CprE 288 – Introduction to Embedded Systems (Analog-to-Digital Converter)

Dr. Phillip Jones

Announcements

- HW 8: Due Sunday 10/23
- Quiz 7: Thursday, first 10 minutes of class on Canvas
  - GPIO Readings from HW 5
  - Interrupt readings from HW6
  - UART readings from HW7
  - ADC readings from HW7
  - CPRE 288 Datasheet trainer
  - **Mandatory one side of 1 page of notes (you will submit to Canvas as part of Class participation grade)**

Activity

- Intro (3)
- Activity (5)
- Lab Strategies (10)
- Topic Brain storm (10)
- Criteria for judging topic goodness (5)
- Review Rubric (10)
- Start topic selection / Plan for completing topic selection.

Looking Forward

- There are generally three phases of the course:
  1. C Programming targeting low-level embedded concepts
  2. I/O Device Programming
  3. Architecture and Assembly programming

- The 2nd and 3rd phases are much more challenging than the 1st phase

Textbook & Data Sheet: Read and ask questions

- Exam 2 will predominantly consist of questions of the form
  - **Program Configure Registers to meet these specs**
    - **UART, ADC, Input Capture, Output Compare, Timers, Interrupts**
    - **Each device has a section in the Datasheet and Textbook**
  - **Based on a given configuration, answer questions about how the program will behave**
    - **E.g. How long will something take to occur?**
    - **E.g. How many time a second with something occur?**
  - **Explain why a given configuration is incorrect for implementing a specified behavior**
  - **Assuming a given configuration, write a short program to implement a specific behavior**
  - **ADC calculation problem**

Overview

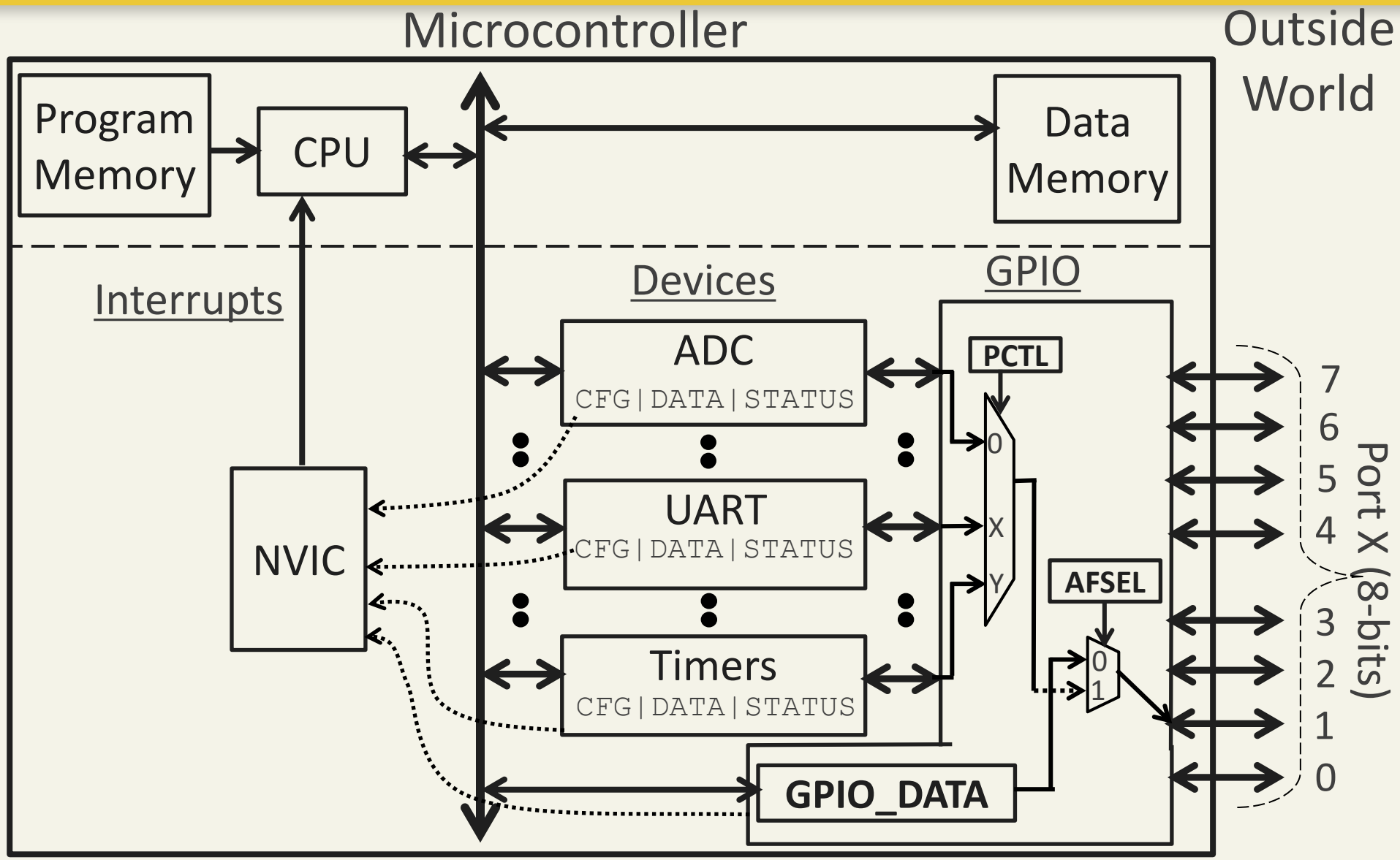
- ADC general knowledge
- TM4C123G ADC programming interface
  - Datasheet reading: Chapter 13
  - Supplemental Textbook reading (as needed):
    - Section 7.5
- Write ADC-related functions
  - Initialize/configure ADC
  - Read from ADC
  - ADC interrupt programming

ADC and DAC

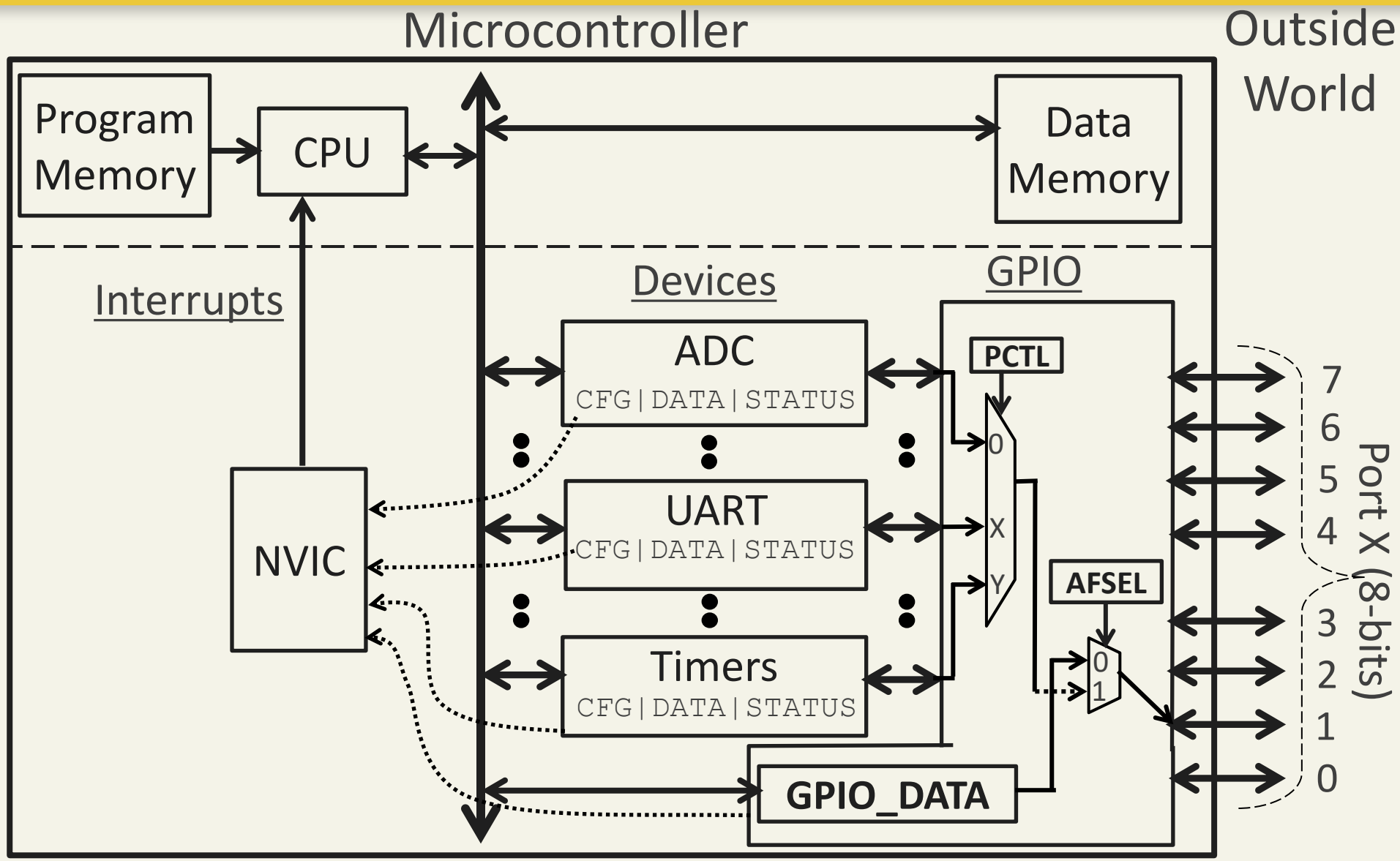
- Analog-to-Digital Conversion (ADC)
- Digital-to-Analog Conversion (DAC)

- Why do we need ADC and DAC?
  - To allow our embedded programs to interact with the World
  - The World is analog, not digital
- Examples of sensors that connect to ADCs
  - Temperature, pressure, light, humidity, compass, and sound

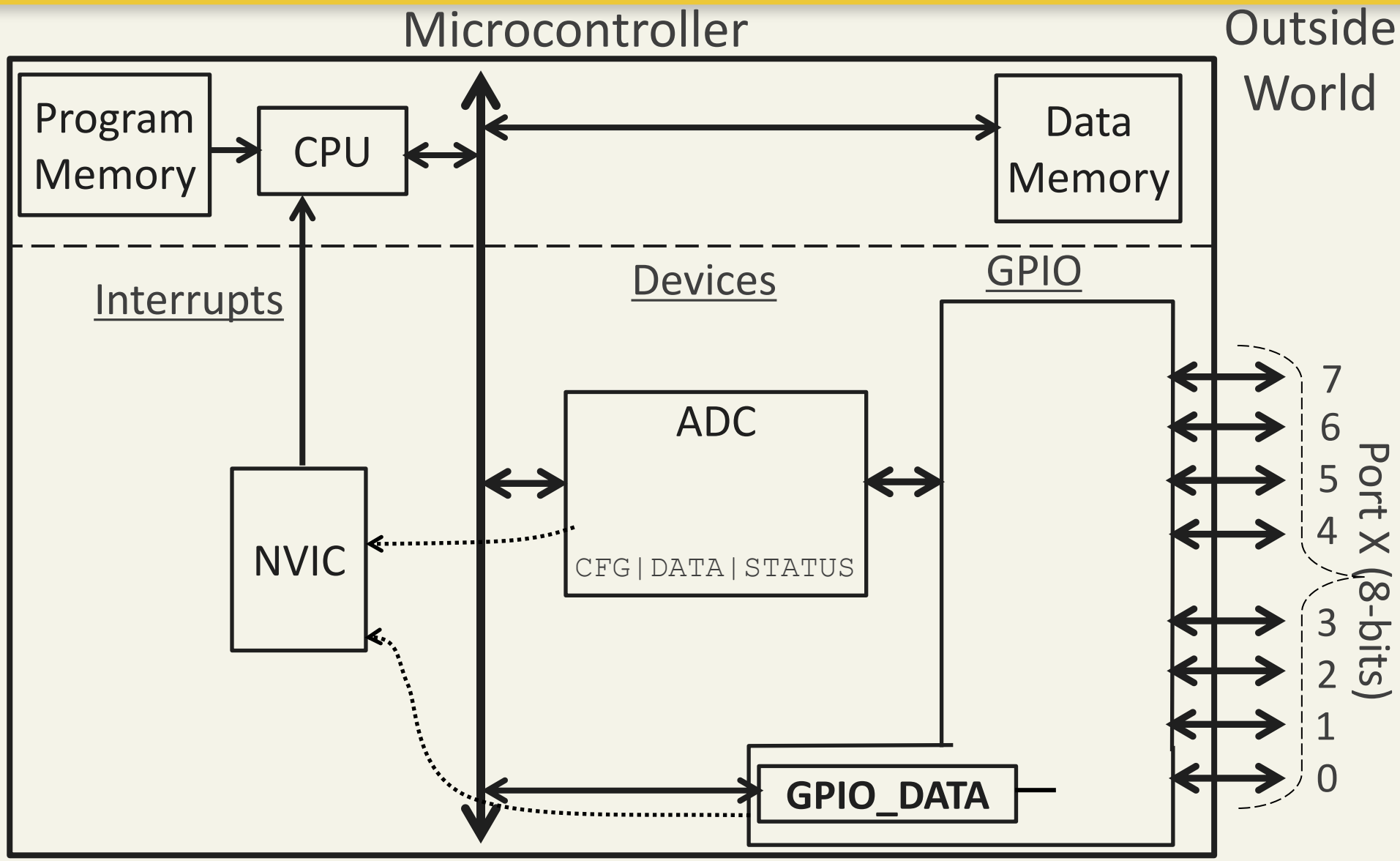
Microcontroller / System-on-Chip (SoC)



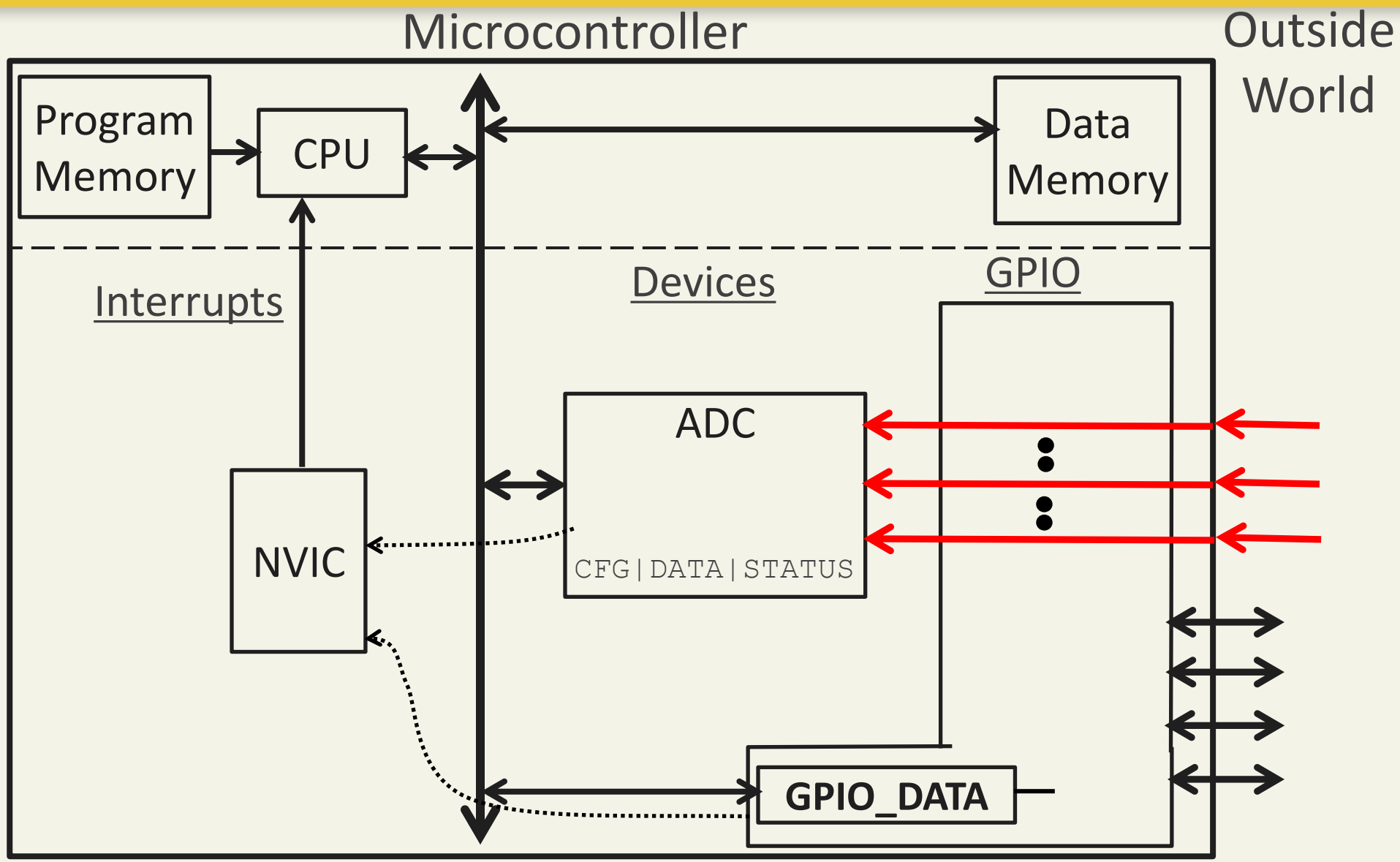
Microcontroller / System-on-Chip (SoC)



Microcontroller / System-on-Chip (SoC)

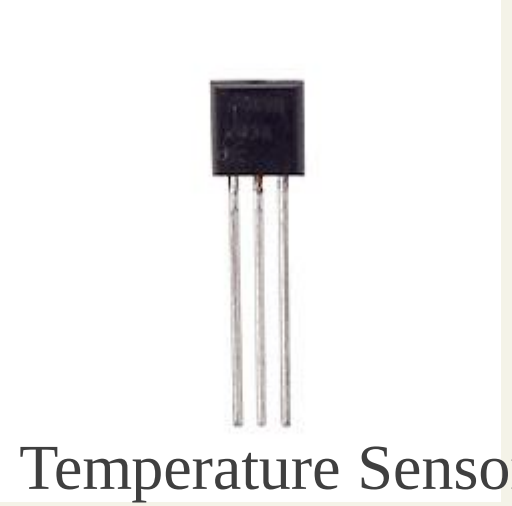


Microcontroller / System-on-Chip (SoC)



ADC and DAC

Analog sensor: Converts a physical signal into an ***analog*** electrical signal



Temperature Sensor $3.95

Photoresistor $1.95 (Light Sensor)

Passive IR sensor $9.95 for motion detection

Sensor pictures and prices from http://www.parallax.com

Terminology

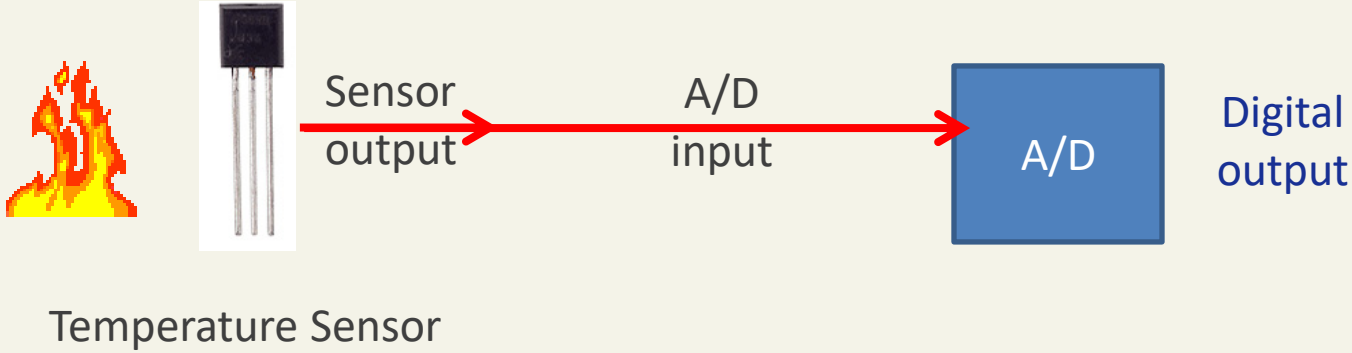
- **analog**: continuously valued signal, such as temperature, speed, or voltage with infinite possible values in between
  - E.g, between 1 m/s and 2 m/s you can have 1.34……2 m/s.
- **digital**: discretely valued signal, such as integers encoded in binary
  - E.g. a 2-bit integer can only have four values- 00, 01, 10, 11

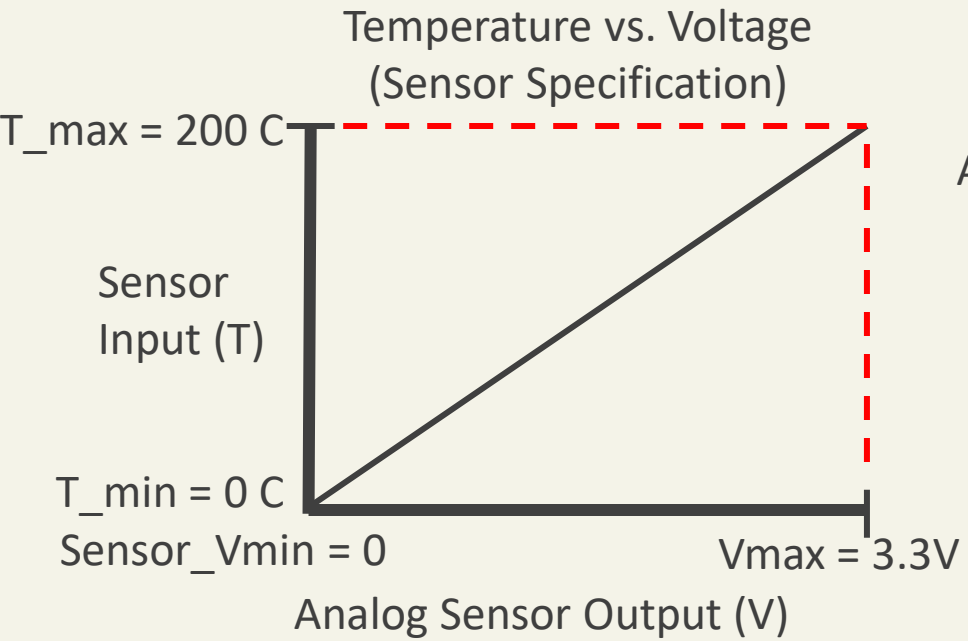
- **analog-to-digital converter** (ADC, A/D, or A2D): converts an analog input signal to a n-bit digital output signal
  - The TM4C123G has a 12-bit ADC
- **digital-to-analog converter** (DAC, D/A, D2A): converts a n-bit digital input signal to an analog output signal

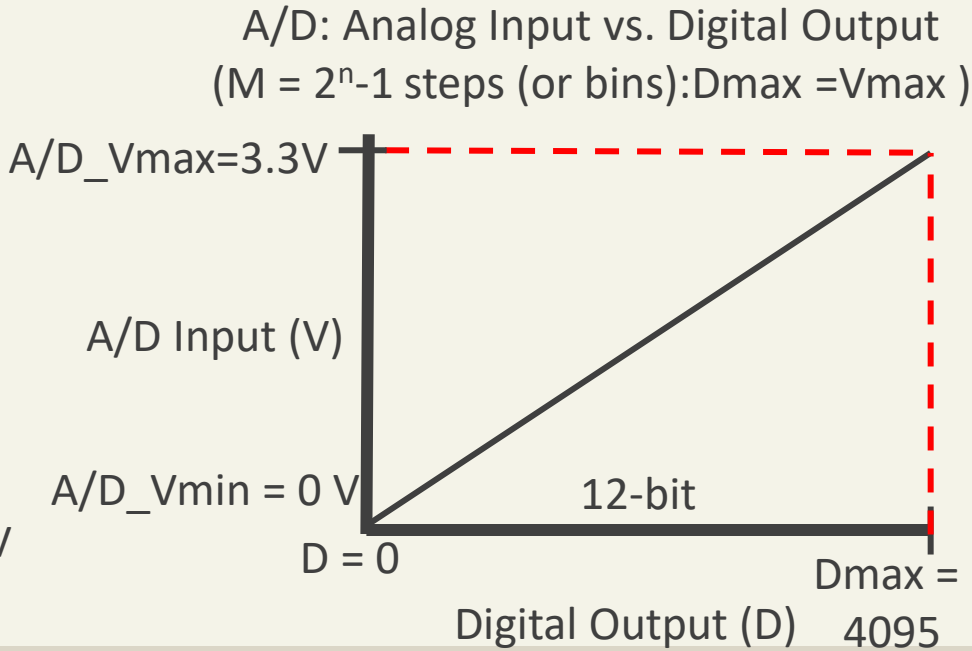
Terminology (cont.)

- **Span** (or **Range**): difference between maximum and minimum analog values (Max – Min)
- **n**: number of bits used for a digital input (DAC) or digital output (ADC) (sometimes referred to as n-bit resolution)
- **Bit Weight**: analog value corresponding to a bit position in a digital number
- **M**: Number of digital steps, either 2^n-1 or 2^n
- **Step Size** (or **Resolution**): smallest analog change resulting from a change of one in a digital value; also the bit weight of the Least Significant Bit (LSB)
  - Step Size (or Resolution) = Span / M
- **Sensitivity**: Amount sensor output changes for a change in sensor input

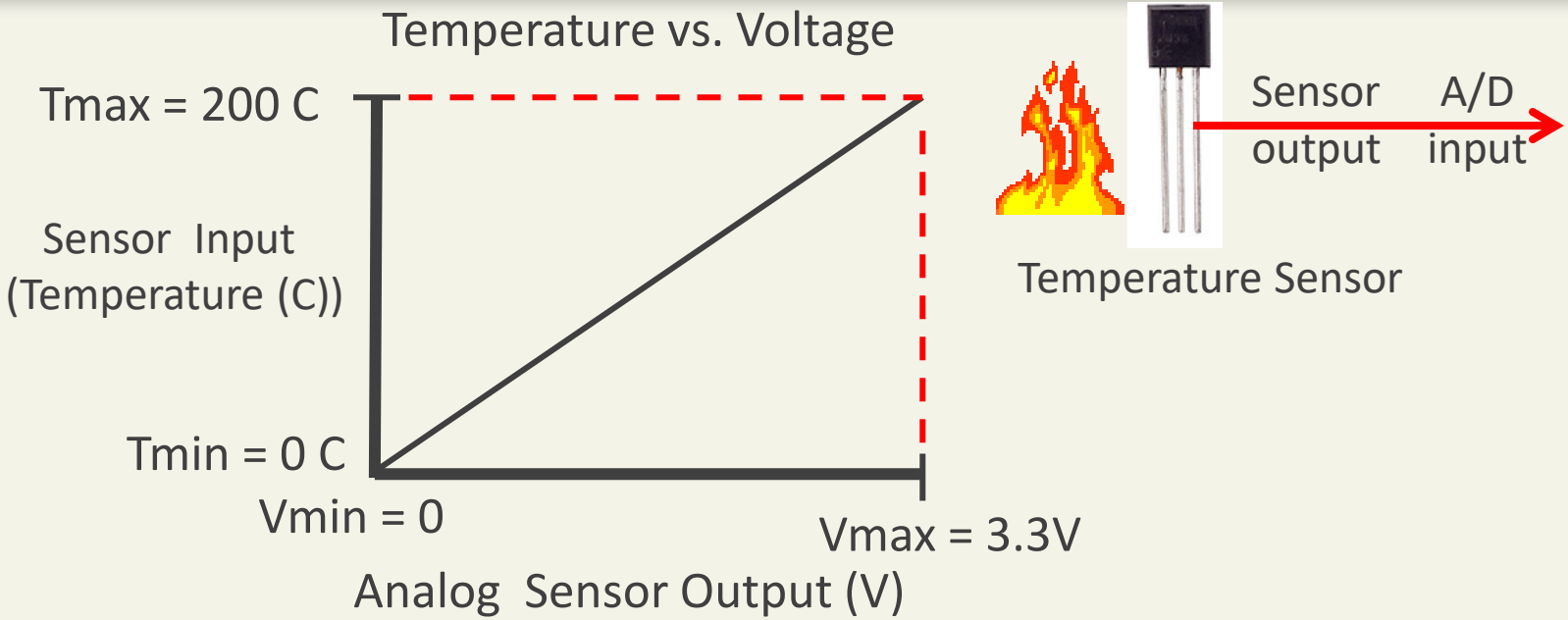
Analog-to-digital converter: Example





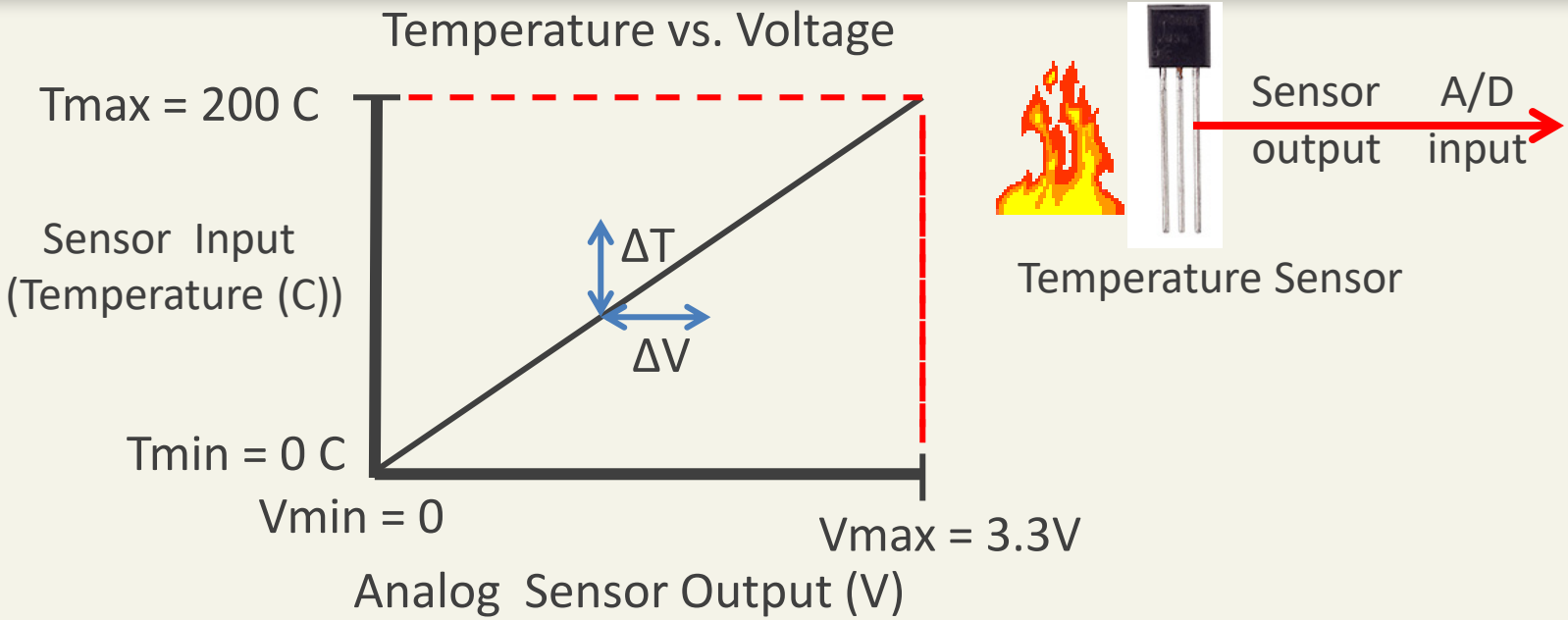


Sensitivity: Analog Sensor (Linear)



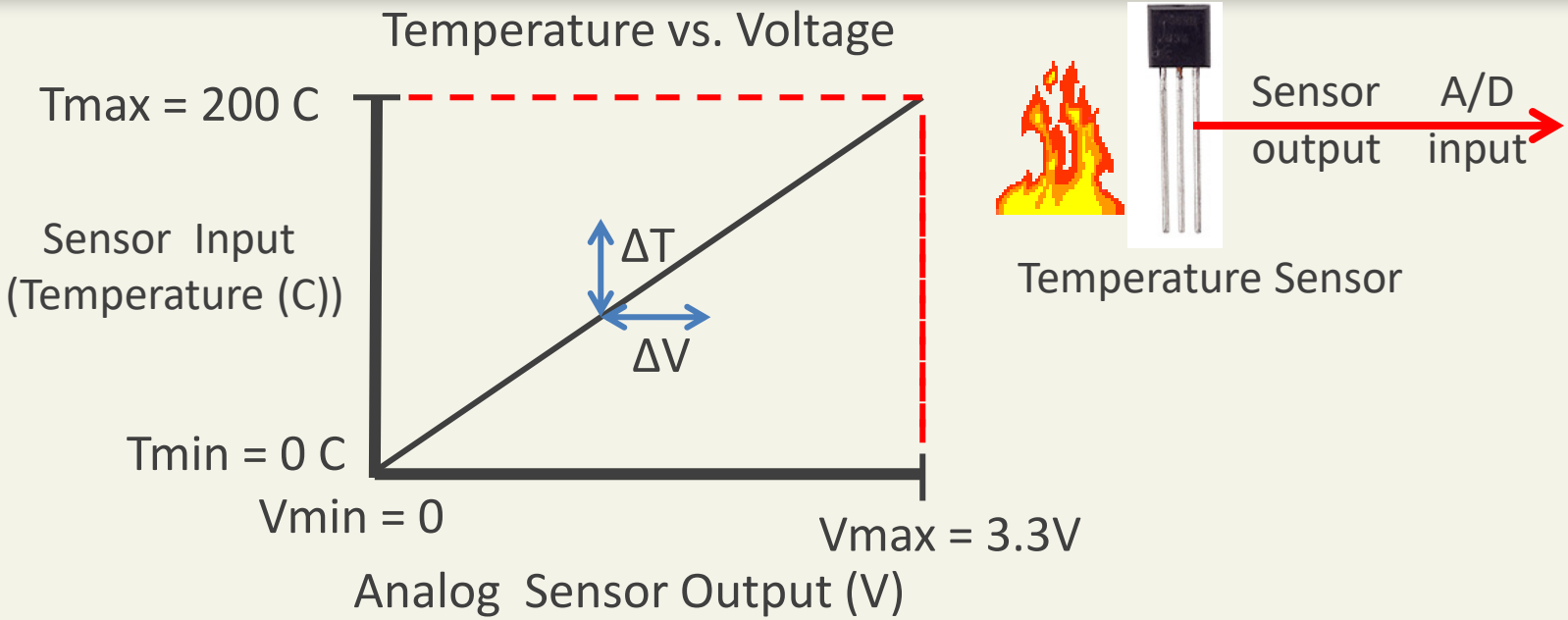
- Sensitivity: How much does a change in a sensor input change the sensor output

