applied to the external reference pins must meet certain specifications. For more details, refer to the **"Electrical Characteristics"** chapter of the specific device data sheet.

Table 16-6: Voltage Reference Selection

VCFG<2:0>	VREFH	VREFL
000	AVDD	AVSS
001	VREF+	AVSS
010	AVDD	VREF-
011	VREF+	VREF-
1xx	AVDD	AVSS

16.4.5 ADC Clock Selection

The ADC module can be clocked from the instruction cycle clock (TCY) or by using the dedicated internal RC clock (see Figure 16-13). When using the instruction cycle clock, a clock divider drives the instruction cycle clock and enables a lower frequency to be chosen. The clock divider is controlled by the ADC Conversion Clock Select bits (ADCS<7:0>) in the ADCx Control Register 3 (ADxCON3<7:0>), which enables 256 settings, from 1:1 to 1:256, to be chosen.

Equation 16-6 shows the ADC clock period (TAD) as a function of the ADCSx control bits and the device instruction cycle clock period, TCY.

> **Note:** Refer to the **"Electrical Characteristics"** chapter in the specific device data sheet for minimum TAD specifications.

Equation 16-6: ADC Clock Period

If ADRC = 0:
$ADC\ Clock\ Period\ (T_{AD}) = T_{CY} \cdot (ADCS\!<\!7\!:\!0\!> + 1)$

If ADRC = 1:
$ADC\ Clock\ Period\ (T_{AD}) = T_{ADRC}$

The ADC module has a dedicated internal RC clock source that can be used to perform conversions. The internal RC clock source is used when Analog-to-Digital conversions are performed while the device is in Sleep mode. The internal RC oscillator is selected by setting the ADC Conversion Clock Source bit (ADRC) in ADCx Control Register 3 (ADxCON3<15>). When the ADRC bit is set, the ADCS<7:0> bits have no effect on the ADC operation.

> **Note:** Refer to the **"Electrical Characteristics"** chapter in the specific device data sheet for ADRC frequency specifications.

Figure 16-13: ADC Clock Generation

Note 1: $T_P = 1/F_P$.
   2: Refer to the **"Electrical Characteristics"** chapter in the specific device data sheet for the exact ADC internal RC value.
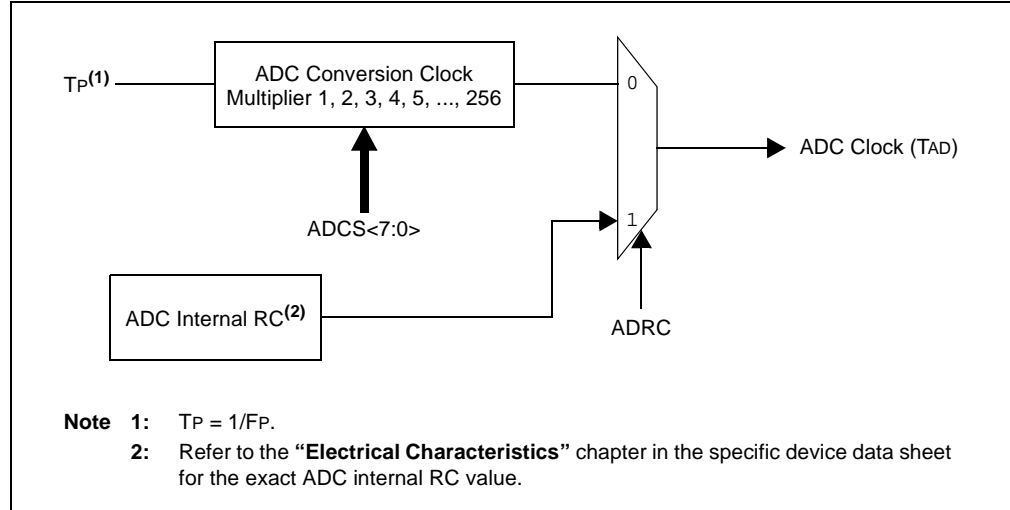
16.4.6 Output Data Format Selection

Figure 16-14 illustrates that the ADC result is available in four different numerical formats. The Data Output Format bits (FORM<1:0>) in the ADCx Control Register 1 (ADxCON1<9:8>) select the output data format. Table 16-7 lists the ADC output format for different bit settings.

Table 16-7: Voltage Reference Selection

FORM<1:0>	Data Information Selection
11	Signed Fractional Format
10	Unsigned Fractional Format
01	Signed Integer Format
00	Unsigned Integer Format

Figure 16-14: ADC Output Format
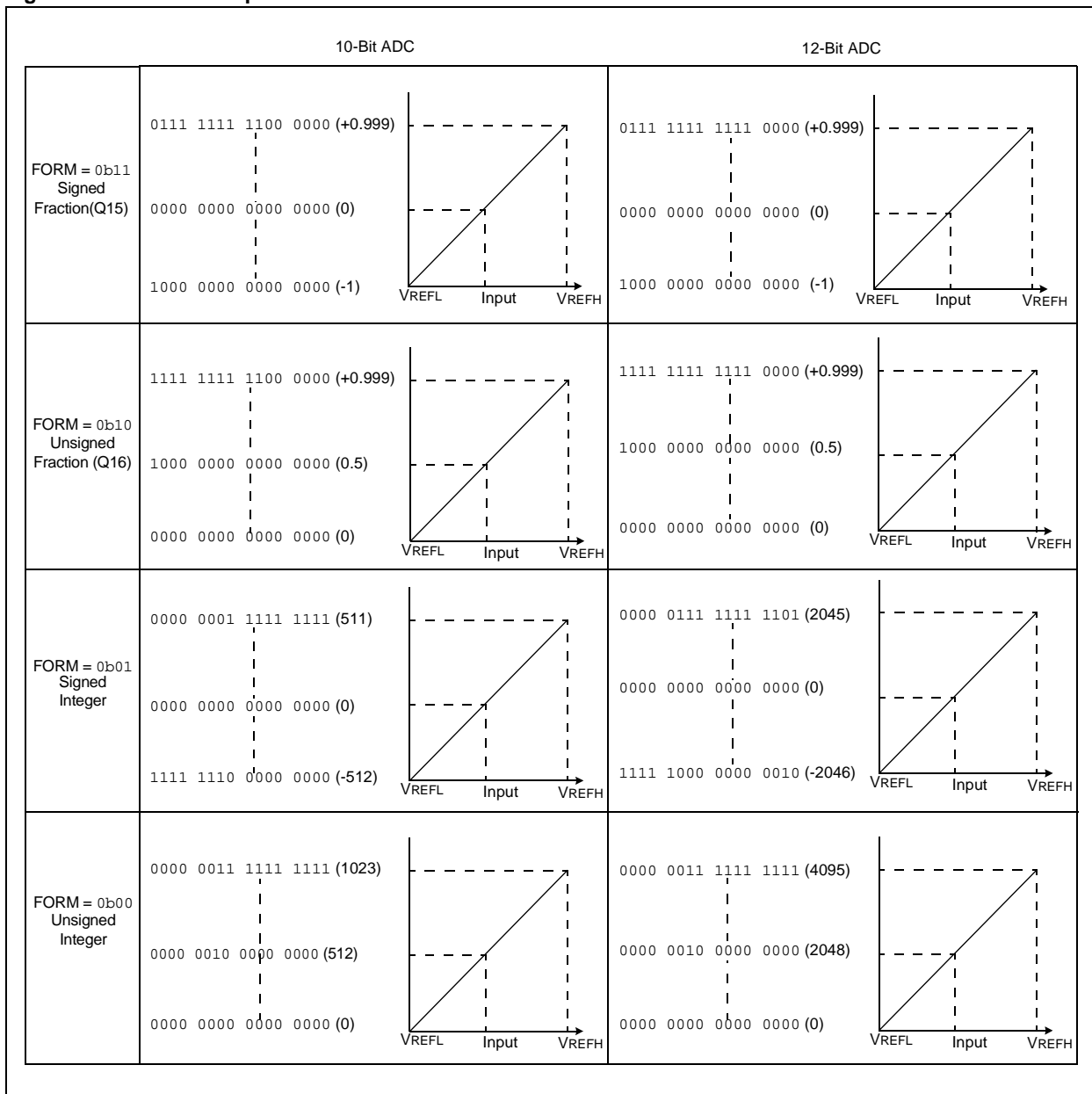
16.4.7 Sample and Conversion Operation (SMPI) Bits

The function of the Samples Per Interrupt control bits (SMPI<4:0>) in the ADC Control Register 2 (ADxCON2<6:2>) for devices with DMA is completely different from the function of the SMPI<4:0> bits for devices without DMA, and for devices with DMA that have the ADC DMA Enable bit (ADDMAEN) clear.

For devices without DMA or with the ADC DMA Enable bit (ADDMAEN) clear, the SMPI<4:0> bits are referred to as the "Number of Samples Per Interrupt Select" bits. For devices with DMA and the ADDMAEN bit set, the SMPI<4:0> bits are referred to as the "Increment Rate for DMA Address Select" bits.

16.4.7.1 SMPIx BITS FOR DEVICES WITHOUT DMA OR WITH THE ADC DMA ENABLE BIT (ADDMAEN) CLEAR

For devices without DMA or with the ADC DMA Enable bit (ADDMAEN) clear, an interrupt can be generated at the end of each sample/convert sequence, or after multiple sample/convert sequences, as determined by the value of the SMPI<4:0> bits. The number of sample/convert sequences between interrupts can vary between 1 and 32. The total number of conversion results between interrupts is the product of the number of channels per sample created by the CHPS<1:0> bits and the value of the SMPI<4:0> bits. See **Section 16.5 "ADC Interrupt Generation"** for the SMPIx values for various sampling modes.

> **Note:** If a manual conversion trigger is used and the number of samples per interrupt is greater than the number of channels per sample, the SAMP bit (ADxCON1<1>) must be manually cleared at suitable intervals in order to generate a sufficient number of ADC conversions.

16.4.7.2 SMPIx BITS FOR DEVICES WITH DMA AND WITH THE ADC DMA ENABLE BIT (ADDMAEN) SET

For devices with DMA and with the ADC DMA Enable bit (ADDMAEN) set, if multiple conversion results need to be buffered, DMA must be used with the ADC module to store the conversion results in a DMA buffer. In this case, the SMPI<4:0> bits are used to select how often the DMA RAM Buffer Pointer is incremented. The number of increments of the DMA RAM Buffer Pointer must not exceed the DMA RAM buffer length per input, as specified by the DMABL<2:0> bits. An ADC interrupt is generated after completion of every conversion, regardless of the setting of the SMPI<4:0> bits.

When single, dual or multiple channels are enabled in Simultaneous or Sequential Sampling modes (and CH0 channel scanning is disabled), the SMPI<4:0> bits are set to '0', indicating the DMA Address Pointer increments every sample.

When all single, dual or multiple channels are enabled in Simultaneous or Sequential Sampling modes with Alternate Input Selection mode enabled (and CH0 channel scanning is disabled), set SMPI<4:0> = 0001 to allow two samples per DMA address point increment.

When channel scanning is used (and Alternate Input Selection mode is disabled), the SMPI<4:0> bits must be set to the number of inputs being scanned, minus one (i.e., SMPI<4:0> = N – 1).

16.4.8 Conversion Trigger Sources

It is often desirable to synchronize the end of sampling and the Start of Conversion with some other time event. The ADC module can use one of the following sources as a conversion trigger:

- External Interrupt Trigger (INT0 only)
- Timer Interrupt Trigger
- Motor Control PWM Special Event Trigger
- PTG Trigger

16.4.8.1 EXTERNAL INTERRUPT TRIGGER (INT0 ONLY)

When SSRCG = 0 and SSRC<2:0> = 001, the Analog-to-Digital conversion is triggered by an active transition on the INT0 pin. The INT0 pin can be programmed for either a rising edge input or a falling edge input.

16.4.8.2 TIMER INTERRUPT TRIGGER

This ADC module Trigger mode is configured by setting SSRCG = 0 and SSRC<2:0> = 010 or 100. When SSRC<2:0> = 010, TMR3 is used to trigger the start of the Analog-to-Digital conversion when a match occurs between the 16-bit Timer Count register (TMR3) and the 16-bit Timer Period register (PR3). The 32-bit timer can also be used to trigger the start of the Analog-to-Digital conversion. When SSRCG = 0 and SSRC<2:0> = 100, TMR5 is used to trigger the start of the Analog-to-Digital conversion when a match occurs between the 16-bit Timer Count Register (TMR5) and the 16-bit Timer Period Register (TPR5).

16.4.8.3 MOTOR CONTROL PWM TRIGGERS

The PWM module has a Special Event Trigger that enables Analog-to-Digital conversions to be synchronized to the PWM time base. When SSRCG = 0 and SSRC<2:0> = 011 or 101, the Analog-to-Digital sampling and conversion times occur at any user programmable point within the PWM period. The Special Event Trigger enables the user to minimize the delay between the time when the Analog-to-Digital conversion results are acquired and the time when the duty cycle value is updated.

Individual PWM event triggers can also be selected for PWM Generators 1 through 7 by setting SSRCG = 1 and SSRC<2:0> = 000, ..., 110.

The application must set the ASAM bit to ensure that the ADC module has sampled the input sufficiently before the next conversion trigger arrives.

16.4.8.4 PTG TRIGGER

The PTG module provides a means to create trigger signals for the ADC and other modules that have complex timing sequences. It offers the user the capability to schedule complex peripheral operations that would be difficult or impossible to achieve through the software solution. When SSRCG = 1 and SSRC = 100, 101 or 110, the PTG module generates a trigger that ends sampling and starts the conversion sequence in the ADC.

The trigger source for the PTG module can vary and depends on the user application. For example, the ADC clock source itself can be used as a trigger source and sets up the PTG to generate a trigger output to the ADC to start the conversion sequence.

16.4.9 Configuring Analog Port Pins

The Analog/Digital Pin Selection register (ANSELy; y = PORTA, PORTB, PORTC, etc.) specifies the input condition of device pins used as analog inputs. Along with the Data Direction register (TRISx) in the Parallel I/O (PIO) port module, these registers control the operation of the ADC pins.

A pin is configured as an analog input when the corresponding ANSy<n> bit (ANSELy<n>) is set. The ANSELy registers are set at Reset, causing the ADC input pins to be configured for analog inputs by default at Reset.

When configured for analog input, the associated port I/O digital input buffer is disabled so that it does not consume current.

The port pins that are desired as analog inputs must have their corresponding TRIS bits set, specifying the port input. If the I/O pin associated with an Analog-to-Digital input is configured as an output, the TRIS bit is cleared and the digital output level (V_{OH} or V_{OL}) of the port is converted.

After a device Reset, all TRIS bits are set.

A pin is configured as a digital I/O when the corresponding ANSy<n> bit is cleared. In this configuration, the input to the analog multiplexer is connected to AV_{SS}.

> **Note 1:** When the ADC PORT register is read, any pin configured as an analog input reads as a '0'.
>
> **2:** Analog levels on any pin that is defined as a digital input may cause the input buffer to consume current that is out of the device specification.

16.4.10 Enabling the ADC Module

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

When ADON is '0', the module is disabled. The digital and analog portions of the circuit are turned off for maximum current savings.

To return to the Active mode from the Off mode, the user application must wait for the analog stages to stabilize. For the stabilization time, refer to the **"Electrical Characteristics"** chapter of the device data sheet.

> **Note:** The SSRCG, SSRC<2:0>, SIMSAM, ASAM, CHPS<1:0>, SMPI<4:0>, BUFM and ALTS bits, as well as the ADCON3 and ADCSSL registers, must not be written to while ADON = 1. This leads to indeterminate results.

16.4.11 Turning the ADC Module Off

Clearing the ADON bit disables the ADC module (stops any scanning, sampling and conversion processes). In this state, the ADC module still consumes some current. Setting the ADxMD bit in the PMD register disables the ADC module and stops the ADC clock source, which reduces device current consumption. Note that setting the ADxMD bit, and then clearing it, resets the ADC module registers to their default state. Additionally, any digital pins that share their function with an ADC input pin revert to the analog function. While the ADxMD bit is set, these pins will be set to digital function.

> **Note:** Clearing the ADON bit during a conversion aborts the current Analog-to-Digital conversion. The ADC buffer is not updated with the partially completed conversion sample.

16.5 ADC INTERRUPT GENERATION

With DMA enabled, the SMPI<4:0> bits (ADxCON2<6:2>) determine the number of sample/conversion operations per channel (CH0/CH1/CH2/CH3) for every DMA Address/Increment Pointer.

The SMPI<4:0> bits have no effect when the ADC module is set up such that DMA buffers are written in Conversion Order mode.

If DMA transfers are enabled, the SMPI<4:0> bits must be cleared, except when channel scanning or alternate sampling is used. Please see **Section 16.7 "Specifying Conversion Results Buffering for Devices with DMA and with ADC DMA Enable Bit (ADDMAEN) Set"** for more details on SMPI<4:0> setup requirements.

When the SIMSAM bit (ADxCON1<3>) specifies sequential sampling, regardless of the number of channels specified by the CHPS<1:0> bits (ADxCON2<9:8>), the ADC module samples once for each conversion and data sample in the buffer. The value specified by the DMAxCNT register for the DMA channel being used corresponds to the number of data samples in the buffer.

For devices with DMA and with the ADC DMA Enable bit (ADDMAEN) set, interrupts are generated after every conversion, which sets the DONE bit since it reflects the ADCx Interrupt Flag (ADxIF) setting.

For devices without DMA or with the ADC DMA Enable bit (ADDMAEN) clear, as conversions are completed, the ADC module writes the results of the conversions into the Analog-to-Digital result buffer. The ADC result buffer is an array of sixteen words, accessed through the SFR space. The user application may attempt to read each Analog-to-Digital conversion result as it is generated. However, this might consume too much CPU time. Generally, to simplify the code, the module fills the buffer with results and generates an interrupt when the buffer is filled. The ADC module supports 16 result buffers. Therefore, the maximum number of conversions per interrupt must not exceed 16.

The number of conversions per ADC interrupt depends on the following parameters, which can vary from one to 16 conversions per interrupt.

• Number of S&H Channels Selected
• Sequential or Simultaneous Sampling
• Samples Convert Sequences Per Interrupt bits (SMPI<4:0>) Settings

Table 16-8 lists the number of conversions per ADC interrupt for different configuration modes.

Table 16-8: Samples Per Interrupt in Alternate Sampling Mode

CHPS<1:0>	SIMSAM	SMPI<4:0>	Conversions/ Interrupt	Description
00	x	N-1	N	1-Channel mode
01	0	N-1	N	2-Channel Sequential Sampling mode
1x	0	N-1	N	4-Channel Sequential Sampling mode
01	1	N-1	2 • N	2-Channel Simultaneous Sampling mode
1x	1	N-1	4 • N	4-Channel Simultaneous Sampling mode

Note 1: In 2-Channel Simultaneous Sampling mode, SMPI<4:0> bit settings must be less than eight.

2: In 4-Channel Simultaneous Sampling mode, SMPI<4:0> bit settings must be less than four.

The DONE bit (ADxCON1<0>) is set when an ADC interrupt is generated to indicate completion of a required sample/conversion sequence. This bit is automatically cleared by the hardware at the beginning of the next sample/conversion sequence.

On devices without DMA or with the ADC DMA Enable bit (ADDMAEN) clear, interrupt generation is based on the SMPI<4:0> and CHPS bits, so the DONE bit is not set after every conversion, but is set when the ADCx Interrupt Flag (ADxIF) is set.

16.5.1 Buffer Fill Mode

When the Buffer Fill Mode bit (BUFM) in the ADC Control Register 2 (ADxCON2<1>) is '1', the 16-word results buffer is split into two 8-word groups: a lower group (ADC1BUF0 through ADC1BUF7) and an upper group (ADC1BUF8 through ADC1BUFF). The 8-word buffers alternately receive the conversion results after each ADC interrupt event. When the BUFM bit is set, each buffer size is equal to eight. Therefore, the maximum number of conversions per interrupt must not exceed eight.

When the BUFM bit is '0', the complete 16-word buffer is used for all conversion sequences. The decision to use the split buffer feature depends on the time available to move the buffer contents, after the interrupt, as determined by the application.

If the application can quickly unload a full buffer within the time taken to sample and convert one channel, the BUFM bit can be '0' and up to 16 conversions may be done per interrupt. The application has one sample/convert time before the first buffer location is overwritten. If the processor cannot unload the buffer within the sample and conversion time, the BUFM bit must be '1'. For example, if an ADC interrupt is generated every eight conversions, the processor has the entire time between interrupts to move the eight conversions out of the buffer.

16.5.2 Buffer Fill Status

When the conversion result buffer is split using the BUFM control bit, the BUFS status bit (ADxCON2<7>) indicates the half of the buffer that the ADC module is currently writing. If BUFS = 0, the ADC module is filling the lower group and the user application should read conversion values from the upper group. If BUFS = 1, the situation is reversed and the user application must read conversion values from the lower group.

16.6   ANALOG INPUT SELECTION FOR CONVERSION

The ADC module provides a flexible mechanism to select analog inputs for conversion:

- Fixed Input Selection
- Alternate Input Selection
- Channel Scanning (CH0 only)

16.6.1   Fixed Input Selection

The 10-bit ADC configuration can use up to four S&H channels, designated CH0-CH3, whereas the 12-bit ADC configuration can use only one S&H channel, CH0. The S&H channels are connected to the analog input pins through the analog multiplexer.

When ALTS = 0, the CH0SA<4:0>, CH0NA, CH123SA and CH123NA<1:0> bits select the analog inputs. Table 16-9 lists the analog inputs and control bits for selecting the channel.

Table 16-9:   Analog Input Selection

		MUXA	
		Control Bits	**Analog Inputs**
CH0	+ve	CH0SA<5:0>	AN0 to AN48
	-ve	CH0NA	V_{REF}-, AN1
CH1	+ve	CH123SA	AN0, AN3
	-ve	CH123NA<1:0>	AN6, AN9, V_{REF}-
CH2	+ve	CH123SA	AN1, AN4, AN0, AN25
	-ve	CH123NA<1:0>	AN7, AN10, V_{REF}-
CH3	+ve	CH123SA	AN2, AN5, AN6, AN25
	-ve	CH123NA<1:0>	AN8, AN11, V_{REF}-

Note: Availability and configuration of inputs varies by device. Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

All four channels can be enabled in Simultaneous or Sequential Sampling modes by configuring the CHPSx bits and the SIMSAM bit.

For devices with DMA and with the ADDMAEN bit set, the SMPI<4:0> bits are set to '00000', indicating the DMA Address Pointer increments every sample.

Example 16-3 shows the code sequence to set up ADC inputs for a 4-channel ADC configuration.

Example 16-3:   Code Sequence to Set Up ADC Inputs

```c
#include <p33Exxxx.h>

/** CONFIGURATION ***************************************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void Delay_us(unsigned int);
int  ADCValues[4] = {0, 0, 0, 0};
int  i;

int  main(void)
{
    // Configure the device PLL to obtain 40 MIPS operation. The crystal frequency is 8 MHz.
    // Divide 8 MHz by 2, multiply by 40 and divide by 2. This results in Fosc of 80 MHz.
    // The CPU clock frequency is Fcy = Fosc/2 = 40 MHz.
    PLLFBD = 38;                        /* M  = 40 */
    CLKDIVbits.PLLPOST = 0;             /* N1 = 2  */
    CLKDIVbits.PLLPRE = 0;              /* N2 = 2  */
    OSCTUN = 0;

    /*Initiate Clock Switch to Primary Oscillator with PLL (NOSC = 0x3) */
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);                 /* Wait for PLL lock at 40 MIPS */

    initAdc1();

    while(1)
    {
      Delay_us(100);                 // Sample for 100 us
      AD1CON1bits.SAMP = 0;          // Start the conversions
      while (!_AD1IF);               // Wait for all 4 conversions to complete
      _AD1IF = 0;                    // Clear conversion done status bit
      ADCValues[0] = ADC1BUF0;       // Read the AN5 conversion result
      ADCValues[1] = ADC1BUF1;       // Read the AN0 conversion result
      ADCValues[2] = ADC1BUF2;       // Read the AN1 conversion result
      ADCValues[3] = ADC1BUF3;       // Read the AN2 conversion result
    }
}

void initAdc1(void)
{
    /* Set port configuration */
    ANSELA = ANSELB = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
    ANSELBbits.ANSB0 = 1;           // Ensure AN0/RB0 is analog
    ANSELBbits.ANSB1 = 1;           // Ensure AN1/RB1 is analog
    ANSELBbits.ANSB2 = 1;           // Ensure AN2/RB2 is analog
    ANSELBbits.ANSB5 = 1;           // Ensure AN5/RB5 is analog

    /* Initialize and enable ADC module */
    AD1CON1 = 0x000C;                    // Enable simultaneous sampling and auto-sample
    AD1CON2 = 0x0300;                    // Sample 4 channels
    AD1CON3 = 0x000F;
    AD1CON4 = 0x0000;
    AD1CSSH = 0x0000;
    AD1CSSL = 0x0000;
    AD1CHS0bits.CH0SA = 5;           // Select AN5 for CH0 +ve input
    AD1CHS0bits.CH0NA = 0;           // Select Vref- for CH0 -ve input
    AD1CHS123bits.CH123SA = 0;       // Select AN0 for CH1 +ve input
                                     // Select AN1 for CH2 +ve input
                                     // Select AN2 for CH3 +ve input
    AD1CHS123bits.CH123NA = 0;       // Select Vref- for CH1/CH2/CH3 -ve inputs
    AD1CON1bits.ADON = 1;
    Delay_us(20);
}

void Delay_us(unsigned int delay)
{
    for (i = 0; i < delay; i++)
    {
      __asm__ volatile ("repeat #39");
      __asm__ volatile ("nop");
    }
}
```

16.6.2 Alternate Input Selection Mode

In an Alternate Input Selection mode, the MUXA and MUXB control bits select the channel for conversion. Table 16-10 lists the analog inputs and control bits for selecting the channel. The ADC completes one sweep using the MUXA selection, and then another sweep using the MUXB selection, and then another sweep using the MUXA selection, and so on. The Alternate Input Selection mode is enabled by setting the Alternate Sample bit (ALTS) in the ADC Control Register 2 (ADxCON2<0>).

The analog input multiplexer is controlled by the AD1CHS123 and AD1CHS0 registers. There are two sets of control bits designated as MUXA (CHySA/CHyNA) and MUXB (CHySB/CHyNB) to select a particular input source for conversion. The MUXB control bits are used in Alternate Input Selection mode.

Table 16-10: Analog Input Selection

		MUXA		MUXB	
		Control Bits	**Analog Inputs**	**Control Bits**	**Analog Inputs**
CH0	+ve	CH0SA<5:0>	AN0 to AN48	CH0SB<5:0>	AN0 to AN48
	-ve	CH0NA	V$_{REF}$-, AN1	CH0NB	AN0 to AN12
CH1	+ve	CH123SA	AN0, AN3	CH123SB	AN0, AN3
	-ve	CH123NA<1:0>	AN6, AN9, V$_{REF}$-	CH123NB<1:0>	AN6, AN9, V$_{REF}$-
CH2	+ve	CH123SA	AN1, AN4, AN0, AN25	CH123SB	AN1, AN4, AN0, AN25
	-ve	CH123NA<1:0>	AN7, AN10, V$_{REF}$-	CH123NB<1:0>	AN7, AN10, V$_{REF}$-
CH3	+ve	CH123SA	AN2, AN5, AN6, AN25	CH123SB	AN2, AN5, AN6, AN25
	-ve	CH123NA<1:0>	AN8, AN11, V$_{REF}$-	CH123NB<1:0>	AN8, AN11, V$_{REF}$-

Note: Availability and configuration of inputs varies by device. Refer to the **"Analog-to-Digital Converter (ADC)"** chapter in the specific device data sheet for availability.

