ADC Parameters

- **Precision** is number of distinguishable ADC inputs.
- **Range** is maximum and minimum ADC inputs.
- **Resolution** is change in input that causes digital output to change by 1.

\[
\text{Range (volts)} = \text{Precision (alternatives)} \cdot \text{Resolution (volts)}
\]

- **Accuracy** usually given for entire instrument (including transducer, analog circuit, ADC, and software).
- ADC is **monotonic** if it has no missing codes.
- ADC is **linear** if resolution is constant through the range.
- ADC **speed** is time to convert.
Common Encoding Schemes

<table>
<thead>
<tr>
<th>Unipolar codes</th>
<th>Straight binary</th>
<th>Complementary binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5.00</td>
<td>1111,1111</td>
<td>0000,0000</td>
</tr>
<tr>
<td>+2.50</td>
<td>1000,0000</td>
<td>0111,1111</td>
</tr>
<tr>
<td>+0.02</td>
<td>0000,0001</td>
<td>1111,1110</td>
</tr>
<tr>
<td>+0.00</td>
<td>0000,0000</td>
<td>1111,1111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bipolar codes</th>
<th>Offset binary</th>
<th>2s Complement binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5.00</td>
<td>1111,1111</td>
<td>0111,1111</td>
</tr>
<tr>
<td>+2.50</td>
<td>1100,0000</td>
<td>0100,0000</td>
</tr>
<tr>
<td>+0.04</td>
<td>1000,0000</td>
<td>0000,00001</td>
</tr>
<tr>
<td>+0.00</td>
<td>1000,0000</td>
<td>0000,0000</td>
</tr>
<tr>
<td>-2.50</td>
<td>0100,0000</td>
<td>1100,0000</td>
</tr>
<tr>
<td>-5.00</td>
<td>0000,0000</td>
<td>1000,0000</td>
</tr>
</tbody>
</table>

Two-Bit Flash ADC

<table>
<thead>
<tr>
<th>$V_{in}$</th>
<th>X3</th>
<th>X2</th>
<th>X1</th>
<th>Z1</th>
<th>Z0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5 $&gt;$ $V_{in}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5.0 $&gt;$ $V_{in}$ $&gt;$ 2.5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7.5 $&gt;$ $V_{in}$ $&gt;$ 5.0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$V_{in}$ $&gt;$ 7.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Successive Approximation ADC

Sixteen-Bit Dual Slope ADC
Waveforms During Dual Slope ADC Conversion

**Dual Slope ADC Conversion: Theory**

\[
V_{out}(t_0) = 0 \\
V_{out}(t_1) = V_{out}(t_0) - \frac{1}{RC} \int_{t_0}^{t_1} V_{in}(s) ds = -\frac{1}{RC} V_{in} t_{ref} \\
V_{out}(t_2) = 0 \\
V_{out}(t_2) = V_{out}(t_1) - \frac{1}{RC} \int_{t_1}^{t_2} V_{ref}(s) ds = V_{out}(t_1) - \frac{1}{RC} V_{ref} t_{in} = 0 \\
0 = -\frac{1}{RC} V_{in} t_{ref} - \frac{1}{RC} V_{ref} t_{in} \\
0 = V_{in} t_{ref} + V_{ref} t_{in} \\
V_{in} = -V_{ref} \frac{t_{in}}{t_{ref}} = 10V \frac{t_{in}}{65,535\mu s}
\]
**Sigma Delta ADC**

![Diagram of Sigma Delta ADC](image)