Sensitivity: Analog Sensor (Linear)



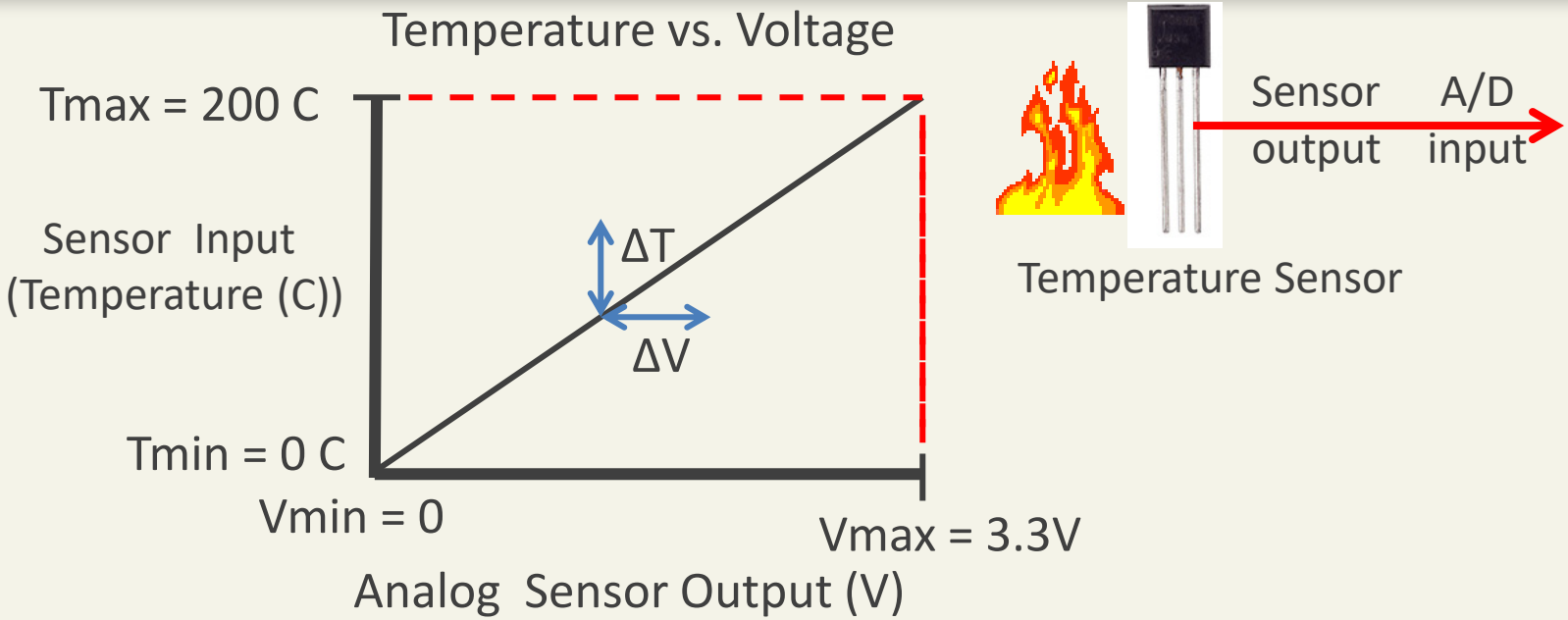
- Sensitivity: How much does a change in a sensor input change the sensor output

Sensitivity: Analog Sensor (Linear)



- Sensitivity: How much does a change in a sensor input change the sensor output
- Assuming a linear sensor (Not all sensors are linear!!)
  - It is just the slop (m) of the Input vs. Output specification
  - Remember slope = RISE/RUN = $\Delta T/\Delta V$

Sensitivity: Analog Sensor (Linear)

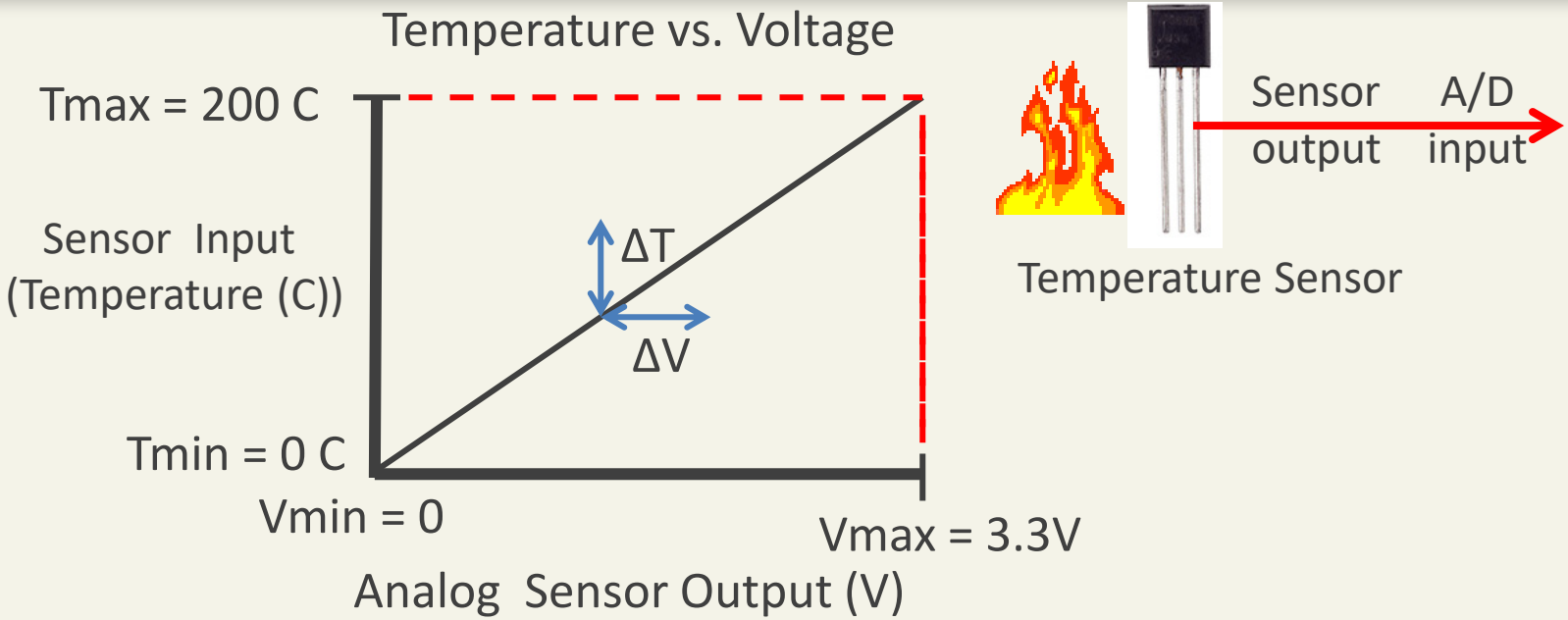


- Sensitivity: How much does a change in a sensor input change the sensor output
- For this example:

m = slope = RISE/RUN = (Tmax – Tmin) / (Vmax –Vmin)

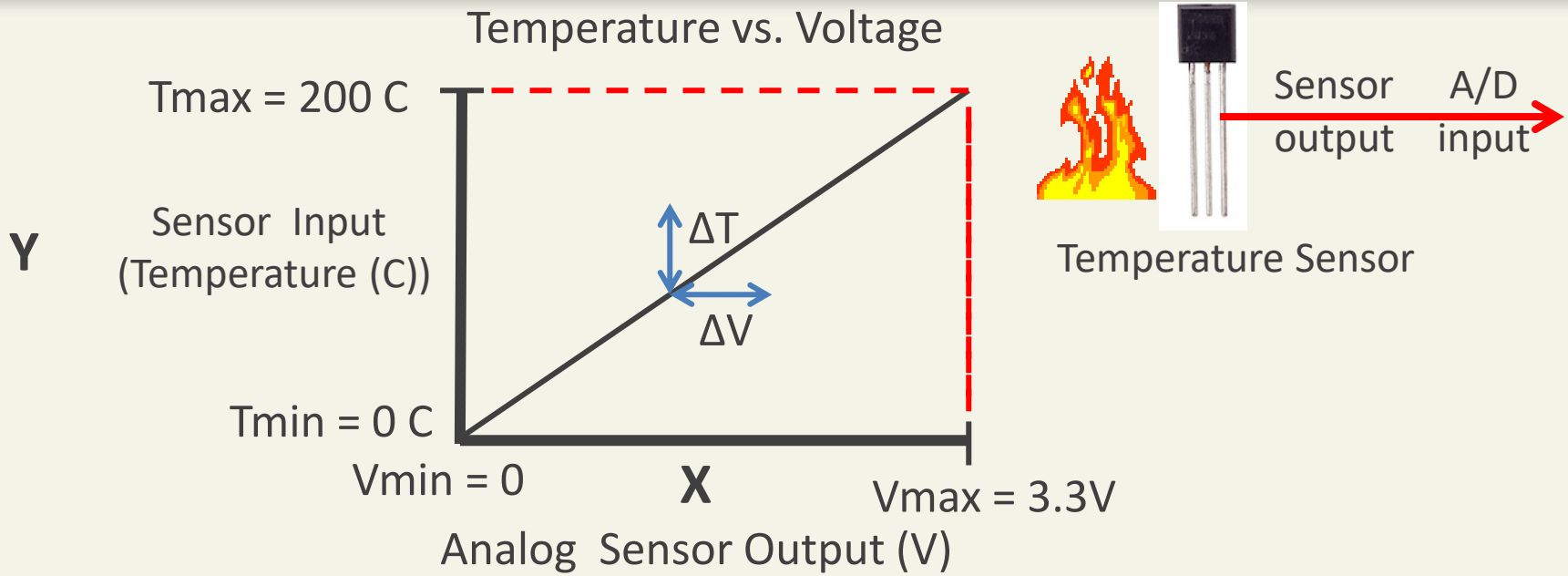
= (200 – 0) / (3.3 – 0) = 60.61 C/V

Analog Sensor: Compute Analog Output



- Question: If the measured Temperature is 100C what is the Analog output of the sensor?
- Again note, in this case the sensor is linear
  - Hint: What is the equation of a line (in y-intercept form, Yes Algebra II was actually an important course)

Analog Sensor: Compute Analog Output



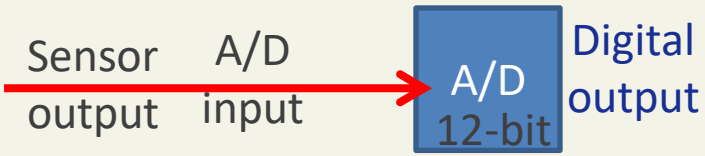
- Question: If the measured Temperature is 100C what is the Analog output of the sensor?
- Again note, in this case the sensor is linear

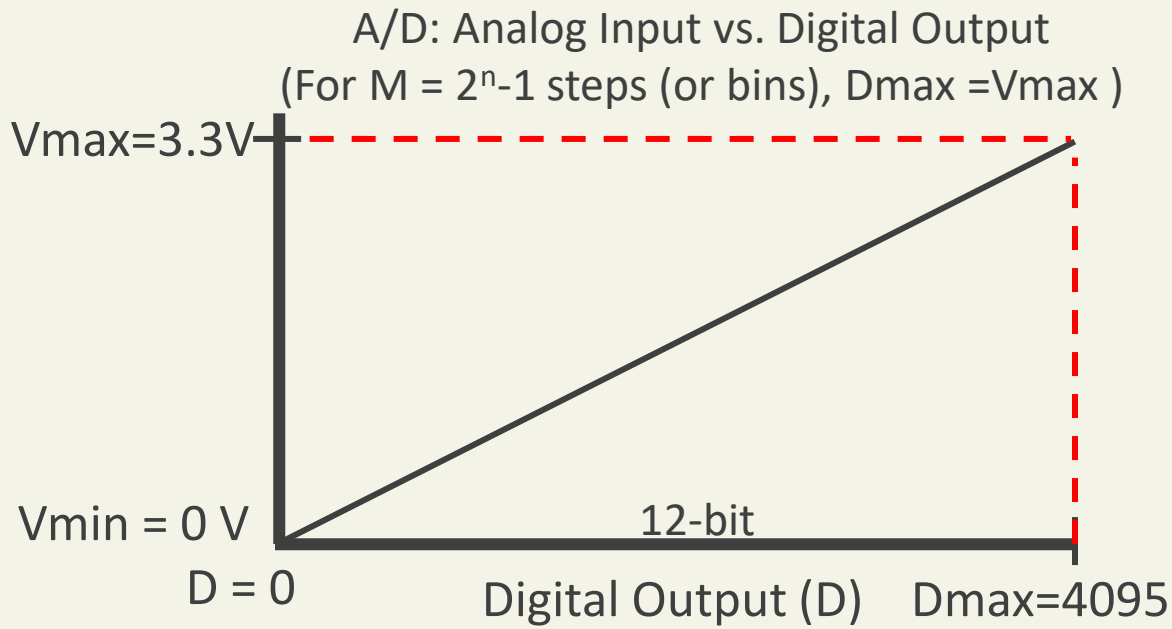
Y = mX + b; in this case the y-intercept b = 0 (!!This is not always the case!!), so

Y = mX; We are given the temperature (Y) = 100 and computed m = 60.61, so

100 = 60.61X; X = 100/60.61 = 1.65 V

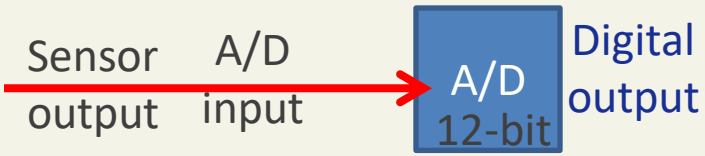
Resolution: A/D

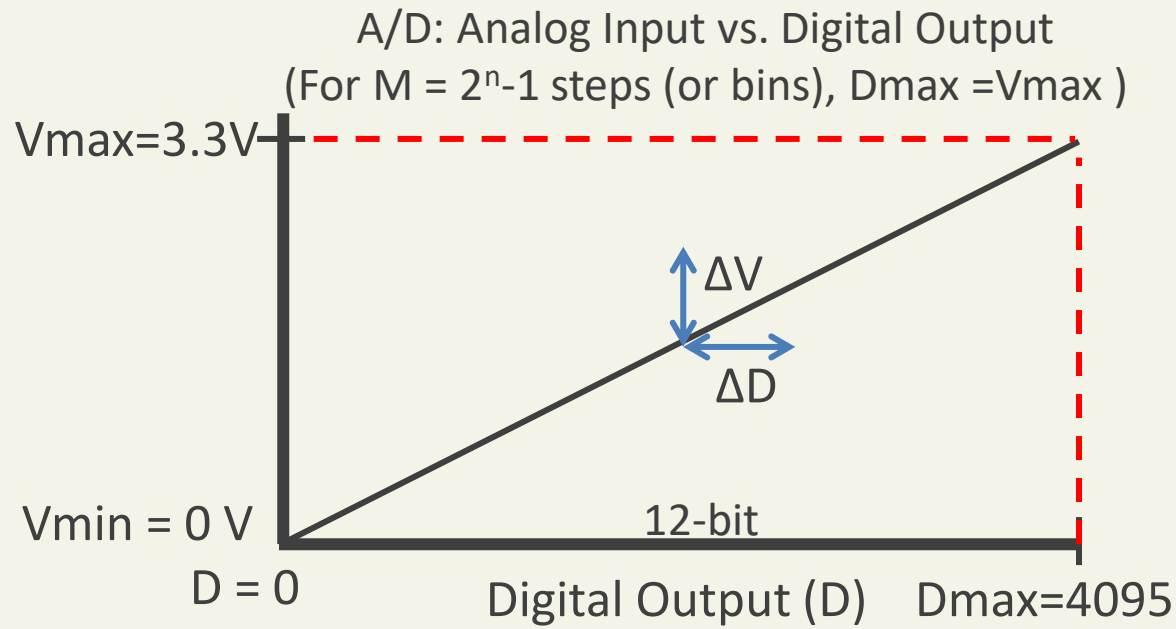




- Resolution: Similar to the concept of Sensitivity for an analog sensor. For a change by 1 of the A/D digital output what size change is detected in the A/D analog input.

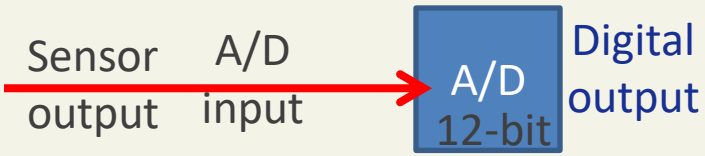
Resolution: A/D

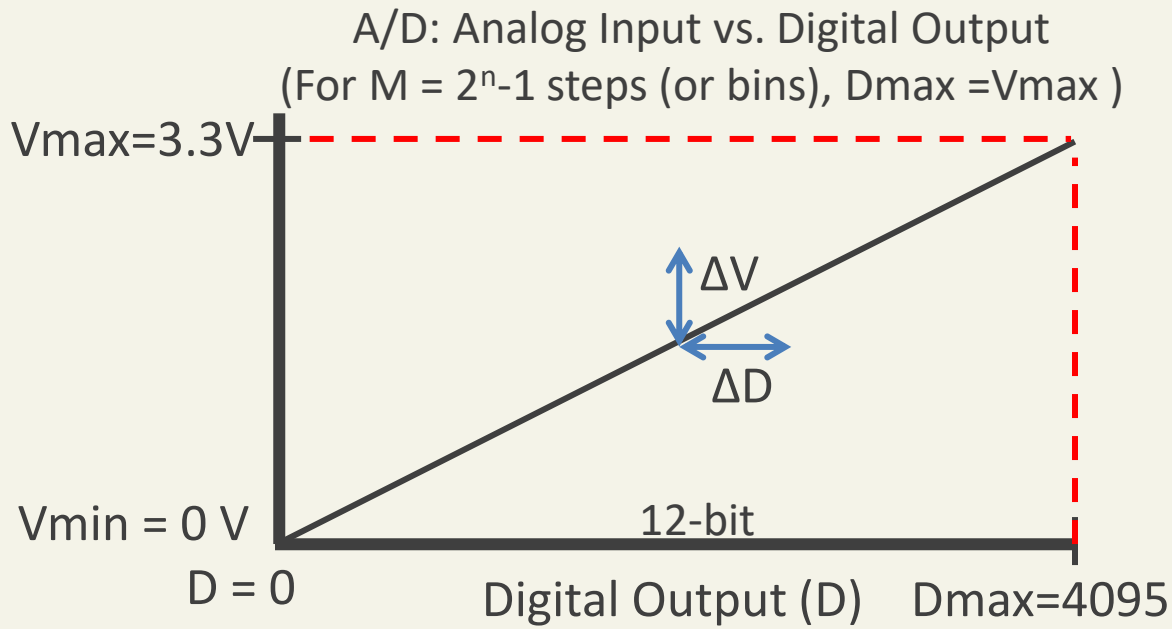




- Resolution: Similar to the concept of Sensitivity for an analog sensor. For a change by 1 of the A/D digital output what size change is detected in the A/D analog input.
- A/D converters typically have a linear relationship between their Analog input and Digital output
  - Resolution is just the slope (m) of the Input vs. Output specification

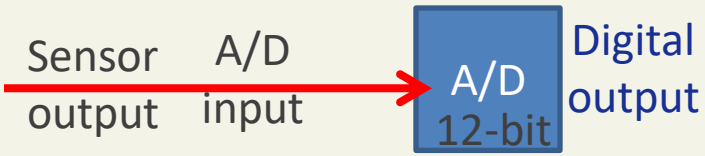
Resolution: A/D

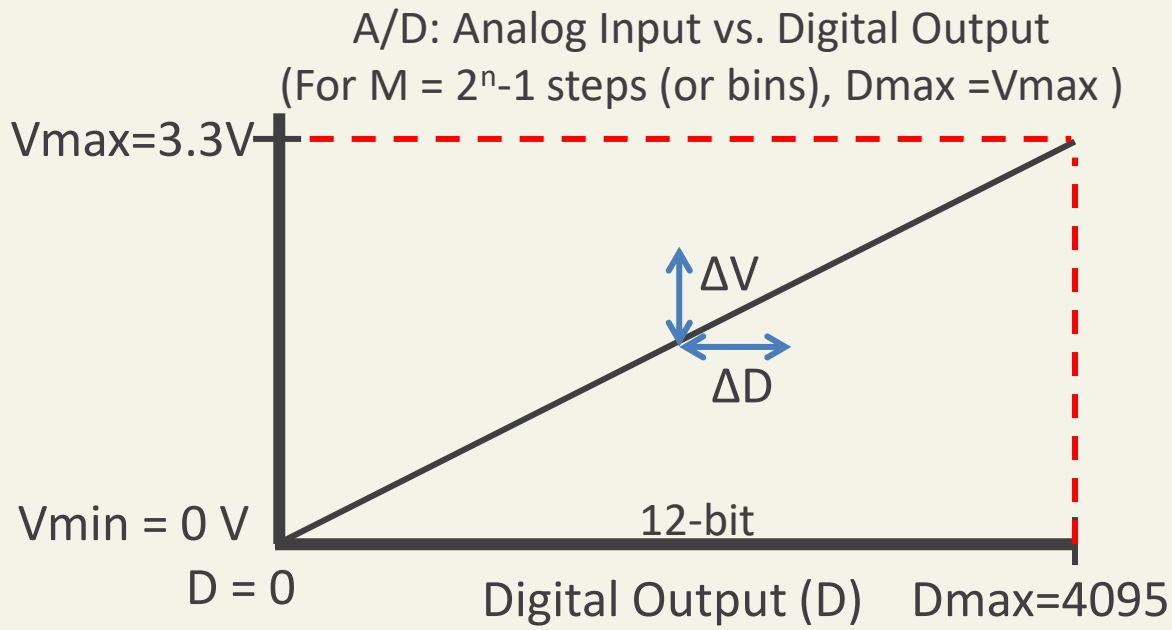




- A/D converters typically have a linear relationship between their Analog input and Digital output
  - Resolution is just the slope of the Input vs. Output specification
- A/D specifications typically specify this in terms of a the Least Significant Bit (LSB) weight

Resolution: A/D



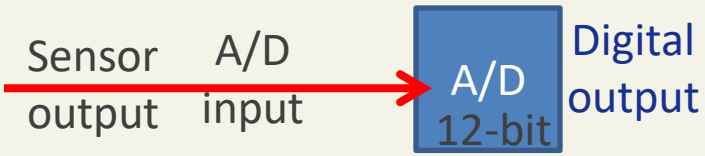


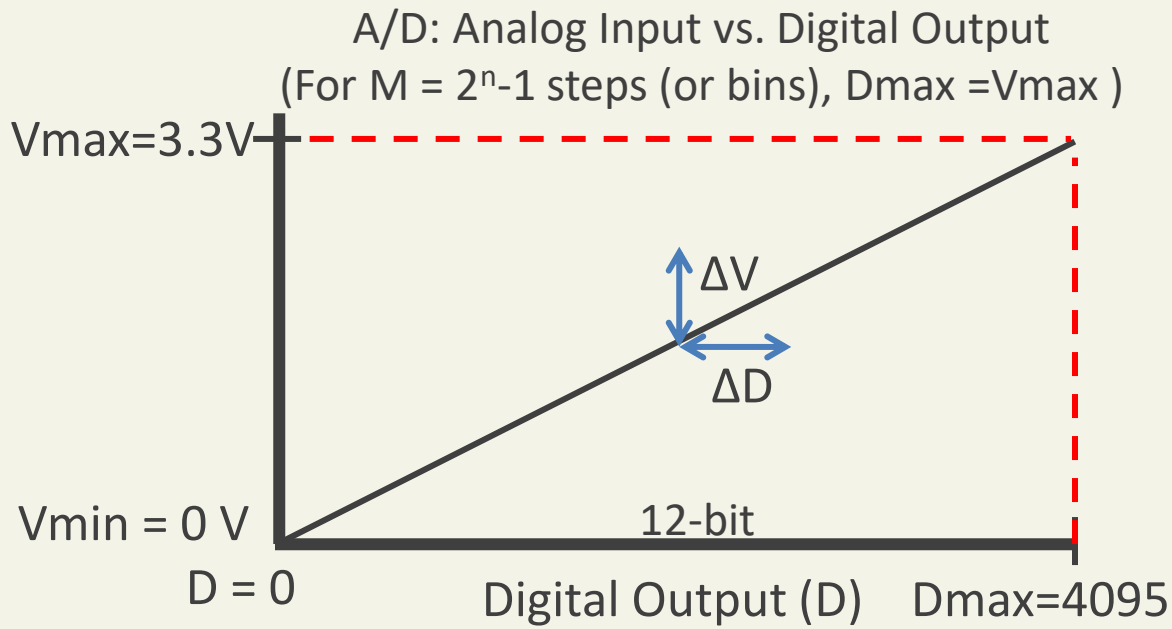
- For this example:

Resolution= slope= RISE/RUN= (3.3 – 0 / 4095 – 0) = .000805V/bit

LSB bit weight = .000805V/bit

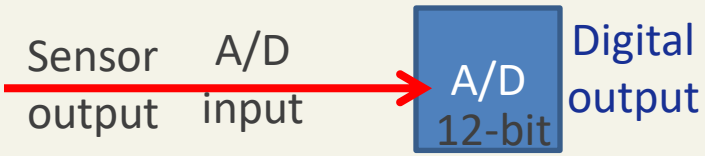
A/D: Compute Digital output

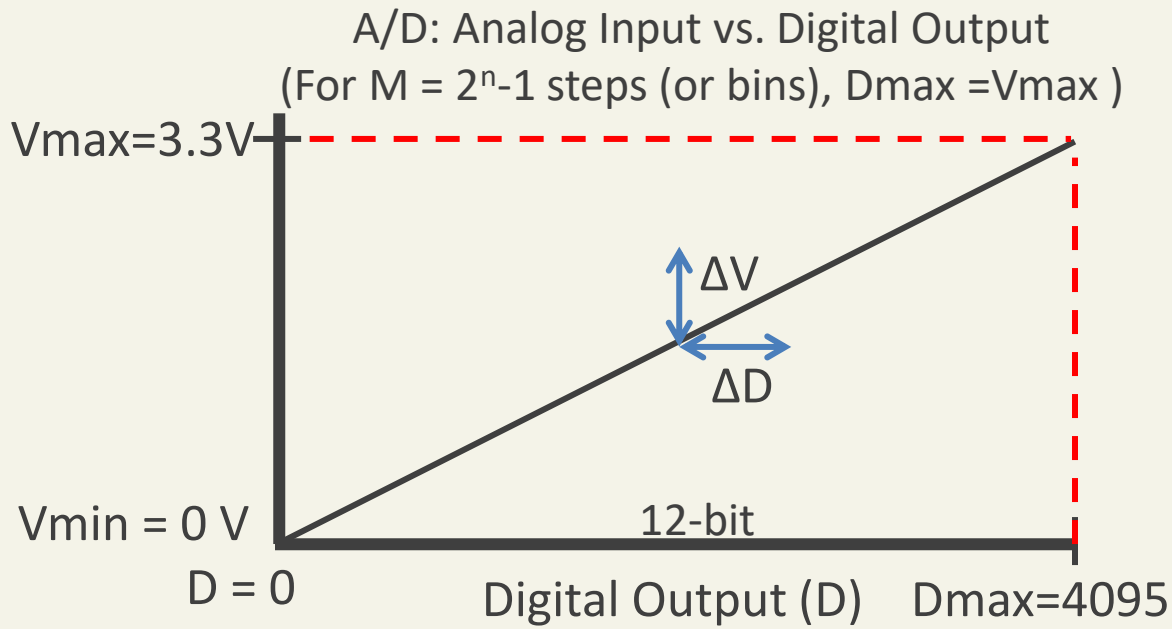




- Question: If the input is 1.65 V what is the Digital output of the A/D?
- Again note, A/D converters are typically linear
  - Hint 1: What is the equation of a line

A/D: Compute Digital output





- Question: If the input is 1.65 V what is the Digital output of the A/D?
- Again note, A/D converters are typically linear

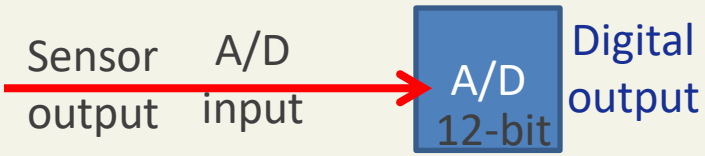
Y = mX + b; in this case the y-intercept b = 0, so

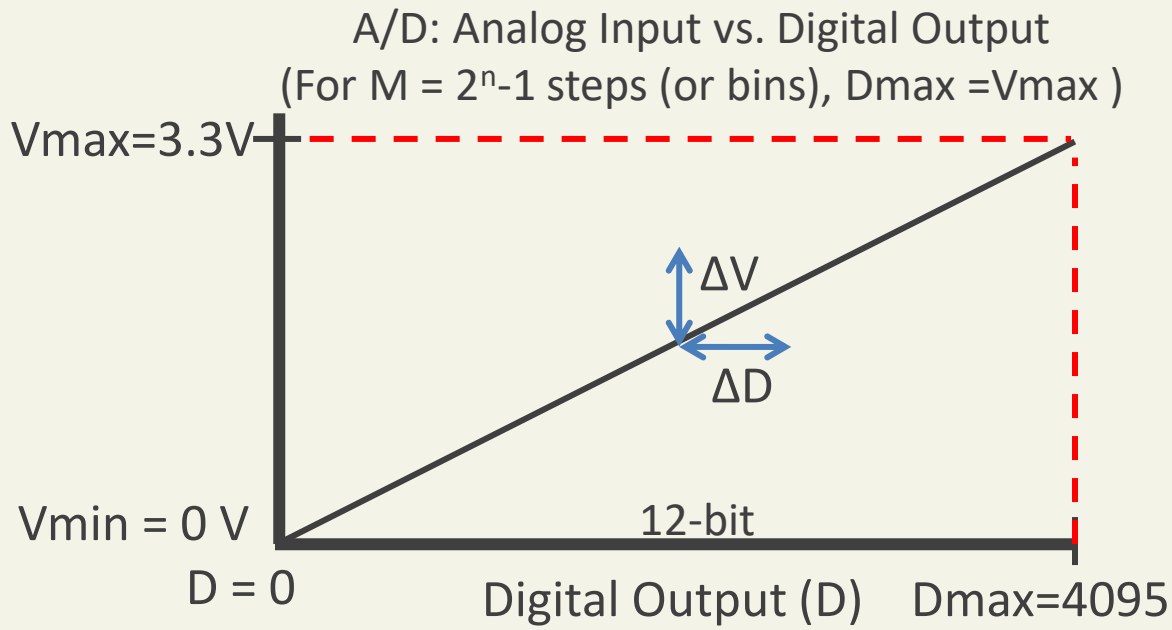
Y = mX; We are given the voltage (Y) is 1.65V and computed m = .000805, so

1.65 = .000805X; X = 1.65/.000805= 2049.68; Truncate to 2049 = 0b1000 0000 0001

For this example, a Temperature of 100 C gives a digital value of 2049

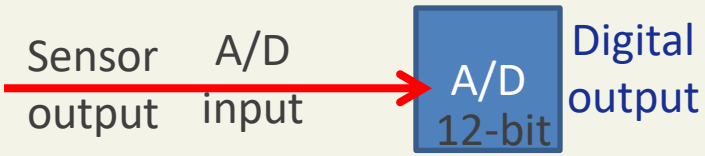
A/D: Compute Digital output

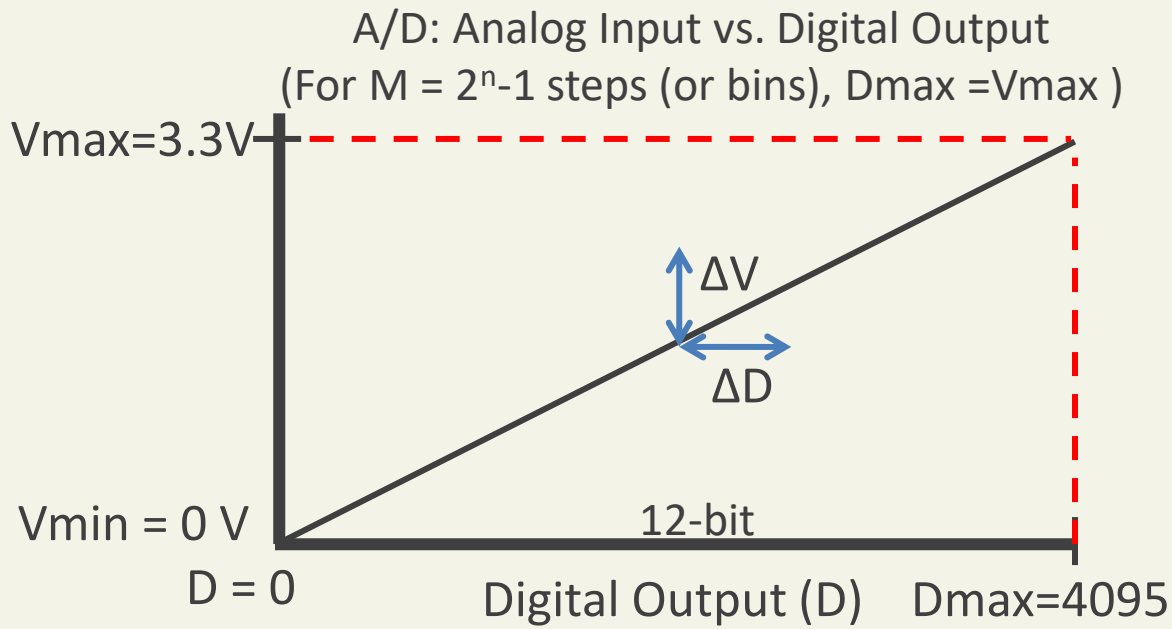




- Question: What is the Temperature LSB weight?