For Alternate Input Selection mode in devices without DMA or with the ADC DMA Enable bit (ADDMAEN) clear, an ADC interrupt must be generated after an even number of sample/conversion sequences by programming the Samples Convert Sequences Per Interrupt bits (SMPI<4:0>). Table 16-11 lists the valid SMPIx values for Alternate Input Selection mode in different ADC configurations.

Table 16-11: Valid SMPIx Values for Alternate Input Selection Mode

CHPS<1:0>	SIMSAM	SMPI<4:0> (Decimal)	Conversions/ Interrupt	Description
00	x	1,3,5,7,9,11,13,15	2,4,6,8,10,12,14,16	1-Channel mode
01	0	3,7,11,15	4,8,12,16	2-Channel Sequential Sampling mode
1x	0	7,15	8,16	4-Channel Sequential Sampling mode
01	1	1,3,5,7	4,8,12,16	2-Channel Simultaneous Sampling mode
1x	1	1,3	8,16	4-Channel Simultaneous Sampling mode

Example 16-4 shows the code sequence to set up the ADC module for Alternate Input Selection mode for devices without DMA in the 4-Channel Simultaneous Sampling configuration. Figure 16-15 illustrates the ADC module operation sequence.

Note: On ADC interrupt, the ADC internal logic is initialized to restart the conversion sequence from the beginning.

Example 16-4: **ADC Code Sequence Setup for Alternate Input Selection Mode for 4-Channel Simultaneous Sampling (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)**

```c
#include <p33Exxxx.h>

/** CONFIGURATION ***********************************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void Delay_us(unsigned int);
int  ADCValues[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int  i;

int  main(void)
{
    // Configure the device PLL to obtain 40 MIPS operation. The crystal frequency is 8 MHz.
    // Divide 8MHz by 2, multiply by 40 and divide by 2. This results in Fosc of 80 MHz.
    // The CPU clock frequency is Fcy = Fosc/2 = 40 MHz.
    PLLFBD = 38;                    /* M  = 40 */
    CLKDIVbits.PLLPOST = 0;         /* N1 = 2  */
    CLKDIVbits.PLLPRE = 0;          /* N2 = 2  */
    OSCTUN = 0;

    /* Initiate Clock Switch to Primary Oscillator with PLL (NOSC = 0x3) */
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);             /* Wait for PLL lock at 40 MIPS */

    initAdc1();

    while(1)
    {
      while (!_AD1IF);              // Wait for all 8 conversions to complete
      _AD1IF = 0;                   // Clear conversion done status bit
      ADCValues[0] = ADC1BUF0;      // Read the AN8 conversion result
      ADCValues[1] = ADC1BUF1;      // Read the AN0 conversion result
      ADCValues[2] = ADC1BUF2;      // Read the AN1 conversion result
      ADCValues[3] = ADC1BUF3;      // Read the AN2 conversion result
      ADCValues[4] = ADC1BUF4;      // Read the AN9 conversion result
      ADCValues[5] = ADC1BUF5;      // Read the AN3 conversion result
      ADCValues[6] = ADC1BUF6;      // Read the AN4 conversion result
      ADCValues[7] = ADC1BUF7;      // Read the AN5 conversion result
    }
}

void initAdc1(void)
{
    /* Set port configuration */
    ANSELA = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
    ANSELB = 0x033F;                // Ensure AN0 - AN5, AN8 and AN9 are analog

    /* Initialize ADC module */
    AD1CON1 = 0x00EC;               // Enable simultaneous sampling, auto-sample and auto-conversion
    AD1CON2 = 0x0305;               // Sample 4 channels at a time, with alternate sampling enabled
    AD1CON3 = 0x0F0F;               // Sample for 15*Tad before triggering conversion
    AD1CON4 = 0x0000;
    AD1CSSH = 0x0000;
    AD1CSSL = 0x0000;

    /* Assign MUXA inputs */
    AD1CHS0bits.CH0SA = 8;          // Select AN8 for CH0 +ve input
    AD1CHS0bits.CH0NA = 0;          // Select VREF- for CH0 -ve input
    AD1CHS123bits.CH123SA = 0;      // Select AN0 for CH1 +ve input
                                    // Select AN1 for CH2 +ve input
                                    // Select AN2 for CH3 +ve input
    AD1CHS123bits.CH123NA = 0;      // Select VREF- for CH1/CH2/CH3 -ve inputs

    /* Assign MUXB inputs */
    AD1CHS0bits.CH0SB = 9;          // Select AN9 for CH0 +ve input
    AD1CHS0bits.CH0NB = 0;          // Select VREF- for CH0 -ve input
    AD1CHS123bits.CH123SB = 1;      // Select AN3 for CH1 +ve input
                                    // Select AN4 for CH2 +ve input
                                    // Select AN5 for CH3 +ve input
    AD1CHS123bits.CH123NB = 0;      // Select VREF- for CH1/CH2/CH3 -ve inputs
    /* Enable ADC module and provide ADC stabilization delay */
    AD1CON1bits.ADON = 1;
    Delay_us(20);
}

void Delay_us(unsigned int delay)
{
    for (i = 0; i < delay; i++)
    {
      __asm__ volatile ("repeat #39");
      __asm__ volatile ("nop");
    }
}
```
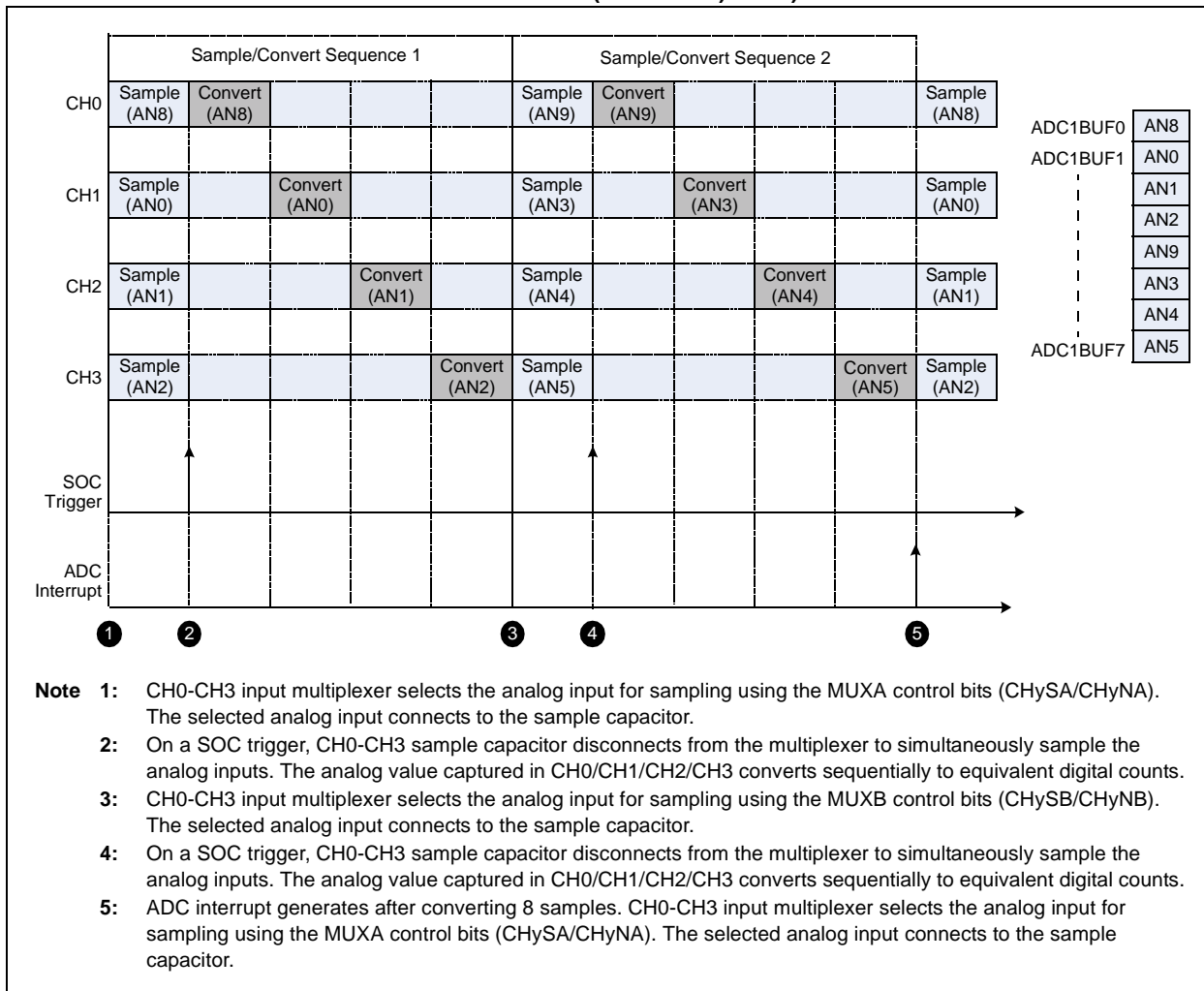
Figure 16-15: **Alternate Input Selection in 4-Channel Simultaneous Sampling Configuration (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)**

Note 1: CH0-CH3 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

2: On a SOC trigger, CH0-CH3 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0/CH1/CH2/CH3 converts sequentially to equivalent digital counts.

3: CH0-CH3 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

4: On a SOC trigger, CH0-CH3 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0/CH1/CH2/CH3 converts sequentially to equivalent digital counts.

5: ADC interrupt generates after converting 8 samples. CH0-CH3 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

Example 16-5 shows the code sequence to set up the ADC module for Alternate Input Selection mode in a 2-channel sequential sampling configuration for devices without DMA. Figure 16-16 shows the ADC operation sequence.

Example 16-5: ADC Code Sequence Setup for Alternate Input Selection for 2-Channel Sequential Sampling (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)

```c
#include <p33Exxxx.h>

/** CONFIGURATION *************************************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void Delay_us(unsigned int);
int  ADCValues[4] = {0, 0, 0, 0};
int  i;

int  main(void)
{
    // Configure the device PLL to obtain 40 MIPS operation. The crystal
    // frequency is 8MHz. Divide 8MHz by 2, multiply by 40 and divide by
    // 2. This results in Fosc of 80MHz. The CPU clock frequency is
    // Fcy = Fosc/2 = 40MHz.
    PLLFBD = 38;                    /* M = 40  */
    CLKDIVbits.PLLPOST = 0;         /* N1 = 2  */
    CLKDIVbits.PLLPRE = 0;          /* N2 = 2  */
    OSCTUN = 0;

    /*       Initiate Clock Switch to Primary
     *       Oscillator with PLL (NOSC= 0x3)*/
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);             /* Wait for PLL lock at 40 MIPS */

    initAdc1();

    while(1)
    {
        while (!_AD1IF);            // Wait for all 4 conversions to complete
        _AD1IF = 0;                // Clear conversion done status bit
        ADCValues[0] = ADC1BUF0;   // Read the AN8 conversion result
        ADCValues[1] = ADC1BUF1;   // Read the AN0 conversion result
        ADCValues[2] = ADC1BUF2;   // Read the AN9 conversion result
        ADCValues[3] = ADC1BUF3;   // Read the AN3 conversion result
    }
}

void initAdc1(void)
{
    /* Set port configuration */
    ANSELA = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
    ANSELB = 0x0309;               // Ensure AN0, AN3, AN8 and AN9 are analog

    /* Initialize ADC module */
    AD1CON1 = 0x00E4;              // Enable sequential sampling, auto-sample and auto-conversion
    AD1CON2 = 0x010D;              // Sample 2 channels, with alternate sampling enabled
    AD1CON3 = 0x0F0F;              // Sample for 15*Tad before triggering conversion
    AD1CON4 = 0x0000;
    AD1CSSH = 0x0000;
    AD1CSSL = 0x0000;

    /* Assign MUXA inputs */
    AD1CHS0bits.CH0SA = 8;         // Select AN8 for CH0 +ve input
    AD1CHS0bits.CH0NA = 0;         // Select Vref- for CH0 -ve input
    AD1CHS123bits.CH123SA = 0;     // Select AN0 for CH1 +ve input
    AD1CHS123bits.CH123NA = 0;     // Select Vref- for CH1/CH2/CH3 -ve inputs

    /* Assign MUXB inputs */
    AD1CHS0bits.CH0SB = 9;         // Select AN9 for CH0 +ve input
    AD1CHS0bits.CH0NB = 0;         // Select Vref- for CH0 -ve input
    AD1CHS123bits.CH123SB = 1;     // Select AN3 for CH1 +ve input
    AD1CHS123bits.CH123NB = 0;     // Select Vref- for CH1/CH2/CH3 -ve inputs

    /* Enable ADC module and provide ADC stabilization delay */
    AD1CON1bits.ADON = 1;
    Delay_us(20);
}
void Delay_us(unsigned int delay)
{
    for (i = 0; i < delay; i++)
    {
        __asm__ volatile ("repeat #39");
        __asm__ volatile ("nop");
    }
}
```
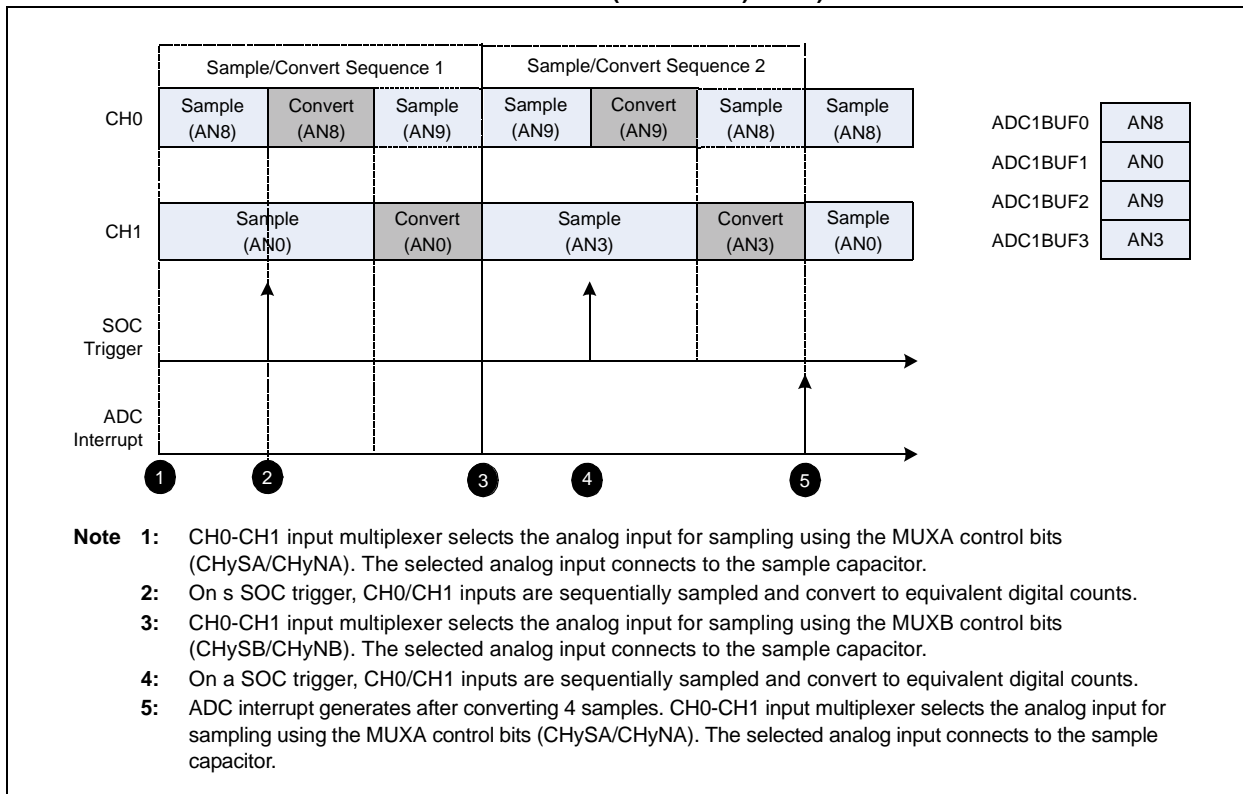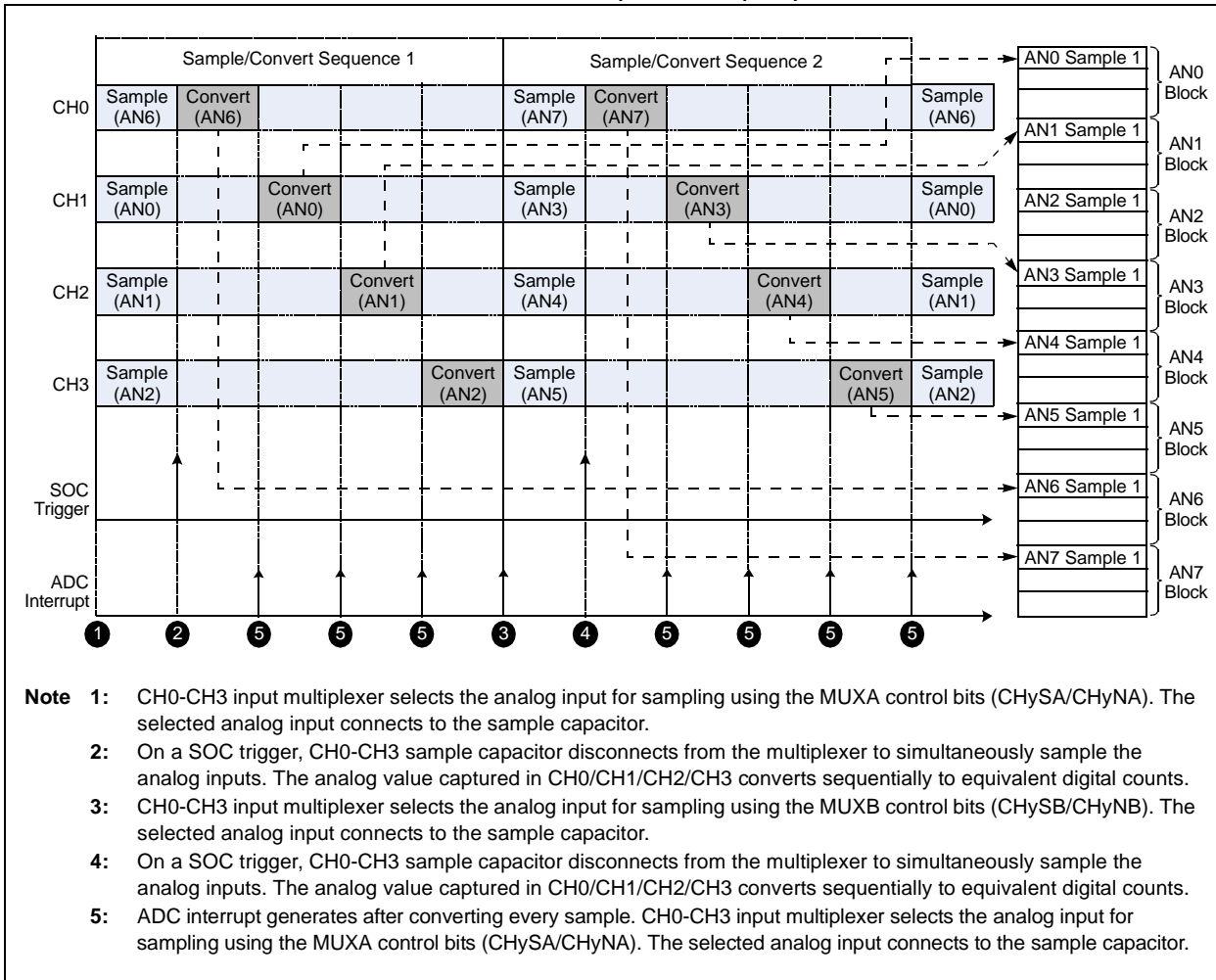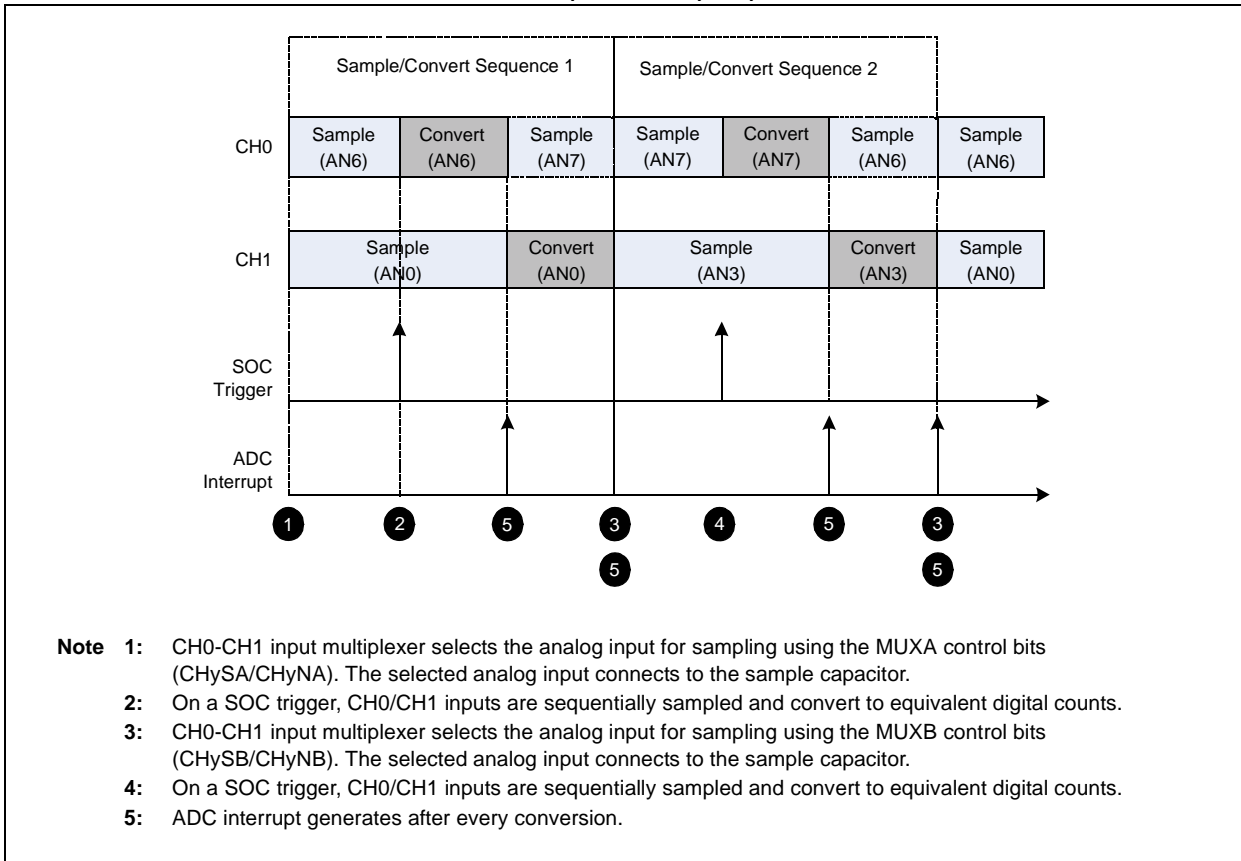
Figure 16-16: **Alternate Input Selection in 2-Channel Sequential Sampling Configuration (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)**

Note 1: CH0-CH1 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

2: On s SOC trigger, CH0/CH1 inputs are sequentially sampled and convert to equivalent digital counts.

3: CH0-CH1 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

4: On a SOC trigger, CH0/CH1 inputs are sequentially sampled and convert to equivalent digital counts.

5: ADC interrupt generates after converting 4 samples. CH0-CH1 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

For devices with DMA and with the ADC DMA Enable bit (ADDMAEN) set, when Alternate Input Selection mode is enabled, set SMPI<4:0> = 00001 to allow two samples per DMA address point increment.

Figure 16-17:   Alternate Input Selection in 4-Channel Simultaneous Sampling Configuration (Devices with DMA and with the ADC DMA Enable Bit (ADDMAEN) Set)

Note 1: CH0-CH3 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

2: On a SOC trigger, CH0-CH3 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0/CH1/CH2/CH3 converts sequentially to equivalent digital counts.

3: CH0-CH3 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

4: On a SOC trigger, CH0-CH3 sample capacitor disconnects from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0/CH1/CH2/CH3 converts sequentially to equivalent digital counts.

5: ADC interrupt generates after converting every sample. CH0-CH3 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

Figure 16-18:   Alternate Input Selection in 2-Channel Sequential Sampling Configuration (Devices with DMA and with the ADC DMA Enable Bit (ADDMAEN) Set)

Note 1: CH0-CH1 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

2: On a SOC trigger, CH0/CH1 inputs are sequentially sampled and convert to equivalent digital counts.

3: CH0-CH1 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

4: On a SOC trigger, CH0/CH1 inputs are sequentially sampled and convert to equivalent digital counts.

5: ADC interrupt generates after every conversion.

16.6.3 Channel Scanning

The ADC module supports the Channel Scanning mode using CH0 (S&H Channel 0). The number of inputs scanned is software-selectable. Any subset of the analog inputs, from AN0 to AN31 (depending on the number of analog inputs present on a specific device), can be selected for conversion. The selected inputs are converted in ascending order. For example, if the input selection includes AN4, AN1 and AN3, the conversion sequence is AN1, AN3 and AN4. The conversion sequence selection is made by programming the ADCx Channel Select register (ADxCSSL). A logic '1' in the ADCx Channel Select register marks the associated analog input channel for inclusion in the conversion sequence. The Channel Scanning mode is enabled by setting the Channel Scan bit (CSCNA) in ADCx Control Register 2 (ADxCON2<10>). In Channel Scanning mode, MUXA software control is ignored and the ADC module sequences through the enabled channels.

In devices without DMA or with the ADC DMA Enable bit (ADDMAEN) clear, for every sample/convert sequence, one analog input is scanned. The ADC interrupt must be generated after all selected channels are scanned. If "N" inputs are enabled for channel scan, an interrupt must be generated after the "N" sample/convert sequence. Table 16-12 lists the SMPIx values to scan "N" analog inputs using CH0 in different ADC configurations.

> **Note:** A maximum of 32 ADC inputs (any) can be configured to be scanned at a time.

Table 16-12: Conversions Per Interrupt in Channel Scanning Mode (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)

CHPS<1:0>	SIMSAM	SMPI<4:0> (Decimal)	Conversions/ Interrupt	Description
00	x	N – 1	N	1-Channel mode
01	0	2N – 1	2N	2-Channel Sequential Sampling mode
1x	0	4N – 1	4N	4-Channel Sequential Sampling mode
01	1	N – 1	2N	2-Channel Simultaneous Sampling mode
1x	1	N – 1	4N	4-Channel Simultaneous Sampling mode

Example 16-6 shows the code sequence to scan four analog inputs using CH0 in devices without DMA or with the ADC DMA Enable bit (ADDMAEN) clear. Figure 16-19 shows the ADC operation sequence.

> **Note:** On ADC interrupt, the ADC internal logic is initialized to restart the conversion sequence from the beginning.

