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

Dr. Phillip Jones
Announcements

• Exam 1: In class Thursday 2/25
• Homework 5: not collected or graded
• Lab this week: Lab 6, ADC IR distance
Week 7 Overview

• ADC general knowledge

• Atmega128 ADC programming interface

• Write ADC-related functions
  – Initialize/configure ADC
  – Read from ADC
  – ADC interrupt programming
Template Functions

• At the end, you will be able to write ADC functions like the follows:

```c
void ADC_init()
{
    // Reference voltage, alignment, channel
    ADMUX = _____;

    // Enable, running mode, interrupt,
    // and clock select
    ADCSRA = __________;
}
```
Template Functions

// Single-shot ADC reading
void ADC_read(int channel)
{
    // set up channel
    ADMUX = _____;  // Select channel
    ADCSRA |= __________;  // Start ADC sampling
    while (ADCSRA | _____) // Wait until done
    {
    }

    return ADCW;  // Read and return
        // the result
}
Looking Forward

• There are generally three phases of the course:
  1. C Programming
  2. I/O Programming
  3. Assembly programming

• The 2\textsuperscript{nd} and 3\textsuperscript{rd} phases are much more challenging than the 1\textsuperscript{st} phase
  – From now, do not miss class meetings!
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
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 ATMega128 has a 10-bit ADC

• **digital-to-analog converter** (DAC, D/A, D2A): converts a n-bit digital input signal to an analog output signal
• **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

Temperature Sensor

- Sensor output
- A/D input

Temperature vs. Voltage (Sensor Specification)

- T_min = 0 C
- T_max = 200 C

Sensor Input (T)

- Sensor_Vmin = 0
- Vmax = 3.3V

- A/D_Vmin = 0 V
- A/D_Vmax = 3.3V

A/D: Analog Input vs. Digital Output

- (M = 2^n-1 steps (or bins): Dmax = Vmax)
- A/D: 10-bit
- Dmax = 1023
Sensitivity: Analog Sensor (Linear)

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

Temperature vs. Voltage

- $T_{\text{max}} = 200 \degree \text{C}$
- $T_{\text{min}} = 0 \degree \text{C}$
- $V_{\text{min}} = 0$
- $V_{\text{max}} = 3.3 \text{V}$

Analog Sensor Output (V)

Sensor Input (Temperature (C))

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:

\[ m = \text{slope} = \frac{\text{RISE}}{\text{RUN}} = \frac{T_{\text{max}} - T_{\text{min}}}{V_{\text{max}} - V_{\text{min}}} = \frac{200 - 0}{3.3 - 0} = 60.61 \text{ C/V} \]
Analog Sensor: Compute Analog Output

- **Question:** If the measured Temperature is 100°C 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)
• Question: If the measured Temperature is 100°C 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

\[ 100 = 60.61X; \] \[ X = \frac{100}{60.61} = 1.65 \text{ V} \]
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 / 1023 – 0) = 0.0032 V/bit

LSB bit weight = 0.0032 V/bit
• 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
• 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 \( m = 0.0032 \), so

\[ 1.65 = 0.0032X; X = 1.65/0.0032 = 515.625 \rightarrow \text{Truncate to } 515 = 0b10_000_0011 \]

• For this example, a Temperature of 100 C gives a digital value of 515
A/D: Compute Digital output

A/D: Analog Input vs. Digital Output
(For $M = 2^{n}-1$ steps (or bins), $D_{\text{max}} = V_{\text{max}}$)

- Question: What is the Temperature LSB weight?

Sensor output  A/D input
A/D 10-bit
Digital output

V_{\text{min}} = 0 \text{ V}
D = 0

V_{\text{max}} = 3.3 \text{ V}

10-bit
Digital Output (D)
$D_{\text{max}} = 1023$
**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** = .0032 V/bit
- **We want C/bit:** So 60.61 C/V * .0032 V/bit = .194 C/bit

A change of 1 in the digital output, corresponds to a change .194 degrees
ADC Bit Weight (a closer look)

LSB bit weight in the last example: bit 0 = .0032V, 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: 515
Binary: 1 0 0 0 0 0 0 0 1 1
1.638 + .0064 + 0032 ~= 1.65 V

<table>
<thead>
<tr>
<th>Digital Bit</th>
<th>Bit Weight (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>512*r = 1.638</td>
</tr>
<tr>
<td>8</td>
<td>256*r = 0.8192</td>
</tr>
<tr>
<td>7</td>
<td>128*r = 0.4096</td>
</tr>
<tr>
<td>6</td>
<td>64*r = 0.2048</td>
</tr>
<tr>
<td>5</td>
<td>32*r = 0.1024</td>
</tr>
<tr>
<td>4</td>
<td>16*r = 0.0512</td>
</tr>
<tr>
<td>3</td>
<td>8*r = 0.0256</td>
</tr>
<tr>
<td>2</td>
<td>4*r = 0.0128</td>
</tr>
<tr>
<td>1</td>
<td>2*r = 0.0064</td>
</tr>
<tr>
<td>0</td>
<td>r = 0.0032</td>
</tr>
</tbody>
</table>
What if $V_{\text{max}} = 1.65$, and $V_{\text{min}} = -1.65$?