A/D: Compute Digital output





- Question: What is the Temperature LSB weight?
- We know the sensor sensitivity = 60.61 C/ V
- We know the A/D resolution = .000805V/bit
- We want C/bit: So 60.61 C/V * .000805V/bit = .049 C/bit

A change of 1 in the digital output, corresponds to a change .049 degrees

ADC Bit Weight (a closer look)

LSB bit weight in the last example:
bit 0 = .000805V, this is the resolution

Each bit position is weighted with an analog value, such that a 1 in that bit position adds its analog value to the total analog value represented by the digital encoding.

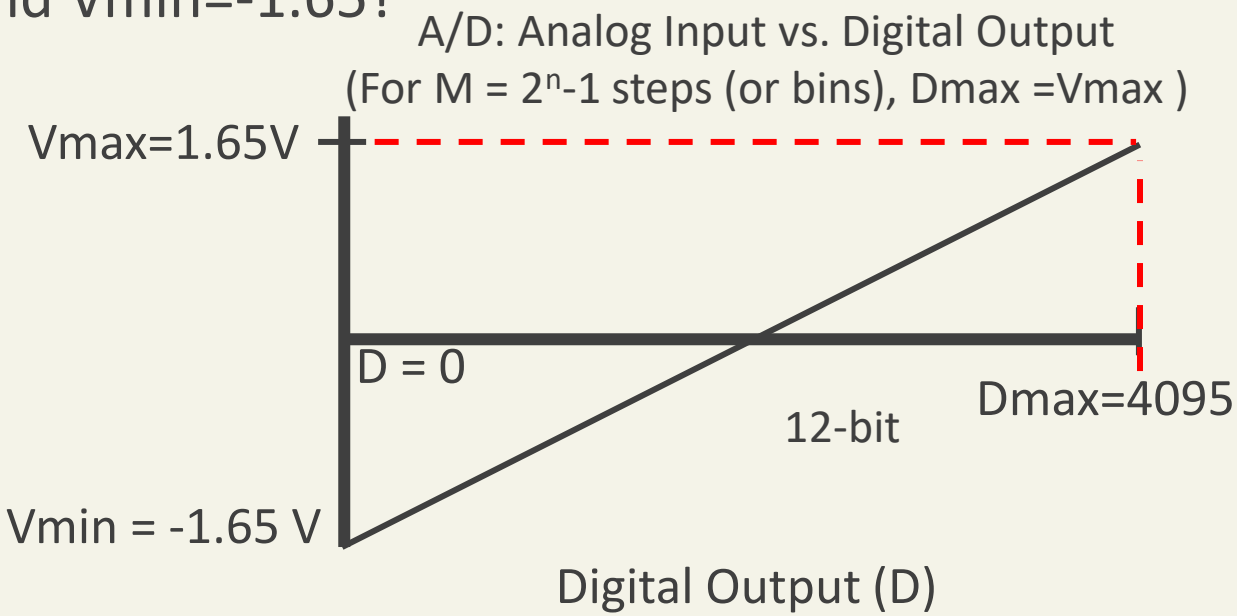
For the previous example:

Decimal: 2049
Binary: 1000 0000 0001
1.648 + .000805 ~= 1.65 V

Digital Bit	Bit Weight (V)
11	2048*r = 1.64864
10	1024*r = 0.82432
9	512*r = 0.41216
8	256*r = 0.20608
7	128*r = 0.10304
6	64*r = 0.05152
5	32*r = 0.02576
4	16*r = 0.01288
3	8*r = 0.00644
2	4*r = 0.00322
1	2*r = 0.00161
0	r = 0.000805

ADC Bit Weight (What if Vmin ≠ 0)

What if Vmax = 1.65, and Vmin=-1.65?



ADC Bit Weight (What if Vmin ≠ 0)

What if Vmax = 1.65, and Vmin=-1.65?

Since the Range stayed the same (3.3V), the resolution is unchanged.

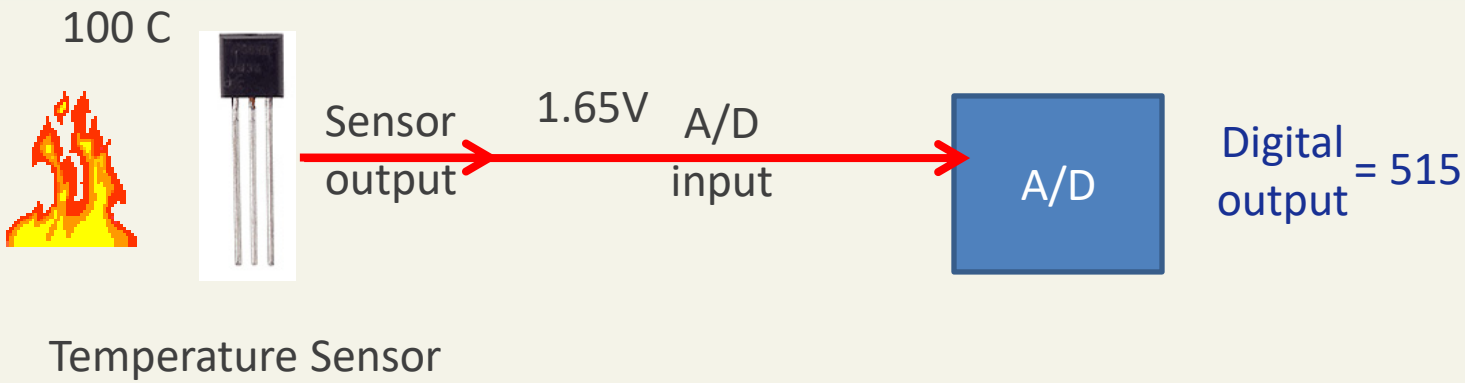
Thus, the LSB bit weight is still:
bit 0 = .000805V, (the resolution)

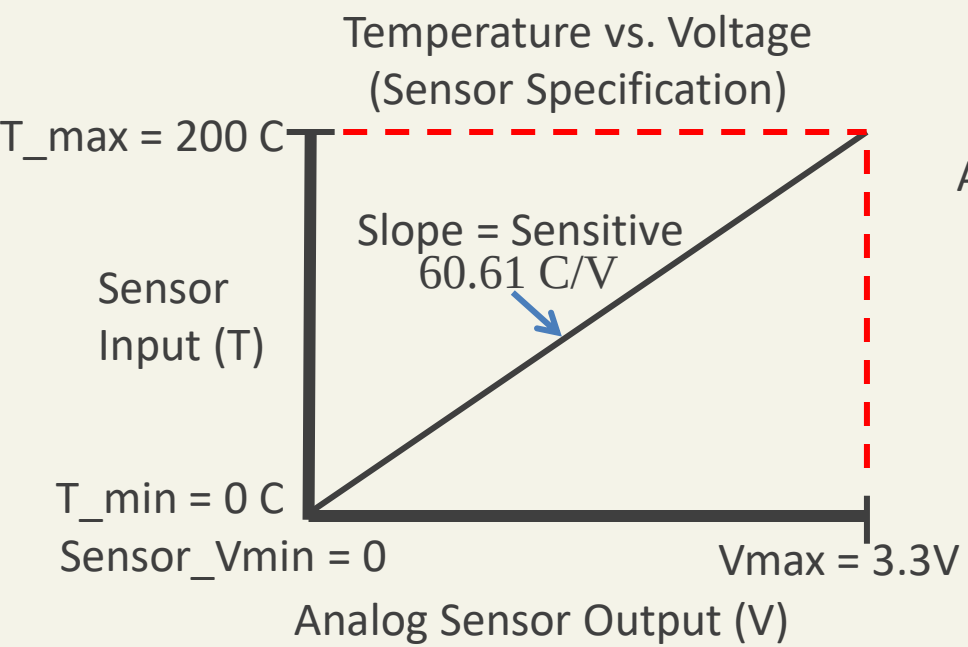
But now you must add an offset of Vmin.
(Can derive from the equation for a line)

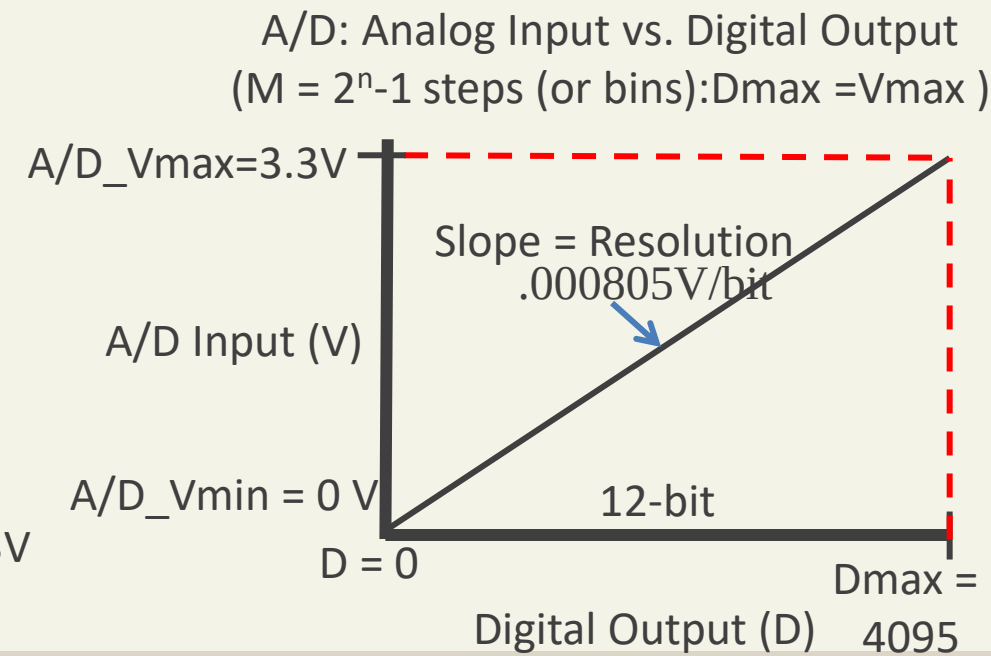
For modified example:
Decimal: 2049
Binary: 1000 0000 0001
1.648 + .000805 ~= 1.65 V
+ -1.65 V
0 V

Digital Bit	Bit Weight (V)
11	2048*r = 1.64864
10	1024*r = 0.82432
9	512*r = 0.41216
8	256*r = 0.20608
7	128*r = 0.10304
6	64*r = 0.05152
5	32*r = 0.02576
4	16*r = 0.01288
3	8*r = 0.00644
2	4*r = 0.00322
1	2*r = 0.00161
0	r = 0.000805

Analog-to-digital converter: Example

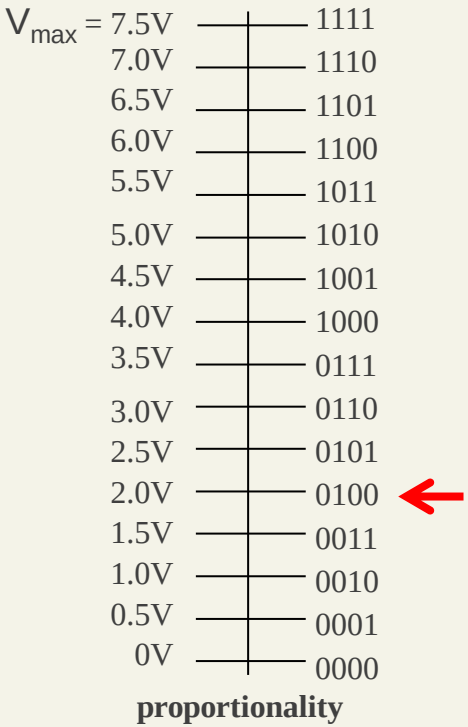


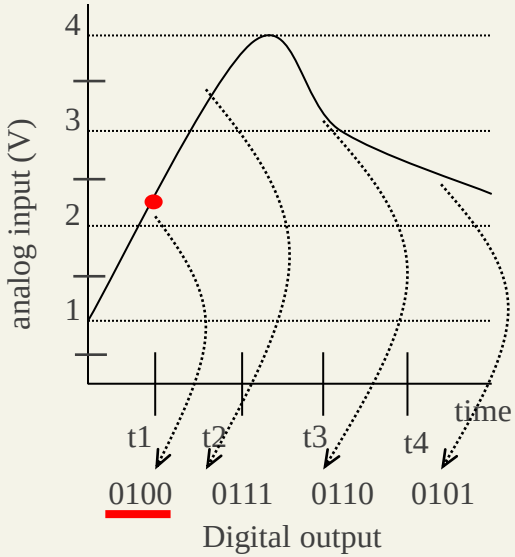




Analog-to-digital converters (Usage)

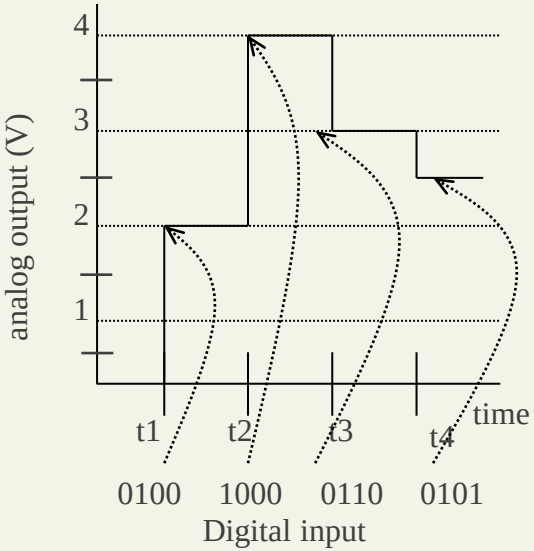
Mapping between
Analog and Digital





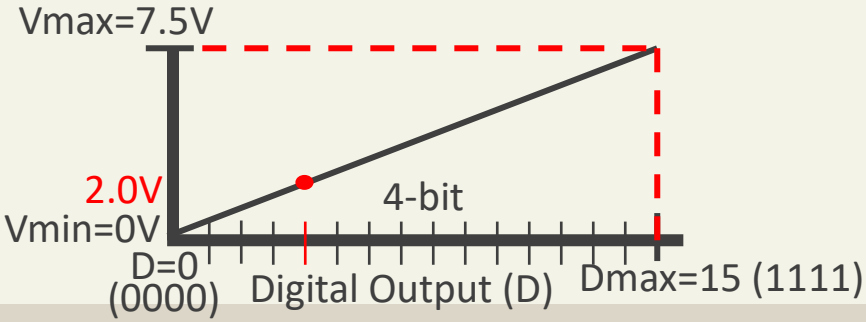
Digital sampling of
an analog signal

analog to digital



Digital generation of
an analog signal

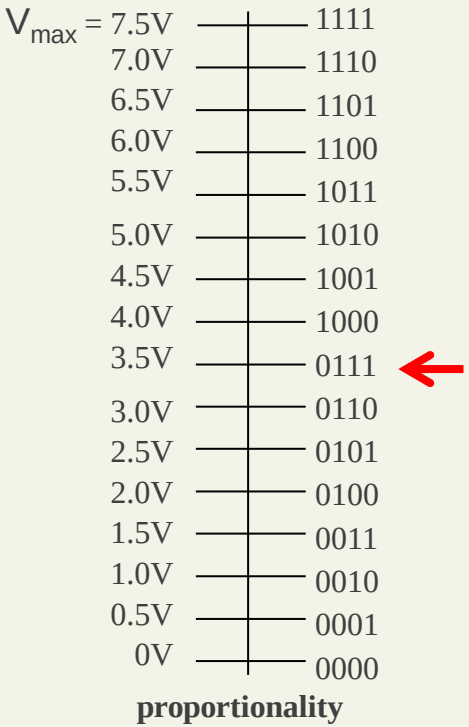
digital to analog

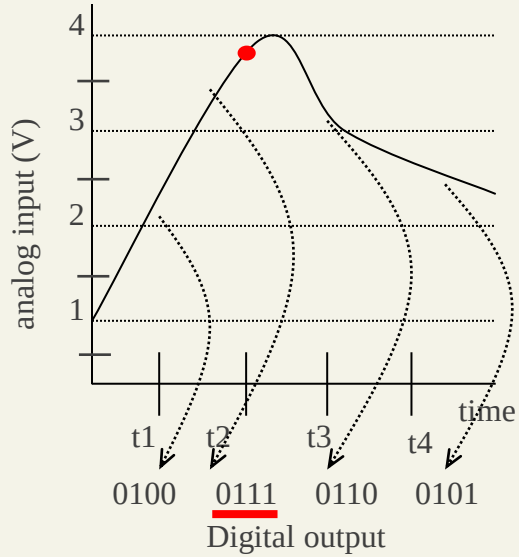


Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

Analog-to-digital converters (Usage)

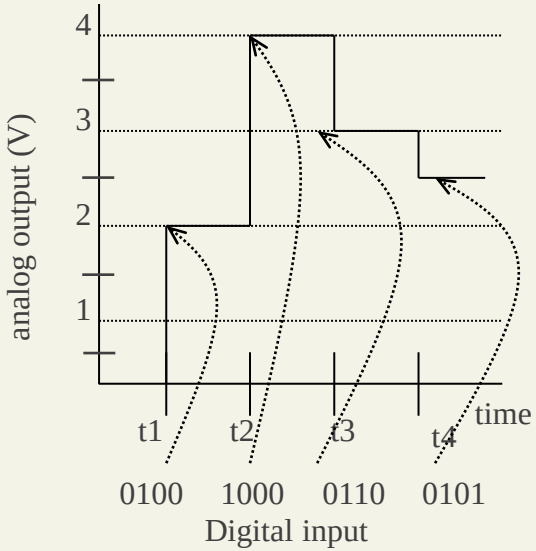
Mapping between
Analog and Digital





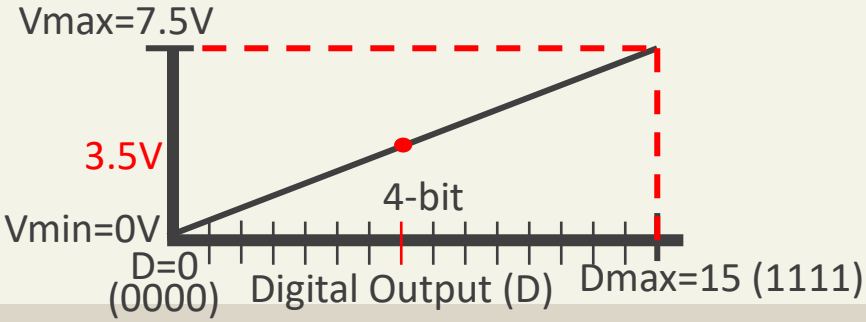
Digital sampling of
an analog signal

analog to digital



Digital generation of
an analog signal

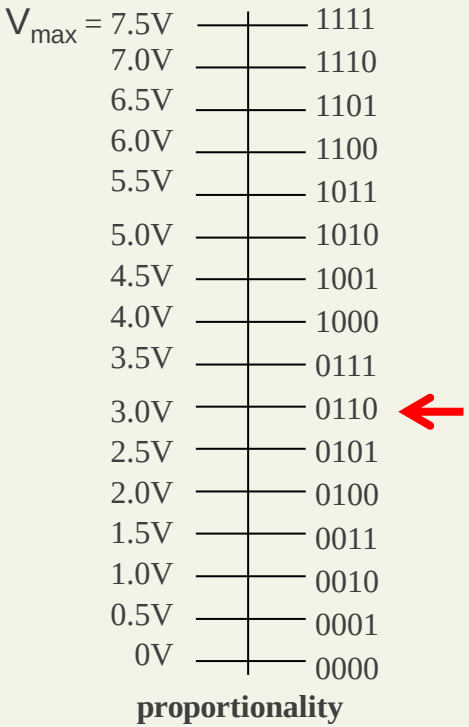
digital to analog

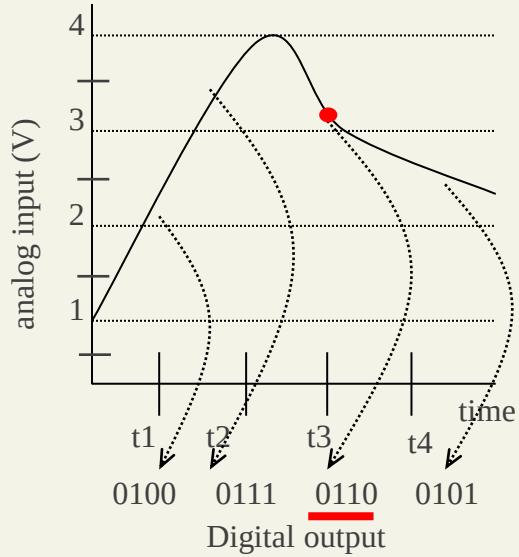


Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

Analog-to-digital converters (Usage)

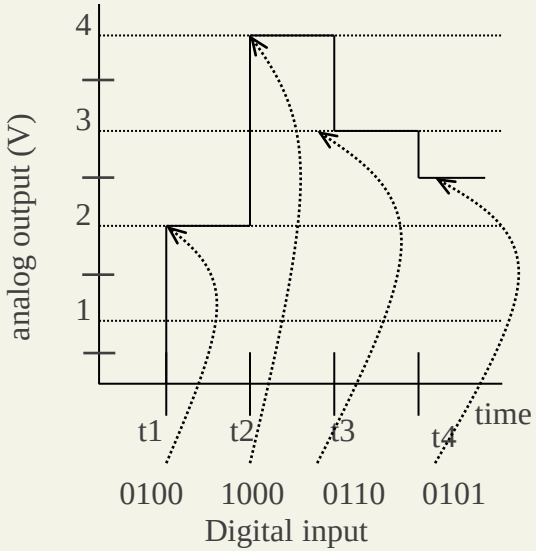
Mapping between
Analog and Digital





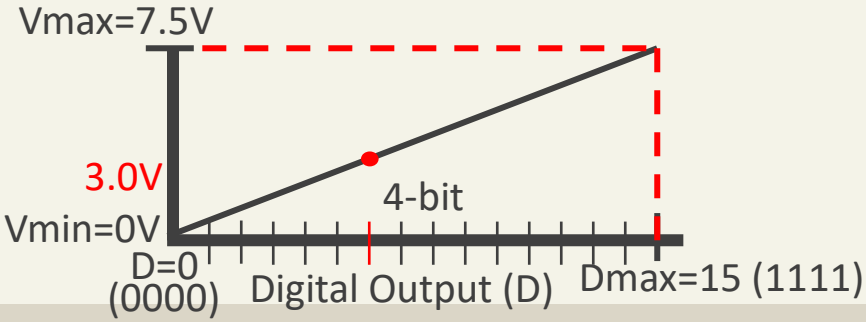
Digital sampling of
an analog signal

analog to digital



Digital generation of
an analog signal

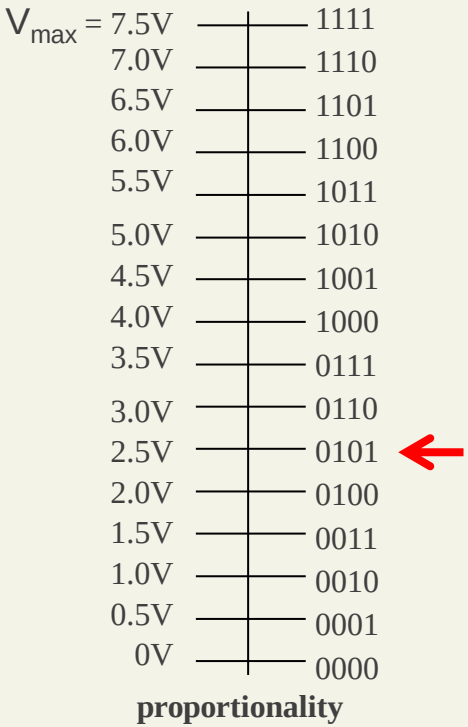
digital to analog

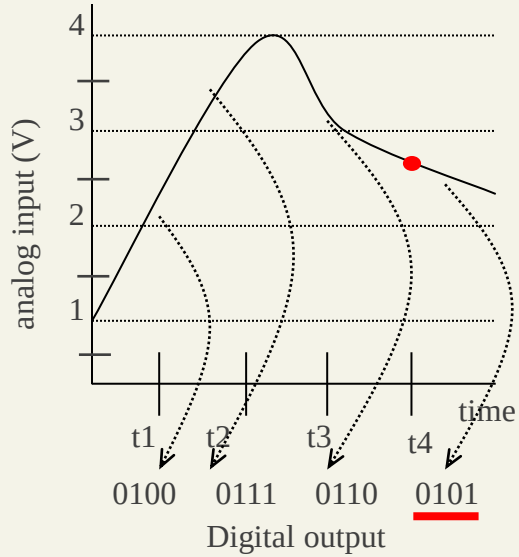


Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

Analog-to-digital converters (Usage)

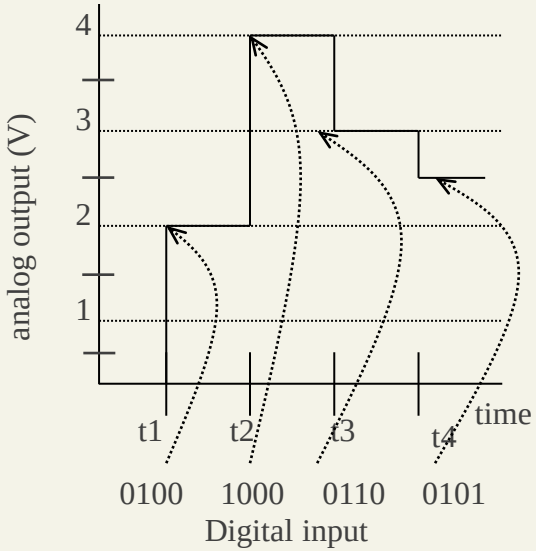
Mapping between
Analog and Digital





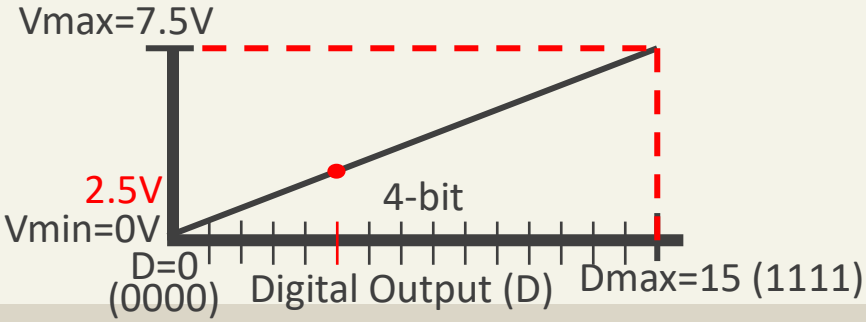
Digital sampling of
an analog signal

analog to digital



Digital generation of
an analog signal

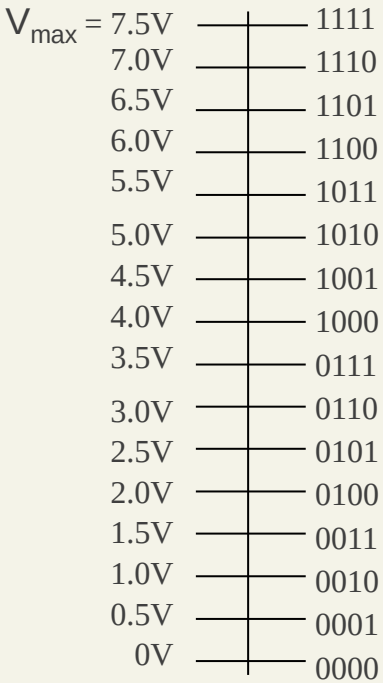
digital to analog



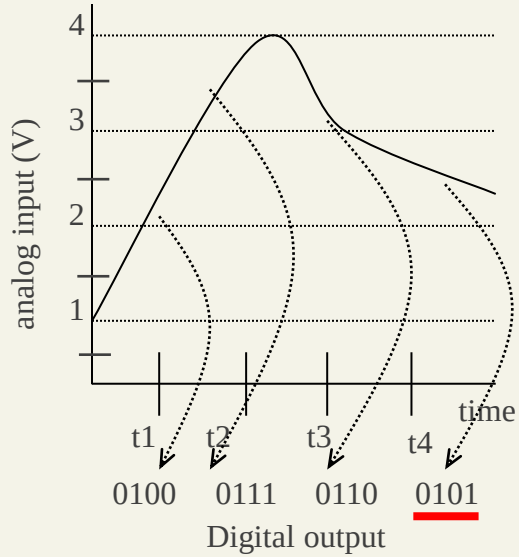
Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

Analog-to-digital converters (Usage)

Mapping between
Analog and Digital

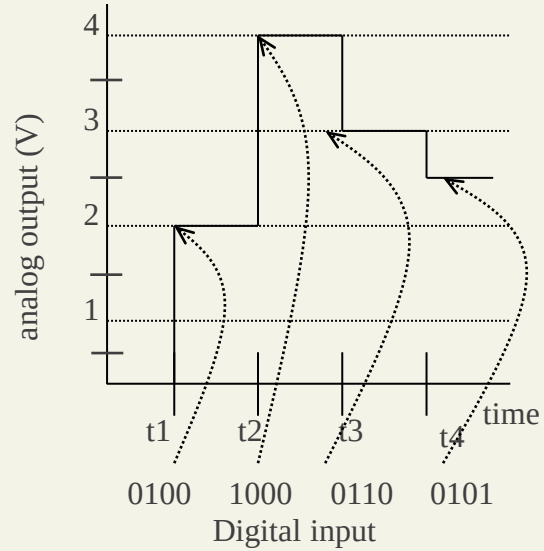


proportionality



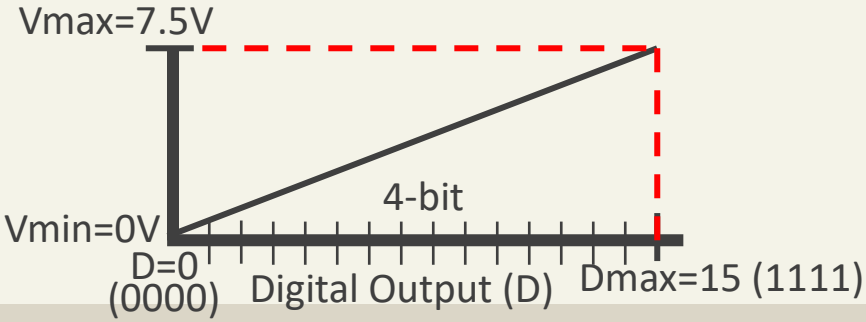
Digital sampling of
an analog signal

analog to digital



Digital generation of
an analog signal

digital to analog



Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

Proportional Signals (Simple case)

Simple Equation

Assume Vmin = 0 V.

Vmax = maximum voltage of the analog signal

a = analog value

n = number of bits for digital encoding

2^n = number of digital codes

M = number of steps, either 2^n or $2^n - 1$

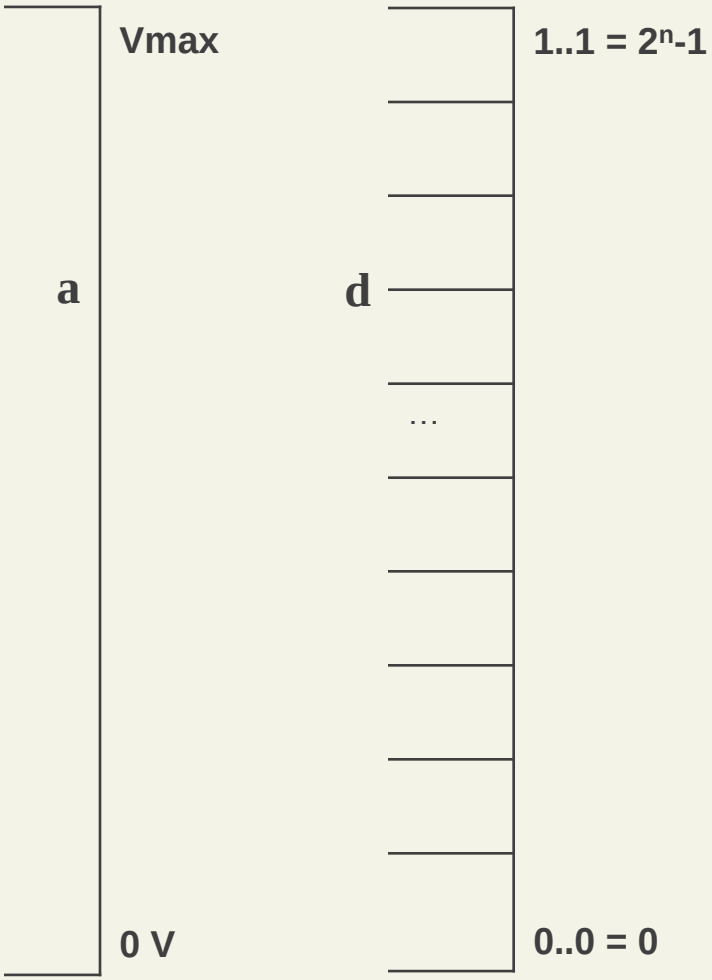
d = digital encoding

$$\boxed{a / Vmax = d / M}$$

This is dervied from the equation for a line

$a = [(Vmax - Vmin)/ (2^n-1 - 0)] * d + 0$

Y = m * X + b



Proportional Signals (General case)

General Equation

Do not assume Vmin = 0 V.