Example 16-6:   Code Sequence to Scan Four Analog Inputs Using CH0 (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)

```c
#include <p33Exxxx.h>

/** CONFIGURATION **************************************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void Delay_us(unsigned int);
int  ADCValues[4] = {0, 0, 0, 0};
int  i;

int  main(void)
{
     // Configure the device PLL to obtain 40 MIPS operation. The crystal frequency is 8 MHz.
     // Divide 8 MHz by 2, multiply by 40 and divide by 2. This results in Fosc of 80 MHz.
     // The CPU clock frequency is Fcy = Fosc/2 = 40 MHz.
     PLLFBD = 38;                          /* M  = 40 */
     CLKDIVbits.PLLPOST = 0;               /* N1 = 2  */
     CLKDIVbits.PLLPRE = 0;                /* N2 = 2  */
     OSCTUN = 0;

     /* Initiate Clock Switch to Primary Oscillator with PLL (NOSC = 0x3) */
     __builtin_write_OSCCONH(0x03);
     __builtin_write_OSCCONL(0x01);
     while (OSCCONbits.COSC != 0x3);
     while (_LOCK == 0);                   /* Wait for PLL lock at 40 MIPS */

     initAdc1();

     while(1)
     {
       while (!_AD1IF);                   // Wait for all 4 conversions to complete
       _AD1IF = 0;                        // Clear conversion done status bit
       ADCValues[0] = ADC1BUF0;           // Read the AN2 conversion result
       ADCValues[1] = ADC1BUF1;           // Read the AN3 conversion result
       ADCValues[2] = ADC1BUF2;           // Read the AN5 conversion result
       ADCValues[3] = ADC1BUF3;           // Read the AN8 conversion result
     }
}

void initAdc1(void)
{
     /* Set port configuration */
     ANSELA = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
     ANSELB = 0x012C;                     // Ensure AN2, AN3, AN5 and AN8 are analog

     /* Initialize ADC module */
     AD1CON1 = 0x04E4;                    // Enable 12-bit mode, auto-sample and auto-conversion
     AD1CON2 = 0x040C;                    // Sample 4 channels alternately using channel scanning
     AD1CON3 = 0x0F0F;                    // Sample for 15*TAD before converting
     AD1CON4 = 0x0000;
     AD1CSSH = 0x0000;
     AD1CSSL = 0x012C;                    // Select AN2, AN3, AN5 and AN8 for scanning

     /* Assign MUXA inputs */
     AD1CHS0bits.CH0SA = 0;               // CH0SA bits ignored for CH0 +ve input selection
     AD1CHS0bits.CH0NA = 0;               // Select VREF- for CH0 -ve input

     /* Enable ADC module and provide ADC stabilization delay */
     AD1CON1bits.ADON = 1;
     Delay_us(20);
}

void Delay_us(unsigned int delay)
{
     for (i = 0; i < delay; i++)
     {
       __asm__ volatile ("repeat #39");
       __asm__ volatile ("nop");
     }
}
```

Figure 16-19: Scan Four Analog Inputs Using CH0 (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)
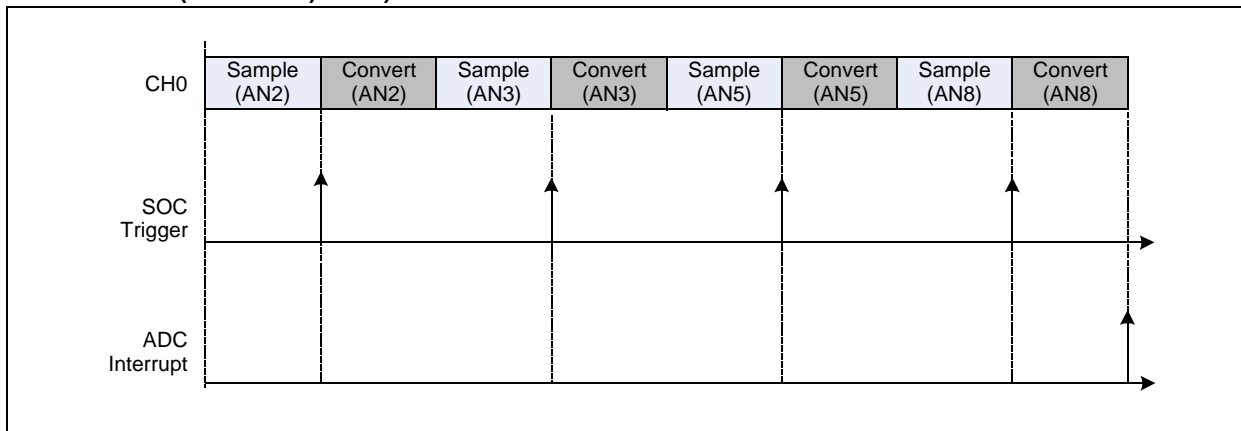
Example 16-7 shows the code sequence to scan two analog inputs using CH0 in a 2-channel alternate input selection configuration for devices without DMA. Figure 16-20 shows the ADC operation sequence.

Example 16-7: Code Sequence for Channel Scan with Alternate Input Selection (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)

```c
#include <p33Exxxx.h>

/** CONFIGURATION **************************************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void Delay_us(unsigned int);
int  ADCValues[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int  i;

int  main(void)
{
    // Configure the device PLL to obtain 40 MIPS operation. The crystal frequency is 8 MHz.
    //Divide 8 MHz by 2, multiply by 40 and divide by 2. This results in Fosc of 80 MHz.
    //The CPU clock frequency is Fcy = Fosc/2 = 40 MHz.
    PLLFBD = 38;                        /* M  = 40 */
    CLKDIVbits.PLLPOST = 0;             /* N1 = 2  */
    CLKDIVbits.PLLPRE = 0;              /* N2 = 2  */
    OSCTUN = 0;

   /*Initiate Clock Switch to Primary Oscillator with PLL (NOSC = 0x3) */
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);                 /* Wait for PLL lock at 40 MIPS */

    initAdc1();

    while(1)
    {
      while (!_AD1IF);                  // Wait for all 8 conversions to complete
      _AD1IF = 0;                       // Clear conversion done status bit
      ADCValues[0] = ADC1BUF0;          // Read the AN2 conversion result
      ADCValues[1] = ADC1BUF1;          // Read the first AN0 conversion result
      ADCValues[2] = ADC1BUF2;          // Read the first AN8 conversion result
      ADCValues[3] = ADC1BUF3;          // Read the first AN3 conversion result
      ADCValues[4] = ADC1BUF4;          // Read the AN4 conversion result
      ADCValues[5] = ADC1BUF5;          // Read the second AN0 conversion result
      ADCValues[6] = ADC1BUF6;          // Read the second AN8 conversion result
      ADCValues[7] = ADC1BUF7;          // Read the second AN3 conversion result
    }
}

void initAdc1(void)
{
    /* Set port configuration */
    ANSELA = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
    ANSELB = 0x011D;              // Ensure AN0, AN2, AN3, AN4 and AN8 are analog

    /* Initialize ADC module */
    AD1CON1 = 0x00E4;             // Enable auto-sample and auto-conversion
    AD1CON2 = 0x051D;             // Select 2-channel mode, enable both scanning and alternate sampling
    AD1CON3 = 0x0F0F;             // Sample for 15 * Tad before converting
    AD1CON4 = 0x0000;
    AD1CSSH = 0x0000;
    AD1CSSL = 0x0014;             // Select AN2 and AN4 for scanning

    /* Assign MUXA inputs */
    AD1CHS0bits.CH0SA = 0;        // CH0SA bits ignored for CH0 +ve input selection
    AD1CHS0bits.CH0NA = 0;        // Select VREF- for CH0 -ve input
    AD1CHS123bits.CH123SA = 0;    // Select AN0 for CH1 +ve input
    AD1CHS123bits.CH123NA = 0;    // Select VREF- for CH1 -ve input

    /* Assign MUXB inputs */
    AD1CHS0bits.CH0SB = 8;        // Select AN8 for CH0 +ve input
    AD1CHS0bits.CH0NB = 0;        // Select VREF- for CH0 -ve input
    AD1CHS123bits.CH123SB = 1;    // Select AN3 for CH1 +ve input
    AD1CHS123bits.CH123NB = 0;    // Select VREF- for CH1 -ve input

    /* Enable ADC module and provide ADC stabilization delay */
    AD1CON1bits.ADON = 1;
    Delay_us(20);
}
void Delay_us(unsigned int delay)
{
    for (i = 0; i < delay; i++)
    {
      __asm__ volatile ("repeat #39");
      __asm__ volatile ("nop");
    }
}
```
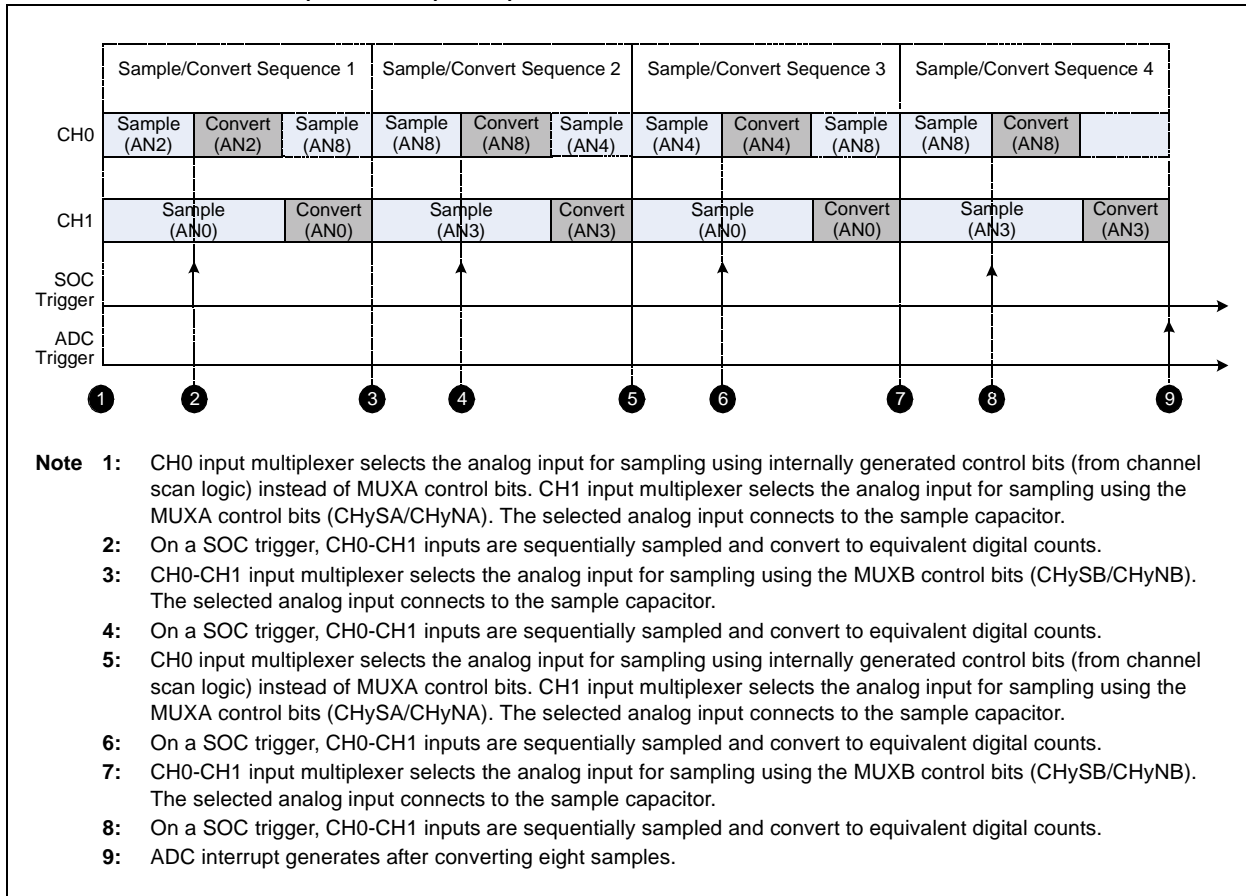
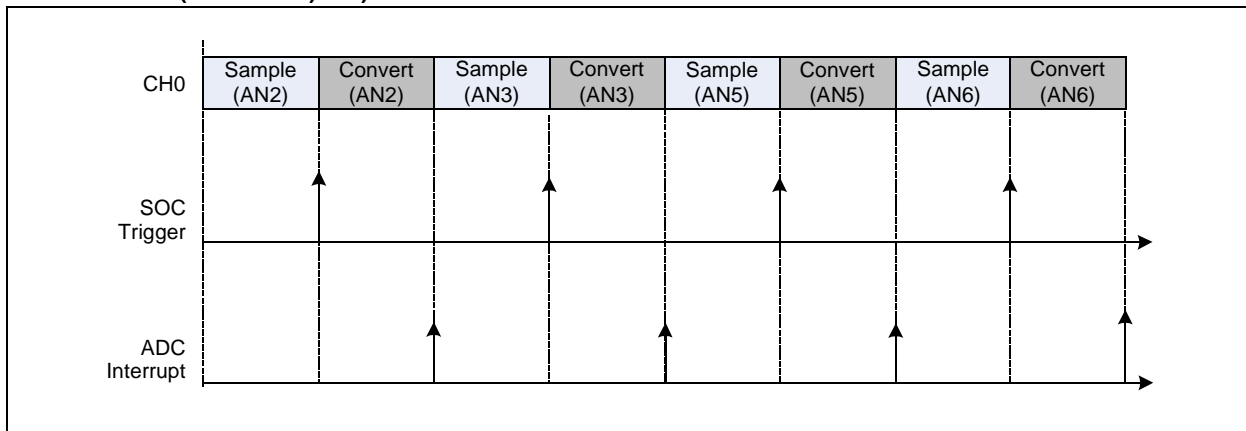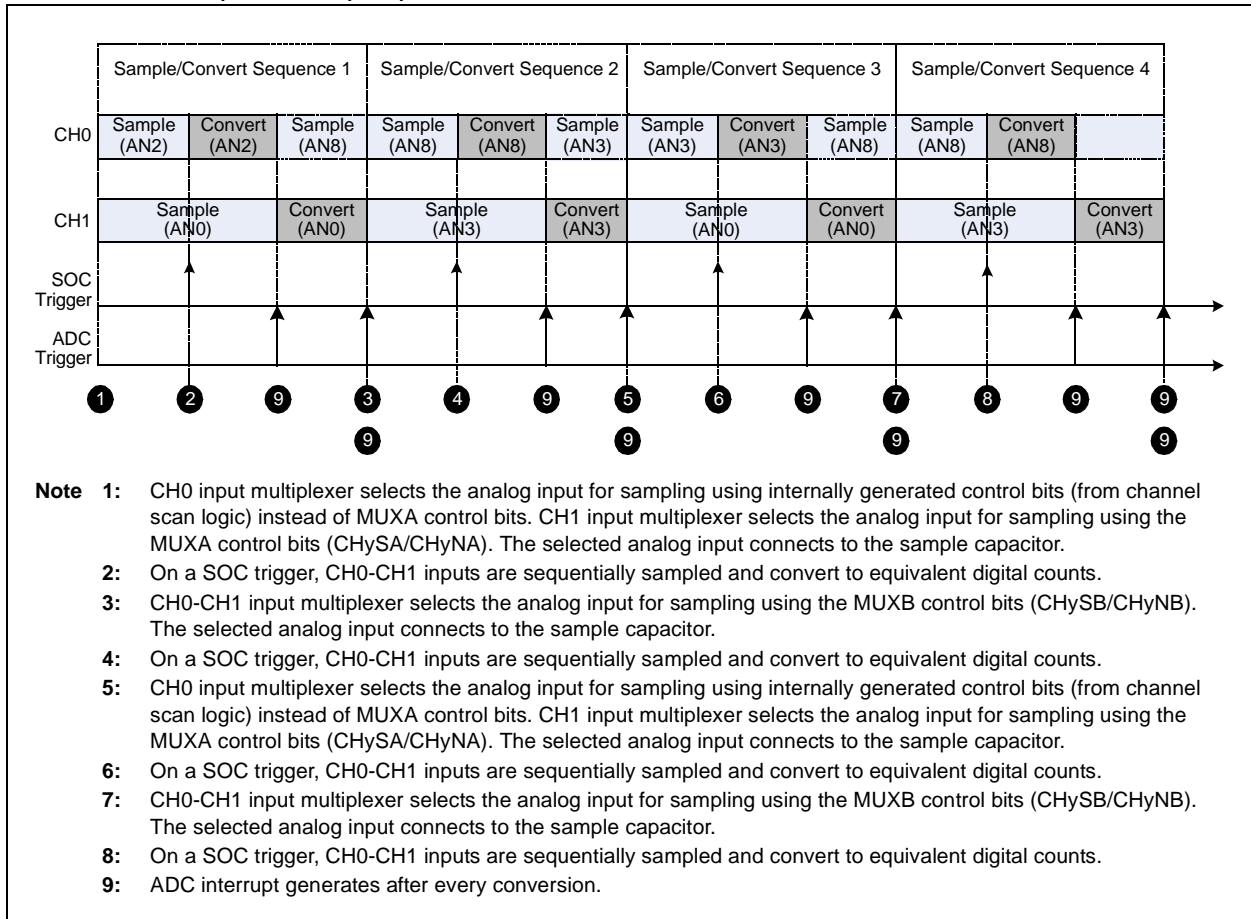**Figure 16-20:   Channel Scan with Alternate Input Selection (Devices without DMA or with the ADC DMA Enable Bit (ADDMAEN) Clear)**

Note   1:   CH0 input multiplexer selects the analog input for sampling using internally generated control bits (from channel scan logic) instead of MUXA control bits. CH1 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

   2:   On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

   3:   CH0-CH1 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

   4:   On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

   5:   CH0 input multiplexer selects the analog input for sampling using internally generated control bits (from channel scan logic) instead of MUXA control bits. CH1 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

   6:   On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

   7:   CH0-CH1 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

   8:   On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

   9:   ADC interrupt generates after converting eight samples.

For devices with DMA and with the ADDMAEN bit set, when channel scanning is used and only CH0 is active (ALTS = 0), the SMPI<4:0> bits must be set to the number of inputs being scanned minus one (i.e., SMPI<4:0> = N – 1).

Figure 16-21:   Scan Four Analog Inputs Using CH0 (Devices with DMA and with the ADC DMA Enable Bit (ADDMAEN) Set)

Figure 16-22: Channel Scan with Alternate Input Selection (Devices with DMA and with the ADC DMA Enable Bit (ADDMAEN) Set)

Note 1: CH0 input multiplexer selects the analog input for sampling using internally generated control bits (from channel scan logic) instead of MUXA control bits. CH1 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

2: On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

3: CH0-CH1 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

4: On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

5: CH0 input multiplexer selects the analog input for sampling using internally generated control bits (from channel scan logic) instead of MUXA control bits. CH1 input multiplexer selects the analog input for sampling using the MUXA control bits (CHySA/CHyNA). The selected analog input connects to the sample capacitor.

6: On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

7: CH0-CH1 input multiplexer selects the analog input for sampling using the MUXB control bits (CHySB/CHyNB). The selected analog input connects to the sample capacitor.

8: On a SOC trigger, CH0-CH1 inputs are sequentially sampled and convert to equivalent digital counts.

9: ADC interrupt generates after every conversion.

16.7 SPECIFYING CONVERSION RESULTS BUFFERING FOR DEVICES WITH DMA AND WITH ADC DMA ENABLE BIT (ADDMAEN) SET

The ADC module contains a single-word, read-only, dual port register (ADCxBUF0), which stores the Analog-to-Digital conversion result. If more than one conversion result needs to be buffered before triggering an interrupt, DMA data transfers can be used. Both ADC channels (ADC1 and ADC2) can trigger a DMA data transfer. Ensure that the ADDMAEN bit is set to use DMA with the ADC module. Depending on which ADC channel is selected as the DMA IRQ source, a DMA transfer occurs when the ADCx Interrupt Flag Status bit (AD1IF or AD2IF) in the Interrupt Flag Status Register x (IFS0 or IFS1, respectively) in the interrupt module gets set as a result of a sample conversion sequence.

The result of every Analog-to-Digital conversion is stored in the ADCxBUF0 register. If a DMA channel is not enabled for the ADC module, each result must be read by the user application before it gets overwritten by the next conversion result. However, if DMA is enabled, multiple conversion results can be automatically transferred from ADCxBUF0 to a user-defined buffer in the DMA RAM area. Thus, the application can process several conversion results with minimal software overhead.

> **Note:** For more information on how to configure a DMA channel to transfer data from the ADC buffer and define a corresponding DMA buffer area from where the data can be accessed by the application, please refer to **Section 22. "Direct Memory Access (DMA)"** (DS70182). For specific information on Interrupt registers, please refer to **Section 6. "Interrupts"** (DS70184).

The DMA Buffer Build Mode bit (ADDMABM) in ADCx Control Register 1 (ADxCON1<12>) determines how the conversion results are filled in the DMA RAM buffer area being used for the ADC. If this bit is set (ADDMABM = 1), DMA buffers are written in the order of conversion. The ADC module provides an address to the DMA channel that is the same as the address used for the non-DMA stand-alone buffer. If the ADDMABM bit is cleared, then DMA buffers are written in Scatter/Gather mode. The ADC module provides a Scatter/Gather mode address to the DMA channel, based on the index of the analog input and the size of the DMA buffer.

When the SIMSAM bit specifies simultaneous sampling, the number of data samples in the buffer is related to the CHPS<1:0> bits. Algorithmically, the Channels per Sample (CH/S) times the number of samples results in the number of data sample entries in the buffer. To avoid loss of data in the buffer due to overruns, the DMAxCNT register must be set to the desired buffer size.

16.7.1 Using DMA in the Scatter/Gather Mode

When the ADDMABM bit is '0', the Scatter/Gather mode is enabled. In this mode, the DMA channel must be configured for Peripheral Indirect Addressing. The DMA buffer is divided into consecutive memory blocks corresponding to all available analog inputs (out of AN0-AN31). Each conversion result for a particular analog input is automatically transferred by the ADC module to the corresponding block within the user-defined DMA buffer area. Successive samples for the same analog input are stored in sequence within the block assigned to that input.

The number of samples that need to be stored in the DMA buffer for each analog input is specified by the DMABL<2:0> bits (ADxCON4<2:0>).

The buffer locations within each block are accessed by the ADC module using an internal pointer, which is initialized to '0' when the ADC module is enabled. When this internal pointer reaches the value defined by the DMABL<2:0> bits, it gets reset to '0'. This ensures that the conversion results of one analog input do not corrupt the conversion results of other analog inputs. The rate at which this internal pointer is incremented when data is written to the DMA buffer is specified by the SMPI<4:0> bits.

When no channel scanning or alternate sampling is required, SMPI<4:0> must be cleared, implying that the pointer increments on every sample per channel. Thus, it is theoretically possible to use every location in the DMA buffer for the blocks assigned to the analog inputs being sampled.

In the example illustrated in Figure 16-23, it can be observed that the conversion results for the AN0, AN1 and AN2 inputs are stored in sequence, leaving no unused locations in their corresponding memory blocks. However, for the four analog inputs (AN4, AN5, AN6 and AN7), which are scanned by CH0, the first location in the AN5 block, the first two locations in the AN6 block and the first three locations in the AN7 block are unused, resulting in a relatively inefficient arrangement of data in the DMA buffer.

When scanning is used, and no simultaneous sampling is performed (SIMSAM = 0), SMPI<4:0> must be set to one less than the number of inputs being scanned. For example, if CHPS<1:0> = 00 (only one S&H channel is used), and AD1CSSL = 0xFFFF, indicating that AN0-AN15 are being scanned, then set SMPI<4:0> = 01111 so that the internal pointer is incremented only after every sixteenth sample/conversion sequence. This avoids unused locations in the blocks corresponding to the analog inputs being scanned.

Similarly, if ALTS = 1, indicating that alternating analog input selections are used, then SMPI<4:0> is set to '00001', thereby incrementing the internal pointer after every second sample.

> **Note:** The ADC module does not perform limit checks on the generated buffer addresses. For example, you must ensure that the Least Significant bits (LSbs) of the DMAxSTA or DMAxSTB register used are indeed '0'. Also, the number of potential analog inputs multiplied by the buffer size, specified by DMABL<2:0>, must not exceed the total length of the DMA buffer.

Figure 16-23:   DMA Buffer in Scatter/Gather Mode
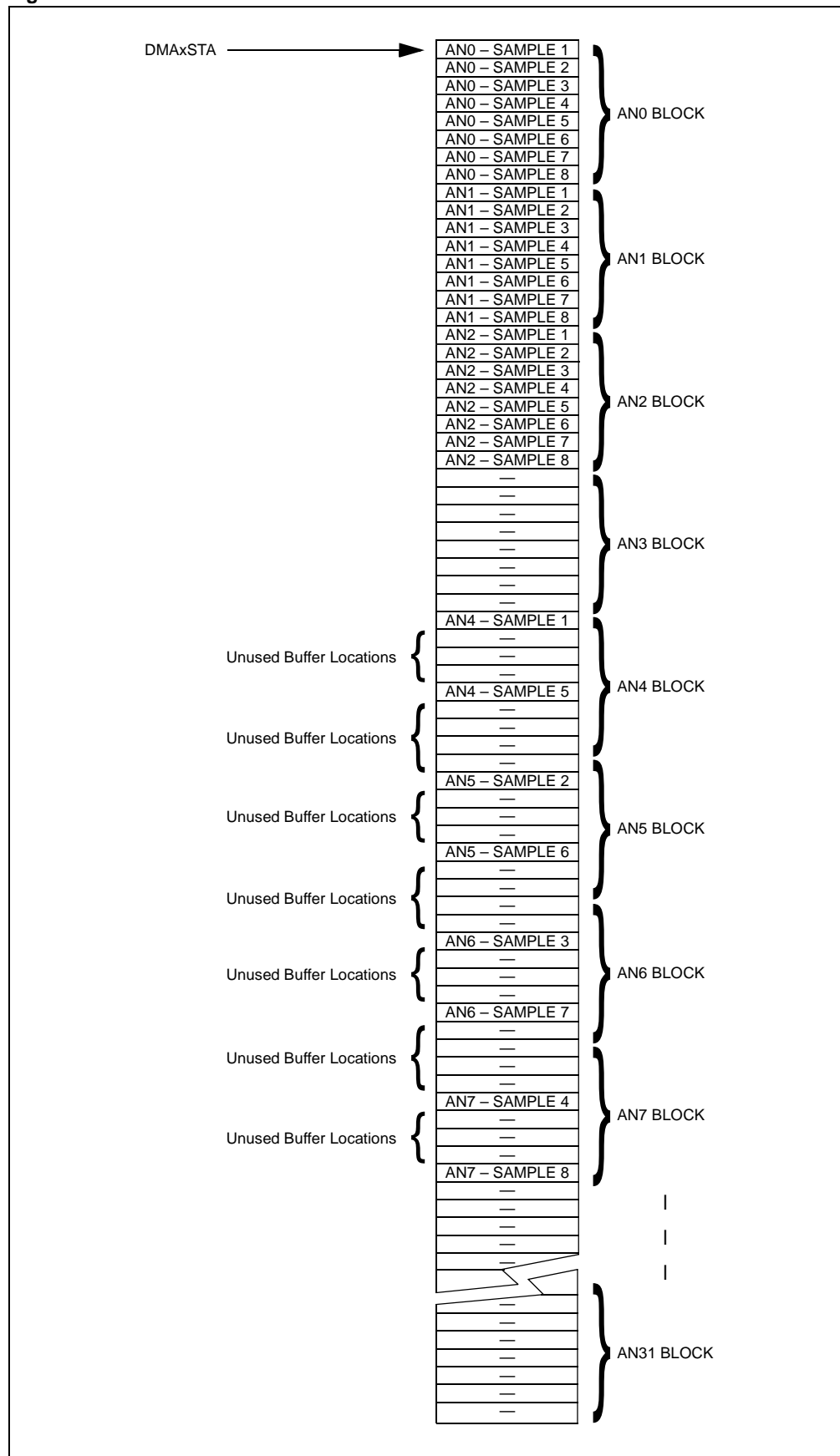
16.7.2   Using DMA in the Conversion Order Mode

When the ADDMABM bit (ADxCON1<12>) = 1, the Conversion Order mode is enabled. In this mode, the DMA channel can be configured for Register Indirect or Peripheral Indirect Addressing mode. All conversion results are stored in the user-specified DMA buffer area in the same order in which the conversions are performed by the ADC module. In this mode, the buffer is not divided into blocks allocated to different analog inputs; rather, the conversion results from different inputs are interleaved according to the specific Buffer Fill modes being used.

In this configuration, the Buffer Pointer is always incremented by one word. In this case, the SMPI<4:0> bits (ADxCON2<6:2>) must be cleared and the DMABL<2:0> bits (ADxCON4<2:0>) are ignored.