A/D: Analog Input vs. Digital Output
(For $M = 2^n - 1$ steps (or bins), $D_{\text{max}} = V_{\text{max}}$)

$V_{\text{max}} = 1.65V$

$V_{\text{min}} = -1.65V$

$D = 0$

10-bit

$D_{\text{max}} = 1023$

Digital Output ($D$)
What if $V_{\text{max}} = 1.65$, and $V_{\text{min}} = -1.65$?

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

Thus, the LSB bit weight is still: bit 0 = .0032V, (the resolution)

But now you must add an offset of $V_{\text{min}}$. (Can derive from the equation for a line)

For modified example:
Decimal: 515
Binary: 1 0 0 0 0 0 0 0 1 1

1.638 + .0064 + 0032 ~ = 1.65 V
+ -1.65 V
  0 V

<table>
<thead>
<tr>
<th>Digital Bit</th>
<th>Bit Weight (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>512*r = 1.638</td>
</tr>
<tr>
<td>8</td>
<td>256*r = 0.8192</td>
</tr>
<tr>
<td>7</td>
<td>128*r = 0.4096</td>
</tr>
<tr>
<td>6</td>
<td>64*r = 0.2048</td>
</tr>
<tr>
<td>5</td>
<td>32*r = 0.1024</td>
</tr>
<tr>
<td>4</td>
<td>16*r = 0.0512</td>
</tr>
<tr>
<td>3</td>
<td>8*r = 0.0256</td>
</tr>
<tr>
<td>2</td>
<td>4*r = 0.0128</td>
</tr>
<tr>
<td>1</td>
<td>2*r = 0.0064</td>
</tr>
<tr>
<td>0</td>
<td>r = 0.0032</td>
</tr>
</tbody>
</table>
Analog-to-digital converter: Example

Temperature Sensor

- Sensor output
- 1.65V
- A/D input
- Digital output

1.65V

A/D: Analog Input vs. Digital Output
\( M = 2^n - 1 \) steps (or bins): 
\[ D_{\max} = V_{\max} \]

A/D Input (V)

- \( A/D_{\text{V}_{\text{max}}} = 3.3V \)
- \( A/D_{\text{V}_{\text{min}}} = 0 \text{ V} \)

Digital Output (D)

- \( D = 0 \)
- \( D_{\max} = 1023 \)
- 10-bit

Temperature vs. Voltage
(Sensor Specification)

- \( T_{\max} = 200 \text{ C} \)
- \( T_{\min} = 0 \text{ C} \)

Slope = Sensitive
- \( 60.61 \text{ C/V} \)

Sensor output

- \( 100 \text{ C} \)
- 1.65V

100 C

Analog Sensor Output (V)

- \( \text{Sensor}_{\text{V}_{\text{min}}} = 0 \text{ V} \)
- \( \text{V}_{\max} = 3.3V \)

Temperature Sensor

Slope = Resolution
- \( .0032 \text{ V/bit} \)

Sensor input (T)

- \( \text{Slope} = 60.61 \text{ C/V} \)

- \( \text{Slope} = 0.0032 \text{ V/bit} \)
Analog-to-digital converters (Usage)

Mapping between Analog and Digital

\[ V_{\text{max}} = 7.5V \]

<table>
<thead>
<tr>
<th>Analog Input (V)</th>
<th>Digital Output (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0V</td>
<td>1111</td>
</tr>
<tr>
<td>6.5V</td>
<td>1110</td>
</tr>
<tr>
<td>6.0V</td>
<td>1101</td>
</tr>
<tr>
<td>5.5V</td>
<td>1110</td>
</tr>
<tr>
<td>5.0V</td>
<td>1011</td>
</tr>
<tr>
<td>4.5V</td>
<td>1010</td>
</tr>
<tr>
<td>4.0V</td>
<td>1001</td>
</tr>
<tr>
<td>3.5V</td>
<td>1000</td>
</tr>
<tr>
<td>3.0V</td>
<td>0111</td>
</tr>
<tr>
<td>2.5V</td>
<td>0110</td>
</tr>
<tr>
<td>2.0V</td>
<td>0101</td>
</tr>
<tr>
<td>1.5V</td>
<td>0111</td>
</tr>
<tr>
<td>1.0V</td>
<td>0100</td>
</tr>
<tr>
<td>0.5V</td>
<td>0011</td>
</tr>
<tr>
<td>0V</td>
<td>0010</td>
</tr>
</tbody>
</table>

proportionality

Digital sampling of an analog signal

Digital generation of an analog signal

Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis
Analog-to-digital converters (Usage)

Mapping between Analog and Digital

$V_{\text{max}} = 7.5 \text{V}$

1111

1110

1101

1100

1011

1010

1001

1000

0111

0110

0101

0100

0011

0010

0001

0000

Vmin = 0V

D = 0

(0000)

Digital Output (D)

Dmax = 15 (1111)

Vmax = 7.5V

3.5V

analog to digital

Digital sampling of an analog signal

analog input (V)

