Simple Active Filter

\[ V_{\text{out}} = \frac{R_2}{R_1} \frac{R_2}{R_1 + j \omega R_2 C} \]

\[ L_\text{c} = \frac{1}{2 \pi R_1 C} \]

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

\[ f_{\text{c}} \text{ Frequency (f)} \]

Two-Pole Butterworth Low-Pass Analog Filter

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

Bandpass Filters

- High pass filter
- Low pass filter
- Bandpass filter
- Low-pass filter
### Band-Reject Filters

- **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.**

### Multiple Feedback Bandpass Filter

### Digital-to-Analog Converters

- **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:**

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

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

DAC Errors: Sources and Solutions

Errors can be due to  Solutions
Incorrect resistor values  Precision resistors
Drift in resistor values  Precision resistors
White noise  Reduce BW w/low pass filter, reduce temperature
Op amp errors  Use more expensive devices
Interference from external fields  Shielding, ground planes

DAC Using a Summing Amplifier
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

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.
- Three approaches for interfacing exist:

### DAC Selection: Package and Cost

- Variety of packages exist:

### DAC Waveform Generation

```c
unsigned short wavem(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;
}
```

### Periodic Interrupt Used to Generate Waveform

```c
#define RATE 2000
#define OC5 0x20
unsigned short Time; // Inc every 1ms
void interrupt 13 TOC5Handler(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, 1391, 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]);
}

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

Periodic Interrupt to Generate Analog Waveform