Figure 16-24 illustrates an example identical to the configuration in Figure 16-23, but using the Conversion Order mode. In this example, the DMAxCNT register has been configured to generate the DMA interrupt after 16 conversion results have been obtained.

Figure 16-24:   DMA Buffer in Conversion Order Mode

DMAxSTA →

AN4 – SAMPLE 1
AN0 – SAMPLE 1
AN1 – SAMPLE 1
AN2 – SAMPLE 1
AN5 – SAMPLE 2
AN0 – SAMPLE 2
AN1 – SAMPLE 2
AN2 – SAMPLE 2
AN6 – SAMPLE 3
AN0 – SAMPLE 3
AN1 – SAMPLE 3
AN2 – SAMPLE 3
AN7 – SAMPLE 4
AN0 – SAMPLE 4
AN1 – SAMPLE 4
AN2 – SAMPLE 4
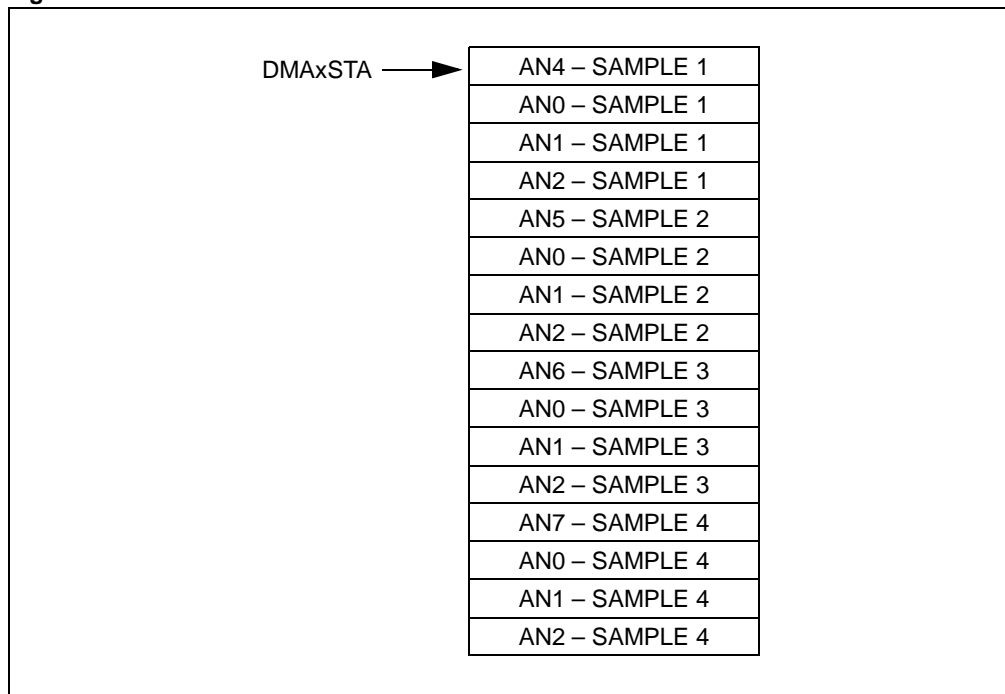
16.8   ADC CONFIGURATION EXAMPLE

The following steps are used for performing an Analog-to-Digital conversion:

1. Select 10-bit or 12-bit mode (ADxCON1<10>).
2. Select the voltage reference source to match the expected range on the analog inputs (ADxCON2<15:13>).
3. Select the analog conversion clock to match the desired data rate with the processor clock (ADxCON3<7:0>).
4. Determine how inputs must be allocated to S&H channels (ADxCHS0<15:0> and ADxCHS123<15:0>).
5. Determine how many S&H channels must be used (ADxCON2<9:8>).
6. Determine how sampling must occur (ADxCON1<3>, ADxCSSH<15:0> and ADxCSSL<15:0>).
7. Select manual or auto-sampling.
8. Select the conversion trigger and sampling time.
9. Select how the data format for the conversion results must be stored in the buffer (ADxCON1<9:8>).
10. Set the ADDMAEN bit to configure the ADC module to use DMA.
11. Select the interrupt rate or DMA Buffer Pointer increment rate (ADxCON2<9:5>).
12. Select the number of samples in the DMA buffer for each ADC module input (ADxCON4<2:0>).
13. Configure the ADC interrupt (if required):
    a) Clear the ADxIF bit
    b) Select the interrupt priority (ADxIP<2:0)
    c) Set the ADxIE bit
14. Configure the DMA channel (if needed).
15. Enable the DMA channel.
16. Turn on the ADC module (ADxCON1<15>).

The options for these configuration steps are described in subsequent sections.

16.9   ADC CONFIGURATION FOR 1.1 Msps

When the device is running at an operating frequency of 40 MIPS, for example, the ADC module can be configured to sample at a 1.1 Msps throughput rate with 10-bit resolution.

The ADC module is set to 10-bit operation by setting the AD12B bit to '0' (ADxCON1<10>). The ASAM bit (ADxCON1<2>) is set to '1' to begin sampling automatically after the conversion completes. The internal counter, which ends sampling and starts conversion, is set as the sample clock source by setting SSRCG (ADxCON1<4>) = 0 and the SSRC<2:0> bits (ADxCON1<7:5>) = 111. The system clock is selected to be the ADC conversion clock by setting the ADRC bit (ADxCON3<15>) to '0'. The automatic sample time bit is set to less than 12 T_{AD}. The ADC conversion clock is configured to 75 ns by setting the ADCS<7:0> bits (ADxCON3<7:0>) to '00000010', as calculated in Equation 16-7.

Equation 16-7:   ADC Conversion Clock

$$T_{AD} = T_{CY} \bullet (ADCS{<}7{:}0{>} + 1) = (1/40M) \bullet 3 = 75 \text{ ns } (13.3 \text{ MHz})$$

The ADC conversion time will be 12 T_{AD} since the ADC module is configured for 10-bit operation, as calculated in Equation 16-8.

Equation 16-8:   ADC Conversion Time

$$T_{CONV} = 12 \bullet T_{AD} = 900 \text{ ns } (1.1 \text{ MHz})$$

The ADC channels, CH0 and CH1 (CHPS<1:0> = 01), are set up to convert analog input, AN0 or AN3 (only one at any time), in Sequential mode (SIMSAM = 0). Figure 16-25 illustrates the sampling sequence.

Figure 16-25:   Sampling Sequence for 1.1 Msps

Note: The 'x' in ANx is either 0 or 3. T is 900 ns and the frequency is 1.1 Msps.
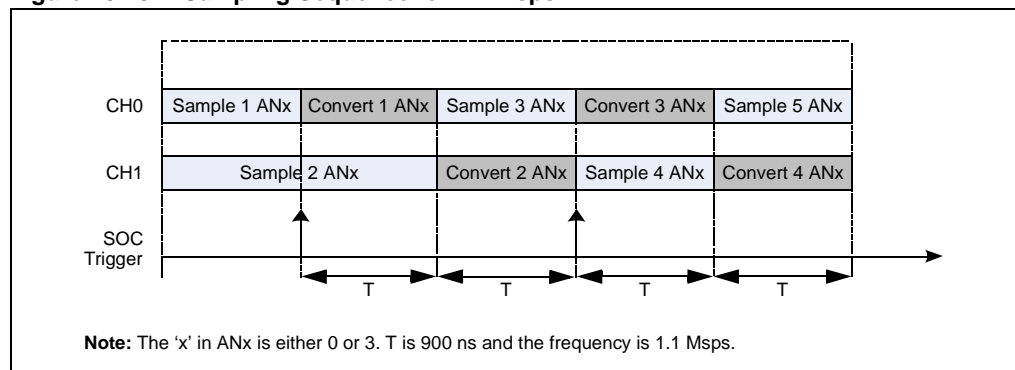
For devices with DMA, the DMA channel can be configured in Ping-Pong mode to move the converted data from the ADC to the DMA RAM. See the ADC and DMA configuration code in Example 16-8.

For devices without DMA, the ADC configuration remains the same. The samples are transferred to ADC1BUF0-ADC1BUFF at a rate of 1.1 Msps. The data can be processed by accessing half of the buffers at a time by setting the BUFS bit.

> **Note:**   The ADC module cannot achieve maximum throughput of 1.1 Msps at the maximum operating frequency of 60 MIPS.

Example 16-8:   ADC Configuration Code for 1.1 Msps

```c
#include <p33Exxxx.h>

/** CONFIGURATION *************************************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);

void initAdc1(void);
void initDma0(void);
void Delay_us(unsigned int);
int  BufferA[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int  BufferB[8] = {0, 0, 0, 0, 0, 0, 0, 0};
int  i;

int  main(void)
{
    // Configure the device PLL to obtain 40 MIPS operation. The crystal frequency is 8 MHz.
    // Divide 8 MHz by 2, multiply by 40 and divide by 2. This results in Fosc of 80 MHz.
    // The CPU clock frequency is Fcy = Fosc/2 = 40 MHz.
    PLLFBD = 38;                        /* M = 40 */
    CLKDIVbits.PLLPOST = 0;             /* N1 = 2  */
    CLKDIVbits.PLLPRE = 0;             /* N2 = 2  */
    OSCTUN = 0;

    /* Initiate Clock Switch to Primary Oscillator with PLL (NOSC = 0x3) */
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);                 /* Wait for PLL lock at 40 MIPS */

    initAdc1();
    initDma0();

    while(1);                          /* Wait for DMA interrupts to occur */
}

void initAdc1(void)
{
    /* Set port configuration */
    ANSELA = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
    ANSELB = 0x0001;      // Ensure AN0 is analog

    /* Initialize ADC module */
    AD1CON1 = 0x13E4;       // DMA Conversion Order, sequential sampling, 10-bit, Signed Fractional
    AD1CON2 = 0x0100;      // Select 2-channel mode, increment DMA pointer after every sample
    AD1CON3bits.ADRC = 0; // ADC Clock is derived from System Clock
    AD1CON3bits.SAMC = 2; // Sample for 2 * Tad before converting
    AD1CON3bits.ADCS = 2; // TAD = TCY * (ADCS + 1) = (1 / 40MHz) * 3 = 75 ns (13.3 MHz)
                          // ADC conversion time for 10-bit Tconv = 12 * Tad = 900 ns (1.1 Msps)
    AD1CON4 = 0x0100;      // Use DMA to store conversion results
    AD1CSSH = 0x0000;
    AD1CSSL = 0x0000;

    /* Assign MUXA inputs */
    AD1CHS0bits.CH0SA = 0;       // Select AN0 for CH0 +ve input
    AD1CHS0bits.CH0NA = 0;       // Select VREF- for CH0 -ve input
    AD1CHS123bits.CH123SA = 0;   // Select AN0 for CH1 +ve input
    AD1CHS123bits.CH123NA = 0;   // Select VREF- for CH1 -ve input

    /* Enable ADC module and provide ADC stabilization delay */
    AD1CON1bits.ADON = 1;
    Delay_us(20);
}

void __attribute__((interrupt, auto_psv)) _DMA0Interrupt(void)
{
    _DMA0IF = 0;                       /* Clear DMA interrupt flag to prepare for next block */
}

void Delay_us(unsigned int delay)
{
    for (i = 0; i < delay; i++)
    {
      __asm__ volatile ("repeat #39");
      __asm__ volatile ("nop");
    }
}
```

16.10   SAMPLE AND CONVERSION SEQUENCE EXAMPLES FOR DEVICES WITHOUT DMA AND FOR DEVICES WITH DMA BUT WITH ADC DMA ENABLE BIT (ADDMAEN) CLEAR

The following configuration examples show the Analog-to-Digital operation in different sampling and buffering configurations. In each example, setting the ASAM bit starts automatic sampling. A conversion trigger ends sampling and starts conversion.

> **Note:** These examples are based on devices without op amps. Availability and configuration of inputs varies by device. **Refer to the "Op Amp/Comparator"** chapter in the specific device data sheet to determine availability.

16.10.1   Sampling and Converting a Single Channel Multiple Times

Figure 16-26 and Table 16-13 illustrate a basic configuration of the ADC. In this case, one ADC input, AN0, is sampled by one S&H channel, CH0, and converted. The results are stored in the ADC buffer (ADC1BUF0-ADC1BUFF). This process repeats 16 times until the buffer is full and then the ADC module generates an interrupt. The entire process then repeats.

The CHPSx bits specify that only S&H CH0 is active. With ALTS clear, only the MUXA inputs are active. The CH0SAx and CH0NA bits are specified (AN0-V_{REF-}) as the input to the S&H channel. All other input selection bits are not used.

Figure 16-26:   Converting One Channel, 16 Times/Interrupt
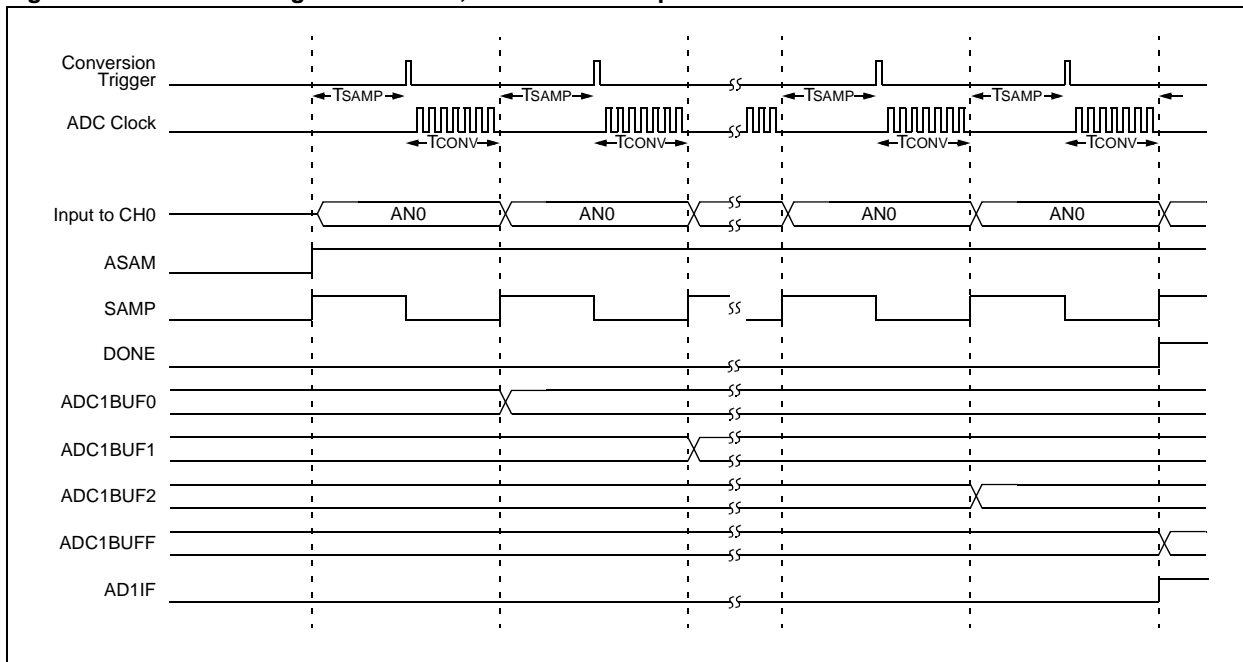
Table 16-13: Converting One Channel, 16 Times per ADC Interrupt

CONTROL BITS	OPERATION SEQUENCE
Sequence Select	Sample MUXA Inputs: AN0 ≥ CH0

CONTROL BITS	OPERATION SEQUENCE
Sequence Select	
SMPI<4:0> = 01111	Sample MUXA Inputs: AN0 ≥ CH0
Interrupt on 16th Sample	Convert CH0, Write ADC1BUF0
CHPS<1:0> = 00	Sample MUXA Inputs: AN0 ≥ CH0
Sample Channel CH0	Convert CH0, Write ADC1BUF1
SIMSAM = n/a	Sample MUXA Inputs: AN0 ≥ CH0
Not Applicable for Single Channel Sample	Convert CH0, Write ADC1BUF2
BUFM = 0	Sample MUXA Inputs: AN0 ≥ CH0
Single 16-Word Result Buffer	Convert CH0, Write ADC1BUF3
ALTS = 0	Sample MUXA Inputs: AN0 ≥ CH0
Always Use MUXA Input Select	Convert CH0, Write ADC1BUF4
ADDMAEN = 0	Sample MUXA Inputs: AN0 ≥ CH0
Do Not Use DMA with ADC	Convert CH0, Write ADC1BUF5
MUXA Input Select	Sample MUXA Inputs: AN0 ≥ CH0
CH0SA<5:0> = 000000	Convert CH0, Write ADC1BUF6
Select AN0 for CH0+ Input	Sample MUXA Inputs: AN0 ≥ CH0
CH0NA = 0	Convert CH0, Write ADC1BUF7
Select VREF- for CH0- Input	Sample MUXA Inputs: AN0 ≥ CH0
CSCNA = 0	Convert CH0, Write ADC1BUF8
No Input Scan	Sample MUXA Inputs: AN0 ≥ CH0
CSS<15:0> = n/a	Convert CH0, Write ADC1BUF9
Scan Input Select Unused	Sample MUXA Inputs: AN0 ≥ CH0
CH123SA<2:0> = n/a	Convert CH0, Write ADC1BUFA
Channels CH1, CH2, CH3+ Inputs Unused	Sample MUXA Inputs: AN0 ≥ CH0
CH123NA<1:0> = n/a	Convert CH0, Write ADC1BUFB
Channels CH1, CH2, CH3- Inputs Unused	Sample MUXA Inputs: AN0 ≥ CH0
MUXB Input Select	Convert CH0, Write ADC1BUFC
CH0SB<5:0> = n/a	Sample MUXA Inputs: AN0 ≥ CH0
Channel CH0+ Input Unused	Convert CH0, Write ADC1BUFD
CH0NB = n/a	Sample MUXA Inputs: AN0 ≥ CH0
Channel CH0- Input Unused	Convert CH0, Write ADC1BUFE
CH123SB<2:0> = n/a	Sample MUXA Inputs: AN0 ≥ CH0
Channels CH1, CH2, CH3+ Inputs Unused	Convert CH0, Write ADC1BUFF
CH123NB<1:0> = n/a	ADC Interrupt
Channels CH1, CH2, CH3- Inputs Unused	**Repeat**

ADC Buffer @ First ADC Interrupt		ADC Buffer @ Second ADC Interrupt
ADC1BUF0	AN0 Sample 1	AN0 Sample 17
ADC1BUF1	AN0 Sample 2	AN0 Sample 18
ADC1BUF2	AN0 Sample 3	AN0 Sample 19
ADC1BUF3	AN0 Sample 4	AN0 Sample 20
ADC1BUF4	AN0 Sample 5	AN0 Sample 21
ADC1BUF5	AN0 Sample 6	AN0 Sample 22
ADC1BUF6	AN0 Sample 7	AN0 Sample 23
ADC1BUF7	AN0 Sample 8	AN0 Sample 24
ADC1BUF8	AN0 Sample 9	AN0 Sample 25
ADC1BUF9	AN0 Sample 10	AN0 Sample 26
ADC1BUFA	AN0 Sample 11	AN0 Sample 27
ADC1BUFB	AN0 Sample 12	AN0 Sample 28
ADC1BUFC	AN0 Sample 13	AN0 Sample 29
ADC1BUFD	AN0 Sample 14	AN0 Sample 30
ADC1BUFE	AN0 Sample 15	AN0 Sample 31
ADC1BUFF	AN0 Sample 16	AN0 Sample 32

16.10.2   Analog-to-Digital Conversions While Scanning Through
16 Analog Inputs

Figure 16-27 and Table 16-14 illustrate a typical setup where all available analog input channels are sampled by one S&H channel, CH0, and converted. The Set Scan Input Selection bit (CSCNA) in the ADC Control Register 2 (ADxCON2<10>) specifies scanning of the ADC inputs to the CH0 positive input. Other conditions are similar to those described in **Section 16.10.1 "Sampling and Converting a Single Channel Multiple Times"**.

Initially, the AN0 input is sampled by CH0 and converted, and then the AN1 input is sampled and converted. This process of scanning the inputs repeats 16 times until the buffer is full. The result is stored in the ADC buffer (ADC1BUF0-ADC1BUFF). Then, the ADC module generates an interrupt. The entire process then repeats.

Figure 16-27:   Scanning Through 16 Inputs/Interrupt
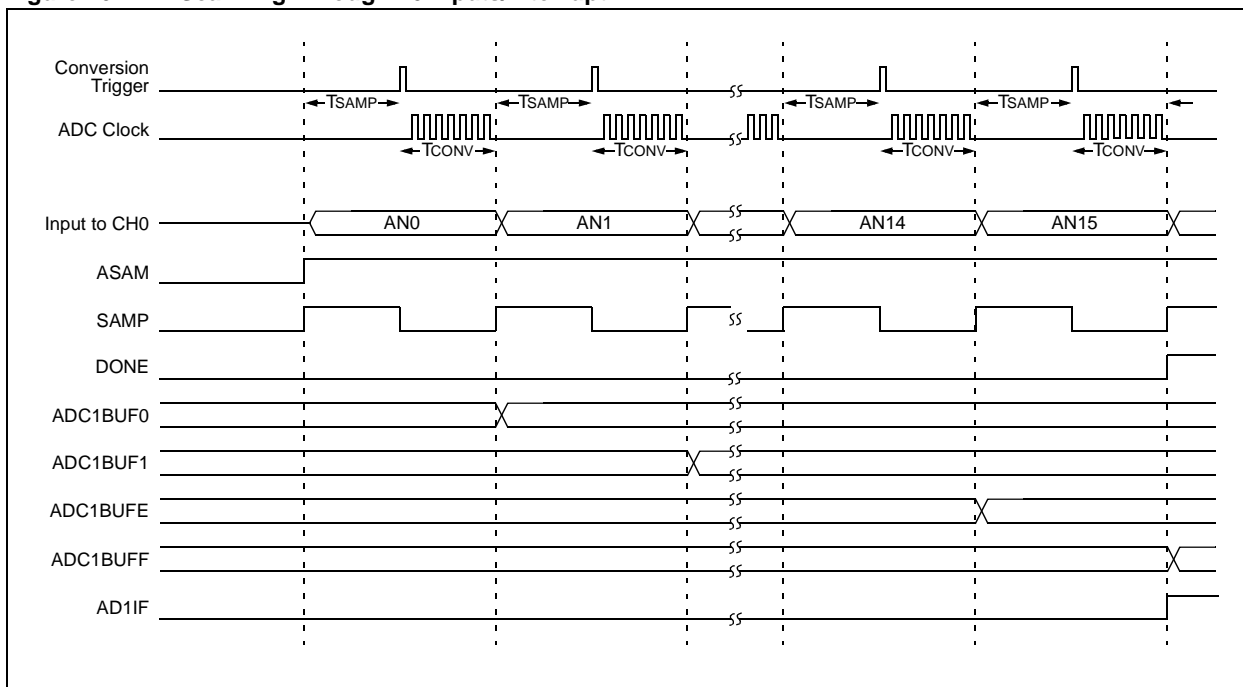
Table 16-14:   Scanning Through 16 Inputs per ADC Interrupt

CONTROL BITS	OPERATION SEQUENCE
Sequence Select	Sample MUXA Inputs: AN0 $\geq$ CH0
SMPI<4:0> = 01111	Convert CH0, Write ADC1BUF0
Interrupt on 16th Sample	Sample MUXA Inputs: AN1 $\geq$ CH0
CHPS<1:0> = 00	Convert CH0, Write ADC1BUF1
Sample Channel CH0	Sample MUXA Inputs: AN2 $\geq$ CH0
SIMSAM = n/a	Convert CH0, Write ADC1BUF2
Not Applicable for Single Channel Sample	Sample MUXA Inputs: AN3 $\geq$ CH0
BUFM = 0	Convert CH0, Write ADC1BUF3
Single 16-Word Result Buffer	Sample MUXA Inputs: AN4 $\geq$ CH0
ALTS = 0	Convert CH0, Write ADC1BUF4
Always Use MUXA Input Select	Sample MUXA Inputs: AN5 $\geq$ CH0
ADDMAEN = 0	Convert CH0, Write ADC1BUF5
Do Not Use DMA with ADC	Sample MUXA Inputs: AN6 $\geq$ CH0
MUXA Input Select	Convert CH0, Write ADC1BUF6
CH0SA<5:0> = n/a	Sample MUXA Inputs: AN7 $\geq$ CH0
Override by CSCNA	Convert CH0, Write ADC1BUF7
CH0NA = 0	Sample MUXA Inputs: AN8 $\geq$ CH0
Select V$_{REF}$- for CH0- Input	Convert CH0, Write ADC1BUF8
CSCNA = 1	Sample MUXA Inputs: AN9 $\geq$ CH0
Scan CH0+ Inputs	Convert CH0, Write ADC1BUF9
CSS<15:0> = 1111 1111 1111 1111	Sample MUXA Inputs: AN10 $\geq$ CH0
Scan Input Select Unused	Convert CH0, Write ADC1BUFA
CH123SA<2:0> = n/a	Sample MUXA Inputs: AN11 $\geq$ CH0
Channels CH1, CH2, CH3+ Inputs Unused	Convert CH0, Write ADC1BUFB
CH123NA<1:0> = n/a	Sample MUXA Inputs: AN12 $\geq$ CH0
Channels CH1, CH2, CH3- Inputs Unused	Convert CH0, Write ADC1BUFC
MUXB Input Select	Sample MUXA Inputs: AN13 $\geq$ CH0
CH0SB<5:0> = n/a	Convert CH0, Write ADC1BUFD
Channel CH0+ Input Unused	Sample MUXA Inputs: AN14 $\geq$ CH0
CH0NB = n/a	Convert CH0, Write ADC1BUFE
Channel CH0- Input Unused	Sample MUXA Inputs: AN15 $\geq$ CH0
CH123SB<2:0> = n/a	Convert CH0, Write ADC1BUFF
Channels CH1, CH2, CH3+ Inputs Unused	ADC Interrupt
CH123NB<1:0> = n/a	**Repeat**
Channels CH1, CH2, CH3- Inputs Unused	

ADC Buffer @ First ADC Interrupt		ADC Buffer @ Second ADC Interrupt
ADC1BUF0	AN0 Sample 1	AN0 Sample 17
ADC1BUF1	AN1 Sample 2	AN1 Sample 18
ADC1BUF2	AN2 Sample 3	AN2 Sample 19
ADC1BUF3	AN3 Sample 4	AN3 Sample 20
ADC1BUF4	AN4 Sample 5	AN4 Sample 21
ADC1BUF5	AN5 Sample 6	AN5 Sample 22
ADC1BUF6	AN6 Sample 7	AN6 Sample 23
ADC1BUF7	AN7 Sample 8	AN7 Sample 24
ADC1BUF8	AN8 Sample 9	AN8 Sample 25
ADC1BUF9	AN9 Sample 10	AN9 Sample 26
ADC1BUFA	AN10 Sample 11	AN10 Sample 27
ADC1BUFB	AN11 Sample 12	AN11 Sample 28
ADC1BUFC	AN12 Sample 13	AN12 Sample 29
ADC1BUFD	AN13 Sample 14	AN13 Sample 30
ADC1BUFE	AN14 Sample 15	AN14 Sample 31
ADC1BUFF	AN15 Sample 16	AN15 Sample 32

16.10.3 Sampling Three Inputs Frequently While Scanning Four Other Inputs

Figure 16-28 and Table 16-15 illustrate how the ADC module could be configured to sample three inputs frequently, using S&H channels, CH1, CH2 and CH3; while four other inputs are sampled less frequently by scanning them, using S&H channel, CH0. In this case, only MUXA inputs are used and all four channels are sampled simultaneously. Four different inputs (AN4, AN5, AN6, AN7) are scanned in CH0, whereas AN0, AN1 and AN2 are the fixed inputs for CH1, CH2 and CH3, respectively. Thus, in every set of 16 samples, AN0, AN1 and AN2 are sampled four times, while AN4, AN5, AN6 and AN7 are sampled only once each.