Digital output

0100 0111 0110 0101

t1 t2 t3 t4

time

digital to analog

Digital generation of an analog signal

analog output (V)

0100 1000 0110 0101

t1 t2 t3 t4

time

Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis
Analog-to-digital converters (Usage)

Mapping between Analog and Digital

V_{max} = 7.5V
7.0V  6.5V  6.0V  5.5V  5.0V  4.5V  4.0V  3.5V  3.0V  2.5V  2.0V  1.5V  1.0V  0.5V  0V

Digital output

0100 0111 0110 0101

analog input (V)

5.0V  5.5V  6.0V  6.5V  7.0V

4.5V  4.0V  3.5V  3.0V

3.0V  2.5V  2.0V  1.5V  1.0V

0.5V  0V

proportionality

0V

4-bit

D=0 (0000) Digital Output (D) D_{max}=15 (1111)

Digital sampling of an analog signal

analog to digital

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

Digital generation of an analog signal
digital to analog
Analog-to-digital converters (Usage)

Mapping between Analog and Digital

![Digital sampling of an analog signal](image1)

![Digital generation of an analog signal](image2)

**Analog-to-digital**

- **$V_{max} = 7.5V$**
- **$V_{min} = 0V$**
- **$D_{max} = 15 (1111)$**

**Digital to analog**

- **4-bit**
- **$V_{max} = 7.5V$**
- **$V_{min} = 0V$**

*Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis*
Analog-to-digital converters (Usage)

Mapping between Analog and Digital

\[ V_{\text{max}} = 7.5V \]

<table>
<thead>
<tr>
<th>Digital Output (D)</th>
<th>Analog Input (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0V</td>
</tr>
<tr>
<td>0001</td>
<td>0.5V</td>
</tr>
<tr>
<td>0010</td>
<td>1.0V</td>
</tr>
<tr>
<td>0011</td>
<td>1.5V</td>
</tr>
<tr>
<td>0100</td>
<td>2.0V</td>
</tr>
<tr>
<td>0101</td>
<td>2.5V</td>
</tr>
<tr>
<td>0110</td>
<td>3.0V</td>
</tr>
<tr>
<td>0111</td>
<td>3.5V</td>
</tr>
<tr>
<td>1000</td>
<td>4.0V</td>
</tr>
<tr>
<td>1001</td>
<td>4.5V</td>
</tr>
<tr>
<td>1010</td>
<td>5.0V</td>
</tr>
<tr>
<td>1011</td>
<td>5.5V</td>
</tr>
<tr>
<td>1100</td>
<td>6.0V</td>
</tr>
<tr>
<td>1101</td>
<td>6.5V</td>
</tr>
<tr>
<td>1110</td>
<td>7.0V</td>
</tr>
<tr>
<td>1111</td>
<td>7.5V</td>
</tr>
</tbody>
</table>

Proportionality

**Digital sampling of an analog signal**

**Digital generation of an analog signal**

\[ V_{\text{max}} = 7.5V \]

\[ V_{\text{min}} = 0V \]

Digital Output (D)

\[ D=0 \] (0000)

4-bit

Digital Input (D)

\[ D_{\text{max}} = 15 \] (1111)

**Embedded Systems Design: A Unified Hardware/Software Introduction**, (c) 2000 Vahid/Givargis
Proportional Signals (Simple case)

Simple Equation

Assume $V_{\text{min}} = 0$ V.

$V_{\text{max}} = \text{maximum voltage of the analog signal}$

$a = \text{analog value}$

$n = \text{number of bits for digital encoding}$

$2^n = \text{number of digital codes}$

$M = \text{number of steps, either } 2^n \text{ or } 2^n - 1$

$d = \text{digital encoding}$

$$a / V_{\text{max}} = d / M$$

This is derived from the equation for a line

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

$$Y = m \times X + b$$
General Equation

Do not assume \( V_{\text{min}} = 0 \) V.

\( V_{\text{max}} \) = 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

\[
\frac{(a-V_{\text{min}})}{(V_{\text{max}}-V_{\text{min}})} = \frac{d}{M}
\]

This is derived from the equation for a line

\[
a = \left[ \frac{(V_{\text{max}} - V_{\text{min}})}{(2^n-1-0)} \right] * d + V_{\text{min}}
\]

\[
Y = m * X + b
\]
Resolution: \( M = 2^n - 1 \) vs. \( 2^n \)

Let \( n = 2 \)

**\( M = 2^n - 1 \)**

3 steps on the digital scale

- \( d_0 = 0 = 0b00 \)
- \( d_{V_{\text{max}}} = 3 = 0b11 \)

**\( M = 2^n \)**

4 steps on the digital scale

- \( d_0 = 0 = 0b00 \)
- \( d_{V_{\text{max}} - r} = 3 = 0b11 \) (no \( d_{V_{\text{max}}} \), 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 $n = 2$