Vmax = maximum voltage of the analog signal

a = analog value

n = number of bits for digital encoding

2^n = number of digital codes

M = number of steps, either 2^n or $2^n - 1$

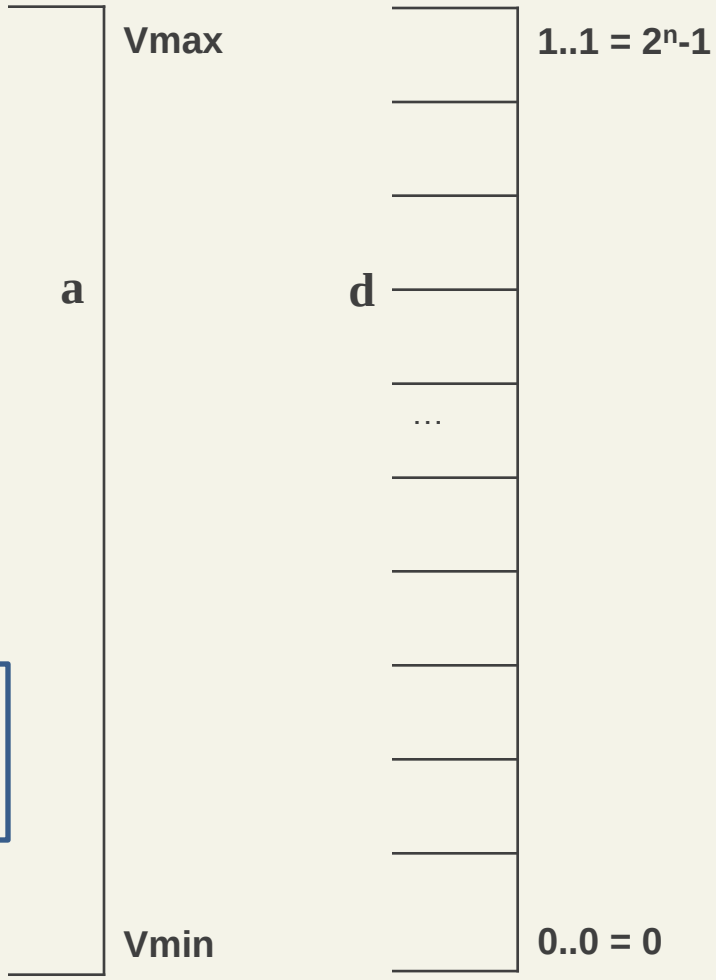
d = digital encoding

$$\boxed{(a-V_{min})/(V_{max}-V_{min}) = d / M}$$

This is derived from the equation for a line

$a = [(V_{max} - V_{min})/ (2^n-1 - 0)] * d + V_{min}$

Y = m * X + b



Resolution: $M = 2^n - 1$ vs. 2^n

Let n = 2

<u>$M = 2^n - 1$</u>

3 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax} = 3 = 0b11$

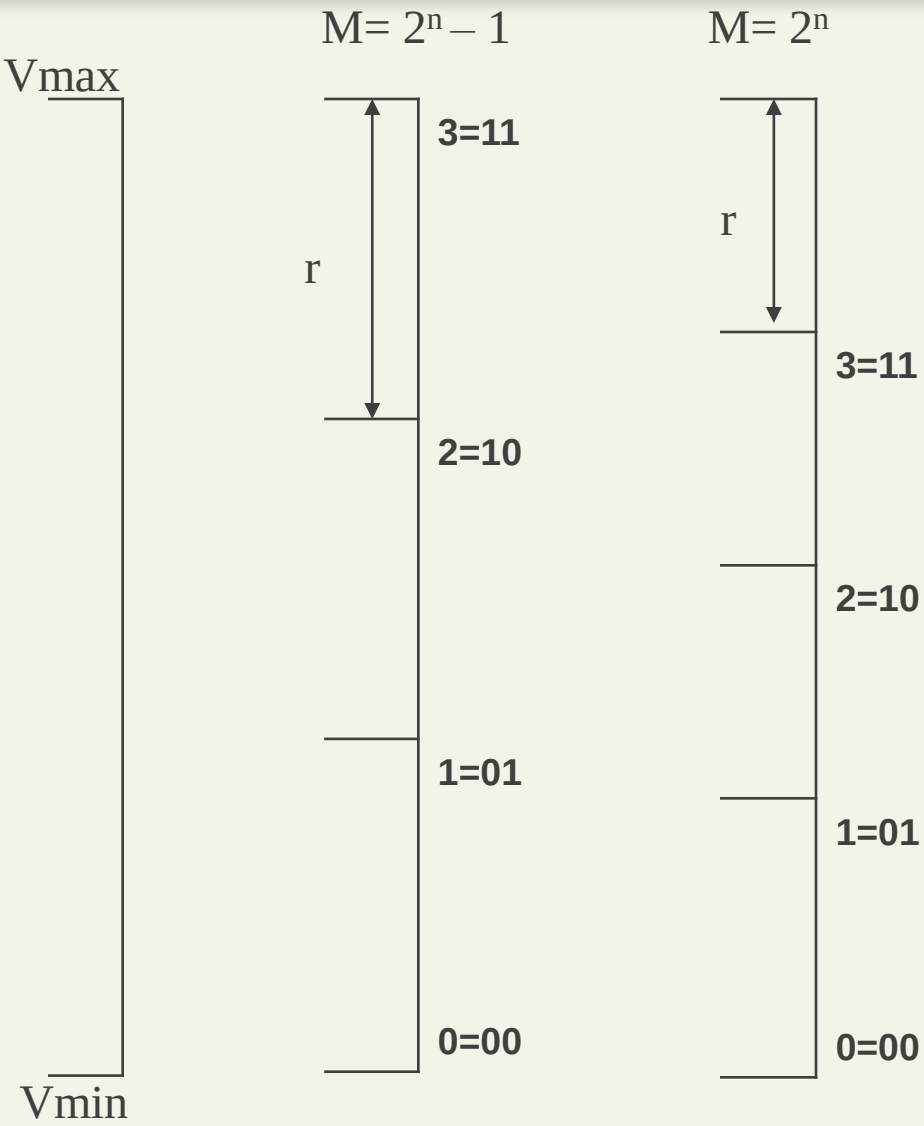
<u>$M = 2^n$</u>

4 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax - r} = 3 = 0b11$ (no d_{Vmax}, it would be at 0b100=4)

r, **resolution**: analog change resulting from a digital change of 1



Resolution: $M = 2^n - 1$ vs. 2^n

Let n = 2

<u>$M = 2^n - 1$</u>

3 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax} = 3 = 0b11$

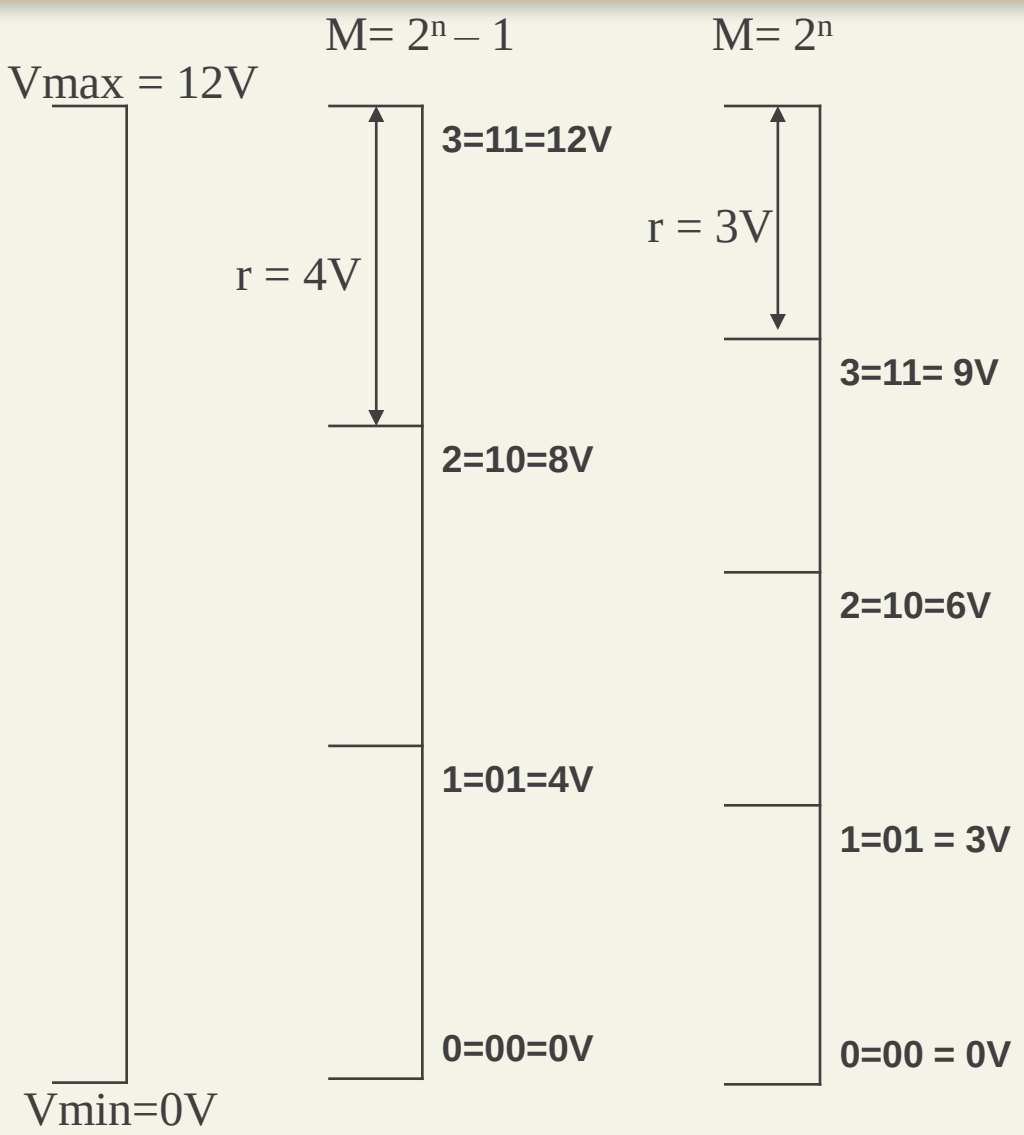
<u>$M = 2^n$</u>

4 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax - r} = 3 = 0b11$ (no d_{Vmax}, it would be at 0b100=4)

r, **resolution**: analog change resulting from a digital change of 1



Resolution: $M = 2^n - 1$ vs. 2^n

Let n = 2

<u>$M = 2^n - 1$</u>

3 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax} = 3 = 0b11$

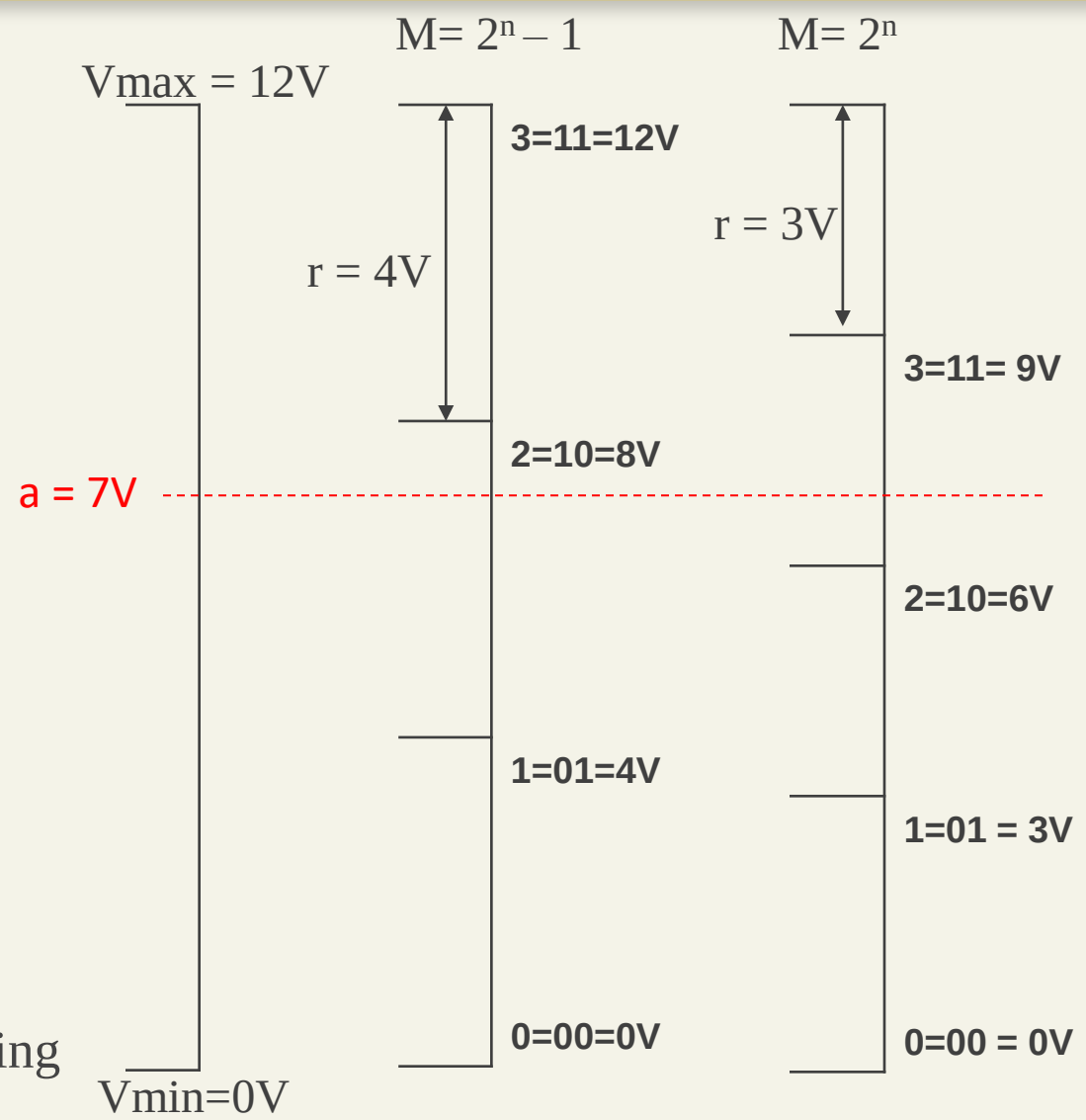
<u>$M = 2^n$</u>

4 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax - r} = 3 = 0b11$ (no d_{Vmax}, it would be at 0b100=4)

r, **resolution**: analog change resulting from a digital change of 1



Resolution: $M = 2^n - 1$ vs. 2^n

Let n = 2

<u>$M = 2^n - 1$</u>

3 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax} = 3 = 0b11$

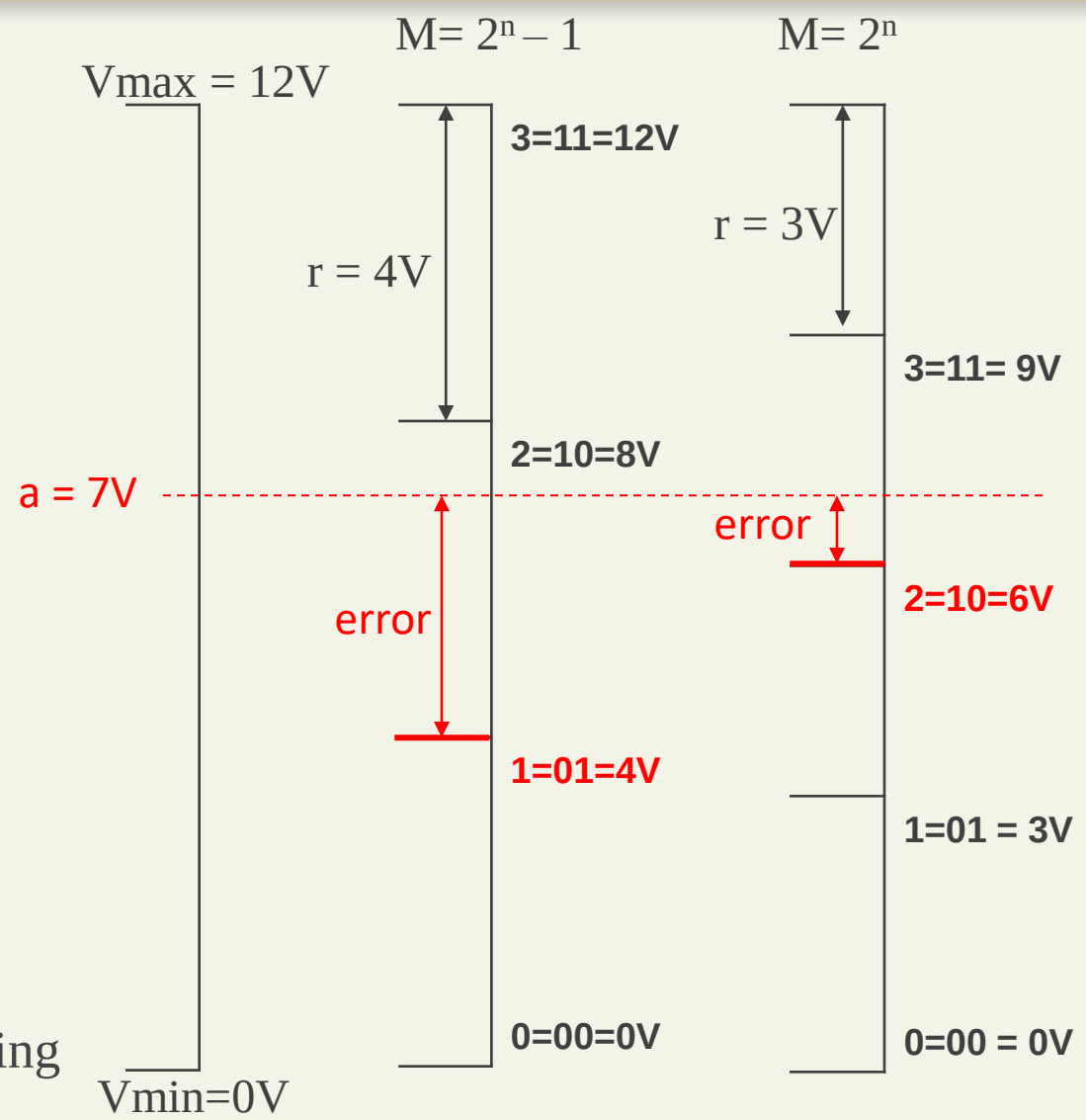
<u>$M = 2^n$</u>

4 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax - r} = 3 = 0b11$ (no d_{Vmax}, it would be at 0b100=4)

r, **resolution**: analog change resulting from a digital change of 1



Resolution: $M = 2^n - 1$ vs. 2^n (Related to slope)

Let n = 2

<u>$M = 2^n - 1$</u>

3 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax} = 3 = 0b11$

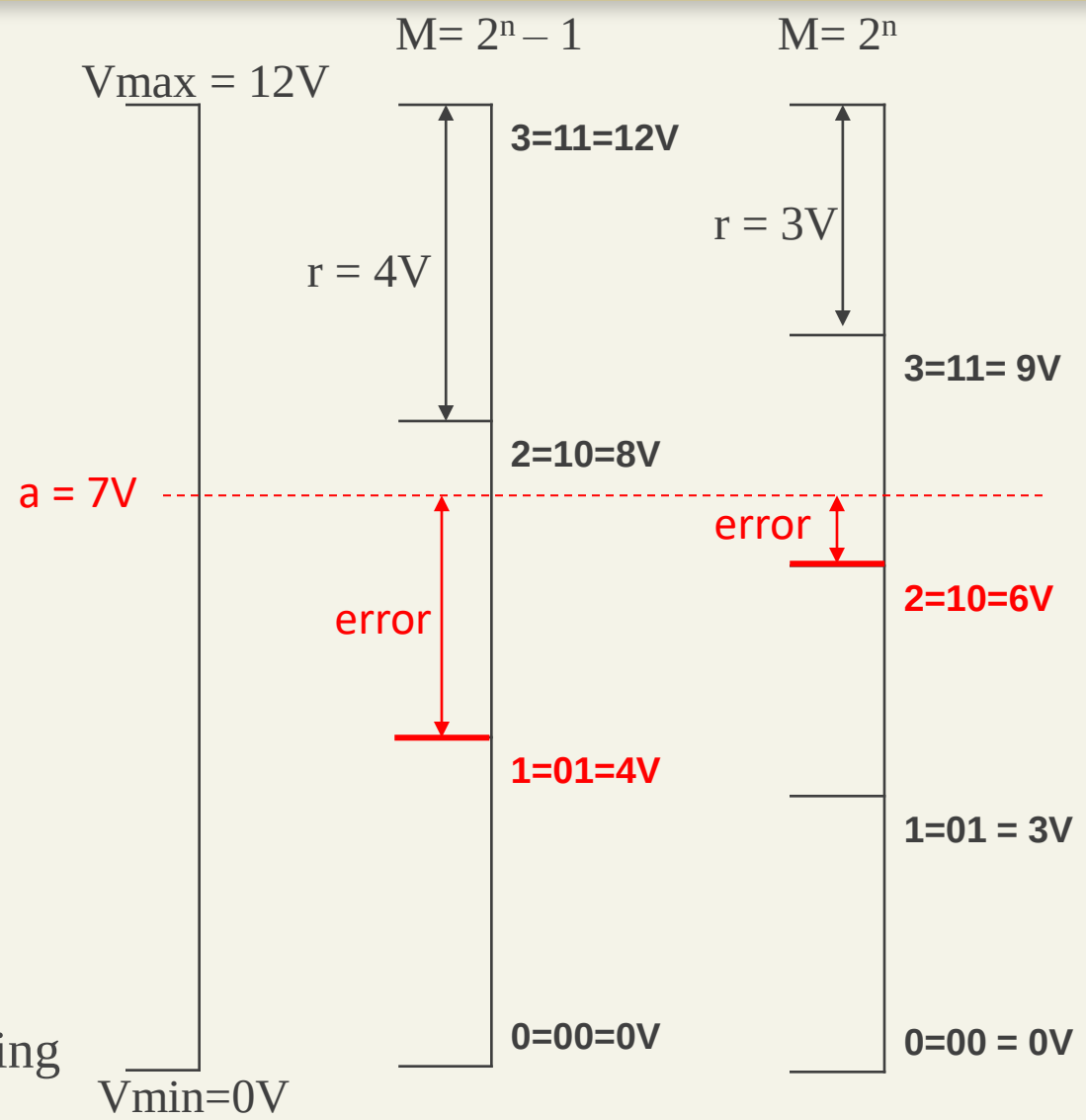
<u>$M = 2^n$</u>

4 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{Vmax - r} = 3 = 0b11$ (no d_{Vmax}, it would be at 0b100=4)

r, **resolution**: analog change resulting from a digital change of 1



DAC vs. ADC

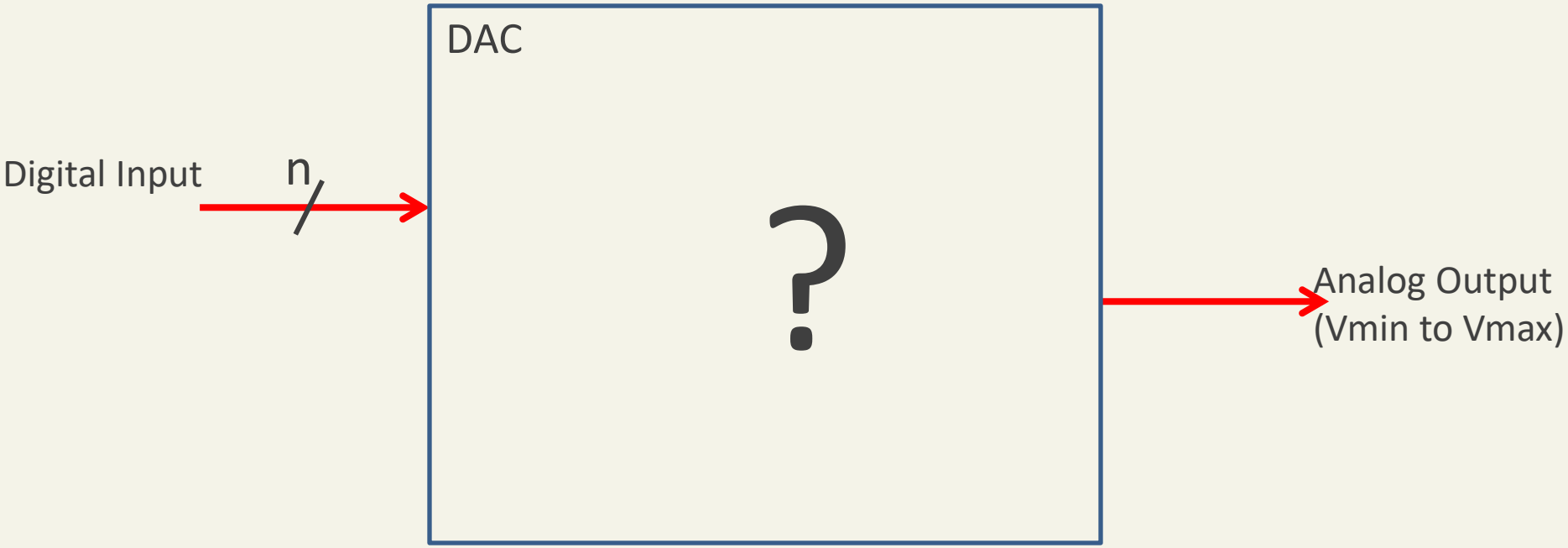
- DAC (Digital to Analog Converter)
  - n-bit digital input (**d)**
  - analog out between **Vmax** and **Vmin (a)**

- ADC (Analog to Digital Converter)
  - analog input between **Vmax** and **Vmin (a)**
  - n-bit digital output **(d)**

DAC: Conceptual Implementation

DAC (Digital-to-Analog Converter):

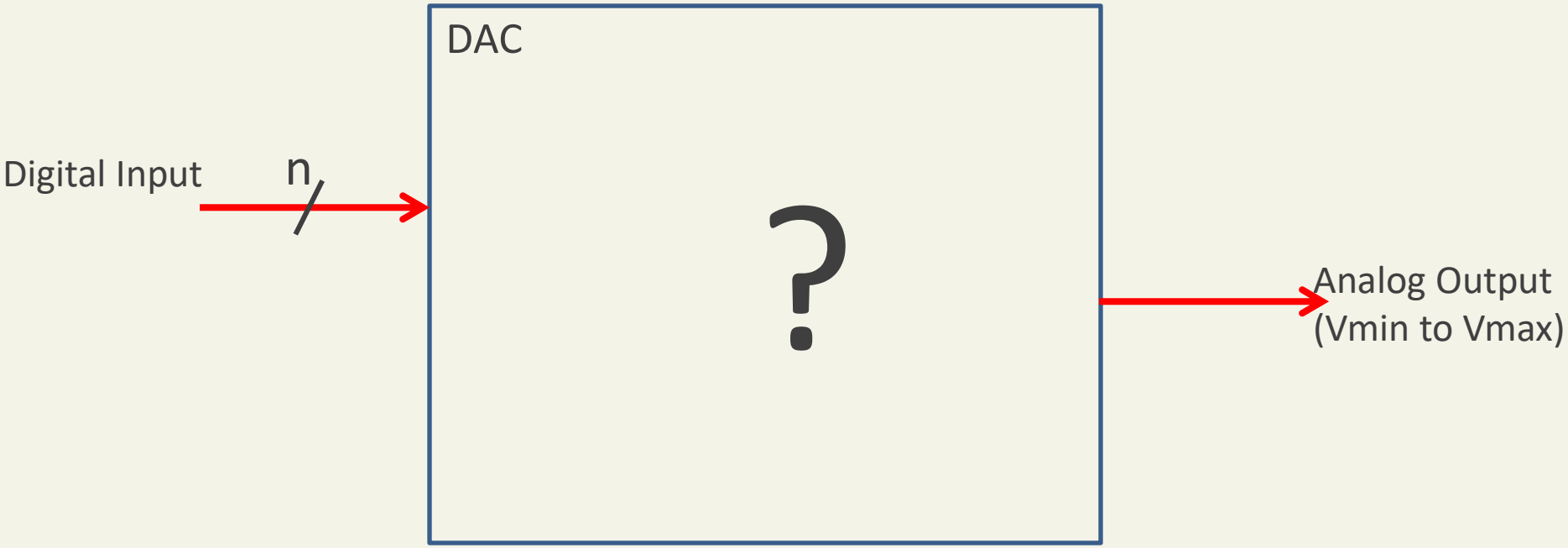
- Conceptually, given a n-bit digital input (**d**), how does the DAC generate an analog output (**a**) between Vmin to Vmax?



DAC: Conceptual Implementation

DAC (Digital-to-Analog Converter):

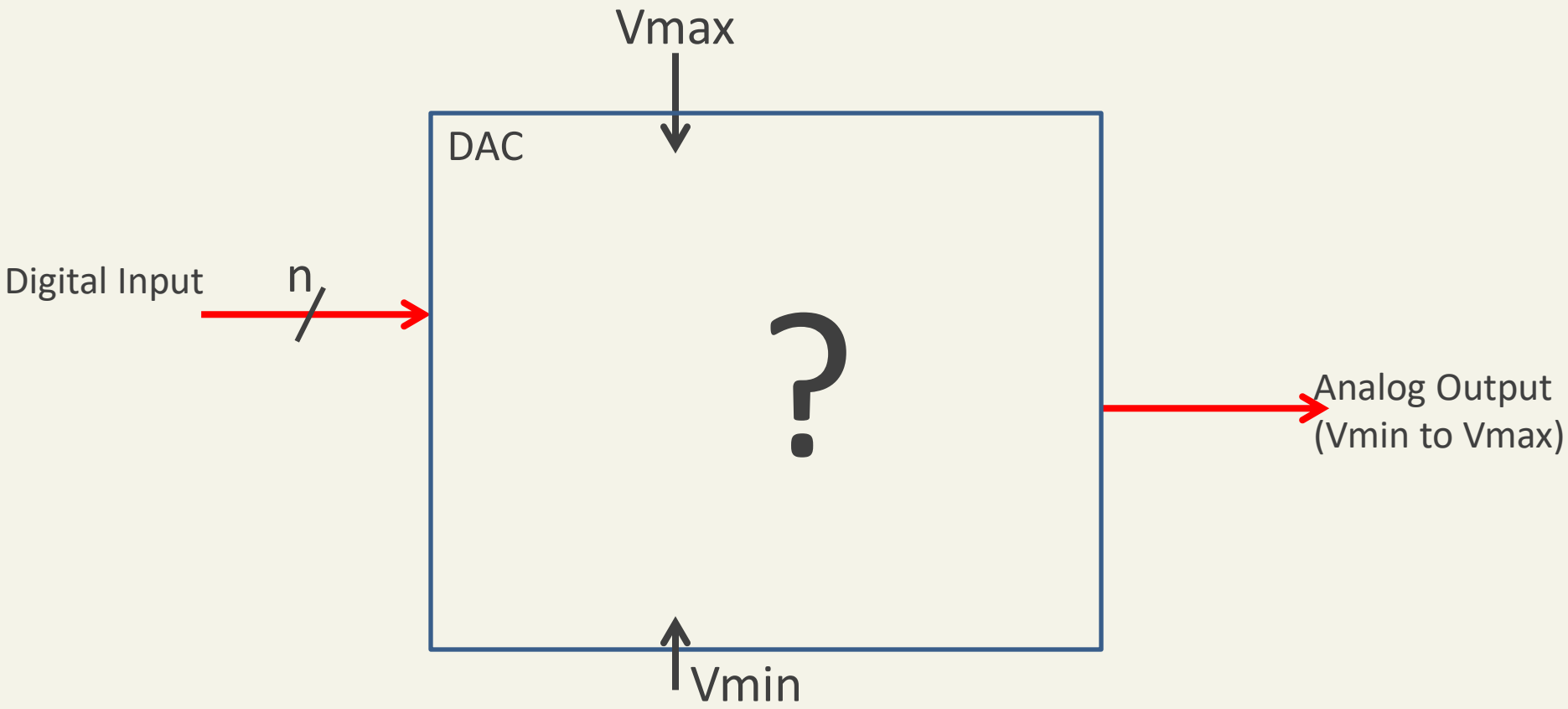
- Conceptually, given a n-bit digital input (**d**), how does the DAC generate an analog output (**a**) between Vmin to Vmax?
- What other information does the DAC need?



DAC: Conceptual Implementation

DAC (Digital-to-Analog Converter):

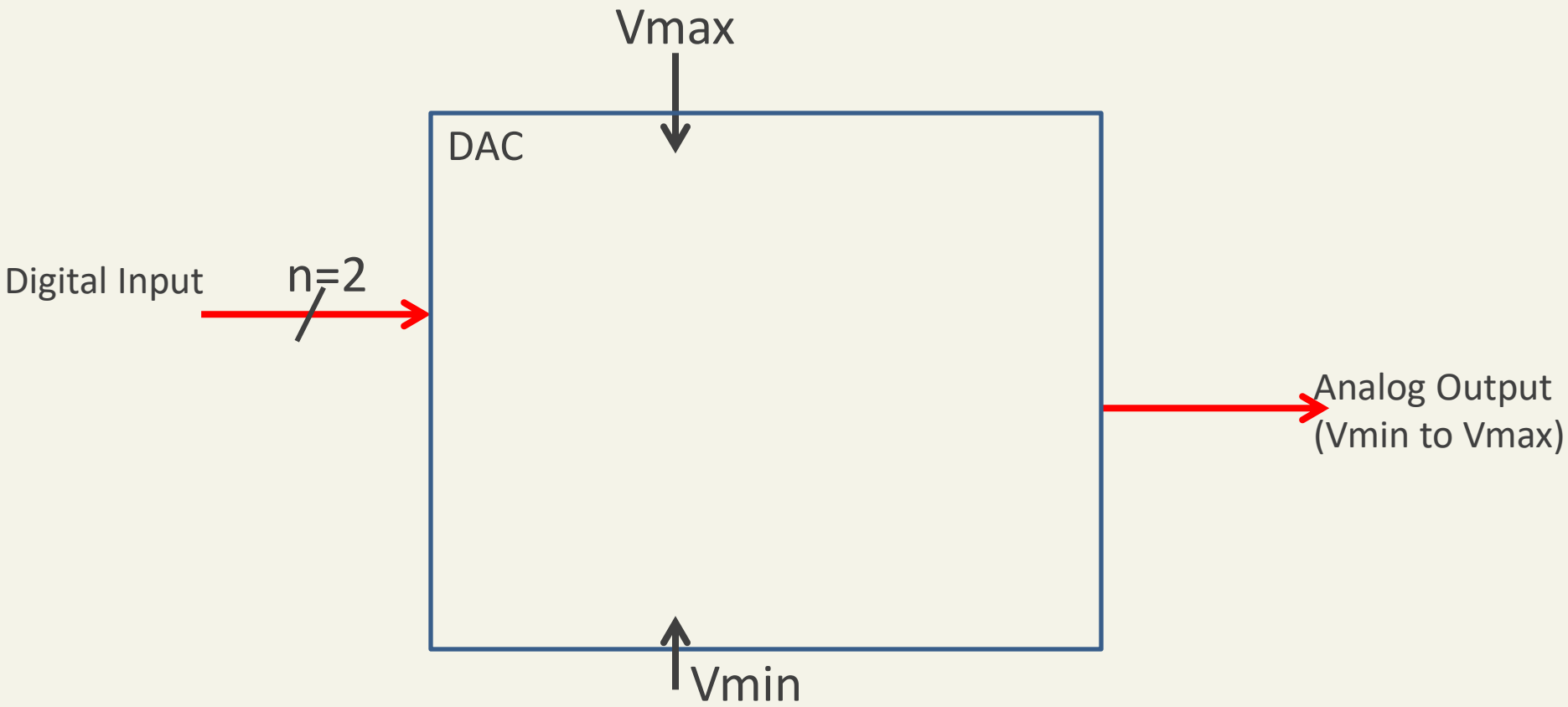
- Conceptually, given a n-bit digital input (**d**), how does the DAC generate an analog output (**a**) between Vmin to Vmax?