Figure 16-28: Converting Three Inputs, Four Times and Four Inputs, One Time/Interrupt
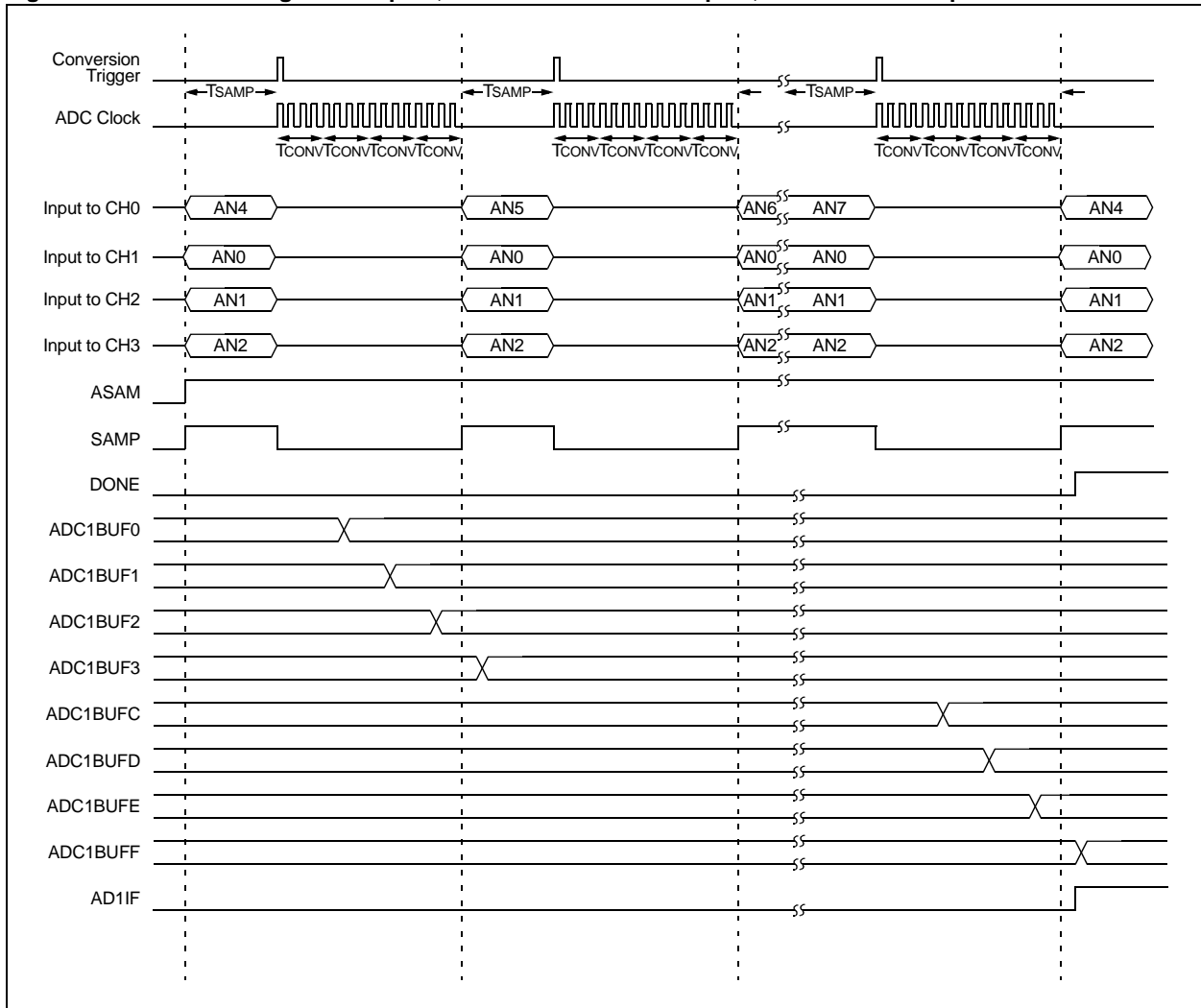
Table 16-15: Converting Three Inputs, Four Times and Four Inputs, One Time per ADC Interrupt

CONTROL BITS	OPERATION SEQUENCE

Sequence Select

SMPI<4:0> = 00011 Interrupt on 4th Sample[1]	
CHPS<1:0> = 1x Sample Channels CH0, CH1, CH2, CH3	
SIMSAM = 1 Sample All Channels Simultaneously	
BUFM = 0 Single 16-Word Result Buffer	
ALTS = 0 Always Use MUXA Input Select	
ADDMAEN = 0 Do Not Use DMA with ADC	

MUXA Input Select

CH0SA<5:0> = n/a   Override by CSCNA
CH0NA = 0   Select VREF- for CH0- Input
CSCNA = 1   Scan CH0+ Inputs
CSS<15:0> = 0000 0000 1111 0000   Scan AN4, AN5, AN6, AN7
CH123SA<2:0> = 000   CH1+ = AN0, CH2+ = AN1, CH3+ = AN2
CH123NA<1:0> = 0x   CH1-, CH2-, CH3- = VREF-

MUXB Input Select

CH0SB<5:0> = n/a   Channel CH0+ Input Unused
CH0NB = n/a   Channel CH0- Input Unused
CH123SB<2:0> = n/a   Channels CH1, CH2, CH3+ Inputs Unused
CH123NB<1:0> = n/a   Channels CH1, CH2, CH3- Inputs Unused

OPERATION SEQUENCE

Sample MUXA Inputs: AN4 ≥ CH0, AN0 ≥ CH1, AN1 ≥ CH2, AN2 ≥ CH3
Convert CH0, Write ADC1BUF0
Convert CH1, Write ADC1BUF1
Convert CH2, Write ADC1BUF2
Convert CH3, Write ADC1BUF3
Sample MUXA Inputs: AN5 ≥ CH0, AN0 ≥ CH1, AN1 ≥ CH2, AN2 ≥ CH3
Convert CH0, Write ADC1BUF4
Convert CH1, Write ADC1BUF5
Convert CH2, Write ADC1BUF6
Convert CH3, Write ADC1BUF7
Sample MUXA Inputs: AN6 ≥ CH0, AN0 ≥ CH1, AN1 ≥ CH2, AN2 ≥ CH3
Convert CH0, Write ADC1BUF8
Convert CH1, Write ADC1BUF9
Convert CH2, Write ADC1BUFA
Convert CH3, Write ADC1BUFB
Sample MUXA Inputs: AN7 ≥ CH0, AN0 ≥ CH1, AN1 ≥ CH2, AN2 ≥ CH3
Convert CH0, Write ADC1BUFC
Convert CH1, Write ADC1BUFD
Convert CH2, Write ADC1BUFE
Convert CH3, Write ADC1BUFF
ADC Interrupt
Repeat

ADC Buffer @ First ADC Interrupt		ADC Buffer @ Second ADC Interrupt
ADC1BUF0	AN4 Sample 1	AN4 Sample 2
ADC1BUF1	AN0 Sample 1	AN0 Sample 5
ADC1BUF2	AN1 Sample 1	AN1 Sample 5
ADC1BUF3	AN2 Sample 1	AN2 Sample 5
ADC1BUF4	AN5 Sample 1	AN5 Sample 2
ADC1BUF5	AN0 Sample 2	AN0 Sample 6
ADC1BUF6	AN1 Sample 2	AN1 Sample 6
ADC1BUF7	AN2 Sample 2	AN2 Sample 6
ADC1BUF8	AN6 Sample 1	AN6 Sample 2
ADC1BUF9	AN0 Sample 3	AN0 Sample 7
ADC1BUFA	AN1 Sample 3	AN1 Sample 7
ADC1BUFB	AN2 Sample 3	AN2 Sample 7
ADC1BUFC	AN7 Sample 1	AN7 Sample 2
ADC1BUFD	AN0 Sample 4	AN0 Sample 8
ADC1BUFE	AN1 Sample 4	AN1 Sample 8
ADC1BUFF	AN2 Sample 4	AN2 Sample 8

Note 1: During this simultaneous sampling, one sample and four conversions are treated as one sample and convert sequence. Therefore, when SMPI<4:0> = 00011, an ADC interrupt is generated after 16 samples are converted and buffered in ADC1BUF0-ADC1BUFF.

16.10.4 Using Alternating MUXA, MUXB Input Selections

Figure 16-29 and Table 16-16 demonstrate alternate sampling of the inputs assigned to MUXA and MUXB. In this example, two channels are enabled to sample simultaneously. Setting the ALTS bit (ADCxCON2<0>) enables alternating input selections. The first sample uses the MUXA inputs specified by the CH0SA<5:0>, CH0NA, CH123SA<2:0> and CH123NA<1:0> bits. The next sample uses the MUXB inputs specified by the CH0SB<5:0>, CH0NB, CH123SB<2:0> and CH123NB<1:0> bits. In this example, one of the MUXB input specifications uses two analog inputs as a differential source to the S&H, sampling (AN3-AN9).

Using four S&H channels without alternating input selections results in the same number of conversions as this example, using two channels with alternating input selections. However, because the CH1, CH2 and CH3 channels are more limited in the selection of the analog inputs, this example method provides more flexibility of input selection than using four channels.

Figure 16-29:   Converting Two Sets of Two Inputs Using Alternating Input Selections
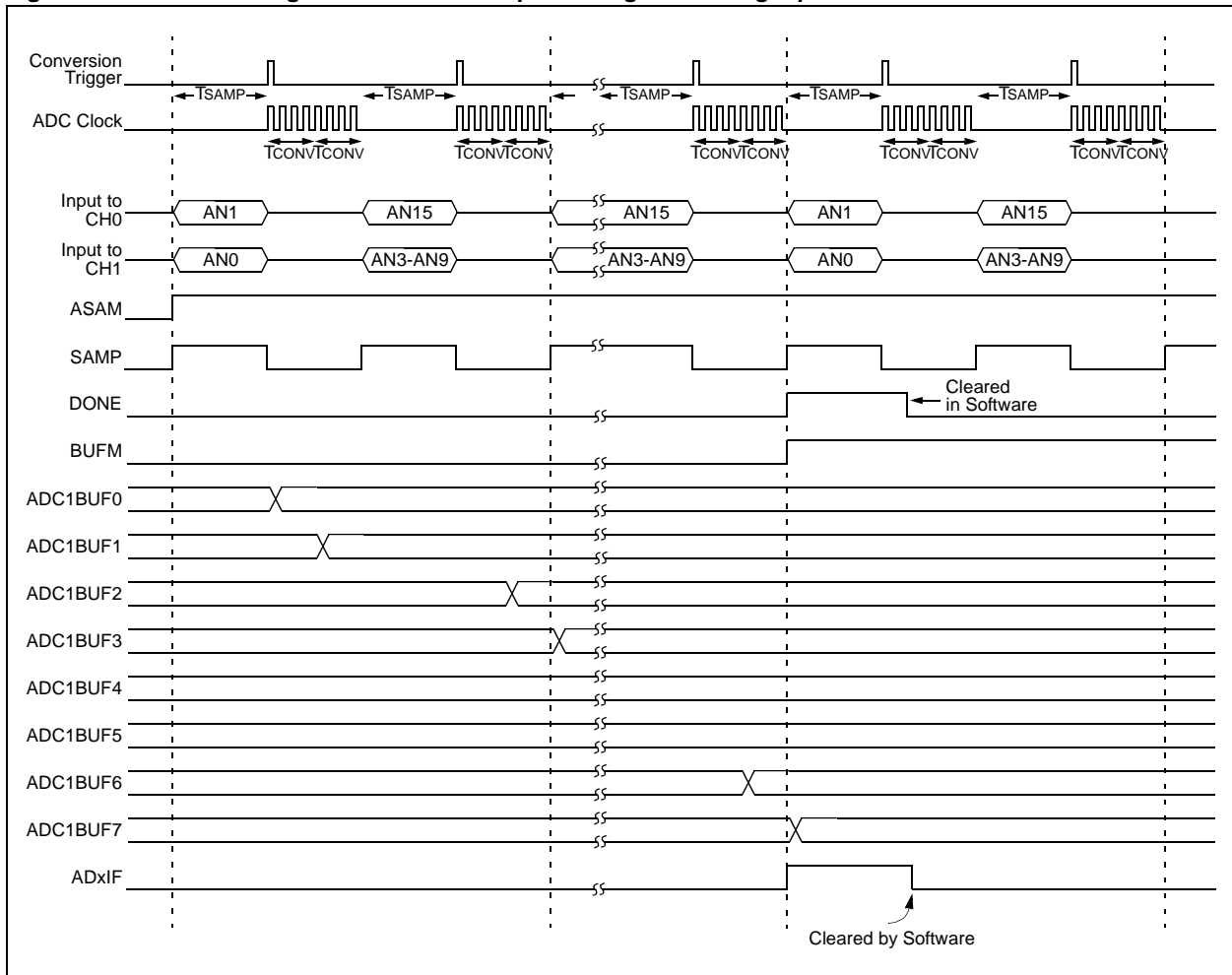
Table 16-16: Converting Two Sets of Two Inputs Using Alternating Input Selections

CONTROL BITS	OPERATION SEQUENCE
Sequence Select	Sample MUXA Inputs: AN1 ≥ CH0, AN0 ≥ CH1
SMPI<4:0> = 00011 Interrupt on 4th Sample	Convert CH0, Write ADC1BUF0
	Convert CH1, Write ADC1BUF1
CHPS<1:0> = 01 Sample Channels CH0, CH1	Sample MUXB Inputs: AN15 ≥ CH0, (AN3-AN9) ≥ CH1
	Convert CH0, Write ADC1BUF2
SIMSAM = 1 Sample All Channels Simultaneously	Convert CH1, Write ADC1BUF3
	Sample MUXA Inputs: AN1 ≥ CH0, AN0 ≥ CH1
BUFM = 1 Dual 8-Word Result Buffers	Convert CH0, Write ADC1BUF4
	Convert CH1, Write ADC1BUF5
ALTS = 1 Alternate MUXA/B Input Select	Sample MUXB Inputs: AN15 ≥ CH0, (AN3-AN9) ≥ CH1
	Convert CH0, Write ADC1BUF6
ADDMAEN = 0 Do Not Use DMA with ADC	Convert CH1, Write ADC1BUF7
	Interrupt; Change Buffer
MUXA Input Select	Sample MUXA Inputs: AN1 ≥ CH0, AN0 ≥ CH1
CH0SA<5:0> = 000001 Select AN1 for CH0+ Input	Convert CH0, Write ADC1BUF8
	Convert CH1, Write ADC1BUF9
CH0NA = 0 Select VREF- for CH0- Input	Sample MUXB Inputs: AN15 ≥ CH0, (AN3-AN9) ≥ CH1
	Convert CH0, Write ADC1BUFA
CSCNA = 0 No Input Scan	Convert CH1, Write ADC1BUFB
	Sample MUXA Inputs: AN1 ≥ CH0, AN0 ≥ CH1
CSS<15:0> = n/a Scan Input Select Unused	Convert CH0, Write ADC1BUFC
	Convert CH1, Write ADC1BUFD
CH123SA<2:0> = 000 CH1+ = AN0, CH2+ = AN1, CH3+ = AN2	Sample MUXB Inputs: AN15 ≥ CH0, (AN3-AN9) ≥ CH1
	Convert CH0, Write ADC1BUFE
CH123NA<1:0> = 0x CH1-, CH2-, CH3- = VREF-	Convert CH1, Write ADC1BUFF
	ADC Interrupt; Change Buffer
MUXB Input Select	**Repeat**
CH0SB<5:0> = 111111 Select AN15 for CH0+ Input	
CH0NB = 0 Select VREF- for CH0- Input	
CH123SB<2:0> = 111 CH1+ = AN3, CH2+ = AN4, CH3+ = AN5	
CH123NB<1:0> = 11 CH1- = AN9, CH2- = AN10, CH3- = AN11	

	ADC Buffer @ First ADC Interrupt	**ADC Buffer @ Second ADC Interrupt**
ADC1BUF0	AN1 Sample 1	
ADC1BUF1	AN0 Sample 1	
ADC1BUF2	AN15 Sample 2	
ADC1BUF3	(AN3-AN9) Sample 2	
ADC1BUF4	AN1 Sample 3	
ADC1BUF5	AN0 Sample 3	
ADC1BUF6	AN15 Sample 4	
ADC1BUF7	(AN3-AN9) Sample 4	
ADC1BUF8		AN1 Sample 5
ADC1BUF9		AN0 Sample 5
ADC1BUFA		AN15 Sample 6
ADC1BUFB		(AN3-AN9) Sample 6
ADC1BUFC		AN1 Sample 7
ADC1BUFD		AN0 Sample 7
ADC1BUFE		AN15 Sample 8
ADC1BUFF		(AN3-AN9) Sample 8

16.10.5 Sampling Eight Inputs Using Simultaneous Sampling

Figure 16-30 and Table 16-17 demonstrate identical setups with the exception that this example uses simultaneous sampling (SIMSAM = 1), and Figure 16-31 uses sequential sampling (SIMSAM = 0). Both examples use alternating inputs and specify differential inputs to the S&H.

Figure 16-30 and Table 16-17 demonstrate simultaneous sampling. When converting more than one channel and selecting simultaneous sampling, the ADC module samples all channels, then performs the required conversions in sequence. In this example, with the ASAM bit set, sampling begins after the conversion is complete.

Figure 16-30:   Sampling Eight Inputs Using Simultaneous Sampling
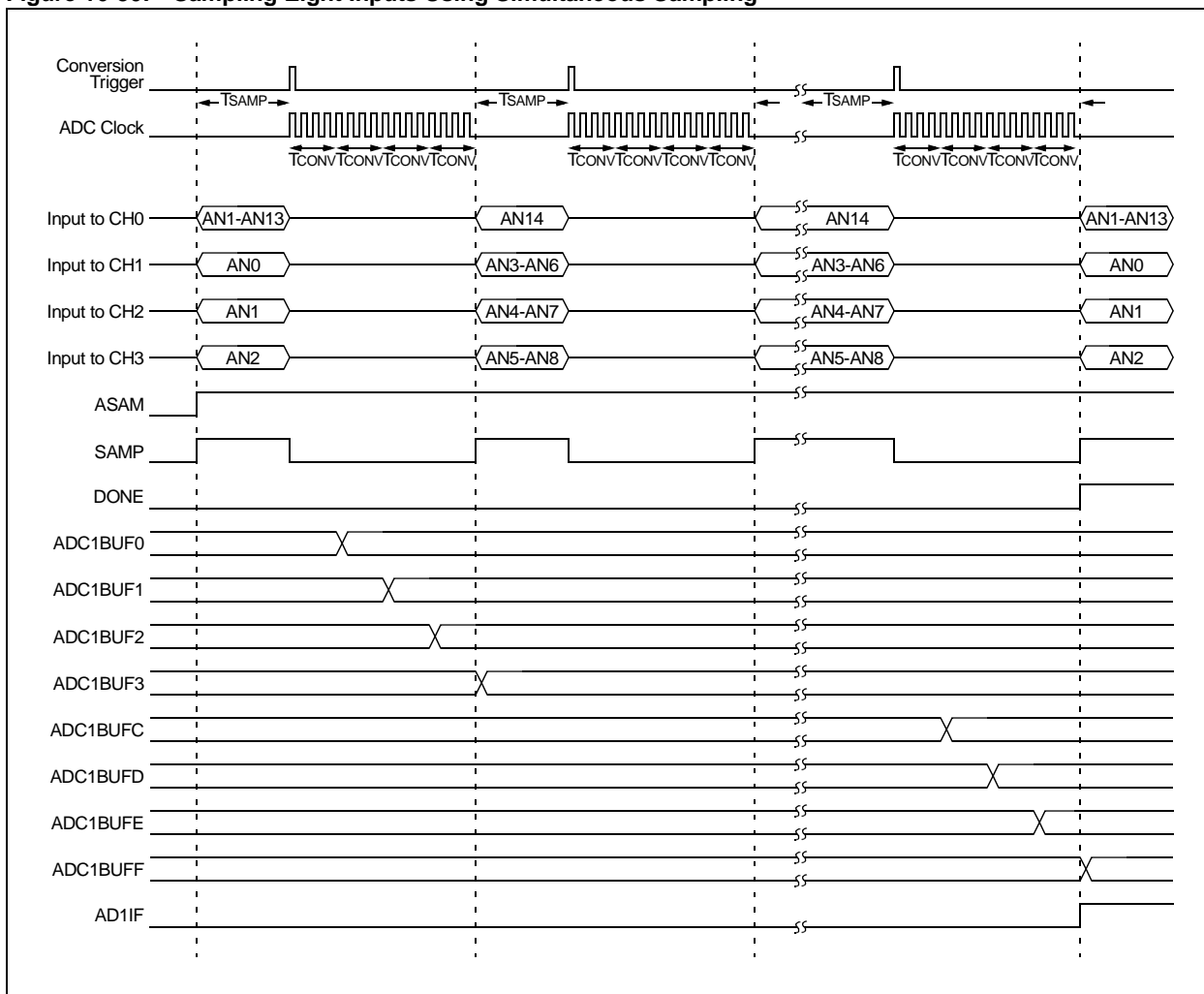
Table 16-17:   Sampling Eight Inputs Using Simultaneous Sampling

CONTROL BITS

Sequence Select

SMPI<4:0> = 00011	
	Interrupt on 4th Sample
CHPS<1:0> = 1x	
	Sample Channels CH0, CH1, CH2, CH3
SIMSAM = 1	
	Sample All Channels Simultaneously
BUFM = 0	
	Single 16-Word Result Buffer
ALTS = 1	
	Alternate MUXA/MUXB Input Select
ADDMAEN = 0	
	Do Not Use DMA with ADC

MUXA Input Select

CH0SA<5:0> = 111101	
	Select AN13 for CH0+ Input
CH0NA = 1	
	Select AN1 for CH0- Input
CSCNA = 0	
	No Input Scan
CSS<15:0> = n/a	
	Scan Input Select Unused
CH123SA<2:0> = 000	
	CH1+ = AN0, CH2+ = AN1, CH3+ = AN2
CH123NA<1:0> = 0x	
	CH1-, CH2-, CH3- = VREF-

MUXB Input Select

CH0SB<5:0> = 111110	
	Select AN14 for CH0+ Input
CH0NB = 0	
	Select VREF- for CH0- Input
CH123SB<2:0> = 111	
	CH1+ = AN3, CH2+ = AN4, CH3+ = AN5
CH123NB<1:0> = 10	
	CH1- = AN6, CH2- = AN7, CH3- = AN8

OPERATION SEQUENCE

Sample MUXA Inputs:
(AN1-AN13) ≥ CH0, AN0 ≥ CH1, AN1 ≥ CH2, AN2 ≥ CH3
Convert CH0, Write ADC1BUF0
Convert CH1, Write ADC1BUF1
Convert CH2, Write ADC1BUF2
Convert CH3, Write ADC1BUF3
Sample MUXB Inputs:
AN14 ≥ CH0,
(AN3-AN6) ≥ CH1, (AN4-AN7) ≥ CH2, (AN5-AN8) ≥ CH3
Convert CH0, Write ADC1BUF4
Convert CH1, Write ADC1BUF5
Convert CH2, Write ADC1BUF6
Convert CH3, Write ADC1BUF7
Sample MUXA Inputs:
(AN1-AN13) ≥ CH0, AN0 ≥ CH1, AN1 ≥ CH2, AN2 ≥ CH3
Convert CH0, Write ADC1BUF8
Convert CH1, Write ADC1BUF9
Convert CH2, Write ADC1BUFA
Convert CH3, Write ADC1BUFB
Sample MUXB Inputs:
AN14 ≥ CH0,
(AN3-AN6) ≥ CH1, (AN4-AN7) ≥ CH2, (AN5-AN8) ≥ CH3
Convert CH0, Write ADC1BUFC
Convert CH1, Write ADC1BUFD
Convert CH2, Write ADC1BUFE
Convert CH3, Write ADC1BUFF
ADC Interrupt
Repeat

	ADC Buffer @ First ADC Interrupt	ADC Buffer @ Second ADC Interrupt
ADC1BUF0	(AN1-AN13) Sample 1	(AN1-AN13) Sample 3
ADC1BUF1	AN0 Sample 1	AN0 Sample 3
ADC1BUF2	AN1 Sample 1	AN1 Sample 3
ADC1BUF3	AN2 Sample 1	AN2 Sample 3
ADC1BUF4	AN14 Sample 1	AN14 Sample 3
ADC1BUF5	(AN3-AN6) Sample 1	(AN3-AN6) Sample 3
ADC1BUF6	(AN4-AN7) Sample 1	(AN4-AN7) Sample 3
ADC1BUF7	(AN5-AN8) Sample 1	(AN5-AN8) Sample 3
ADC1BUF8	(AN1-AN13) Sample 1	(AN1-AN13) Sample 4
ADC1BUF9	AN0 Sample 2	AN0 Sample 4
ADC1BUFA	AN1 Sample 2	AN1 Sample 4
ADC1BUFB	AN2 Sample 2	AN2 Sample 4
ADC1BUFC	AN14 Sample 2	AN14 Sample 4
ADC1BUFD	(AN3-AN6) Sample 2	(AN3-AN6) Sample 4
ADC1BUFE	(AN4-AN7) Sample 2	(AN4-AN7) Sample 4
ADC1BUFF	(AN5-AN8) Sample 2	(AN5-AN8) Sample 4

16.10.6   Sampling Eight Inputs Using Sequential Sampling

Figure 16-31 and Table 16-18 demonstrate sequential sampling. When converting more than one channel and selecting sequential sampling, the ADC module starts sampling a channel at the earliest opportunity, then performs the required conversions in sequence. In this example, with the ASAM bit set, sampling of a channel begins after the conversion of that channel completes.

When the ASAM bit is clear, sampling does not resume after conversion completion, but occurs when the SAMP bit (ADxCON1<1>) is set.

When utilizing more than one channel, sequential sampling provides more sampling time since a channel can be sampled while a conversion occurs on another.

