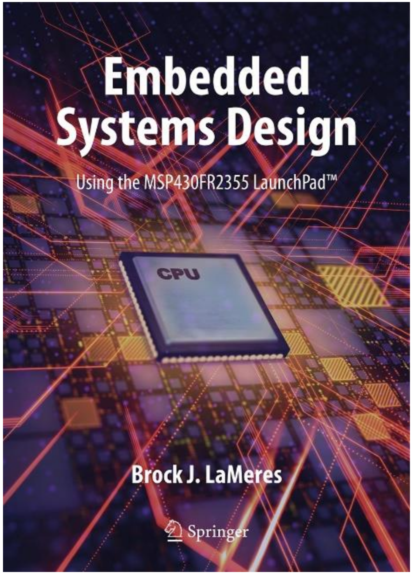
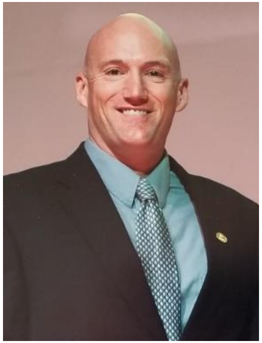
Embedded Systems Design

Chapter 15: Analog to Digital Converters

15.1 Analog to Digital Converters - Overview

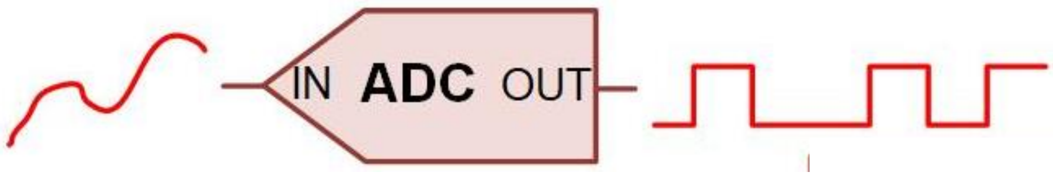




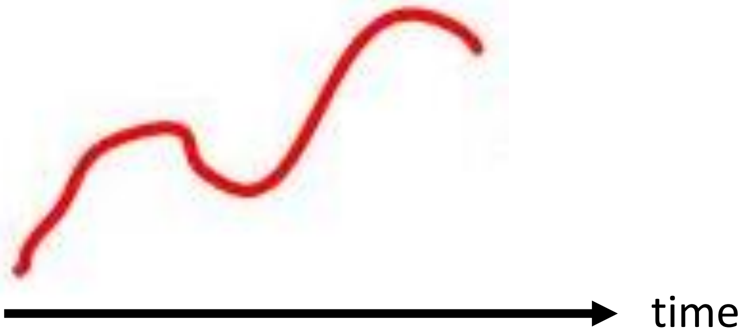
Brock J. LaMeres, Ph.D.

15.1 ANALOG TO DIGITAL CONVERTERS

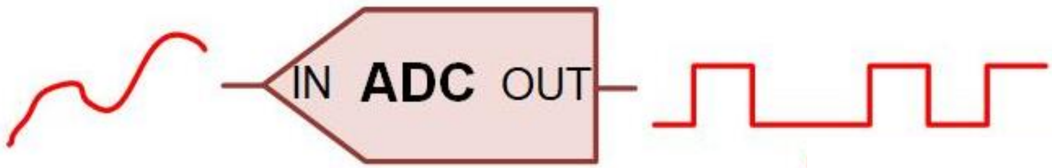
- **Analog-to-Digital Converter (ADC or A2D)** – a circuit that takes in an analog voltage and produces a digital representation of its value.

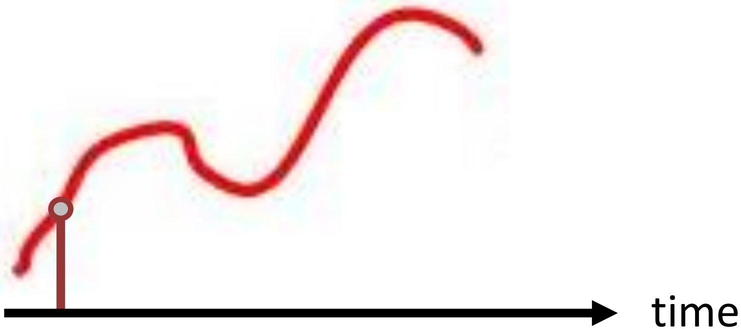


15.1 ANALOG TO DIGITAL CONVERTERS



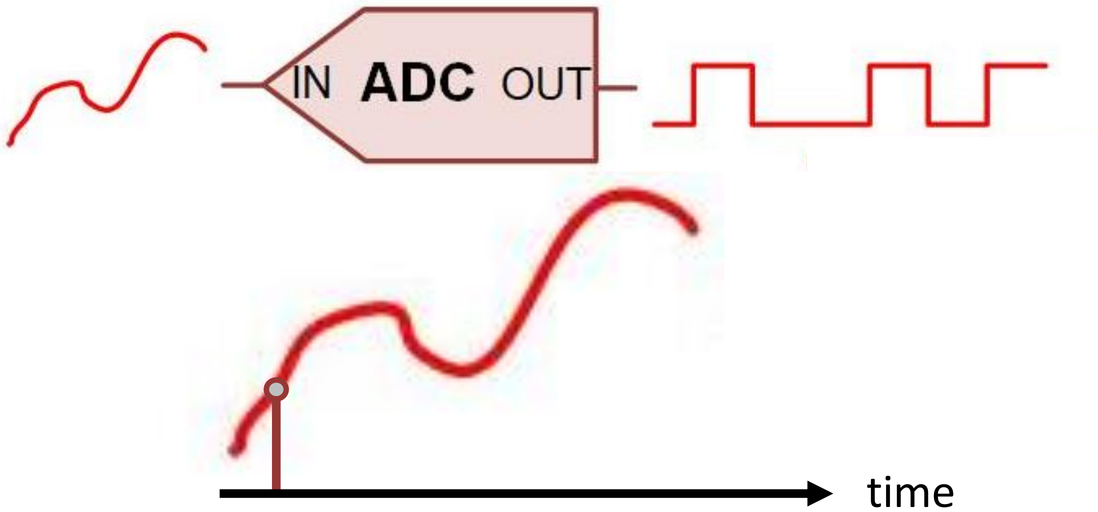
15.1 ANALOG TO DIGITAL CONVERTERS





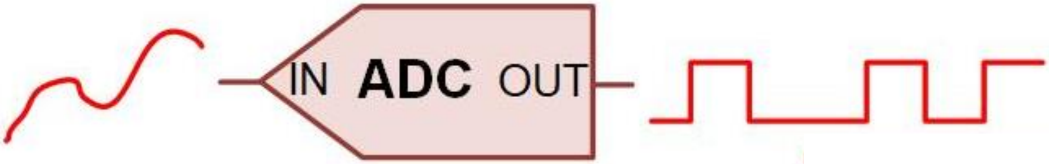
- **Sample** – the action of duplicating the voltage value of the input signal.
- **Hold** – the action of holding the voltage for a brief amount of time by use of a capacitor.

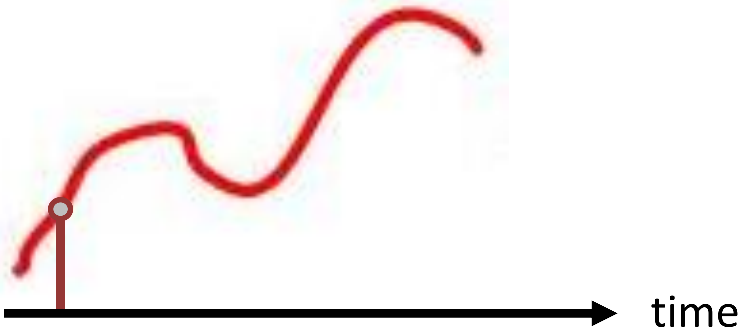
15.1 ANALOG TO DIGITAL CONVERTERS



- The **Sample** is accomplished by momentarily connecting the signal to a capacitor in order to charge it up to the same voltage.
- The **Hold** is accomplished by disconnecting the signal and allowing the conversion to be conducted on the voltage on the capacitor.

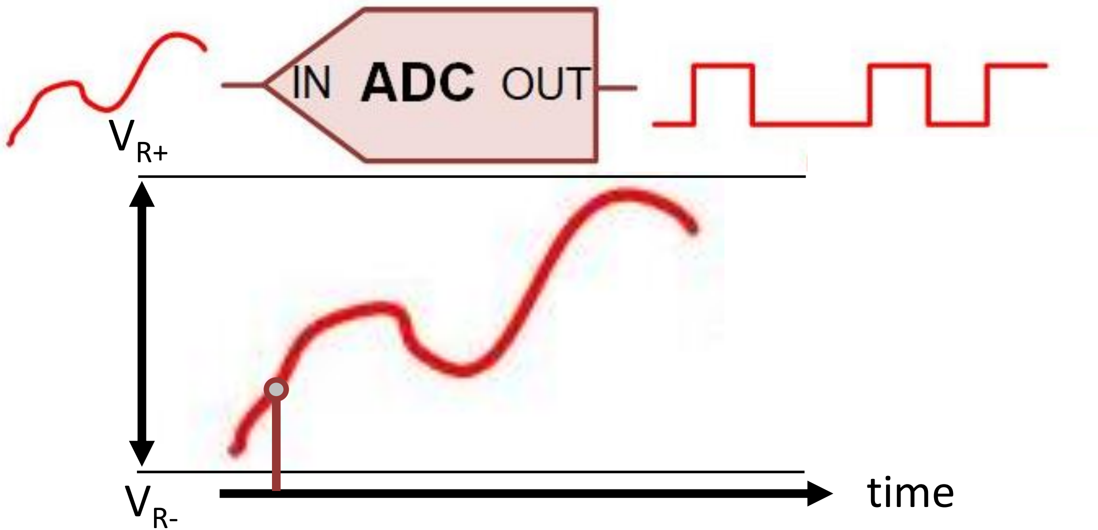
15.1 ANALOG TO DIGITAL CONVERTERS





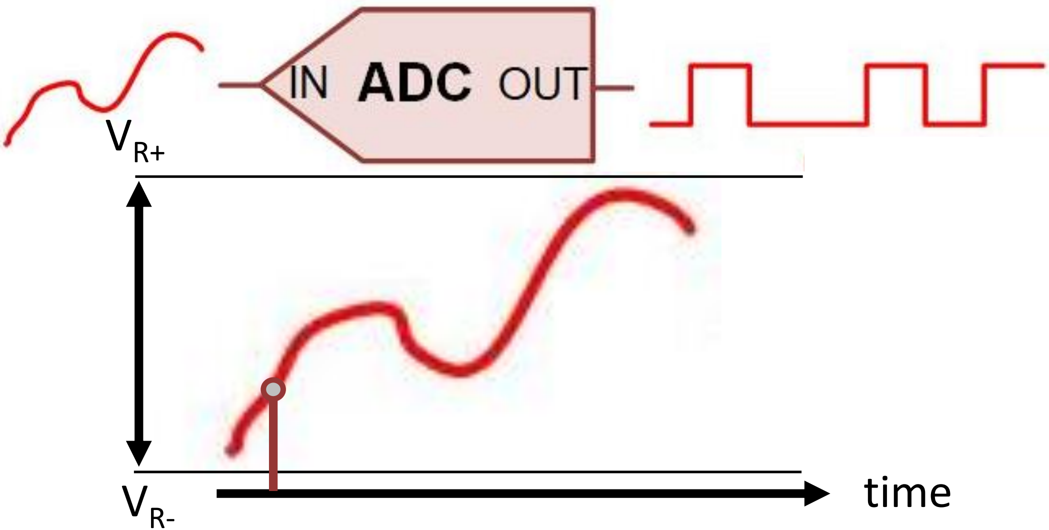
- The **Sample-and-Hold** circuitry is designed so that this can be accomplished very quickly so that it can disconnect from the input signal as soon as possible to avoid altering its signal integrity.

15.1 ANALOG TO DIGITAL CONVERTERS



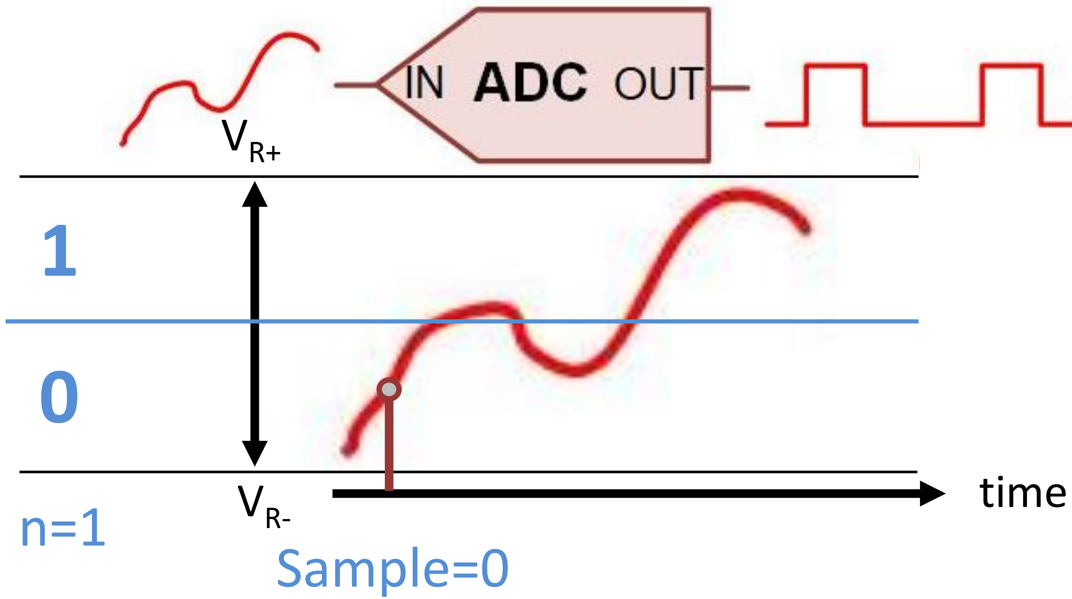
- **Input Voltage Range –** The voltage is digitized within a range of voltages from:
  - Voltage Reference High (V_{R+})
  - Voltage Reference High (V_{R-})

15.1 ANALOG TO DIGITAL CONVERTERS



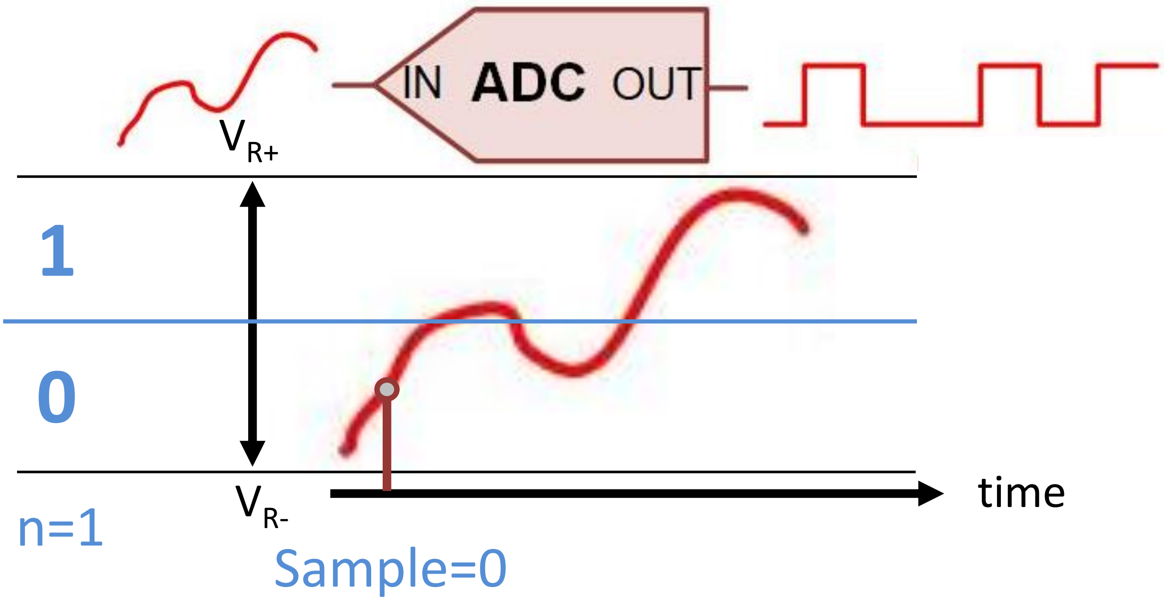
- The goal of the conversion is to convert the analog voltage into a digital number.
- This is called *digitizing, quantizing, or discretizing*.

15.1 ANALOG TO DIGITAL CONVERTERS



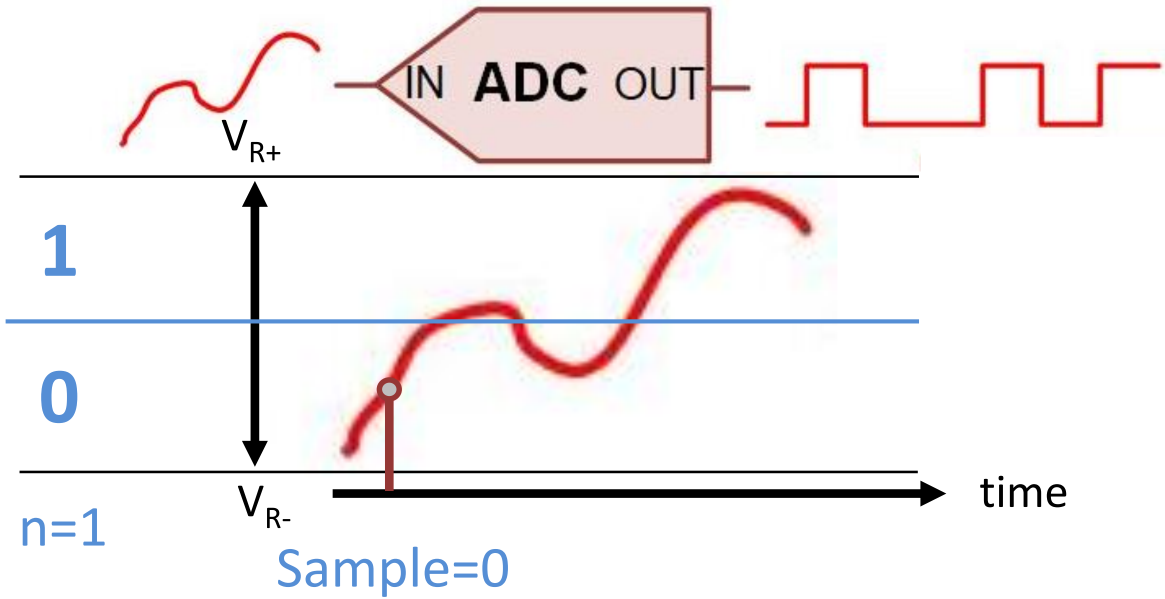
- *n* represents the number of bits in the digital value of the conversion.

15.1 ANALOG TO DIGITAL CONVERTERS



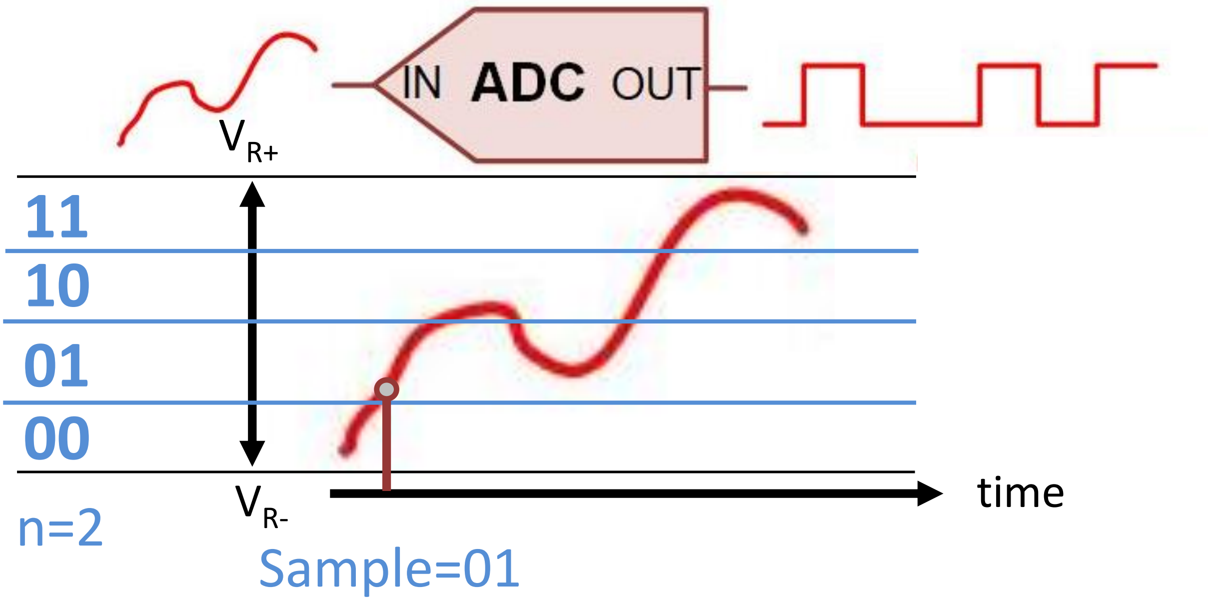
- ***n*** represents the number of bits in the digital value of the conversion.
- The input voltage range is divided into 2^n discrete zones.

15.1 ANALOG TO DIGITAL CONVERTERS



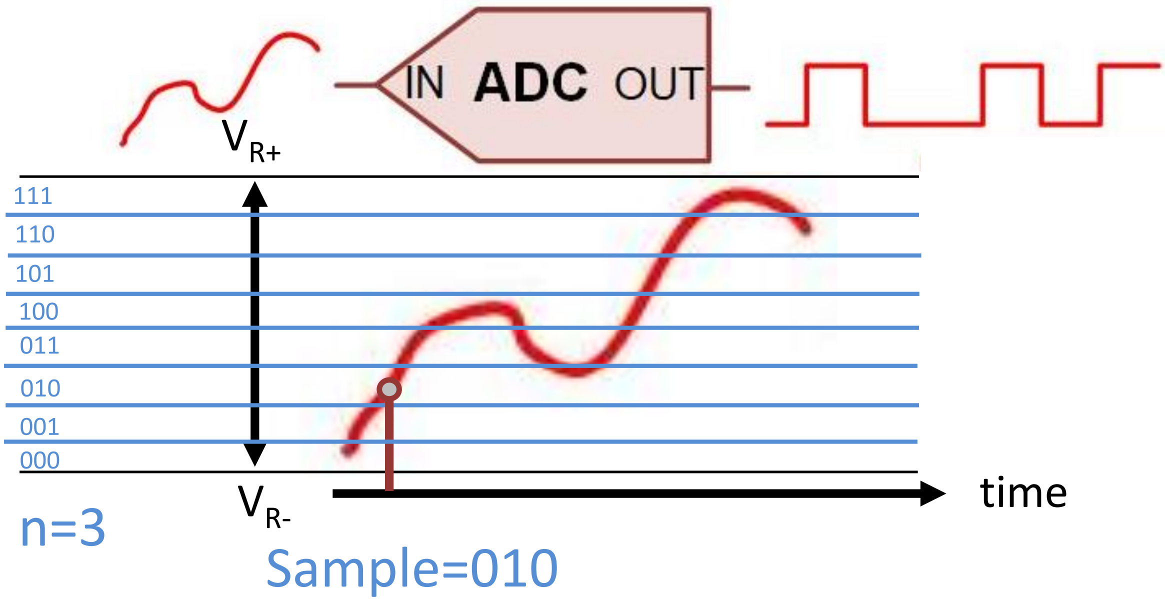
- The larger the n, the closer the digital value is to the actual analog voltage.

15.1 ANALOG TO DIGITAL CONVERTERS



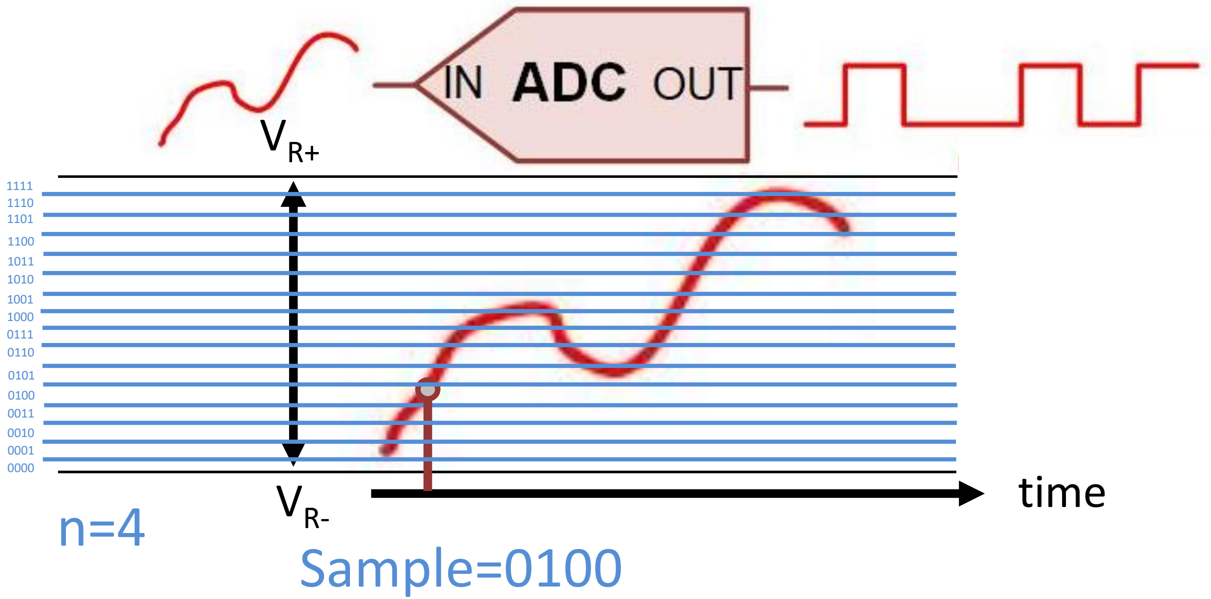
- The larger the n, the closer the digital value is to the actual analog voltage.

15.1 ANALOG TO DIGITAL CONVERTERS



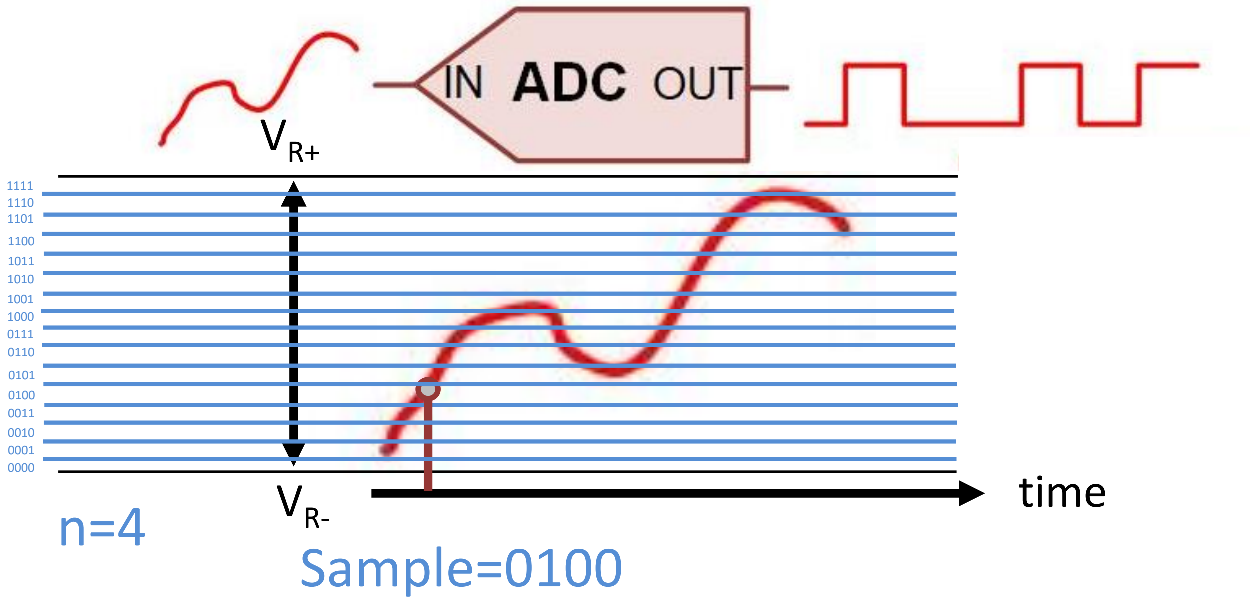
- The larger the n, the closer the digital value is to the actual analog voltage.

