Analog-to-digital conversion
Digital-to-analog conversion

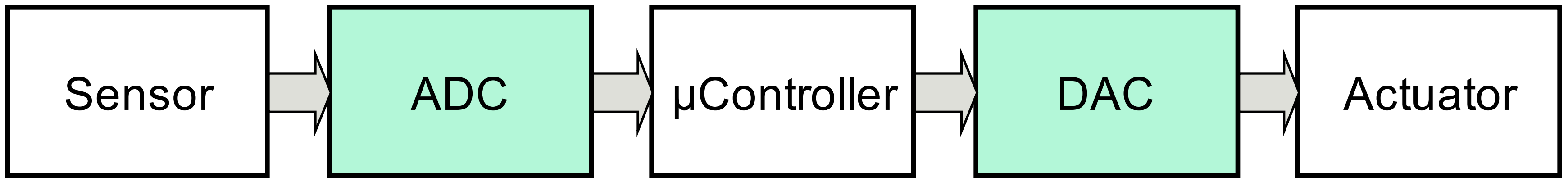
Part 1

ECE 2534

Outline

- Motivation
- Basic idea
- ADC on the PIC32
- Digital-to-analog conversion methods
- Analog-to-digital conversion methods

MOTIVATION:
Need to interface with the real (analog) world



- *Transducer*
  - A device that converts one form of energy to a different form
  - E.g., a photodiode produces an electrical current that is a function of the number of photons striking the device

- *Example phenomena (and sensors)*

Light intensity	(photodiode, etc.)
Sound intensity	(microphone)
Angle	(potentiometer)
Temperature	(thermistor)
Force	(accelerometer, strain gauge)
Rotational motion	(gyroscope)
Radio signal	(antenna)

- *Example outputs (and actuators)*

Position, orientation	(motor)
Sound	(acoustic speaker)
Current, etc.	(relay, solenoid, SCR, etc.)

THE BASIC IDEA:

