Simple Active Filter

\[ \frac{V_{\text{out}}}{V_{\text{in}}} = G \cdot \sqrt{\frac{1}{1 + \left( \frac{f}{f_c} \right)^2}} \]

\[ f_c = \frac{1}{2\pi R_3 C} \]

\[ R_3 = \frac{R_1 \cdot R_2}{R_1 + R_2} \]

\[ f_c = \frac{R_2}{1 + j\omega R_2 C} \]
Two-Pole Butterworth Low-Pass Analog Filter

Select the cutoff frequency $f_c$.
Divide the two capacitors by $2\pi f_c$.
$C_{1A} = \frac{141.4\mu F}{2\pi f_c}$
$C_{2A} = \frac{70.7\mu F}{2\pi f_c}$
Select standard capacitors with same order of magnitude.
$C_{1B} = \frac{C_{1A}}{x}$
$C_{2B} = \frac{C_{2A}}{x}$
Adjust resistors to maintain $f_c$ (i.e., $R = 10k\Omega \cdot x$).

Bandpass Filters

High-pass filter
Low-pass filter
$Q = \frac{f_0}{\Delta f}$
Band-Reject Filters

Multiple Feedback Bandpass Filter

- Select a convenient capacitance value for the two capacitors.
- Calculate the three resistor values for $x = 1/(2\pi f_0 C)$.
  
  $R_1 = Q \cdot x \quad R_2 = x/(2Q - 1/Q) \quad R_3 = 2 \cdot Q \cdot x$
- Resistors should be in the 5kΩ to 5MΩ range. If not, repeat with different capacitance value.
**Digital-to-Analog Converters**

**DAC Parameters**

- **Precision** is number of distinguishable DAC outputs.
- **Range** is maximum and minimum DAC output.
- **Resolution** is smallest distinguishable change in output.

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

- **Accuracy** is (actual-ideal)/ideal.
- Two common encoding schemes:

\[
V_{out} = V_{fs} \left( \frac{b_7}{2} + \frac{b_6}{4} + \frac{b_5}{8} + \frac{b_4}{16} + b_3 + \frac{b_2}{32} + \frac{b_1}{64} + \frac{b_0}{128} + \frac{b_0}{256} \right) + V_{os}
\]

\[
V_{out} = V_{fs} \left( -\frac{b_7}{2} + \frac{b_6}{4} + \frac{b_5}{8} + \frac{b_4}{16} + b_3 + \frac{b_2}{32} + \frac{b_1}{64} + \frac{b_0}{128} + \frac{b_0}{256} \right) + V_{os}
\]
Three-Bit DAC Examples

DAC Performance Measures
DAC Errors: Sources and Solutions

<table>
<thead>
<tr>
<th>Errors can be due to</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect resistor values</td>
<td>Precision resistors w/low tolerances</td>
</tr>
<tr>
<td>Drift in resistor values</td>
<td>Precision resistors w/good temperature coefficients</td>
</tr>
<tr>
<td>White noise</td>
<td>Reduce BW w/low pass filter, reduce temperature</td>
</tr>
<tr>
<td>Op amp errors</td>
<td>Use more expensive devices w/low noise and low drift</td>
</tr>
<tr>
<td>Interference from external fields</td>
<td>Shielding, ground planes</td>
</tr>
</tbody>
</table>

DAC Using a Summing Amplifier

\[ V_{out} = b_2 \cdot 25k\Omega + b_1 \cdot 50k\Omega + b_0 \cdot 100k\Omega \]

<table>
<thead>
<tr>
<th>( b_2 )</th>
<th>( b_1 )</th>
<th>( b_0 )</th>
<th>( V_{out} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>+4</td>
</tr>
</tbody>
</table>
Three-Bit DAC with an R-2R Ladder

\[ V_{\text{out}} = +1 \text{V} \]
Three-Bit DAC with an R-2R Ladder

\[ V_{\text{out}} = +2V \]

Three-Bit DAC with an R-2R Ladder

\[ V_{\text{out}} = +4V \]
Variable-Offset and Gain Using 3-bit DACs

Twelve-Bit DAC with a DAC8043

<table>
<thead>
<tr>
<th>Digital Input</th>
<th>Unipolar $V_{out}$</th>
<th>Bipolar $V_{out}$</th>
<th>Unipolar gain</th>
<th>Bipolar gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111,1111,1111</td>
<td>-4.999</td>
<td>4.998</td>
<td>-4095</td>
<td>+2047</td>
</tr>
<tr>
<td>1000,0000,0001</td>
<td>-2.501</td>
<td>0.002</td>
<td>-4095</td>
<td>+2048</td>
</tr>
<tr>
<td>1000,0000,0000</td>
<td>-2.500</td>
<td>0.000</td>
<td>-4096</td>
<td>+2048</td>
</tr>
<tr>
<td>0111,1111,1111</td>
<td>-2.499</td>
<td>-0.002</td>
<td>-4096</td>
<td>-2047</td>
</tr>
<tr>
<td>0000,0000,0001</td>
<td>-0.001</td>
<td>-4.998</td>
<td>-3</td>
<td>-2049</td>
</tr>
<tr>
<td>0000,0000,0000</td>
<td>0.000</td>
<td>-5.000</td>
<td>-4096</td>
<td>-2048</td>
</tr>
</tbody>
</table>
**DAC Selection: Precision, Range, and Resolution**