15.1 ANALOG TO DIGITAL CONVERTERS



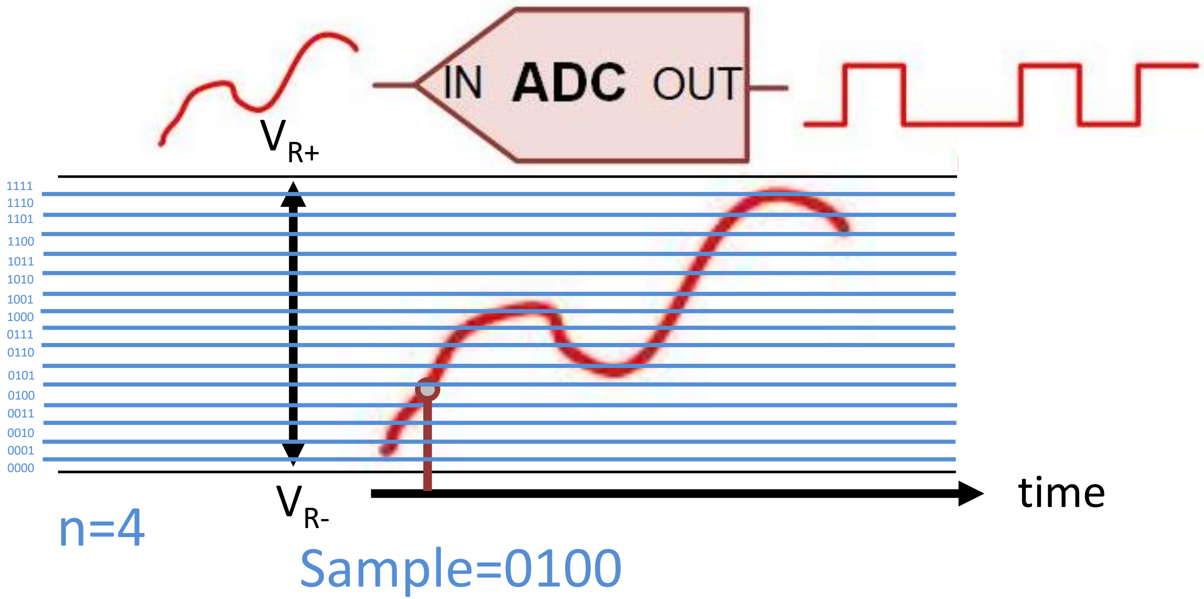
- The larger the n, the closer the digital value is to the actual analog voltage.

15.1 ANALOG TO DIGITAL CONVERTERS



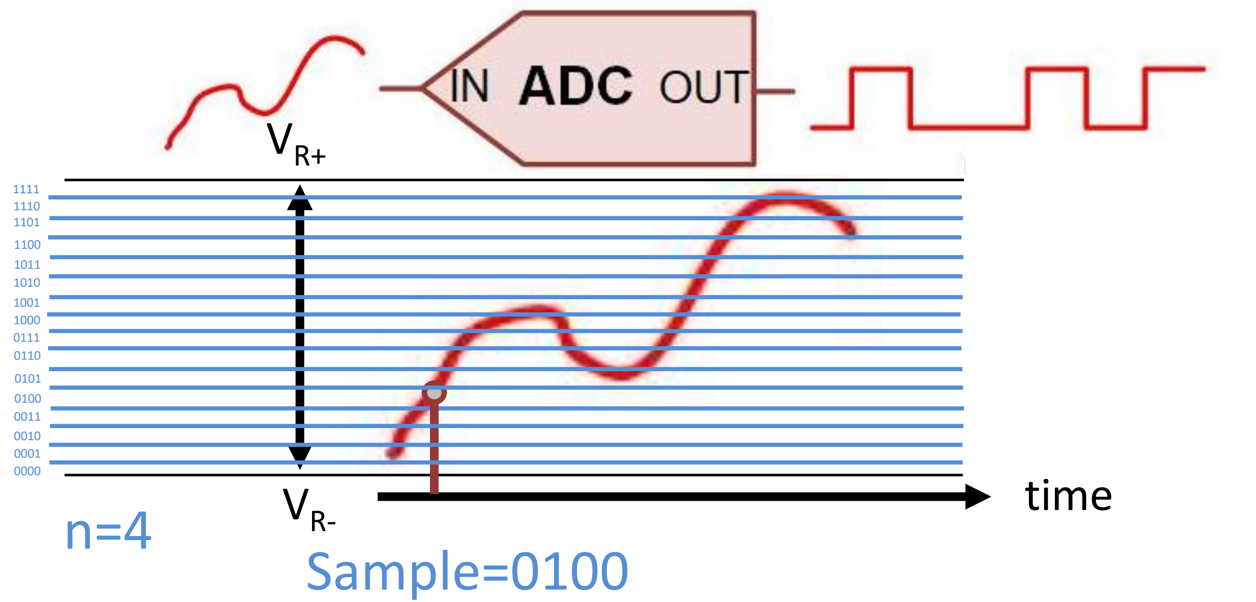
- The number of bits ***n*** is called the ADC's **resolution.**

15.1 ANALOG TO DIGITAL CONVERTERS



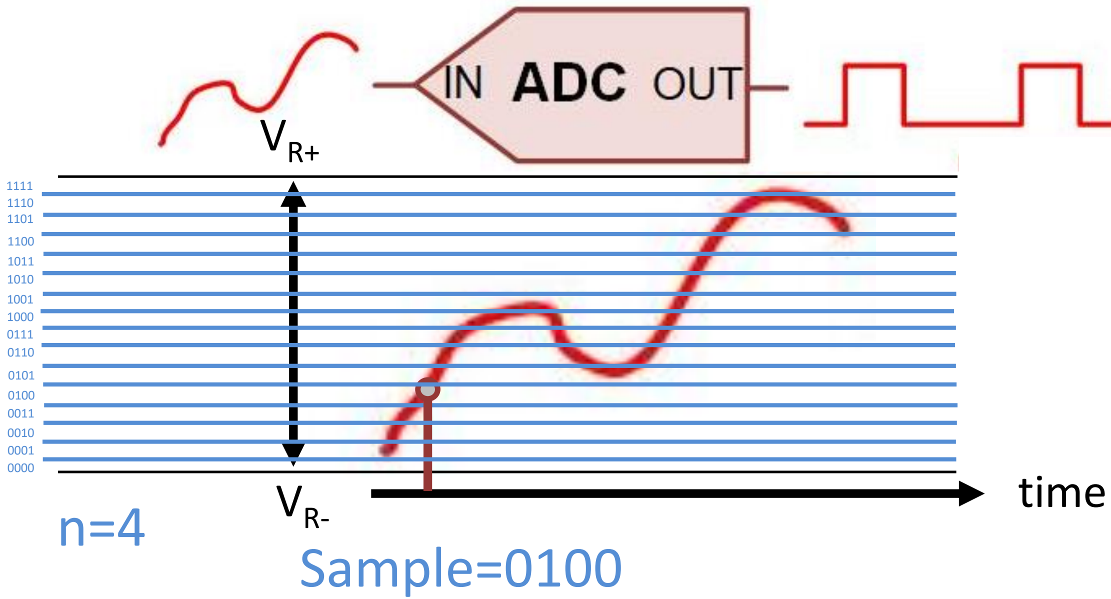
- The number of bits ***n*** is called the ADC's **resolution.**
- MCU's typically have ADC's with resolutions of 8 to 16 bits.

15.1 ANALOG TO DIGITAL CONVERTERS



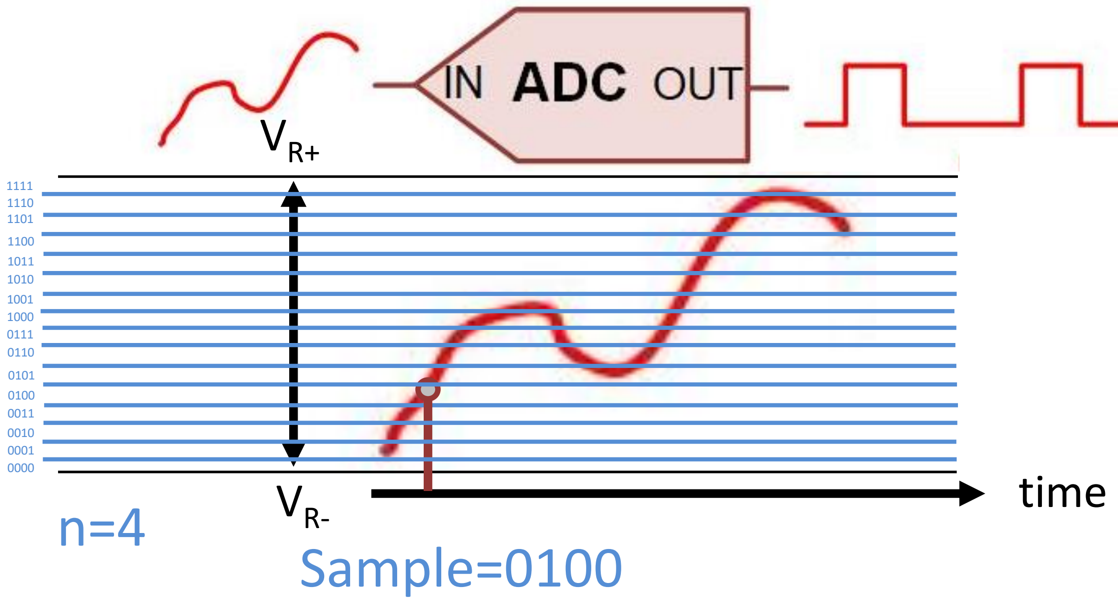
- The number of bits ***n*** is called the ADC's **resolution.**
- MCU's typically have ADC's with resolutions of 8 to 16 bits.
- The MSP430 has up to 12-bits of resolution.

15.1 ANALOG TO DIGITAL CONVERTERS



- The **precision** of an ADC is the smallest voltage that the LSB of the digital output can represent.

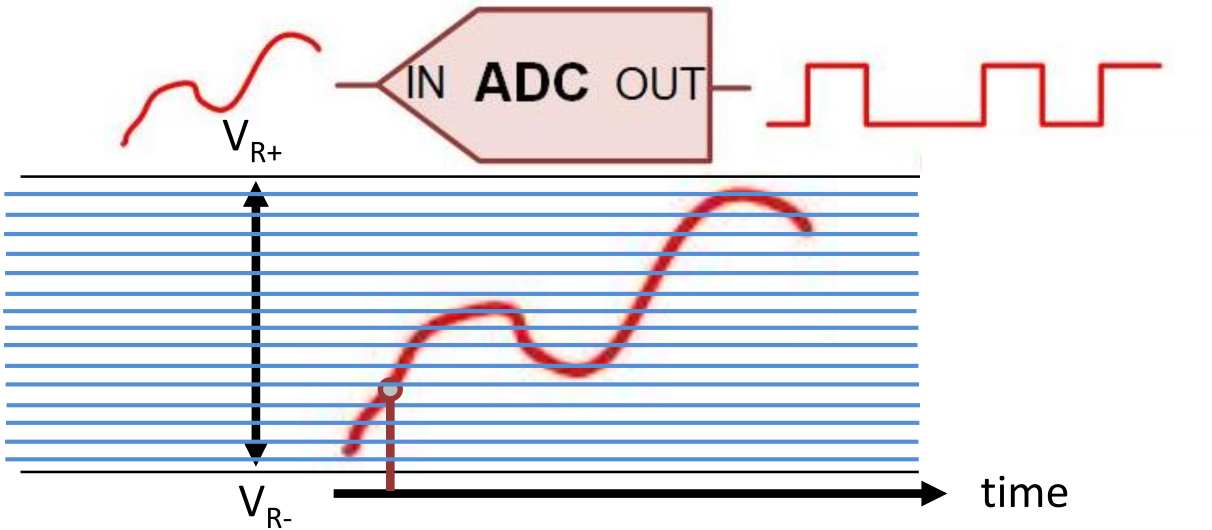
15.1 ANALOG TO DIGITAL CONVERTERS



- The **precision** of an ADC is the smallest voltage that the LSB of the digital output can represent.
- This is found by dividing the input voltage range by the number of discrete zones.

$$\textbf{Precision} = \frac{V_{R+} - V_{R-}}{2^n}$$

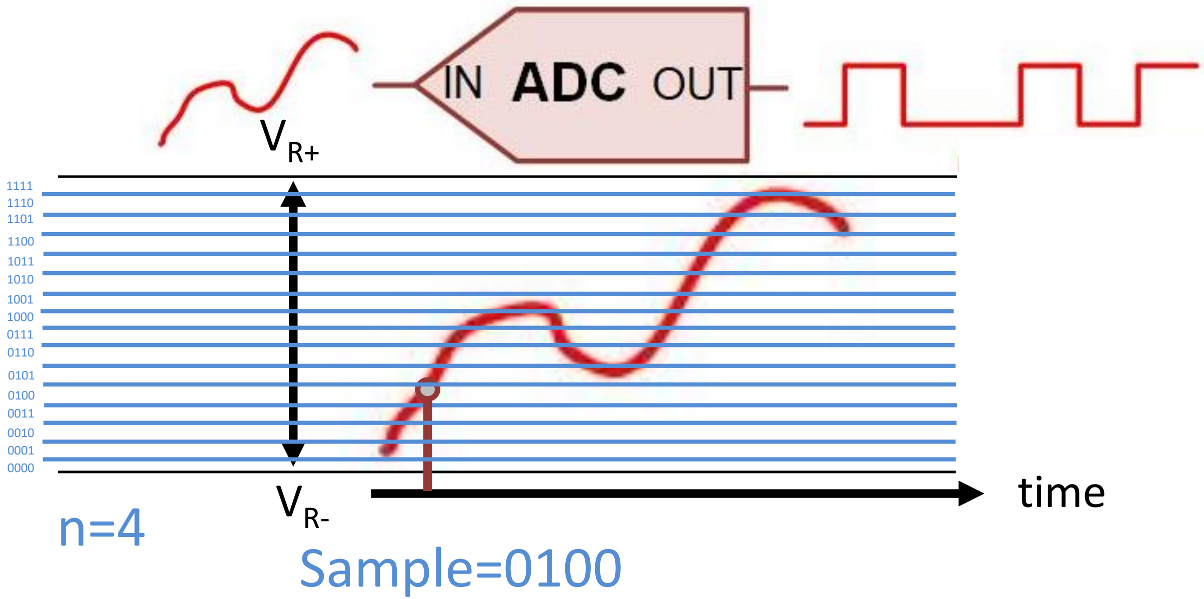
15.1 ANALOG TO DIGITAL CONVERTERS



Example: What is the precision of a 12-bit ADC digitizing between V_{R+} = +3.4v and V_{R-} = 0v?

$$\textbf{Precision} = \frac{3.4 - 0}{2^{12}} = 830\ \mu V$$

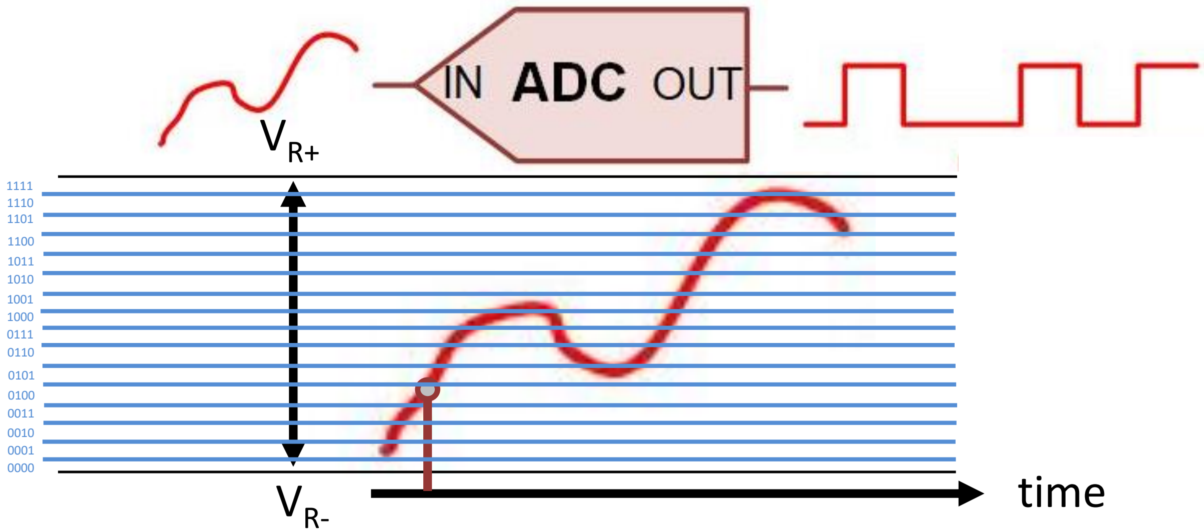
15.1 ANALOG TO DIGITAL CONVERTERS



- The **original analog value** is found by multiplying the digital conversion result (N_{ADC}) with the resolution.

$$\boxed{V_{analog} = N_{ADC} \cdot \text{Resolution}}$$

15.1 ANALOG TO DIGITAL CONVERTERS



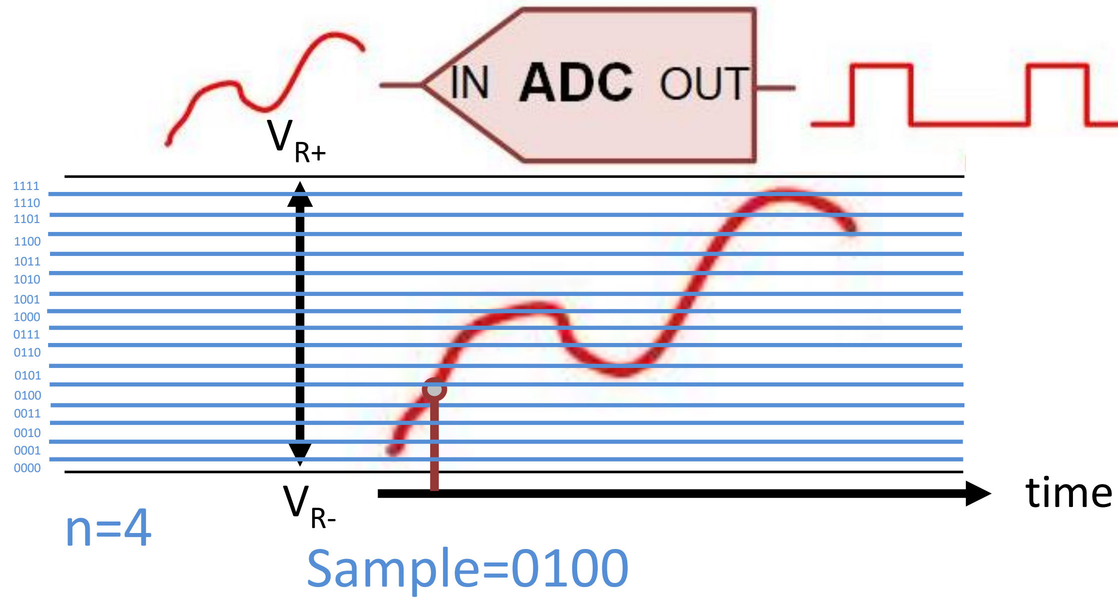
Example: For the ADC configuration in the above example, a conversion produced a result of 0x08A5. What is the analog value that was read?

$$V_{analog} = \text{0x08A5} \cdot 830\ \mu V$$

$$V_{analog} = 2213_{10} \cdot 830\ \mu V$$

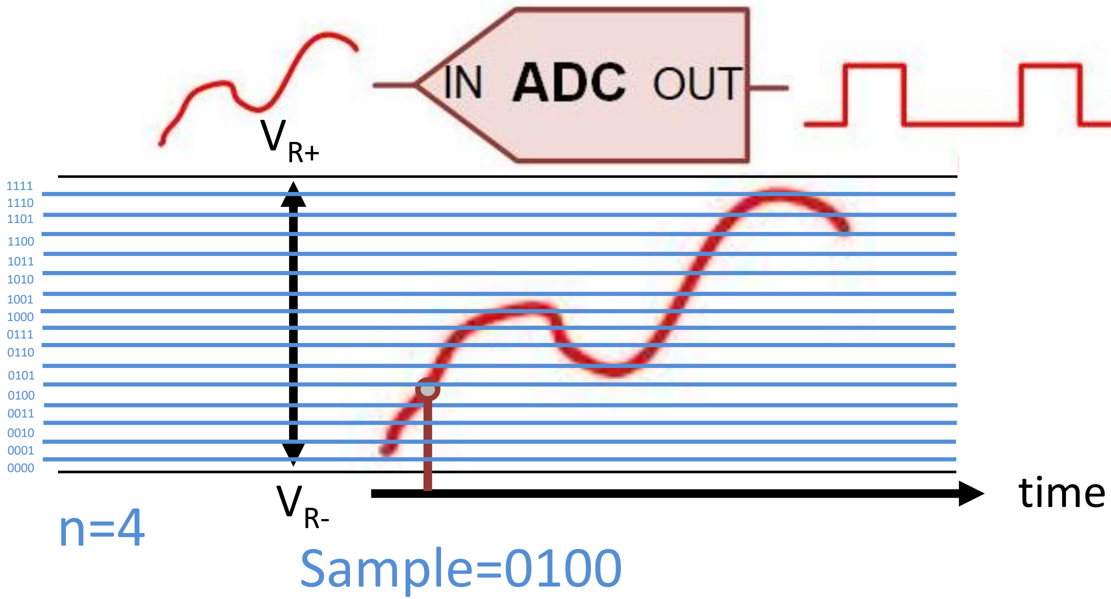
$$V_{analog} = +1.836962\ V$$

15.1 ANALOG TO DIGITAL CONVERTERS



- The **accuracy** is how close the digital output is to the input signal.

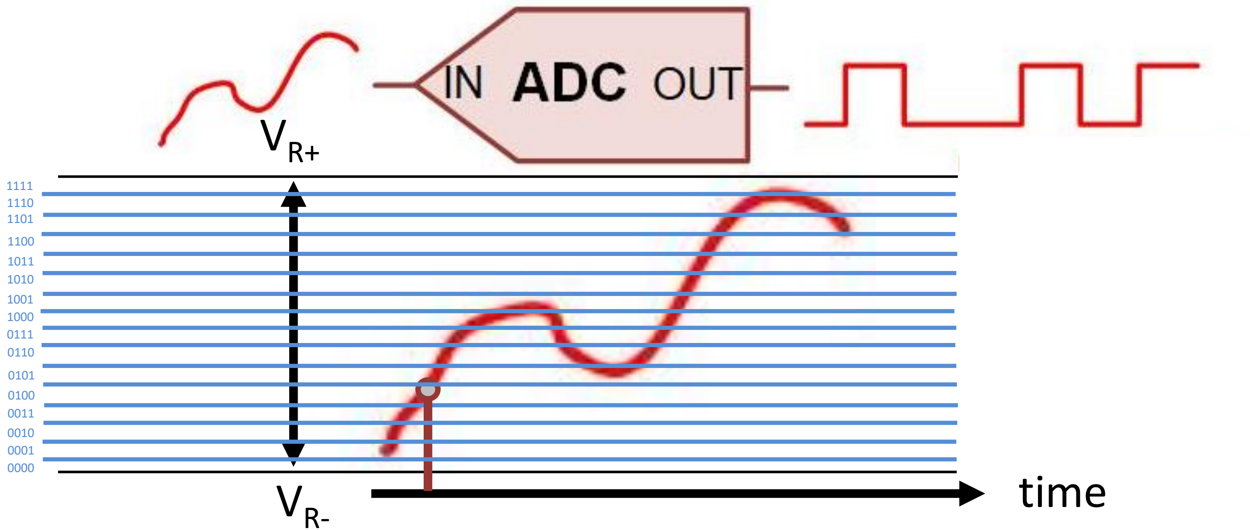
15.1 ANALOG TO DIGITAL CONVERTERS



- The **accuracy** is how close the digital output is to the input signal.
- By design, an ADC will only ever be able to get within **+/- ½ LSB** of the original analog value.

Accuracy = +/- ½ LSB

15.1 ANALOG TO DIGITAL CONVERTERS

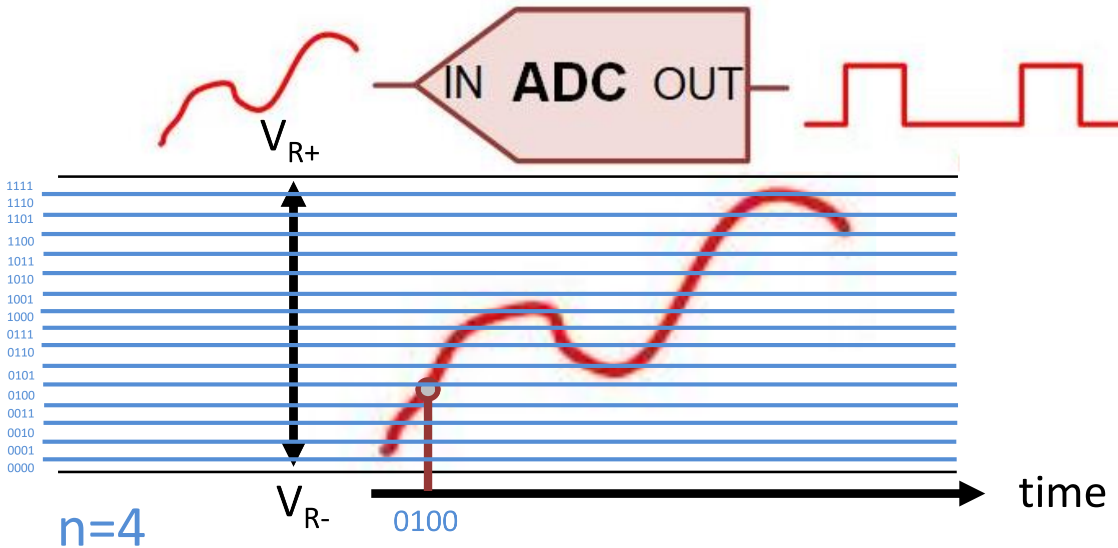


Example: For the V_{analog} voltage read in the above example, what is the accuracy of the result?

$$V_{analog} = +1.836962 \text{ +/- } 415\ \mu V$$

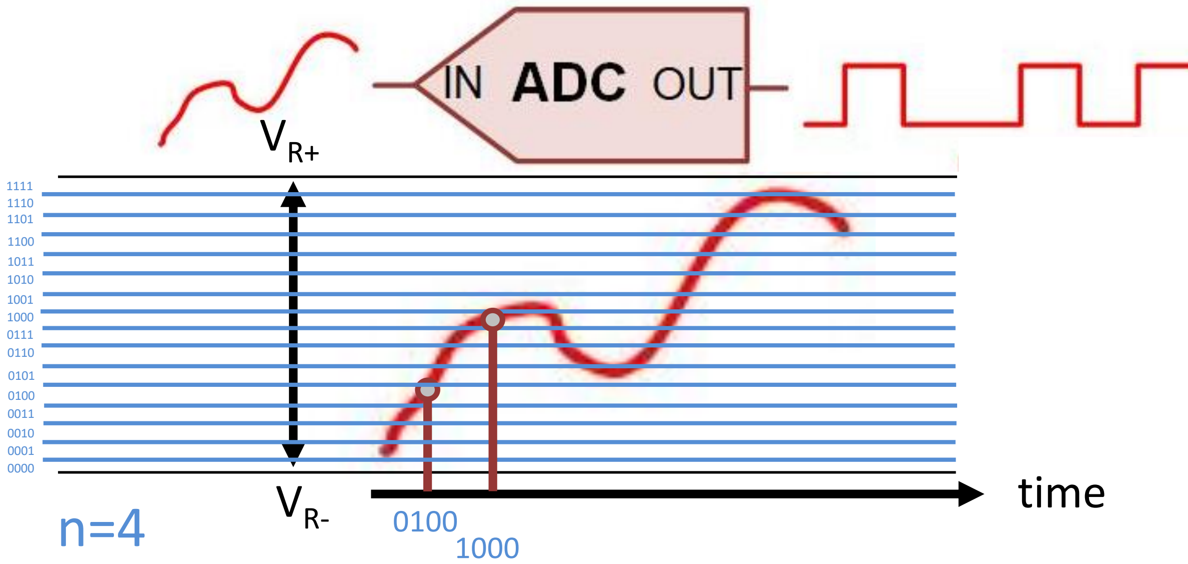
This means that the result 0x08A5 could be any voltage between **+1.836548** and **+1.837378**.