$M = 2^n - 1$

3 steps on the digital scale

$d_0 = 0 = 0b00$

$d_{V_{\text{max}}} = 3 = 0b11$

$r = 4V$

$V_{\text{max}} = 12V$

$V_{\text{min}} = 0V$

$M = 2^n$

4 steps on the digital scale

$d_0 = 0 = 0b00$

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

$r = 3V$

$V_{\text{max}} = 12V$

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

$V_{\text{min}} = 0V$
Resolution: $M = 2^n - 1$ vs. $2^n$

Let $n = 2$

$M = 2^n - 1$
3 steps on the digital scale

$d_0 = 0 = 0b00$
$d_{V_{\text{max}}} = 3 = 0b11$

$V_{\text{max}} = 12V$
$r = 4V$
$a = 7V$

$M = 2^n$
4 steps on the digital scale

$d_0 = 0 = 0b00$
$d_{V_{\text{max}}} - r = 3 = 0b11$ (no $d_{V_{\text{max}}}$, it would be at $0b100=4$)

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

$V_{\text{min}} = 0V$

$V_{\text{max}} = 12V$
$3=11=12V$
$r = 3V$
$3=11=9V$

$2=10=8V$
$2=10=6V$

$1=01=4V$
$1=01=3V$

$0=00=0V$
$0=0=0V$
Resolution: \( M = 2^n - 1 \) vs. \( 2^n \)

Let \( n = 2 \)

**\( M = 2^n - 1 \)**

3 steps on the digital scale

\[
\begin{align*}
  d_0 &= 0 = 0b00 \\
  d_{V_{\text{max}}} &= 3 = 0b11
\end{align*}
\]

**\( M = 2^n \)**

4 steps on the digital scale

\[
\begin{align*}
  d_0 &= 0 = 0b00 \\
  d_{V_{\text{max}}} - r &= 3 = 0b11 \text{ (no } d_{V_{\text{max}}}, \text{ it would be at } 0b100=4) \\
  r &= 3V
\end{align*}
\]

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

**\( V_{\text{max}} = 12V \)**

\[
\begin{align*}
  3 = 0b11 &= 12V \\
  r &= 4V \\
  a &= 7V
\end{align*}
\]

\[
\begin{align*}
  2 = 0b10 &= 8V \\
  1 = 0b01 &= 4V \\
  0 = 0b00 &= 0V
\end{align*}
\]

**\( V_{\text{min}} = 0V \)**

\[
\begin{align*}
  3 = 0b11 &= 9V \\
  2 = 0b10 &= 6V \\
  1 = 0b01 &= 3V \\
  0 = 0b00 &= 0V
\end{align*}
\]
Resolution: $M = 2^n - 1$ vs. $2^n$ (Related to slope)

Let $n = 2$

$M = 2^n - 1$
3 steps on the digital scale

$d_0 = 0 = 0b00$
$d_{V_{max}} = 3 = 0b11$

$M = 2^n$
4 steps on the digital scale

$d_0 = 0 = 0b00$
$d_{V_{max} - r} = 3 = 0b11$ (no $d_{V_{max}}$, 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 \(V_{\text{max}}\) and \(V_{\text{min}}\) \(a\)

• ADC (Analog to Digital Converter)
  – analog input between \(V_{\text{max}}\) and \(V_{\text{min}}\) \(a\)
  – 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 $V_{min}$ to $V_{max}$?
DAC (Digital-to-Analog Converter):

• Conceptually, given a n-bit digital input ($d$), how does the DAC generate an analog output ($a$) between $V_{\text{min}}$ to $V_{\text{max}}$?
• What other information does the DAC need?
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 \( V_{\text{min}} \) to \( V_{\text{max}} \)?

![Diagram of DAC concept](image)
DAC (Digital-to-Analog Converter):

- Conceptually, given a $n$-bit digital input ($d$), how does the DAC generate an analog output ($a$) between $V_{\text{min}}$ to $V_{\text{max}}$?
DAC (Digital-to-Analog Converter):

- Conceptually, given a n-bit digital input ($d$), how does the DAC generate an analog output ($a$) between $V_{min}$ to $V_{max}$?
DAC (Digital-to-Analog Converter):

- Conceptually, given a n-bit digital input ($d$), how does the DAC generate an analog output ($a$) between $V_{min}$ to $V_{max}$?
ADC:

Given an analog input (a), how does the ADC know what binary value to assign to digital output (d)?
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. ($2^{10} \sim 1,000$)
  - For a 20-bit ADC? ($2^{20} \sim 1$ Million)
  - For a 30-bit ADC? ($2^{30} \sim 1$ 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. ($2^{10} \sim 1,000$): 1ms vs. 10us
  – For a 20-bit ADC? ($2^{20} \sim 1$ Million)
  – For a 30-bit ADC? ($2^{30} \sim 1$ 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. ($2^{10} \sim 1,000$): 1ms vs. 10us
  – For a 20-bit ADC? ($2^{20} \sim 1$ Million): 1s vs. 20us
  – For a 30-bit ADC? ($2^{30} \sim 1$ 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. ($2^{10} \sim 1,000$): 1ms vs. 10us
  - For a 20-bit ADC? ($2^{20} \sim 1$ Million): 1s vs. 20us
  - For a 30-bit ADC? ($2^{30} \sim 1$ Billion): 1000s = 16min vs. 30us
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

