Strukturierte Vorgehensweise bei der Entwicklung von AudioDSP Modulen

1. Modul in C++ programmieren

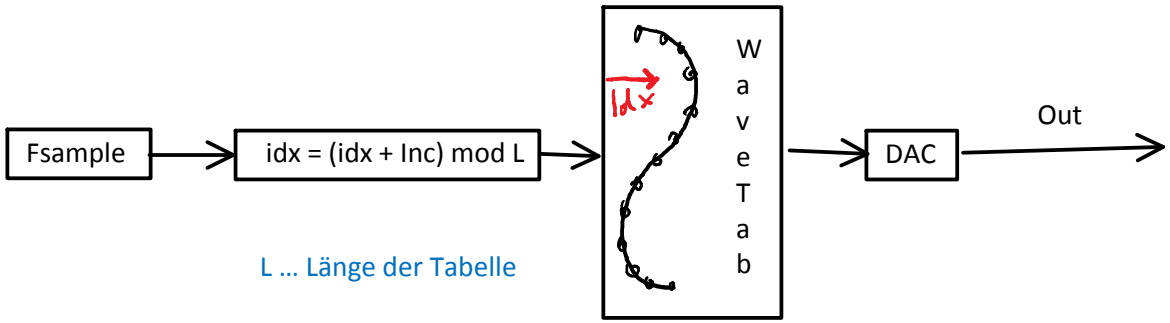
2. Einfachen Testfall *(z.B. nur kleine Wavetables)* überlegen

3. Testfall programmieren und die generierten Daten *(z.B. Ausgangssignal eines Oszillators)* auf Datenfile schreiben

4. Generierte Daten mit Matlab einlesen und plotten.
   Ggf. mit Matlab eine Sollkurve plotten und die generierten Daten für die Analyse über die Sollkurve plotten.

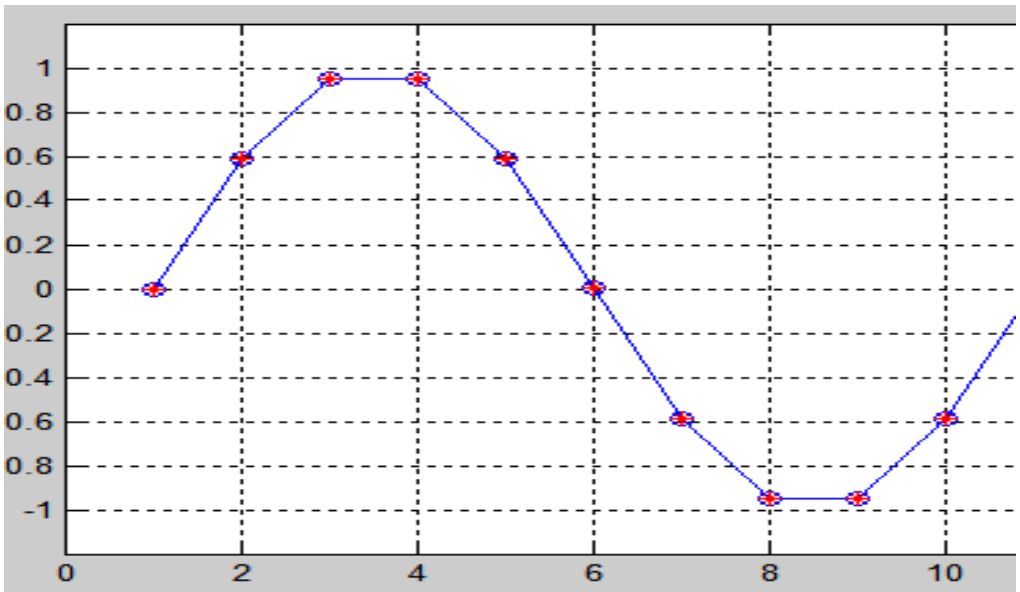
5. DSP-Modul in ein RackAFX Modul verpacken, mit Parametern versorgen
   und in Echtzeit *(Anhören)* austesten.

Prinzipielle Funktionsweise von Wavetable Oszillatoren



L … Länge der Tabelle

In der WaveTable sind **L-Abtastpunkte** einer beliebeigen periodischen Schwingung gespeichert
Im untenstehenden Bild ist dieWaveTable eines Sinus mit **L=10** zu sehen.