15.1 ANALOG TO DIGITAL CONVERTERS



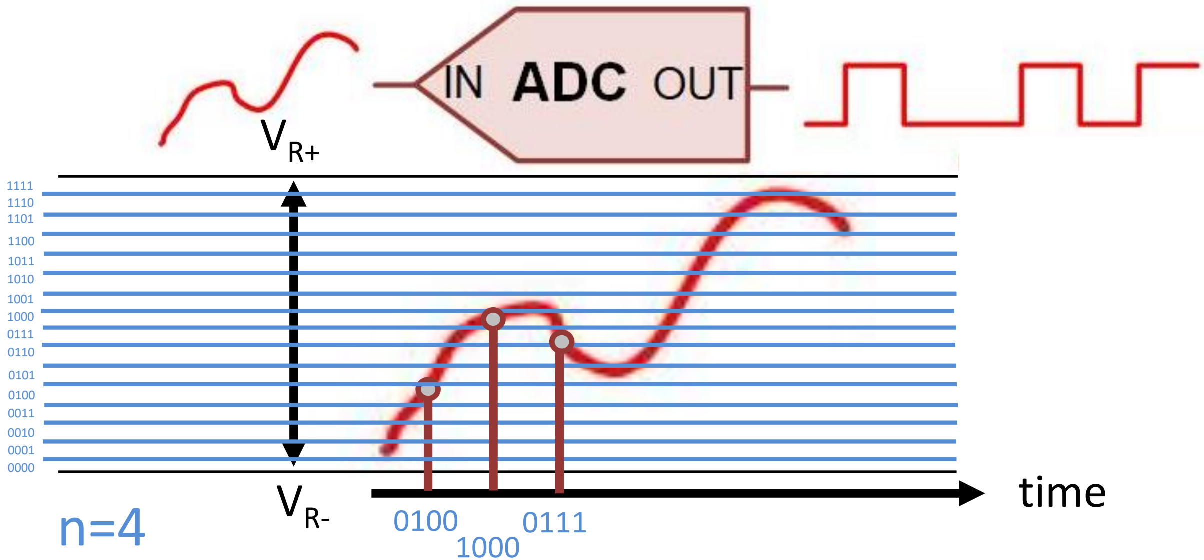
- The voltage can be sampled over time to create a list of digital values.

15.1 ANALOG TO DIGITAL CONVERTERS



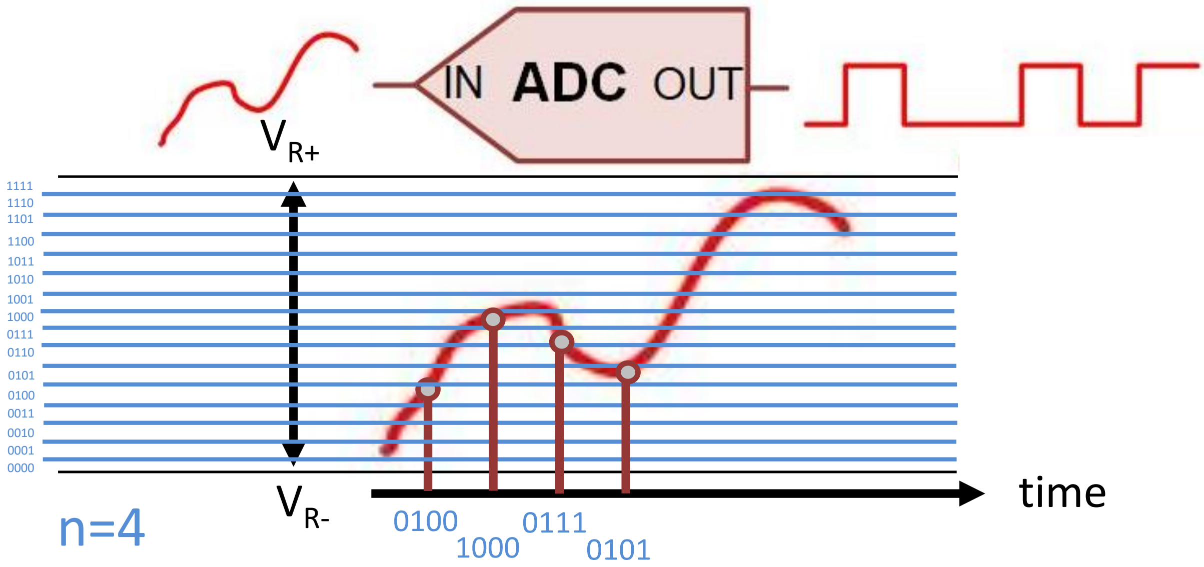
- The voltage can be sampled over time to create a list of digital values.

15.1 ANALOG TO DIGITAL CONVERTERS



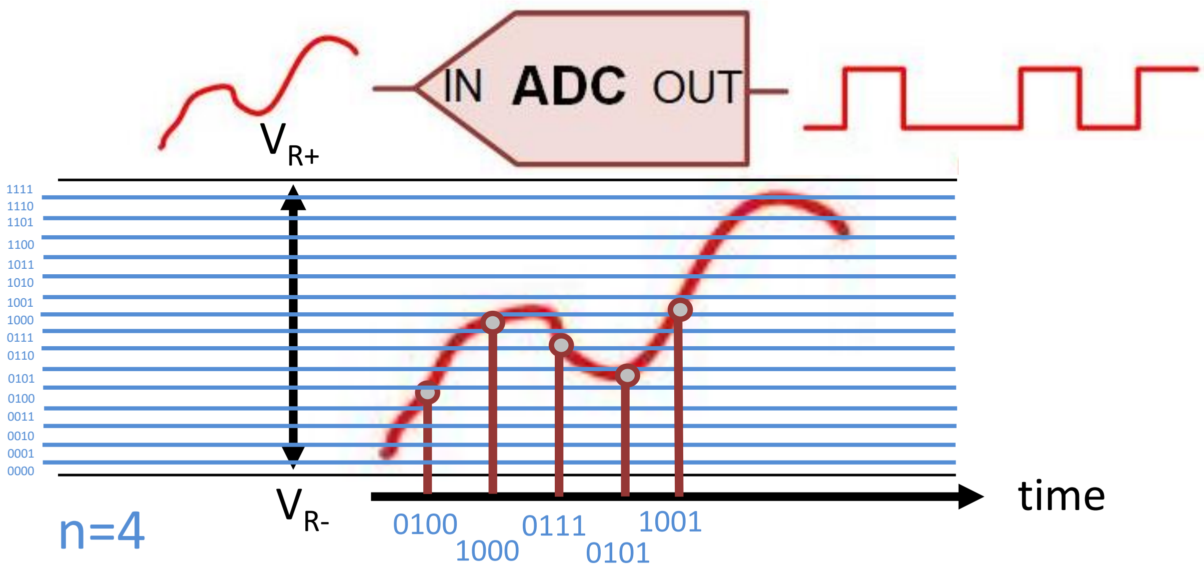
- The voltage can be sampled over time to create a list of digital values.

15.1 ANALOG TO DIGITAL CONVERTERS



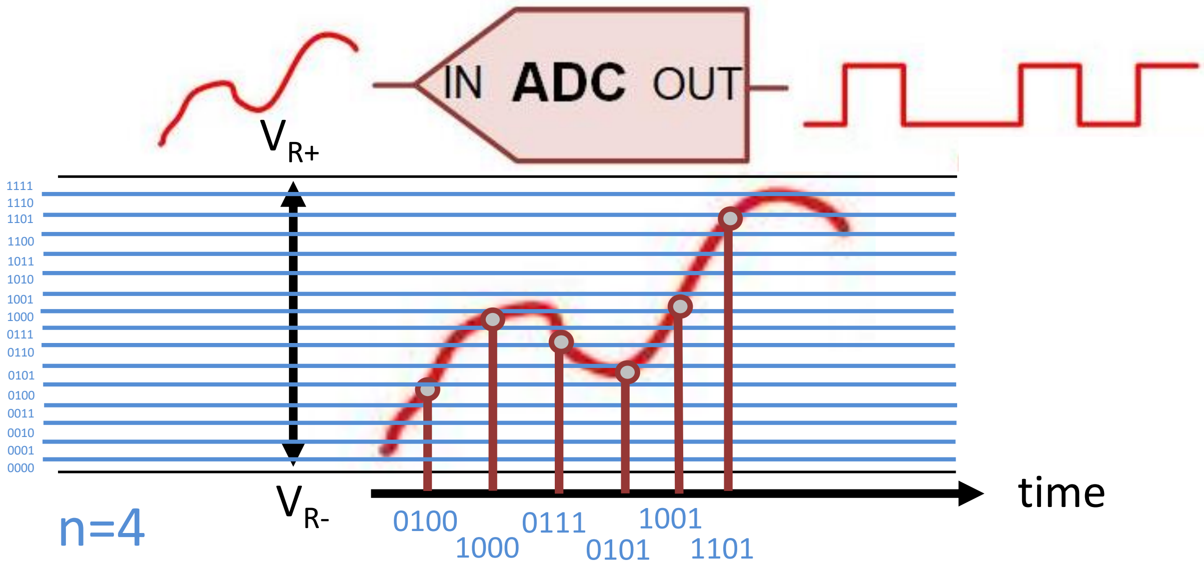
- The voltage can be sampled over time to create a list of digital values.

15.1 ANALOG TO DIGITAL CONVERTERS



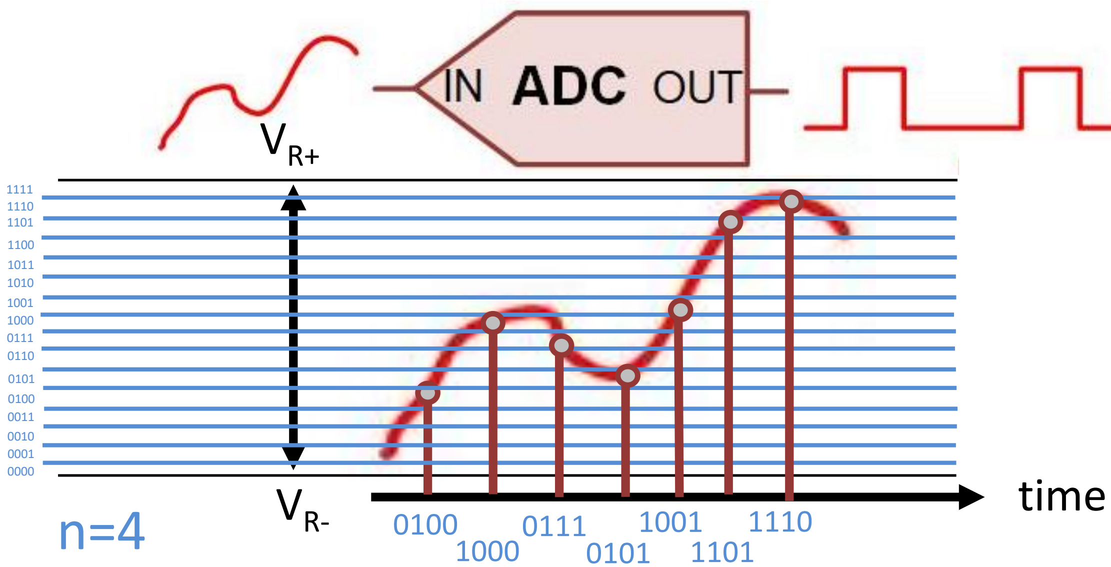
- The voltage can be sampled over time to create a list of digital values.

15.1 ANALOG TO DIGITAL CONVERTERS



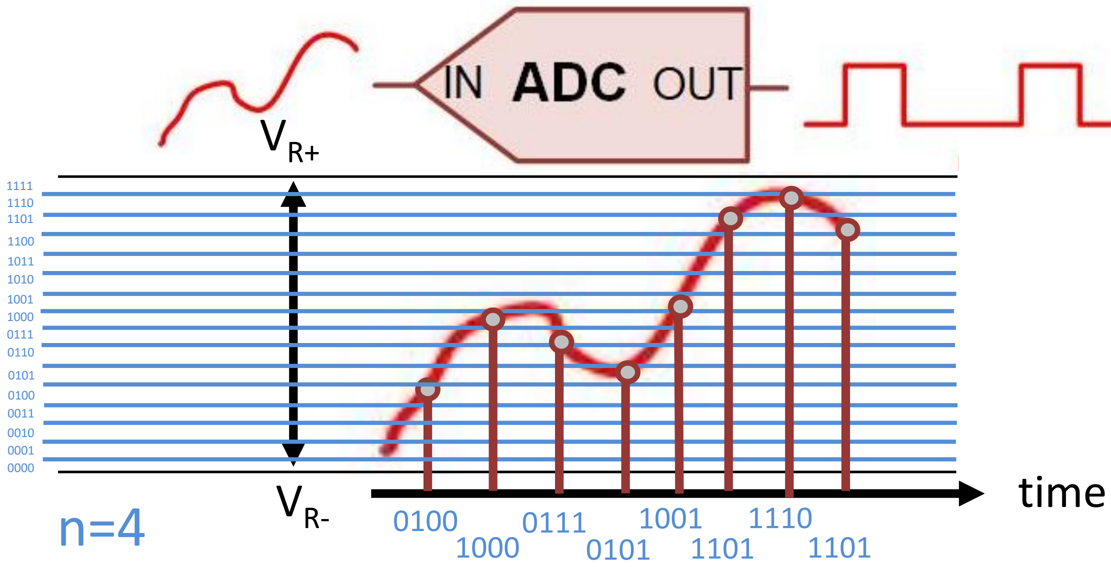
- The voltage can be sampled over time to create a list of digital values.

15.1 ANALOG TO DIGITAL CONVERTERS



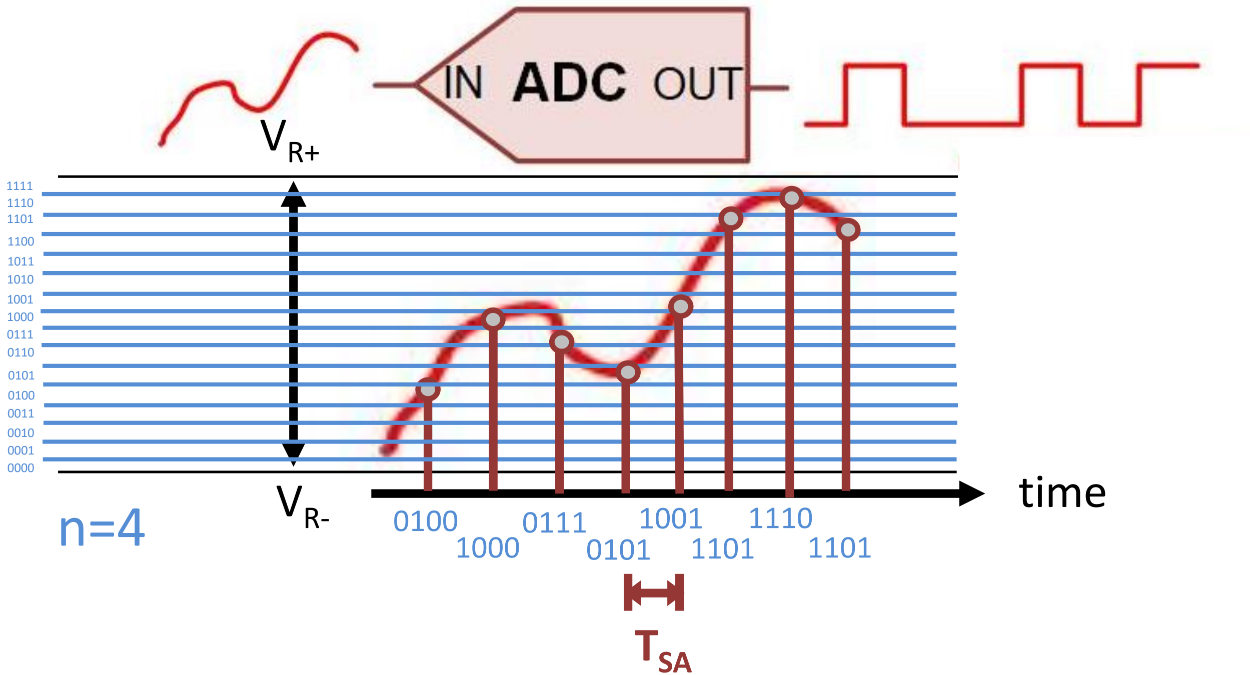
- The voltage can be sampled over time to create a list of digital values.

15.1 ANALOG TO DIGITAL CONVERTERS



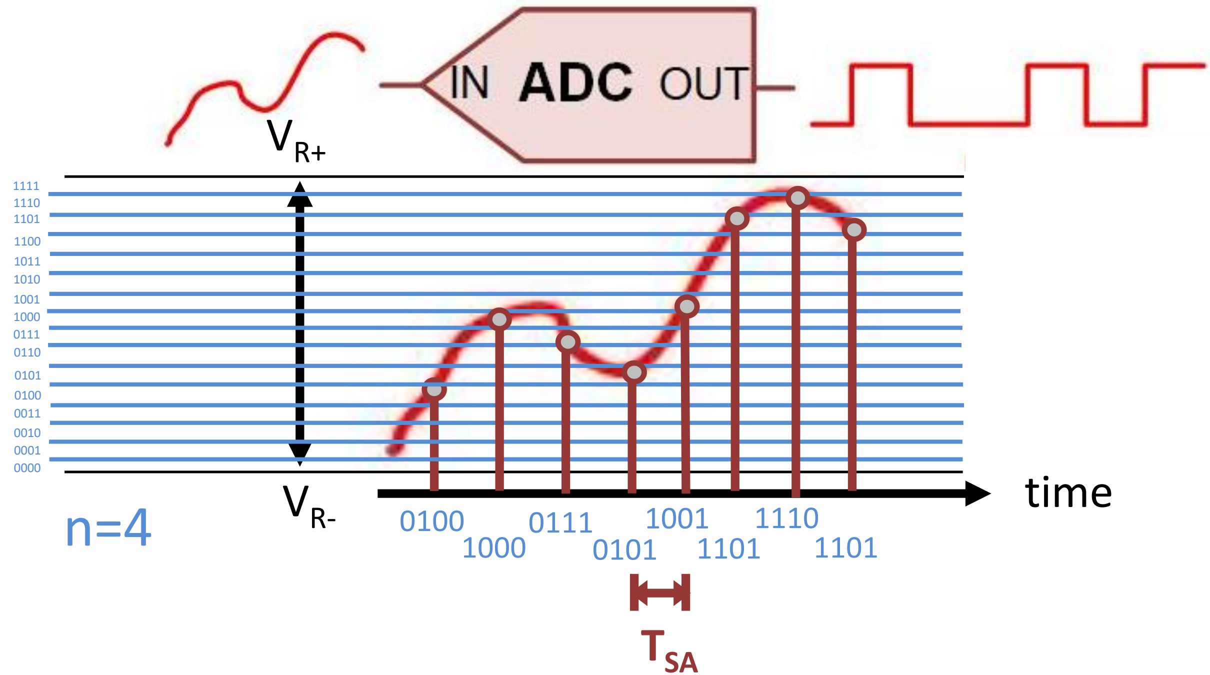
- The voltage can be sampled over time to create a list of digital values.

15.1 Analog to Digital Converters



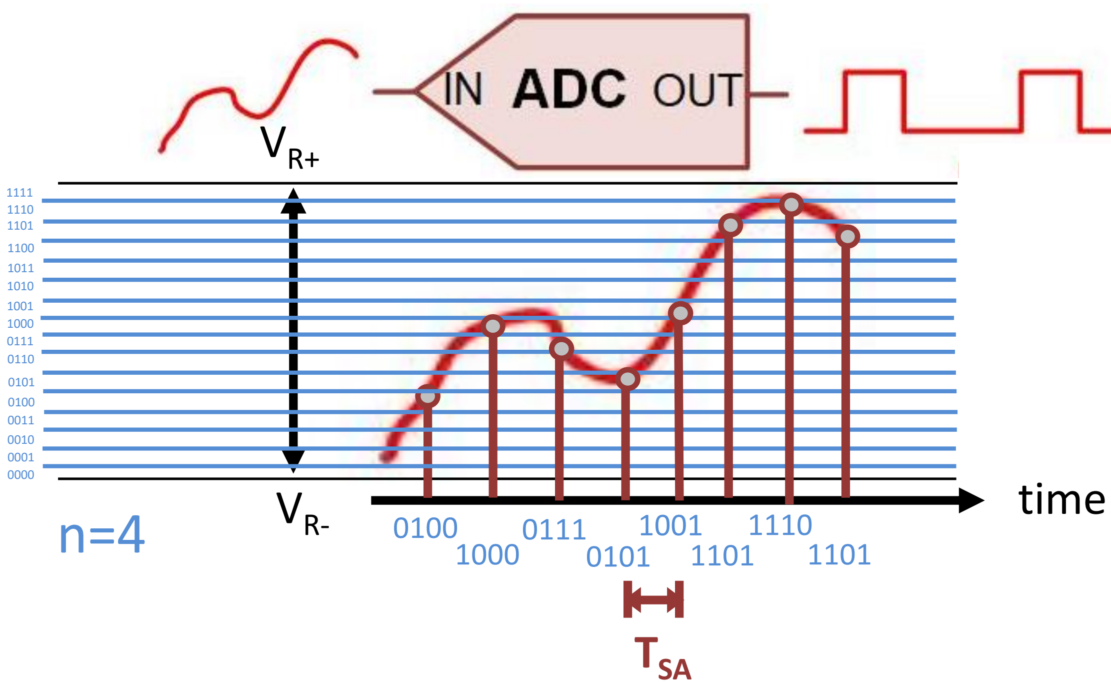
- The sample period (T_{SA}) is the time between samples.

15.1 ANALOG TO DIGITAL CONVERTERS



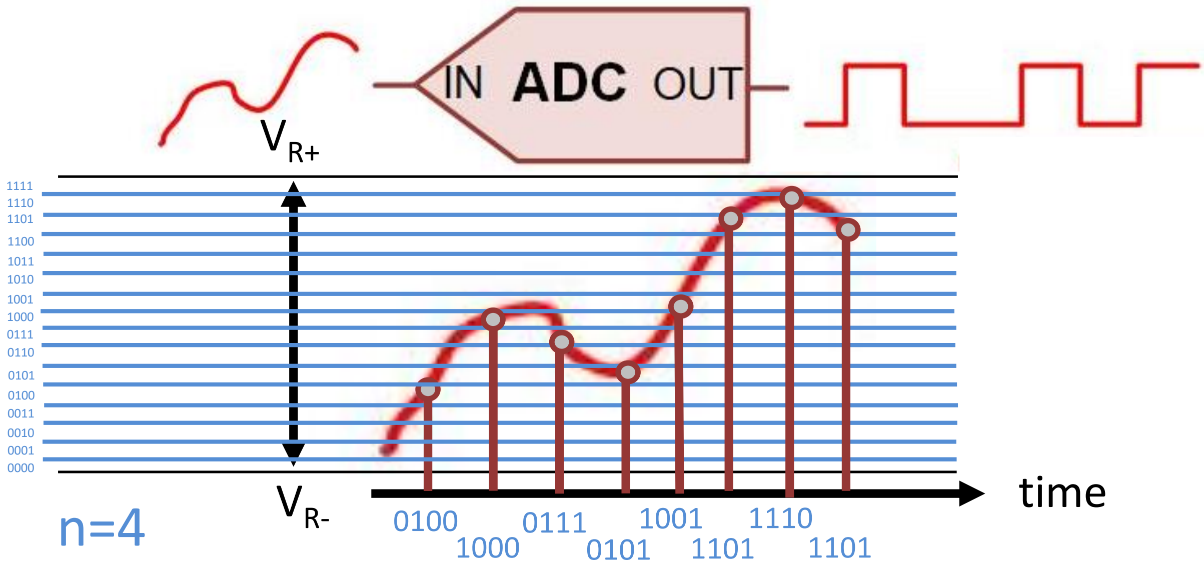
- The *sampling rate* is the frequency of sampling: $\mathbf{f_{SA} = 1/T_{SA}}$

15.1 ANALOG TO DIGITAL CONVERTERS



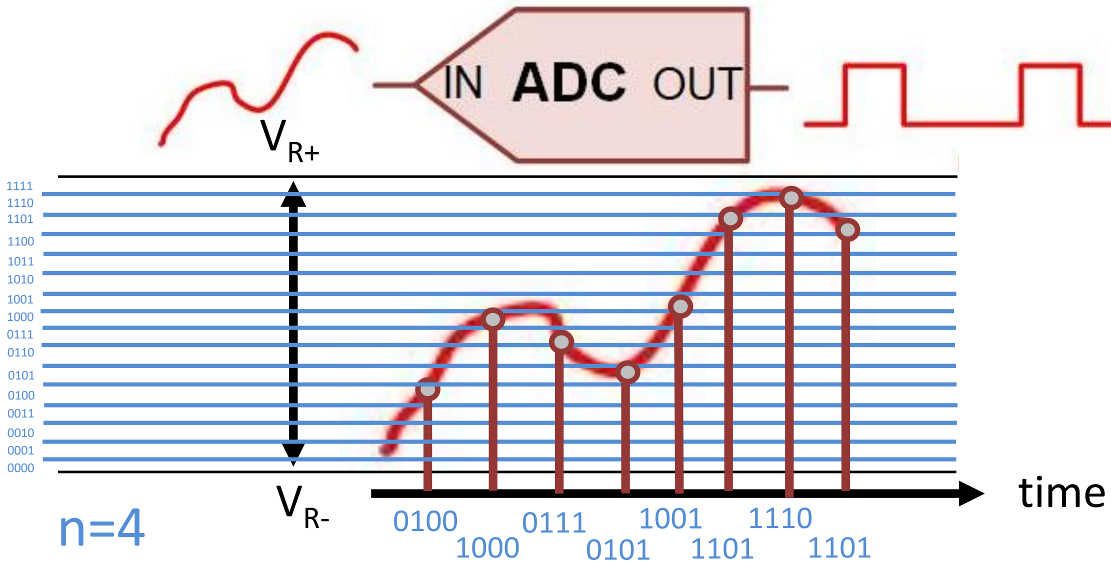
- The *sampling rate* is the frequency of sampling: $\mathbf{f_{SA}= 1/T_{SA}}$
- This has units of *samples-per-second* (i.e., *ksps, Msps*).

15.1 ANALOG TO DIGITAL CONVERTERS



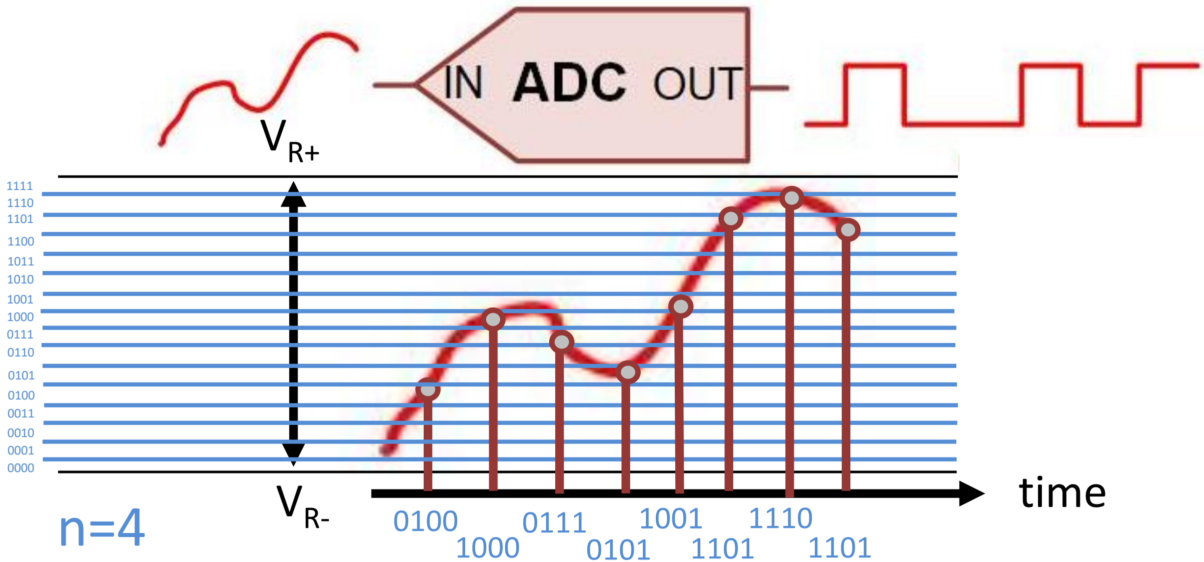
- If you sample fast enough, you can reconstruct the original signal.

15.1 ANALOG TO DIGITAL CONVERTERS



- If you sample fast enough, you can reconstruct the original signal.
- ***Nyquist-Shannon Sampling Theorem*** states you need to sample at least twice as fast as the frequency of the incoming signal to accurately reconstruct the original waveform.