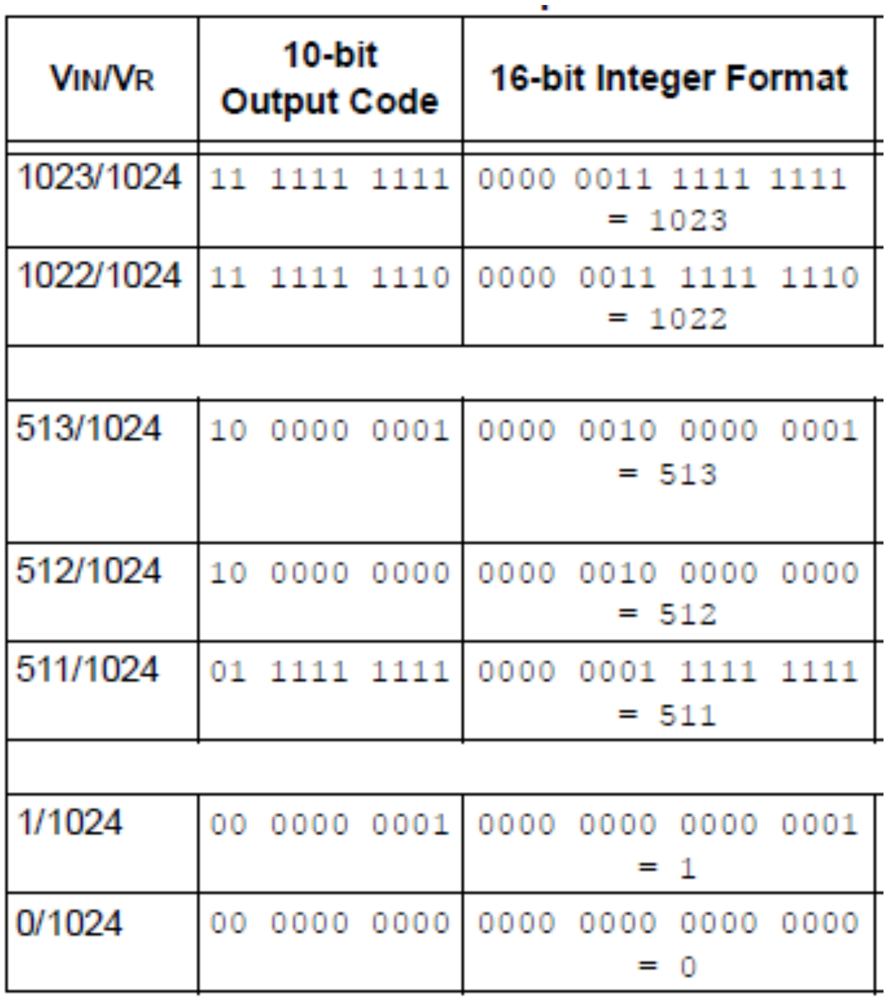
Typical analog-to-digital conversion

$$D = \left\lfloor \left(\frac{V_{In} - V_{RefLow}}{V_{RefHigh} - V_{RefLow}} \right) \times 2^n \right\rfloor$$

- V_{In} = analog input voltage
- D = binary output value
- n = number of bits
- $V_{RefHigh}$ = upper reference voltage
- V_{RefLow} = lower reference voltage = “offset”
- $V_{RefHigh} - V_{RefLow}$ = “span”
- “resolution” = voltage difference corresponding to 1 increment in the digital value $= \left(\frac{V_{RefHigh} - V_{RefLow}}{2^n} \right)$

Example ADC values *(PIC32 preview!)*

- n = 10 bits
- V_{RefLow} = 0 V



VIN/VR	10-bit Output Code	16-bit Integer Format