Für `Fs=1kHz` und `Inc=1` würde sich ein `Fout=100Hz` ergeben

Für `Inc=2` würde sich ein `Fout=200Hz` ergeben

Für `Inc=0.5` würde sich ein `Fout=50Hz` ergeben

$$Inc = TabLen \cdot \frac{Fosz}{Fsample}$$

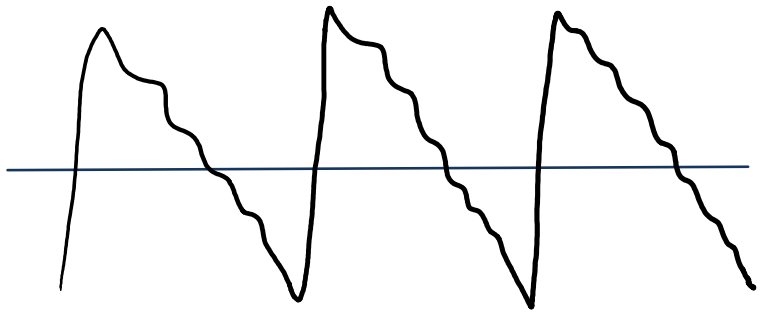
Inkement zum Auslesen der Wavetable als Funktion der gewünschten **`Fosz`**

Wavetable Oszillatoren machen sich die Tatsache zunutze, daß die Frequenz eines period. abgetasteten Signals nur von *PointsPerPeriod* festgelegt wird.

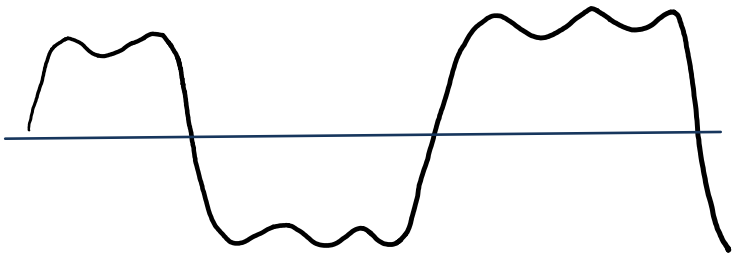
Vorteile von Wavetable Oszillatoren

- **Performance!!** Ist wesentlich schneller als math.sin()
- Es können beliebige periodische Kurvenformen erzeugt werden bis hin zu Samples von Instrumentenklängen

Bandbegrenzte Sägezahn (WaveTable)
Die WaveTable einer Violine würde ziemlich ähnlich aussehen

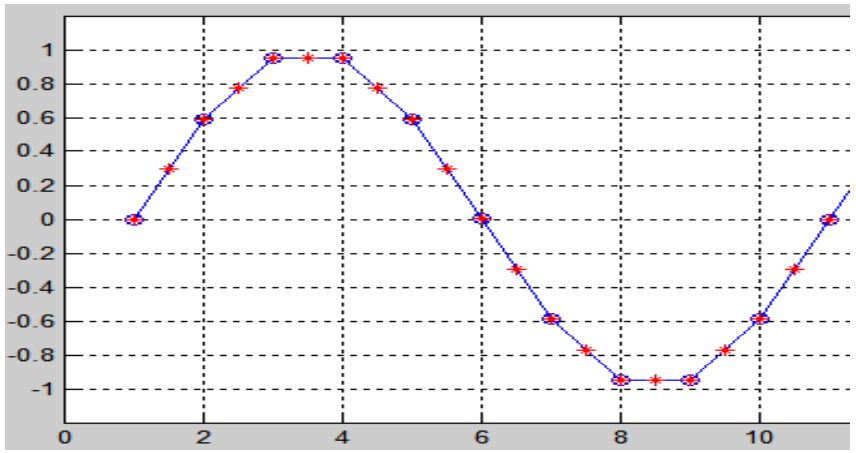


Bandbegrenzte Rechteschwingung (WaveTable)

