*Filters* are used to suppress unwanted frequencies in a signal, or enhance wanted ones.

Filters can be:

- Analog or digital
- Passive or active
- High-pass, low-pass, band-pass, band-reject, or all-pass
- Discrete time or continuous time
Passive filters do not depend on an external power supply
Inductors block high-frequency signals and pass low-frequency signals
Capacitors block low-frequency signals and pass high-frequency signals
When the signal passes through an inductor, or when a capacitor provides a path to ground, we have a low-pass filter
When the signal passes through a capacitor, or has a path to ground through an inductor, we have a high-pass filter
Resistors are used in filters to help select the operating frequencies, but by themselves are not frequency-selective
Simple Active Filter

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

\[
f_c = \frac{1}{2\pi R_2 C}
\]

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

\[
\frac{V_{out}}{V_{in}} = -\frac{R_2}{R_1} \frac{R_2}{1 + j\omega R_2 C}
\]
Select the cutoff frequency $f_c$.
Divide the two capacitors by $2\pi f_c$.

$$C_{1A} = \frac{141.4 \mu F}{2\pi f_c} \quad C_{2A} = \frac{70.7 \mu F}{2\pi f_c}$$

Select standard capacitors with same order of magnitude.

$$C_{1B} = \frac{C_{1A}}{x} \quad C_{2B} = \frac{C_{2A}}{x}$$

Adjust resistors to maintain $f_c$ (i.e., $R = 10k\Omega \cdot x$).
Bandpass Filters

- High pass
- Low pass
- Bandpass filter

$Q = \frac{f_0}{\Delta f}$
Band-Reject Filters

High-pass filter

Low-pass filter

\[ Q = \frac{f_0}{\Delta f} \]
Digital-to-Analog Converters

A DAC is used to create an analog waveform from digital data. In practice, DAC output is usually processed by an analog low-pass filter to remove unwanted high-frequency components; this is called a reconstruction filter. PWM is a form of DAC. PWM is efficient because it delivers variable power without using resistors. Common in motor control. Becoming common in audio; if you see a “class D amplifier” it is based on PWM.
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:

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

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

- **Output Voltage (Volts)**
  - $V_{out} = b_2 b_1 b_0$
  - $V_{out}$ can range from 0 to 7 volts.

- **Offset**
  - $V_{out} = V_{in} + b$
  - $b$ is a binary input that shifts the output by a fixed voltage.

- **Gain**
  - $V_{out} = m \cdot V_{in}$
  - $m$ is a scalar that scales the input voltage.

Digital inputs are processed by the DAC (Digital-to-Analog Converter) to produce analog output voltages.
DAC Performance Measures

- Offset error
- Gain error
- Ideal

- Digital input
- Time
- Delay
- Slew
- Ringing phases
- Digital input
- Time

- Nonlinear
- Ideal

- Nonmonotonic
- Ideal
## 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</td>
</tr>
<tr>
<td></td>
<td>w/low tolerances</td>
</tr>
<tr>
<td>Drift in resistor values</td>
<td>Precision resistors</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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

\[
\begin{array}{cccc}
 b_2 & b_1 & b_0 & V_{out} \\
 0 & 0 & 1 & +1 \\
 0 & 1 & 0 & +2 \\
 1 & 0 & 0 & +4 \\
\end{array}
\]
Three-Bit DAC with an R-2R Ladder
Three-Bit DAC with an R-2R Ladder
Three-Bit DAC with an R-2R Ladder
Three-Bit DAC with an R-2R Ladder
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>$-\frac{4095}{4096}$</td>
<td>$+\frac{2047}{2048}$</td>
</tr>
<tr>
<td>1000,0000,0001</td>
<td>-2.501</td>
<td>0.002</td>
<td>$-\frac{4096}{2049}$</td>
<td>$+\frac{1}{2048}$</td>
</tr>
<tr>
<td>1000,0000,0000</td>
<td>-2.500</td>
<td>0.000</td>
<td>$-\frac{4096}{2048}$</td>
<td>$+\frac{1}{2048}$</td>
</tr>
<tr>
<td>0111,1111,1111</td>
<td>-2.499</td>
<td>-0.002</td>
<td>$-\frac{4096}{2047}$</td>
<td>$-\frac{1}{2048}$</td>
</tr>
<tr>
<td>0000,0000,0001</td>
<td>-0.001</td>
<td>-4.998</td>
<td>$-\frac{4096}{2047}$</td>
<td>$-\frac{2048}{2048}$</td>
</tr>
<tr>
<td>0000,0000,0000</td>
<td>0.000</td>
<td>-5.000</td>
<td>$-\frac{4096}{2047}$</td>
<td>$-\frac{2048}{2048}$</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.
DAC Selection: Channels, Configuration, and Speed

Usually more efficient to implement multiple channels using a single 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.

![Generated waveform](image1)

![Desired waveform](image2)

![Generated waveform](image3)

![Desired waveform](image4)
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.
DAC Waveform Generation

The graph shows the comparison between the generated waveform and the desired waveform over time. The vertical axis represents the DAC value, ranging from 0 to 3000. The horizontal axis represents time, ranging from 0 to 30 milliseconds. The generated waveform is indicated by a dotted line, while the desired waveform is a solid line. The points on the graph represent the values at specific time intervals.
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 interrupt 13 T0C5handler(void){
    TFLG1 = OC5;    // ack C5F
    TC5 = TC5+RATE; // Executed every 1 ms
    Time++;
    DACout(wave(Time));
}
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 T0C5handler(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
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};
Periodic Interrupt Used to Generate Waveform

```c
#define RATE 2000
#define OC5 0x20
void interrupt 13 T0C5handler(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

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, 1500, 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]);}