- Affect quality of signal that can be generated.
- More bits means finer control over the waveform.
- Can be hard to specify a priori.

![Diagram of DAC output compared to desired waveform]

**DAC Selection: Channels, Configuration, and Speed**

- Usually more efficient to implement multiple *channels* using a signal DAC.
- *Configuration*: can have voltage or current outputs, internal or external references, etc.
- *Speed* specified in many ways: *settling time*, *maximum output rate*, *gain/BW product*, etc.

![Diagram of DAC output compared to desired waveform]
DAC Selection: Power and Interface

- Three power issues: type of power required, amount of power required, and need for low-power sleep mode.
- Three approaches for interfacing exist:

DAC Selection: Package and Cost

- Variety of packages exist:

- Cost includes direct cost of components, power supply requirements, manufacturing costs, labor in calibration, and software development costs.
unsigned short wave(unsigned short t){
    float result, time;
    time = 2*pi*((float)t)/1000.0;
    // integer t in msec into floating point time in seconds
    result = 2048.0+1000.0*cos(31.25*time)-500.0*sin(125.0*time);
    return (unsigned short) result;
}
#define RATE 2000
#define OC5 0x20
unsigned short Time; // Inc every 1ms
void interrrupt 13 T0C5handler(void){
    TFLG1 = OC5; // ack C5F
    TC5 = TC5+RATE; // Executed every 1 ms
    Time++;
    DACout(wave(Time));
}
Periodic Interrupt Used to Generate Waveform

```c
unsigned short I; // incremented every 1ms
const unsigned short wave[32] = {
    3048, 2675, 2472, 2526, 2755, 2957, 2931, 2597,
    2048, 1499, 1165, 1139, 1341, 1570, 1624, 1421,
    1048, 714, 624, 863, 1341, 1846, 2165, 2206, 2048,
    1890, 1931, 2250, 2755, 3233, 3472, 3382};
#define RATE 2000
#define OC5 0x20
void interrupt 13 TOC5handler(void){
    TFLG1 = OC5;  // ack C5F
    TC5 = TC5+RATE;  // Executed every 1 ms
    if((++I)==32) I = 0;
    DACout(wave[I]);
}
```

Generated Waveform Using Linear Interpolation

![Generated waveform using linear interpolation](image_url)
Periodic Interrupt Used to Generate Waveform

short I; // incremented every 1ms
short J; // index into these two tables
const short t[10] = {0, 2, 6, 10, 14, 18, 22, 25, 30, 32};
const short wave[10] = {3048, 2472, 2931, 1165, 1624, 624, 2165, 1890, 3472, 3048};

#define RATE 2000
#define OC5 0x20
void interrupt 13 TOC5handler(void){
    TFLG1 = OC5; // ack C5F
    TC5 = TC5+RATE; // Executed every 1 ms
    if((++I)==32) {I=0; J=0;}
    if(I==t[J])
        DACout(wave[J]);
    else if (I==t[J+1]){
        J++;
        DACout(wave[J]);
    } else
        DACout(wave[J]+((wave[J+1]-wave[J])
            *(I-t[J]))/(t[J+1]-t[J]));
}
Generated Waveform Using Uneven-Time

Periodic Interrupt to Generate Analog Waveform

```c
unsigned short I;  // incremented every sample
const unsigned short wave[32]= {
    3048,2675,2472,2526,2817,2981,2800,2337,1901,1499,1165,
    1341,1570,1597,1337, 952, 662, 654, 863,1210,1605,1950,
    2202,2141,1955,1876,2057,2366,2755,3129,3442,3382};
const unsigned short dt[32]= { // 500 ns cycles
    2000,2000,2000,2500,2500,2000,2000,1500,1500,2000,4000,
    2000,2500,2000,2000,2000,2000,1500,1500,1500,1500,2000,
    2500,2000,2000,2000,1500,1500,1500,2000,2500,2000};
#define OC5 0x20

void interrupt 13 TOC5handler(void){
    TFLG1 = OC5;   // ack C5F
    if((++I)==32) I=0;
    TC5 = TC5+dt[I];   // variable rate
    DACout(wave[I]);}
```