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

Dr. Phillip Jones
Announcements

• Exam 1: In class Wed 6/8
• 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 = __________;
}
```
// 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
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) = \(\text{Span} / M\)

- **Sensitivity**: Amount sensor output changes for a change in sensor input
**Analog-to-digital converter: Example**

- **Temperature Sensor**: Analog to digital converter (A/D) input
  - Sensor output
  - A/D input

### Temperature vs. Voltage (Sensor Specification)
- **T_{\text{max}} = 200 \, ^\circ\text{C}**
- **T_{\text{min}} = 0 \, ^\circ\text{C}**
- **Sensor_{\text{Vmin}} = 0 \, \text{V}**
- **V_{\text{max}} = 3.3 \, \text{V}**

### A/D: Analog Input vs. Digital Output
- (M = 2^n - 1 steps (or bins): D_{\text{max}} = V_{\text{max}})
- **A/D_{\text{Vmax}} = 3.3 \, \text{V}**
- **A/D_{\text{Vmin}} = 0 \, \text{V}**
- **D = 0**
- **Digital Output (D)**
  - 10-bit
  - D_{\text{max}} = 1023
Sensitivity: Analog Sensor (Linear)

- **Sensitivity:** How much does a change in a sensor input change the sensor output
Sensitivity: Analog Sensor (Linear)

- **Tmin = 0 C**
- **Tmax = 200 C**
- **Vmin = 0**
- **Vmax = 3.3V**

**Temperature vs. Voltage**

**Sensor Input**
(Temperature (C))

**Analog Sensor Output (V)**

- **ΔT**
- **ΔV**

- **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 slope (m) of the Input vs. Output specification
  - Remember slope = \( \frac{\Delta V}{\Delta T} \)
Sensitivity: Analog Sensor (Linear)

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

\[
m = \text{slope} = \frac{\text{RISE}}{\text{RUN}} = \frac{\text{Tmax} - \text{Tmin}}{\text{Vmax} - \text{Vmin}}
\]

\[
= \frac{200 - 0}{3.3 - 0} = 60.61 \text{ C/V}
\]
• 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

\[ 100 = 60.61X; \] \[ X = 100/60.61 \approx 1.65 \text{ V} \]
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

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

- V_{min} = 0 V
- V_{max} = 3.3 V
- D = 0
- D_{max} = 1023
- \Delta D
- \Delta V
- A/D output
- A/D input
- Digital output
- Sensor output
- 10-bit
- 10-bit

Diagram showing A/D conversion process and resolution concept.

10-bit

Digital Output (D)

D_{max} = 1023

\Delta D

\Delta V

V_{max} = 3.3 V

V_{min} = 0 V

D = 0

A/D: Analog Input vs. Digital Output
(For M = 2^{n-1} steps (or bins), D_{max} = V_{max})
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 the Least Significant Bit (LSB) weight

\[ V_{\text{min}} = 0 \text{ V} \]
\[ V_{\text{max}} = 3.3 \text{ V} \]
\[ D_{\text{max}} = 1023 \]
\[ \Delta D \]
\[ \Delta V \]

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

For this example:
Resolution = slope = RISE/RUN = (3.3 – 0 / 1023 – 0) = .0032 V/bit
LSB bit weight = .0032 V/bit
A/D: Compute Digital output

- Vmin = 0 V
- Digital Output (D)
  - D = 0

Digital: A/D: Compute Digital output

A/D: Analog Input vs. Digital Output
(For M = 2^n-1 steps (or bins), Dmax = Vmax)

Sensor output

A/D input

Sensor output

A/D input

A/D: 10-bit

- Vmax = 3.3 V
- Vmin = 0 V
- D = 0

Digital output

10-bit

1023

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 \( m = 0.0032 \), so
  \[ 1.65 = 0.0032X; \]
  \[ X = 1.65/0.0032 = 515.625 \] -> Truncate to 515 = 0b10_0000_0011
- For this example, a Temperature of 100 C gives a digital value of 515
Question: What is the Temperature LSB weight?
Question: What is the Temperature LSB weight?

- We know the sensor sensitivity = 60.61°C/V
- We know the A/D resolution = 0.0032 V/bit
- We want C/bit: So $60.61\,\text{C/V} \times 0.0032\,\text{V/bit} = 0.194\,\text{C/bit}$