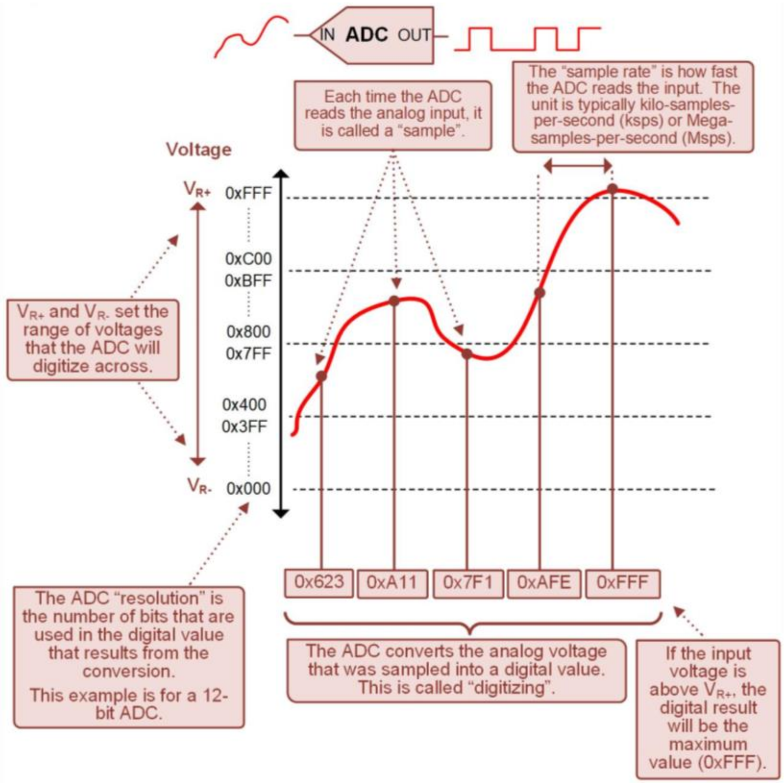
15.1 ANALOG TO DIGITAL CONVERTERS



- The MSP430FR2355 can sample up to 200 ksps.

15.1 ANALOG TO DIGITAL CONVERTERS

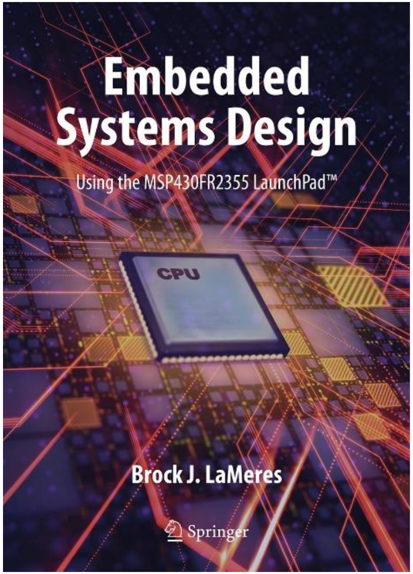
12-bit Example



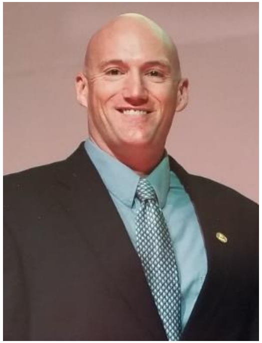
Embedded Systems Design

Chapter 15: Analog to Digital Converters

15.1 Analog to Digital Converters - Overview



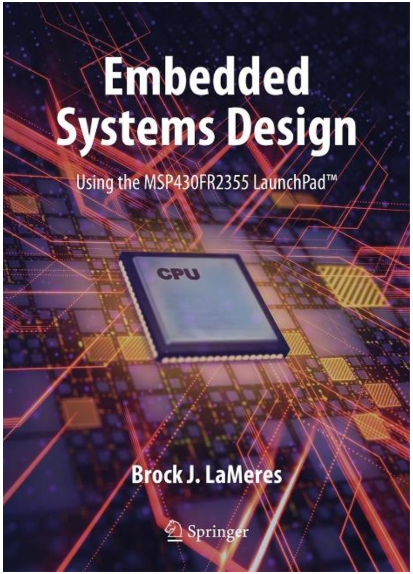
www.youtube.com/c/DigitalLogicProgramming_LaMeres

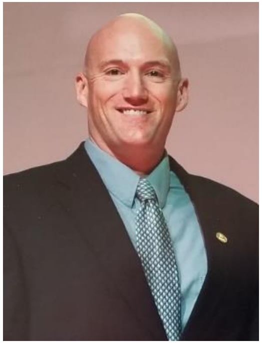


Brock J. LaMeres, Ph.D.

Chapter 15: Analog to Digital Converters

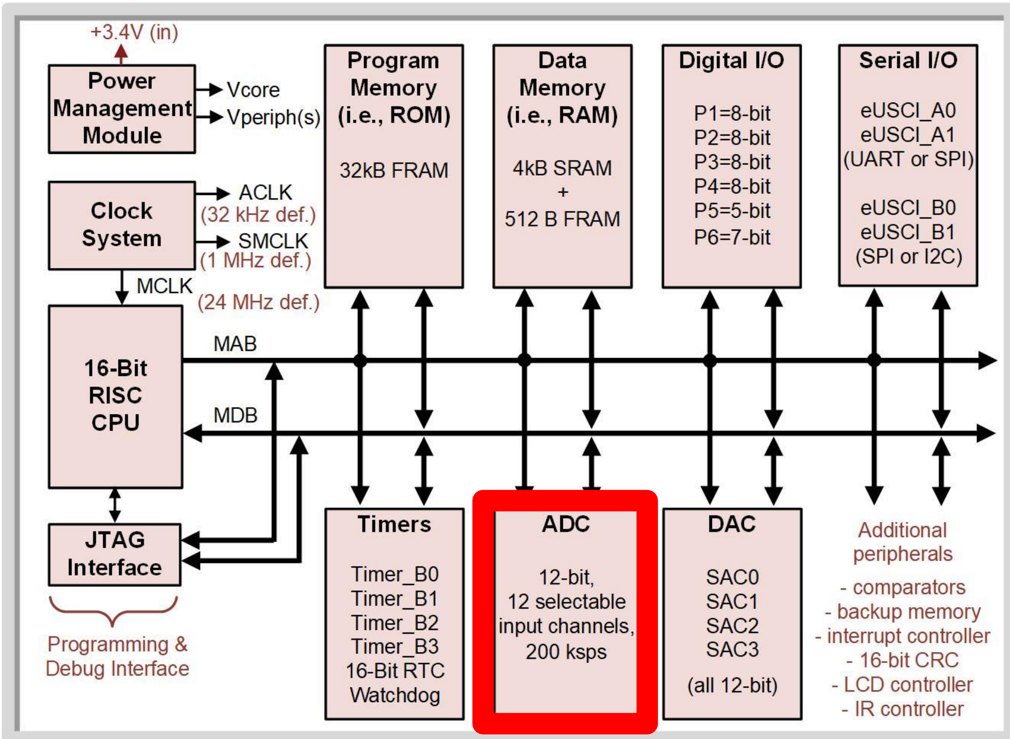
15.2 ADC Operation on the MSP430FR2355 - Configuration



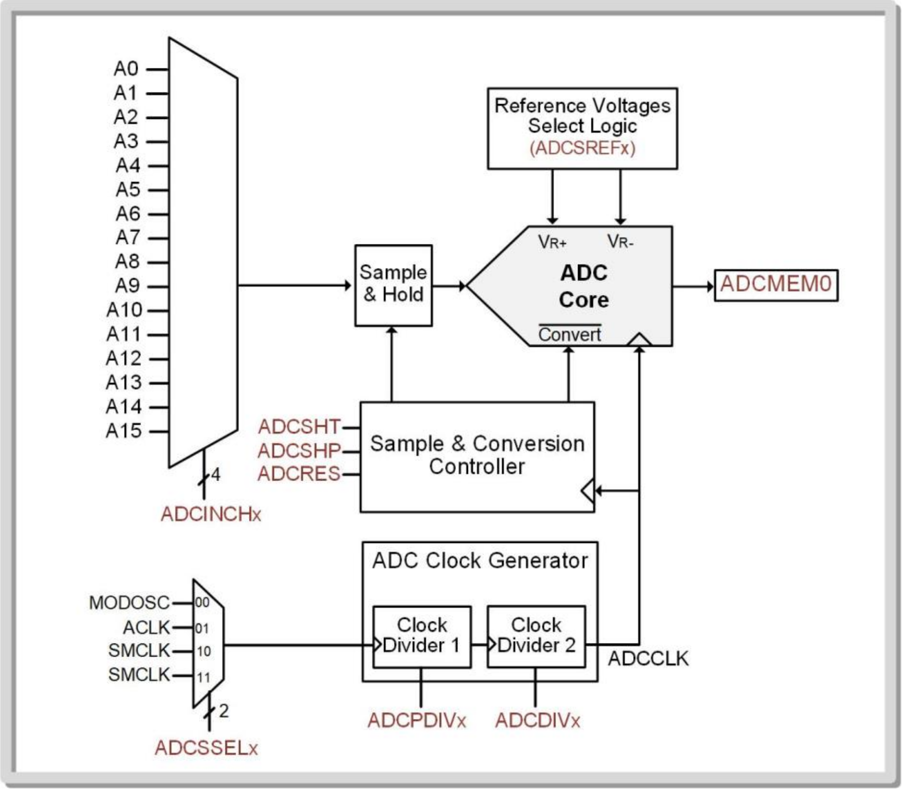


Brock J. LaMeres, Ph.D.

15.2 ADC OPERATION ON THE MSP430FR2355

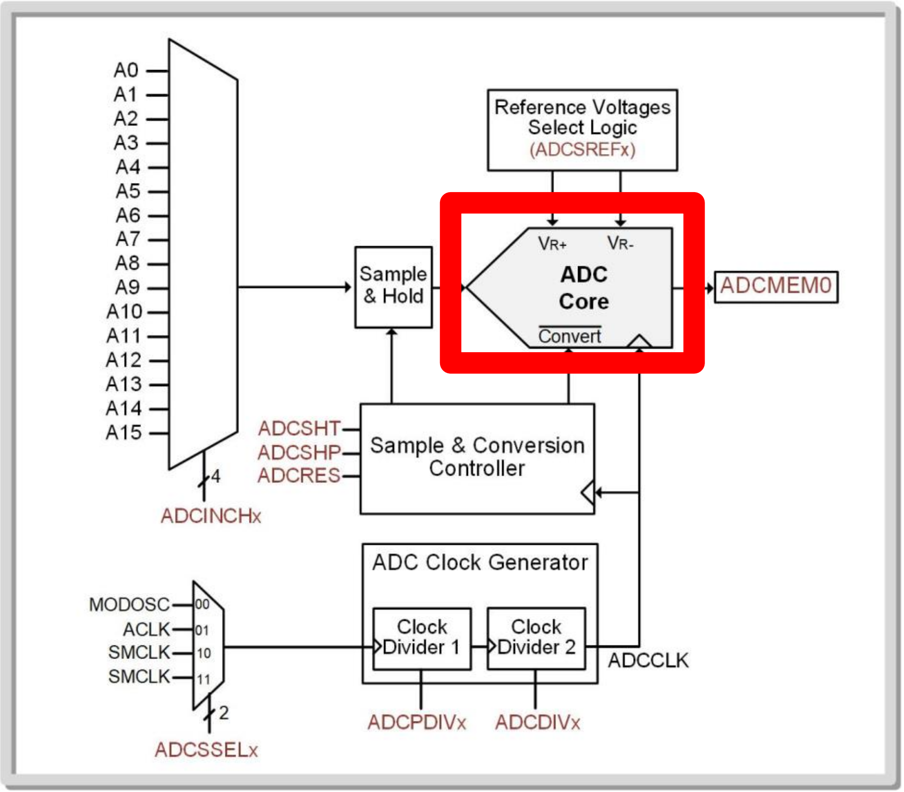


15.2 ADC Operation on the MSP430FR2355



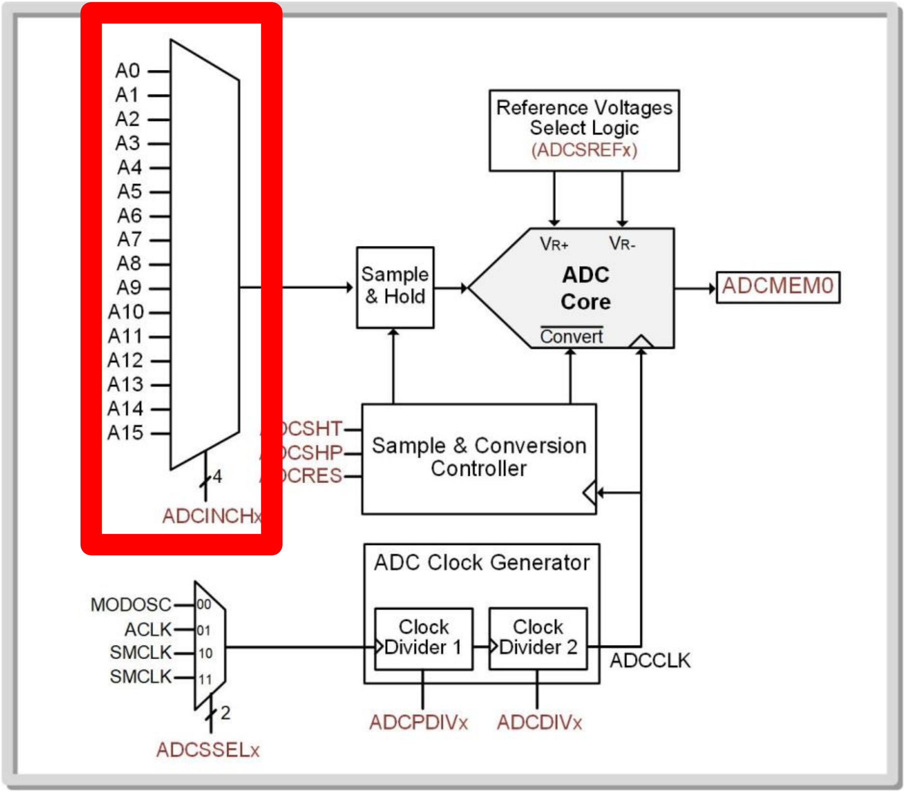
15.2 ADC OPERATION ON THE MSP430FR2355

- The MSP430 contains an ADC core with selectable resolution (8-bit, 10-bit, 12-bit).



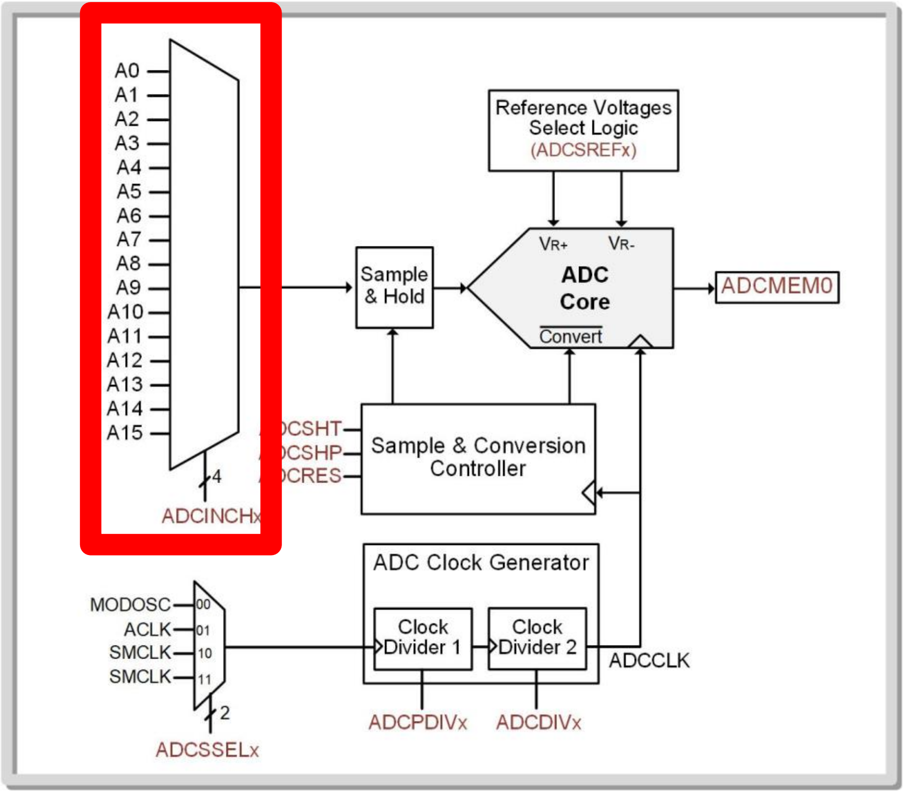
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC can be driven with 1 of 16 inputs that are selected using an analog multiplexer that sits in front of the ADC core.



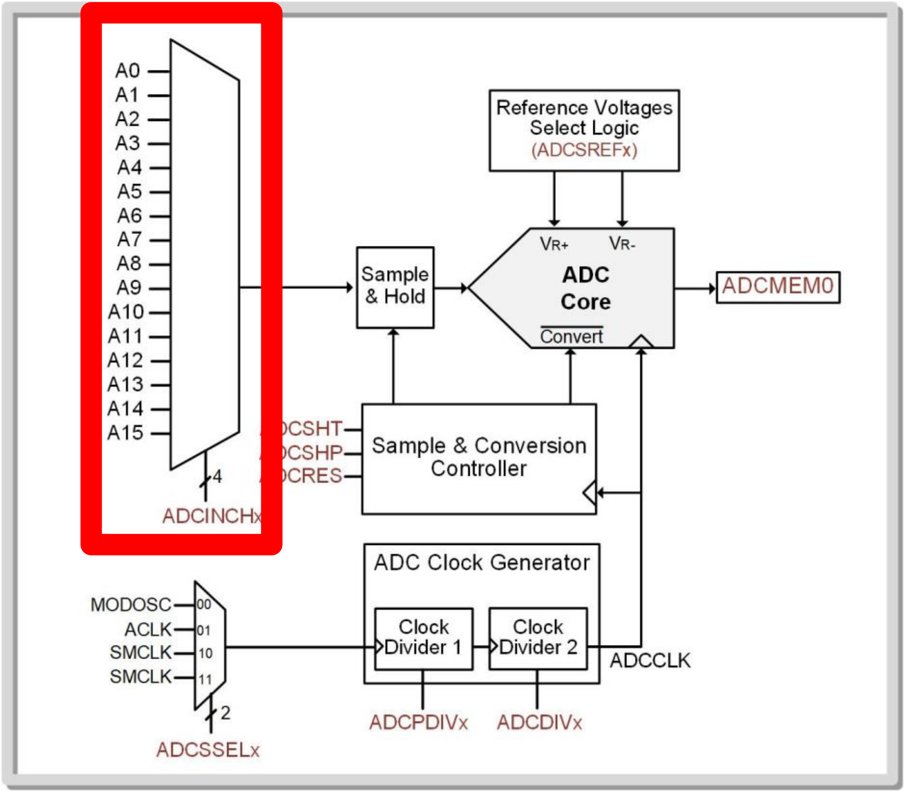
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC can be driven with 1 of 16 inputs that are selected using an analog multiplexer that sits in front of the ADC core.
- A0 → A11 are connected to pins that share with ports (use Port Function Select Registers = 11).



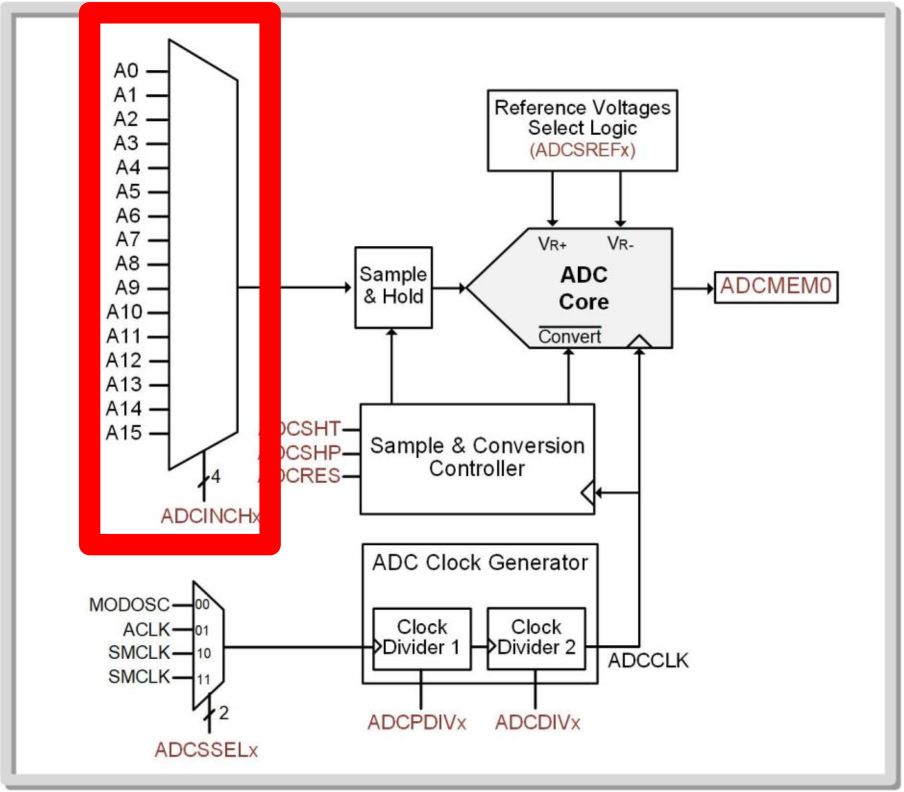
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC can be driven with 1 of 16 inputs that are selected using an analog multiplexer that sits in front of the ADC core.
- A0 → A11 are connected to pins that share with ports (use Port Function Select Registers = 11).
- A12 is an on-chip temperature sensor



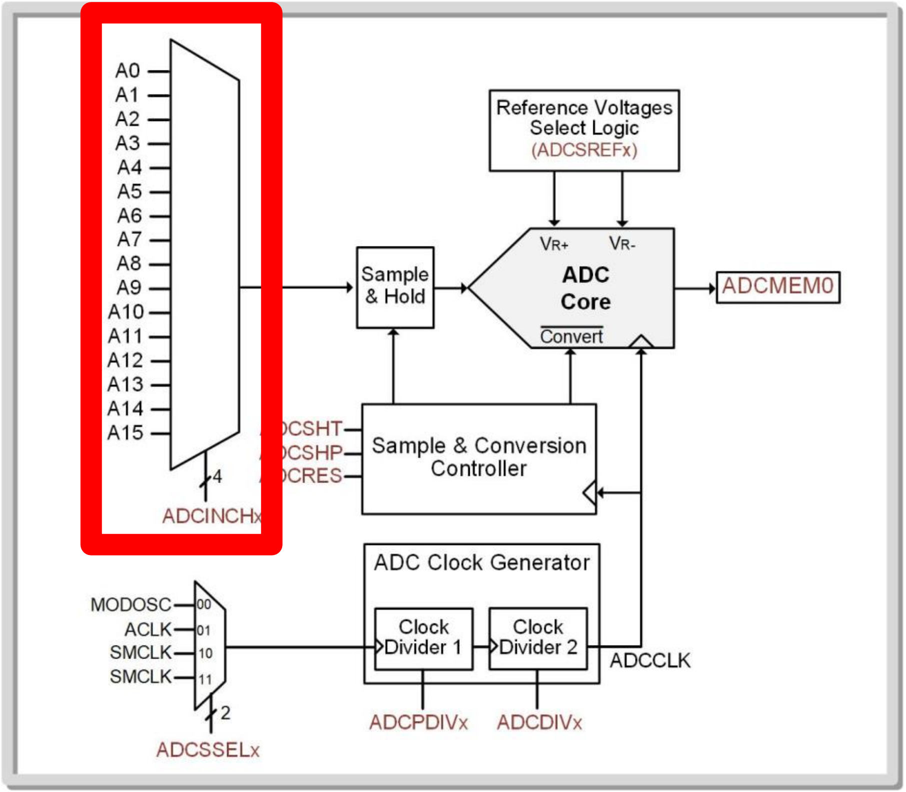
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC can be driven with 1 of 16 inputs that are selected using an analog multiplexer that sits in front of the ADC core.
- A0 → A11 are connected to pins that share with ports (use Port Function Select Registers = 11).
- A12 is an on-chip temperature sensor
- A13 is the internal voltage reference.



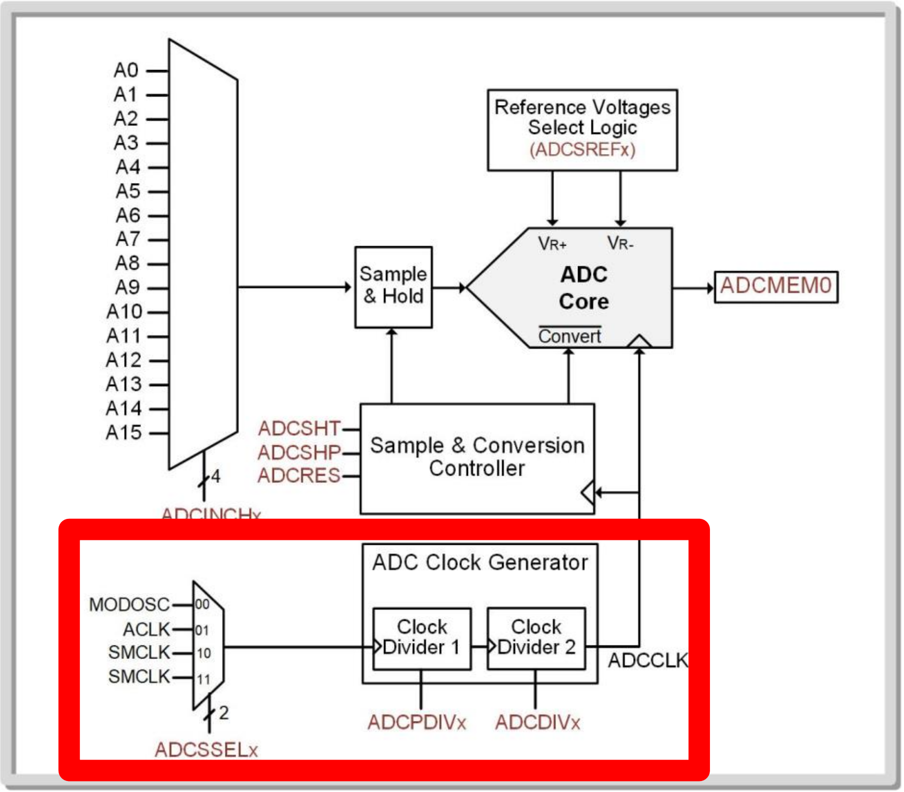
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC can be driven with 1 of 16 inputs that are selected using an analog multiplexer that sits in front of the ADC core.
- A0 → A11 are connected to pins that share with ports (use Port Function Select Registers = 11).
- A12 is an on-chip temperature sensor
- A13 is the internal voltage reference.
- A14 and A15 are VSS and VCC.



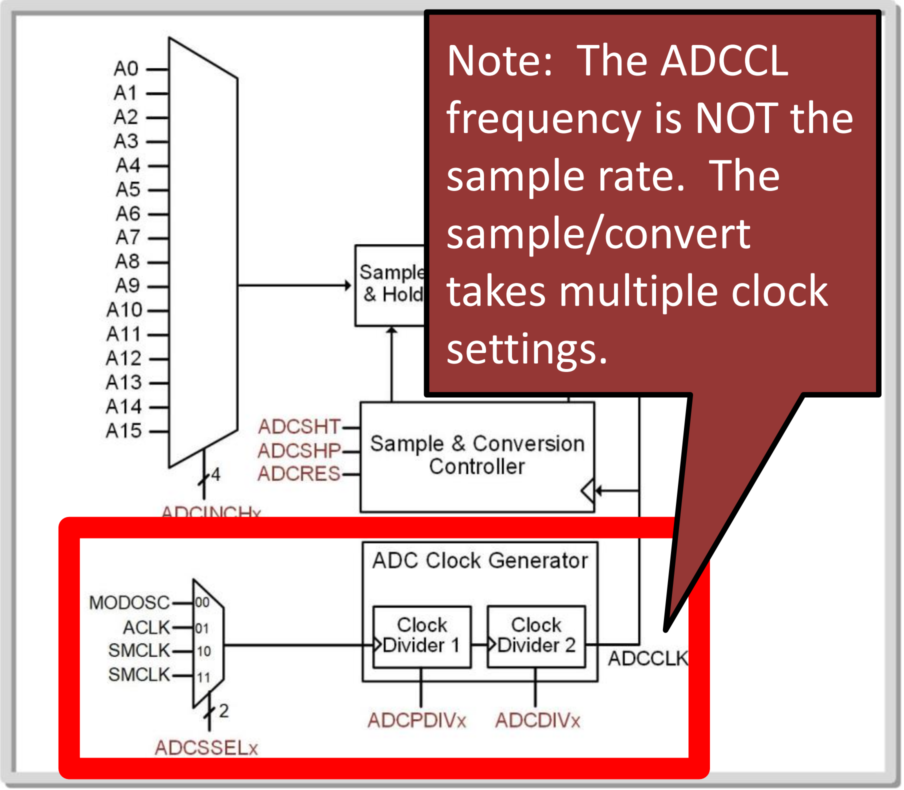
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC clock source is selectable with two stages of programmable dividers/prescalers.