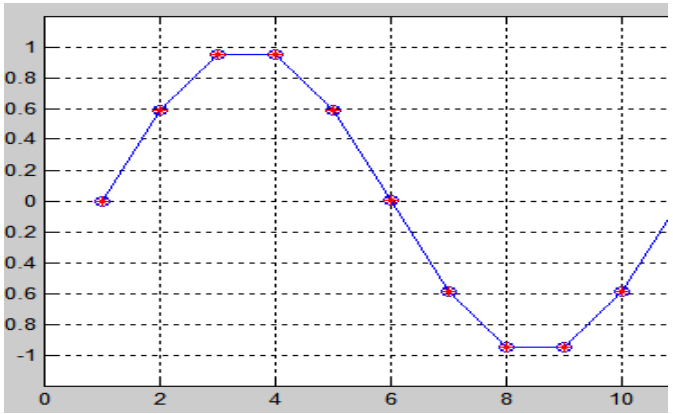


```
40kHz_Audio_ISR  // Ueffizient  und langsam
{
   t  =  t + Tsamp;
   x = math.sin(2*pi*t);
   DAC = x;
}
```

Inc = 0.5

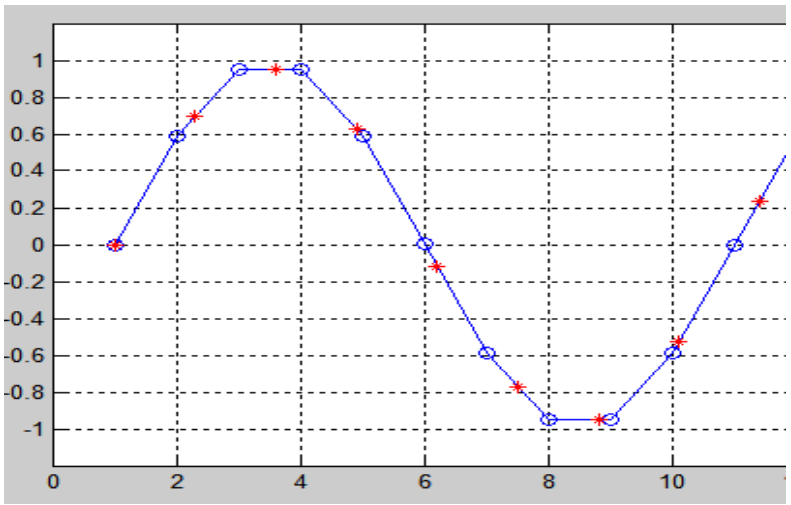


Inc = 1.0



Inc=**1.0 und Inc=2.0** sind triviale Fälle
Bei **Inc=1.0** wird jeder Wert aus der WaveTable verwendet bei **Inc=2.0** wird jeder 2te Wert verwendet

Inc = 1.3



Aber wie funktionieren
Inc=0.5 und **Inc=1.3**

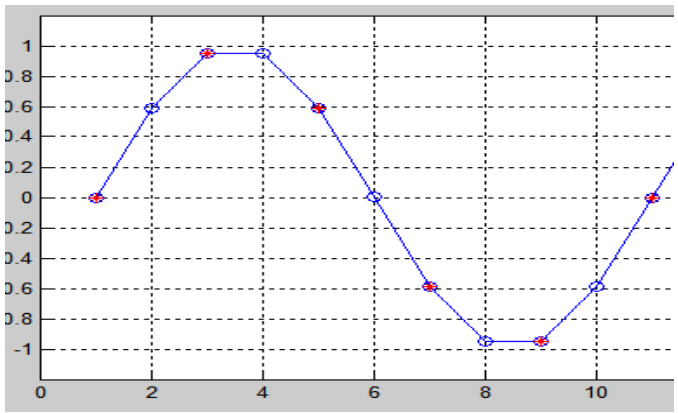
Die auszugebenden Werte sind **nicht**
in der WaveTable enthalten
Die Werte müssen berechnet werden.

Für die Berechnung der Zwischenwerte kommen die folgenden Verfahren in Frage:

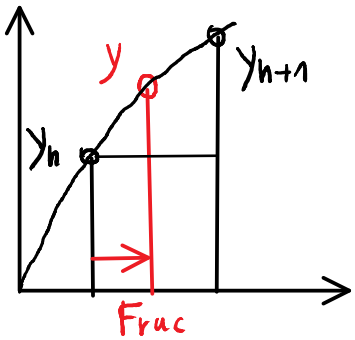
- Linare Interpolation
- Spline Interpolation

Die Linare-Interpolation verwendet Geradenstücke zw. den bekannten Punkten der WaveTable.
Die Spline-Interpolation verwendet Kurvenstücke zw. den bekannten Punkten der WaveTable.