Analog Input

Vmax
Vmin

DAC

Comparator

Guess

1 OR 0

SAR

SAR BUF

State machine

Timing control

Digital output

SAR: Successive approximation register

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

- DAC
  - Vmax = 16V
  - Vmin = 0V
- Comparator
- State machine
- Timing control
- SAR BUF
- Digital output
- SAR: Successive approximation register
- n = 4
- Guess
- Let M = 2^n
- Analog Input a = 9.5V

Let M = 2^n

Digital output

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

- DAC
- Analog Input
  - $a = 9.5V$
- SAR: Successive approximation register
- SAR BUF
- State machine
- Timing control
- Digital output

Let $M = 2^n$

- $V_{\text{max}} = 16V$
- $V_{\text{min}} = 0V$

Guess

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a $\geq$ Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

Let \( M = 2^n \)

V_{\text{max}} = 16V

V_{\text{min}} = 0V

DAC

Comparator

\( =< \)

\( =< \)

\( a = 9.5V \)

\( n = 4 \)

Digital output

State machine

Timing control

SAR: Successive approximation register

Digital output

\( \text{Guess} \)

1 OR 0

\begin{array}{|c|c|c|c|}
\hline
\text{Step} & \text{Range} & \text{Mid (digital)} & \text{Mid (voltage)} & \text{Is a } \geq \text{ Guess (voltage)}? \\
\hline
0 & 0bxxxx & 0b1000 & & \\
1 & & & & \\
2 & & & & \\
3 & & & & \\
4 & & & & \\
\hline
\end{array}

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

DAC

Vmax=16V
Vmin=0V

Let M = 2^n

n=4

Analog Input

a=9.5V

Comparator

Digital output

SAR: Successive approximation register

State machine

Timing control

SAR BUF

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

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

V_{\text{max}} = 16V
V_{\text{min}} = 0V

DAC

Let \( M = 2^n \)

\( n = 4 \)

\( V_{\text{max}} = 16V \)
\( V_{\text{min}} = 0V \)

Analog Input \( a = 9.5V \)

Comparator

Let \( M = 2^n \)

\( n = 4 \)

\( \text{Mid (voltage)} = 8 \text{ Volts} \)

\( \text{Is a} \geq \text{Guess (voltage)?} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a ( \geq ) Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

V_{\text{max}} = 16V, V_{\text{min}} = 0V

DAC

Let M = 2^n

n = 4

SAR: Successive approximation register

$\text{SAR\;BUF}$

Digital output

Comparator

Let \( a = 9.5V \)

Guess = 8V

Analog Input

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

Analog Input

Let M = 2^n

V_{\text{max}} = 16V

V_{\text{min}} = 0V

DAC

Comparator

Guess = 8V

State machine

Timing control

SAR BUF

Digital output

SAR: Successive approximation register

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a &gt;= Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

- Analog Input: \( a = 9.5V \)
- \( V_{\text{max}} = 16V \)
- \( V_{\text{min}} = 0V \)
- \( \text{Guess} = 12V \)
- \( n = 4 \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a &gt;= Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SAR: Successive approximation register

Digital output

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

- DAC
- SAR: Successive approximation register

Timing control

Analog Input

a = 9.5V

Let M = 2^n

- n = 4

V_{max} = 16V
V_{min} = 0V

Guess = 12V

Digital output

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a &gt;= Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

DAC

Let $M = 2^n$

Analog Input $a=9.5V$

Comparator

State machine

Timing control

SAR BUF

Digital output

SAR: Successive approximation register

Vmax=16V
Vmin=0V

Guess = 12V

0 (No)

Let $M = 2^n$

$n=4$

Vmax=16V
Vmin=0V

http://class.ece.iastate.edu/cpre288

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a &gt;= Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

Let $M = 2^n$

$V_{\text{max}} = 16V$
$V_{\text{min}} = 0V$

Guess = 12V

DAC

Comparator

SAR: Successive approximation register

State machine

Timing control

Digital output

Analog Input

Let $M = 2^n$

$V_{\text{max}} = 16V$
$V_{\text{min}} = 0V$

$n = 4$

$V_{\text{max}} = 16V$

Comparing

0

$0 (\text{No})$

Step | Range | Mid (digital) | Mid (voltage) | Is a $\geq$ Guess (voltage)?
--- | --- | --- | --- | ---
0 | 0bxxxx | 0b1000 | 8 Volts | Yes
1 | 0b1xxx | 0b1100 | 12 Volts | No
2 | 0b10xx | 0b1010 | | |
3
4

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

- **DAC**: Digital to Analog Converter
  - \( V_{\text{max}} = 16\text{V} \)
  - \( V_{\text{min}} = 0\text{V} \)

- **Comparator**: Compares DAC output with Analog Input
- **SAR**: Successive Approximation Register
- **State Machine**: Controls the process
- **Timing Control**: Manages the timing of the process

**Analog Input**
- \( a = 9.5\text{V} \)

**Guess** = 10V

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a &gt;= Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td>0b1010</td>
<td>10 Volts</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Digital Output**: 0b1010