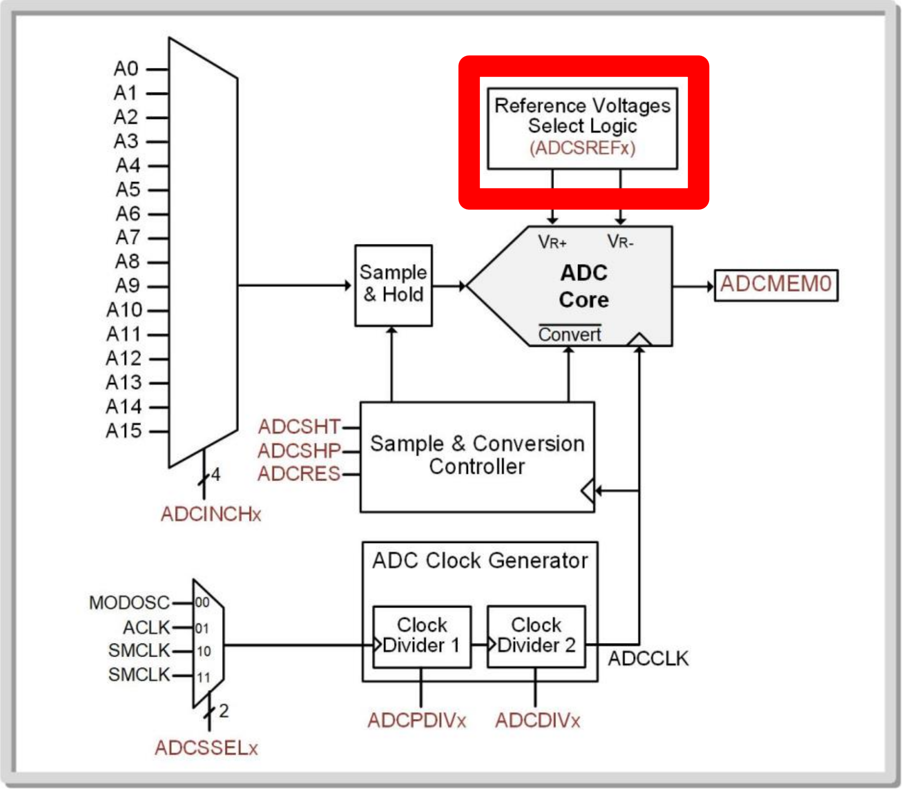
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC clock source is selectable with two stages of programmable dividers/prescalers.



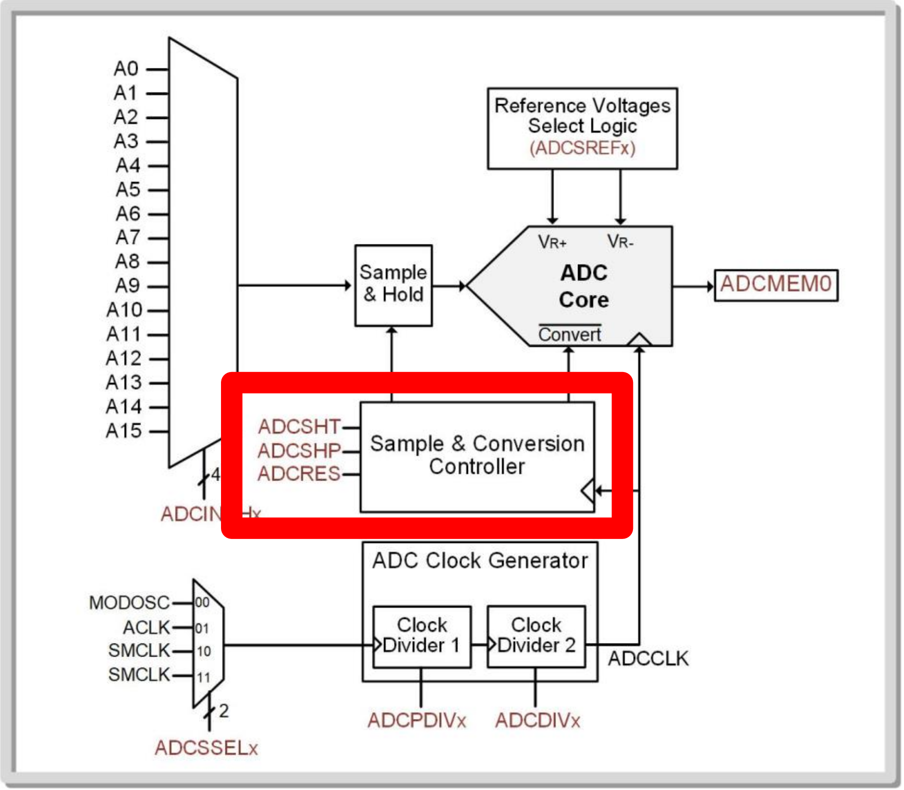
15.2 ADC OPERATION ON THE MSP430FR2355

- The voltage range of the ADC is also programmable with options of using the power supply (VCC) and GND (VSS), input pins, or a variety of internally generated voltages.



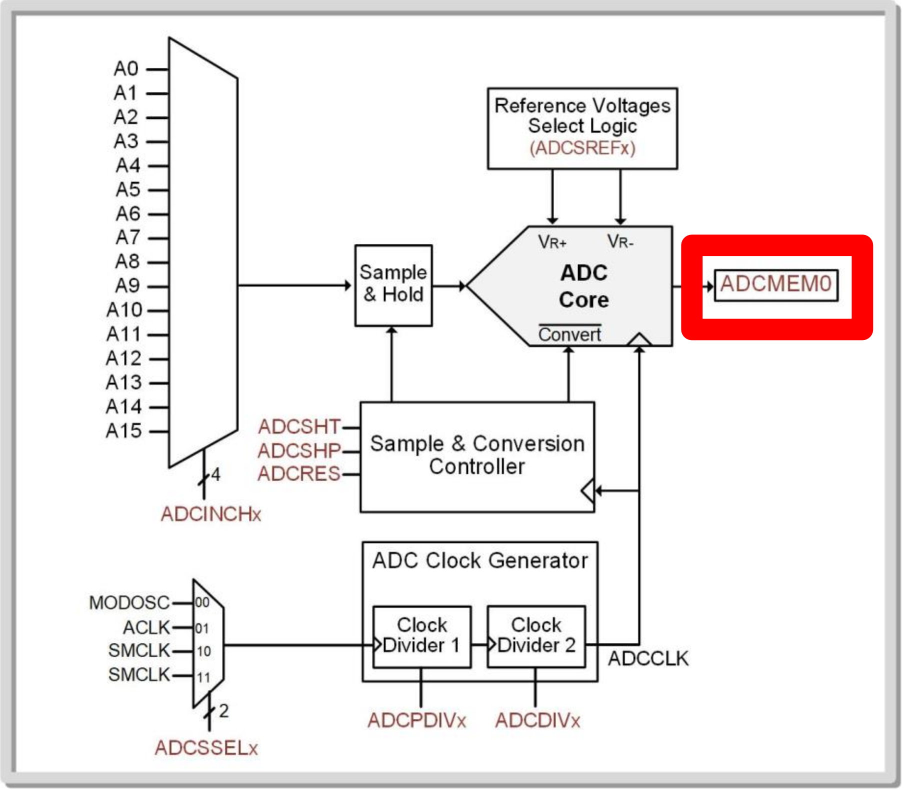
15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC peripheral also has a large number of programmable options that dictate its behavior in addition to 6 interrupts that can be triggered upon certain conditions.



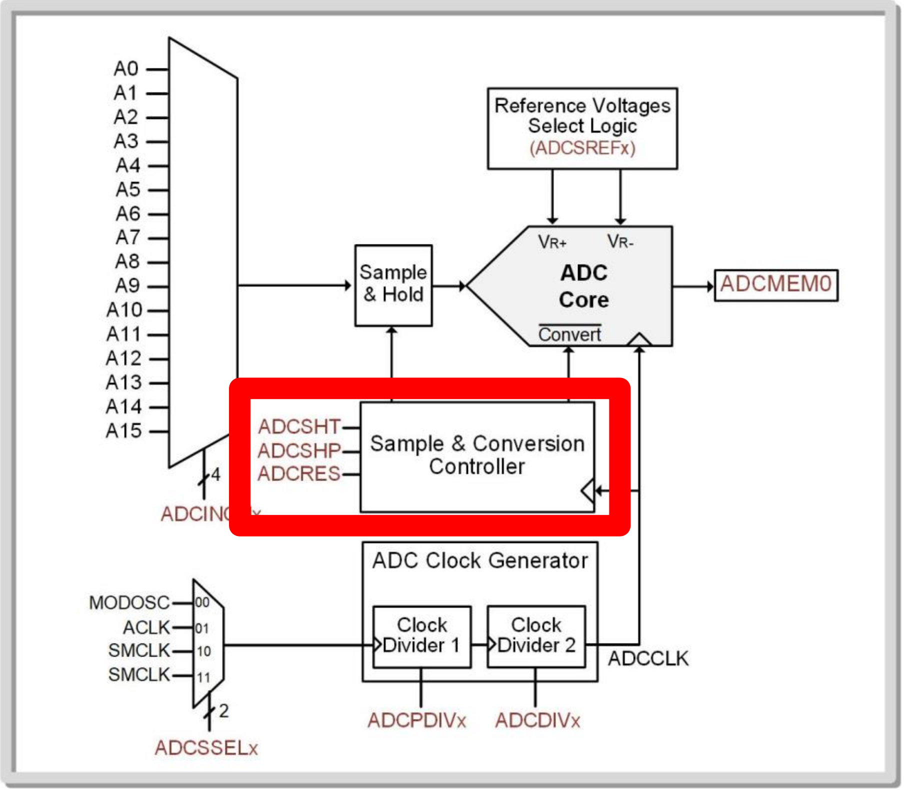
15.2 ADC OPERATION ON THE MSP430FR2355

- The basic use model for the ADC peripheral is:
  - Configure peripheral using a set of registers.
  - The conversation is started by the user (or by the successful completion of a prior conversation).
  - The result of the conversation is ready from the ADC's conversion memory register.



15.2 ADC OPERATION ON THE MSP430FR2355

- Flags can be used to track the status of the conversion and trigger interrupts to react to various states of the conversion.



15.2 ADC OPERATION ON THE MSP430FR2355

- ADC Control 0 (ADCCTL0) Register
- ADC Control 1 (ADCCTL1) Register
- ADC Control 2 (ADCCTL2) Register
- ADC Memory Control (ADCMCTL0) Register
- ADC Conversion Memory (ADCMEM0) Register
- ADC Interrupt Enable (ADCIE) Register
- ADC Interrupt Flag (ADCIFG) Register
- ADC Interrupt Vector (ADCIV) Register
- ADC Window Comparator Low Threshold (ADCLO) Register
- ADC Window Comparator High Threshold (ADCHI) Register

Image Courtesy of https://neodem.wp.horizon.ac.uk/

15.2 ADC Operation on the MSP430FR2355

ADCCTL0 contains:

- settings for the number of ADCCLK cycles to use during the conversion (ADCSHTx)
- how the ADC is triggered (ADCMSC)
- turning the ADC on (ADCON)
- enabling the conversion (ADCENC)
- Starting a conversion (ADCSC).

ADC Control Register 0 (ADCCTL0)

Bit	Field	Description
15:12	Reserved	-
11:8	ADCSHTx	ADC Sample-and-Hold Time. This defines the number of ADCCLK cycles in the sampling period for the ADC. 0000 = 4 ADCCLK Cycles 0001 = 8 ADCCLK Cycles (default) 0010 = 16 ADCCLK Cycles 0011 = 32 ADCCLK Cycles 0100 = 64 ADCCLK Cycles 0101 = 96 ADCCLK Cycles 0110 = 128 ADCCLK Cycles 0111 = 192 ADCCLK Cycles 1000 = 256 ADCCLK Cycles 1001 = 384 ADCCLK Cycles 1010 = 512 ADCCLK Cycles 1011 = 768 ADCCLK Cycles 1100 = 1024 ADCCLK Cycles 1101 = 1024 ADCCLK Cycles 1110 = 1024 ADCCLK Cycles 1111 = 1024 ADCCLK Cycles
7	ADCMSC	ADC Multiple Sample-and-Conversion. Valid for only sequence or repeated modes. 0 = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1 = The first rising edge of the SHI signal triggers the sampler timer, but subsequence sample-and-conversions happen automatically.
6:5	Reserved	-
4	ADCON	ADC On. (0 = ADC On; 1 = ADC Off)
3:2	Reserved	-
1	ADCENC	ADC Enable Conversion. (0 = ADC Disabled; 1 = ADC Enabled)
0	ADCSC	ADC Start Conversion. (0 = No Start; 1 =Start Sample-and-Convert)

15.2 ADC OPERATION ON THE MSP430FR2355

ADCCTL0 contains:

- settings for the number of ADCCLK cycles to use during the conversion (ADCSHTx)
- how the ADC is triggered (ADCMSC)
- turning the ADC on (ADCON)
- enabling the conversion (ADCENC)
- Starting a conversion (ADCSC).

ADC Control Register 0 (ADCCTL0)

Bit	Field	Description
15:12	Reserved	-
11:8	ADCSHTx	A … LK cycles in th …
7	ADCMSC	A … or repeated modes. … ing timer requires a rising edge of the SHI signal … er each sample-and-convert. … rst rising edge of the SHI signal triggers the sampler timer, … t subsequence sample-and-conversions happen automatically.
6:5	Reserved	-
4	ADCON	A … (0 = ADC On; 1 = ADC Off)
3:2	Reserved	
1	ADCENC	… DC Enable Conversion. (0 = ADC Disabled; 1 = ADC Enabled)
0	ADCSC	ADC Start Conversion. (0 = No Start; 1 =Start Sample-and-Convert)

The ADCENC and ADCSC bits are used to start a conversion by asserting them simultaneously.

15.2 ADC OPERATION ON THE MSP430FR2355

ADCCTL0

We will use the following for the basic ADC setup

ADC Control Register 0 (ADCCTL0)

Bit	Field	Description
15:12	Reserved	-
11:8	ADCSHTx	ADC Sample-and-Hold Time. This defines the number of ADCCLK cycles in the sampling period for the ADC. 0000 = 4 ADCCLK Cycles 0001 = 8 ADCCLK Cycles (default) 0010 = 16 ADCCLK Cycles 0011 = 32 ADCCLK Cycles 0100 = 64 ADCCLK Cycles 0101 = 96 ADCCLK Cycles 0110 = 128 ADCCLK Cycles 0111 = 192 ADCCLK Cycles 1000 = 256 ADCCLK Cycles 1001 = 384 ADCCLK Cycles 1010 = 512 ADCCLK Cycles 1011 = 768 ADCCLK Cycles 1100 = 1024 ADCCLK Cycles 1101 = 1024 ADCCLK Cycles 1110 = 1024 ADCCLK Cycles 1111 = 1024 ADCCLK Cycles
7	ADCMSC	ADC Multiple Sample-and-Conversion. Valid for only sequence or repeated modes. 0 = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1 = The first rising edge of the SHI signal triggers the sampler timer, but subsequence sample-and-conversions happen automatically.
6:5	Reserved	-
4	ADCON	ADC On. (0 = ADC On; 1 = ADC Off)
3:2	Reserved	-
1	ADCENC	ADC Enable Conversion. (0 = ADC Disabled; 1 = ADC Enabled)
0	ADCSC	ADC Start Conversion. (0 = No Start; 1 =Start Sample-and-Convert)

15.2 ADC OPERATION ON THE MSP430FR2355

ADCCTL0

We will use the following for the basic ADC setup

We'll assert these to start a conversion.

ADC Control Register 0 (ADCCTL0)

Bit	Field	Description
15:12	Reserved	-
11:8	ADCSHTx	ADC Sample-and-Hold Time. This defines the number of ADCCLK cycles in the sampling period for the ADC. 0000 = 4 ADCCLK Cycles 0001 = 8 ADCCLK Cycles (default) 0010 = 16 ADCCLK Cycles 0011 = 32 ADCCLK Cycles 0100 = 64 ADCCLK Cycles 0101 = 96 ADCCLK Cycles 0110 = 128 ADCCLK Cycles 0111 = 192 ADCCLK Cycles 1000 = 256 ADCCLK Cycles 1001 = 384 ADCCLK Cycles 1010 = 512 ADCCLK Cycles 1011 = 768 ADCCLK Cycles 1100 = 1024 ADCCLK Cycles 1101 = 1024 ADCCLK Cycles 1110 = 1024 ADCCLK Cycles 1111 = 1024 ADCCLK Cycles
7	ADCMSC	ADC Multiple Sample-and-Conversion. Valid for only sequence or repeated modes. 0 = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1 = The first rising edge of the SHI signal triggers the sampler timer, but subsequence sample-and-conversions happen automatically.
6:5	Reserved	-
4	ADCON	ADC On. (0 = ADC On; 1 = ADC Off)
3:2	Reserved	-
1	ADCENC	ADC Enable Conversion. (0 = ADC Disabled; 1 = ADC Enabled)
0	ADCSC	ADC Start Conversion. (0 = No Start; 1 =Start Sample-and-Convert)

15.2 ADC OPERATION ON THE MSP430FR2355

ADCCTL1 contains:

- settings for the source of the sample trigger (ADCSHSx)
- an option for running the sampler in pulse mode (ADCSHP)
- an input inversion option (ADCISSH)
- settings for the second clock divider stage (ADCDIVx)

ADC Control Register 1 (ADCCTL1)

Bit	Field	Description
15:12	Reserved	-
11-10	ADCSHSx	ADC Sample-and-Hold Source Select. 00 = ADCSC bit 01 = Timer Trigger 0 (see device-specific data sheet) 10 = Timer Trigger 1 (see device-specific data sheet) 11 = Timer Trigger 2 (see device-specific data sheet)
9	ADCSHP	ADC Sample-and-Hold Pulse-Mode Select. This selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. 0 = SAMPCON signal is sourced from the sample input signal. 1 = SAMPCON signal is sourced from the sampling timer.
8	ADCISSH	ADC Invert Signal Sample-and-Hold. 0 = The sample input is not inverted. 1 = The sample input signal is inverted.
7:5	ADCDIVx	ADC Clock Divider. 000 = Divide by 1 001 = Divide by 2 010 = Divide by 3 011 = Divide by 4 100 = Divide by 5 101 = Divide by 6 110 = Divide by 7 111 = Divide by 8
4:3	ADCSSELx	ADC Clock Source Select. 00 = MODCLK (internal high-speed oscillator) 01 = ACLK 10 = SMCLK 11 = SMCLK
2:1	ADCCONSEQx	ADC Conversion Sequence Mode Select. 00 = Single-channel, single-conversion 01 = Sequence-of-channels 10 = Repeat-single-channel 11 = Repeat-sequence-of-channels
0	ADCBUSY	ADC Busy. (0 = No Operation is Active; 1 = ADC is Active)

15.2 ADC Operation on the MSP430FR2355

ADCCTL1 contains:

- the ADC clock source (ADCSSELx)
- whether the conversion is to run once or multiple times upon a trigger (ADCCONSEQx)
- a status flag indicating the conversion is busy (ADCBSUSY).

ADC Control Register 1 (ADCCTL1)

Bit	Field	Description
15:12	Reserved	-
11-10	ADCSHSx	ADC Sample-and-Hold Source Select. 00 = ADCSC bit 01 = Timer Trigger 0 (see device-specific data sheet) 10 = Timer Trigger 1 (see device-specific data sheet) 11 = Timer Trigger 2 (see device-specific data sheet)
9	ADCSHP	ADC Sample-and-Hold Pulse-Mode Select. This selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. 0 = SAMPCON signal is sourced from the sample input signal. 1 = SAMPCON signal is sourced from the sampling timer.
8	ADCISSH	ADC Invert Signal Sample-and-Hold. 0 = The sample input is not inverted. 1 = The sample input signal is inverted.
7:5	ADCDIVx	ADC Clock Divider. 000 = Divide by 1 001 = Divide by 2 010 = Divide by 3 011 = Divide by 4 100 = Divide by 5 101 = Divide by 6 110 = Divide by 7 111 = Divide by 8
4:3	ADCSSELx	ADC Clock Source Select. 00 = MODCLK (internal high-speed oscillator) 01 = ACLK 10 = SMCLK 11 = SMCLK
2:1	ADCCONSEQx	ADC Conversion Sequence Mode Select. 00 = Single-channel, single-conversion 01 = Sequence-of-channels 10 = Repeat-single-channel 11 = Repeat-sequence-of-channels
0	ADCBUSY	ADC Busy. (0 = No Operation is Active; 1 = ADC is Active)

15.2 ADC OPERATION ON THE MSP430FR2355

ADCCTL1

We will use the following for the basic ADC setup

ADC Control Register 1 (ADCCTL1)

Bit	Field	Description
15:12	Reserved	-
11-10	ADCSHSx	ADC Sample-and-Hold Source Select. 00 = ADCSC bit 01 = Timer Trigger 0 (see device-specific data sheet) 10 = Timer Trigger 1 (see device-specific data sheet) 11 = Timer Trigger 2 (see device-specific data sheet)
9	ADCSHP	ADC Sample-and-Hold Pulse-Mode Select. This selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. 0 = SAMPCON signal is sourced from the sample input signal. 1 = SAMPCON signal is sourced from the sampling timer.
8	ADCISSH	ADC Invert Signal Sample-and-Hold. 0 = The sample input is not inverted. 1 = The sample input signal is inverted.
7:5	ADCDIVx	ADC Clock Divider. 000 = Divide by 1 001 = Divide by 2 010 = Divide by 3 011 = Divide by 4 100 = Divide by 5 101 = Divide by 6 110 = Divide by 7 111 = Divide by 8
4:3	ADCSSELx	ADC Clock Source Select. 00 = MODCLK (internal high-speed oscillator) 01 = ACLK 10 = SMCLK 11 = SMCLK
2:1	ADCCONSEQx	ADC Conversion Sequence Mode Select. 00 = Single-channel, single-conversion 01 = Sequence-of-channels 10 = Repeat-single-channel 11 = Repeat-sequence-of-channels
0	ADCBUSY	ADC Busy. (0 = No Operation is Active; 1 = ADC is Active)

15.2 ADC Operation on the MSP430FR2355

ADCCTL2 contains:

- settings for the first clock divider stage (ADCPDIVx)
- the resolution of the ADC (ADCRES)
- the data format of the result (ADCDF)
- the range for the anticipated sample rate (ADCSR).

ADC Control Register 2 (ADCCTL2)

Bit	Field	Description
15:10	Reserved	-
9:8	ADCPDIVx	ADC Pre-divider. 00 = Divide by 1 01 = Divide by 4 10 = Divide by 64 11 = Reserved
7:6	Reserved	-
5:4	ADCRES	ADC Resolution. 00 = 8-bit (10 clock cycle conversion time) 01 = 10-bit (10 clock cycle conversion time) (default) 10 = 12-bit (10 clock cycle conversion time) 11 = Reserved
3	ADCDF	ADC Data Read-Back Format. 0 = Unsigned binary 1 = Signed binary (2s complement)
2	ADCSR	ADC Sampling Rate. 0 = ADC buffer supports up to ~200 ksps 1 = ADC buffer supports up to ~50 ksps
1:0	Reserved	-

15.2 ADC OPERATION ON THE MSP430FR2355

ADCCTL2 contains:

- settings for the first clock divider stage (ADCPDIVx)
- the resolution of the ADC (ADCRES)
- the data format of the result (ADCDF)
- the range for the anticipated sample rate (ADCSR).

ADC Control Register 2 (ADCCTL2)

Bit	Field	Description
15:10	Reserved	-
9:8	ADCPDIVx	ADC Pre-divider. 00 = Divide by 1 01 = Divide by 4 10 = Divide by 64 11 = Reserved
7:6	Reserved	-
5:4	ADCRES	ADC Resolution. 00 = 8-bit (10 clock cycle conversion time) 01 = 10-bit (10 clock cycle conversion time) (default) 10 = 12-bit (10 clock cycle conversion time) 11 = Reserved
3	ADCDF	Read-Back Format. 0 = Unsigned binary 1 = Signed binary (2s complement)
2		pling Rate. 0 = ADC buffer supports up to ~200 ksps 1 = ADC buffer supports up to ~50 ksps

Note that the default setting for ADCRES is 01 (10-bit).

If this setting is to be changed, the LSB of ADCRES needs to be cleared before the new settings are written.

15.2 ADC OPERATION ON THE MSP430FR2355

ADCCTL2

We will use the following for the basic ADC setup

ADC Control Register 2 (ADCCTL2)

Bit	Field	Description
15:10	Reserved	-
9:8	ADCPDIVx	ADC Pre-divider. 00 = Divide by 1 01 = Divide by 4 10 = Divide by 64 11 = Reserved
7:6	Reserved	-
5:4	ADCRES	ADC Resolution. 00 = 8-bit (10 clock cycle conversion time) 01 = 10-bit (10 clock cycle conversion time) (default) 10 = 12-bit (10 clock cycle conversion time) 11 = Reserved
3	ADCDF	ADC Data Read-Back Format. 0 = Unsigned binary 1 = Signed binary (2s complement)
2	ADCSR	ADC Sampling Rate. 0 = ADC buffer supports up to ~200 ksps 1 = ADC buffer supports up to ~50 ksps
1:0	Reserved	-

15.2 ADC OPERATION ON THE MSP430FR2355

ADCMCTL0 contains:

- settings for the reference voltage selection (ADCSREFx).
  - **"VREF"** – refers to the internal reference voltages that the MCU can produce.
  - **"VEREF+/-"** – refers to the external pins.
  - **"AVCC"** – refers to the MCU power supply (+3.4v).
- the ADC input channel that will be routed to the sample-and-hold stage (ADCINCHx).

ADC Conversion Memory Control (ADCMCTL0) Register

Bit	Field	Description
15:7	Reserved	-
6:4	ADCSREFx	ADC Reference Voltage Select. 000 = {V_{R+} = AVCC, V_{R-} = AVSS} 001 = {V_{R+} = VREF, V_{R-} = AVSS} 010 = {V_{R+} = VEREF+ buffered, V_{R-} = AVSS} 011 = {V_{R+} = VEREF+, V_{R-} = AVSS} 100 = {V_{R+} = AVCC, V_{R-} = VEREF-} 101 = {V_{R+} = VREF, V_{R-} = VEREF-} 110 = {V_{R+} = VEREF+ buffered, V_{R-} = VEREF-} 111 = {V_{R+} = VEREF+, V_{R-} = VEREF-}
3:0	ADCINCHx	ADC Input Channel Select. Connection 0000 = A0/VEREF+ (P1.0) 0001 = A1 (P1.1) 0010 = A2/VEREF- (P1.2) 0011 = A3 (P1.3) 0100 = A4 (P1.4) 0101 = A5 (P1.5) 0110 = A6 (P1.6) 0111 = A7 (P1.7) 1000 = A8 (P5.0) 1001 = A9 (P5.1) 1010 = A10 (P5.2) 1011 = A11 (P5.3) 1100 = A12 (on-chip temperature sensor) 1101 = A13 (internal reference voltage) 1110 = A14 (DVSS) 1111 = A15 (DVCC)

15.2 ADC OPERATION ON THE MSP430FR2355

ADCMCTL0 contains:

- settings for the reference voltage selection (ADCSREFx).
  - **"VREF"** – refers to the internal reference voltages that the MCU can produce.
  - **"VEREF+/-"** – refers to the external pins.
  - **"AVCC"** – refers to the MCU power supply (+3.4v).
- the ADC input channel that will be routed to the sample-and-hold stage (ADCINCHx).

ADC Conversion Memory Control (ADCMCTL0) Register

Bit	Field	Description
15:7	Reserved	-
6:4	ADCSREFx	ADC Reference Voltage Select. 000 = {VR+ = AVCC, VR- = AVSS} 001 = {VR+ = VREF, VR- = AVSS} 010 = {VR+ = VEREF+ buffered, VR- = AVSS} 011 = {VR+ = VEREF+, VR- = AVSS} 100 = {VR+ = AVCC, VR- = VEREF-} 101 = {VR+ = VREF, VR- = VEREF-} 110 = {VR+ = VEREF+ buffered, VR- = VEREF-} 111 = {VR+ = VEREF+, VR- = VEREF-}
3:0	ADCINCHx	ADC Input Channel Select. Connection 0000 = A0/VEREF+ (P1.0) 0001 = A1 (P1.1) 0010 = A2/VEREF- (P1.2) 0011 = A3 (P1.3) 0100 = A4 (P1.4) 0101 = A5 (P1.5) 0110 = A6 (P1.6) 0111 = A7 (P1.7) 1000 = A8 (P5.0) 1001 = A9 (P5.1) 1010 = A10 (P5.2) 1011 = A11 (P5.3) 1100 = A12 (on-chip temperature sensor) 1101 = A13 (internal reference voltage) 1110 = A14 (DVSS) 1111 = A15 (DVCC)

Note: the Port Function Select settings of 0b11 selects the Analog function.