Inc = 2.0



Wie funktioniert die lineare Interpolation



x-Achse

$$y = y_n + (y_{n+1} - y_n) \cdot Frac$$

oder

$$y = y_n \cdot (1 - Frac) + y_{n+1} \cdot Frac$$

Frac ist der Teil hinter dem Komma
Frac ist zw. 0 .. 0.99999

Angenommen wir brauchen den Wert Y(2.3) dann ist X = 2.3
und Frac = 0.3

Xn = 2; Xn+1 = 3;
Yn=Y[2] Yn+1 = Y[3]

Als C++ Code

```
void WaveTGenF::CalcOneStep()
{
  _pos += _inc;
  if( _pos >= _N )
    _pos -= _N;

  // frac-Part berechnen
  int i = floor(_pos);
  double frac = _pos - i;

  // i+1 berechnen
  int ii = i + 1;
  if( ii >= _N )
    ii = 0;

  // Y = Yn + (Yn+1 - Yn)*Frac
  val = _tab[i] + (_tab[ii] - _tab[i])*frac;
}
```

Wichtige Formeln für Wavetable Oszillatoren

$$F_{osz} = \frac{F_{sample}}{TabLen}$$

absolute Ausgangsfrequenz des Oszillators

$$Inc = \frac{TabLen}{PointsPerPeriod}$$

Inkement zum Auslesen der Wavetable als Funktion der gewünschten **`PointsPerPeriod`**

$$\frac{F_{osz}}{F_{sample}} = \frac{1}{TabLen}$$

Ausgangsfrequenz des Oszillators bezogen auf die Abtastfrequenz

$$Inc = TabLen \cdot \frac{F_{osz}}{F_{sample}}$$

Inkement zum Auslesen der Wavetable als Funktion der gewünschten **`Fosz`**

F_{osz} ... absolute Ausgangsfrequenz des Oszillators

F_{sample} ... Abtastfrequenz

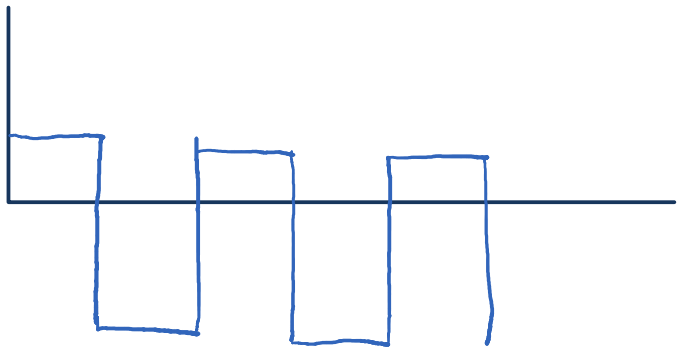
$TabLen$ ... Länge der Wavetable in Abtastpunkten

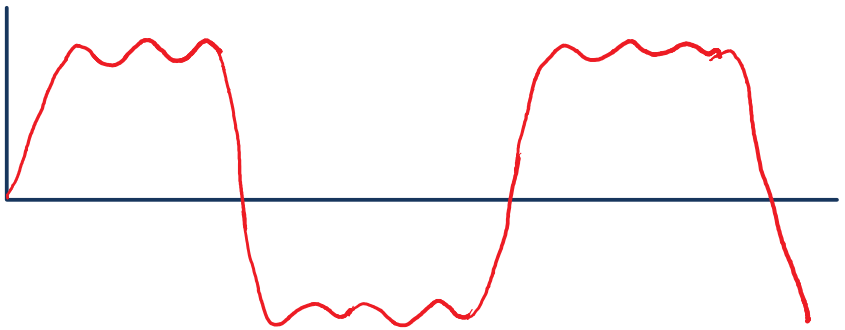
Inc ... Ausleseincrement der Wavetable

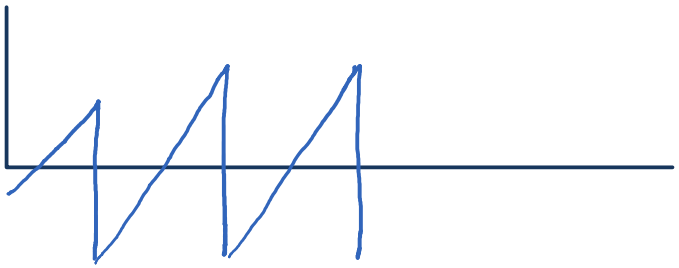
Bandbegrenztes Rechteck Dreieck und Sägezahn Signal