**Software Implementation of Sigma Delta ADC**

```c
unsigned char DOUT; // 8-bit sample
unsigned char SUM; // number of times Z=0 and V0=1
unsigned char CNT; // 8-bit counter
void interrupt 13 TOC5handler(void){
    TFLG1 = 0C5; // ack C5F
    TC5 = TC5+rate; // interrupt 256 times faster
    if(Z()) // check input
        DACout(0); // too high, set D/A output, V0=0
    else {
        DACout(1); // too low, set D/A output, V0=+5v
        SUM++;
    }
    if(++CNT==0){ // end of 256 loops?
        DOUT = SUM; // new sample
        SUM = 0; // get ready for the next
    }
}
Sample and Hold

- Should use polystyrene capacitor because of its high insulation resistance and low dielectric absorption.
- A larger value of $C$ decreases (improves) droop rate. If droop current is $I_{DR}$, then droop rate is:
  $$\frac{dV_{out}}{dt} = \frac{I_{DR}}{C}$$
- A smaller $C$ decreases (improves) acquisition time.
BiFET Analog Multiplexer

<table>
<thead>
<tr>
<th>Enable A_2 A_1 A_0</th>
<th>Drain</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 X X X</td>
<td>None</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>S_8</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>S_7</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>S_6</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>S_5</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>S_4</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>S_3</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>S_2</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>S_1</td>
</tr>
</tbody>
</table>

Bilateral Switch

"Good circuit"

"Bad circuit"
Variable-Gain Amplifier

ADC Block Diagram
Power and Grounding for the ADC System

Input Protection for CMOS Analog Inputs
6812 has built-in ADCs with following features:

- Eight-channel operation.
- 8-bit or 10-bit resolution.
- Successive approximation conversion technique.
- Clock and charge pump to create higher voltages.
- Two operation modes: single sequence of conversions then stop, and continuous conversion.
- Supports multiple conversion of single channel, and one conversion each for group of channels.
- External $V_{RH}$, $V_{RL}$ analog high/low references.

6812 ADC System: Setup

- 8 pins of Port AD can be individually configured as analog or digital inputs using the ATDDIEN register (1 for digital, 0 for analog).
- If pin is digital, DDRAD register is used to set pins direction.
- Can use 8-bit or 10-bit resolution which is selected by setting the SRES8 bit (1=8-bit) in the ADTCTL4 register.
- ATDCTL2 register:
  - ADC system is enabled by setting ADPU to 1.
  - Interrupts are enabled by setting ASCIE to 1.
  - ASCIF flag is set 1 when conversion sequence is complete, if ASCIE is 1.
6812 ADC System: Conversions

- When ADC is triggered, it performs 1 to 8 conversions.
- Number of conversions is selected by the value written into the S8C, S4C, S2C, and S1C bits of ATDCTL3 (values of 0, 8-15 are all 8).
- The channel used is selected by the CC, CB, CA bits of ATDCTL5.
- All conversions can be on one channel or on multiple channels if MULT in ATDCTL5 is set (channel sequence determined by S8C, S4C, S2C, S1C).
- ATDSTAT0 register:
  - SCF flag in ATDSTAT0 is set to 1 when conversion is complete.
  - CC2, CC1, and CC0 bits are a counter to show conversion progress.
- ATDSTAT1 register contains CCFn flag bits for each conversion.

6812 ADC System: Triggers

- Conversion can be triggered in three ways:
  - Writing to ATDCTL5 and when done SCF bit in ATDSTAT0 is set.
  - Trigger continuously if SCAN in ATDCTL5 is set.
  - Using an external trigger connected to PAD7.
- External trigger enabled when ETRIGE bit in ATDCTL2 is set.

<table>
<thead>
<tr>
<th>ETRIGLE</th>
<th>ETRIGP</th>
<th>External trigger mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Falling edge of PAD7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Rising edge of PAD7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Convert while PAD7 is low</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Convert while PAD7 is high</td>
</tr>
</tbody>
</table>
6812 ADC System: Sample Period

- Determined by ATDCTL4 register and E clock.
- Sample done in two phases:
  - Phase one transfers sample to ADC’s storage node.
  - Phase two attaches external analog signal to the storage node.

<table>
<thead>
<tr>
<th>MP1</th>
<th>MP0</th>
<th>First sample</th>
<th>Second sample</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2 ADC clocks</td>
<td>2 ADC clocks</td>
<td>4 ADC</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2 ADC clocks</td>
<td>4 ADC clocks</td>
<td>6 ADC</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2 ADC clocks</td>
<td>8 ADC clocks</td>
<td>10 ADC</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2 ADC clocks</td>
<td>16 ADC clocks</td>
<td>18 ADC</td>
</tr>
</tbody>
</table>

- If \( m \) is 5-bit number formed by PRS4-0 and \( f_E \) is E clock frequency:

\[
\text{ATD clock frequency} = \frac{1}{2 (m + 1)} \frac{f_E}{E}
\]

6812 ADC System: Binary Formats

- Results returned in the 16-bit ATDDR0 to ATDDR7 registers.
- 10-bit results can unsigned or signed (DSGN=1 in ATDCTL5).
- Results can be left or right justified (DJM=1 in ATDCTL5).

<table>
<thead>
<tr>
<th>Input (V)</th>
<th>8-bit (u)</th>
<th>10-bit (ur)</th>
<th>10-bit (ul)</th>
<th>10-bit (sr)</th>
<th>10-bit (sl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>$00</td>
<td>$0000</td>
<td>$0000</td>
<td>$FE00</td>
<td>$8000</td>
</tr>
<tr>
<td>0.005</td>
<td>$00</td>
<td>$0001</td>
<td>$0040</td>
<td>$FE01</td>
<td>$8040</td>
</tr>
<tr>
<td>0.020</td>
<td>$01</td>
<td>$0004</td>
<td>$0100</td>
<td>$FE04</td>
<td>$8100</td>
</tr>
<tr>
<td>2.500</td>
<td>$80</td>
<td>$0200</td>
<td>$8000</td>
<td>$0000</td>
<td>$0000</td>
</tr>
<tr>
<td>3.750</td>
<td>$C0</td>
<td>$0300</td>
<td>$C000</td>
<td>$0100</td>
<td>$4000</td>
</tr>
<tr>
<td>5.000</td>
<td>$FF</td>
<td>$03FF</td>
<td>$FFC0</td>
<td>$01FF</td>
<td>$7FC0</td>
</tr>
</tbody>
</table>
ADC Software

```c
void ADC_Init(void){
    ATDCTL2 = 0x80; // enable ADC
    ATDCTL3 = 0x08;
    ATDCTL4 = 0x05; // 10-bit, divide by 12
}
unsigned short ADC_In(unsigned short chan){
    ATDCTL5 = (unsigned char)chan; // start sequence
    while((ATDSTAT1&0x01)==0){}; // wait for CCF0
    return ATDDRO;
}
```
Computing a Derivative

- Simple approach:
  \[ d(n) = \frac{x(n) - x(n - 1)}{\Delta t} \]

- Approach that is more robust to noise:
  \[ d(n) = \frac{x(n) - 3x(n - 1) - 3x(n - 2) - x(n - 3)}{\Delta t} \]
#define RATE 2000
#define OC5 0x20
unsigned short x[4];    // MACQ (mV)
unsigned short d;       // derivative (V/s)
void interrupt 13 TOC5handler(void){
    TC5 = TC5+RATE;     // Executed every 1 ms
    TFLG1 = 0x20;       // ack OC5F
    x[3] = x[2];        // shift MACQ data
    x[2] = x[1];        // units of mV
    x[1] = x[0];
    x[0] = ADC_In(0x85); // current data, from Channel 5
    d = x[0]+3*x[1]-3*x[2]-x[3]; // mV/ms
}