DAC: Conceptual Implementation

DAC (Digital-to-Analog Converter):

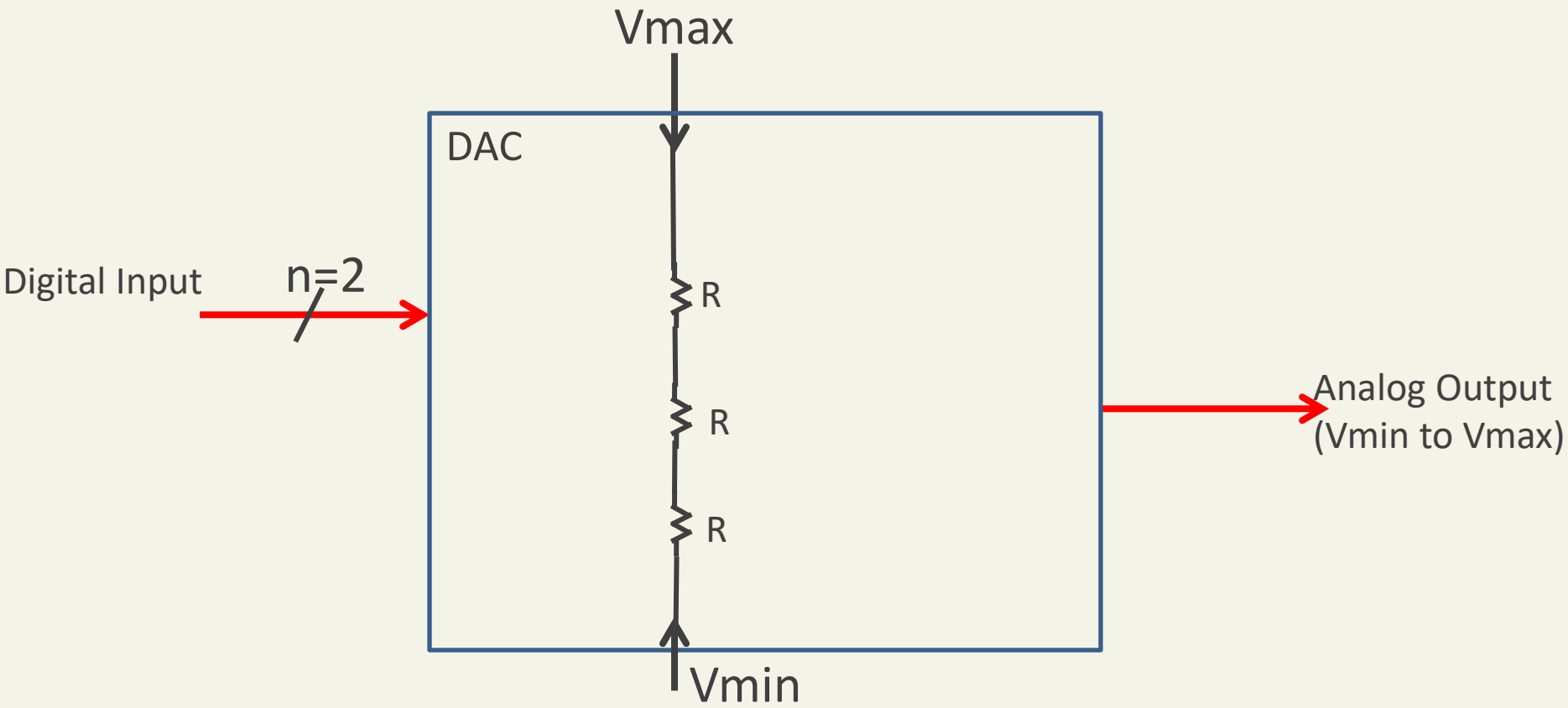
- Conceptually, given a n-bit digital input (**d**), how does the DAC generate an analog output (**a**) between Vmin to Vmax?



DAC: Conceptual Implementation

DAC (Digital-to-Analog Converter):

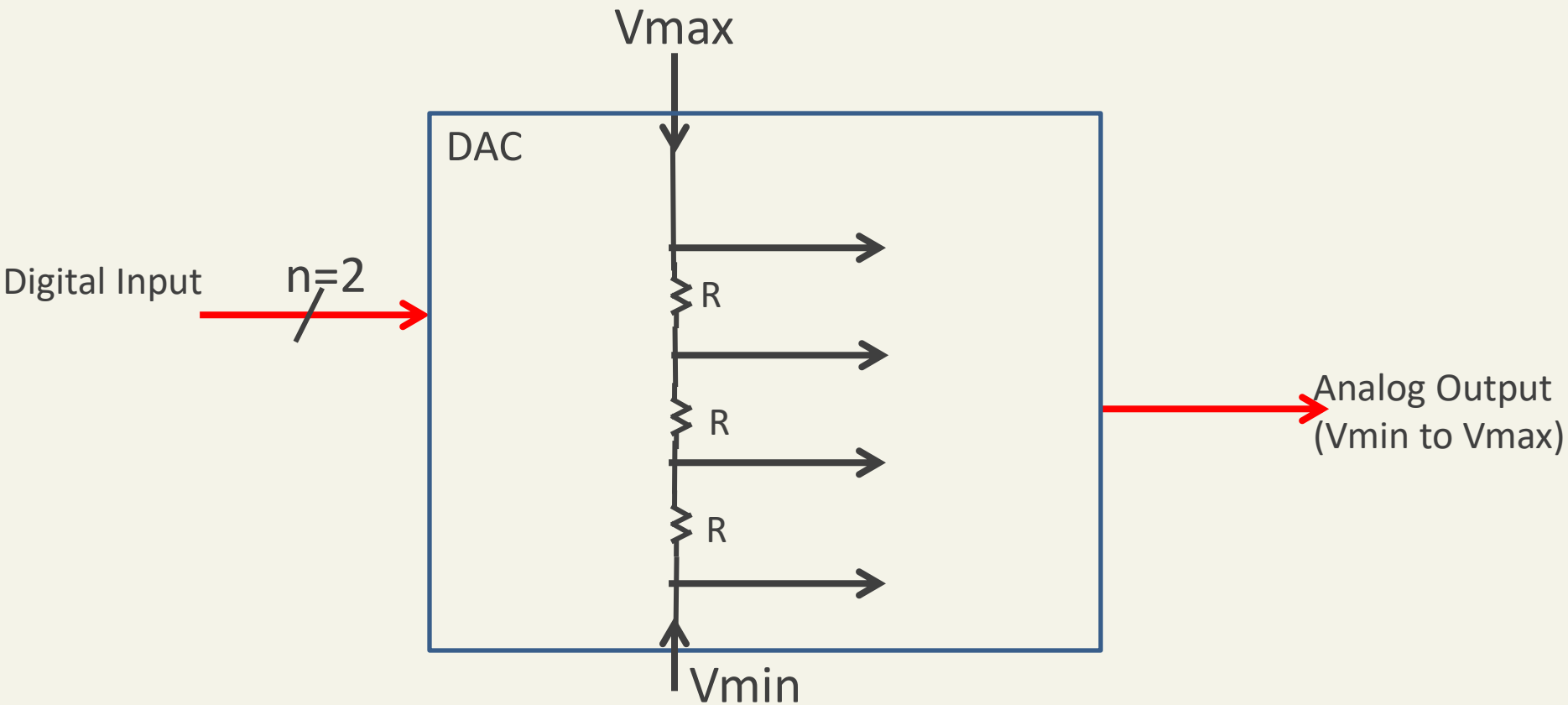
- Conceptually, given a n-bit digital input (**d**), how does the DAC generate an analog output (**a**) between Vmin to Vmax?



DAC: Conceptual Implementation

DAC (Digital-to-Analog Converter):

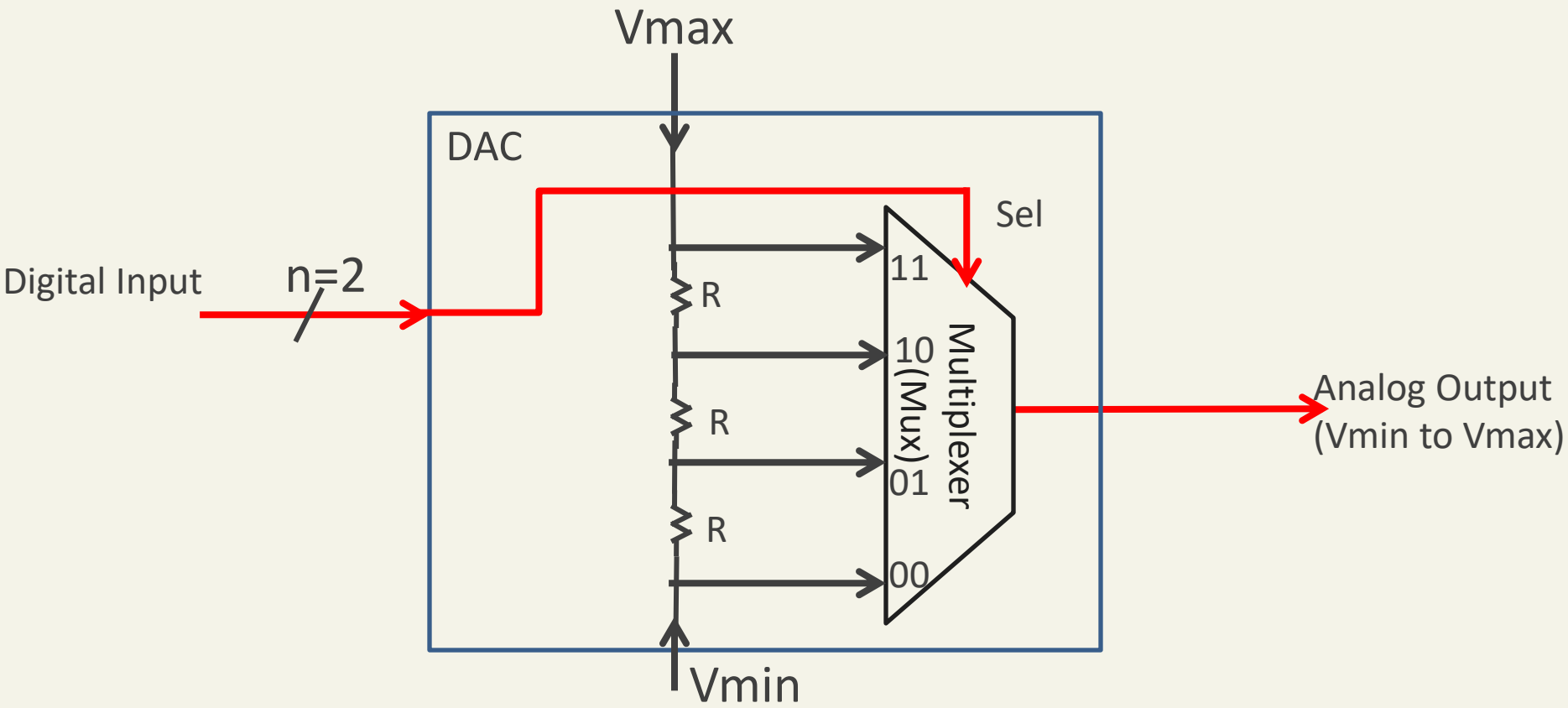
- Conceptually, given a n-bit digital input (**d**), how does the DAC generate an analog output (**a**) between Vmin to Vmax?



DAC: Conceptual Implementation

DAC (Digital-to-Analog Converter):

- Conceptually, given a n-bit digital input (**d**), how does the DAC generate an analog output (**a**) between Vmin to Vmax?



ADC: Conceptual Implementation

ADC:

Given an analog input (**a**), how does the ADC know what binary value to assign to digital output (**d**)?

ADC: Conceptual Implementation

ADC:

Given an analog input (**a**), how does the ADC know what binary value to assign to digital output (**d**)?

- Use a DAC to generate analog values for comparison with (**a**)
- ADC "guesses" a (**d**), then checks its guess by inputting (**d)** into the DAC and comparing the generated analog output (**a'**) with original analog input (**a)**
- How does the ADC guess the correct encoding?

ADC: Digital Encoding

Guessing the encoding is similar to finding an item in a list.

1. Sequential search – counting up: start with an encoding of 0, then 1, then 2, etc. until find a match.
   - 2^n comparisons: Slow!
2. Binary search – successive approximation: start with an encoding for half of maximum; then compare analog result with original analog input; if result is greater (less) than the original, set the new encoding to halfway between this one and the minimum (maximum); continue dividing encoding range in half until the compared voltages are equal
   - n comparisons: Faster, but more complex converter

- Each guess takes time (e.g. Assume 1us per guess)
  - 10-bit ADC, what is the time difference for 1. vs. 2. (2^{10} ~ 1,000)
  - For a 20-bit ADC? (2^{20} ~ 1 Million)
  - For a 30-bit ADC? (2^{30} ~ 1 Billion)

ADC: Digital Encoding

Guessing the encoding is similar to finding an item in a list.

1. Sequential search – counting up: start with an encoding of 0, then 1, then 2, etc. until find a match.
   - 2^n comparisons: Slow!
2. Binary search – successive approximation: start with an encoding for half of maximum; then compare analog result with original analog input; if result is greater (less) than the original, set the new encoding to halfway between this one and the minimum (maximum); continue dividing encoding range in half until the compared voltages are equal
   - n comparisons: Faster, but more complex converter

- Each guess takes time (e.g. Assume 1us per guess)
  - 10-bit ADC, what is the time difference for 1. vs. 2. (2^{10} ~ 1,000): 1ms vs. 10us
  - For a 20-bit ADC? (2^{20} ~ 1 Million)
  - For a 30-bit ADC? (2^{30} ~ 1 Billion)

ADC: Digital Encoding

Guessing the encoding is similar to finding an item in a list.

1. Sequential search – counting up: start with an encoding of 0, then 1, then 2, etc. until find a match.
   - 2^n comparisons: Slow!
2. Binary search – successive approximation: start with an encoding for half of maximum; then compare analog result with original analog input; if result is greater (less) than the original, set the new encoding to halfway between this one and the minimum (maximum); continue dividing encoding range in half until the compared voltages are equal
   - n comparisons: Faster, but more complex converter

- Each guess takes time (e.g. Assume 1us per guess)
  - 10-bit ADC, what is the time difference for 1. vs. 2. (2^{10} ~ 1,000): 1ms vs. 10us
  - For a 20-bit ADC? (2^{20} ~ 1 Million): 1s vs. 20us
  - For a 30-bit ADC? (2^{30} ~ 1 Billion)

ADC: Digital Encoding

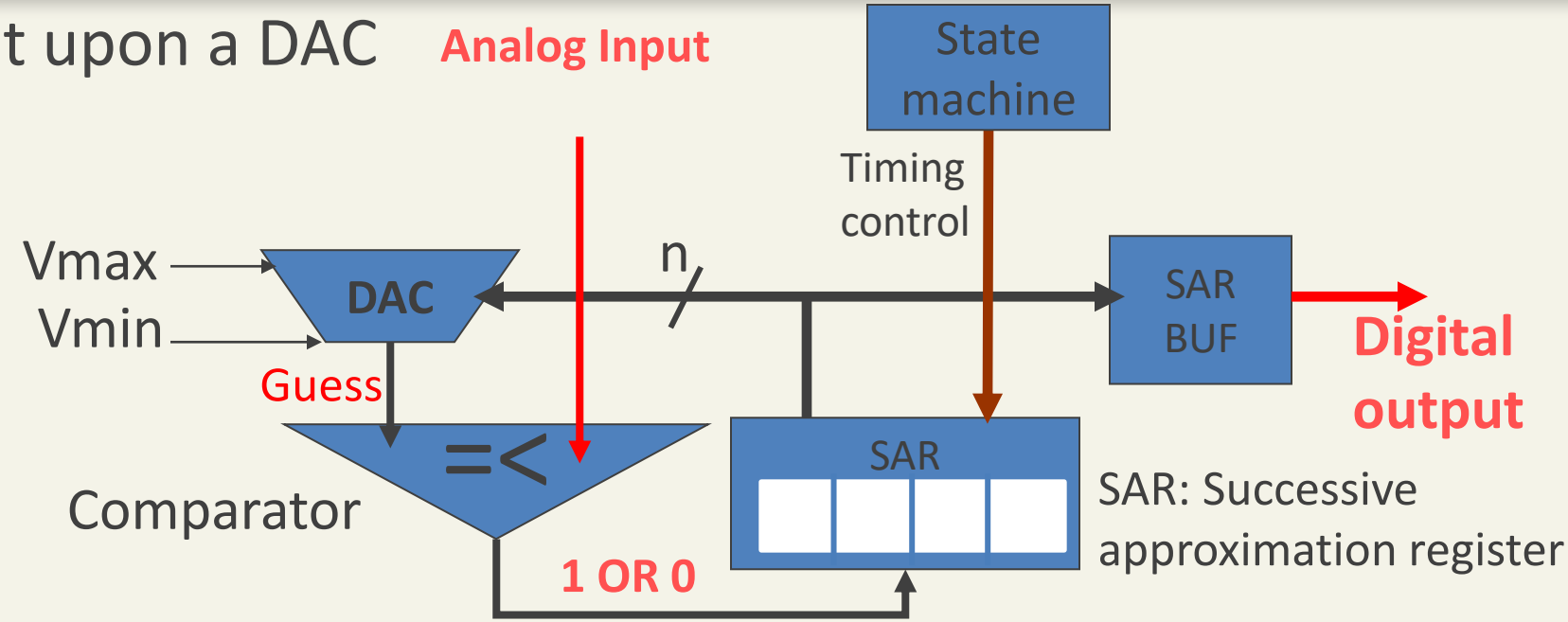
Guessing the encoding is similar to finding an item in a list.

1. Sequential search – counting up: start with an encoding of 0, then 1, then 2, etc. until find a match.
   - 2^n comparisons: Slow!
2. Binary search – successive approximation: start with an encoding for half of maximum; then compare analog result with original analog input; if result is greater (less) than the original, set the new encoding to halfway between this one and the minimum (maximum); continue dividing encoding range in half until the compared voltages are equal
   - n comparisons: Faster, but more complex converter

- Each guess takes time (e.g. Assume 1us per guess)
  - 10-bit ADC, what is the time difference for 1. vs. 2. (2^{10} ~ 1,000): 1ms vs. 10us
  - For a 20-bit ADC? (2^{20} ~ 1 Million): 1s vs. 20us
  - For a 30-bit ADC? (2^{30} ~ 1 Billion): 1000s = 16min vs. 30us

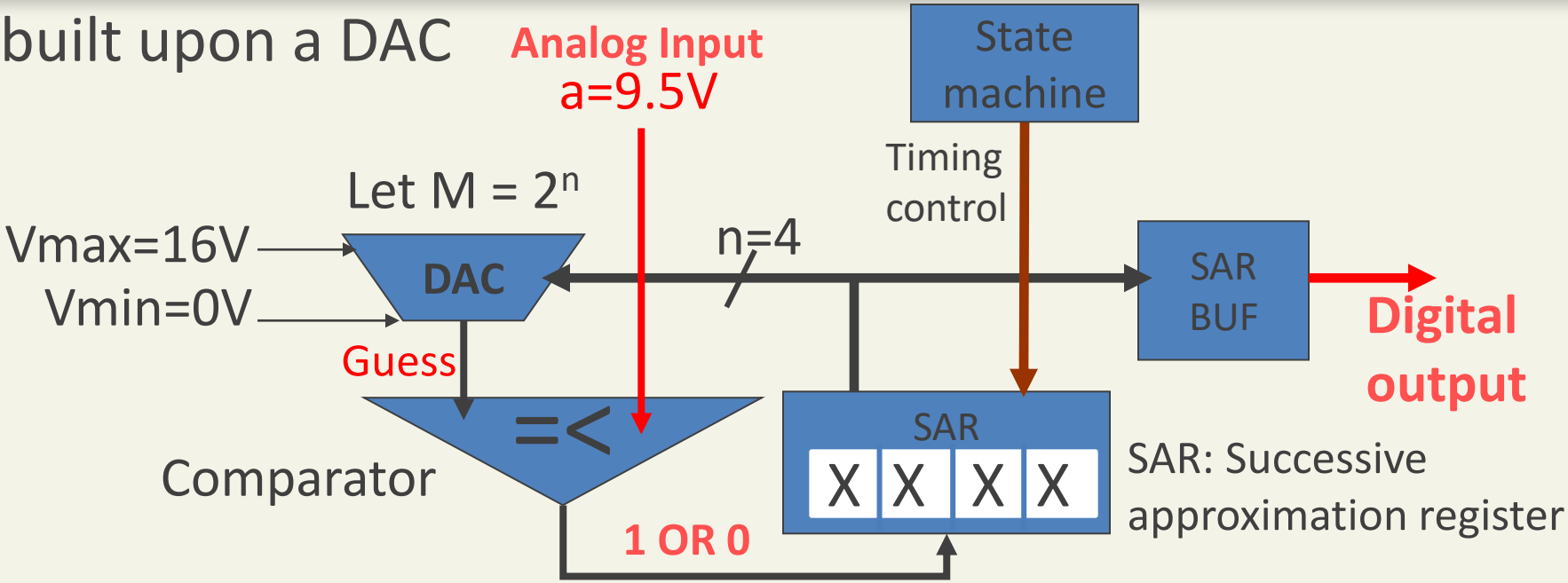
Constructing the ADC (Successive Approximation)

It's built upon a DAC



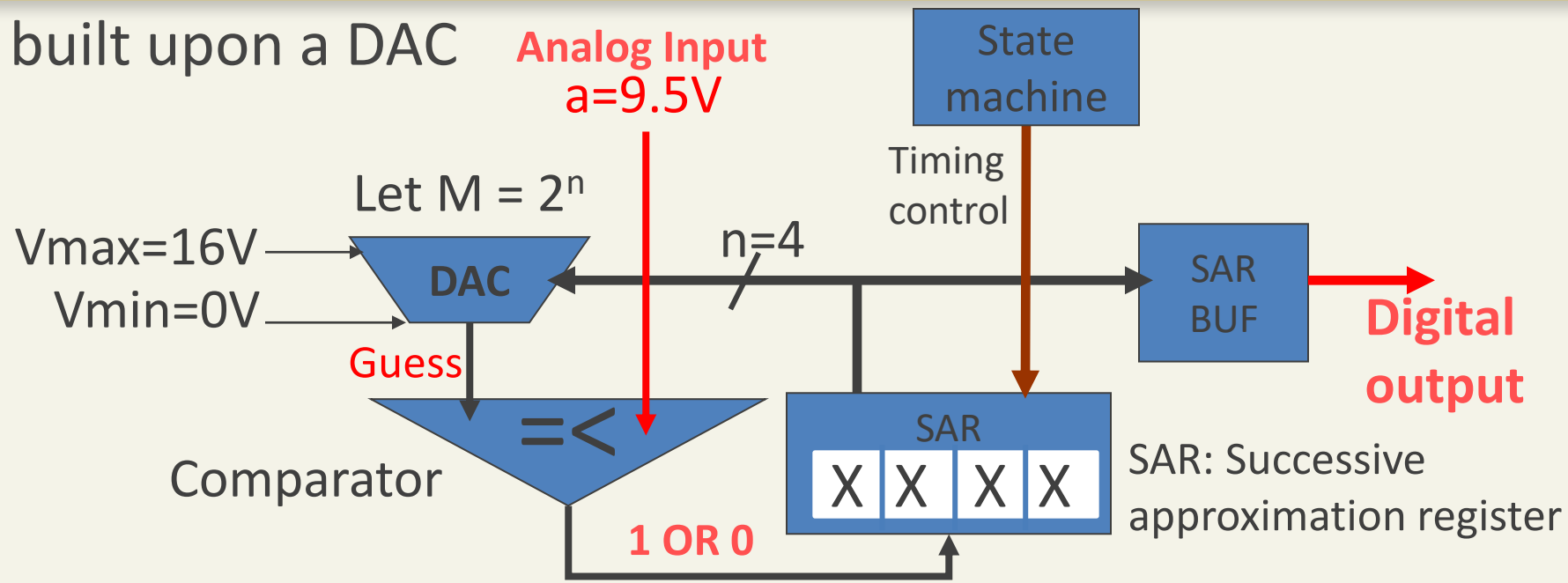
Constructing the ADC (Successive Approximation)

It's built upon a DAC



Constructing the ADC (Successive Approximation)

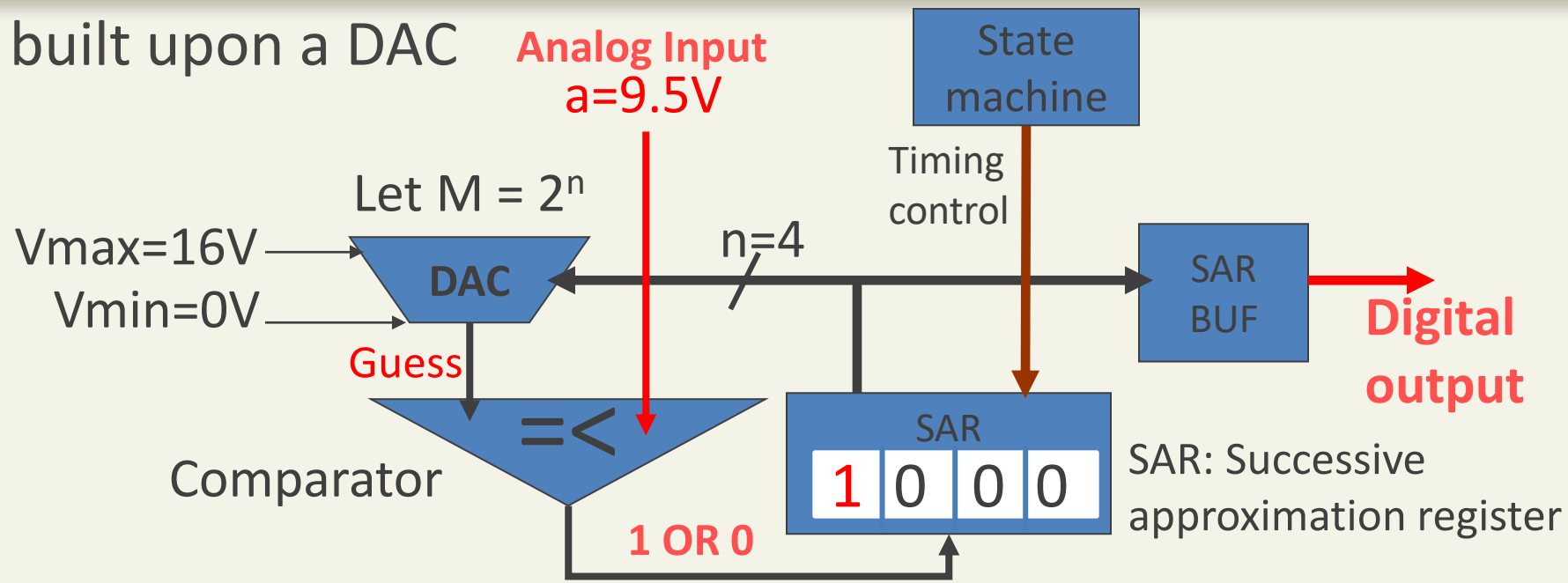
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx			
1				
2				
3				
4				

Constructing the ADC (Successive Approximation)

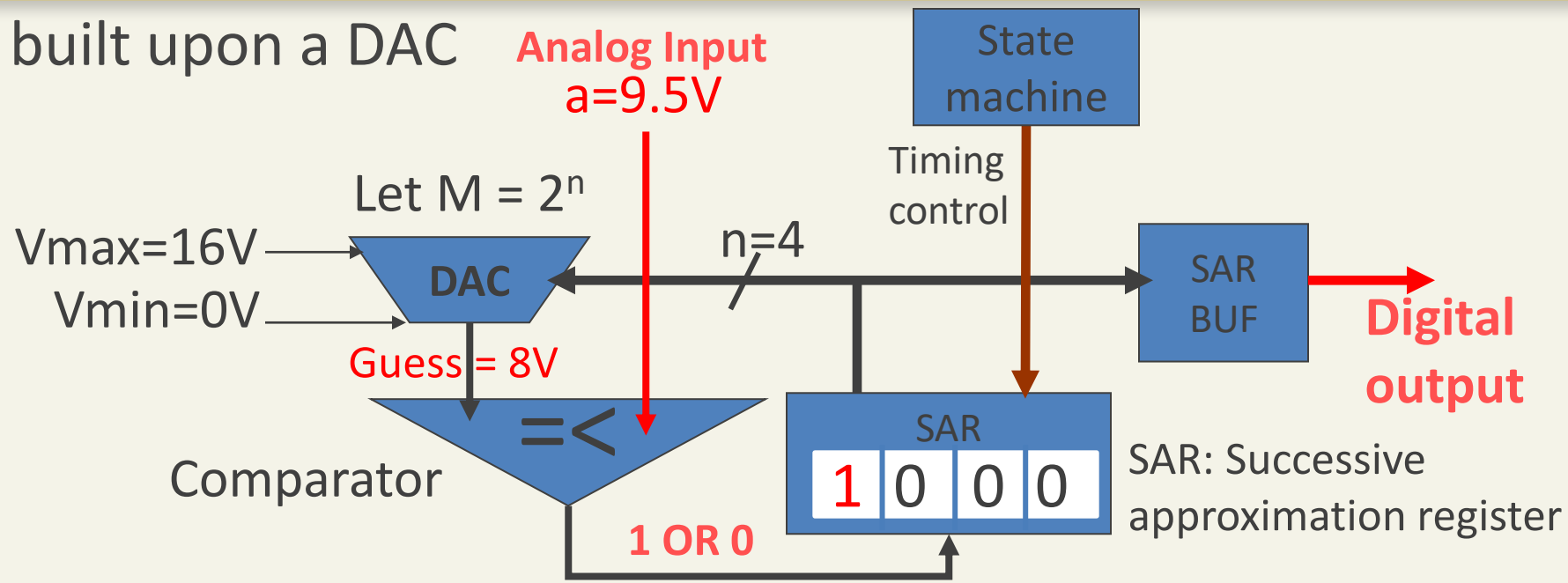
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000		
1				
2				
3				
4				

Constructing the ADC (Successive Approximation)

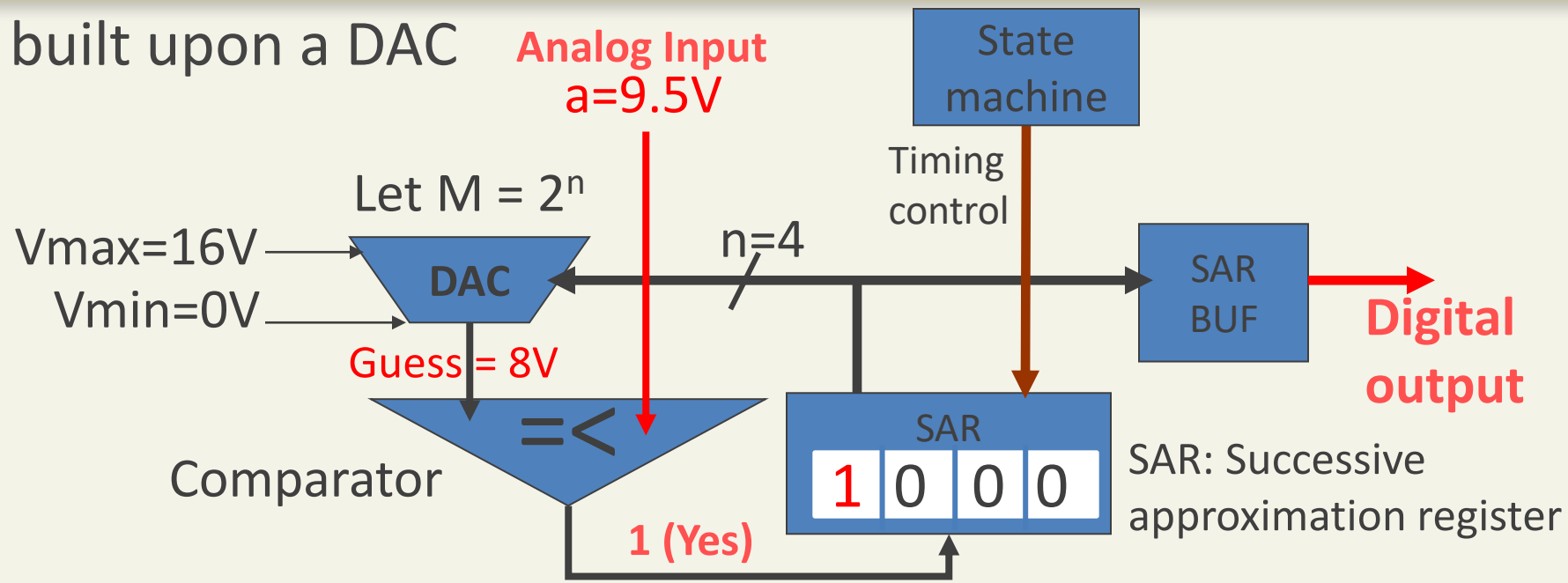
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	
1				
2				
3				
4				