A change of 1 in the digital output, corresponds to a change 0.194 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>

http://class.ece.iastate.edu/cpre288
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), \( D_{\text{max}} = V_{\text{max}} \))

- \( V_{\text{max}} = 1.65V \)
- \( V_{\text{min}} = -1.65V \)
- Digital Output (\( D \))
- \( D_{\text{max}} = 1023 \)
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

\[
1.638 + .0064 + 0.0032 \approx 1.65 \text{ V} \\
+ (-1.65 \text{ V}) \\
0 \text{ V}
\]

<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

100 C

Sensor output 1.65V A/D input

Digital output

T_max = 200 C
T_min = 0 C
Sensor_Vmin = 0

Sensor Input (T)

Temperature vs. Voltage (Sensor Specification)

Slope = Sensitive 60.61 C/V

Vmax = 3.3V

Analog Sensor Output (V)

A/D: Analog Input vs. Digital Output

(M = 2^n-1 steps (or bins): Dmax = Vmax)

A/D_Vmax = 3.3V

A/D Input (V)

A/D_Vmin = 0 V

D = 0

Digital Output (D)

10-bit

Slope = Resolution .0032 V/bit

Dmax = 1023

Slope = Resolution .0032 V/bit

Slope = Resolution .0032 V/bit

Slope = Resolution .0032 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>7.5V</td>
<td>1111</td>
</tr>
<tr>
<td>7.0V</td>
<td>1110</td>
</tr>
<tr>
<td>6.5V</td>
<td>1101</td>
</tr>
<tr>
<td>6.0V</td>
<td>1100</td>
</tr>
<tr>
<td>5.5V</td>
<td>1011</td>
</tr>
<tr>
<td>5.0V</td>
<td>1010</td>
</tr>
<tr>
<td>4.5V</td>
<td>1001</td>
</tr>
<tr>
<td>4.0V</td>
<td>1000</td>
</tr>
<tr>
<td>3.5V</td>
<td>0111</td>
</tr>
<tr>
<td>3.0V</td>
<td>0110</td>
</tr>
<tr>
<td>2.5V</td>
<td>0101</td>
</tr>
<tr>
<td>2.0V</td>
<td>0100</td>
</tr>
<tr>
<td>1.5V</td>
<td>0011</td>
</tr>
<tr>
<td>1.0V</td>
<td>0010</td>
</tr>
<tr>
<td>0.5V</td>
<td>0001</td>
</tr>
<tr>
<td>0V</td>
<td>0000</td>
</tr>
</tbody>
</table>

proportionality

Digital sampling of an analog signal

Digital generation of an analog signal

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

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

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

Mapping between Analog and Digital

\[ V_{max} = 7.5V \]

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

proportionality

V_{\text{max}}=7.5V

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_{max} = 7.5V

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Binary Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0V</td>
<td>1110</td>
</tr>
<tr>
<td>6.5V</td>
<td>1111</td>
</tr>
<tr>
<td>6.0V</td>
<td>1100</td>
</tr>
<tr>
<td>5.5V</td>
<td>1101</td>
</tr>
<tr>
<td>5.0V</td>
<td>1010</td>
</tr>
<tr>
<td>4.5V</td>
<td>1011</td>
</tr>
<tr>
<td>4.0V</td>
<td>1000</td>
</tr>
<tr>
<td>3.5V</td>
<td>0111</td>
</tr>
<tr>
<td>3.0V</td>
<td>0110</td>
</tr>
<tr>
<td>2.5V</td>
<td>0101</td>
</tr>
<tr>
<td>2.0V</td>
<td>0100</td>
</tr>
<tr>
<td>1.5V</td>
<td>0011</td>
</tr>
<tr>
<td>1.0V</td>
<td>0010</td>
</tr>
<tr>
<td>0.5V</td>
<td>0001</td>
</tr>
<tr>
<td>0V</td>
<td>0000</td>
</tr>
</tbody>
</table>

Proportionality

Digital sampling of an analog signal

Digital generation of an analog signal

V_{max} = 7.5V

Digital output

analog to digital

Digital input

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

<table>
<thead>
<tr>
<th>Analog Input (V)</th>
<th>Digital Output (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00 V</td>
<td>0000</td>
</tr>
<tr>
<td>0.50 V</td>
<td>0001</td>
</tr>
<tr>
<td>1.00 V</td>
<td>0010</td>
</tr>
<tr>
<td>1.50 V</td>
<td>0011</td>
</tr>
<tr>
<td>2.00 V</td>
<td>0100</td>
</tr>
<tr>
<td>2.50 V</td>
<td>0101</td>
</tr>
<tr>
<td>3.00 V</td>
<td>0110</td>
</tr>
<tr>
<td>3.50 V</td>
<td>0111</td>
</tr>
<tr>
<td>4.00 V</td>
<td>1000</td>