Figure 16-31:   Sampling Eight Inputs Using Sequential Sampling
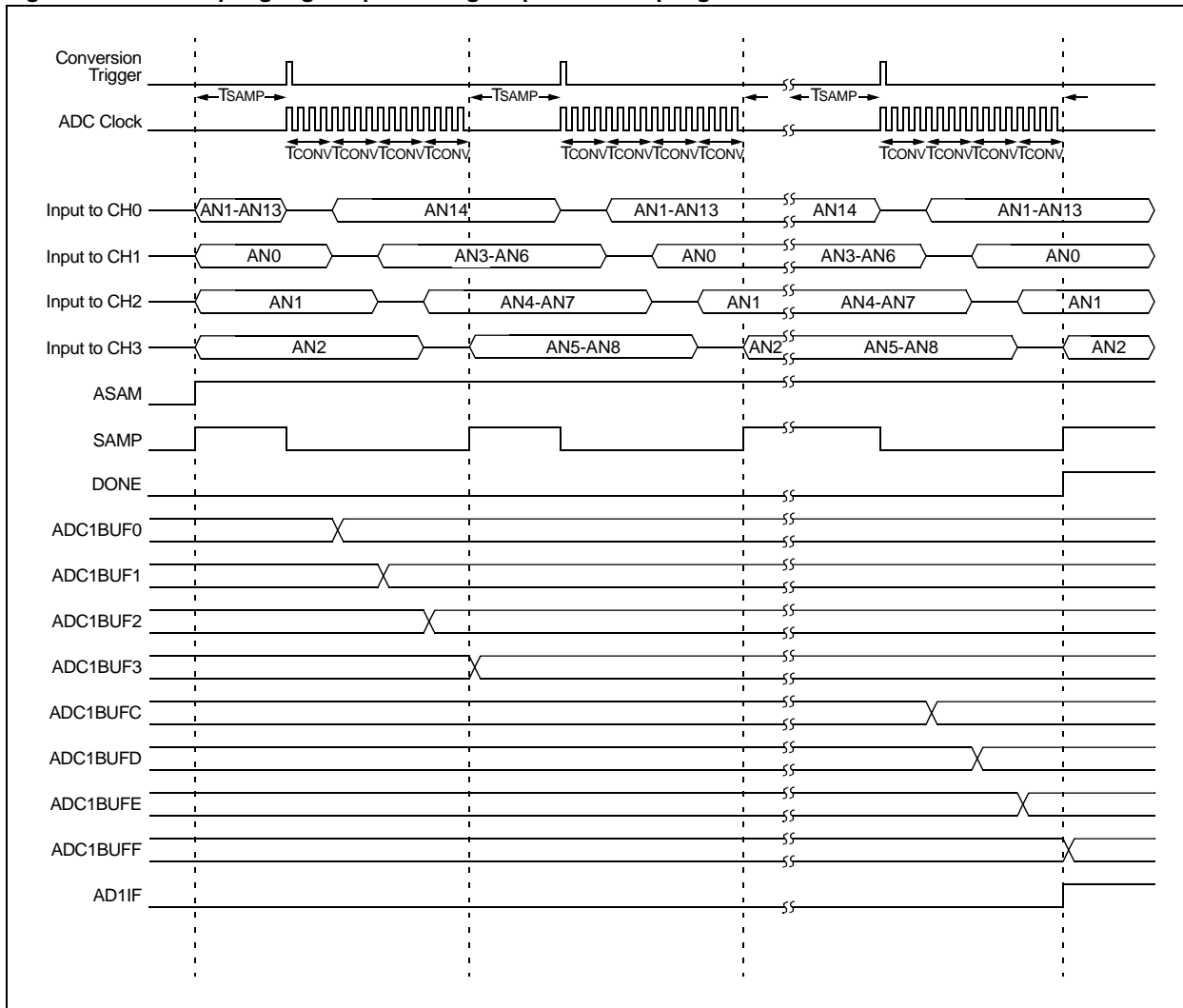
Table 16-18: Sampling Eight Inputs Using Sequential Sampling

CONTROL BITS	OPERATION SEQUENCE

CONTROL BITS Sequence Select	OPERATION SEQUENCE
SMPI<4:0> = `01111` Interrupt on 16th Sample	Sample: (AN1-AN13) ≥ CH0
	Convert CH0, Write ADC1BUF0
CHPS<1:0> = `1x` Sample Channels CH0, CH1, CH2, CH3	Sample: AN0 ≥ CH1
	Convert CH1, Write ADC1BUF1
SIMSAM = `0` Sample All Channels Sequentially	Sample: AN1 ≥ CH2
	Convert CH2, Write ADC1BUF2
BUFM = `0` Single 16-Word Result Buffer	Sample: AN2 ≥ CH3
	Convert CH3, Write ADC1BUF3
ALTS = `1` Alternate MUXA/MUXB Input Select	Sample: AN14 ≥ CH0
	Convert CH0, Write ADC1BUF4
ADDMAEN = `0` Do Not Use DMA with ADC	Sample: (AN3-AN6) ≥ CH1
	Convert CH1, Write ADC1BUF5
MUXA Input Select	Sample: (AN4-AN7) ≥ CH2
	Convert CH2, Write ADC1BUF6
CH0SA<5:0> = `111101` Select AN13 for CH0+ Input	Sample: (AN5-AN8) ≥ CH3
	Convert CH3, Write ADC1BUF7
CH0NA = `1` Select AN1 for CH0- Input	Sample: (AN1-AN13) ≥ CH0
	Convert CH0, Write ADC1BUF8
CSCNA = `0` No Input Scan	Sample: AN0 ≥ CH1
	Convert CH1, Write ADC1BUF9
CSS<15:0> = `n/a` Scan Input Select Unused	Sample: AN1 ≥ CH2
	Convert CH2, Write ADC1BUFA
CH123SA<2:0> = `000` CH1+ = AN0, CH2+ = AN1, CH3+ = AN2	Sample: AN2 ≥ CH3
	Convert CH3, Write ADC1BUFB
CH123NA<1:0> = `0x` CH1-, CH2-, CH3- = V_{REF}-	Sample: AN14 ≥ CH0
	Convert CH0, Write ADC1BUFC
MUXB Input Select	Sample: (AN3-AN6) ≥ CH1
	Convert CH1, Write ADC1BUFD
CH0SB<5:0> = `111110` Select AN14 for CH0+ Input	Sample: (AN4-AN7) ≥ CH2
	Convert CH2, Write ADC1BUFE
CH0NB = `0` Select V_{REF}- for CH0- Input	Sample: (AN5-AN8) ≥ CH3
	Convert CH3, Write ADC1BUFF
CH123SB<2:0> = `111` CH1+ = AN3, CH2+ = AN4, CH3+ = AN5	ADC Interrupt
CH123NB<1:0> = `10` CH1- = AN6, CH2- = AN7, CH3- = AN8	**Repeat**

	ADC Buffer @ First ADC Interrupt	ADC Buffer @ Second ADC Interrupt
ADC1BUF0	(AN1-AN13) Sample 1	(AN1-AN13) Sample 3
ADC1BUF1	AN0 Sample 1	AN0 Sample 3
ADC1BUF2	AN1 Sample 1	AN1 Sample 3
ADC1BUF3	AN2 Sample 1	AN2 Sample 3
ADC1BUF4	AN14 Sample 1	AN14 Sample 3
ADC1BUF5	(AN3-AN6) Sample 1	(AN3-AN6) Sample 3
ADC1BUF6	(AN4-AN7) Sample 1	(AN4-AN7) Sample 3
ADC1BUF7	(AN5-AN8) Sample 1	(AN5-AN8) Sample 3
ADC1BUF8	(AN1-AN13) Sample 2	(AN1-AN13) Sample 4
ADC1BUF9	AN0 Sample 2	AN0 Sample 4
ADC1BUFA	AN1 Sample 2	AN1 Sample 4
ADC1BUFB	AN2 Sample 2	AN2 Sample 28
ADC1BUFC	AN14 Sample 2	AN14 Sample 4
ADC1BUFD	(AN3-AN6) Sample 2	(AN3-AN6) Sample 4
ADC1BUFE	(AN4-AN7) Sample 2	(AN4-AN7) Sample 4
ADC1BUFF	(AN5-AN8) Sample 2	(AN5-AN8) Sample 4

16.11 SAMPLE AND CONVERSION SEQUENCE EXAMPLES FOR DEVICES WITH DMA AND WITH ADDMAEN BIT SET

The following configuration examples show the Analog-to-Digital operation in different sampling and buffering configurations. In each example, setting the ASAM bit starts automatic sampling. A conversion trigger ends sampling and starts conversion.

16.11.1 Sampling and Converting a Single Channel, Multiple Times

Figure 16-32 and Table 16-19 illustrate a basic configuration of the ADC. In this case, one ADC input, AN0, is sampled by one S&H channel, CH0, and converted. The results are stored in the user-configured DMA RAM buffer. This process repeats 16 times until the buffer is full and then the DMA module generates an interrupt. The entire process then repeats.

The CHPS<1:0> bits specify that only S&H CH0 is active. With ALTS clear, only the MUXA inputs are active. The CH0SA<5:0> bits and CH0NA bit are specified (AN0-V$_{REF}$-) as the input to the S&H channel. All other input selection bits are not used.

Figure 16-32: Converting One Channel, 16 Times/DMA Interrupt
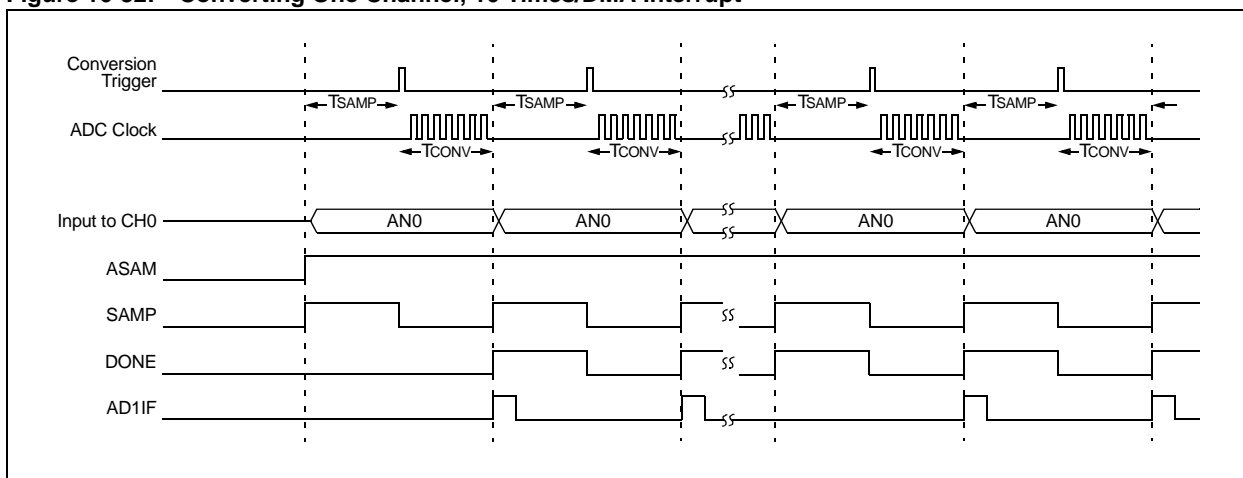
Table 16-19: Converting One Channel, 16 Times per DMA Interrupt

CONTROL BITS	OPERATION SEQUENCE

Sequence Select

SMPI<4:0> = 00000 　　　DMA Address Increments After Every 　　　Sample/Conversion Operation
CHPS<1:0> = 00 　　　Sample Channel CH0
SIMSAM = n/a 　　Not Applicable for Single Channel Sample
ADDMABM = 1 　　DMA Buffer Written in Order of Conversion
DMABL<2:0> = 100 　　16-Word Buffer Allocated to Analog Input
ALTS = 0 　　　Always Use MUXA Input Select
ADDMAEN = 1 　　　Use DMA with ADC

MUXA Input Select

CH0SA<5:0> = 000000 　　　Select AN0 for CH0+ Input
CH0NA = 0 　　　Select VREF- for CH0- Input
CSCNA = 0 　　　No Input Scan
CSS<15:0> = n/a 　　　Scan Input Select Unused
CH123SA<2:0> = n/a 　　Channels CH1, CH2, CH3+ Input Unused
CH123NA<1:0> = n/a 　　Channels CH1, CH2, CH3- Input Unused

MUXB Input Select

CH0SB<5:0> = n/a 　　　Channel CH0+ Input Unused
CH0NB = n/a 　　　Channel CH0- Input Unused
CH123SB<2:0> = n/a 　　Channels CH1, CH2, CH3+ Input Unused
CH123NB<1:0> = n/a 　　Channels CH1, CH2, CH3- Input Unused

OPERATION SEQUENCE

Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
Sample MUXA Inputs: AN0 ≥ CH0
Convert CH0
DMA Interrupt
Repeat

DMA Buffer @ First DMA Interrupt

AN0 Sample 1
AN0 Sample 2
AN0 Sample 3
AN0 Sample 4
AN0 Sample 5
AN0 Sample 6
AN0 Sample 7
AN0 Sample 8
AN0 Sample 9
AN0 Sample 10
AN0 Sample 11
AN0 Sample 12
AN0 Sample 13
AN0 Sample 14
AN0 Sample 15
AN0 Sample 16

DMA Buffer @ Second DMA Interrupt

AN0 Sample 17
AN0 Sample 18
AN0 Sample 19
AN0 Sample 20
AN0 Sample 21
AN0 Sample 22
AN0 Sample 23
AN0 Sample 24
AN0 Sample 25
AN0 Sample 26
AN0 Sample 27
AN0 Sample 28
AN0 Sample 29
AN0 Sample 30
AN0 Sample 31
AN0 Sample 32

Note: The DMA module must be configured correctly to compliment the ADC module.

16.11.2 Analog-to-Digital Conversions While Scanning Through 16 Analog Inputs

Figure 16-33 and Table 16-20 illustrate a typical setup, where all available analog input channels are sampled by one S&H channel, CH0, and converted. The Set Scan Input Selection bit (CSCNA) in ADC Control Register 2 (ADxCON2<10>) specifies the scanning of the ADC inputs to the CH0 positive input. Other conditions are similar to those described in **Section 16.10.1 "Sampling and Converting a Single Channel Multiple Times"**.

Initially, the AN0 input is sampled by CH0 and converted. The result is stored in the user-configured DMA buffer. Then, the AN1 input is sampled and converted. This process of scanning the inputs repeats 16 times until the buffer is full. Then the DMA module generates an interrupt. The entire process then repeats.

Figure 16-33: Scanning Through 16 Inputs/DMA Interrupt
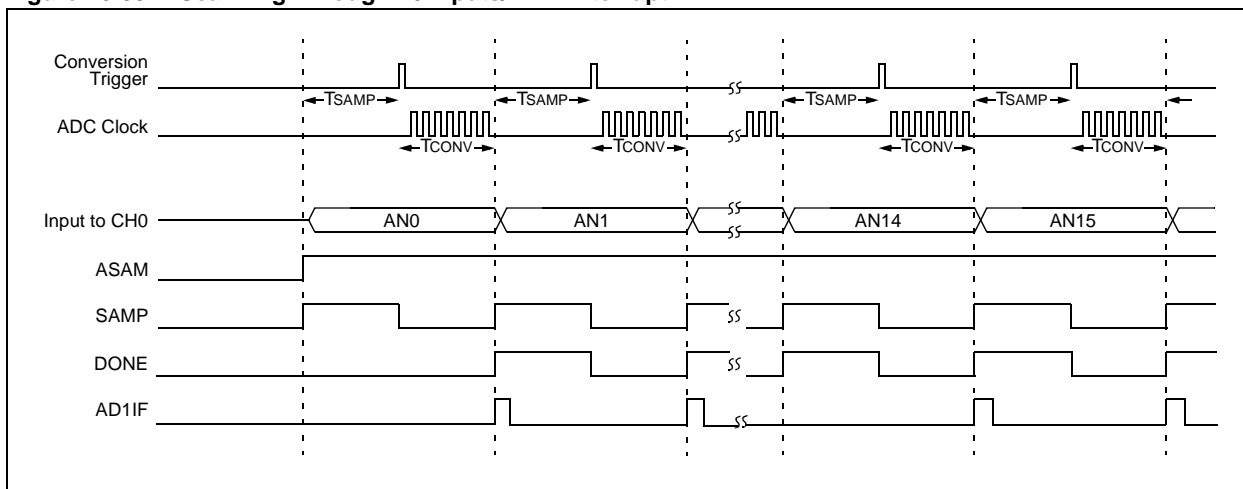
Table 16-20: Scanning Through 16 Inputs per DMA Interrupt

CONTROL BITS	OPERATION SEQUENCE		
Sequence Select			
SMPI<4:0> (# of channels to scan – 1) = 01111 DMA Address Increments After Every 16th Sample/Conversion Operation	Sample MUXA Inputs: AN0 ≥ CH0	AN0 Sample 1	AN0 Block
	Convert CH0	AN0 Sample 17	
	Sample MUXA Inputs: AN1 ≥ CH0	Sample 33	
	Convert CH0	Sample 49	
CHPS<1:0> = 00 Sample Channel CH0	Sample MUXA Inputs: AN2 ≥ CH0	Sample 2	AN1 Block
	Convert CH0	Sample 18	
SIMSAM = n/a Not Applicable for Single Channel Sample	Sample MUXA Inputs: AN3 ≥ CH0	Sample 34	
	Convert CH0	Sample 50	
ADDMABM = 0 DMA Buffer Written in Scatter Gather Fashion	Sample MUXA Inputs: AN4 ≥ CH0	Sample 3	AN1 Block
	Convert CH0	Sample 19	
DMABL<2:0> = 010 Each Analog Input Buffer Contains 4 Words	Sample MUXA Inputs: AN5 ≥ CH0	Sample 35	
	Convert CH0	Sample 51	
ALTS = 0 Always Use MUXA Input Select	Sample MUXA Inputs: AN6 ≥ CH0	Sample 4	AN3 Block
	Convert CH0	Sample 20	
ADDMAEN = 1 Use DMA with ADC	Sample MUXA Inputs: AN7 ≥ CH0	Sample 36	
	Convert CH0	Sample 52	
MUXA Input Select	Sample MUXA Inputs: AN8 ≥ CH0	Sample 5	AN4 Block
CH0SA<5:0> = n/a Override by CSCNA	Convert CH0	Sample 21	
	Sample MUXA Inputs: AN9 ≥ CH0	Sample 37	
CH0NA = 0 Select VREF- for CH0- Input	Convert CH0	Sample 53	
CSCNA = 1 Scan CH0+ Inputs	Sample MUXA Inputs: AN10 ≥ CH0	Sample 6	
	Convert CH0		
CSS<15:0> = 1111 1111 1111 1111 Scan Input Select Unused	Sample MUXA Inputs: AN11 ≥ CH0		
	Convert CH0		
CH123SA<2:0> = n/a Channels CH1, CH2, CH3+ Input Unused	Sample MUXA Inputs: AN12 ≥ CH0		
	Convert CH0	Sample 30	AN13 Block
CH123NA<1:0> = n/a Channels CH1, CH2, CH3- Input Unused	Sample MUXA Inputs: AN13 ≥ CH0	Sample 46	
	Convert CH0	Sample 62	
MUXB Input Select	Sample MUXA Inputs: AN14 ≥ CH0	Sample 15	AN14 Block
CH0SB<5:0> = n/a Channel CH0+ Input Unused	Convert CH0	Sample 31	
	Sample MUXA Inputs: AN15 ≥ CH0	Sample 47	
CH0NB = n/a Channel CH0- Input Unused	Convert CH0	Sample 63	
	DMA Interrupt	Sample 16	AN15 Block
CH123SB<2:0> = n/a Channels CH1, CH2, CH3+ Input Unused	**Repeat**	Sample 32	
		Sample 48	
CH123NB<1:0> = n/a Channels CH1, CH2, CH3- Input Unused		Sample 64	
		•	
		•	
		•	
			AN31 Block

16.11.3 Using Alternating MUXA, MUXB Input Selections

Figure 16-34 and Table 16-21 demonstrate alternate sampling of the inputs assigned to MUXA and MUXB. In this example, two channels are enabled to sample simultaneously. Setting the ALTS bit (ADCxCON2<0>) enables alternating input selections. The first sample uses the MUXA inputs, specified by the CH0SA<5:0>, CH0NA, CH123SA<2:0> and CH123NA<1:0> bits. The next sample uses the MUXB inputs, specified by the CH0SB<5:0>, CH0NB, CH123SB<2:0> and CH123NB<1:0> bits. In this example, one of the MUXB input specifications uses two analog inputs as a differential source to the S&H, sampling (AN3-AN9).

Using four S&H channels without alternating input selections results in the same number of conversions as this example, using two channels with alternating input selections. However, because the CH1, CH2 and CH3 channels are more limited in the selectivity of the analog inputs, this example method provides more flexibility of input selection than using four channels.

Figure 16-34: Converting Two Sets of Two Inputs Using Alternating Input Selections
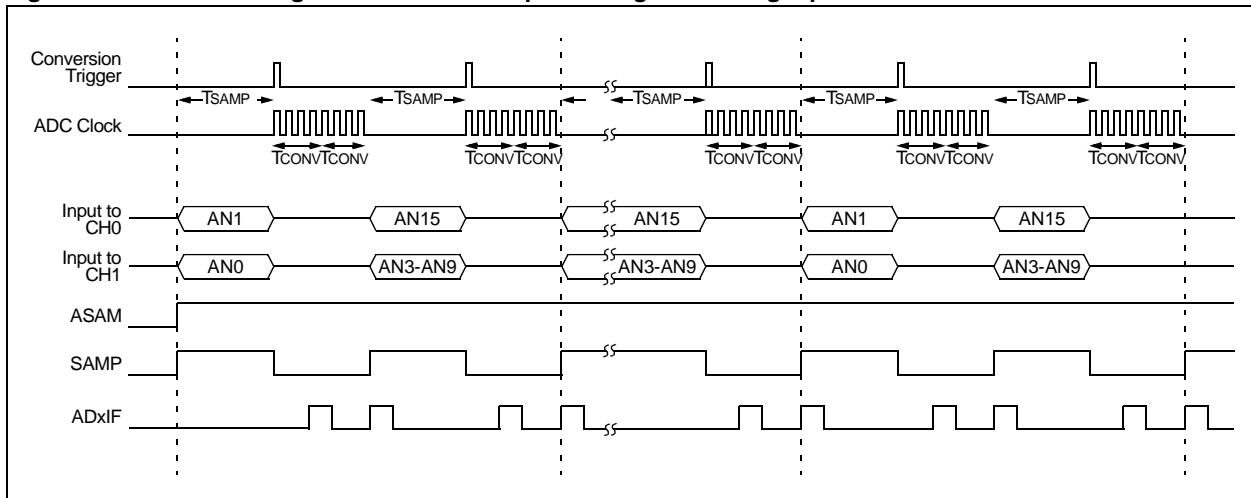
Table 16-21:   Converting Two Sets of Two Inputs Using Alternating Input Selections

CONTROL BITS	OPERATION SEQUENCE

CONTROL BITS

Sequence Select

SMPI<4:0> = `00001`
DMA Address Increments After Every 2nd Sample/Conversion Operation
CHPS<1:0> = `01`
Sample Channels CH0, CH1
SIMSAM = `1`
Sample All Channels Simultaneously
ADDMABM = `1`
DMA Buffer Written in Order of Conversion
ALTS = `1`
Alternate MUXA/MUXB Input Select
ADDMAEN = `1`
Use DMA with ADC

MUXA Input Select

CH0SA<5:0> = `000001`
Select AN1 for CH0+ Input
CH0NA = `0`
Select V_{REF}- for CH0- Input
CSCNA = `0`
No Input Scan
CSS<15:0> = n/a
Scan Input Select Unused
CH123SA<2:0> = `000`
CH1+ = AN0, CH2+ = AN1, CH3+ = AN2
CH123NA<1:0> = `0x`
CH1-, CH2-, CH3- = V_{REF}-

MUXB Input Select

CH0SB<5:0> = `111111`
Select AN15 for CH0+ Input
CH0NB = `0`
Select V_{REF}- for CH0- Input
CH123SB<2:0> = `111`
CH1+ = AN3, CH2+ = AN4, CH3+ = AN5
CH123NB<1:0> = `11`
CH1- = AN9, CH2- = AN10, CH3- = AN11

OPERATION SEQUENCE

Sample MUXA Inputs: AN1 $\geq$ CH0, AN0 $\geq$ CH1
Convert CH0
Convert CH1
Sample MUXB Inputs: AN15 $\geq$ CH0, (AN3-AN9) $\geq$ CH1
Convert CH0
Convert CH1
Sample MUXA Inputs: AN1 $\geq$ CH0, AN0 $\geq$ CH1
Convert CH0
Convert CH1
Sample MUXB Inputs: AN15 $\geq$ CH0, (AN3-AN9) $\geq$ CH1
Convert CH0
Convert CH1
DMA Interrupt
Sample MUXA Inputs: AN1 $\geq$ CH0, AN0 $\geq$ CH1
Convert CH0
Convert CH1
Sample MUXB Inputs: AN15 $\geq$ CH0, (AN3-AN9) $\geq$ CH1
Convert CH0
Convert CH1
Sample MUXA Inputs: AN1 $\geq$ CH0, AN0 $\geq$ CH1
Convert CH0
Convert CH1
Sample MUXB Inputs: AN15 $\geq$ CH0, (AN3-AN9) $\geq$ CH1
Convert CH0
Convert CH1
DMA Interrupt
Repeat

DMA Buffer @ First DMA Interrupt

AN1 Sample 1
AN0 Sample 1
AN15 Sample 1
(AN3-AN9) Sample 1

DMA Buffer @ Second DMA Interrupt

AN1 Sample 3
AN0 Sample 3
AN15 Sample 3
(AN3-AN9) Sample 3

16.11.4 Sampling Eight Inputs Using Simultaneous Sampling

Figure 16-35 and Table 16-22 demonstrate identical setups, with the exception that this example uses simultaneous sampling (SIMSAM = 1), and Figure 16-36 uses sequential sampling (SIMSAM = 0). Both examples use alternating inputs and specify differential inputs to the S&H.

Figure 16-35 and Table 16-22 demonstrate simultaneous sampling. When converting more than one channel, and selecting simultaneous sampling, the ADC module samples all channels, then performs the required conversions in sequence. In this example, with the ASAM bit set, sampling begins after the conversion is complete.

Figure 16-35:   Sampling Eight Inputs Using Simultaneous Sampling
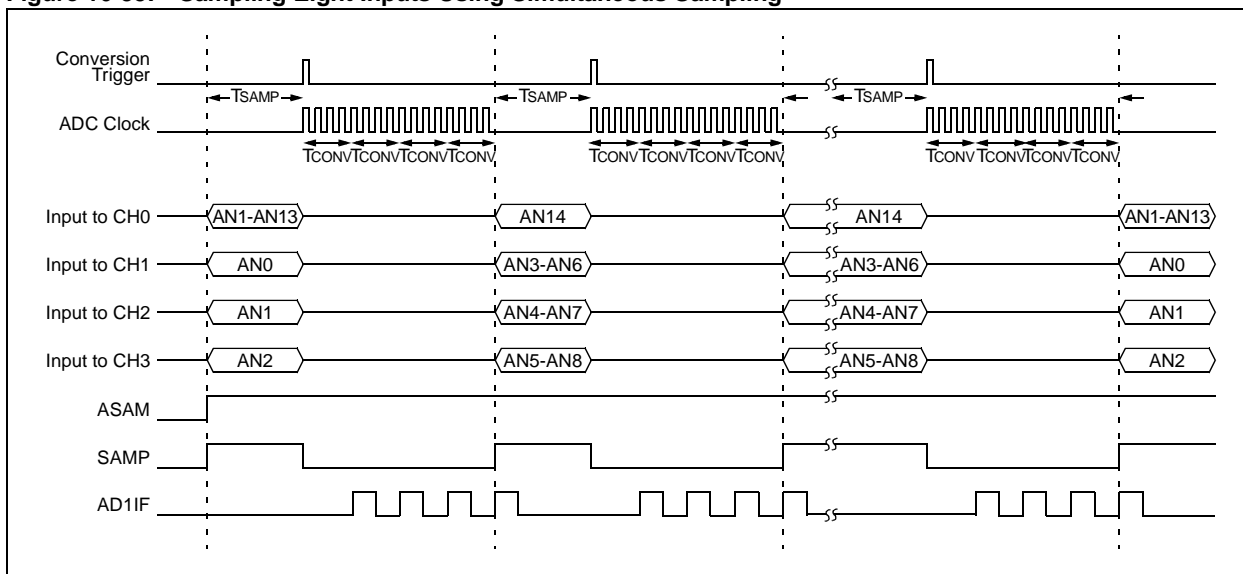
Table 16-22: Sampling Eight Inputs Using Simultaneous Sampling

CONTROL BITS	OPERATION SEQUENCE
Sequence Select	

<table>
<tr><td colspan="2">CONTROL BITS</td><td colspan="2">OPERATION SEQUENCE</td></tr>
</table>

CONTROL BITS

Sequence Select

SMPI<4:0> = 00001
DMA Address Increments After Every 2nd Sample/Conversion Operation
CHPS<1:0> = 1x
Sample Channels CH0, CH1, CH2, CH3
SIMSAM = 1
Sample All Channels Simultaneously
ADDMABM = 0
DMA Buffer Written in Order of Conversion
ALTS = 1
Alternate MUXA/MUXB Input Select
ADDMAEN = 1
Use DMA with ADC

MUXA Input Select

CH0SA<5:0> = 111101
Select AN13 for CH0+ Input
CH0NA = 1
Select AN1 for CH0- Input
CSCNA = 0
No Input Scan
CSS<15:0> = n/a
Scan Input Select Unused
CH123SA<2:0> = 000
CH1+ = AN0, CH2+ = AN1, CH3+ = AN2
CH123NA<1:0> = 0x
CH1-, CH2-, CH3- = VREF-

MUXB Input Select

CH0SB<5:0> = 111110
Select AN14 for CH0+ Input
CH0NB = 0
Select VREF- for CH0- Input
CH123SB<2:0> = 111
CH1+ = AN3, CH2+ = AN4, CH3+ = AN5
CH123NB<1:0> = 10
CH1- = AN6, CH2- = AN7, CH3- = AN8

OPERATION SEQUENCE

Sample MUXA Inputs: (AN1-AN13) $\geq$ CH0, AN0 $\geq$ CH1, AN1 $\geq$ CH2, AN2 $\geq$ CH3
Convert CH0
Convert CH1
Convert CH2
Convert CH3
Sample MUXB Inputs: AN14 $\geq$ CH0, (AN3-AN6) $\geq$ CH1, (AN4-AN7) $\geq$ CH2, (AN5-AN8) $\geq$ CH3
Convert CH0
Convert CH1
Convert CH2
Convert CH3
Sample MUXA Inputs: (AN1-AN13) $\geq$ CH0, AN0 $\geq$ CH1, AN1 $\geq$ CH2, AN2 $\geq$ CH3
Convert CH0
Convert CH1
Convert CH2
Convert CH3
Sample MUXB Inputs: AN14 $\geq$ CH0, (AN3-AN6) $\geq$ CH1, (AN4-AN7) $\geq$ CH2, (AN5-AN8) $\geq$ CH3
Convert CH0
Convert CH1
Convert CH2
Convert CH3
DMA Interrupt
Repeat

DMA Buffer @ First DMA Interrupt

(AN1-AN13) Sample 1
AN0 Sample 1
AN1 Sample 1
AN2 Sample 1
AN14 Sample 1
(AN3-AN6) Sample 1
(AN4-AN7) Sample 1
(AN5-AN8) Sample 1
(AN1-AN13) Sample 1
AN0 Sample 2
AN1 Sample 2
AN2 Sample 2
AN14 Sample 2
(AN3-AN6) Sample 2
(AN4-AN7) Sample 2
(AN5-AN8) Sample 2

DMA Buffer @ Second DMA Interrupt

(AN1-AN13) Sample 3
AN0 Sample 3
AN1 Sample 3
AN2 Sample 3
AN14 Sample 3
(AN3-AN6) Sample 3
(AN4-AN7) Sample 3
(AN5-AN8) Sample 3
(AN1-AN13) Sample 4
AN0 Sample 4
AN1 Sample 4
AN2 Sample 4
AN14 Sample 4
(AN3-AN6) Sample 4
(AN4-AN7) Sample 4
(AN5-AN8) Sample 4

16.11.5 Sampling Eight Inputs Using Sequential Sampling

Figure 16-36 and Table 16-23 demonstrate sequential sampling. When converting more than one channel and selecting sequential sampling, the ADC module starts sampling a channel at the earliest opportunity, then performs the required conversions in sequence. In this example, with the ASAM bit set, sampling of a channel begins after the conversion of that channel completes.