15.2 ADC Operation on the MSP430FR2355

ADCMCTL0

We will use the following for the basic ADC setup

ADC Conversion Memory Control (ADCMCTL0) Register

Bit	Field	Description
15:7	Reserved	-
6:4	ADCSREFx	ADC Reference Voltage Select. 000 = {V_{R+} = AVCC, V_{R-} = AVSS} 001 = {V_{R+} = VREF, V_{R-} = AVSS} 010 = {V_{R+} = VEREF+ buffered, V_{R-} = AVSS} 011 = {V_{R+} = VEREF+, V_{R-} = AVSS} 100 = {V_{R+} = AVCC, V_{R-} = VEREF-} 101 = {V_{R+} = VREF, V_{R-} = VEREF-} 110 = {V_{R+} = VEREF+ buffered, V_{R-} = VEREF-} 111 = {V_{R+} = VEREF+, V_{R-} = VEREF-}
3:0	ADCINCHx	ADC Input Channel Select. Connection 0000 = A0/VEREF+ (P1.0) 0001 = A1 (P1.1) 0010 = A2/VEREF- (P1.2) 0011 = A3 (P1.3) 0100 = A4 (P1.4) 0101 = A5 (P1.5) 0110 = A6 (P1.6) 0111 = A7 (P1.7) 1000 = A8 (P5.0) 1001 = A9 (P5.1) 1010 = A10 (P5.2) 1011 = A11 (P5.3) 1100 = A12 (on-chip temperature sensor) 1101 = A13 (internal reference voltage) 1110 = A14 (DVSS) 1111 = A15 (DVCC)

15.2 ADC OPERATION ON THE MSP430FR2355

- The ADC peripheral contains six interrupt flags that can be used to monitor the status of the conversion.

ADC Interrupt Flag (ADCIFG) Register

p:	15	14	13	12	11	10	9	8
	Reserved							
Reset:	0	0	0	0	0	0	0	0

p:	7	6	5	4	3	2	1	0
	Reserved		ADCTOVIFG	ADCOVIFG	ADCHIIFG	ADCLOIFG	ADCINIFG	ADCIFG0
Reset:	0	0	0	0	0	0	0	0

Bit	Field	Description
15:6	Reserved	-
5	ADCTOVIFG	ADCTOVIFG is set when an ADC conversion is triggered before the current conversion has completed. (0=No IRQ Pending; 1=IRQ Pending)
4	ADCOVIFG	ADCOVIFG is set when ADCMEM0 is written before the last conversion result has been read. (0=No IRQ Pending; 1=IRQ Pending)
3	ADCHIIFG	ADCHIIFG is set when result is above the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
2	ADCLOIFG	ADCLOIFG is set when result is below the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
1	ADCINIFG	ADCINIFG is set when result is within the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
0	ADCIFG0	ADCIFG0 set when ADC conversion is complete. ADCIFG0 is cleared when ADCMEM0 is read. (0=No IRQ Pending; 1=IRQ Pending)

15.2 ADC Operation on the MSP430FR2355

- There is an interrupt flag that will trigger when a conversion is complete (ADCIFG0).

ADC Interrupt Flag (ADCIFG) Register

p:	15	14	13	12	11	10	9	8
	Reserved							
Reset:	0	0	0	0	0	0	0	0

p:	7	6	5	4	3	2	1	0
	Reserved		ADCTOVIFG	ADCOVIFG	ADCHIIFG	ADCLOIFG	ADCINIFG	ADCIFG0
Reset:	0	0	0	0	0	0	0	0

Bit	Field	Description
15:6	Reserved	-
5	ADCTOVIFG	ADCTOVIFG is set when an ADC conversion is triggered before the current conversion has completed. (0=No IRQ Pending; 1=IRQ Pending)
4	ADCOVIFG	ADCOVIFG is set when ADCMEM0 is written before the last conversion result has been read. (0=No IRQ Pending; 1=IRQ Pending)
3	ADCHIIFG	ADCHIIFG is set when result is above the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
2	ADCLOIFG	ADCLOIFG is set when result is below the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
1	ADCINIFG	ADCINIFG is set when result is within the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
0	ADCIFG0	ADCIFG0 set when ADC conversion is complete. ADCIFG0 is cleared when ADCMEM0 is read. (0=No IRQ Pending; 1=IRQ Pending)

15.2 ADC OPERATION ON THE MSP430FR2355

- There are three interrupts that can be used with a threshold window feature where the ADC watches for whether the input is within the window (ADCINIFG), below the window (ADCLOIFG), or above the window (ADCHIIFG).

ADC Interrupt Flag (ADCIFG) Register

p:	15	14	13	12	11	10	9	8
	Reserved							
Reset:	0	0	0	0	0	0	0	0

p:	7	6	5	4	3	2	1	0
	Reserved		ADCTOVIFG	ADCOVIFG	ADCHIIFG	ADCLOIFG	ADCINIFG	ADCIFG0
Reset:	0	0	0	0	0	0	0	0

Bit	Field	Description
15:6	Reserved	-
5	ADCTOVIFG	ADCTOVIFG is set when an ADC conversion is triggered before the current conversion has completed. (0=No IRQ Pending; 1=IRQ Pending)
4	ADCOVIFG	ADCOVIFG is set when ADCMEM0 is written before the last conversion result has been read. (0=No IRQ Pending; 1=IRQ Pending)
3	ADCHIIFG	ADCHIIFG is set when result is above the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
2	ADCLOIFG	ADCLOIFG is set when result is below the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
1	ADCINIFG	ADCINIFG is set when result is within the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
0	ADCIFG0	ADCIFG0 set when ADC conversion is complete. ADCIFG0 is cleared when ADCMEM0 is read. (0=No IRQ Pending; 1=IRQ Pending)

15.2 ADC OPERATION ON THE MSP430FR2355

- There is also one interrupt to indicate if ADCMEM0 has been written to before the last conversion result has been read (ADCOVIFG).

ADC Interrupt Flag (ADCIFG) Register

p:	15	14	13	12	11	10	9	8
	Reserved							
Reset:	0	0	0	0	0	0	0	0

p:	7	6	5	4	3	2	1	0
	Reserved		ADCTOVIFG	ADCOVIFG	ADCHIIFG	ADCLOIFG	ADCINIFG	ADCIFG0
Reset:	0	0	0	0	0	0	0	0

Bit	Field	Description
15:6	Reserved	-
5	ADCTOVIFG	ADCTOVIFG is set when an ADC conversion is triggered before the current conversion has completed. (0=No IRQ Pending; 1=IRQ Pending)
4	ADCOVIFG	ADCOVIFG is set when ADCMEM0 is written before the last conversion result has been read. (0=No IRQ Pending; 1=IRQ Pending)
3	ADCHIIFG	ADCHIIFG is set when result is above the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
2	ADCLOIFG	ADCLOIFG is set when result is below the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
1	ADCINIFG	ADCINIFG is set when result is within the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
0	ADCIFG0	ADCIFG0 set when ADC conversion is complete. ADCIFG0 is cleared when ADCMEM0 is read. (0=No IRQ Pending; 1=IRQ Pending)

15.2 ADC OPERATION ON THE MSP430FR2355

- Finally, there is one interrupt to indicate that a new conversion is triggered before the current conversion has completed (ADCTOVIFG).

ADC Interrupt Flag (ADCIFG) Register

p:	15	14	13	12	11	10	9	8
	Reserved							
Reset:	0	0	0	0	0	0	0	0

p:	7	6	5	4	3	2	1	0
	Reserved		ADCTOVIFG	ADCOVIFG	ADCHIIFG	ADCLOIFG	ADCINIFG	ADCIFG0
Reset:	0	0	0	0	0	0	0	0

Bit	Field	Description
15:6	Reserved	-
5	ADCTOVIFG	ADCTOVIFG is set when an ADC conversion is triggered before the current conversion has completed. (0=No IRQ Pending; 1=IRQ Pending)
4	ADCOVIFG	ADCOVIFG is set when ADCMEM0 is written before the last conversion result has been read. (0=No IRQ Pending; 1=IRQ Pending)
3	ADCHIIFG	ADCHIIFG is set when result is above the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
2	ADCLOIFG	ADCLOIFG is set when result is below the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
1	ADCINIFG	ADCINIFG is set when result is within the window comparator thresholds. (0=No IRQ Pending; 1=IRQ Pending)
0	ADCIFG0	ADCIFG0 set when ADC conversion is complete. ADCIFG0 is cleared when ADCMEM0 is read. (0=No IRQ Pending; 1=IRQ Pending)

15.2 ADC OPERATION ON THE MSP430FR2355

ADC Interrupt Enable (ADCIE) Register

p:	15	14	13	12	11	10	9	8
	Reserved							
Reset:	0	0	0	0	0	0	0	0

p:	7	6	5	4	3	2	1	0
	Reserved		ADCTOVIE	ADCOVIE	ADCHIIE	ADCLOIE	ADCINIE	ADCIE0
Reset:	0	0	0	0	0	0	0	0

Bit	Field	Description
15:6	Reserved	-
5	ADCTOVIE	ADC Conversion-Time-Overflow IRQ Enable (0=Disabled; 1=Enabled)
4	ADCOVIE	ADCMEM0 Register Overflow IRQ Enable (0=Disabled; 1=Enabled)
3	ADCHIE	ADC Above Upper Threshold of Window IRQ Enable (0=Disabled; 1=Enabled)
2	ADCLOIE	ADC Below Lower Threshold of Window IRQ Enable (0=Disabled; 1=Enabled)
1	ADCINIE	ADC Inside Window Comparator IRQ Enable (0=Disabled; 1=Enabled)
0	ADCIE0	ADC Conversion Complete IRQ Enable (0=Disabled; 1=Enabled)

15.2 ADC OPERATION ON THE MSP430FR2355

ADC Interrupt Vector (ADCIV) Register

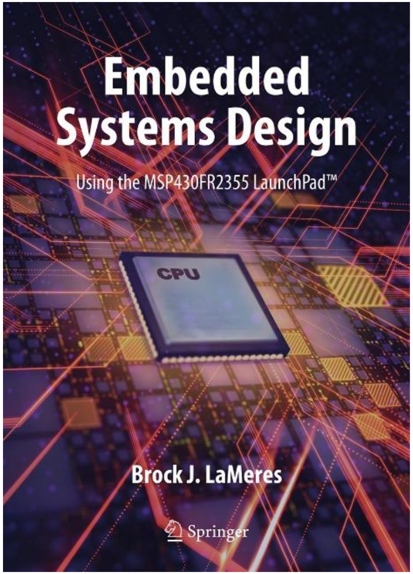
p:	15	14	13	12	11	10	9	8
	ADCIVx							
Reset:	0	0	0	0	0	0	0	0

p:	7	6	5	4	3	2	1	0
	ADCIVx							
Reset:	0	0	0	0	0	0	0	0

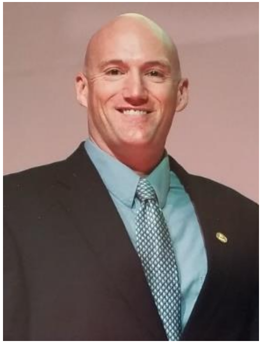
Bit	Field	Description
15:0	ADCIVx	ADC Interrupt Vector. 00h = No IRQ Pending. 02h = Interrupt Source → ADCMEM0 overflow; Interrupt Flag → ADCOVIFG (highest priority). 04h = Interrupt Source → Conversion time overflow; Interrupt Flag → ADCTOVIFG. 06h = Interrupt Source → Above window comparator threshold; Interrupt Flag → ADCHIIFG. 08h = Interrupt Source → Below window comparator threshold; Interrupt Flag → ADCLOIFG. 0Ah = Interrupt Source → Within window comparator threshold; Interrupt Flag → ADCINIFG. 0Ch = Interrupt Source → ADC conversion complete; Interrupt Flag → ADCIFG0 (lowest priority).

CHAPTER 15: ANALOG TO DIGITAL CONVERTERS

15.2 ADC OPERATION ON THE MSP430FR2355 - CONFIGURATION

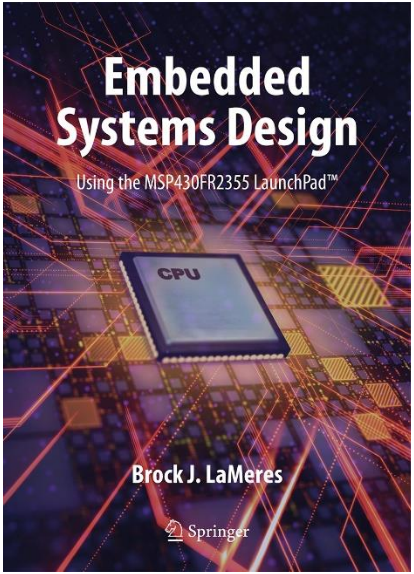


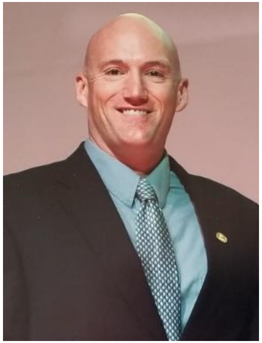
www.youtube.com/c/DigitalLogicProgramming_LaMeres



CHAPTER 15: ANALOG TO DIGITAL CONVERTERS

15.2 ADC OPERATION ON THE MSP430FR2355 – EXAMPLE: READING AN ANALOG VOLTAGE USING POLLING CONVERSION-COMPLETE IFG



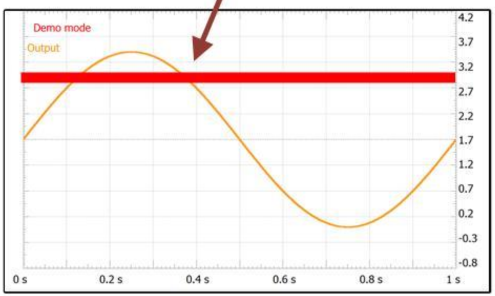


BROCK J. LAMERES, PH.D.

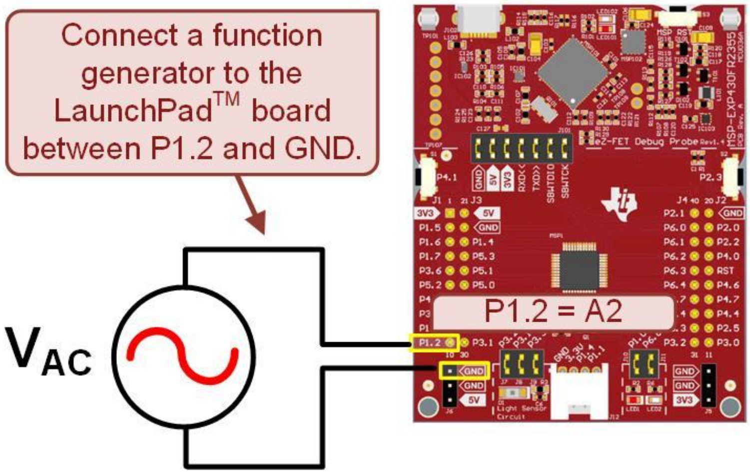
READING AN ANALOG VOLTAGE WITH THE ADC USING POLLING TO MONITOR CONVERSION-COMPLETE

We are going to set up the ADC on the LaunchPad™ board to read an analog voltage on P1.2. We will drive P1.2 with a sine wave voltage from a function generator. The sine wave will have an amplitude of 1.7v, an offset of 1.7v, and frequency of 1Hz. These settings will produce an analog voltage that will cycle between 0v and +3.4v every second. When the voltage is **below +3.0v**, we will light up LED2 (green) on the LaunchPad™ board. When the voltage is **above +3.0v**, we will light up LED1 (red) on the LaunchPad™ board. This program emulates an "over-voltage monitor" application. The following is the setup for this example.

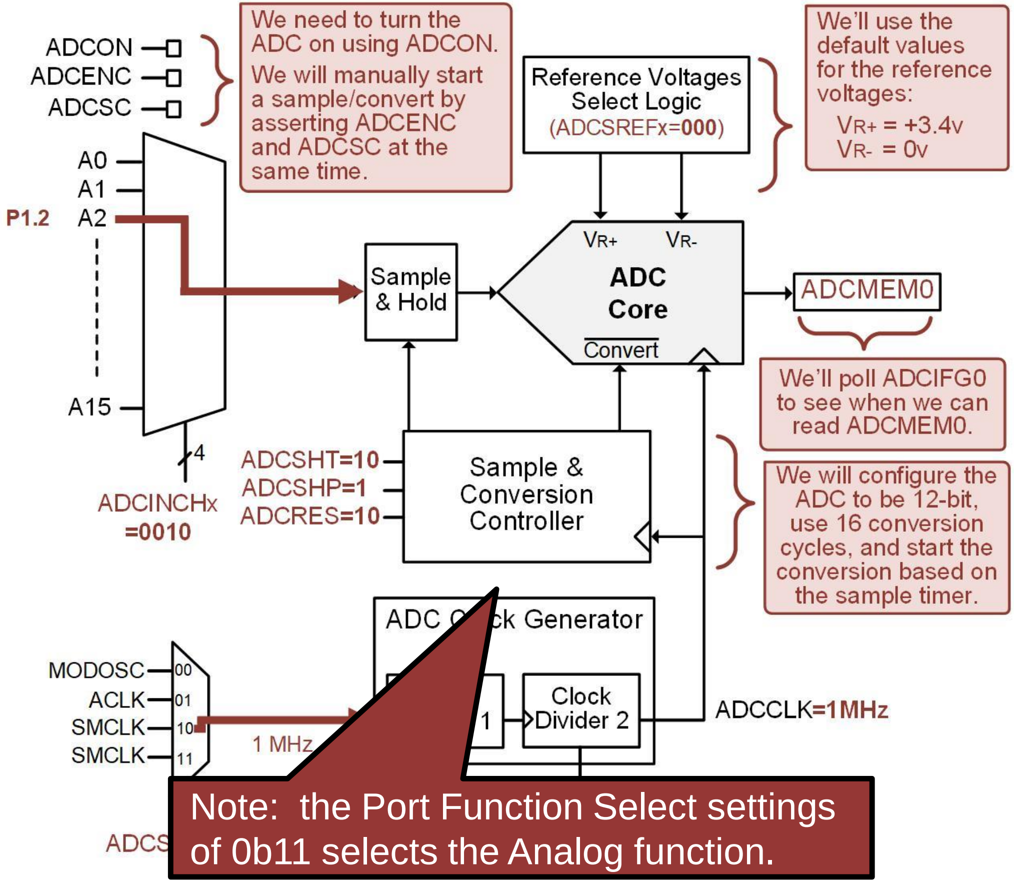
When the sine wave gets above +3.0v, we will light LED1 (red); otherwise we will light LED2 (green).



Connect a function generator to the LaunchPad™ board between P1.2 and GND.



READING AN ANALOG VOLTAGE WITH THE ADC USING POLLING TO MONITOR CONVERSION-COMPLETE



Reading an Analog Voltage With the ADC Using Polling to Monitor Conversion-Complete

Step 1: In CCS, create a new C/C++ Empty Project (with main.c) titled:

C_ADC_Sampling_P1.2_Polling

Step 2: Type in the following code in main.c after the statement to stop the watchdog timer.

READING AN ANALOG VOLTAGE WITH THE ADC USING POLLING TO MONITOR CONVERSION-COMPLETE

```
#include <msp430.h>
unsigned int ADC_Value;
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
    //-- Configure Ports
    P1DIR |= BIT0;            // Config P1.0 (LED1) as output
    P6DIR |= BIT6;            // Config P6.6 (LED2) as output
    P1SEL1 |= BIT2;           // Configure P1.2 Pin for A2
    P1SEL0 |= BIT2;
    PM5CTL0 &= ~LOCKLPM5;     // Turn on GPIO
    //-- Configure ADC
    ADCCTL0 &= ~ADCSHT;       // Clear ADCSHT from def. of ADCSHT=01
    ADCCTL0 |= ADCSHT_2;      // Conversion Cycles = 16 (ADCSHT=10)
    ADCCTL0 |= ADCON;         // Turn ADC ON
    ADCCTL1 |= ADCSSEL_2;     // ADC Clock Source = SMCLK
    ADCCTL1 |= ADCSHP;        // Sample signal source = sampling timer
    ADCCTL2 &= ~ADCRES;       // Clear ADCRES from def. of ADCRES=01
    ADCCTL2 |= ADCRES_2;      // Resolution = 12-bit (ADCRES=10)
    ADCMCTL0 |= ADCINCH_2;    // ADC Input Channel = A2 (P1.2)
    while(1)
    {
        ADCCTL0 |= ADCENC | ADCSC;         // Enable and Start conversion
        while((ADCIFG & ADCIFG0) == 0 );   // wait for conv. complete
        ADC_Value = ADCMEM0;               // Read ADC result

        if(ADC_Value > 3613){ // If (A2 > 3v)
            P1OUT |= BIT0;    //   LED1=ON (red)
            P6OUT &= ~BIT6;   //   LED2=OFF
        } else {              // If (A2 < 3v)
            P1OUT &= ~BIT0;   //   LED1=OFF
            P6OUT |= BIT6;    //   LED2=ON (green)
        }
    }
    return 0;
}
```

- Change mode for P1.2 to ADC channel A2.

- Conversion cycles = 16.
- ADC = on.

- Use SMCLK.
- Sample timer.

- 12-bit resolution

- Send A2 to ADC.

- Start ADC.

- Wait.

- Read result.

- Check the value that was read and assert / de assert the LEDs accordingly.

READING AN ANALOG VOLTAGE WITH THE ADC USING POLLING TO MONITOR CONVERSION-COMPLETE

Step 3: Save, debug, and run your program. Also run your function generator to produce the sine wave on P1.2.

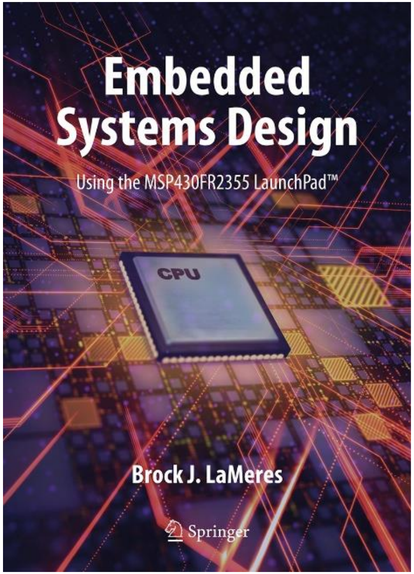
Did it work? You should see the green LED on most of the time. But when the input signal gets above +3.0v during the top part of the sine wave, the red LED will turn on and the green LED will turn off.

Image Courtesy of https://neodem.wp.horizon.ac.uk/

Embedded Systems Design

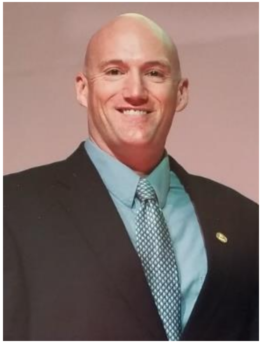
Chapter 15: Analog to Digital Converters

15.2 ADC Operation on the MSP430FR2355 – Example: Reading an Analog Voltage using Polling Conversion-Complete IFG



Subscribe

www.youtube.com/c/DigitalLogicProgramming_LaMeres

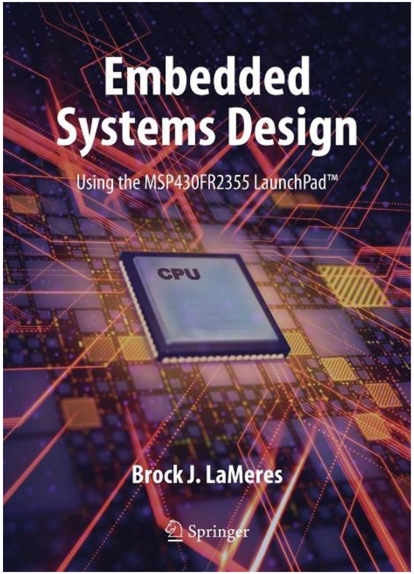


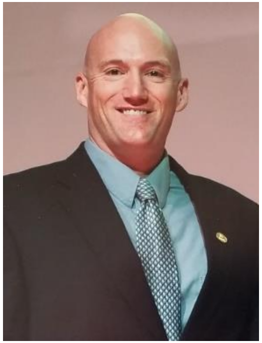
Brock J. LaMeres, Ph.D.

Embedded Systems Design

Chapter 15: Analog to Digital Converters

15.2 ADC Operation on the MSP430FR2355 – Example: Reading an Analog Voltage using Polling Conversion-Complete IRQ



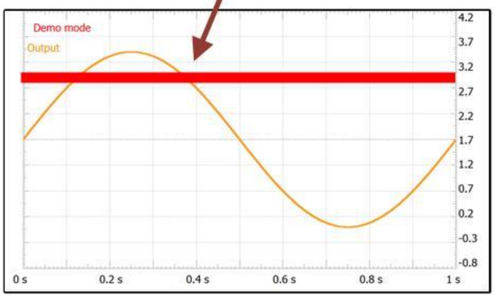


Brock J. LaMeres, Ph.D.

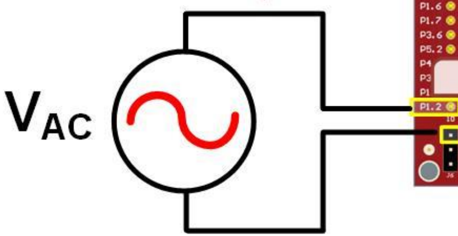
READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE

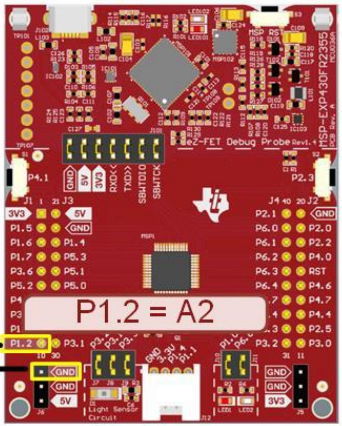
We are going to set up the ADC on the LaunchPad™ board to read an analog voltage on P1.2. We will drive P1.2 with a sine wave voltage from a function generator. The sine wave will have an amplitude of 1.7v, an offset of 1.7v, and frequency of 1Hz. These settings will produce an analog voltage that will cycle between 0v and +3.4v every second. When the voltage is **below +3.0v**, we will light up LED2 (green) on the LaunchPad™ board. When the voltage is **above +3.0v**, we will light up LED1 (red) on the LaunchPad™ board. This program emulates an "over-voltage monitor" application. The following is the setup for this example.

When the sine wave gets above +3.0v, we will light LED1 (red); otherwise we will light LED2 (green).

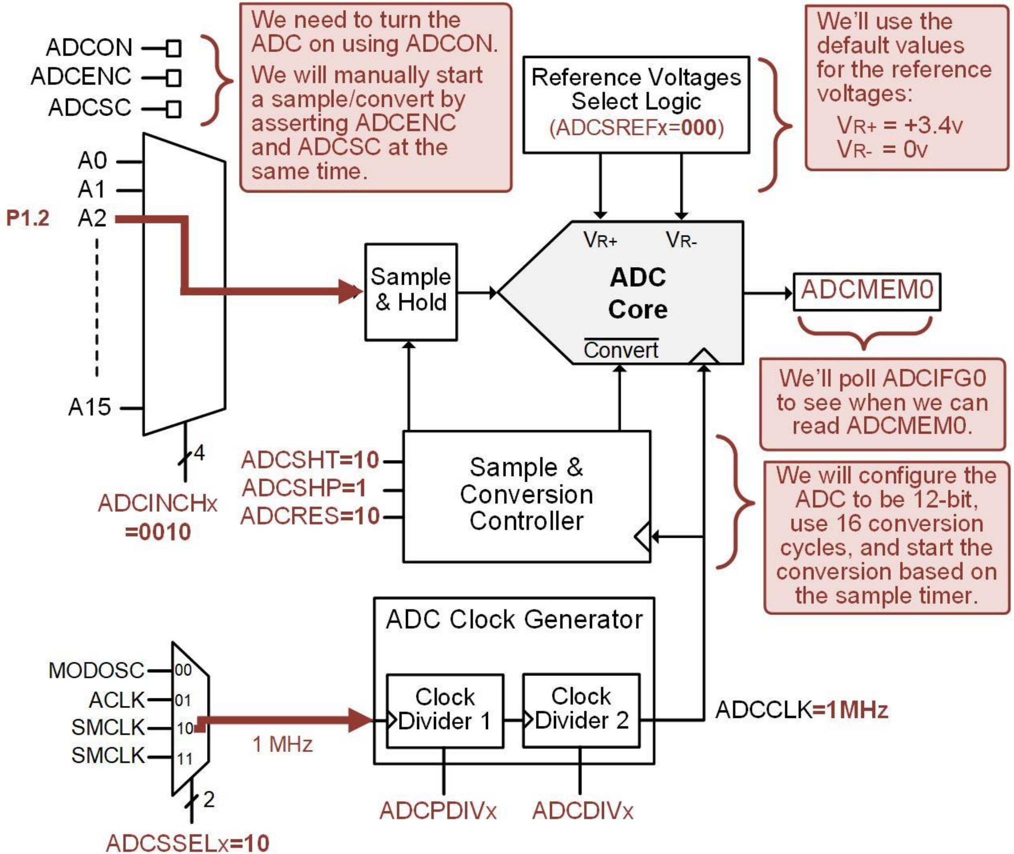


Connect a function generator to the LaunchPad™ board between P1.2 and GND.