Constructing the ADC (Successive Approximation)

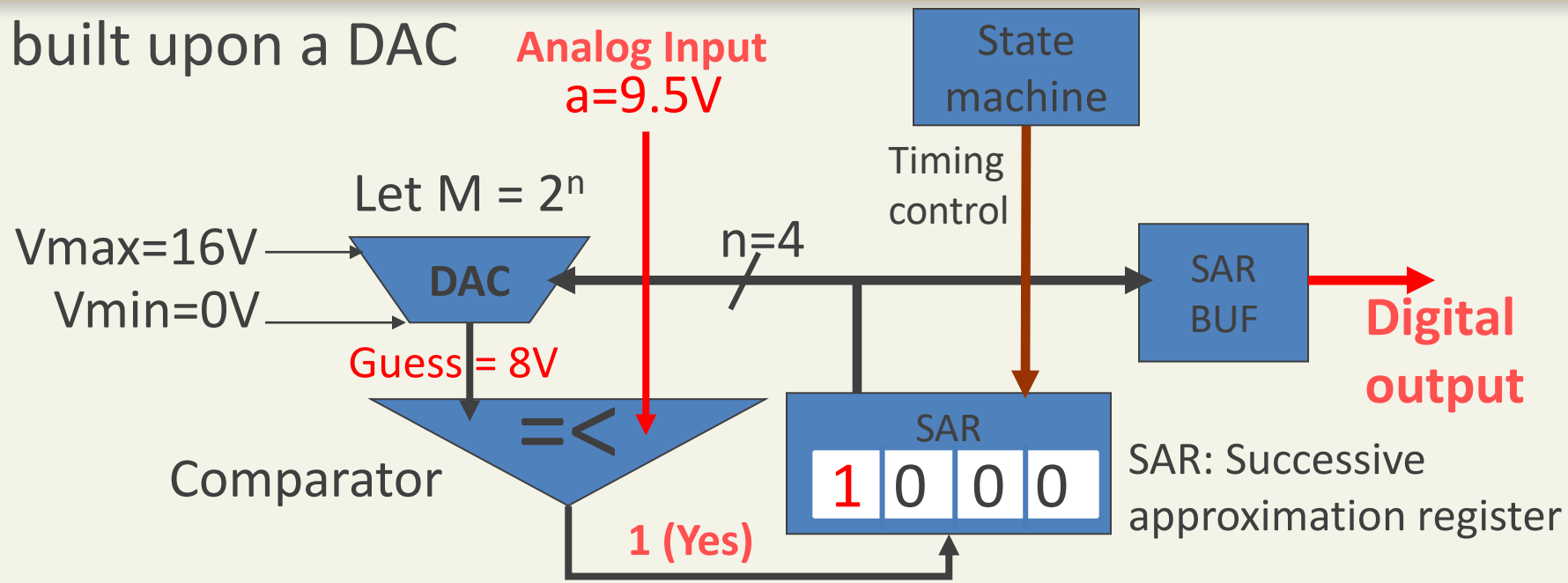
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1				
2				
3				
4				

Constructing the ADC (Successive Approximation)

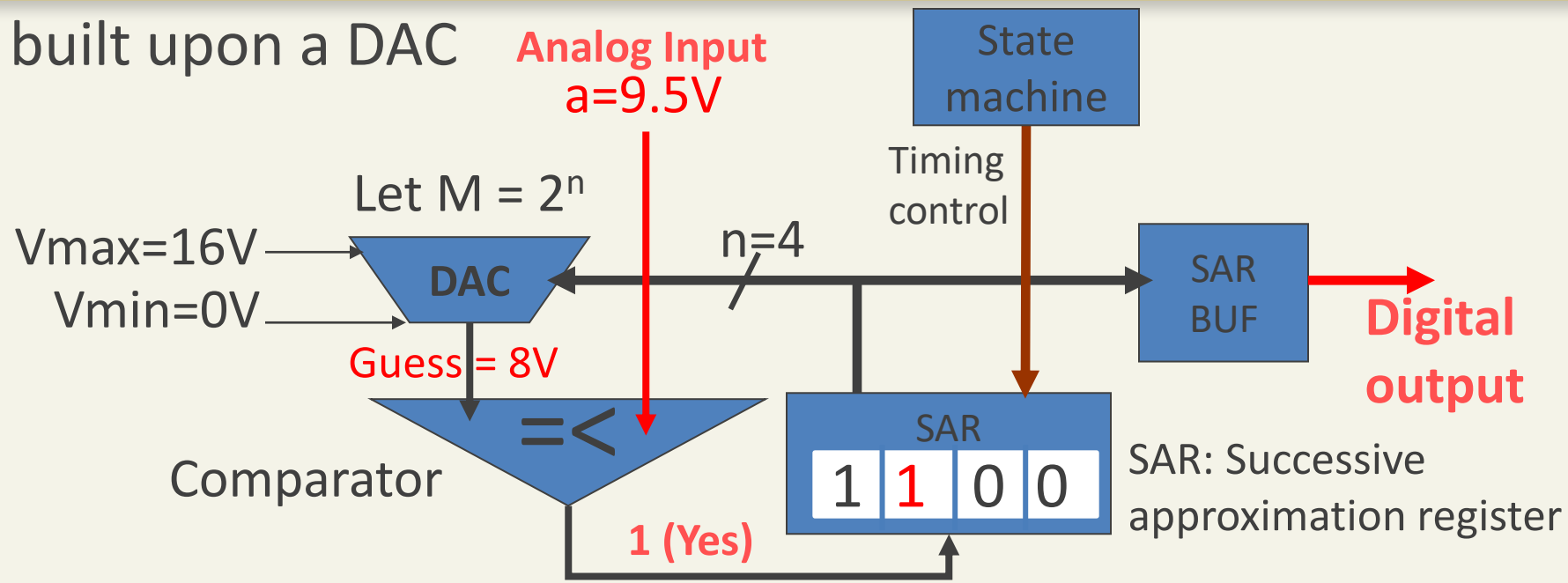
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx			
2				
3				
4				

Constructing the ADC (Successive Approximation)

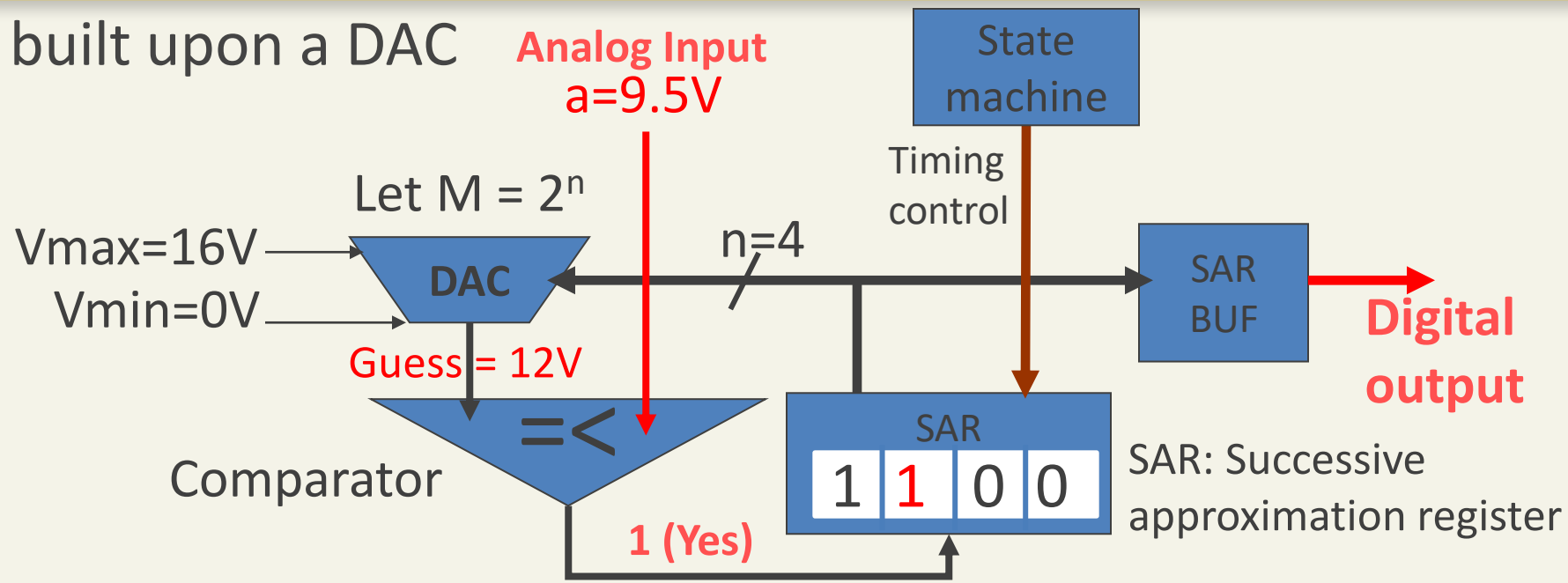
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100		
2				
3				
4				

Constructing the ADC (Successive Approximation)

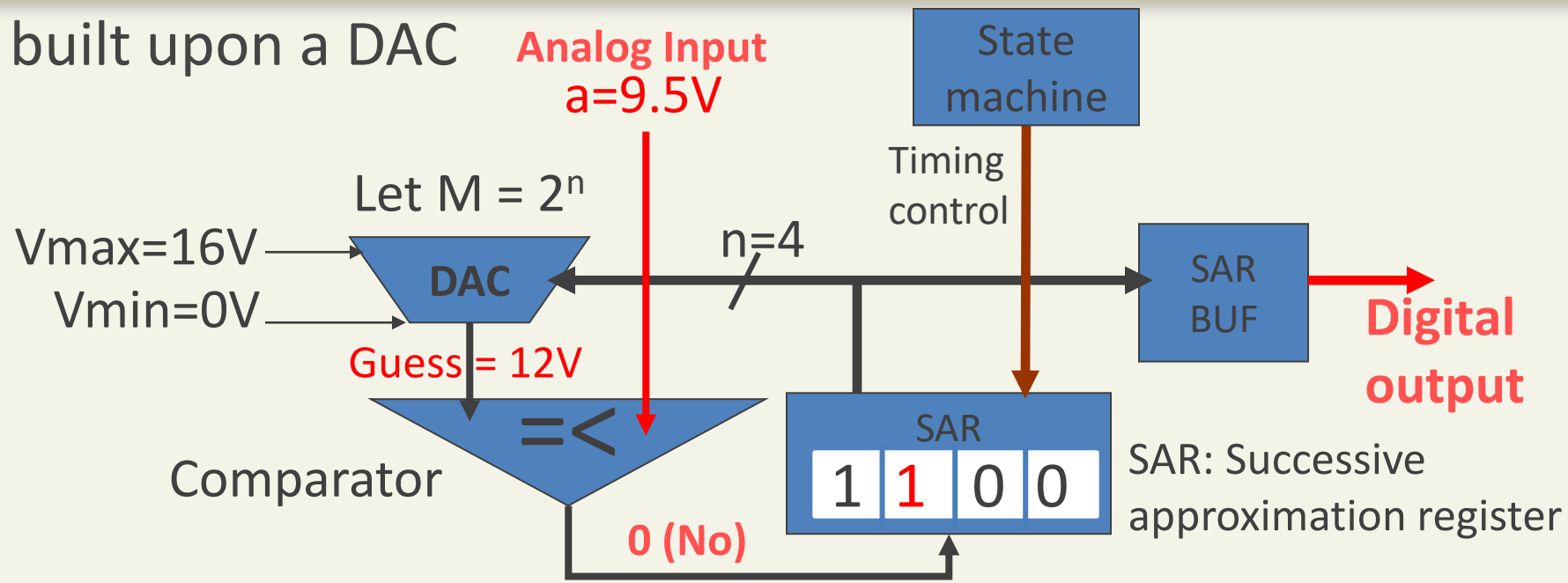
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	
2				
3				
4				

Constructing the ADC (Successive Approximation)

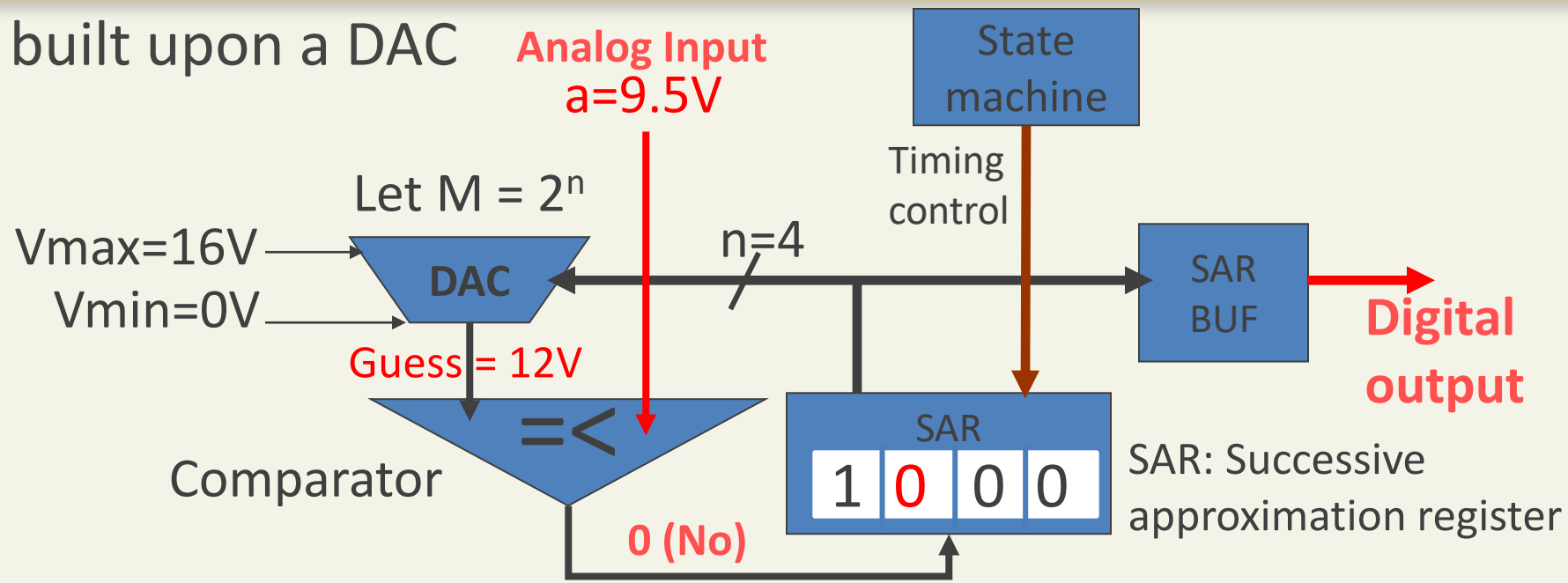
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2				
3				
4				

Constructing the ADC (Successive Approximation)

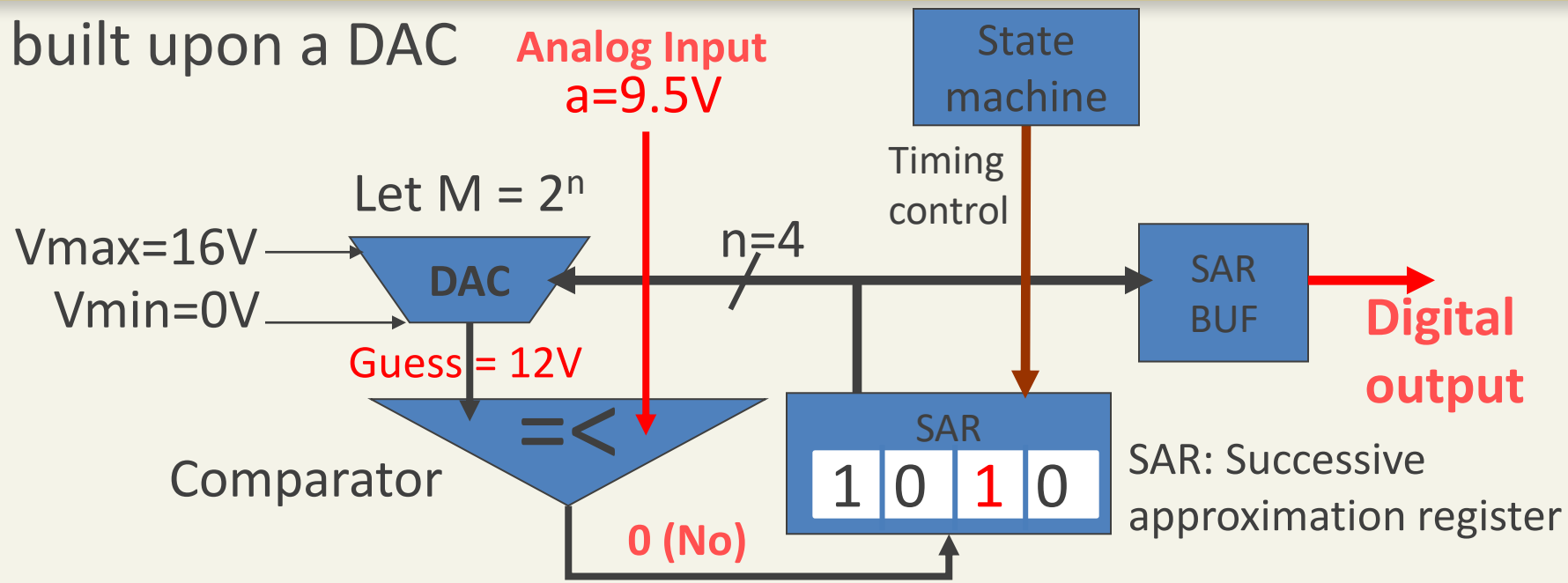
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx			
3				
4				

Constructing the ADC (Successive Approximation)

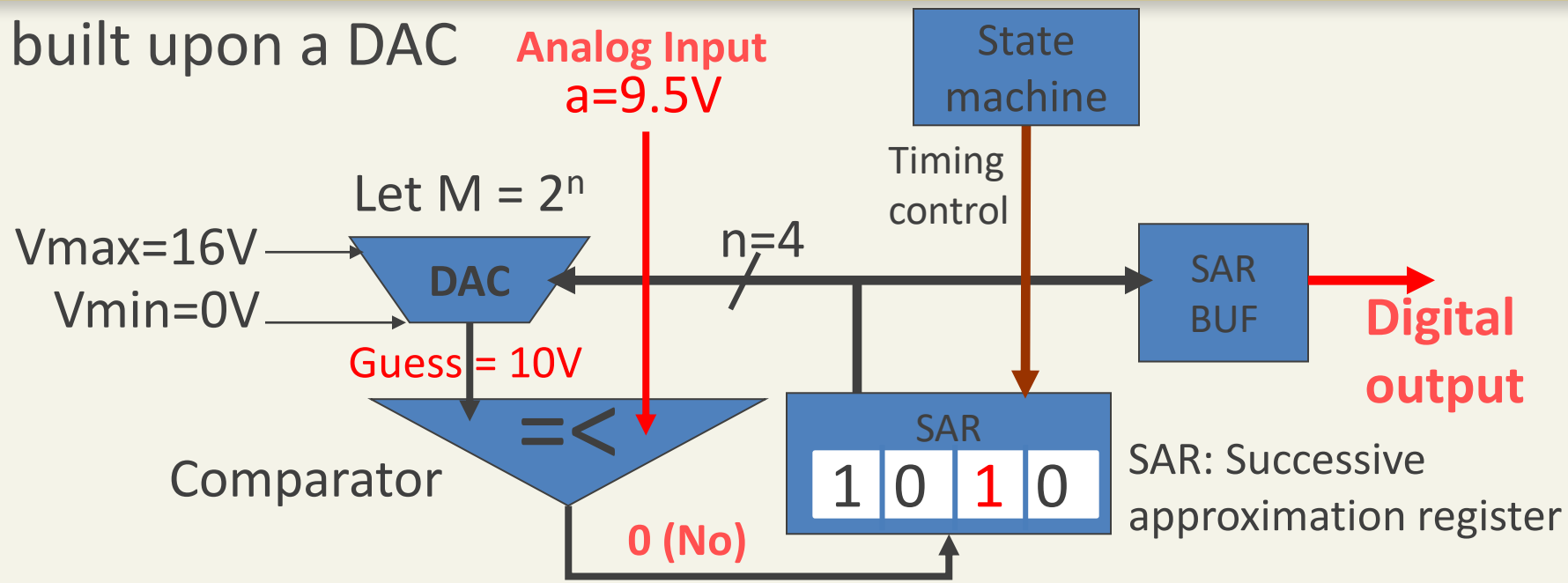
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010		
3				
4				

Constructing the ADC (Successive Approximation)

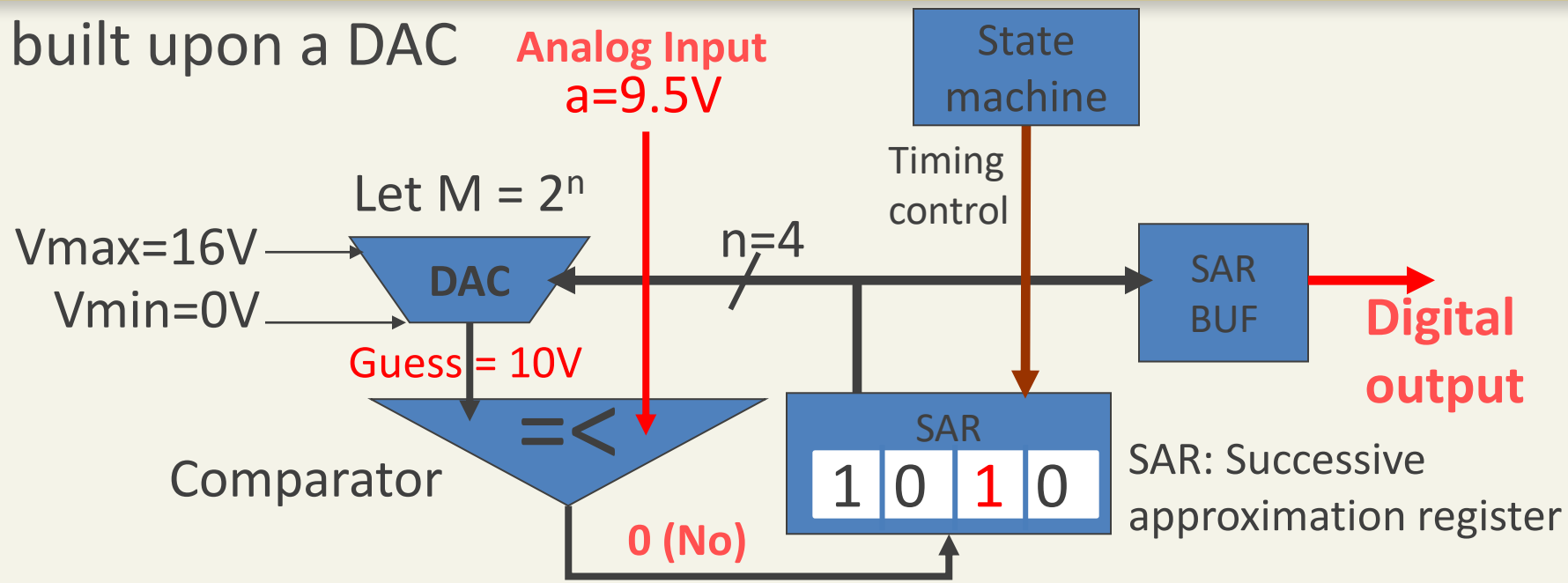
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010	10 Volts	
3				
4				

Constructing the ADC (Successive Approximation)

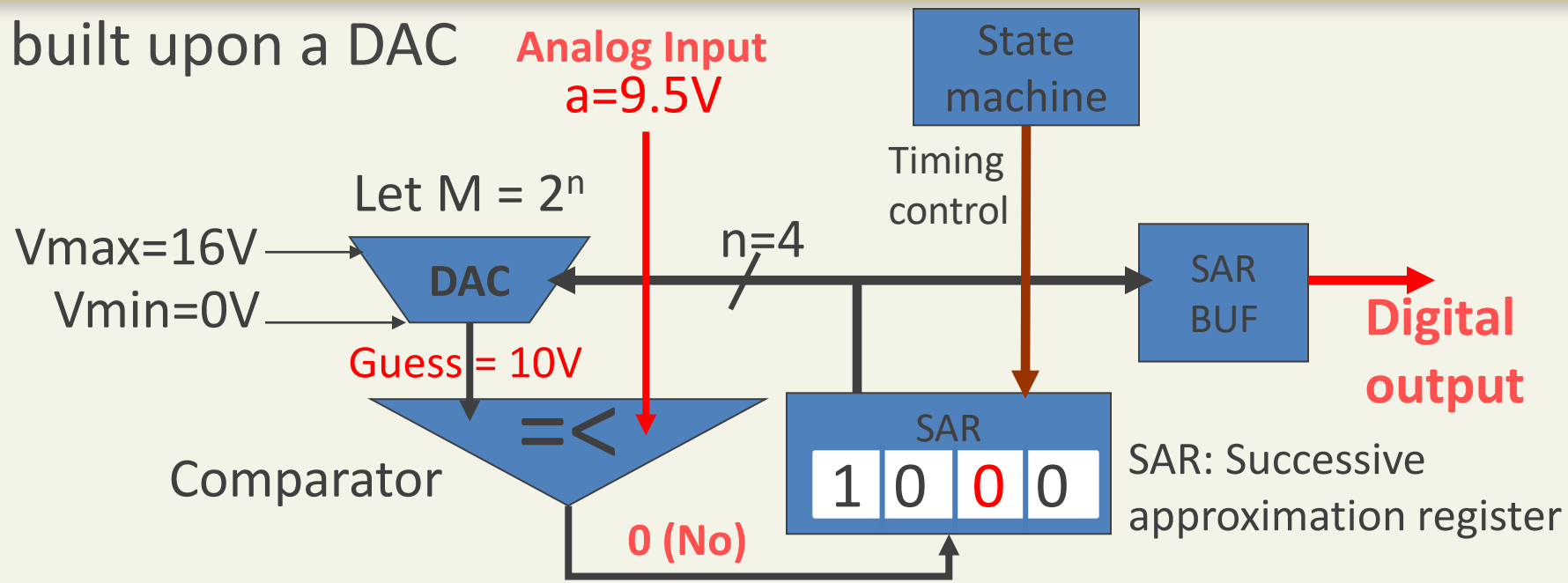
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010	10 Volts	No
3				
4				

Constructing the ADC (Successive Approximation)

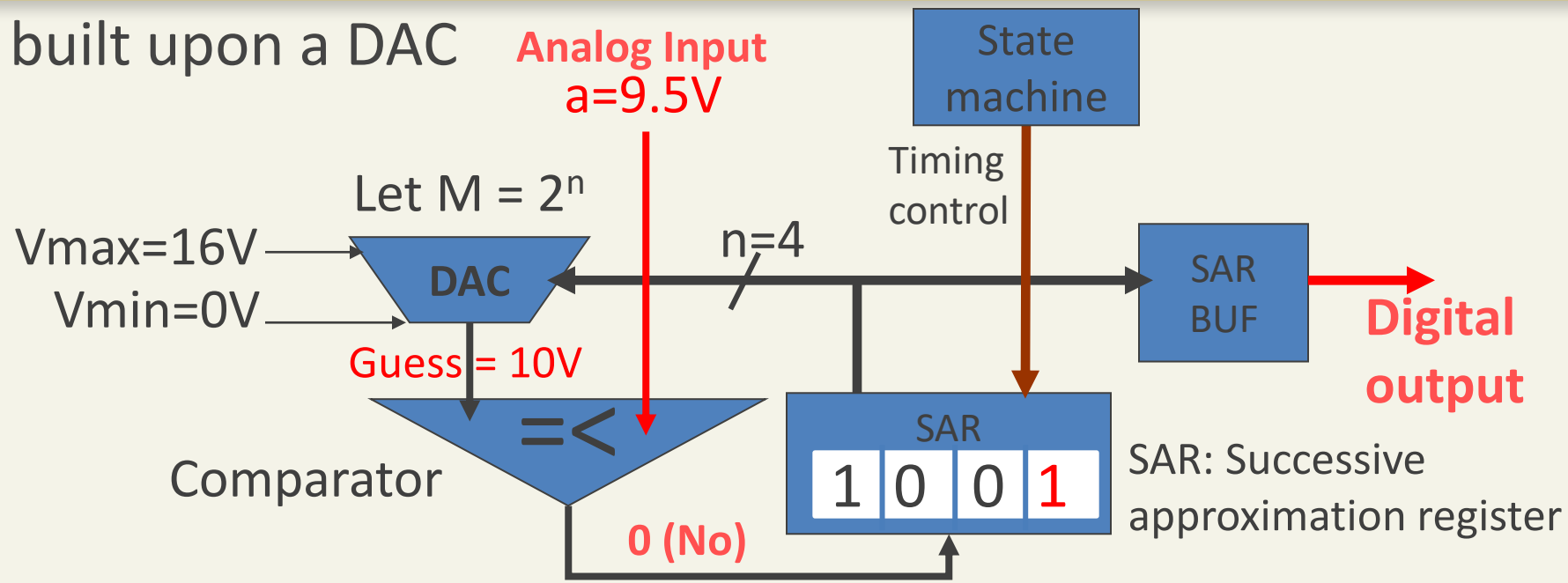
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010	10 Volts	No
3	0b100x			
4				

Constructing the ADC (Successive Approximation)

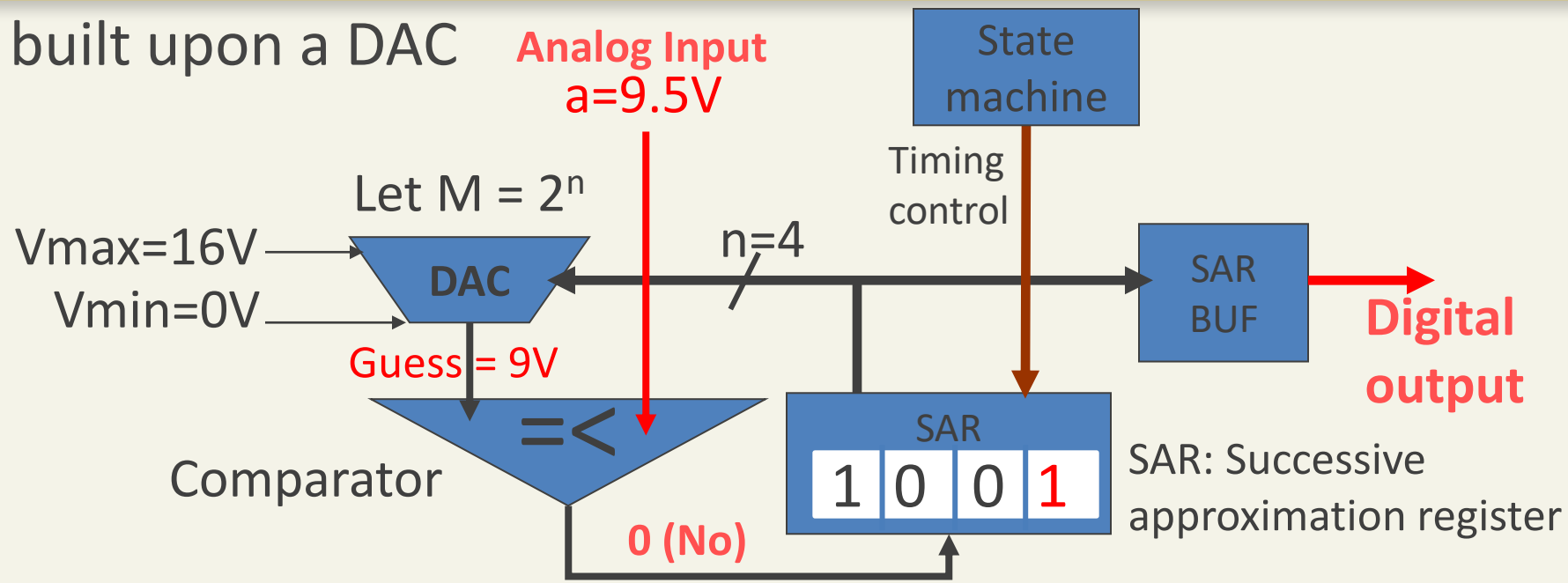
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010	10 Volts	No
3	0b100x	0b1001		
4				

Constructing the ADC (Successive Approximation)