State machine: SAR BUF

SAR: Successive approximation register

Timing control

http://class.ece.iastate.edu/cpre288
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

Let $M = 2^n$ where $n=4$

Vmax=16V, Vmin=0V

Guess = 10V

Analog Input $a = 9.5V$

State machine

Timing control

Digital output

SAR: Successive approximation register

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a $\geq$ Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td>0b1010</td>
<td>10 Volts</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

Analog Input

Vmax = 16V
Vmin = 0V

DAC

Let M = 2ⁿ

n = 4

Guess = 10V

SAR: Successive approximation register

State machine

Digital output

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a &gt;= Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td>0b1010</td>
<td>10 Volts</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>0b100x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

- Analog Input: $a=9.5V$
- DAC: $V_{\text{max}}=16V$, $V_{\text{min}}=0V$
- SAR: Successive approximation register
- Timing control

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a $\geq$ Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td>0b1010</td>
<td>10 Volts</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>0b100x</td>
<td>0b1001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

DAC: Digital to Analog Converter

Vmax=16V
Vmin=0V

Guess = 9V

Let M = 2^n
n=4

SAR: Successive Approximation Register

Timing control

Comparator

Analog Input
a=9.5V

Digital output

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is a &gt;= Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td>0b1010</td>
<td>10 Volts</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>0b100x</td>
<td>0b1001</td>
<td>9 Volts</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

Let $M = 2^n$

$V_{\text{max}} = 16\text{V}$

$V_{\text{min}} = 0\text{V}$

Analog Input

$V_{\text{max}} = 16\text{V}$

$V_{\text{min}} = 0\text{V}$

$\text{DAC}$

$\text{Comparator}$

Guess = 9V

$\text{SAR} = \text{Successive approximation register}$

Timing control

Digital output

State machine

$\text{SAR BUF}$

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>$\text{Mid (digital)}$</th>
<th>$\text{Mid (voltage)}$</th>
<th>Is $a \geq \text{Guess (voltage)}$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td>0b1010</td>
<td>10 Volts</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>0b100x</td>
<td>0b1001</td>
<td>9 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>0b1001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://class.ece.iastate.edu/cpre288
It’s built upon a DAC

Vmax = 16V
Vmin = 0V

Let $M = 2^n$

Guess = 9V

$\frac{V_{\text{max}} - V_{\text{min}}}{2^n} = 8$ Volts

<table>
<thead>
<tr>
<th>Step</th>
<th>Range</th>
<th>Mid (digital)</th>
<th>Mid (voltage)</th>
<th>Is $a \geq$ Guess (voltage)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0bxxxx</td>
<td>0b1000</td>
<td>8 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0b1xxx</td>
<td>0b1100</td>
<td>12 Volts</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0b10xx</td>
<td>0b1010</td>
<td>10 Volts</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>0b100x</td>
<td>0b1001</td>
<td>9 Volts</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>0b1001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Analog Input

$a = 9.5V$

State machine

Timing control

Digital output 1001

SAR: Successive approximation register

http://class.ece.iastate.edu/cpre288
• Example: 0V-16V range, 9.5V input, 4-bit resolution.

\[
\frac{a - V_{\text{min}}}{V_{\text{max}} - V_{\text{min}}} = \frac{d}{2^n}
\]

\[
\frac{9.5}{16 - 0} = \frac{d}{16}
\]

\[
d = \frac{9.5 \times 16}{16} = 9.5
\]

\[d = 0b1001 = 9\] (Why not just use the above equation?)
ADC Using Successive Approximation

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

\[
\frac{a - V_{\text{min}}}{V_{\text{max}} - V_{\text{min}}} = \frac{d}{2^n}
\]

\[
\frac{9.5}{16 - 0} = \frac{d}{16}
\]

\[
d = \frac{9.5 \times 16}{16} = 9.5
\]

\[
d = 0b1001 = 9
\]

(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

\[ M = 2^n \]

\[ 2^2 = 4 \text{ buckets (or ranges) for the analog signal to fall} \]

_Resolution_: analog change resulting from a digital change of 1
Practice Problem 1 (Linear equation)

• Question:
  – Given:
    • n = 2 bit resolution
    • $V_{\text{min}} = 0$ volts
    • $V_{\text{max}} = 12$ volts
    • $a = 5$ volts
    • $M = 2^n$ bins
  – What is $d$?
Practice Problem 1 (Linear equation)

• Question:
  – Given:
    • \( n = 2 \) bit resolution
    • \( V_{\text{min}} = 0 \) volts
    • \( V_{\text{max}} = 12 \) volts
    • \( a = 5 \) volts
    • \( M = 2^n \) bins
  – What is \( d \)?

• Answer:
  – Analog and Digital signals are proportional
    \[
    \frac{a - V_{\text{min}}}{V_{\text{max}} - V_{\text{min}}} = \frac{d}{2^n}
    \]
  – \( d \) is 0b01
Practice Problem 2 (Successive Approximation)

• Question:
  – Given:
    • $n = 4$ bit resolution
    • $V_{\text{min}} = 0$ volts
    • $V_{\text{max}} = 12$ volts
    • $a = 5$ volts
    • $M = 2^n$ bins
  – What is $d$?
• Question:
  – Given:
    • $n = 4$ bit resolution
    • $V_{min} = 0$ volts
    • $V_{max} = 12$ volts
    • $a = 5$ volts
    • $M = 2^n$ bins
  – What is $d$?