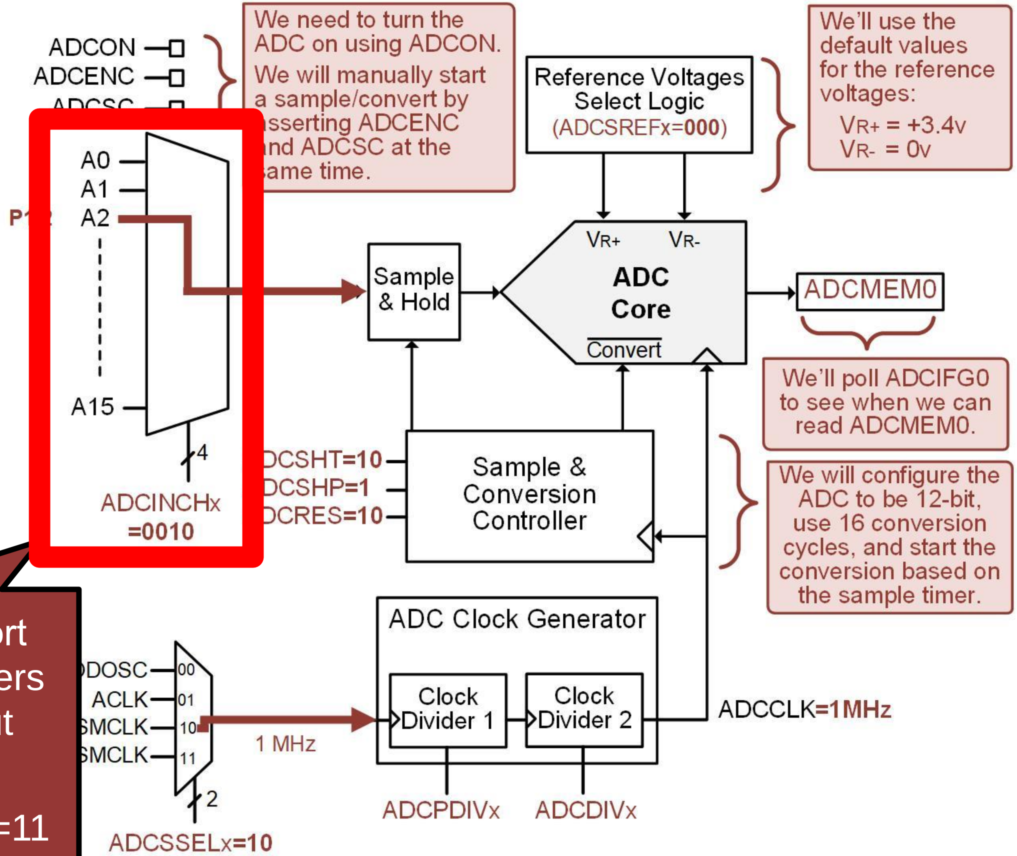




READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE



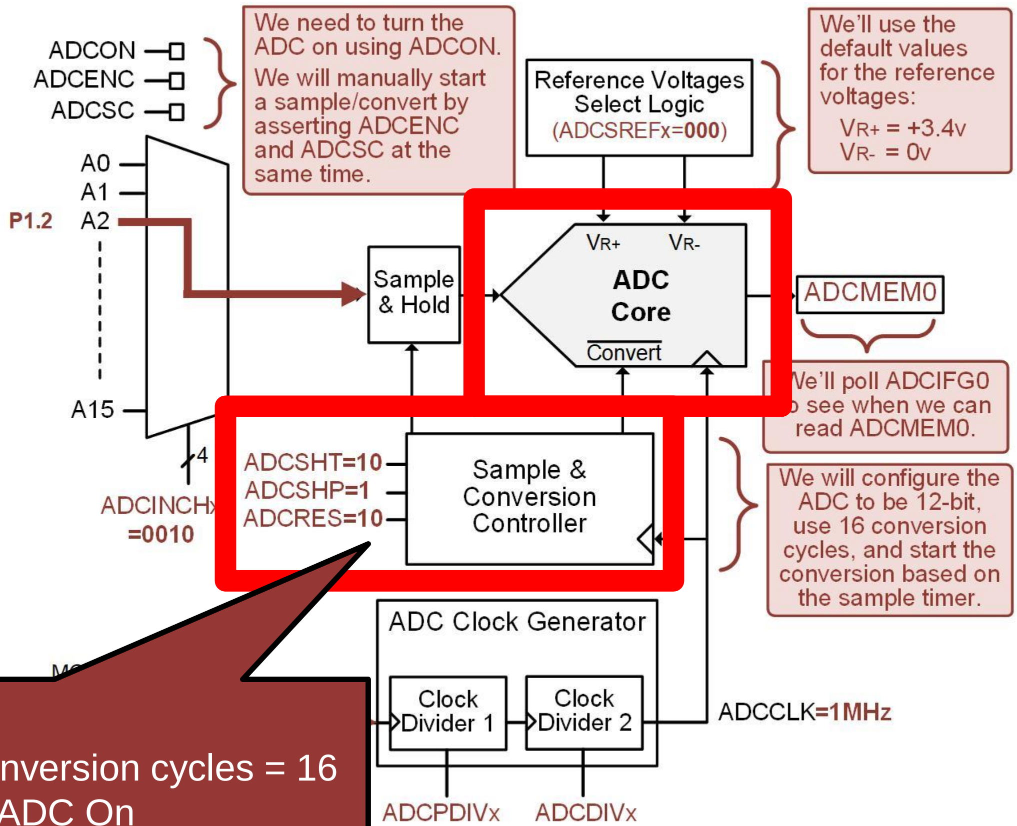
READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE



We'll first setup the Port Function Select registers to use the Analog input on P1.2.

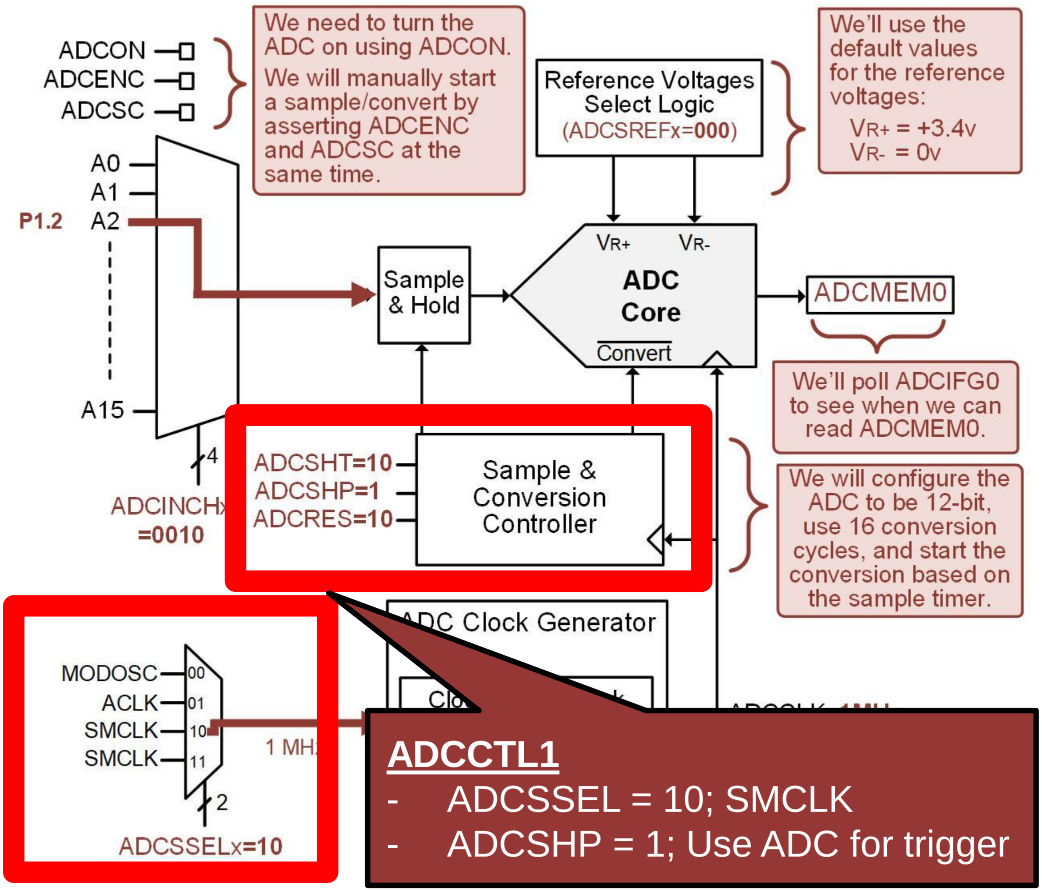
P1SEL1(2):P1SEL(0)=11

READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE

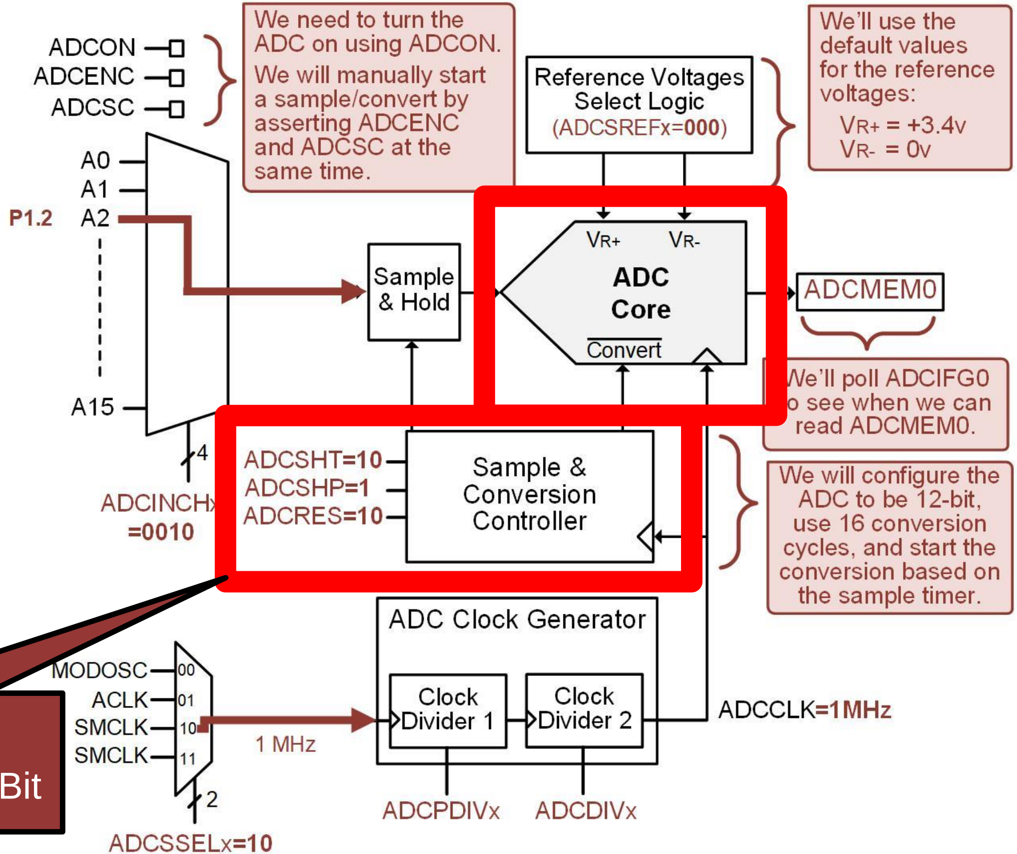


ADCCTL0
- ADCSHT = 10; Conversion cycles = 16
- ADCON = 1; Turn ADC On

READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE



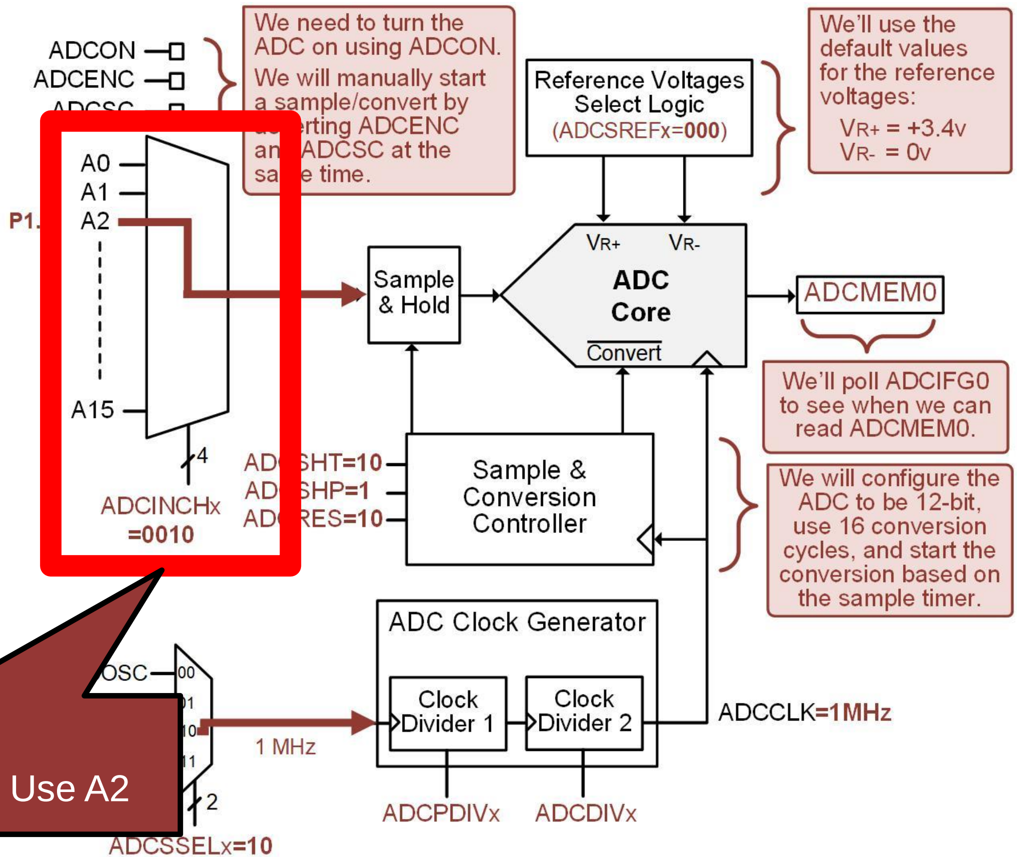
READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE



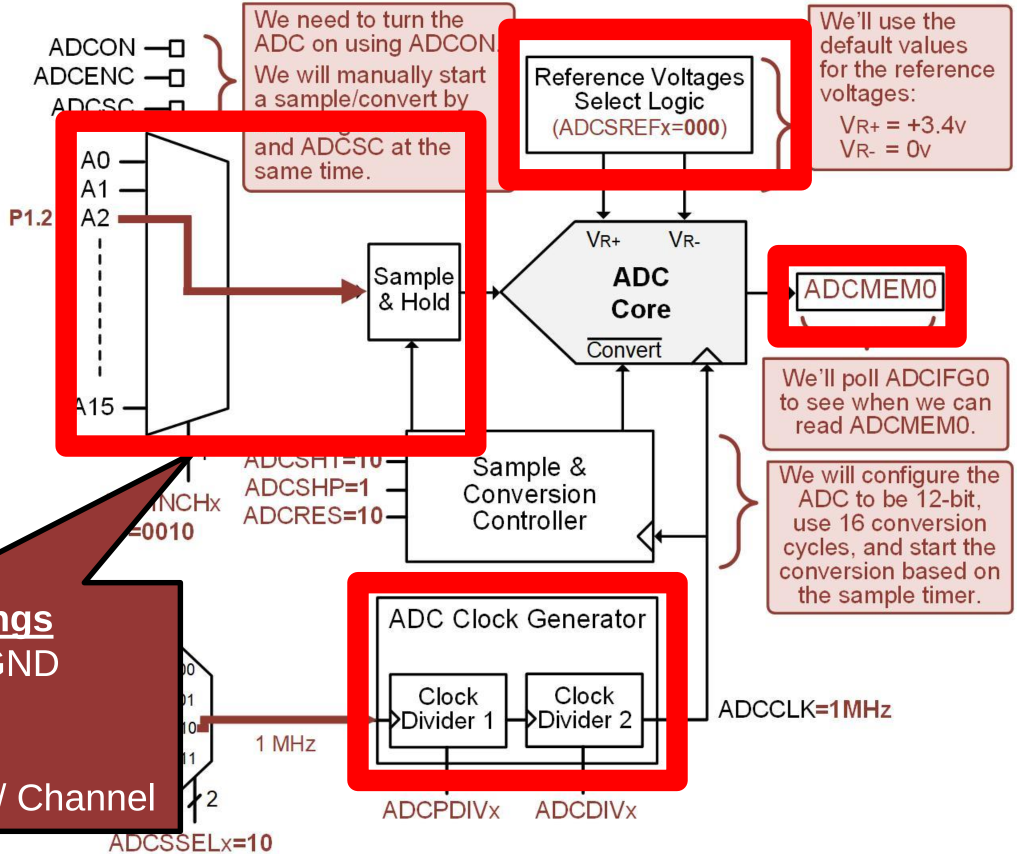
ADCCTL2

- ADCRES = 10; 12-Bit

READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE



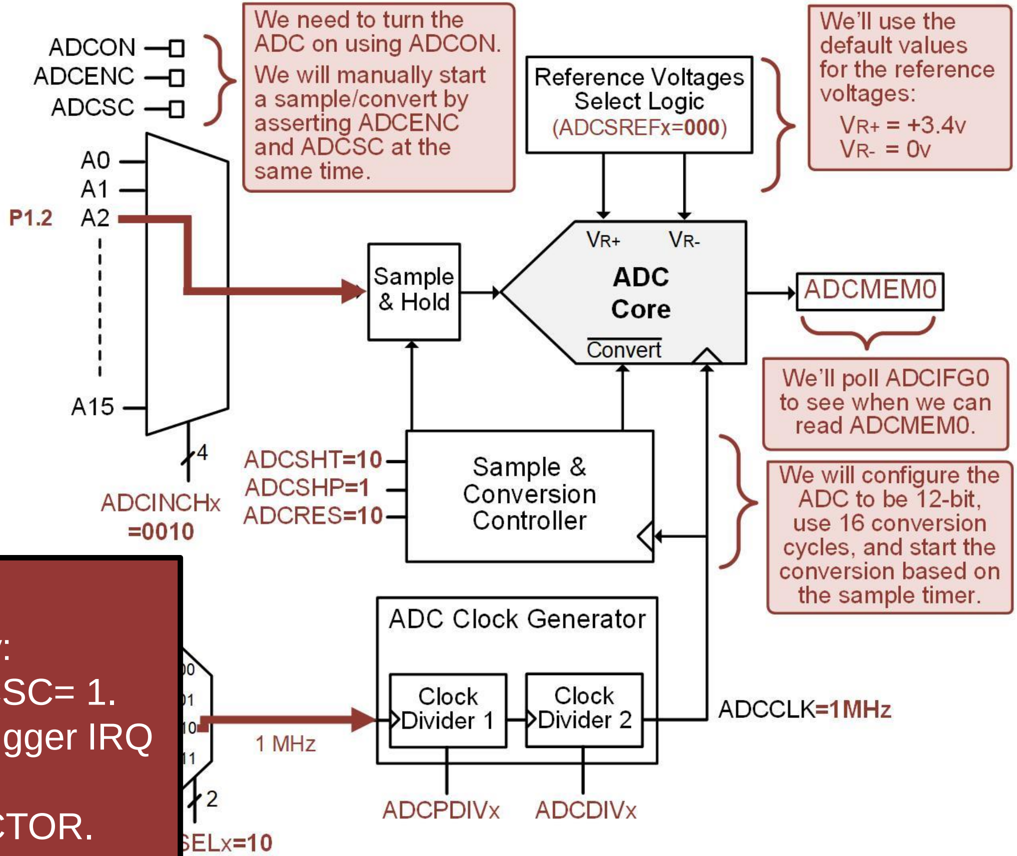
READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE



Notable Default Settings

- V_{R+} = VCC, V_{R-} = GND
- Prescalars = 1
- Format = Unsigned
- Single Conversion / Channel

READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE



Our Design

- Start conversion by: ADCENC = 1, ADCSC= 1.
- Use ADCIFG0 to trigger IRQ when complete.
- Vector = ADC_VECTOR.

Reading an Analog Voltage With the ADC Using an IRQ to Monitor Conversion-Complete

Step 1: In CCS, create a new C/C++ Empty Project (with main.c) titled:

C_ADC_Sampling_P1.2_IRQ

Step 2: Type in the following code in main.c after the statement to stop the watchdog timer.

Image Courtesy of https://neodem.wp.horizon.ac.uk/

READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE

Step 3: Save, debug, and run your program. Also run your function generator to produce the sine wave on P1.2.

Did it work? You should see the same behavior as in the last polling example.

```c
#include <msp430.h>
unsigned int ADC_Value;
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

    //-- Configure Ports
    P1DIR |= BIT0;             // Config P1.0 (LED1) as output
    P6DIR |= BIT6;             // Config P6.6 (LED2) as output

    P1SEL1 |= BIT2;            // Configure P1.2 Pin for A2
    P1SEL0 |= BIT2;

    PM5CTL0 &= ~LOCKLPM5;      // Turn on GPIO

    //-- Configure ADC
    ADCCTL0 &= ~ADCSHT;        // Clear ADCSHT from def. of ADCSHT=01
    ADCCTL0 |= ADCSHT_2;       // Conversion Cycles = 16 (ADCSHT=10)
    ADCCTL0 |= ADCON;          // Turn ADC ON

    ADCCTL1 |= ADCSSEL_2;      // ADC Clock Source = SMCLK
    ADCCTL1 |= ADCSHP;         // Sample signal source = sampling timer

    ADCCTL2 &= ~ADCRES;        // Clear ADCRES from def. of ADCRES=01
    ADCCTL2 |= ADCRES_2;       // Resolution = 12-bit (ADCRES=10)

    ADCMCTL0 |= ADCINCH_2;     // ADC Input Channel = A2 (P1.2)

    ADCIE |= ADCIE0;           // Enable ADC Conv Complete IRQ
    __enable_interrupt();      // Enable Maskable IRQs

    while(1)
    {
        ADCCTL0 |= ADCENC | ADCSC;        // Enable and Start conversion
        while((ADCIFG & ADCIFG0) == 0 );  // wait for conv. complete
    }
    return 0;
}

//------- Interrupt Service Routines --------------------
#pragma vector=ADC_VECTOR
__interrupt void ADC_ISR(void){
    ADC_Value = ADCMEM0;         // Read ADC value

    if(ADC_Value > 3613){        // If (A2 > 3v)
        P1OUT |= BIT0;           //   LED1=ON (red)
        P6OUT &= ~BIT6;          //   LED2=OFF
    } else {                     // If (A2 < 3v)
        P1OUT &= ~BIT0;          //   LED1=OFF
        P6OUT |= BIT6;           //   LED2=ON (green)
    }
}
```

- Change mode for P1.2 to ADC channel A2.

- Conversion cycles = 16.
- ADC = on.

- Use SMCLK.
- Sample timer.

- 12-bit resolution

- Send A2 to ADC.

- Enable ADC conversion complete IRQ (ADCIFG0).

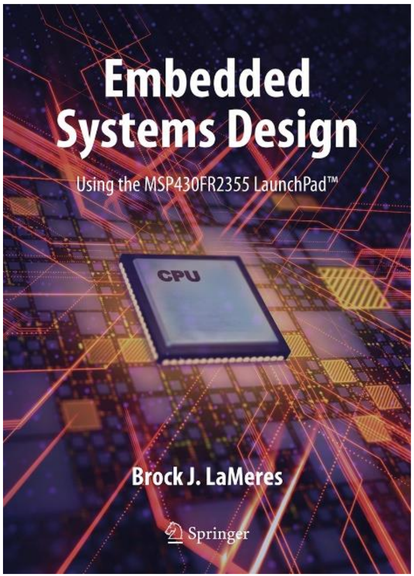
- Start ADC.

- Wait until conversion is complete before starting loop over.

- The functionality to read the ADC result and configure the LEDs is put into an ISR.

CHAPTER 15: ANALOG TO DIGITAL CONVERTERS

15.2 ADC OPERATION ON THE MSP430FR2355 – EXAMPLE: READING AN ANALOG VOLTAGE USING POLLING CONVERSION-COMPLETE IRQ



Subscribe

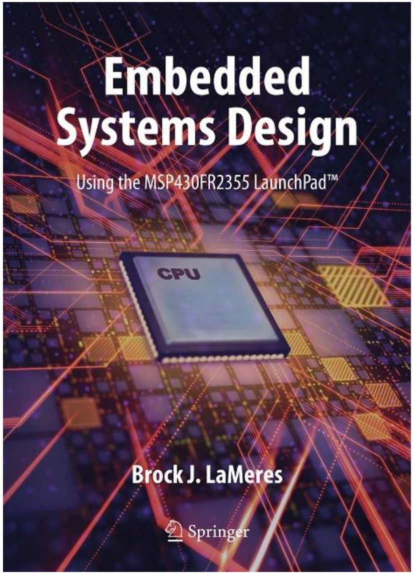
www.youtube.com/c/DigitalLogicProgramming_LaMeres

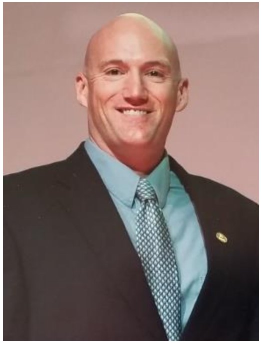
BROCK J. LAMERES, PH.D.

Embedded Systems Design

Chapter 15: Analog to Digital Converters

15.2 ADC Operation on the MSP430FR2355 – Example: Reading Analog Voltage w Conversion-Complete IRQ & LPM



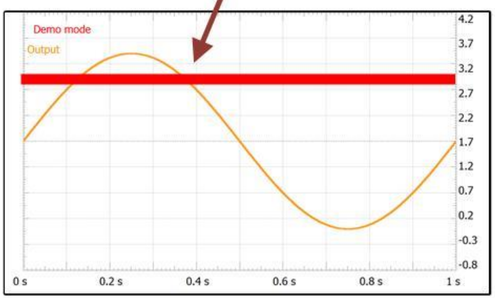


Brock J. LaMeres, Ph.D.

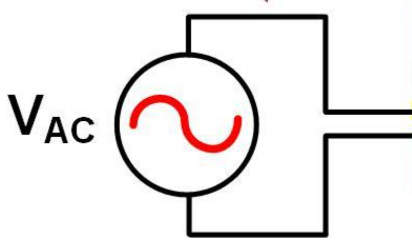
Reading Analog Voltage w Conversion-Complete IRQ & LPM

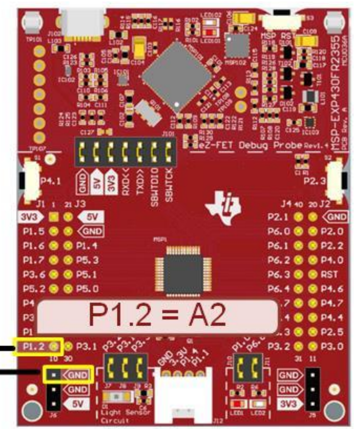
We are going to set up the ADC on the LaunchPad™ board to read an analog voltage on P1.2. We will drive P1.2 with a sine wave voltage from a function generator. The sine wave will have an amplitude of 1.7v, an offset of 1.7v, and frequency of 1Hz. These settings will produce an analog voltage that will cycle between 0v and +3.4v every second. When the voltage is **below +3.0**v, we will light up LED2 (green) on the LaunchPad™ board. When the voltage is **above +3.0**v, we will light up LED1 (red) on the LaunchPad™ board. This program emulates an "over-voltage monitor" application. The following is the setup for this example.