1023/1024	11 1111 1111	0000 0011 1111 1111 = 1023
1022/1024	11 1111 1110	0000 0011 1111 1110 = 1022
513/1024	10 0000 0001	0000 0010 0000 0001 = 513
512/1024	10 0000 0000	0000 0010 0000 0000 = 512
511/1024	01 1111 1111	0000 0001 1111 1111 = 511
1/1024	00 0000 0001	0000 0000 0000 0001 = 1
0/1024	00 0000 0000	0000 0000 0000 0000 = 0

Analog to digital:

$$D = \left\lfloor \left(\frac{V_{In} - V_{RefLow}}{V_{RefHigh} - V_{RefLow}} \right) \times 2^{n} \right\rfloor$$

Digital to analog:

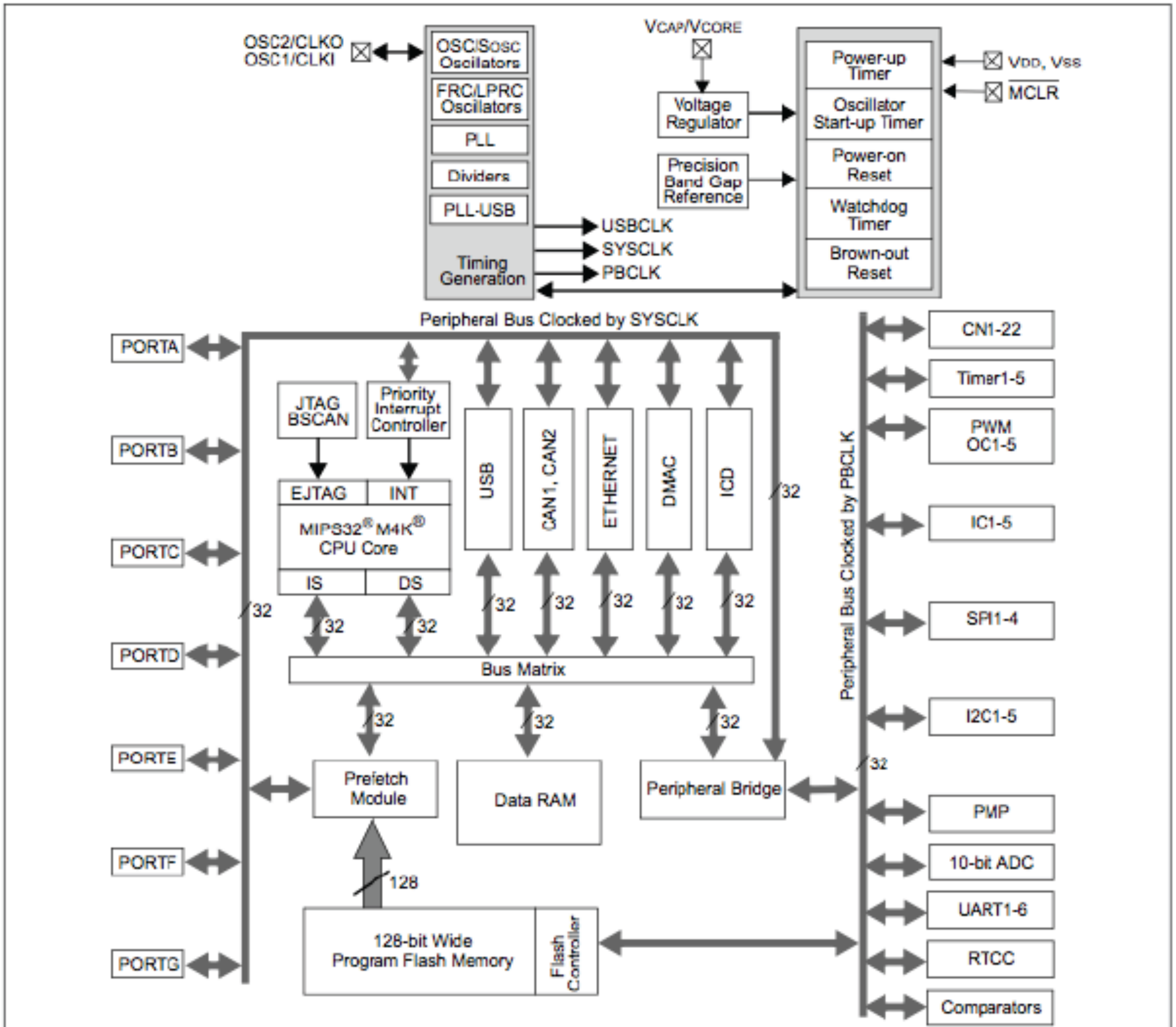
$$V_{Out} = \left(\frac{D}{2^{n}} \right) \left(V_{RefHigh} - V_{RefLow} \right) + V_{RefLow}$$

