Two-Pole Butterworth Low-Pass Analog Filter

1. Select the cutoff frequency $f_c$.
2. Divide the two capacitors by $2\pi f_c$.
   
   $$C_{1A} = \frac{141.4 \mu F}{2\pi f_c} \quad C_{2A} = \frac{70.7 \mu F}{2\pi f_c}$$

3. Select standard capacitors with same order of magnitude.
   
   $$C_{1B} = \frac{C_{1A}}{4} \quad C_{2B} = \frac{C_{2A}}{4}$$
4. Adjust resistors to maintain $f_c$ (i.e., $R = 10k \cdot x$).

Three-Pole Butterworth Low-Pass Analog Filter

1. Select the cutoff frequency $f_c$.
2. Divide the three capacitors by $2\pi f_c$.
   
   $$C_{1A} = \frac{354.6 \mu F}{2\pi f_c} \quad C_{2A} = \frac{109.2 \mu F}{2\pi f_c} \quad C_{3A} = \frac{20.2 \mu F}{2\pi f_c}$$

3. Select standard capacitors with same order of magnitude.
   
   $$C_{1B} = \frac{C_{1A}}{4} \quad C_{2B} = \frac{C_{2A}}{4} \quad C_{3B} = \frac{C_{3A}}{4}$$
4. Adjust resistors to maintain $f_c$ (i.e., $R = 10k \cdot x$).
Two-Pole Butterworth High-Pass Analog Filter

1. Select the cutoff frequency $f_c$.
2. Divide the two capacitors by $2\pi f_c$.
   \[ C_A = \frac{1}{2\pi f_c} \]
3. Select standard capacitors with same order of magnitude.
   \[ C_B = C_A \]
4. Adjust resistors to maintain $f_c$.
   \[ R_1 = 107k\Omega \cdot x \quad R_2 = 1414k\Omega \cdot x \]

Three-Pole Butterworth High-Pass Analog Filter

1. Select the cutoff frequency $f_c$.
2. Divide the two capacitors by $2\pi f_c$.
   \[ C_A = \frac{1}{2\pi f_c} \]
3. Select standard capacitors with same order of magnitude.
   \[ C_B = C_A \]
4. Adjust resistors to maintain $f_c$.
   \[ R_1 = 282k\Omega \cdot x \quad R_2 = 718k\Omega \cdot x \quad R_3 = 4960k\Omega \cdot x \]

Bandpass and Band-Reject Filters

1. Select the cutoff frequency $f_c$.
2. Divide the two capacitors by $2\pi f_c$.
   \[ C_A = \frac{1}{2\pi f_c} \]
3. Select standard capacitors with same order of magnitude.
   \[ C_B = C_A \]
4. Adjust resistors to maintain $f_c$.
   \[ R_1 = 107k\Omega \cdot x \quad R_2 = 1414k\Omega \cdot x \]

Multiple Feedback Bandpass Filter

1. Select a convenient capacitance value for the two capacitors.
2. Calculate the three resistor values for $x = 1/(2\pi f_0 C)$.
   \[ R_1 = Q \cdot x \quad R_2 = x/(2Q - 1/Q) \quad R_3 = 2 \cdot Q \cdot x \]
3. Resistors should be in the 5kΩ to 5MΩ range. If not, repeat with different capacitance value.
**Slide 9**

**Bootstrapped Twin-T Band-Reject Filter**

- The notch frequency $f_0$ is:
  \[ f_0 = \frac{1}{2\pi R_1 C_1} = 60\text{Hz} \]

  where $R_1 = R_2 = 2 \cdot R_3$ and $C_1 = C_2 = 0.5 \cdot C_3$.

**Slide 11**

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

**Slide 10**

**Digital-to-Analog Converters**

- Embedded Microcomputer System
- Parallel port
- Serial port
- D/A
- analog processing
- digital voltage or current

**Slide 12**

**Three-Bit DAC Examples**
DAC Performance Measures

DAC Errors: Sources and Solutions

- Incorrect resistor values: Precision resistors w/low tolerances
- Drift in resistor values: Precision resistors w/good temperature coefficients
- White noise: Reduce BW w/low pass filter, reduce temperature
- Op amp errors: Use more expensive devices w/low noise and low drift
- Interference from external fields: Shielding, ground planes

Signed DAC Using a Summing Amplifier
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>-</td>
<td>+</td>
</tr>
<tr>
<td>1000,0000,0001</td>
<td>-2.501</td>
<td>0.002</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>1000,0000,0000</td>
<td>-2.500</td>
<td>0.000</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>0111,1111,1111</td>
<td>-2.499</td>
<td>-0.002</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>0000,0000,0001</td>
<td>-0.001</td>
<td>-4.998</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>0000,0000,0000</td>
<td>0.000</td>
<td>-5.000</td>
<td>-</td>
<td>+</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 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.

DAC Selection: Power and Interface

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

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

Periodic Interrupt Used to Generate Waveform

unsigned int wave(unsigned int t){
    float result, time;
    time = 2*3.141592654f * (float)t / 1000.0;
    result = 2048.0 + 1000.0 * cos(31.25f * time) - 500.0 * sin(125.0f * time);
    return (unsigned int) result;
}

#define Rate 2000
#define OC5 0x08
#pragma interrupt_handler TOC5_handler()
void TOC5_handler(void){
    TFLG1 = OC5; // Ack interrupt
    TOC5 = TOC5 + Rate; // Executed every 1 ms
    if (++I == 32) I = 0;
    DACout(wave[I]);
}

Periodic Interrupt Used to Generate Waveform

unsigned int I; // incremented every 1ms

#define Rate 2000
#define OC5 0x08
#pragma interrupt_handler TOC5_handler()
void TOC5_handler(void){
    TFLG1 = OC5; // Ack interrupt
    TOC5 = TOC5 + Rate; // Executed every 1 ms
    if (++I == 32) I = 0;
    DACout(wave[I]);
}

Generated Waveform Using Linear Interpolation
Periodic Interrupt Used to Generate Waveform

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

Periodic Interrupt Used to Generate Waveform

#define Rate 2000
#define OC5 0x08
#pragma interrupt_handler TOC5handler()
void TOC5handler(void){
    TFLG1=OC5; // Ack interrupt
    TOC5=TOC5+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]));}