When the sine wave gets above +3.0v, we will light LED1 (red); otherwise we will light LED2 (green).

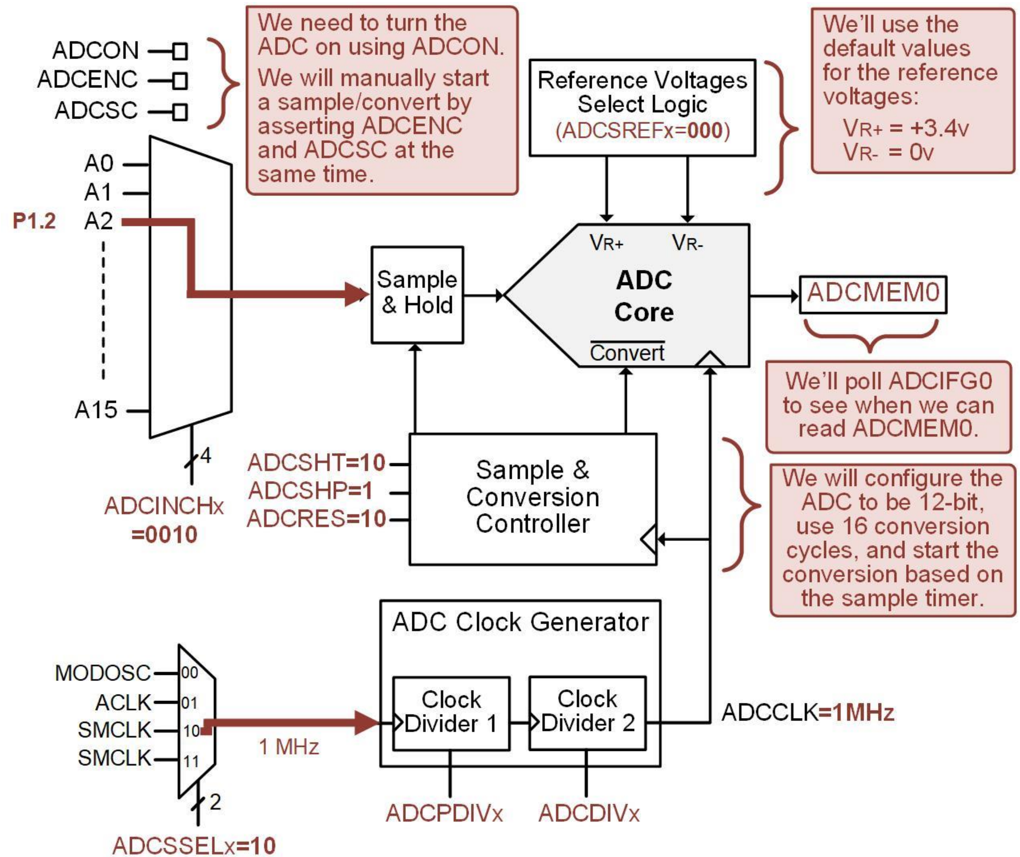


Connect a function generator to the LaunchPad™ board between P1.2 and GND.

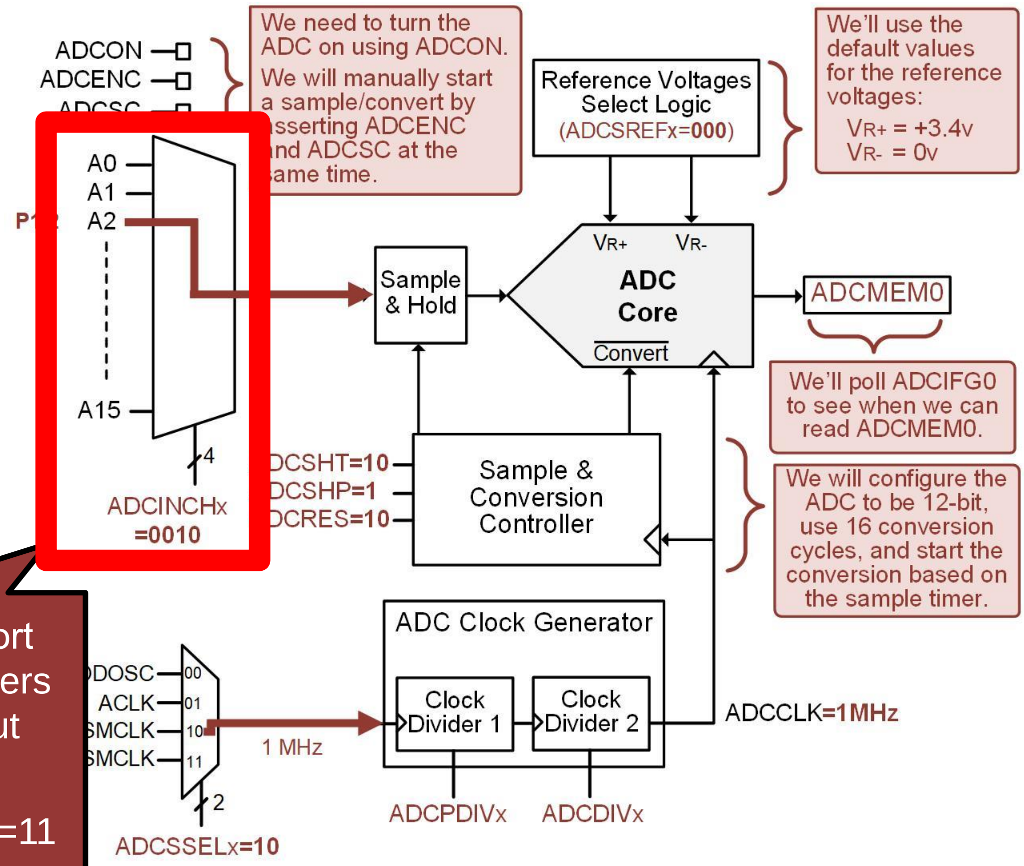




READING ANALOG VOLTAGE W CONVERSION-COMPLETE IRQ & LPM



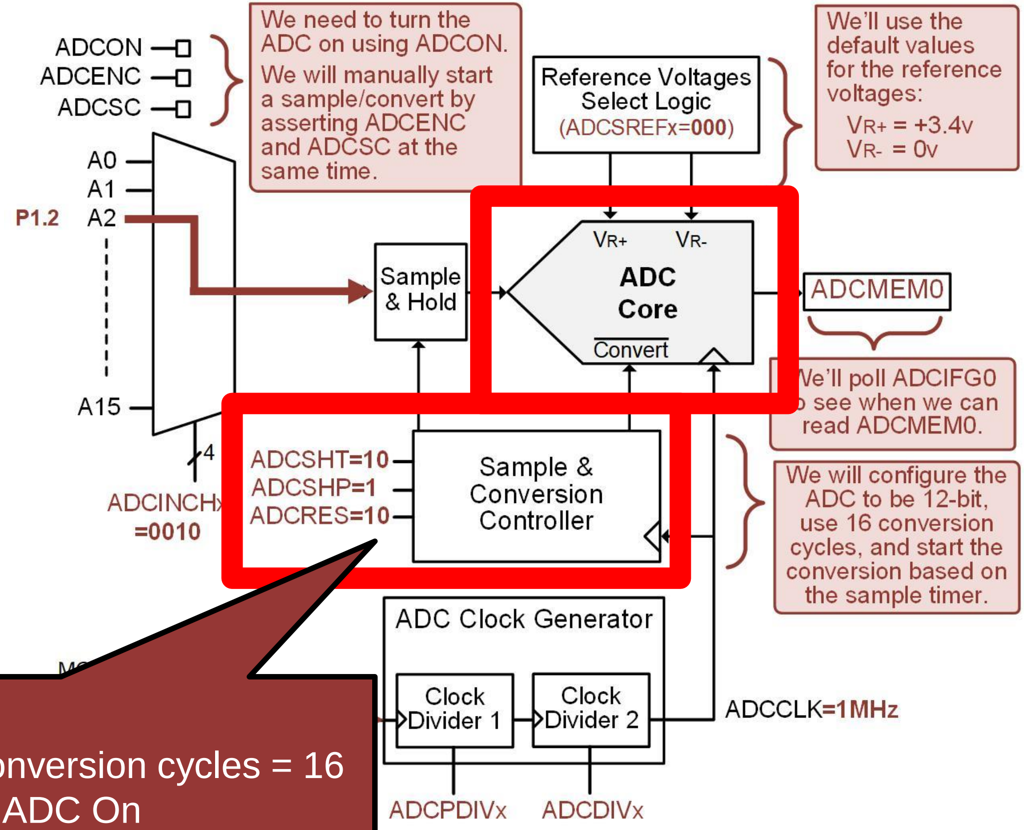
READING ANALOG VOLTAGE W CONVERSION-COMPLETE IRQ & LPM



We'll first setup the Port Function Select registers to use the Analog input on P1.2.

P1SEL1(2):P1SEL(0)=11

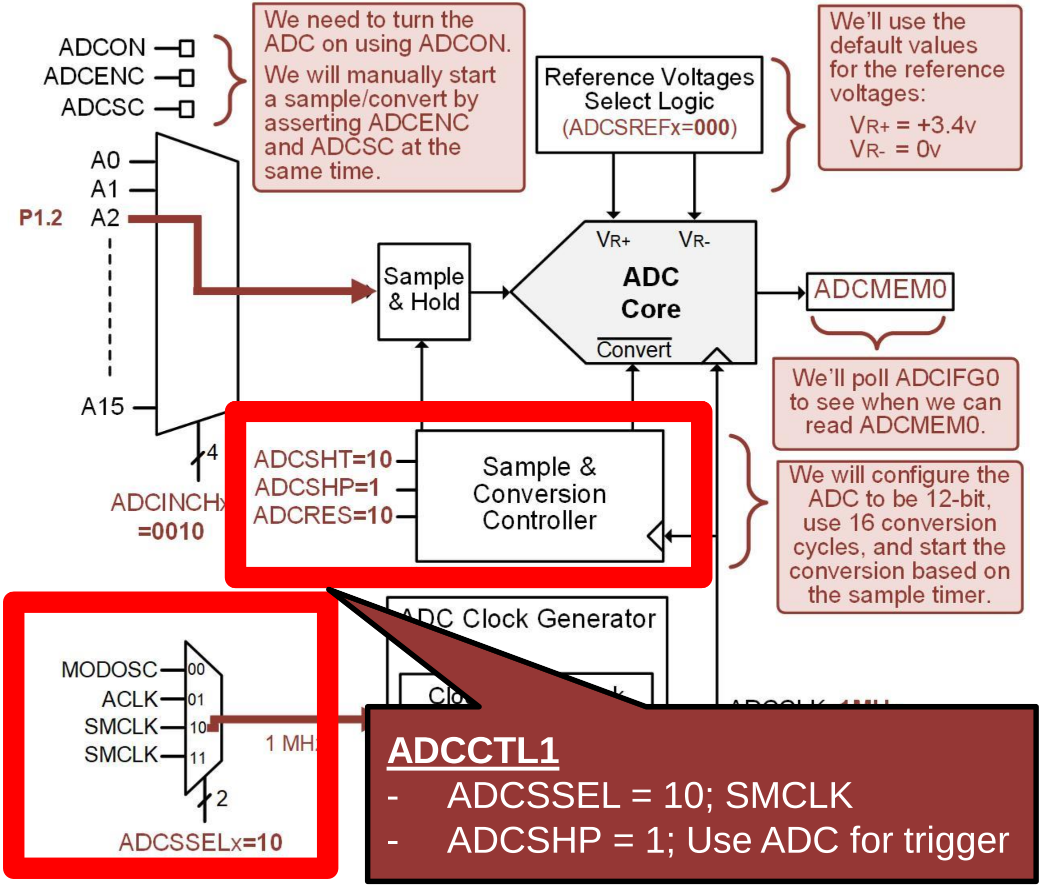
READING ANALOG VOLTAGE W CONVERSION-COMPLETE IRQ & LPM



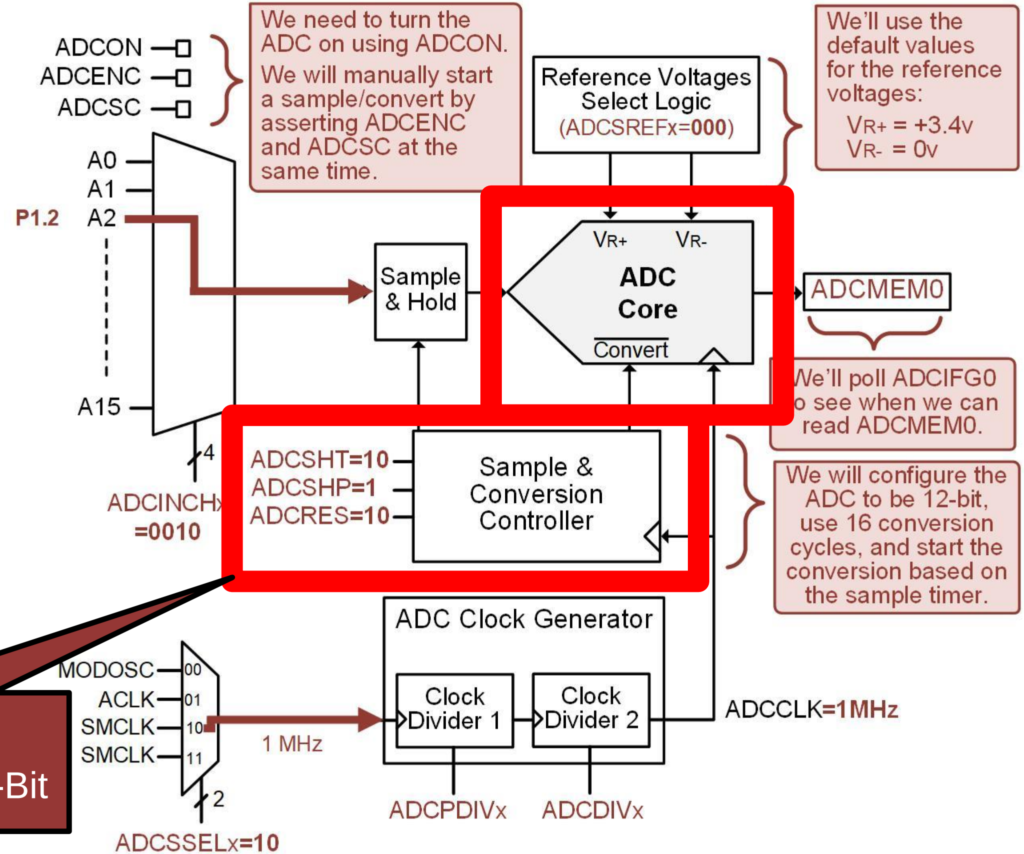
ADCCTL0

- ADCSHT = 10; Conversion cycles = 16
- ADCON = 1; Turn ADC On

READING ANALOG VOLTAGE W CONVERSION-COMPLETE IRQ & LPM



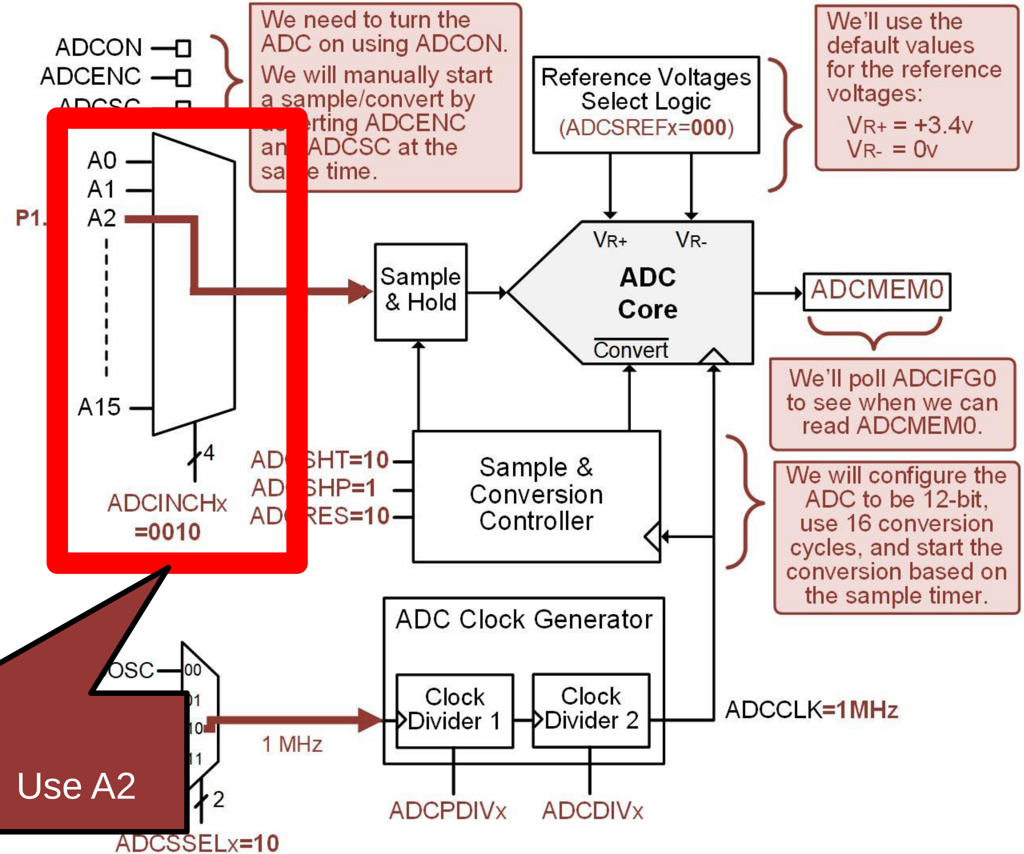
READING ANALOG VOLTAGE W CONVERSION-COMPLETE IRQ & LPM



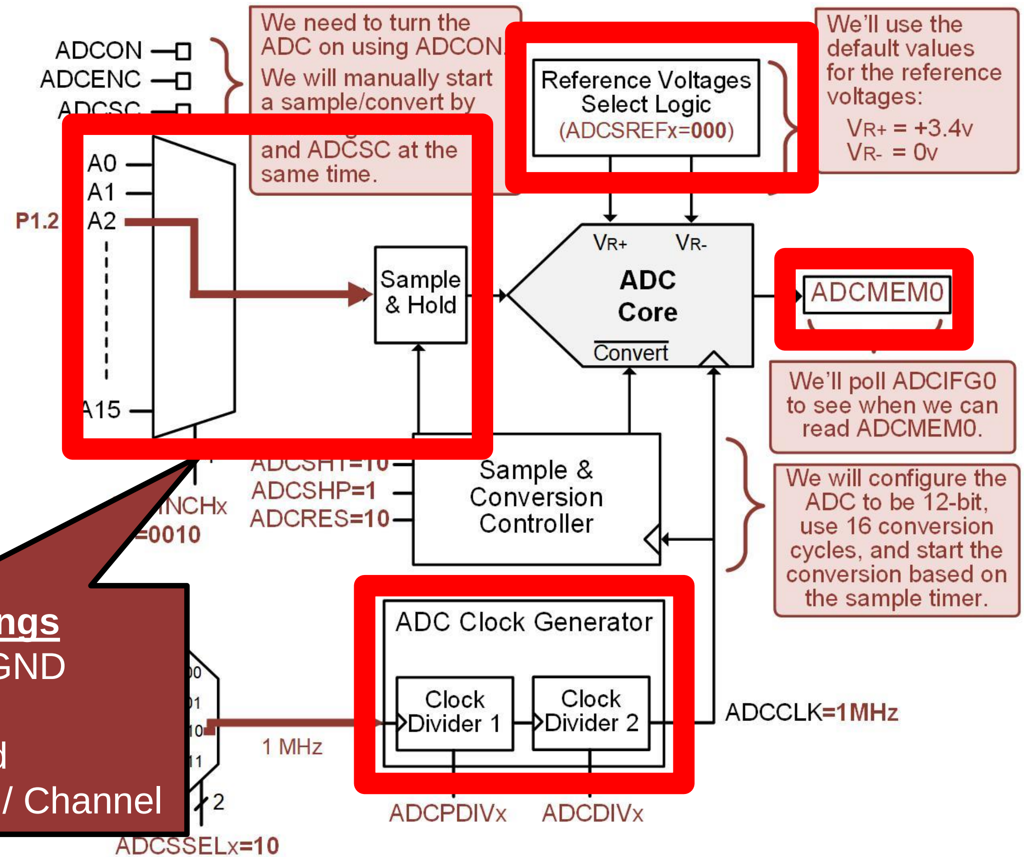
ADCCTL2

- ADCRES = 10; 12-Bit

READING ANALOG VOLTAGE W CONVERSION-COMPLETE IRQ & LPM



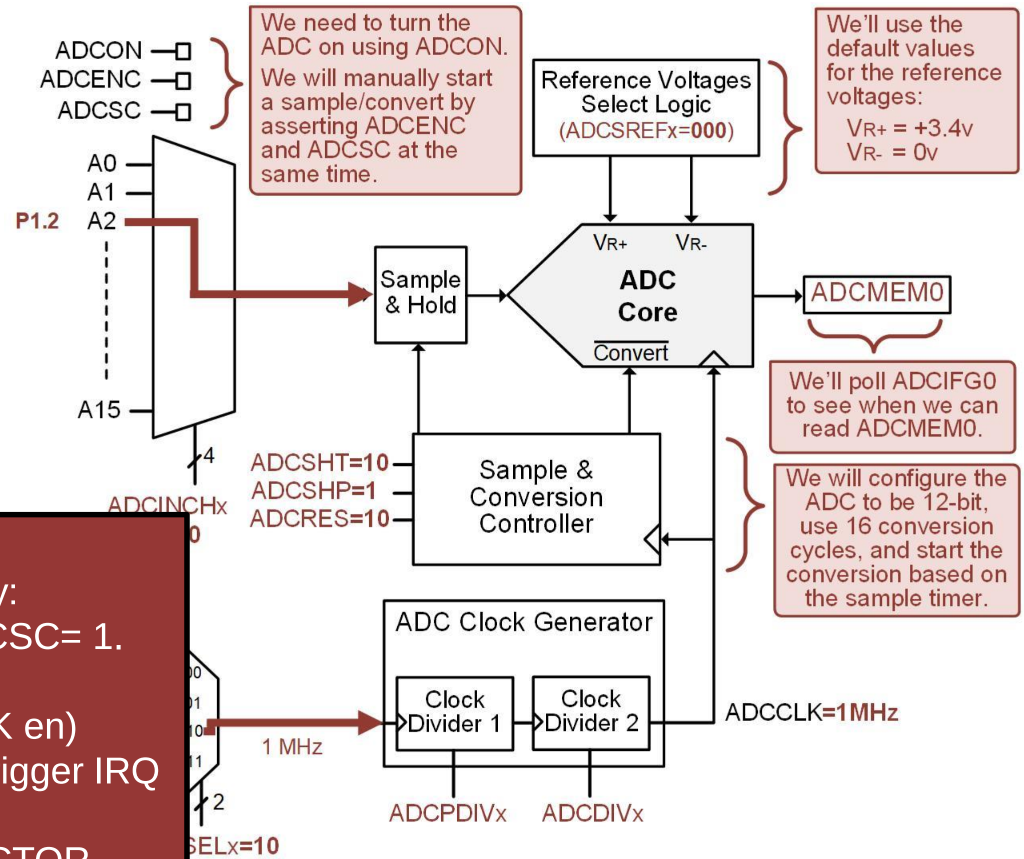
Reading Analog Voltage w Conversion-Complete IRQ & LPM



Notable Default Settings

- V_{R+} = VCC, V_{R-} = GND
- Prescalars = 1
- Format = Unsigned
- Single Conversion / Channel

READING ANALOG VOLTAGE W CONVERSION-COMPLETE IRQ & LPM



Our Design

- Start conversion by: ADCENC = 1, ADCSC= 1.
- Go into LPM0 (MCLK dis, SMCLK en)
- Use ADCIFG0 to trigger IRQ when complete.
- Vector = ADC_VECTOR.

Reading an Analog Voltage With the ADC Using an IRQ and Low Power Mode

Step 1: In CCS, create a new C/C++ Empty Project (with main.c) titled:

C_ADC_Sampling_P1.2_LPM

Step 2: Type in the following code in main.c after the statement to stop the watchdog timer.

Image Courtesy of https://neodem.wp.horizon.ac.uk/

READING AN ANALOG VOLTAGE WITH THE ADC USING AN IRQ TO MONITOR CONVERSION-COMPLETE

Step 3: Save, debug, and run your program. Also run your function generator to produce the sine wave on P1.2.

Did it work? You should see the same behavior as in the last IRQ example.

```c
#include <msp430.h>
unsigned int ADC_Value;
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
    //-- Configure Ports
    P1DIR |= BIT0;           // Config P1.0 (LED1) as output
    P6DIR |= BIT6;           // Config P6.6 (LED2) as output

    P1SEL1 |= BIT2;          // Configure P1.2 Pin for A2
    P1SEL0 |= BIT2;

    PM5CTL0 &= ~LOCKLPM5;    // Turn on GPIO

    //-- Configure ADC
    ADCCTL0 &= ~ADCSHT;      // Clear ADCSHT from def. of ADCSHT=01
    ADCCTL0 |= ADCSHT_2;     // Conversion Cycles = 16 (ADCSHT=10)
    ADCCTL0 |= ADCON;        // Turn ADC ON

    ADCCTL1 |= ADCSSEL_2;    // ADC Clock Source = SMCLK
    ADCCTL1 |= ADCSHP;       // Sample signal source = sampling timer

    ADCCTL2 &= ~ADCRES;      // Clear ADCRES from def. of ADCRES=01
    ADCCTL2 |= ADCRES_2;     // Resolution = 12-bit (ADCRES=10)

    ADCMCTL0 |= ADCINCH_2;   // ADC Input Channel = A2 (P1.2)

    ADCIE |= ADCIE0;         // Enable ADC Conv Complete IRQ

    while(1)
    {
        ADCCTL0 |= ADCENC | ADCSC;            // Enable and Start conv
        __bis_SR_register(GIE | LPM0_bits);   // Enable maskable IRQs,
    }                                         // Turn off CPU for LPM
    return 0;
}
//------- Interrupt Service Routines --------------------
#pragma vector=ADC_VECTOR
__interrupt void ADC_ISR(void){

    ADC_Value = ADCMEM0;                   // Read ADC value
    __bic_SR_register_on_exit(LPM0_bits);  // Wake up CPU

    if(ADC_Value > 3613){       // If (A2 > 3v)
        P1OUT |= BIT0;          //   LED1=ON (red)
        P6OUT &= ~BIT6;         //   LED2=OFF
    } else {                    // If (A2 < 3v)
        P1OUT &= ~BIT0;         //   LED1=OFF
        P6OUT |= BIT6;          //   LED2=ON (green)
    }
}
```

- Change mode for P1.2 to ADC channel A2.

- Conversion cycles = 16.
- ADC = on.

- Use SMCLK.
- Sample timer.

- 12-bit resolution

- Send A2 to ADC.

- Enable ADCIFG0.

- Start ADC.

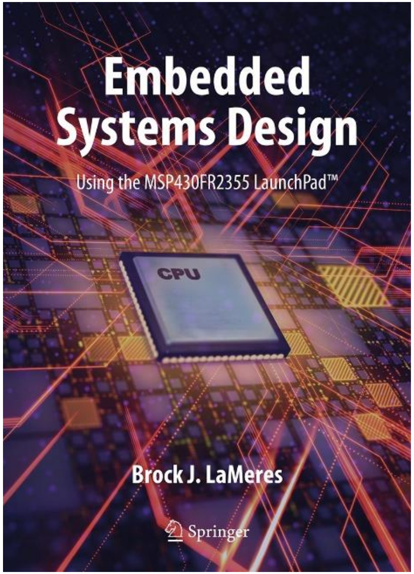
- Instead of waiting, we'll just turn off the CPU by setting the "CPUOFF" bit in SR.

- At this point conversion is complete. We'll read the ADC result and wake up the CPU.

Embedded Systems Design

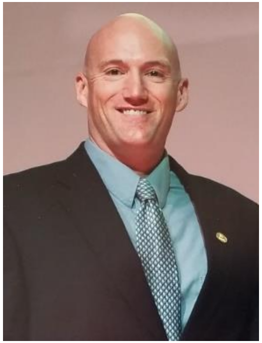
Chapter 15: Analog to Digital Converters

15.2 ADC Operation on the MSP430FR2355 – Example: Reading Analog Voltage w Conversion-Complete IRQ & LPM



Subscribe

www.youtube.com/c/DigitalLogicProgramming_LaMeres



Brock J. LaMeres, Ph.D.