Outline

- Motivation
- Basic idea
- ADC on the PIC32

- Digital-to-analog conversion methods
- Analog-to-digital conversion methods

FIGURE 1-1: BLOCK DIAGRAM(1,2)



Note 1: Some features are not available on all device variants.

2: BOR functionality is provided when the on-board voltage regulator is enabled.

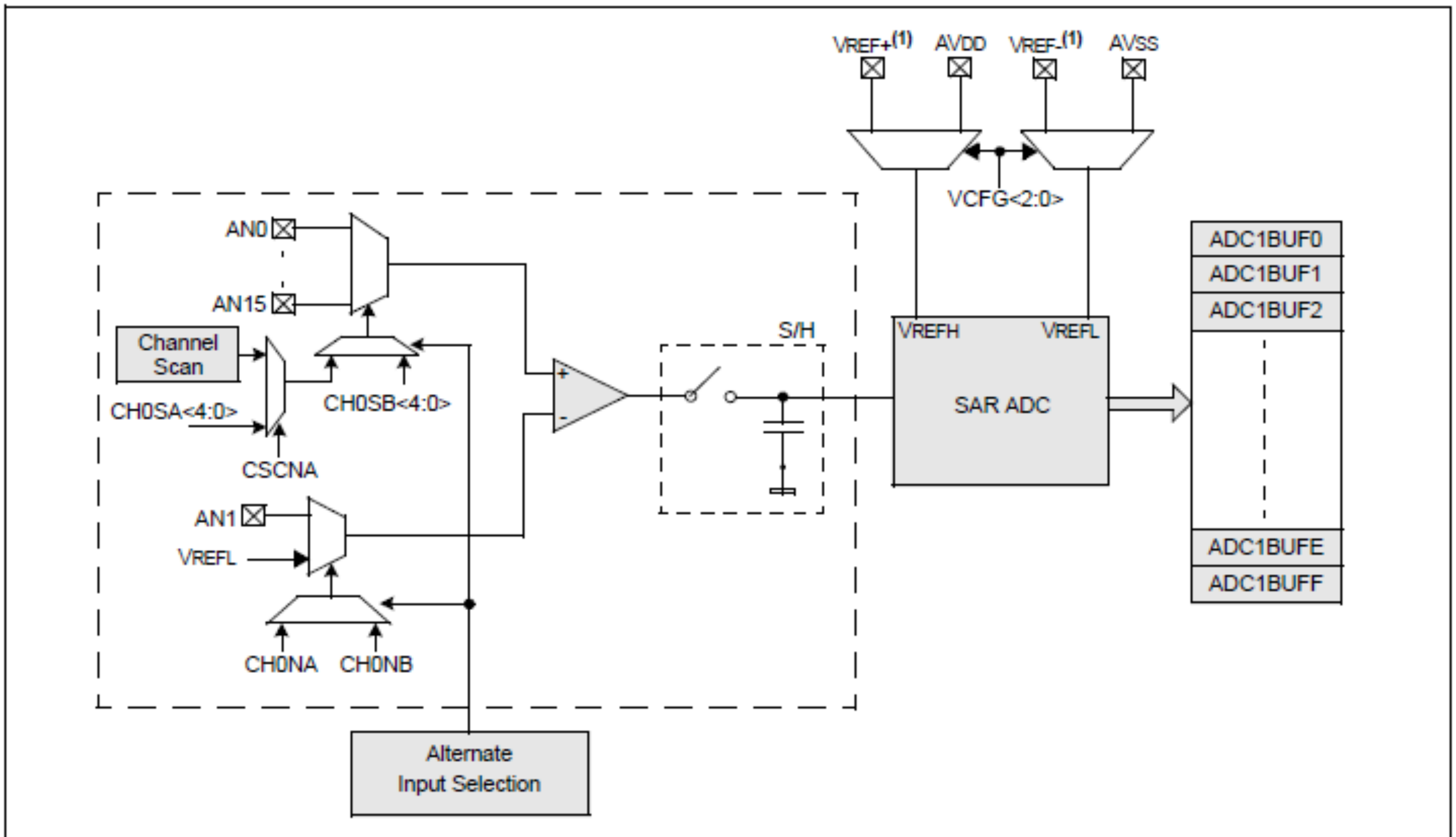
(PIC32 datasheet, section 1)

ADC on the PIC32

- n = 10, so there are 1024 different levels
- Typically $V_{RefHigh}$ = 3.3 V and V_{RefLow} = 0 V
  - So, span = 3.3 V and resolution = 3.22 mV

- The PIC32 also allows the designer to provide external reference voltages
- The PIC32 allows up to 16 analog inputs
- The Digilent board provides access to 10 of the analog inputs via Pmod connectors
- Max voltage allowed on Digilent analog inputs: 3.6 V (<u>not</u> 5-V tolerant)