• Answer:
  – Use successive approximation
  – $d = \text{?} \text{?} \text{?} \text{?}$
    • 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

http://class.ece.iastate.edu/cpre288
Practice Problem 2 (Successive Approximation)

• Question:
  – Given:
    • n = 4 bit resolution
    • V_{\text{min}} = 0 volts
    • V_{\text{max}} = 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
    - \( V_{\text{min}} = 0 \) volts
    - \( V_{\text{max}} = 12 \) volts
    - \( a = 5 \) volts
    - \( M = 2^n \) bins
  - 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
• Question:
  – Given:
    • $n = 4$ bit resolution
    • $V_{\text{min}} = 0$ volts
    • $V_{\text{max}} = 12$ volts
    • $a = 5$ volts
    • $M = 2^n \text{bins}$
  – 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
Question:

- Given:
  - $n = 4$ bit resolution
  - $V_{\text{min}} = 0$ volts
  - $V_{\text{max}} = 12$ volts
  - $a = 5$ volts
  - $M = 2^n$ bins
- 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 ATMEGA128
ATMega128 ADC

- 10-bit ADC conversion
- Eight input channels through a MUX
- Analog input on one of ADC0-ADC7 pins
- Up to 15K samples per second
- 0 – Vcc or 0 – 2.56V ADC input voltage range

Which pins on the ATMega128 are used for the ADC?
- Alternative I/O Functions (See next slides)
ATMega128 I/O Ports and Alternative Functions

Ports A-G, each pin can be configured as General Purpose Digital I/O Pin

- DDRx decides if a pin is for input or output *
- PORTx is for writing to output pins
- PINx is for reading from input pins
(* There is a special tri-state configuration)

Alternatively, those pins can be used as I/O pins for internal I/O devices

- Activate a pin’s alternative function by enabling the corresponding I/O device
- Then, the pin MAY NOT be used as GP I/O pin
Most pins have Alternative Functions: USART, ADC, input capture, output compare, and others

**USART0 uses port E**
- PE2: External Clock (PE – Port E)
- PE1: Transmit Pin
- PE0: Receive Pin

**USART1 uses port D**
- PD5: External Clock
- PD3: Transmit Pin
- PD2: Receive Pin
ATMega128 I/O Pins (Alternative Functions)

From ATMega128 Data Sheet
ADC I/O Pins

The ADC uses Port F, all eight pins
  – There are eight input channels
  – PF0 – PF7 for channels 0-7

If ADC is enabled, then avoid using port F for external I/O device
  – No need to configure Port F as input, just enable the ADC
Programming Interface Registers

Three registers

**ADCSRA**: ADC control and status register A

**ADMUX**: ADC input selection

**ADCW**: ADC result register, 16-bit
Programming Interface Registers

Datasheet Page: 231

ADCSRA

ADCW

ADMUX
Program Interface: ADCSRA

**ADEN**: ADC Enable
Write 1 to this bit to enable ADC

**ADSC**: ADC Start Conversion
Write 1 to this bit to start ADC conversion. *It turns to 0 after conversion is done*
Enable ADC

```
ADCSRA |= 0x80;
```

Disable ADC

```
ADCSRA &= 0x7F;
```

Read an ADC word, assuming it’s ready

```
unsigned reading = ADCW;
```
Coding Examples

Using a different coding style

Enable ADC

\[
\text{ADCSRA} \ | = (1 << \text{ADEN});
\]

Disable ADC

\[
\text{ADCSRA} \ & = \sim (1 << \text{ADEN});
\]

Notes

– `<avr/io.h>` declares `ADEN` and other names as macros
– `ADEN` is defined as 7: `(1 << 7) == 0x80`
Assume ADC has been configured appropriately and in one-shot mode with interrupts disabled.

Write code to (1) start ADC, (2) wait for the conversion to complete, and (3) read the output.

```c
ADCSRA |= (1<<ADSC);
while (ADCSRA & (1<<ADSC)) {}
unsigned ADC_reading = ADCW;
```
Program Interface: ADCSRA

**ADFR**: ADC Free Running Select
- 1: continues sampling, 0: one shot

**ADIF**: ADC Interrupt Flag
- 1: interrupt raised, 0: otherwise

**ADIE**: ADC Interrupt Enable
Enable ADC interrupt

\[
\text{ADCSRA} \ |= (1 \ll \text{ADIE});
\]

Disable ADC interrupt

\[
\text{ADCSRA} \ &= \ \sim (1 \ll \text{ADIE});
\]

Check ADC interrupt flag manually

\[
\text{if (ADCSRA} \ &\ (1 \ll \text{ADIF}))
\]

\[
... \ // \ do \ something
\]
Use a different coding style

Enable ADC interrupt

```c
ADCSRA |= _BV(ADIE);
```

Disable ADC interrupt

```c
ADCSRA &= ~_BV(ADIE);
```

_BV (Bit Vector) is declared in `<avr/io.h>` like follows:

```c
#define _BV(x)  (1<<x)
```
Program Interface: ADCSRA

**ADPS2:0**: ADC Prescaler Select Bits

Select one of seven division factors

- 000: 2, 001: 2, 010: 4, 011: 8,
- 100: 16, 101: 32, 110: 64, 111: 128

Typical conversion times:

- 25 ADC clock cycles for the first conversion
- Faster time (13 or 14 ADC cycles) for second conversion and thereafter in continuous sampling mode

**Required ADC clock frequency**: 50KHz – 200KHz

- Higher frequency possible with less than 10-bit accuracy
System clock is 16MHZ. What prescalar value is valid?