</tr>
<tr>
<td>4.50 V</td>
<td>1001</td>
</tr>
<tr>
<td>5.00 V</td>
<td>1010</td>
</tr>
<tr>
<td>5.50 V</td>
<td>1011</td>
</tr>
<tr>
<td>6.00 V</td>
<td>1100</td>
</tr>
<tr>
<td>6.50 V</td>
<td>1101</td>
</tr>
<tr>
<td>7.00 V</td>
<td>1110</td>
</tr>
<tr>
<td>7.50 V</td>
<td>1111</td>
</tr>
</tbody>
</table>

Proportionality

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

Digital sampling of an analog signal

Digital generation of an analog signal

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

Simple Equation

Assume $V_{min} = 0$ V.

$V_{max} =$ 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

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

This is derived from the equation for a line

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

$$Y = m * X + b$$
Proportional Signals (General case)

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

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

\[
Y = m \times X + b
\]
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

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

\( r, \text{ resolution} \): 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

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

$M = 2^n$
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

Vmax = 12V

M = $2^n - 1$

3 = 11 = 12V

M = $2^n$

3 = 11 = 9V

2 = 10 = 8V

1 = 01 = 4V

0 = 00 = 0V

Vmin = 0V

r = 4V

r = 3V

3 = 11 = 9V

2 = 10 = 6V

1 = 01 = 3V

0 = 00 = 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)

$r = 3V$

$r, \text{ resolution}$: 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
$V_{\text{max}} = 3 = 0b11$

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

d_0 = 0 = 0b00
$V_{\text{max}} - r = 3 = 0b11$ (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. \( 2^n \) (Related to slope)

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

\[ V_{\text{max}} = 12V \]
\[ V_{\text{min}} = 0V \]
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 (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 $V_{max}$?
- 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 \( V_{\text{min}} \) 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 \(V_{\text{min}}\) to \(V_{\text{max}}\)?
Conceptually, given a n-bit digital input (d), how does the DAC generate an analog output (a) between Vmin to Vmax?
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?
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? \((2^{20} \sim 1\text{ Million})\)
  - For a 30-bit ADC? \((2^{30} \sim 1\text{ Billion})\)
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? \((2^{20} \sim 1\text{ Million})\): 1s vs. 20us
  - For a 30-bit ADC? \((2^{30} \sim 1\text{ Billion})\)
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? ($2^{20} \sim 1\ \text{Million}$): 1s vs. 20us
  - For a 30-bit ADC? ($2^{30} \sim 1\ \text{Billion}$): 1000s = 16min vs. 30us
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

SAR: Successive approximation register

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

- **DAC**: Digital-to-Analog Converter
- **Comparator**: Compares analog input to DAC output
- **SAR** (Successive Approximation Register): Stores the digital output
- **State machine**: Controls the timing
- **Timing control**: Regulates the operation

**Analog Input**:\[ a = 9.5 \text{V} \]

**Vmax**: 16V

**Vmin**: 0V

Let \( M = 2^n \)

\( n = 4 \)

1 OR 0

Digital output

SAR: Successive approximation register

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

It’s built upon a DAC

Analog Input
\[ a = 9.5V \]

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

Let \( M = 2^n \)
\( n = 4 \)

\[ a = 9.5V \]