FIGURE 22-1: ADC1 MODULE BLOCK DIAGRAM



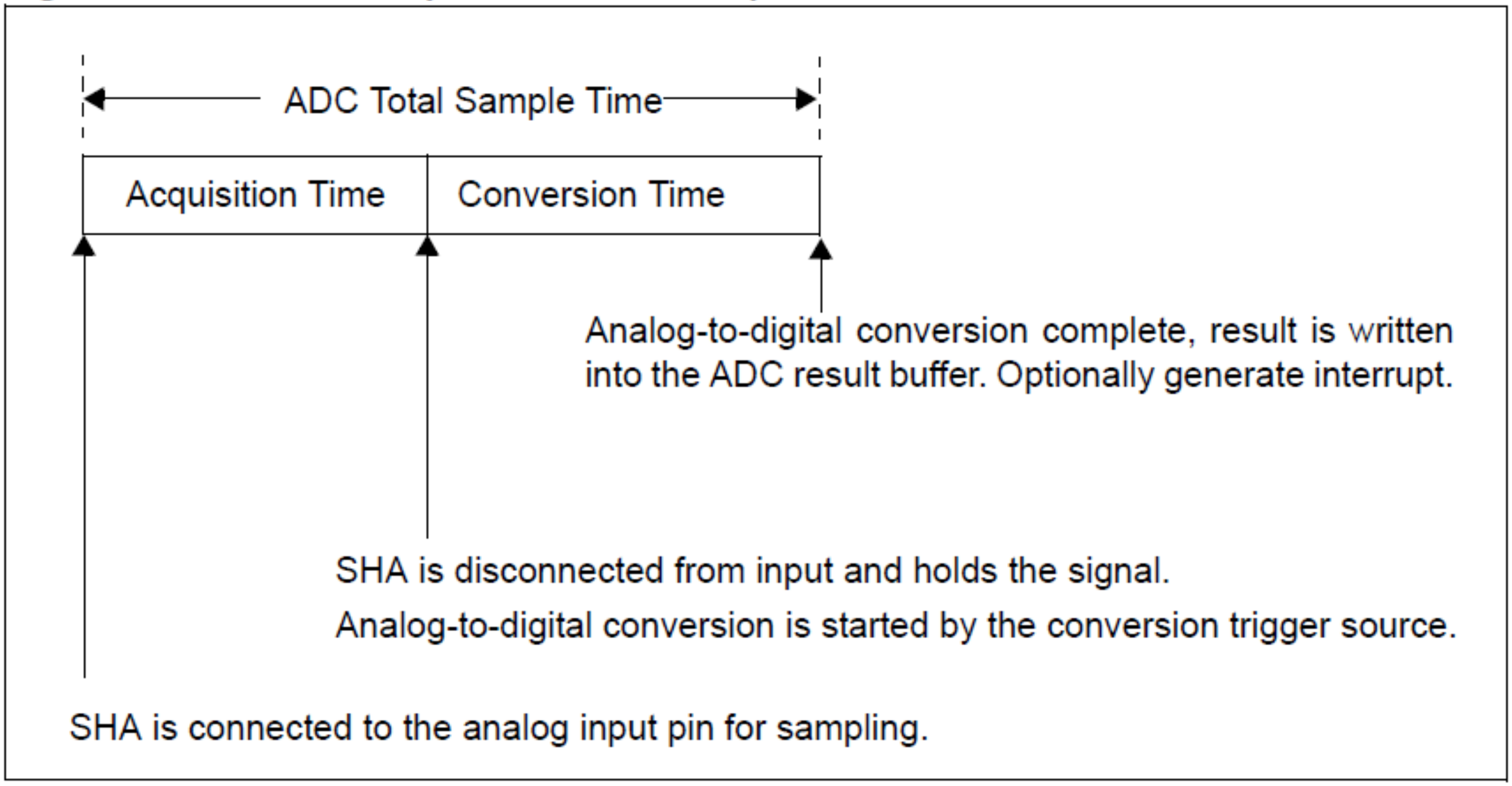
Note 1: VREF+ and VREF- inputs can be multiplexed with other analog inputs.

The PIC32 10-bit Analog-to-Digital Converter (ADC) includes the following features:

- Successive Approximation Register (SAR) conversion
- Up to 16 analog input pins
- External voltage reference input pins
- One unipolar differential Sample-and-Hold Amplifier (SHA)
- Automatic Channel Scan mode
- Selectable conversion trigger source
- 16-word conversion result buffer
- Selectable Buffer Fill modes
- Eight conversion result format options
- Operation during CPU Sleep and Idle modes

- Analog sampling consists of two steps: “acquisition” and “conversion”
- Acquisition time can be controlled manually or automatically

Figure 17-2: ADC Sample/Conversion Sequence


ADC Total Sample Time
Acquisition Time
Conversion Time
Analog-to-digital conversion complete, result is written into the ADC result buffer. Optionally generate interrupt.
SHA is disconnected from input and holds the signal.
Analog-to-digital conversion is started by the conversion trigger source.
SHA is connected to the analog input pin for sampling.

Control Registers for the ADC

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

- **AD1CON1: ADC Control Register 1**
- **AD1CON2: ADC Control Register 2**
- **AD1CON3: ADC Control Register 3**

  The AD1CON1, AD1CON2 and AD1CON3 registers control the operation of the ADC module.

- **AD1CHS: ADC Input Select Register**

  The AD1CHS register selects the input pins to be connected to the SHA.

- **AD1PCFG: ADC Port Configuration Register**(1,2)

  The AD1PCFG register configures the analog input pins as analog inputs or as digital I/O.

- **AD1CSSL: ADC Input Scan Select Register**(1)

  The AD1CSSL register selects inputs to be sequentially scanned.

ADC initialiation

Sect. 17.4

Operation of the ADC module is directed through bit settings in the appropriate registers. The following instructions summarize the actions and the settings. Options and details for each configuration step are provided in subsequent sections.