When ASAM is clear, sampling does not resume after conversion completion, but occurs when the SAMP bit (ADxCON1<1>) is set.

When utilizing more than one channel, sequential sampling provides more sampling time since a channel can be sampled while a conversion occurs on another channel.

Figure 16-36:   Sampling Eight Inputs Using Sequential Sampling

Table 16-23: Sampling Eight Inputs Using Sequential Sampling

CONTROL BITS	OPERATION SEQUENCE

Sequence Select

SMPI<4:0> = 00001	
DMA Address Increments After Every 2nd Sample/Conversion Operation	
CHPS<1:0> = 1x	
Sample Channels CH0, CH1, CH2, CH3	
SIMSAM = 0	
Sample All Channels Sequentially	
ADDMABM = 1	
DMA Buffer Written in Order of Conversion	
ALTS = 1	
Alternate MUXA/MUXB Input Select	
ADDMAEN = 1	
Use DMA with ADC	

MUXA Input Select

CH0SA<5:0> = 011110
Select AN6 for CH0+ Input
CH0NA = 0
Select VREF- for CH0- Input
CSCNA = 0
No Input Scan
CSS<15:0> = n/a
Scan Input Select Unused
CH123SA<2:0> = 000
CH1+ = AN0, CH2+ = AN1, CH3+ = AN2
CH123NA<1:0> = 0x
CH1-, CH2-, CH3- = VREF-

MUXB Input Select

CH0SB<5:0> = 011111
Select AN7 for CH0+ Input
CH0NB = 0
Select VREF- for CH0- Input
CH123SB<2:0> = 111
CH1+ = AN3, CH2+ = AN4, CH3+ = AN5
CH123NB<1:0> = 0x
CH1-, CH2-, CH3- = VREF-

OPERATION SEQUENCE

Sample: (AN1-AN13) ≥ CH0
Convert CH0
Sample: AN0 ≥ CH1
Convert CH1
Sample: AN1 ≥ CH2
Convert CH2
Sample: AN2 ≥ CH3
Convert CH3
Sample: AN14 ≥ CH0
Convert CH0
Sample: (AN3-AN6) ≥ CH1
Convert CH1
Sample: (AN4-AN7) ≥ CH2
Convert CH2
Sample: (AN5-AN8) ≥ CH3
Convert CH3
Sample: (AN1-AN13) ≥ CH0
Convert CH0
Sample: AN0 ≥ CH1
Convert CH1
Sample: AN1 ≥ CH2
Convert CH2
Sample: AN2 ≥ CH3
Convert CH3
Sample: AN14 ≥ CH0
Convert CH0
Sample: (AN3-AN6) ≥ CH1
Convert CH1
Sample: (AN4-AN7) ≥ CH2
Convert CH2
Sample: (AN5-AN8) ≥ CH3
Convert CH3
DMA Interrupt
Repeat

DMA Buffer @ First DMA Interrupt

(AN1-AN13) Sample 1
AN0 Sample 1
AN1 Sample 1
AN2 Sample 1
AN14 Sample 1
(AN3-AN6) Sample 1
(AN4-AN7) Sample 1
(AN5-AN8) Sample 1
(AN1-AN13) Sample 2
AN0 Sample 2
AN1 Sample 2
AN2 Sample 2
AN14 Sample 2
(AN3-AN6) Sample 2
(AN4-AN7) Sample 2
(AN5-AN8) Sample 2

DMA Buffer @ Second DMA Interrupt

(AN1-AN13) Sample 3
AN0 Sample 3
AN1 Sample 3
AN2 Sample 3
AN14 Sample 3
(AN3-AN6) Sample 3
(AN4-AN7) Sample 3
(AN5-AN8) Sample 3
(AN1-AN13) Sample 4
AN0 Sample 4
AN1 Sample 4
AN2 Sample 28
AN14 Sample 4
(AN3-AN6) Sample 4
(AN4-AN7) Sample 4
(AN5-AN8) Sample 4

16.12 CONFIGURATION EXAMPLES FOR DEVICES WITH INTERNAL OP AMPS

The block diagram in Figure 16-2 depicts all of the available connection options to the four Sample-and-Hold (S&H) amplifiers (CH0-CH3) for devices that include internal op amps. Depending on the particular device pinout, the ADC module can have up to 16 analog input pins, designated AN0 through AN15, which are shared with op amps pins.

The following three examples (Figure 16-37, Figure 16-38 and Figure 16-39) illustrate the effects of internal op amp use on the ability of the device to simultaneously sample. These three examples show the use of no op amps, all three op amps and two op amps.

Figure 16-37 shows the use of no op amps, with all of the ANx inputs available to sample external voltages in MUXA/MUXB and alternate simultaneous sampling schemes.

Figure 16-37: Simultaneous/Alternate Sampling Using ANx Pins with All Op Amps Disabled

Note: Refer to Figure 16-2 for the full list of pins.
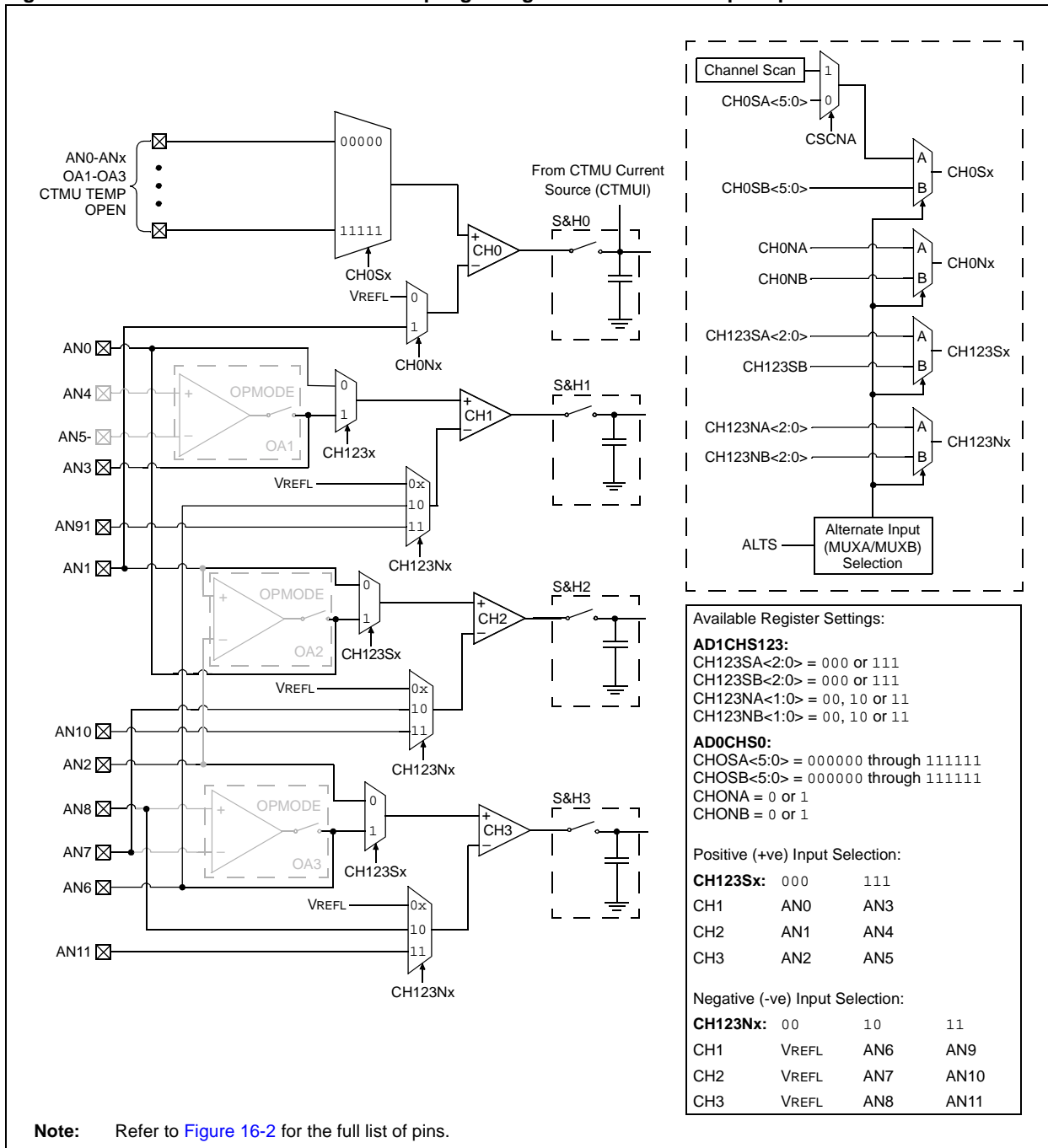
It is important to note that in Figure 16-38, where all three op amps (Op Amp 1, Op Amp 2 and Op Amp 3) are used, all of the analog channels for MUXA sampling are used for op amp functionality. This limits the usefulness of the alternate MUXA/MUXB sampling scheme.

Figure 16-38:   Simultaneous/Alternate Sampling Using ANx Pins and Two Op Amps
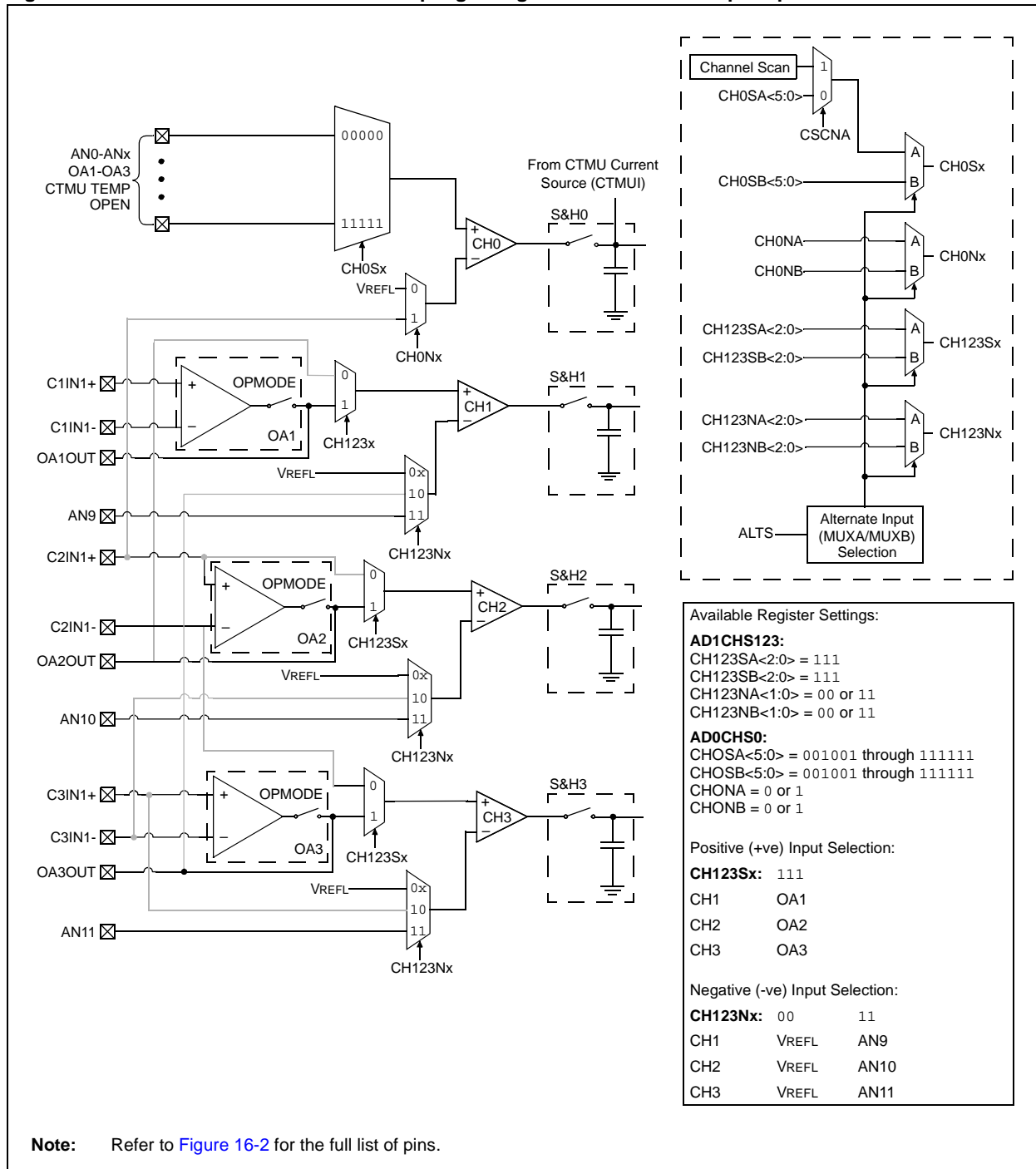
In Figure 16-39, Op Amp 1 and Op Amp 3 are used, while Op Amp 2 remains disabled. This frees AN0, AN1 and AN2 to be used for sampling external voltages. This configuration enables alternate MUXA/MUX B sampling of up to four channels on MUXA, and two op amps and two ANx channels on MUXB. Note that AN0 is duplicated on each sampling.

Figure 16-39:   Simultaneous Sampling Using All Three Op Amps

Note: Refer to Figure 16-2 for the full list of pins.
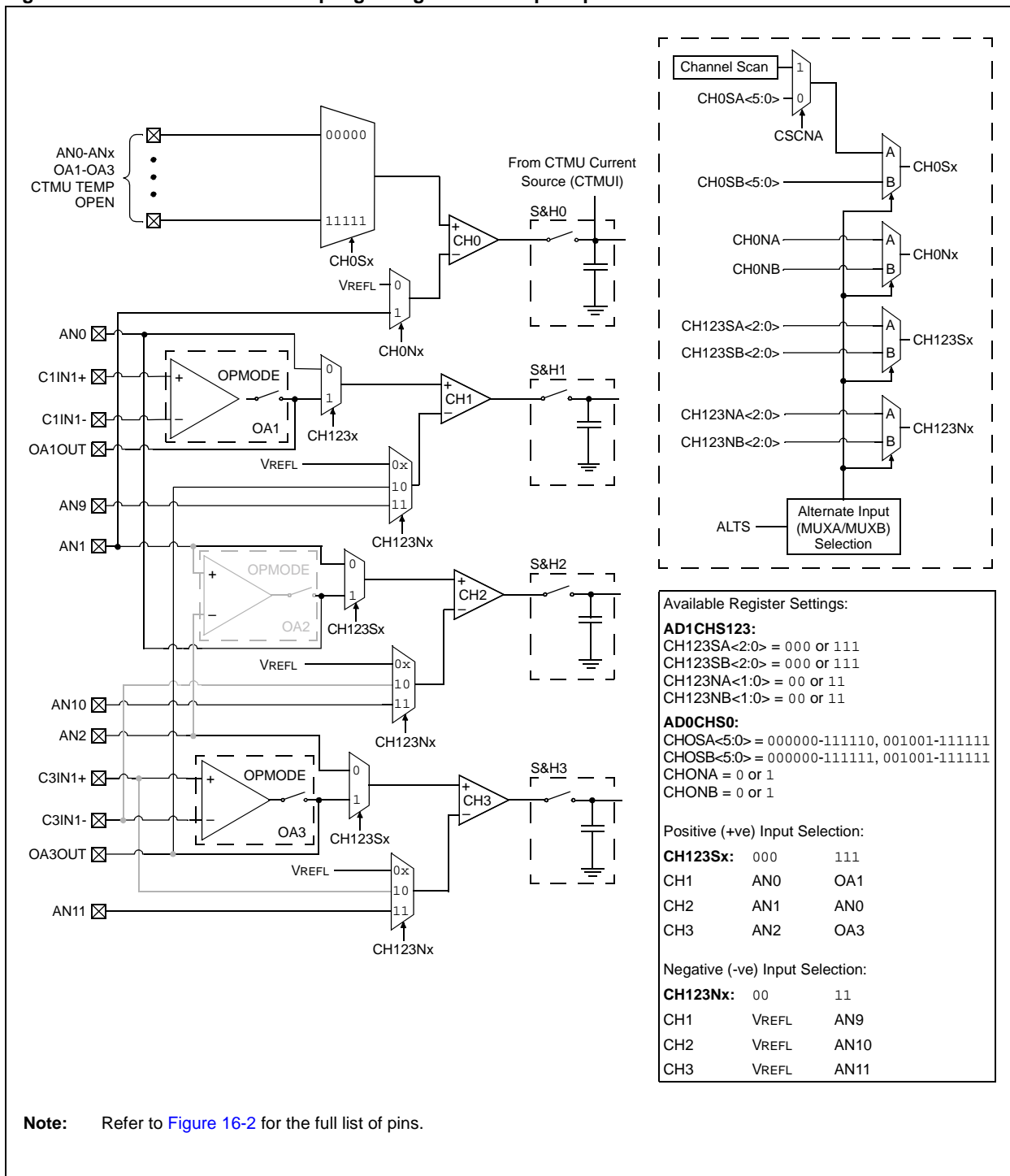
16.13 ANALOG-TO-DIGITAL SAMPLING REQUIREMENTS

The analog input models of the 10-bit and 12-bit ADC modes are shown in Figure 16-40 and Figure 16-41. The total sampling time for the Analog-to-Digital conversion is a function of the internal amplifier settling time and the holding capacitor charge time.

For the ADC module to meet its specified accuracy, the Charge Holding Capacitor (C_{HOLD}) must be allowed to fully charge to the voltage level on the analog input pin. The Analog Output Source (R_S) impedance, the Interconnect Resistance (R_{IC}) impedance and the internal Sampling Switch Resistance (R_{SS}) impedance combine to directly affect the time required to charge the capacitor, C_{HOLD}. Therefore, the combined impedance must be small enough to fully charge the holding capacitor within the chosen sample time. To minimize the effects of pin leakage currents on the accuracy of the ADC module, the maximum recommended source impedance, R_S, is 200Ω. After the analog input channel is selected, this sampling function must be completed prior to starting the conversion. The internal holding capacitor is in a discharged state prior to each sample operation.

A minimum time period must be enabled between conversions for the sample time. For more information on the minimum sampling time for a device, refer to the **"Electrical Characteristics"** chapter in the specific device data sheet.

Figure 16-40: Analog Input Model (10-Bit Mode)
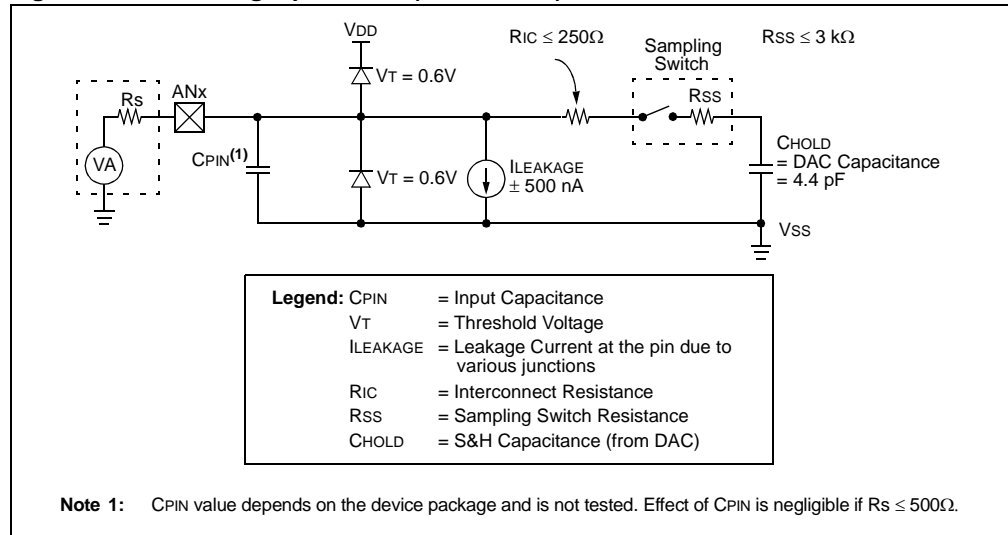
Figure 16-41: Analog Input Model (12-Bit Mode)
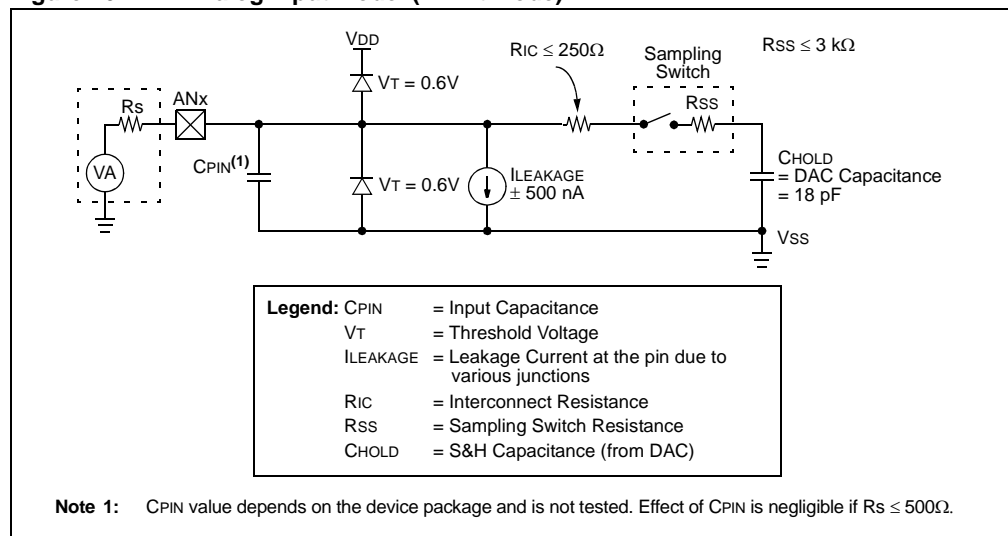
16.14 READING THE ADC RESULT BUFFER

The RAM is 10 bits or 12 bits wide, but the data is automatically formatted to one of four selectable formats when the buffer is read. The FORM<1:0> bits (ADxCON1<9:8>) select the format. The formatting hardware provides a 16-bit result on the data bus for all of the data formats. Figure 16-42 and Figure 16-43 illustrate the data output formats that can be selected using the FORM<1:0> control bits.