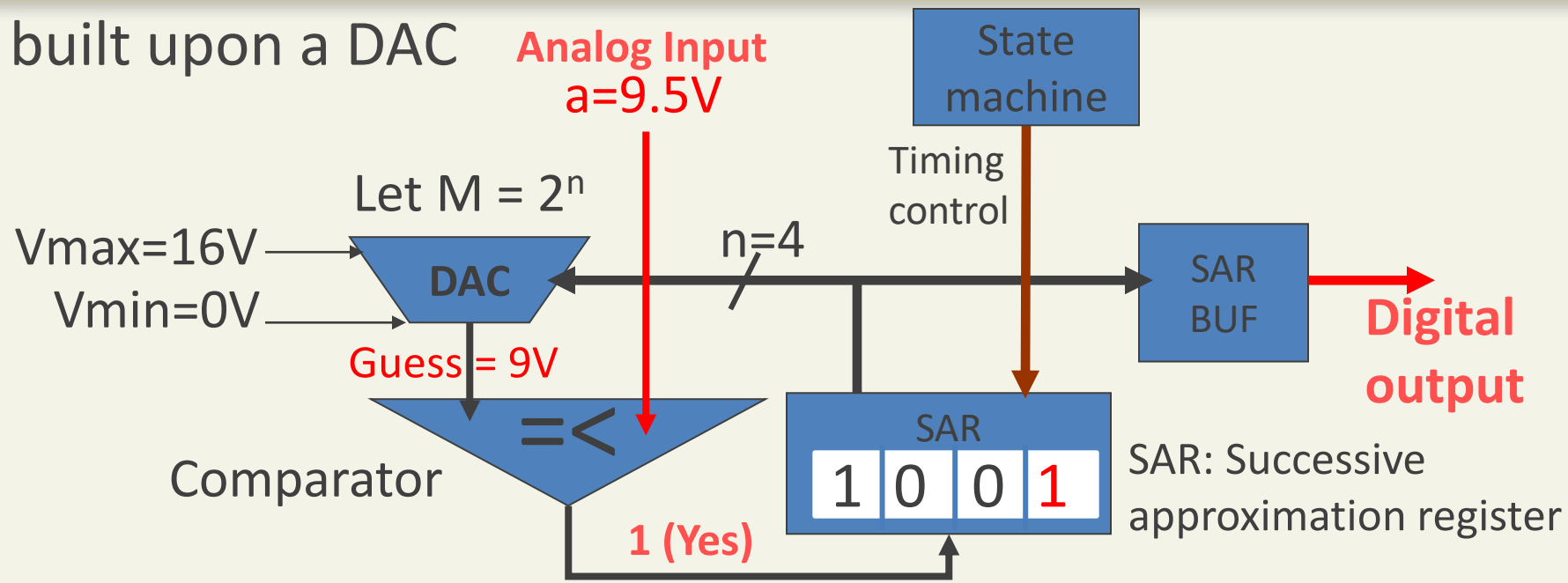
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010	10 Volts	No
3	0b100x	0b1001	9 Volts	
4				

Constructing the ADC (Successive Approximation)

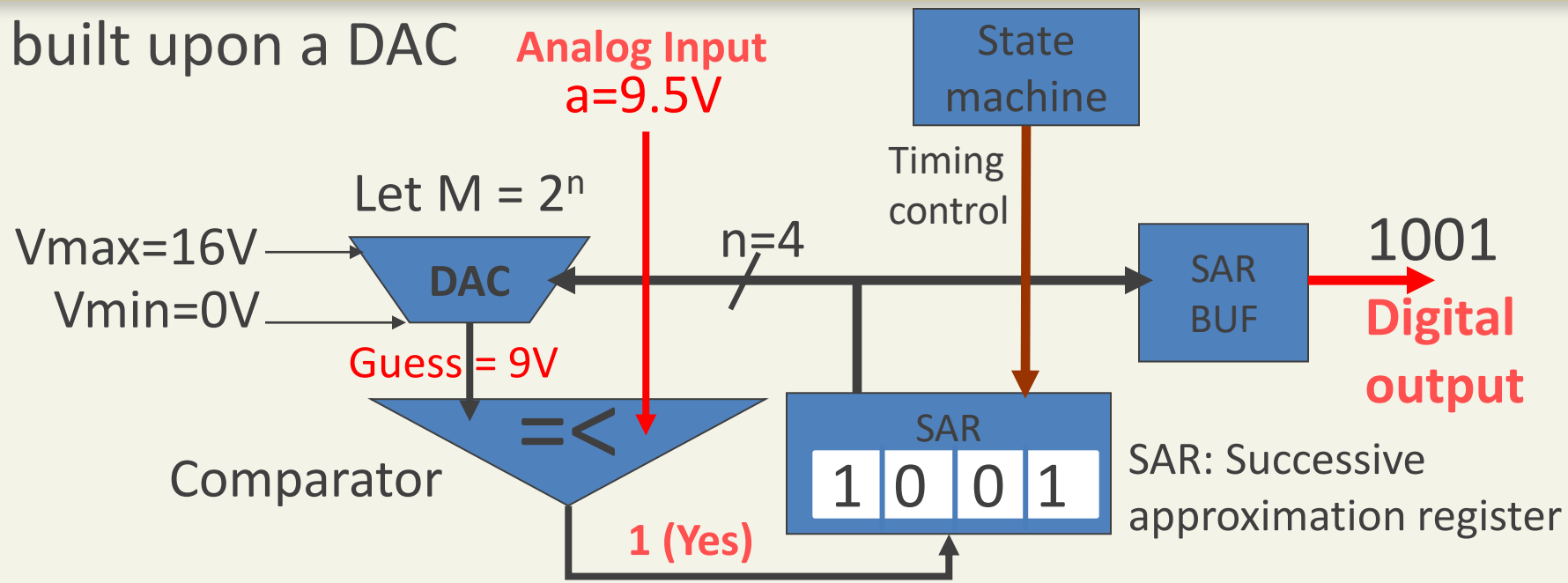
It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010	10 Volts	No
3	0b100x	0b1001	9 Volts	Yes
4	0b1001			

Constructing the ADC (Successive Approximation)

It's built upon a DAC



Step	Range	Mid (digital)	Mid (voltage)	Is a >= Guess (voltage)?
0	0bxxxx	0b1000	8 Volts	Yes
1	0b1xxx	0b1100	12 Volts	No
2	0b10xx	0b1010	10 Volts	No
3	0b100x	0b1001	9 Volts	Yes
4	0b1001			

ADC Using Successive Approximation

- Example: 0V-16V range, 9.5V input, 4-bit resolution.

$$\frac{a - Vmin}{Vmax - Vmin} = \frac{d}{2^n}$$

$$\frac{9.5}{16 - 0} = \frac{d}{16}$$

$$d = \frac{9.5 * 16}{16} = 9.5$$

d = 0b1001 = 9 (Why not just use the above equation?)

ADC Using Successive Approximation

- Example: 0V-16V range, 9.5V input, 4-bit resolution.

$$\frac{a - Vmin}{Vmax - Vmin} = \frac{d}{2^n}$$

$$\frac{9.5}{16 - 0} = \frac{d}{16}$$

$$d = \frac{9.5 * 16}{16} = 9.5$$

d = 0b1001 = 9 (Why not just use the above equation?)

(i.e. Why is Successive Approximation needed?)

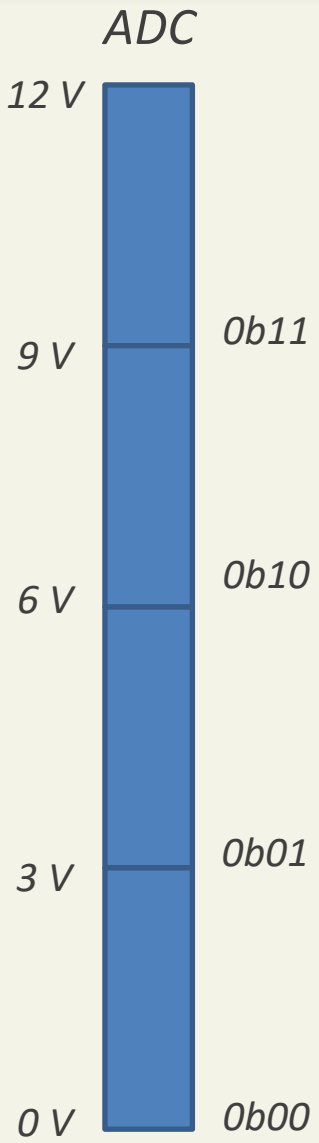
Practice Problem 1 (Linear equation)

- Example
  - 2-bits of resolution
  - Vmin = 0 Volts
  - Vmax = 12 Volts

$M = 2^n$

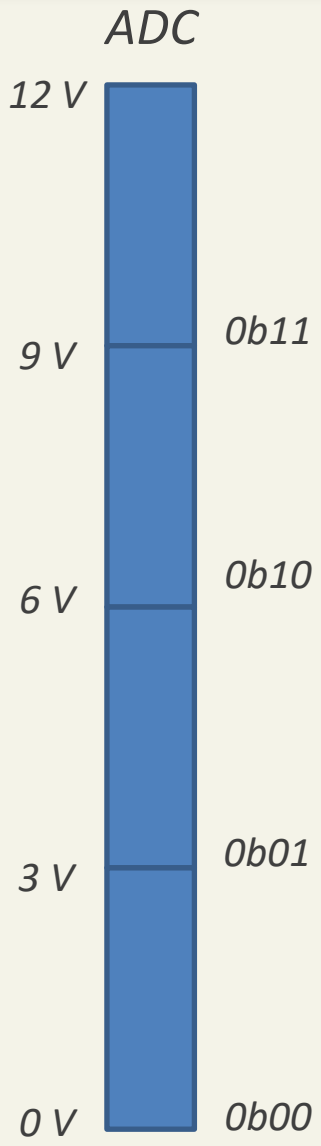
$2^2 = 4$ buckets (or ranges) for the analog signal to fall

r, **resolution**: analog change resulting from a digital change of 1



Practice Problem 1 (Linear equation)

- Question:
  - Given:
    - **n** = 2 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^n bins
  - What is **d**?



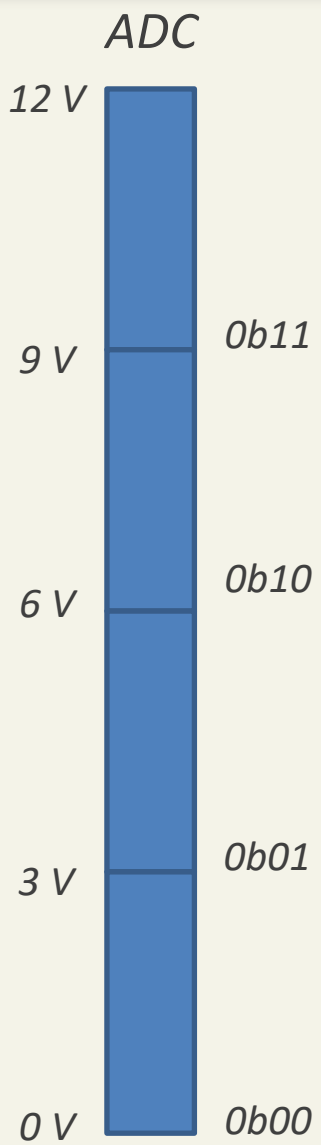
Practice Problem 1 (Linear equation)

- Question:
  - Given:
    - **n** = 2 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^n bins
  - What is **d**?

- Answer:
  - Analog and Digital signals are proportional

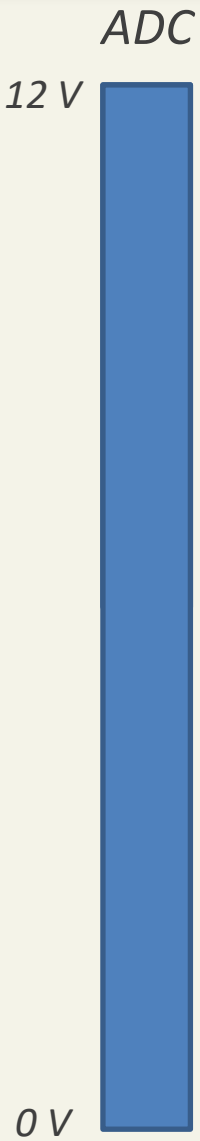
$$\frac{a - Vmin}{Vmax - Vmin} = \frac{d}{2^n}$$

  - **d** is 0b01



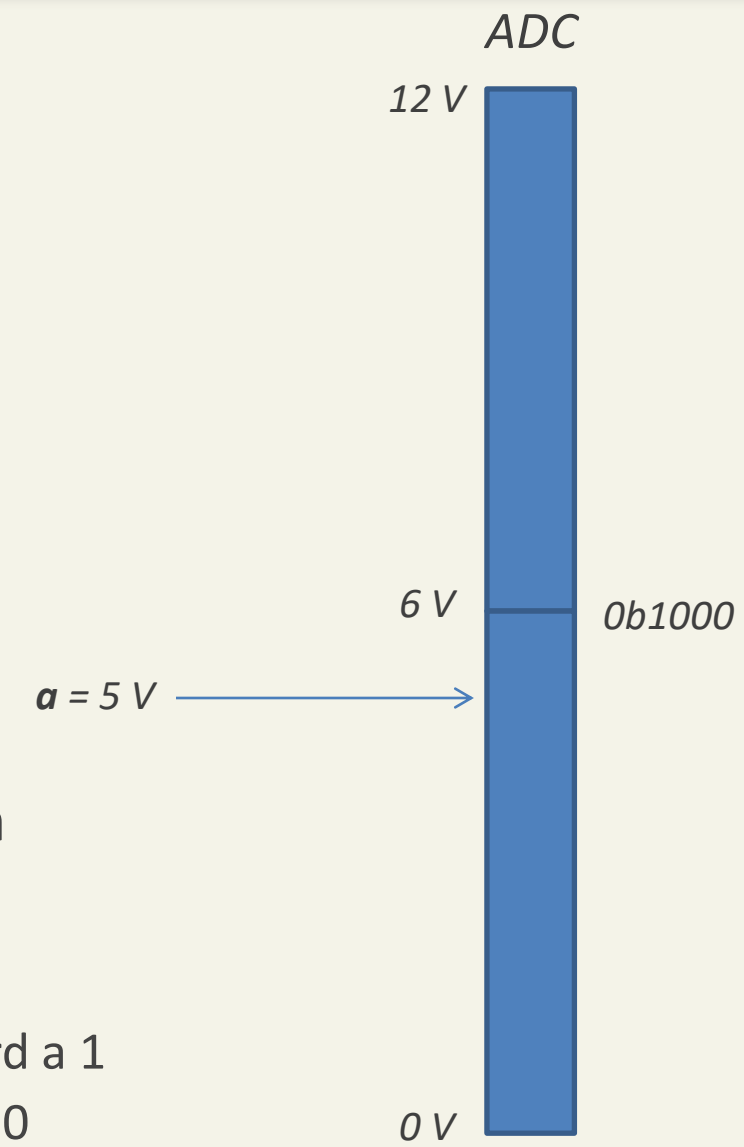
Practice Problem 2 (Successive Approximation)

- Question:
  - Given:
    - **n** = 4 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^n bins
  - What is **d**?



Practice Problem 2 (Successive Approximation)

- Question:
  - Given:
    - **n** = 4 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^n bins
  - What is **d**?

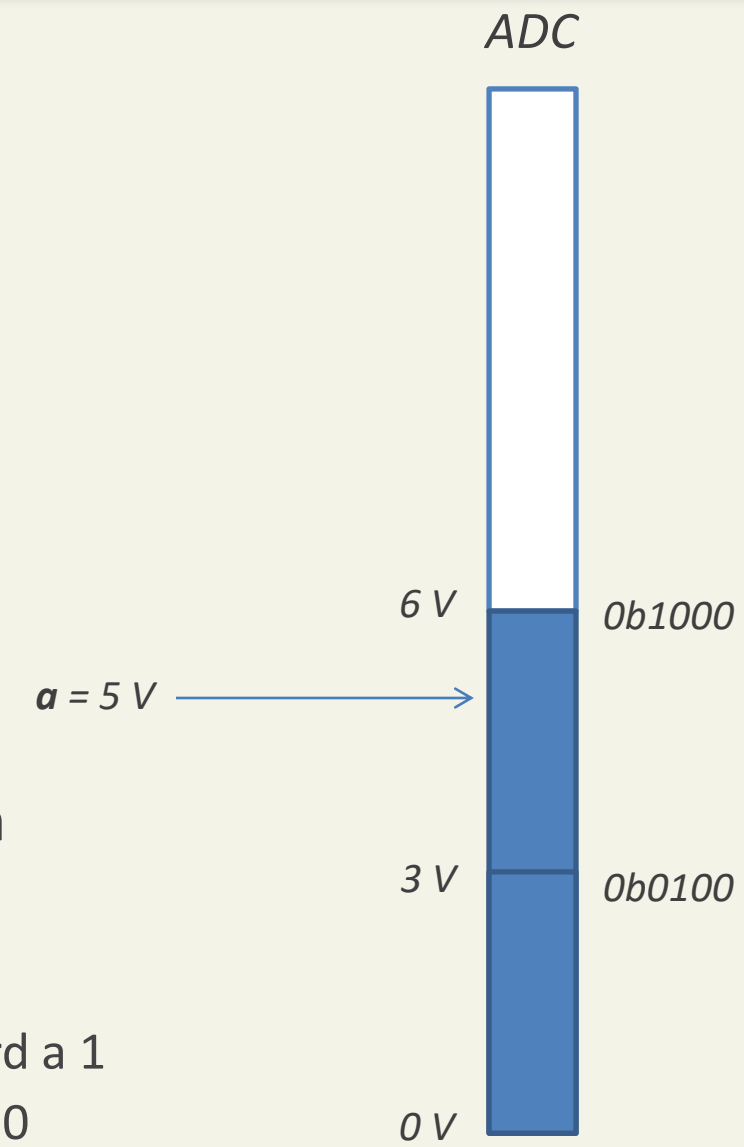


- Answer:
  - Use successive approximation
  - **d** = 0b**?**???
    - Midpoint is 0b1000 at 6 volts
    - If **a** is greater than 6 volts, record a 1
    - If **a** is less than 6 volts, record a 0

Practice Problem 2 (Successive Approximation)

- Question:
  - Given:
    - **n** = 4 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^n bins
  - What is **d**?

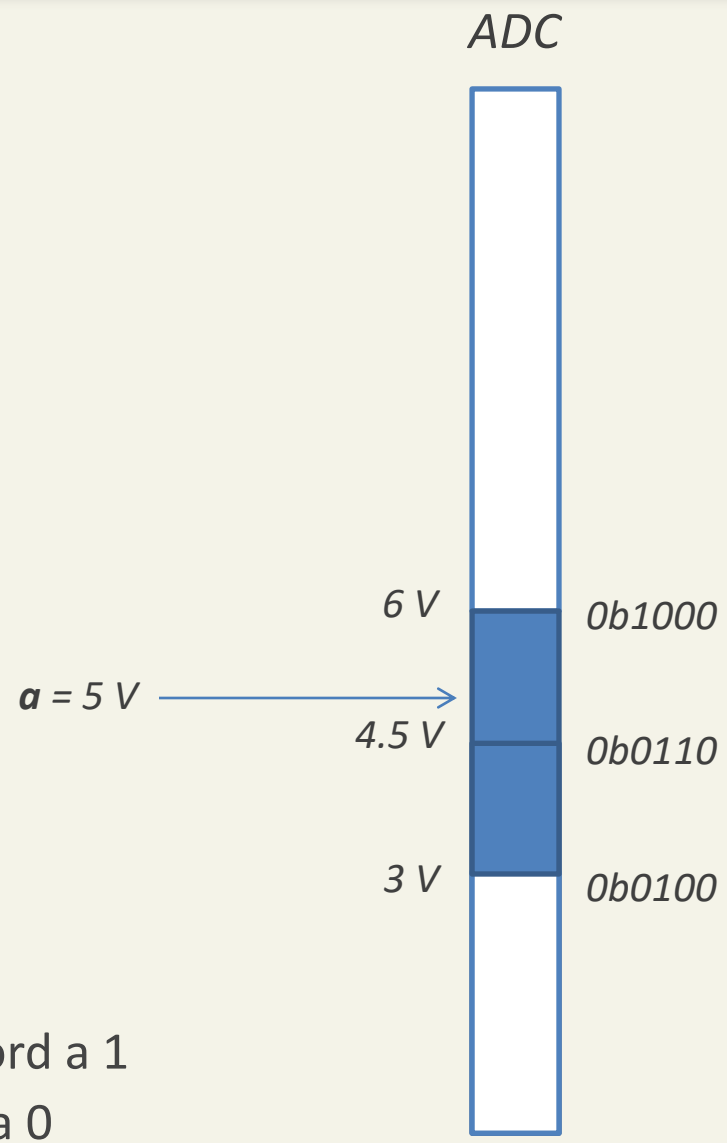
- Answer:
  - Use successive approximation
  - **d** = 0b0**?**??
    - Midpoint is 0b0100 at 3 volts
    - If **a** is greater than 3 volts, record a 1
    - If **a** is less than 3 volts, record a 0



Practice Problem 2 (Successive Approximation)

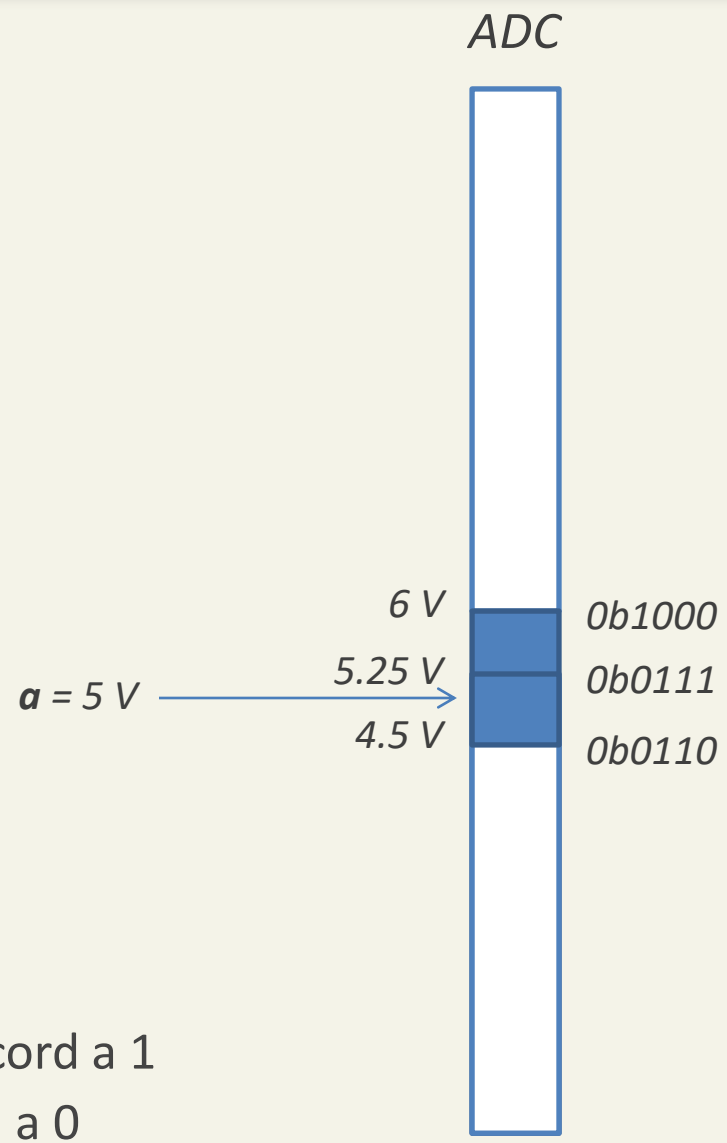
- Question:
  - Given:
    - **n** = 4 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^nbins
  - What is **d**?

- Answer:
  - Use successive approximation
  - **d** = 0b01**?**?
    - Midpoint is 0b0110 at 4.5 volts
    - If **a** is greater than 4.5 volts, record a 1
    - If **a** is less than 4.5 volts, record a 0



Practice Problem 2 (Successive Approximation)

- Question:
  - Given:
    - **n** = 4 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^nbins
  - What is **d**?

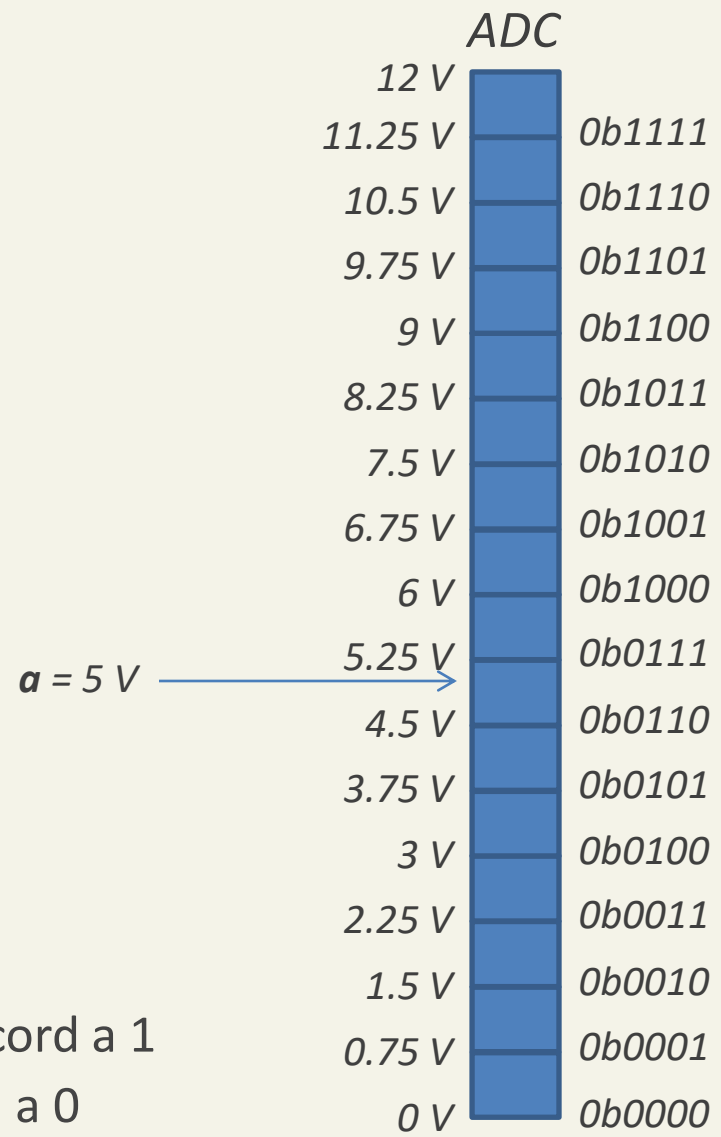


- Answer:
  - Use successive approximation
  - **d** = 0b011**?**
    - Midpoint is 0b0111 at 5.25 volts
    - If **a** is greater than 5.25 volts, record a 1
    - If **a** is less than 5.25 volts, record a 0

Practice Problem 2 (Successive Approximation)

- Question:
  - Given:
    - **n** = 4 bit resolution
    - **Vmin** = 0 volts
    - **Vmax** = 12 volts
    - **a** = 5 volts
    - **M** = 2^nbins
  - What is **d**?

- Answer:
  - Use successive approximation
  - **d** = 0b0110
    - Midpoint is 0b0111 at 5.25 volts
    - If **a** is greater than 5.25 volts, record a 1
    - If **a** is less than 5.25 volts, record a 0



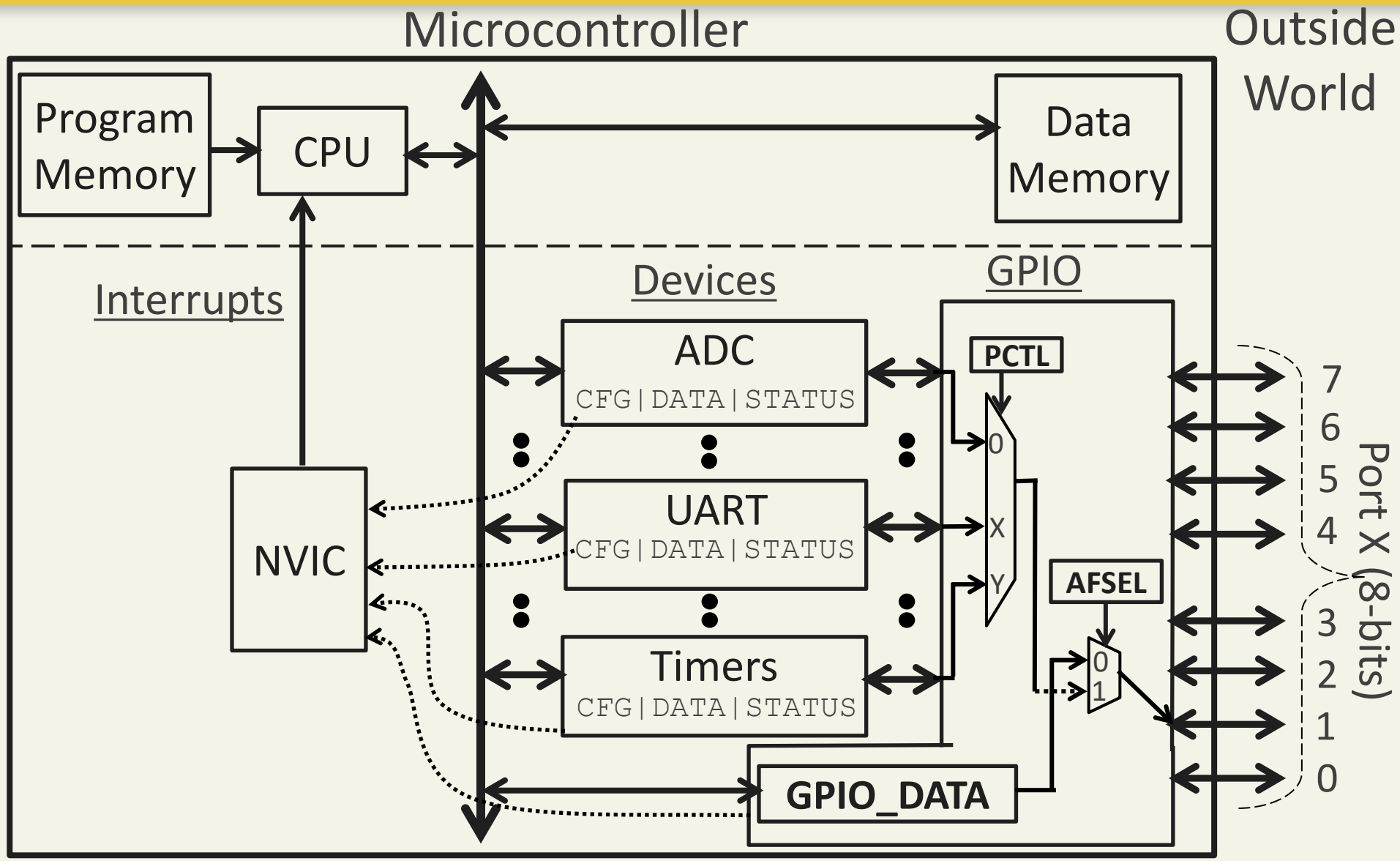
ADC ON TM4C123G

TM4C123G ADC

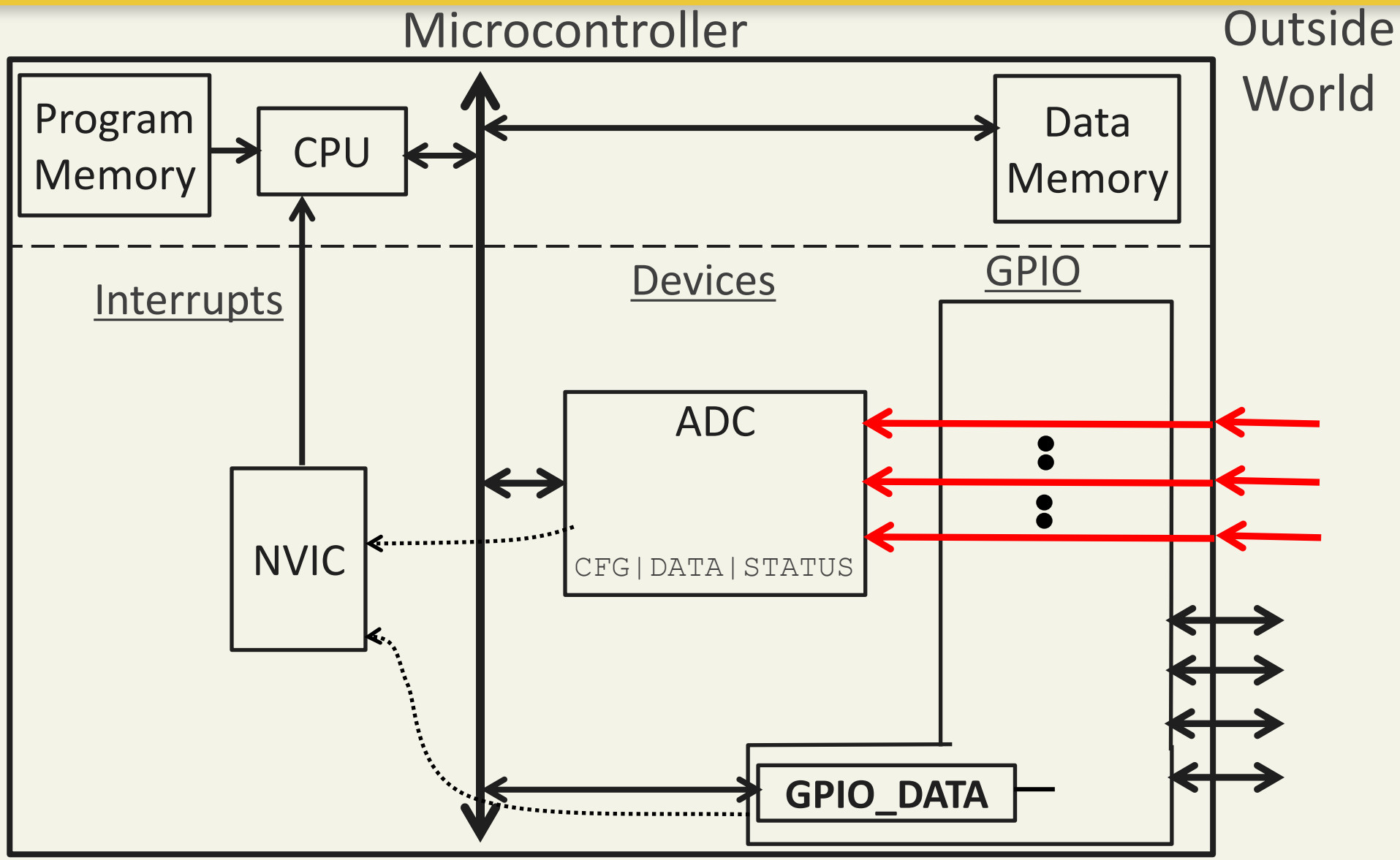
- 2 12-bit ADCs
- 12 shared input channels
- Up to 1M samples per second
- 0 – Vcc or 0 – 2.56V ADC input voltage range

- Which pins on the TM4C123G are used for the ADC?