<table>
<thead>
<tr>
<th>Pres. Bits</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Div. Factor</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>Freq. (Hz)</td>
<td>8M</td>
<td>8M</td>
<td>4M</td>
<td>2M</td>
<td>1M</td>
<td>500K</td>
<td>250K</td>
<td>125K</td>
</tr>
</tbody>
</table>

What happens with higher frequency?

To set the frequency:

\[
\text{ADCSRA} \ |= (7 << \text{ADSP0});
\]

Or

\[
\text{ADCSRA} \ |= \_\BV(\text{ADSP2}) | \_\BV(\text{ADSP1}) | \_\BV(\text{ADSP0});
\]
**REFS1:0**: Reference Selection Bits

- 00 to select an external reference voltage (AREF)
- 01 to select Vcc as the reference voltage
- 10: Reserved configuration
- 11 select 2.56 as reference voltage

**ADLAR**: ADC Left Adjust Result

- 1: 10-bit data is left adjusted in 16-bit reg
- 0: right adjusted
Set the reference voltage to 2.56V:

```c
ADMUX |= _BV(REFS1) | _BV(REFS0);
```

Make the result left adjusted:

```c
ADMUX |= _BV(ADLAR);
```
MUX4:0: Analog Channel and Gain Selection Bits

There are eight pins connected to ADC through a MUX

Code 00000 for ADC pin 0, 00001 for pin 1, …, and 00111 for pin 7.

01xxx, 10xxx, 11xxx are reserved for differential inputs (not to be discussed) with gain
Coding Example

Get a reading from a given ADC channel

```c
unsigned ADC_read(char channel)
{
    ADMUX |= (channel & 0x1F);
    ADCSRA |= _BV(ADSC);
    while (ADCSRA & _BV(ADCS)) {}
    return ADCW;
}
```
Configure ADC as single shot, interrupt disabled, result right adjusted, precalar 128 and use 2.56 as the reference voltage

First, decide bits in ADCSR and ADMUX
Programming Example

ADC_init()
{
  // REFS=11, ADLAR=0, MUX don’t care
  ADMUX = _BV(REFS1) | _BV(REFS0);

  // ADEN=1, ADFR=0, ADIE=0, ADSP=111
  // others don’t care
  // See page 246 of user guide
  ADCSRA = _BV(ADEN) | (7<<ADPS0);
}

Note: It’s a good idea to enable the device as the last step, so make change to ADCSRA as the last step
## Typical Conversion Times

<table>
<thead>
<tr>
<th>Condition</th>
<th>Sample &amp; Hold</th>
<th>Conversion Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Conversion</td>
<td>13.5</td>
<td>25</td>
</tr>
<tr>
<td>Normal conversion, single ended</td>
<td>1.5</td>
<td>13</td>
</tr>
<tr>
<td>Normal conversion, differential</td>
<td>1.5/2.5</td>
<td>13/14</td>
</tr>
</tbody>
</table>
Measuring Distance with the IR Sensor

- The IR sensor emits an IR beam, and sets a voltage based on the distance of an object.
From the IR Sensor Datasheet

- The voltage from the IR sensor depends on the distance.
- As the distance increases, the voltage decreases (see graph).
How To Measure Distance with the IR Sensor

Getting a distance from the IR sensor involves the following process:

1. The IR sensor measures a distance and sets the voltage on the wire leading to **ADC2 (channel 2)**
2. The ADC converts this voltage into a digital value between 0 and 1023 and stores it in the registers ADCL and ADCH (ADCW)
3. Your program reads ADCW and converts the value into a distance... but how?!?
Two methods to calibrate your distance

Measure 50 points, create a table for comparing
- Create a table that has the value of ADCW when an object is x centimeters away
- Use this table to lookup the distance when a similar ADCW result is returned

Measure 5 points, use Excel to get a trend line
ADC Summary

• ADC general knowledge
  – Applications, sampling and quantization
  – ADC conversion formulas
  – ADC design: Successive approximation
  – Terminology, Performance and other issues

• Atmega128 ADC programming interface
  – **ADCSRA**: Control and status register
    • Enable, Start Conversion, Running Mode, Interrupt Enable, Interrupt Flag, Clock Select (3)
  – **ADMUX**: Reference voltage, left/right adjustment, input mux
  – **ADCW**: ADC Word

• API functions
  – ADC_init()
  – ADC_read()