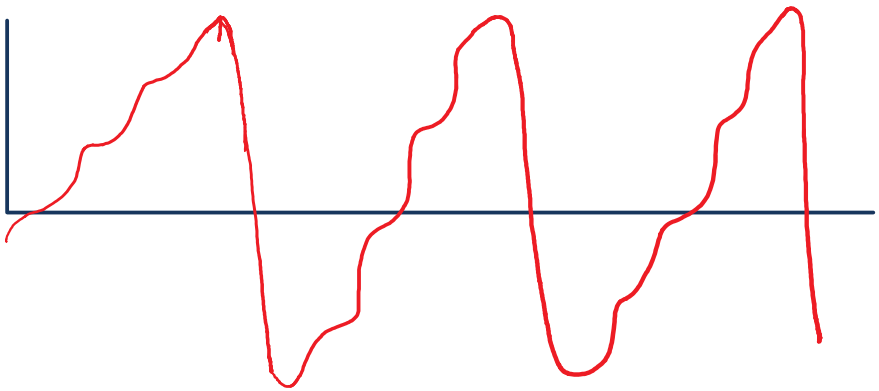
Rechteck oder Sägezahn Schwingungen mit unendlich steilen Flanken
so wie wir sie in der Mbed-Synthesizer Übung erzeut haben
enthalten Frequenzen welche aufgrund der fixen Abtastrate (48kHz)
unserer Audioumgebung zu **hörbarem Aliasing** führen.

Die Wavetables für periodische Schwingungen sollten daher mit **Bandbegrenzung**
erzeugt werden.
Bandbegernzte Wavetables erhält man indem man die Rechteck oder Sägezahn Schwingung
mithilfe einer **Fourierreihe** zusammensetzt.
Siehe dazu auch *Fourier_Formelzettel_V2.doc*



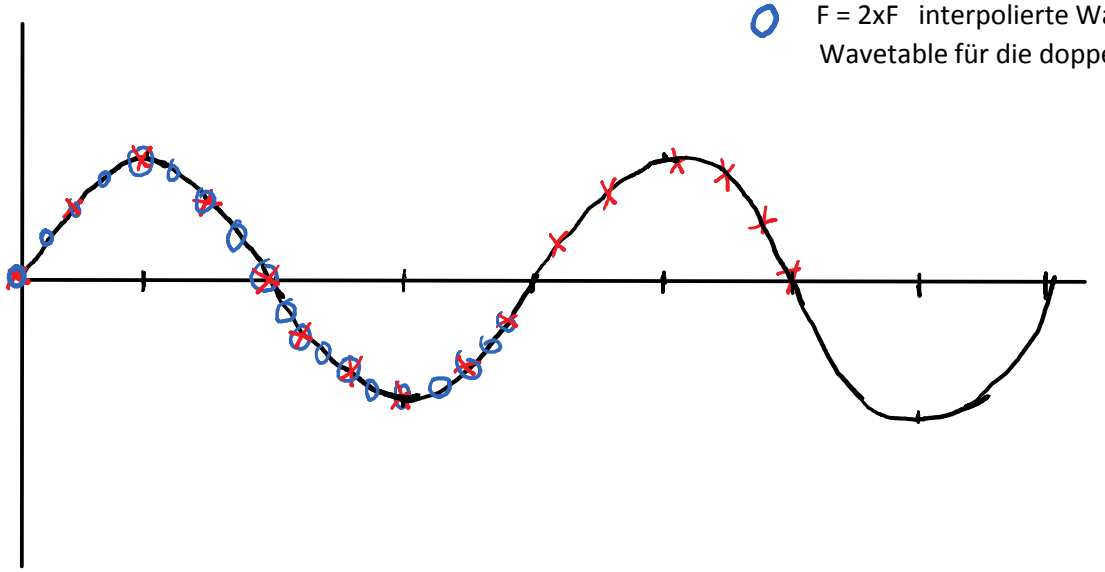






S5

Dienstag, 29. Juli 2014 21:59



Originale Wavetable Punkte

F = 2xF interpolierte WaveTable Punkte
Wavetable für die doppelte Frequenz