Problem: Switch Bounce

Typically 10-20ms

Devices whose resistance changes (usually decreases) with light striking it
(also called photocells, photoconductors)
Light striking a semiconducting material can provide sufficient energy to cause electrons to break away from atoms.
Free electrons and holes can be created which causes resistance to be reduced

Typical materials used are Cadmium Sulphide (CdS), Cadmium Selenide (CdSe), Lead Sulphide
With no illumination, resistance can be greater than 1 MΩ (dark resistance).
Resistance varies inversely proportional to light intensity.
Reduces down to 10-100s ohms
100ms/10ms response time
Light Dependent Resistors (LDRs)

- LDRs have a low energy gap
- Operate over a wide wavelengths (some, into infrared)
- Indium antimonide is good for IR. When cooled is very sensitive, used for thermal scanning of earth’s surface

Analog to digital conversion

- Use charge-redistribution technique
  - no sample and hold circuitry needed
  - even with perfect circuits quantization error occurs
- Basic capacitors
  - sum parallel capacitance

![Capacitors Diagram]

- Two reference voltage
  - mark bottom and top end of range of analog values that can be converted (V_L and V_H)
  - voltage to convert must be within these bounds (V_X)
- Successive approximation
  - most approaches to A/D conversion are based on this
  - 8 to 16 bits of accuracy

Approach

- sample value
- hold it so it doesn’t change
- successively approximate
- report closest match

![A/D Conversion Diagram]
A-to-D – sample

- During the sample time, the top plate of all capacitors is switched to reference low \( V_L \).
- Bottom plate is set to unknown analog input \( V_X \).
- \( Q = CV \)
- \( Q_S = 16 (V_X - V_L) \)

A-to-D – hold

- Hold state using logically controlled analog switches
  - Top plates disconnected from \( V_L \).
  - Bottom plates switched from \( V_X \) to \( V_L \).
- \( Q_H = 16 (V_L - V_I) \)
- Conservation of charge \( Q_S = Q_H \)
- \( 16 (V_X - V_L) = 16 (V_L - V_I) \)
- \( V_X - V_L = V_L - V_I \) (output of op-amp)

A-to-D – successive approximation

- Each capacitor successively switched from \( V_L \) to \( V_H \)
  - Largest capacitor corresponds to MSB.
  - Output of comparator determines bottom plate voltage of cap.
  - \( > 0 \) : remain connected to \( V_H \).
  - \( < 0 \) : return to \( V_L \).

A-to-D example - MSB

- Suppose \( V_X = 21/32 (V_H - V_L) \) and already sampled.
- Compare after shifting half of capacitance to \( V_H \)
  - \( V_I \) goes up by \( +8/16 (V_H - V_I) - 8/16 (V_L - V_I) = +8/16 (V_H - V_L) \)
  - original \( V_L - V_I \) goes down and becomes
  - \( V_L - (V_I + .5 (V_H - V_L)) = V_L - V_I - .5 (V_H - V_L) \)
- Output > 0
### A-to-D example - (MSB-1)

- Compare after shifting another part of cap. to $V_H$
  - $V_I$ goes up by $+ \frac{4}{16} (V_H - V_L) - \frac{4}{16} (V_L - V_I) = + \frac{4}{16} (V_H - V_I)$
  - original $V_L - V_I$ goes down and becomes
  - $V_L - (V_I + \frac{25}{V_H} (V_H - V_L)) = V_L - V_I - \frac{25}{V_H} (V_H - V_L)$
- Output $< 0$ (went too far)

### A-to-D example - (MSB-2)

- Compare after shifting another part of cap. to $V_H$
  - $V_I$ goes up by $+ \frac{2}{16} (V_H - V_I) - \frac{2}{16} (V_L - V_I) = + \frac{2}{16} (V_H - V_I)$
  - original $V_L - V_I$ goes down and becomes
  - $V_L - (V_I + \frac{125}{2} (V_H - V_L)) = V_L - V_I - \frac{125}{2} (V_H - V_L)$
- Output $> 0$

### A-to-D example - LSB

- Compare after shifting another part of cap. to $V_H$
  - $V_I$ goes up by $+ \frac{1}{16} (V_H - V_I) - \frac{1}{16} (V_L - V_I) = + \frac{1}{16} (V_H - V_I)$
  - original $V_L - V_I$ goes down and becomes
  - $V_L - (V_I + \frac{0.0625}{V_H} (V_H - V_L)) = V_L - V_I - \frac{0.0625}{V_H} (V_H - V_L)$
- Output $< 0$ (went too far again)