```
<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></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>
```

Digital output

SAR: Successive approximation register

Comparator

State machine

Timing control

DAC

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

It’s built upon a DAC

Analog Input

\[ a = 9.5\text{V} \]

Let \( M = 2^n \)

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

Digital output

<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></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>

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 \( M = 2^n \)

\( n = 4 \)

\( a = 9.5V \)

Guess = 8V

Comparator

\begin{align*}
\text{Is } a & \geq \text{Guess (voltage)}? \\
0 & \quad 0bxxxx & 0b1000 & 8 \text{ Volts} \\
1 & \\
2 & \\
3 & \\
4 & \\
\end{align*}

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

SAR: Successive approximation register

Digital output

SAR BUF

State machine

Timing control

Analog Input

1 OR 0

Constructing the ADC (Successive Approximation)

It’s built upon a DAC

<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></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>

Analog Input

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

Let $M = 2^n$

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

DAC

Comparator

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

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

Let M = 2^n

Digital output

SAR: Successive approximation register

State machine

Timing control

Comparator

Analog Input

a=9.5V

Guess = 8V

n=4

SAR

1 0 0 0

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

DAC

Let \( M = 2^n \)

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

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

\( \text{Guess} = 8\text{V} \)

Analog Input

\( a = 9.5\text{V} \)

\( n = 4 \)

State machine

Timing control

SAR: Successive approximation register

SAR BUF

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></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

Analog Input

Let $M = 2^n$

$a = 9.5V$

$n = 4$

$V_{max} = 16V$

$V_{min} = 0V$

Guess = 12V

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></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>

State machine

Timing control

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

It’s built upon a DAC

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

\( n = 4 \)
\( a = 9.5\text{V} \)

**State machine**

**Timing control**

**DAC**

**Comparator**

**SAR**

**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>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

**Analog Input**

- Let $M = 2^n$
- $V_{max} = 16V$
- $V_{min} = 0V$
- Guess = 12V
- $a = 9.5V$

<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></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>

Digital output

**SAR: Successive approximation register**

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

It’s built upon a DAC

Analog Input

a = 9.5V

Vmax = 16V
Vmin = 0V

Let M = 2^n

n = 4

Digital output

State machine

Timing control

DAC

Comparator

SAR

SAR: Successive approximation register

Guess = 12V

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

- DAC
- Analog Input
  - a = 9.5V
- SAR: Successive approximation register
- n = 4
- Vmax = 16V
- Vmin = 0V

Comparator

Guess = 10V

Let M = 2^n

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 | }

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

It’s built upon a DAC

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

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

**Vmax=16V**

DAC

Comparator

**Guess = 10V**

\[ \text{Analog Input a}=9.5V \]

**Step**

<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></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 \( M = 2^n \)

\( n=4 \)

Guess = 10V

Comparator

Comparing the DAC output:

<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></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

State machine

Timing control

Digital output

Analog Input \( a=9.5V \)

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

It’s built upon a DAC

DAC

Let M = $2^n$

V_max = 16V
V_min = 0V

Analog Input

Let $a = 9.5V$

n = 4

Guess = 10V

Comparator

SAR

SAR: Successive approximation register

Timing control

Digital output

State machine

Table:

<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></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
It’s built upon a DAC

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

Let M = 2^n

n = 4

Guess = 9V

\text{Analogue Input} a = 9.5V

\text{SAR: Successive approximation register}

\text{Timing control}

\text{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

Analog Input
a=9.5V

DAC

Let M = 2^n

Vmax=16V
Vmin=0V

Guess = 9V

Comparator

SAR

Let M = 2^n

n=4

SAR: Successive approximation register

State machine

Timing control

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>Yes</td>
</tr>
<tr>
<td>4</td>
<td>0b1001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constructing the ADC (Successive Approximation)

It’s built upon a DAC

Let $M = 2^n$

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

Guess = 9V

Analog Input

a = 9.5V

Timing control

State machine

DAC

Comparator

SAR

Digital output

0b1001

1001

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>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>
ADC Using Successive Approximation

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

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

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

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

\[
d = 0b1001 = 9 \text{ (Why not just use the above equation?)}
\]
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?)
• Example
  – 2-bits of resolution
  – $V_{\text{min}} = 0$ Volts
  – $V_{\text{max}} = 12$ 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

http://class.ece.iastate.edu/cpre288
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 $d$?
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
  – \( d = 0b??? \)
    • 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

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

http://class.ece.iastate.edu/cpre288
• 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
  – $d = 0b01??$
    • 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

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
  – \( d = 0b011 \)?
    • 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

http://class.ece.iastate.edu/cpre288
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 = 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
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
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
Coding Examples

Enable ADC

\[ \text{ADCSRA} \ |= 0x80; \]

Disable ADC

\[ \text{ADCSRA} \ &= 0x7F; \]

Read an ADC word, assuming it’s ready

\[ \text{unsigned reading} = \text{ADCW}; \]
Coding Examples

Using a different coding style

Enable ADC

```c
ADCSRA |= (1 << ADEN);
```

Disable ADC

```c
ADCSRA &= ~(1 << 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;
```
**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

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

Disable ADC interrupt

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

Check ADC interrupt flag manually

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

\[
... \text{ // do something}
\]
Coding Examples

Use a different coding style

Enable ADC interrupt

\[
\text{ADCSRA} \quad | = \quad _{BV}(\text{ADIE})\nonumber
\]

Disable ADC interrupt

\[
\text{ADCSRA} \quad & = \quad _{BV}(\text{ADIE})\nonumber
\]

_{BV} \quad (\text{Bit Vector}) \quad \text{is declared in} \quad \langle\text{avr/io.h}\rangle \quad \text{like follows:}

\[
\text{#define} \quad _{BV}(x) \quad (1 <<< (x))
\]
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:

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

Or

\[
\text{ADCSRA} \ |= \_\_BV(\text{ADSP2}) \_\_BV(\text{ADSP1}) \_\_BV(\text{ADSP0});
\]
**Programming Interface: ADMUX**

**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:

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

Make the result left adjusted:

```
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
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?!?
How To Measure Distance with the IR Sensor

• 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()