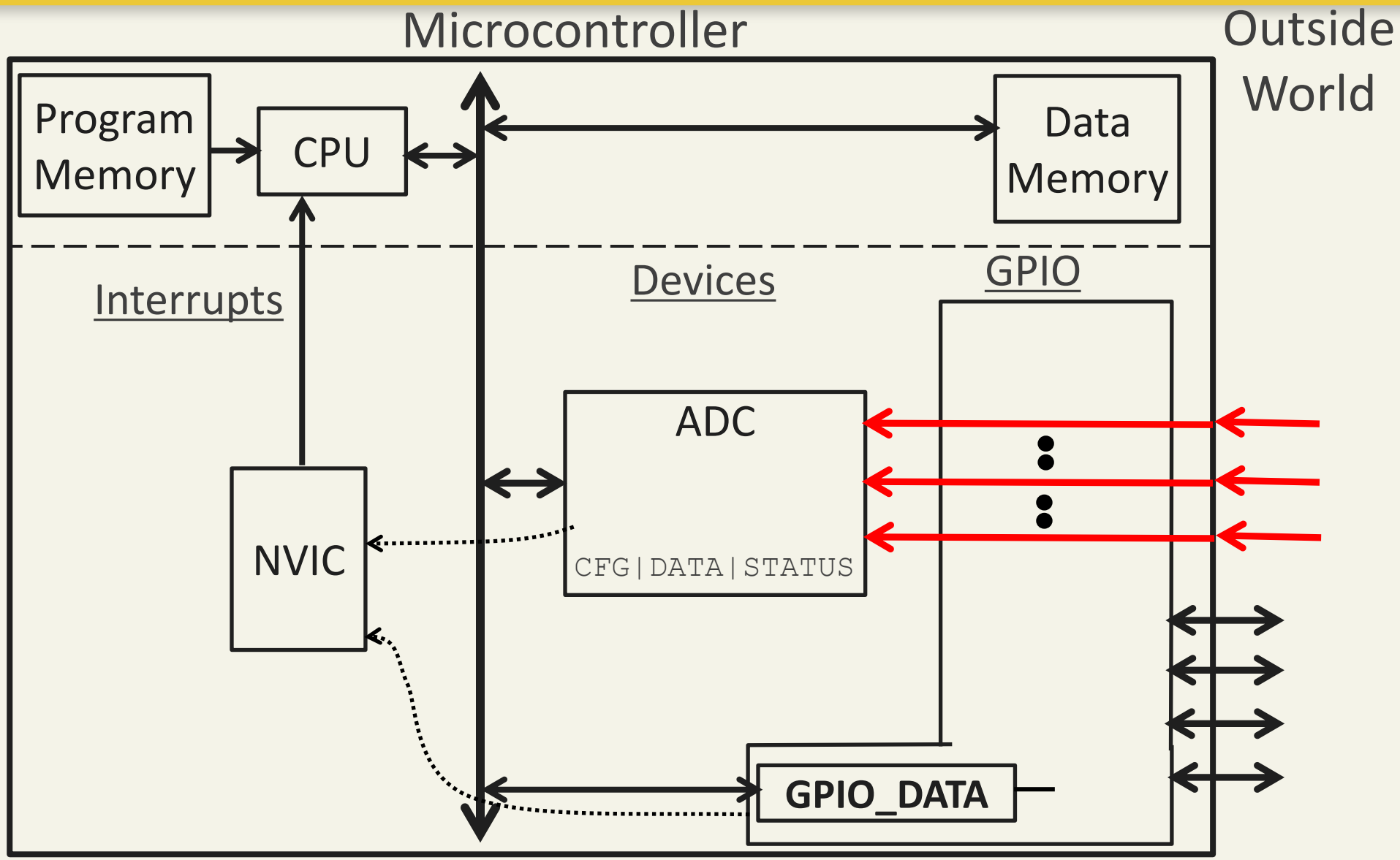
Microcontroller / System-on-Chip (SoC)



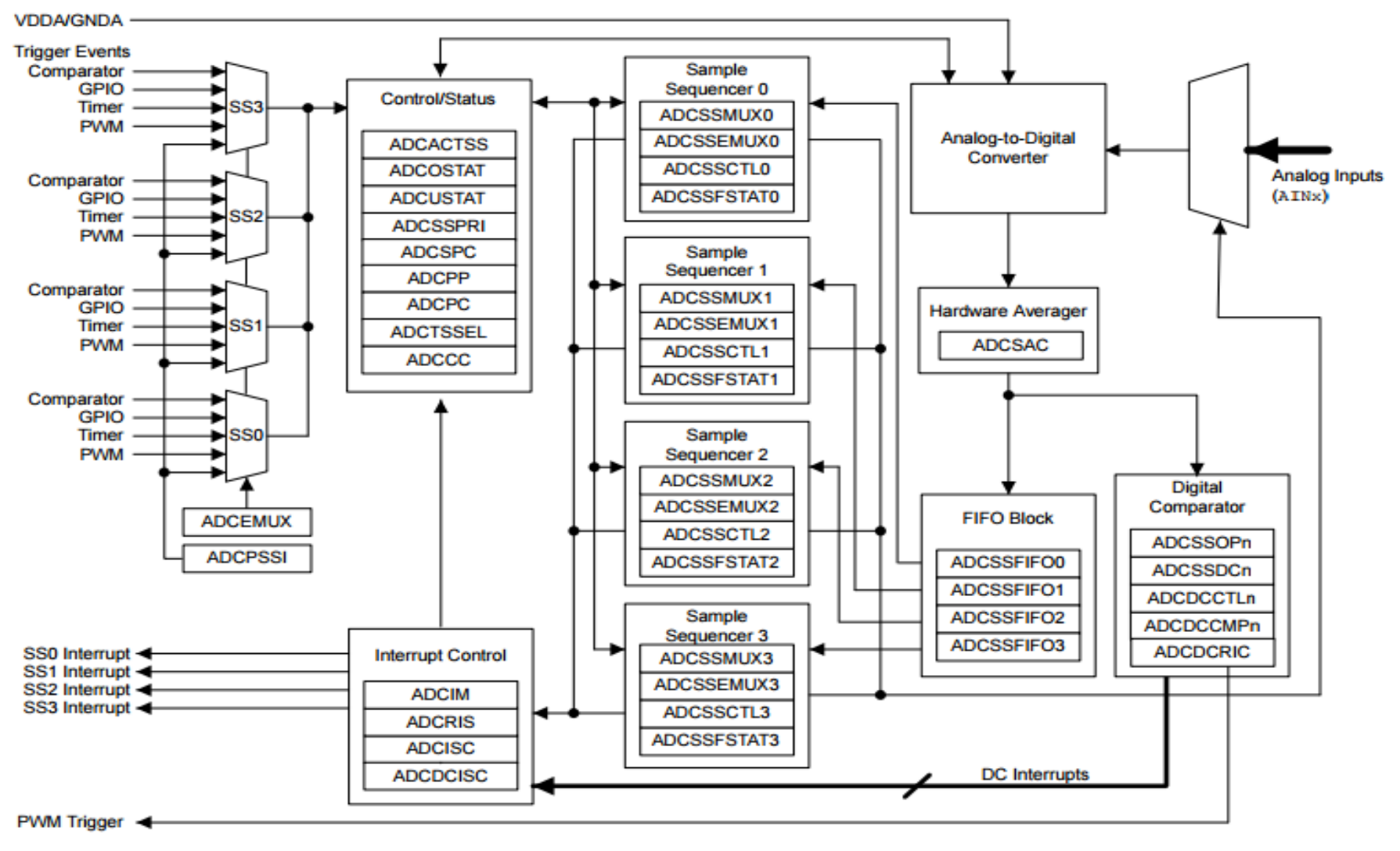
Microcontroller / System-on-Chip (SoC)



Microcontroller / System-on-Chip (SoC)



ADC Device Architecture



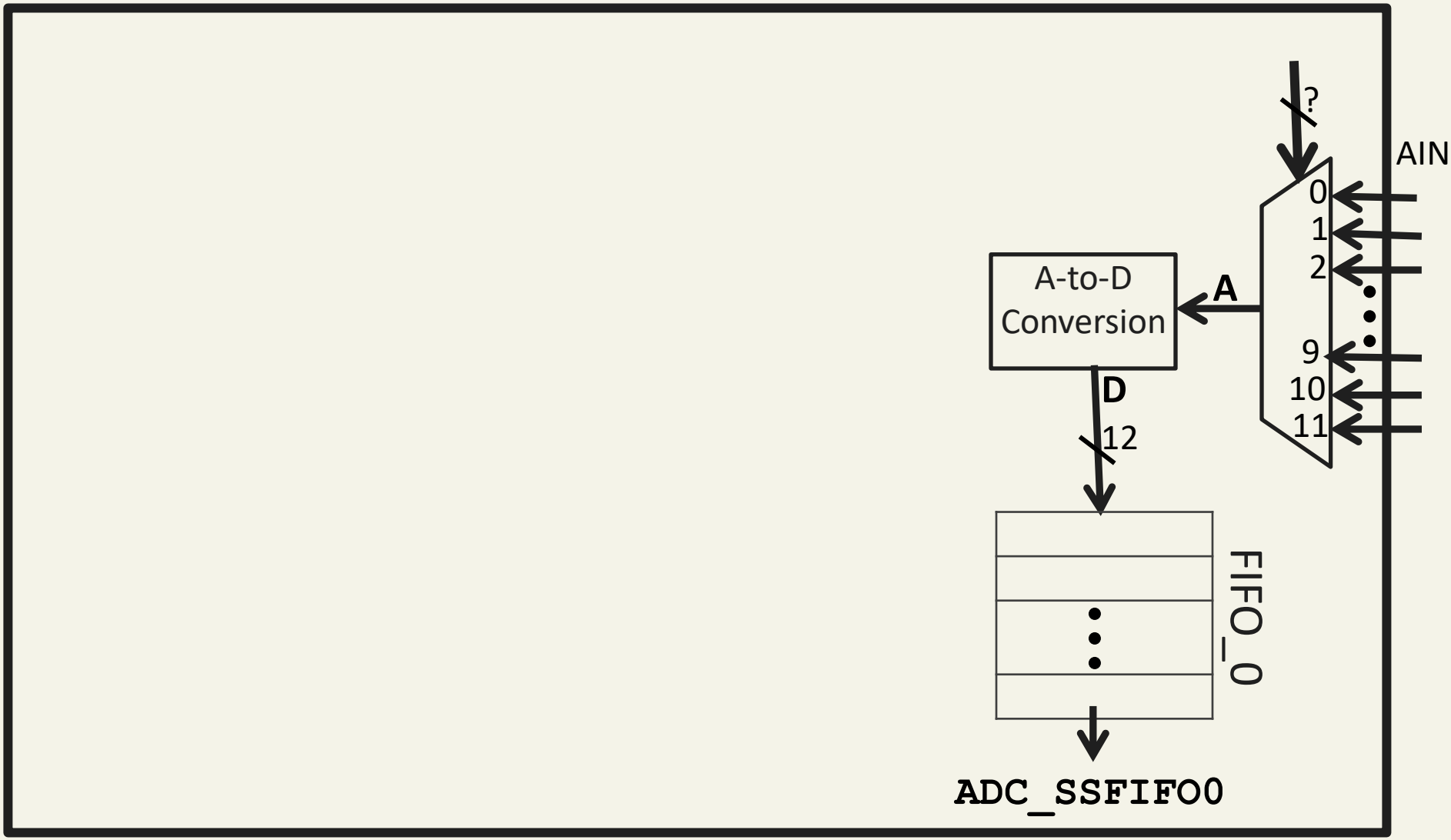
Datasheet Page:801

ADC Device Architecture

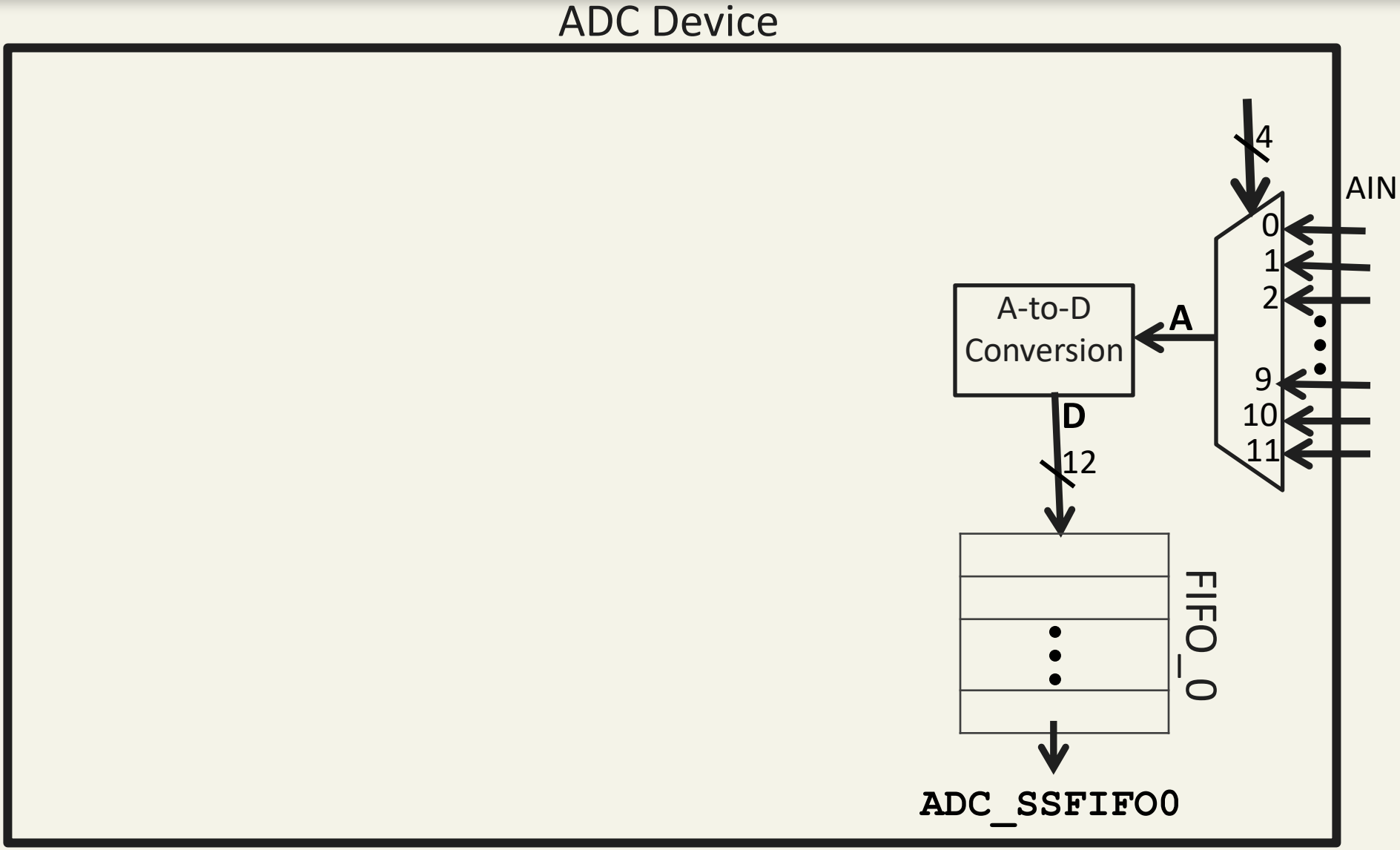
ADC Device

ADC Device Architecture

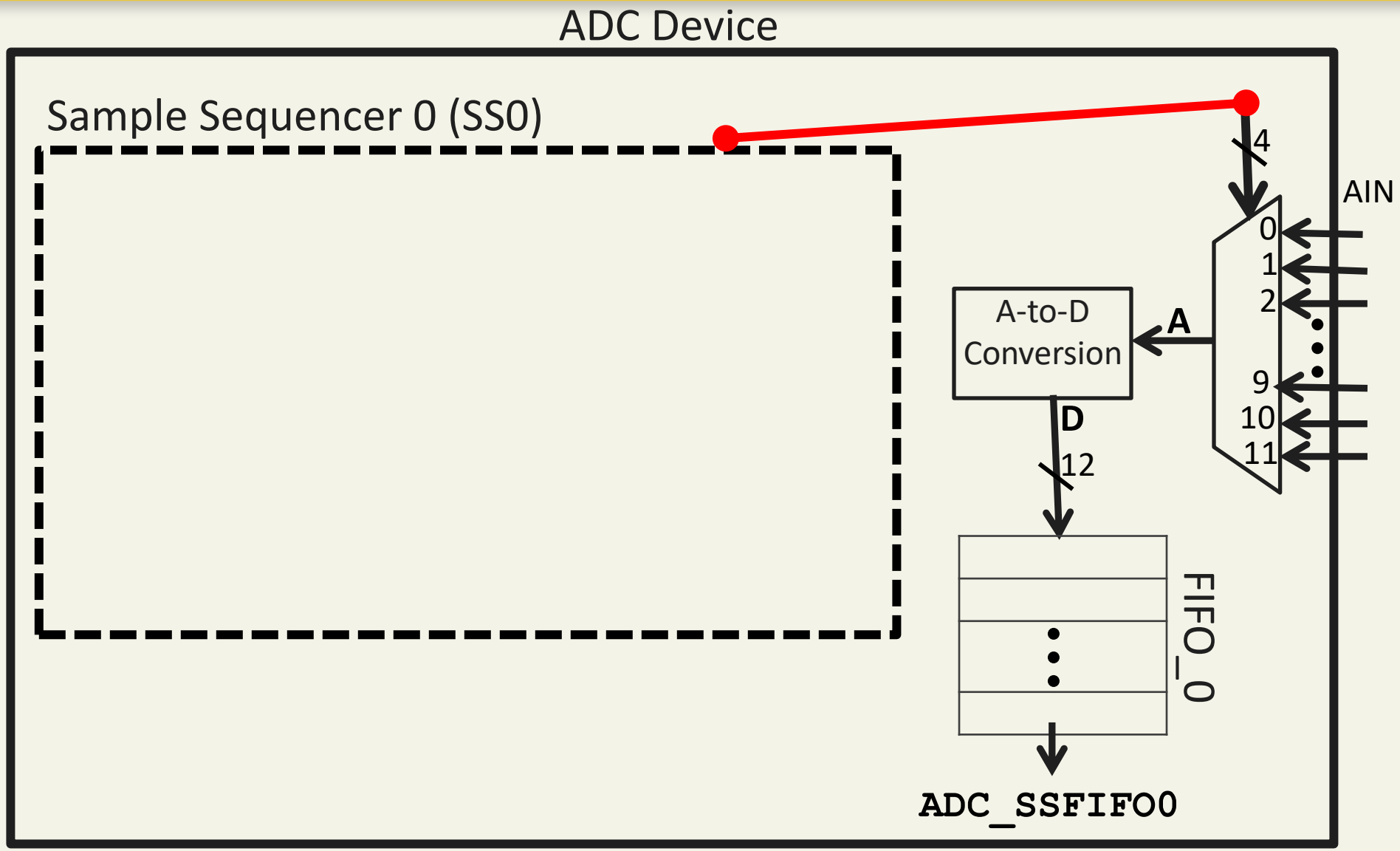
ADC Device



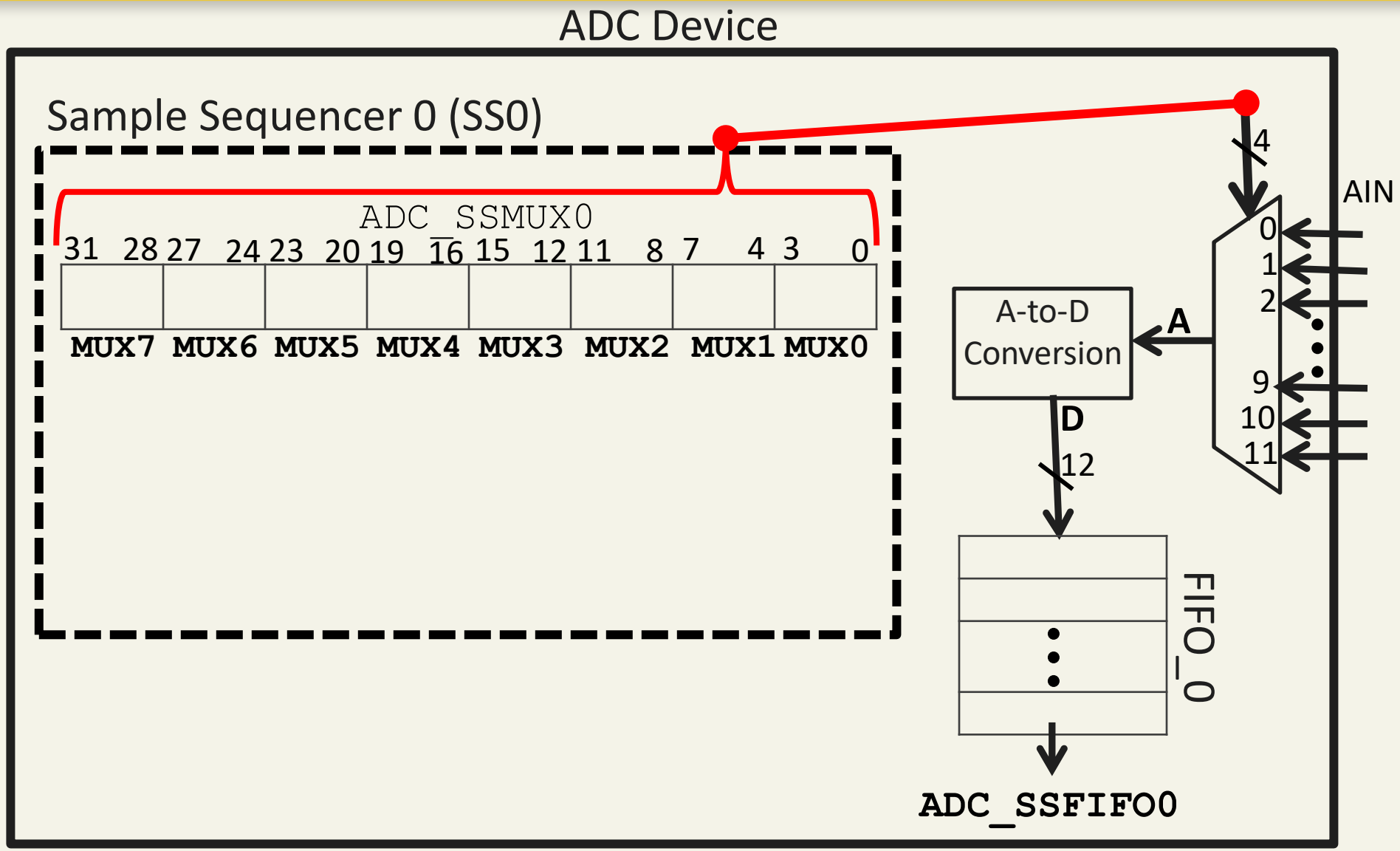
ADC Device Architecture



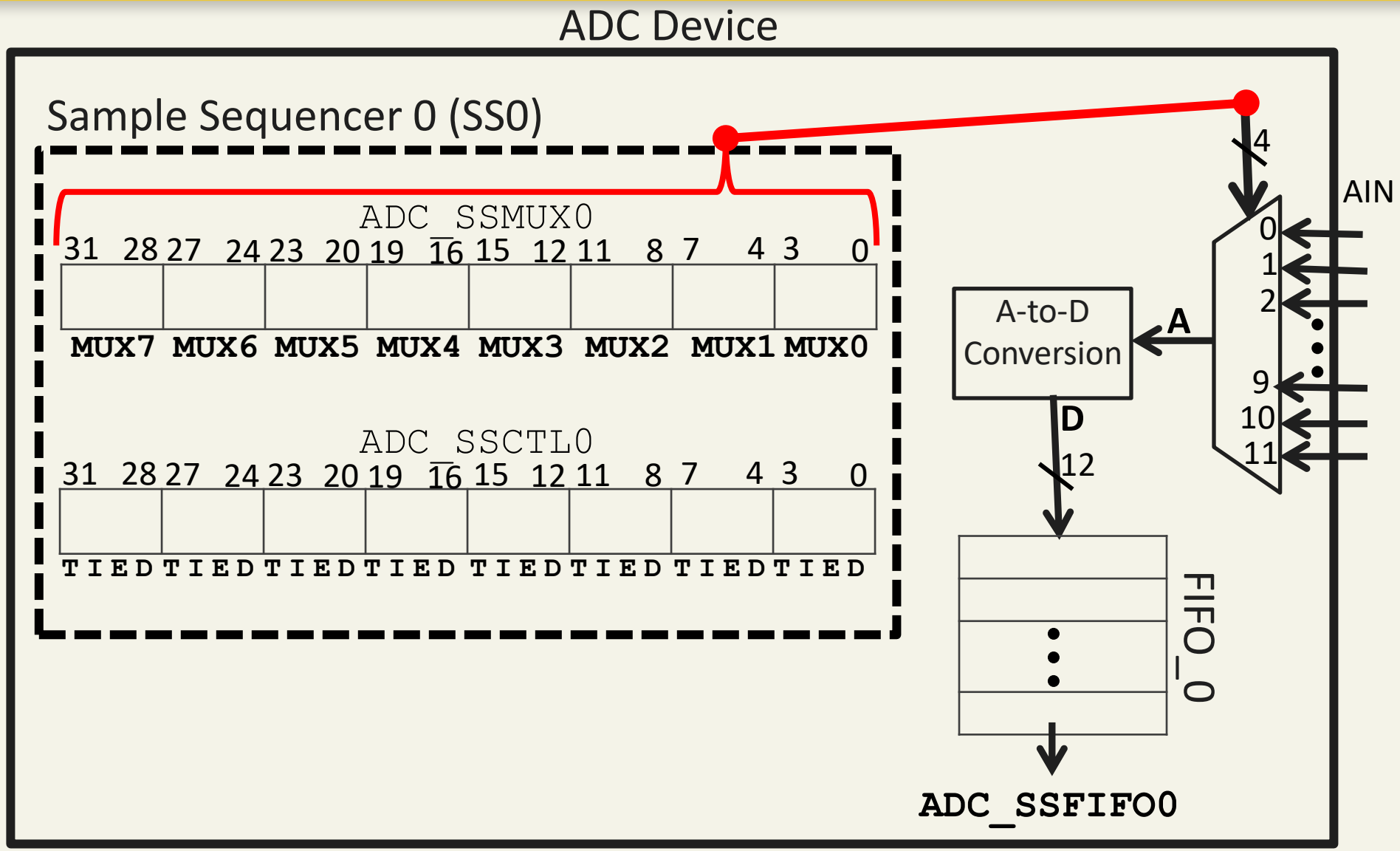
ADC Device Architecture



ADC Device Architecture



ADC Device Architecture



Programming Interface Registers

ADC GPIO Registers

RCGCGPIO General-Purpose Input/Output Run Mode Clock Gating Control

GPIOAFSEL GPIO Alternate Function Select

GPIODIR GPIO Direction

GPIODEN GPIO Digital Enable

GPIOAMSEL GPIO Analog Mode Select

GPIOADCCTL GPIO ADC Control

RCGCGPIO – GPIO run mode clock gating control

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 R5 – 0 disable port F, 1 provide clock to port F
4 R4 – 0 disable port E, 1 provide clock to port E
3 R3 – 0 disable port D, 1 provide clock to port D
2 R2 – 0 disable port C, 1 provide clock to port C
1 R1 – 0 disable port B, 1 provide clock to port B
0 R0 – 0 disable port A, 1 provide clock to port A

In lab we will provide clock to port B

GPIOAFSEL – Alternate function select

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								AFSEL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

7:0 AFSEL – 0 pin functions as normal GPIO, 1 pin works with alternate function

In lab we will set PB4 to alternate function

GPIODIR – GPIO direction

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DIR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7:0 DIR – 0 pin is input, 1 pin is output

In lab we will set PB4 to input

GPIODEN – GPIO digital enable

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

7:0 DEN – 0 digital function disabled, 1 digital function enabled

In lab we will disable PB4 digital function

GPIOAMSEL – Analog mode select

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								GPIOAMSEL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7:0 GPIOAMSEL – 0 analog function disabled, 1 analog function enabled

In lab we will enable PB4 analog function

GPIOADCCTL – ADC control

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								ADCEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7:0 ADCEN – 0 = pin not used to trigger ADC, 1 = pin is used to trigger ADC

In lab we will not be using any pins to trigger ADC conversion,
so this can be written 0x00, which should be the default value

ADC Registers

RCGCADC Analog-to-Digital Converter Run Mode Clock Gating Control

ADCACTSS ADC Active Sample Sequencer

ADCEMUX ADC Event Multiplexer Select

ADCPSSI ADC Processor Sample Sequence Initiate

ADCSSMUX ADC Sample Sequence Multiplexer Input Select

ADCSSCTL ADC Sample Sequence Control

ADCSSFIFO ADC Sample Sequence Result FIFO

ADCIM ADC Interrupt Mask

ADCRIS ADC Raw Interrupt Status

ADCISC ADC Interrupt Status and Clear

ADCSAC ADC Sample Averaging Control

Pg 819 of datasheet

RCGCADC – ADC Run Mode Clock Gating Control

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 R1 – 1 provide clock to ADC1, 0 disable ADC1

0 R0 – 1 provide clock to ADC0, 0 disable ADC0

ADCACTSS – Active Sample Sequencer

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															BUSY
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												ASEN3	ASEN2	ASEN1	ASEN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3 ASEN3 – 0 disable SS3, 1 enable SS3
2 ASEN2 – 0 disable SS2, 1 enable SS2
1 ASEN1 – 0 disable SS1, 1 enable SS1
0 ASEN0 – 0 disable SS0, 1 enable SS0

ADCEMUX – Event Multiplexor Select

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EM3				EM2				EM1				EM0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3:0 EM0 – SS0 trigger select
7:4 EM1 – SS1 trigger select
11:8 EM2 – SS2 trigger select
15:12 EM3 – SS3 trigger select

In lab we will be using SS1. EM1 will be set to 0x0 using the default trigger, which means your C program must write to a memory mapped register to start a conversion.

ADCPSSI – Processor Sample Sequence Initiate

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	GSYNC	reserved			SYNCWAIT	reserved										
Type	RW	RO	RO	RO	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												SS3	SS2	SS1	SS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-

31 GSYNC – global synchronize, 1 will start all initialized and syncwait ADCs,
0 cleared once initiated

27 SYNCWAIT – synchronize wait, 1 allows for gsync to initiate all initialized ADCs

3 SS3 – SS3 initiate, 1 begins sampling SS3

2 SS2 – SS2 initiate, 1 begins sampling SS2

1 SS1 – SS1 initiate, 1 begins sampling SS1

0 SS0 – SS0 initiate, 1 begins sampling SS0

Used to trigger the start of a conversion

ADCSSMUXn – Sample Sequence Mux Input Select

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX7				MUX6				MUX5				MUX4			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX3				MUX2				MUX1				MUX0			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31:28 MUX7 – Select 8th sample to convert in the sample sequence.
27:23 MUX6 – Select 7th sample to convert in the sample sequence.
23:20 MUX5 – Select 6th sample to convert in the sample sequence.
19:16 MUX4 – Select 5th sample to convert in the sample sequence.
15:12 MUX3 – Select 4th sample to convert in the sample sequence.
11:8 MUX2 – Select 3rd sample to convert in the sample sequence.
7:4 MUX1 – Select 2nd sample to convert in the sample sequence.
3:0 MUX0 – Select 1st sample to convert in the sample sequence.

Note this is for **SS0 though others will be nearly **identical check the datasheet**

ADCSSCTLn – Sample Sequence Control

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3 TS0 – 1 if 1st sample should be temp sensor, 0 if ADCSSMUX should be used
2 IE0 – 1 if 1st sample raw interrupt enable, else 0
1 END0 – 1 if 1st sample is end of sequence, else 0
0 D0 – 1st sample should use differential ended input

Note: In lab we will poll the raw interrupt status register

*Note this specific register is for **SS1** though others will be nearly identical **check the datasheet**

ADCSSFIFOn – Sample Sequence result FIFO

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				DATA											
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-

11:0 DATA – contains the conversion result for SSn

ADCIM – ADC Interrupt Mask

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved												DCONSS3	DCONSS2	DCONSS1	DCONSS0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												MASK3	MASK2	MASK1	MASK0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3 MASK3 – SS3 interrupt mask; 1 SS3 RIS sent to interrupt controller, 0 not sent
2 MASK2 – SS2 interrupt mask, 1 SS2 RIS sent to interrupt controller, 0 not sent
1 MASK1 – SS1 interrupt mask, 1 SS1 RIS sent to interrupt controller, 0 not sent
0 MASK0 – SS0 interrupt mask, 1 SS0 RIS sent to interrupt controller, 0 not sent

ADCRIS – Raw Interrupt Status

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															INRDC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												INR3	INR2	INR1	INR0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3 INR3 – SS3 raw interrupt status, 1 means conversion complete on sample sequencer
2 INR2 – SS2 raw interrupt status, 1 means conversion complete on sample sequencer
1 INR1 – SS1 raw interrupt status, 1 means conversion complete on sample sequencer
0 INR0 – SS0 raw interrupt status, 1 means conversion complete on sample sequencer

In lab can use this to check when conversions are complete

ADCISC – Interrupt Status and Clear

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												DCINSS3	DCINSS2	DCINSS1	DCINSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

19:16 DCINSSn – digital comparator interrupt status on SSn

3:0 INn – Interrupt Status and clear if read 1 ADCRIS and MASKn set,
write 1 to clear interrupt

ADCSAC – Sample Averaging Control

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													AVG		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2:0 AVG –

0x0 = No oversample
0x1 = 2x hardware oversample
0x2 = 4x hardware oversample
0x3 = 8x hardware oversample
0x4 = 16x hardware oversample
0x5 = 32x hardware oversample
0x6 = 64x hardware oversample
0x7 = reserved