Figure 16-42: Analog-to-Digital Output Data Formats (10-Bit Mode)

RAM Contents:							d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

Read to Bus: Unsigned Integer

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

Signed Integer

| $\overline{d09}$ | $\overline{d09}$ | $\overline{d09}$ | $\overline{d09}$ | $\overline{d09}$ | $\overline{d09}$ | $\overline{d09}$ | d08 | d07 | d06 | d05 | d04 | d03 | d02 | d01 | d00 |

Unsigned Fractional (1.15)

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

Signed Fractional (1.15)

| $\overline{d09}$ | d08 | d07 | d06 | d05 | d04 | d03 | d02 | d01 | d00 | 0 | 0 | 0 | 0 | 0 | 0 |
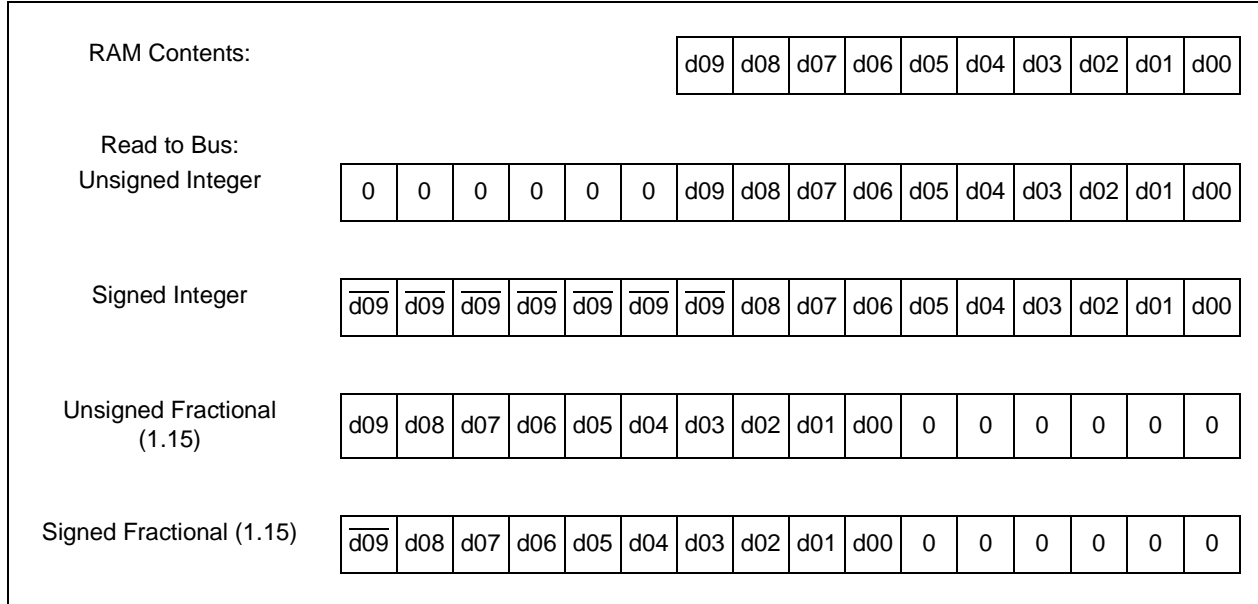
Figure 16-43: Analog-to-Digital Output Data Formats (12-Bit Mode)

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

RAM Contents:

Read to Bus: Unsigned Integer

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

Signed Integer

| $\overline{d11}$ | $\overline{d11}$ | $\overline{d11}$ | $\overline{d11}$ | $\overline{d11}$ | d10 | d09 | d08 | d07 | d06 | d05 | d04 | d03 | d02 | d01 | d00 |

Unsigned Fractional

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

Signed Fractional (1.15)

| $\overline{d11}$ | d10 | d09 | d08 | d07 | d04 | d03 | d02 | d01 | d00 | d01 | d00 | 0 | 0 | 0 | 0 |
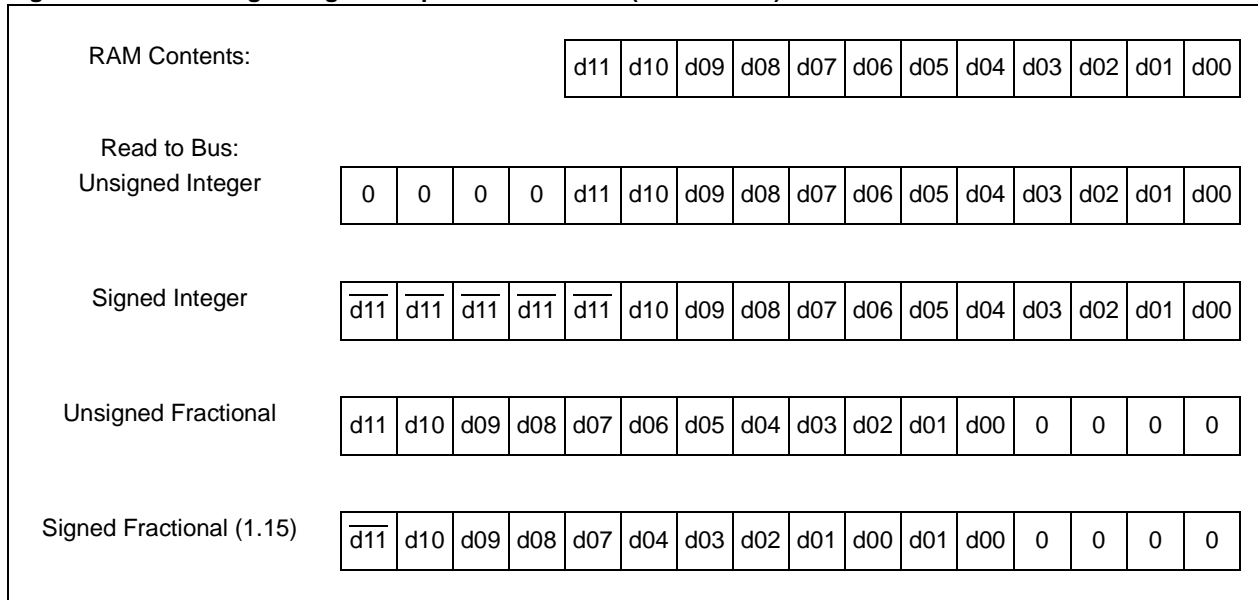
Table 16-24 and Table 16-25 list the numerical equivalents of various result codes for 10-bit and 12-bit modes, respectively.

Table 16-24: **Numerical Equivalents of Various Result Codes (10-Bit Mode)**

V_{IN}/V_{REF}	10-Bit Output Code	16-Bit Integer Format	16-Bit Signed Integer Format	16-Bit Fractional Format	16-Bit Signed Fractional Format
1023/1024	11 1111 1111	0000 0011 1111 1111 = 1023	0000 0001 1111 1111 = 511	1111 1111 1100 0000 = 0.999	0111 1111 1100 0000 = 0.99804
1022/1024	11 1111 1110	0000 0011 1111 1110 = 1022	0000 0001 1111 1110 = 510	1111 1111 1000 0000 = 0.998	0111 1111 1000 0000 = 0.499609
			•		
			•		
			•		
513/1024	10 0000 0001	0000 0010 0000 0001 = 513	0000 0000 0000 0001 = 1	1000 0000 0100 0000 = 0.501	0000 0000 0100 0000 = 0.00195
512/1024	10 0000 0000	0000 0010 0000 0000 = 512	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = 0.500	0000 0000 0000 0000 = 0
511/1024	01 1111 1111	0000 0001 1111 1111 = 511	1111 1111 1111 1111 = –1	0111 1111 1100 0000 = .499	1111 1111 1100 0000 = –0.00195
			•		
			•		
			•		
1/1024	00 0000 0001	0000 0000 0000 0001 = 1	1111 1110 0000 0001 = –511	0000 0000 0100 0000 = 0.001	1000 0000 0100 0000 = –0.99804
0/1024	00 0000 0000	0000 0000 0000 0000 = 0	1111 1110 0000 0000 = –512	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = –1

Table 16-25: **Numerical Equivalents of Various Result Codes (12-Bit Mode)**

V_{IN}/V_{REF}	12-Bit Output Code	16-Bit Unsigned Integer Format	16-Bit Signed Integer Format	16-Bit Unsigned Fractional Format	16-Bit Signed Fractional Format
4095/4096	1111 1111 1111	0000 1111 1111 1111 = 4095	0000 0111 1111 1111 = 2047	1111 1111 1111 0000 = 0.9998	0111 1111 1111 0000 = 0.9995
4094/4096	1111 1111 1110	0000 1111 1111 1110 = 4094	0000 0111 1111 1110 = 2046	1111 1111 1110 0000 = 0.9995	0111 1111 1110 0000 = 0.9990
			•		
			•		
			•		
2049/4096	1000 0000 0001	0000 1000 0000 0001 = 2049	0000 0000 0000 0001 = 1	1000 0000 0001 0000 = 0.5002	0000 0000 0001 0000 = 0.0005
2048/4096	1000 0000 0000	0000 1000 0000 0000 = 2048	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = 0.500	0000 0000 0000 0000 = 0.000
2047/4096	0111 1111 1111	0000 0111 1111 1111 = 2047	1111 1111 1111 1111 = –1	0111 1111 1111 0000 = 0.4998	1111 1111 1111 0000 = –0.0005
			•		
			•		
			•		
1/4096	0000 0000 0001	0000 0000 0000 0001 = 1	1111 1000 0000 0001 = –2047	0000 0000 0001 0000 = 0.0002	1000 0000 0001 0000 = –0.9995
0/4096	0000 0000 0000	0000 0000 0000 0000 = 0	1111 1000 0000 0000 = –2048	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = –1.000

16.15   TRANSFER FUNCTIONS

16.15.1   10-Bit Mode

The ideal transfer function of the ADC module is shown in Figure 16-44. The difference of the input voltages, ($V_{INH} - V_{INL}$), is compared to the reference voltages, ($V_{REFH} - V_{REFL}$).

- The first code transition (**A**) occurs when the input voltage is ($V_{REFH} - V_{REFL}/2048$) or 0.5 LSb
- The '00 0000 0001' code is centered at ($V_{REFH} - V_{REFL}/1024$) or 1.0 LSb (**B**)
- The '10 0000 0000' code is centered at ($512 * (V_{REFH} - V_{REFL})/1024$) (**C**)
- An input voltage less than ($1 * (V_{REFH} - V_{REFL})/2048$) converts as '00 0000 0000' (**D**)
- An input greater than ($2045 * (V_{REFH} - V_{REFL})/2048$) converts as '11 1111 1111' (**E**)

Figure 16-44:   ADC Module Transfer Function (10-Bit Mode)
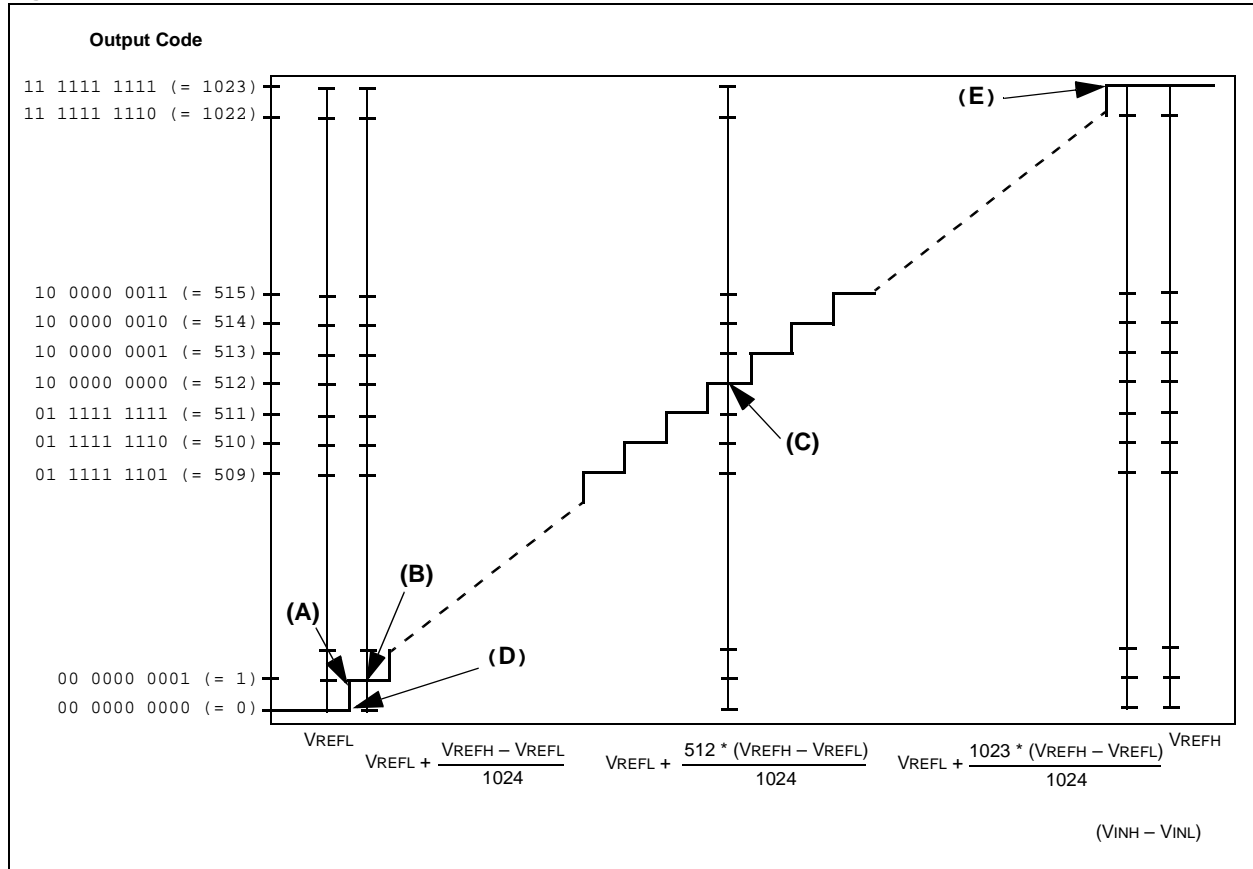
16.15.2 12-Bit Mode

The ideal transfer function of the ADC is shown in Figure 16-45. The difference of the input voltages, (V_{INH} – V_{INL}), is compared to the reference voltages, (V_{REFH} – V_{REFL}).

- The first code transition (**A**) occurs when the input voltage is (V_{REFH} – V_{REFL}/8192) or 0.5 LSb
- The '00 0000 0001' code is centered at (V_{REFH} – V_{REFL}/4096) or 1.0 LSb (**B**)
- The '10 0000 0000' code is centered at (2048 * (V_{REFH} – V_{REFL}))/4096 (**C**)
- An input voltage less than (1 * (V_{REFH} – V_{REFL})/8192) converts as '00 0000 0000' (**D**)
- An input greater than (8192 * (V_{REFH} – V_{REFL})/8192) converts as '11 1111 1111' (**E**)

Figure 16-45:   Analog-to-Digital Transfer Function (12-Bit Mode)
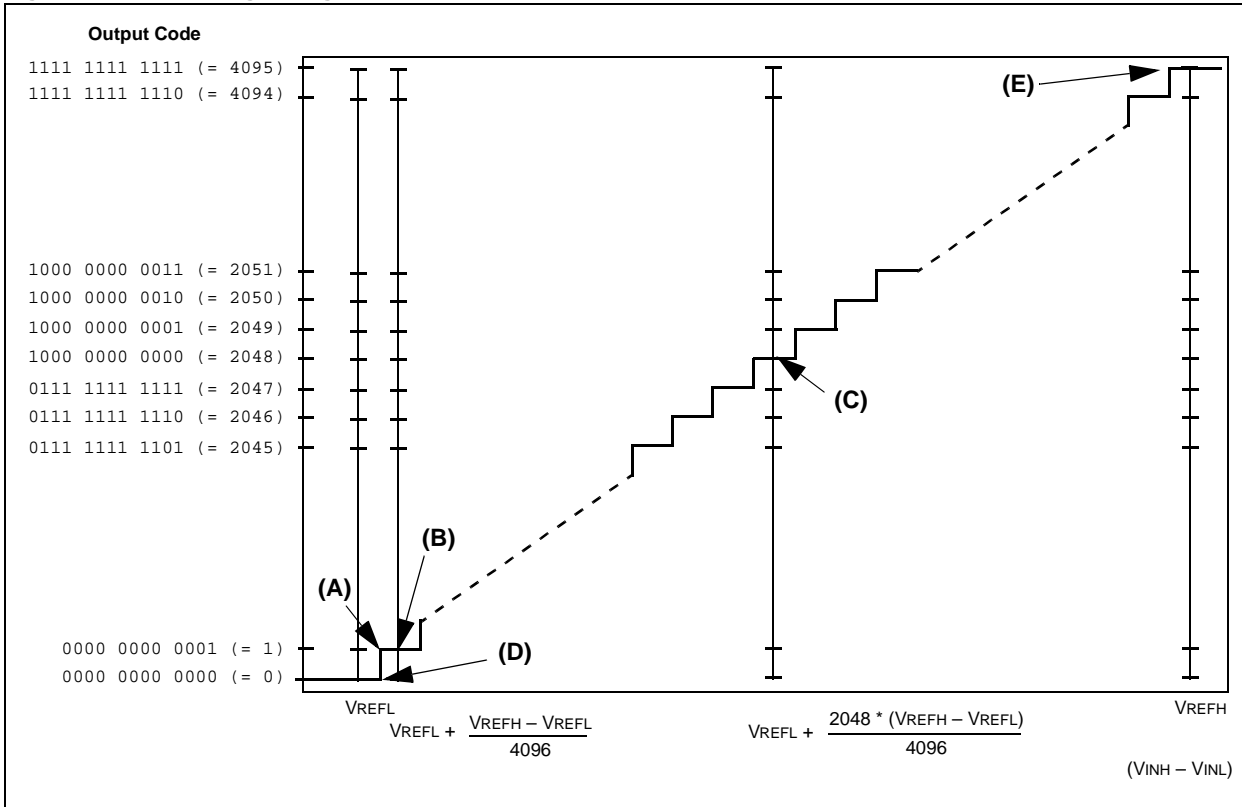
16.16 ADC ACCURACY/ERROR

Refer to the **"Electrical Characteristics"** chapter of the specific device data sheet for more information on the INL, DNL, gain and offset errors. In addition, see **Section 16.21 "Related Application Notes"** for a list of documents that discuss ADC accuracy.

16.17 CONNECTION CONSIDERATIONS

Since the analog inputs employ ESD protection, they have diodes to VDD and VSS. As a result, the analog input must be between VDD and VSS. If the input voltage exceeds this range by greater than 0.3V (either direction), one of the diodes becomes forward biased and it may damage the device if the input current specification is exceeded.

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

16.18 OPERATION DURING SLEEP AND IDLE MODES

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

16.18.1 CPU Sleep Mode without RC Analog-to-Digital Clock

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

If Sleep occurs in the middle of a conversion, the conversion is aborted unless the ADC is clocked from its internal RC clock generator. The converter does not resume a partially completed conversion on exiting from Sleep mode.

Register contents are not affected by the device entering or leaving Sleep mode.

16.18.2 CPU Sleep Mode with RC Analog-to-Digital Clock

The ADC module can operate during Sleep mode if the Analog-to-Digital clock source is set to the internal Analog-to-Digital RC oscillator (ADRC = 1). This eliminates digital switching noise from the conversion. When the conversion is completed, the DONE bit is set and the result is loaded into the ADC result buffer, ADCxBUF0.

If the ADC interrupt is enabled (ADxIE = 1), the device wakes up from Sleep when the ADC interrupt occurs. Program execution resumes at the ADC Interrupt Service Routine (ISR) if the ADC interrupt is greater than the current CPU priority. Otherwise, execution continues from the instruction after the PWRSAV instruction that placed the device in Sleep mode.

If the ADC interrupt is not enabled, the ADC module is turned off, although the ADON bit remains set.

To minimize the effects of digital noise on the ADC module operation, the user must select a conversion trigger source that ensures the Analog-to-Digital conversion takes place in Sleep mode. The automatic conversion trigger option can be used for sampling and conversion in Sleep (SSRCG = 0 and SSRC<2:0> = 111). To use the automatic conversion option, the ADON bit must be set in the instruction before the PWRSAV instruction.

> **Note:** For the ADC module to operate in Sleep, the ADC clock source must be set to RC (ADRC = 1).

16.18.3 ADC Operation During CPU Idle Mode

For the Analog-to-Digital conversion, the ADSIDL bit (ADxCON1<13>) selects if the ADC module stops or continues on Idle. If ADSIDL = 0, the ADC module continues normal operation when the device enters Idle mode. If the ADC interrupt is enabled (ADxIE = 1), the device wakes up from Idle mode when the ADC interrupt occurs. Program execution resumes at the ADC Interrupt Service Routine if the ADC interrupt is greater than the current CPU priority. Otherwise, execution continues from the instruction after the PWRSAV instruction that placed the device in Idle mode.

If ADSIDL = 1, the ADC module stops in Idle. If the device enters Idle mode in the middle of a conversion, the conversion is aborted. The converter does not resume a partially completed conversion on exiting from Idle mode.

16.19 EFFECTS OF A RESET

A device Reset forces all registers to their Reset state. This forces the ADC module to be turned off and any conversion in progress to be aborted. All pins that are multiplexed with analog inputs are configured as analog inputs. The corresponding TRIS bits are set.

The ADCxBUF0 through ADCxBUFF registers are not initialized during a Reset and contain unknown data.

16.20 DESIGN TIPS

Question 1: **How can I optimize the system performance of the ADC module?**

Answer: Here are three suggestions for optimizing performance:

1. Make sure you meet all of the timing specifications. If you are turning the ADC module off and on, there is a minimum delay you must wait before taking a sample. If you are changing input channels, there is a minimum delay you must wait for this as well. Finally, there is T_{AD}, which is the time selected for each bit conversion. T_{AD} is selected in ADxCON3 and must be within a range, as specified in the **"Electrical Characteristics"** chapter of the specific device data sheet. If T_{AD} is too short, the result may not be fully converted before the conversion is terminated. If T_{AD} is too long, the voltage on the sampling capacitor can decay before the conversion is complete. These timing specifications are provided in the **"Electrical Characteristics"** chapter of the specific device data sheet.

2. Often, the source impedance of the analog signal is high (greater than 10 kΩ), so the current drawn from the source to charge the sample capacitor can affect accuracy. If the input signal does not change too quickly, try putting a 0.1 μF capacitor on the analog input. This capacitor charges to the analog voltage being sampled and supplies the instantaneous current needed to charge the 4.4 pF internal holding capacitor.

3. Put the device into Sleep mode before the start of the Analog-to-Digital conversion. The RC clock source selection is required for conversions in Sleep mode. This technique increases accuracy because digital noise from the CPU and other peripherals is minimized.

Question 2: **Do you know of a good reference on ADCs?**

Answer: A good reference for understanding Analog-to-Digital conversions is the "*Analog-Digital Conversion Handbook*" third edition, published by Prentice Hall (ISBN 0-13-03-2848-0).

Question 3: **My combination of channels/sample and samples/interrupt is greater than the size of the buffer. What will happen to the buffer?**

Answer: This configuration is not recommended. The buffer will contain unknown results.

16.21   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 dsPIC33E/PIC24E product family, but the concepts are pertinent and can be used with modification and possible limitations. The current application notes related to the Analog-to-Digital Converter (ADC) module are:

Title	Application Note #
Using the Analog-to-Digital (A/D) Converter	AN546
Four-Channel Digital Voltmeter with Display and Keyboard	AN557
Understanding A/D Converter Performance Specifications	AN693
Using the dsPIC30F for Sensorless BLDC Control	AN901
Using the dsPIC30F for Vector Control of an ACIM	AN908
Sensored BLDC Motor Control Using the dsPIC30F2010	AN957
An Introduction to AC Induction Motor Control Using the dsPIC30F MCU	AN984
Achieving Higher ADC Resolution Using Oversampling	AN1152

> **Note:**   Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the dsPIC33E/PIC24E family of devices.

16.22   REVISION HISTORY

Revision A (January 2010)

This is the initial released version of this document.

Revision B (March 2011)

This revision includes the following updates:

- All code examples have been revised (see Example 16-1 through Example 16-8)
- Distinctions regarding devices *without* DMA, and for devices *with* DMA that have the ADC DMA Enable bit (ADDMAEN) **clear** have been added throughout the document. Affected content includes:
  - Third and last paragraphs of **Section 16.1 "Introduction"**
  - Title of Figure 16-1 (**Figure 16-2** was removed)
  - First paragraph of **Section 16.2.1 "ADC Result Buffer"**
  - Note 3 and bits 6-2 in the ADCx Control Register (see Register 16-2)
  - **Section 16.10 "Sample and Conversion Sequence Examples for Devices without DMA and for Devices with DMA but with ADC DMA Enable Bit (ADDMAEN) Clear"** title
- Distinctions regarding devices *without* DMA, and for devices *with* DMA that have the ADC DMA Enable bit (ADDMAEN) **set** have been added throughout the document. Affected content includes:
  - First paragraph of **Section 16.4.7 "Sample and Conversion Operation (SMPI) Bits"**
  - Added a note box to **Section 16.7.1 "Using DMA in the Scatter/Gather Mode"**
  - Fourth paragraph of **Section 16.6.1 "Fixed Input Selection"**
  - Fourth paragraph of **Section 16.6.3 "Channel Scanning"**
  - **Section 16.11 "Sample and Conversion Sequence Examples for Devices with DMA and with ADDMAEN Bit Set"** title
- Added Note 4 to the ADCx Control Register 3 (see Register 16-3)
- Updated the Automatic Sample and Manual Conversion Sequence (see Figure 16-5)
- Updated the second paragraph to clarify stabilization of ADC results in **Section 16.3.5.2 "External Conversion Trigger"**
- Updated CH3 Sample/Convert in 4-Channel Sequential Sampling (see Figure 16-12)
- Added a note regarding TAD specifications to **Section 16.4.5 "ADC Clock Selection"**
- Updated **Section 16.4.8.2 "Timer Interrupt Trigger"**
- Updated the first paragraph and changed the number of ADC inputs from 16 to 32 in the first note of **Section 16.6.3 "Channel Scanning"**
- Updated the Analog-to-Digital conversion process steps in **Section 16.8 "ADC Configuration Example"**
- Updated the Sequence Select bit value and description for ADDMAEN (see Table 16-13, Table 16-14, Table 16-15, Table 16-16, Table 16-17, and Table 16-18)
- Updated the Sequence Select bit value for SMPI<4:0> (see Table 16-15, Table 16-16, and Table 16-17)
- Updated the MUXA Input Select bit value and description for CHOSA<3:0> and CHONA, and the MUXB Input Select bit value and description for CHOSB<3:0> and CH123NB<1:0> (see Table 16-18)
- Changed ADC1BUF2 to ADC1BUFE (see Figure 16-27)
- Removed ADC1BUF8 from Converting Two Sets of Two Inputs Using Alternating Input Selections (see Figure 16-29)
- Changed the ADCxBUF0 register reference to ADCxBUF0-ADCxBUFF in **Section 16.19 "Effects of a Reset"**
- Updated the SMPI bit range from <4:0> to <3:0> for AD2CON2 in the ADC1 and ADC2 Register Map (see Table 16-27)
- Updates to formatting and minor typographical changes were incorporated throughout the document

Revision C (May 2013)

This revision includes the following updates:

- Figures:
  - Updated Figure 16-1, Figure 16-2 and Figure 16-13
  - Updated the Channel Scan numbers in Figure 16-37 through Figure 16-39
  - Updated the available register settings in Figure 16-37 through Figure 16-39
- Notes:
  - Updated the pin numbers and the number of op amp outputs in Note 2, in **Section 16.1 "Introduction"**
  - Added Note 4 in Figure 16-2
  - Updated the Notes in Register 16-1 through Register 16-9
  - Updated the Note in Table 16-9 and Table 16-10
- Registers:
  - Updated the bit setting descriptions for bit 7-5 and bit 3 in Register 16-1
  - Updated Register 16-5 and Register 16-6
- Sections:
  - Updated the features list in **Section 16.1 "Introduction"**
  - Updated the pin numbers and the number of op amp outputs in the last paragraph on page 16-2 in **Section 16.1 "Introduction"**
  - Added PTG Trigger as a source in the list of sources that the ADC module can use as a conversion trigger, in **Section 16.4.8 "Conversion Trigger Sources"**
  - Added **Section 16.4.8.4 "PTG Trigger"**
  - Added **Section 16.12 "Configuration Examples for Devices with Internal Op Amps"**
  - Removed **Section 16.19 "Special Function Registers"**
- Tables:
  - Removed **Table 16-2: SOC Trigger Selection**
  - Updated the following in Table 16-9:
    - Updated the CH0 +ve Control bits, CHOSA<4:0> to CHOSA<5:0>
    - Updated the CH0 +ve Analog Inputs, AN0 to AN12 as AN0 to AN48
    - Added AN0 and AN25 to the CH2 +ve CH123SA Analog Inputs
    - Added AN6 and AN25 to the CH3 +ve CH123SA Analog Inputs
  - Updated the following in Table 16-10:
    - Updated the CH0 +ve Control bits, CHOSB<4:0> to CHOSB<5:0>
    - Updated the CH0 +ve Analog Inputs, AN0 to AN12 as AN0 to AN48
    - Added AN0 and AN25 to the CH2 +ve CH123SB Analog Inputs
    - Added AN6 and AN25 to the CH3 +ve CH123SB Analog Inputs
  - Updated the following in Table 16-13 through Table 16-19 and Table 16-21 through Table 16-23:
    - Updated CH0SA<3:0> to CH0SA<5:0> in the MUXA Input Select table
    - Updated CH123SA to CH123SA<2:0> in the MUXA Input Select table
    - Updated CH0SB<3:0> to CH0SB<5:0> in the MUXB Input Select table
    - Updated CH123SB to CH123SB<2:0> in the MUXB Input Select table
- Updates to formatting and minor typographical changes were incorporated throughout the document

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

• Microchip products meet the specification contained in their particular Microchip Data Sheet.

• Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.

• There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.

• Microchip is willing to work with the customer who is concerned about the integrity of their code.

• Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

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

QUALITY MANAGEMENT SYSTEM

CERTIFIED BY DNV

═ ISO/TS 16949 ═

Trademarks

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.

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

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

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

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

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

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

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

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

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
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 - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

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

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

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

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

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

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

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

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

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

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

ASIA/PACIFIC

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: 91-20-2566-1512
Fax: 91-20-2566-1513

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

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

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

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

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

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

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

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

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

Taiwan - Kaohsiung
Tel: 886-7-213-7828
Fax: 886-7-330-9305

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

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

EUROPE

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

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

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

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

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

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

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

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/12