### A-to-D example final result

- Input sample of $21/32$
- Gives result of $1010$ or $10/16 = 20/32$
- 3% error
### A-to-D Conversion Errors

- Offset Error
- Integral Non-Linearity (INL)
- Gain Error
- Differential Non-Linearity (DNL)

#### Closest Look at A-to-D Conversion
- Needs a comparator and a D-to-A converter
- Takes time to do successive approximation
- Interrupt generated when conversion is completed

### A-to-D Conversion on the ATmega16
- 10-bit resolution (adjusted to 8 bits as needed)
- 65-260 usec conversion time
- 8 multiplexed input channels
- Capability to do differential conversion
  - Difference of two pins
  - Optional gain on differential signal (amplifies difference)
- Interrupt on completion of A-to-D conversion
- 0-VCC input range
- 2^LSB accuracy (2^1/1024 = ~0.2%)
  - Susceptible to noise — special analog supply pin (AVCC) and capacitor connection for reference voltage (AREF)

### A-to-D Conversion (cont’d)

#### ADC Multiplexer Selection
- 8 multiplexer selection inputs
- 8 channels
- 10-bit accuracy
- Interrupt on conversion completion

#### ADC Reference Voltage
- 0-VCC input range
- 2^LSB accuracy
- Susceptible to noise
- Special analog supply pin (AVCC) and capacitor connection for reference voltage (AREF)

#### Table 3: Voltage Reference Selections for ADC

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-6</td>
<td>REF7-REF6: Reference Selection Bits</td>
</tr>
<tr>
<td></td>
<td>0: VREF internal (VREF turned off)</td>
</tr>
<tr>
<td></td>
<td>1: VREF with external capacitor at AVCC pin</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>1: internal 2.048V Reference Reference with external capacitor at AVREF pin</td>
</tr>
</tbody>
</table>

- Bit 5: ADLAR: ADC Left Adjust Result
- The ADLAR bit affects the presentation of the 8-bit conversion result in the ADC Data Register. If this bit is set to 1, then the 8-bit result will be unaffected. However, the result is right-adjusted if the ADLAR bit is set to 0. For a complete description of the bit, see "The ADC Data Register — ADCH and ADCL" on page 218.
A-to-D Conversion (cont’d)

- Single-ended or differential
  - 1 of 8 single-ended
  - ADCx – ADC1 at 1x gain
  - ADC[0,1] – ADC0 at 10x
  - ADC[0,1] – ADC0 at 200x
  - ADC[2,3] – ADC2 at 10x
  - ADC[2,3] – ADC3 at 200x
  - ADC[0,1,2,3,4,5] – ADC2 at 1x

A-to-D Conversion (cont’d)

The ADC Data Register –
ADC0 and ADC1

A-to-D Conversion (cont’d)

Special Function0/1 Register –

A-to-D Conversion (cont’d)

ADC Control and Status Register A – ADCSCRA

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>ADON</td>
</tr>
<tr>
<td>A0</td>
<td>AIF</td>
</tr>
<tr>
<td>A8</td>
<td>AIFEN</td>
</tr>
<tr>
<td>A7</td>
<td>ASCSRA</td>
</tr>
<tr>
<td>A6</td>
<td>ASPR0</td>
</tr>
<tr>
<td>A5</td>
<td>ASPR1</td>
</tr>
<tr>
<td>A4</td>
<td>ASPR2</td>
</tr>
<tr>
<td>A3</td>
<td>ASPR3</td>
</tr>
<tr>
<td>A2</td>
<td>ASPR4</td>
</tr>
<tr>
<td>A1</td>
<td>ASCSRA</td>
</tr>
<tr>
<td>A0</td>
<td>ASCSRA</td>
</tr>
</tbody>
</table>

A-to-D Conversion (cont’d)

- Bit 4: Reserved Bits
  - Bit 4 is reserved for future use. To ensure compatibility with future devices, this bit must be written to zero when writing to an external device.
Writing an Interrupt Handler in C (again)
- Ensure main program sets up all registers
- Enable interrupts as needed
- Enable global interrupts (SEI)
- Write handler routine for each enabled interrupt
  - What if an interrupt occurs and a handler isn’t defined?
- Make sure routine does not disrupt others
  - Data sharing problem
  - Save any state that might be changed (done by compiler)
- Re-enable interrupts upon return
  - done by compiler with RETI

Power modes
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks
- Processor can go to “sleep” and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont’d)
- Wake up sources and active clocks