To configure the ADC module, perform the following steps:

1. Configure the analog port pins in AD1PCFG<15:0> (see 17.4.1).
2. Select the analog inputs to the ADC multiplexers in AD1CHS<32:0> (see 17.4.2).
3. Select the format of the ADC result using FORM<2:0> (AD1CON1<10:8>) (see 17.4.3).
4. Select the sample clock source using SSRC<2:0> (AD1CON1<7:5>) (see 17.4.4).
5. Select the voltage reference source using VCFG<2:0> (AD1CON2<15:13>) (see 17.4.7).
6. Select the Scan mode using CSCNA (AD1CON2<10>) (see 17.4.8).
7. Set the number of conversions per interrupt SMP<3:0> (AD1CON2<5:2>), if interrupts are to be used (see 17.4.9).
8. Set Buffer Fill mode using BUFM (AD1CON2<1>) (see 17.4.10).
9. Select the MUX to be connected to the ADC in ALTS AD1CON2<0> (see 17.4.11).
10. Select the ADC clock source using ADRC (AD1CON3<15>) (see 17.4.12).
11. Select the sample time using SAMC<4:0> (AD1CON3<12:8>), if auto-convert is to be used (see 17-2).
12. Select the ADC clock prescaler using ADCS<7:0> (AD1CON3<7:0>) (see 17.4.12).
13. Turn the ADC module on using AD1CON1<15> (see 17.4.14).

Note: Steps 1 through 12, above, can be performed in any order, but Step 13 must be the final step in every case.

14. To configure ADC interrupt (if required):
    a) Clear the AD1IF bit (IFS1<1>) (see 17.7).
    b) Select ADC interrupt priority AD1IP<2:0> (IPC<28:26>) and subpriority AD1IS<1:0> (IPC<24:24>) if interrupts are to be used (see 17.7).
15. Start the conversion sequence by initiating sampling (see 17.4.15).

Plib functions for the ADC

…/pic32mx/include/peripheral/adc10.h

- OpenADC10() - Configures and enables the ADC module
- EnableADC10() - Turns the ADC on
- AcquireADC10() - Starts sample acquisition for the currently select channel
- ConvertADC10() - Starts a conversion for the acquired sample
- ReadADC10() - Returns the value in the specified location of the ADC result buffer
- CloseADC10() - Disables and turns off the ADC

- SetChanADC10() - Configures the ADC input multiplexers
- ConfigIntADC10() - Configures the priorty and sub-priority for the ADC interrupt and enables the interrupt
- BusyADC10() - Returns the status of the conversion done bit

OpenADC10

Description: This function configures the ADC using the 5 parameters passed to it.

Include: `plib.h`

Prototype:

```
void OpenADC10(unsigned long int config1,
               unsigned long int config2,
               unsigned long int config3,
               unsigned long int configport,
               unsigned long int configscan)
```

Return Value: None

Remarks: This function configures the ADC for the following parameters: Operating mode, Sleep mode behavior, Data output format, Sample Clk Source, VREF source, No of samples/int, Buffer Fill mode, Alternate input sample mod, Auto sample time, Conv clock source, Conv Clock Select bits, Port Config Control bits. Channel select for manual and alternate sample modes is are not configured by this macro.

(See Plib manual)

Output formats

FORM<2:0>: Data Output Format bits

011 = Signed Fractional 16-bit (DOUT = 0000 0000 0000 0000 sddd dddd dd00 0000)
010 = Fractional 16-bit (DOUT = 0000 0000 0000 0000 dddd dddd dd00 0000)
001 = Signed Integer 16-bit (DOUT = 0000 0000 0000 0000 ssss sssd dddd dddd)
000 = Integer 16-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)
111 = Signed Fractional 32-bit (DOUT = sddd dddd dd00 0000 0000 0000 0000)
110 = Fractional 32-bit (DOUT = dddd dddd dd00 0000 0000 0000 0000 0000)
101 = Signed Integer 32-bit (DOUT = ssss ssss ssss ssss ssss sssd dddd dddd)
100 = Integer 32-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)

Summary (so far)

- Motivation
- Basic idea
- ADC on the PIC32
- Digital-to-analog conversion methods
- Analog-to-digital conversion methods