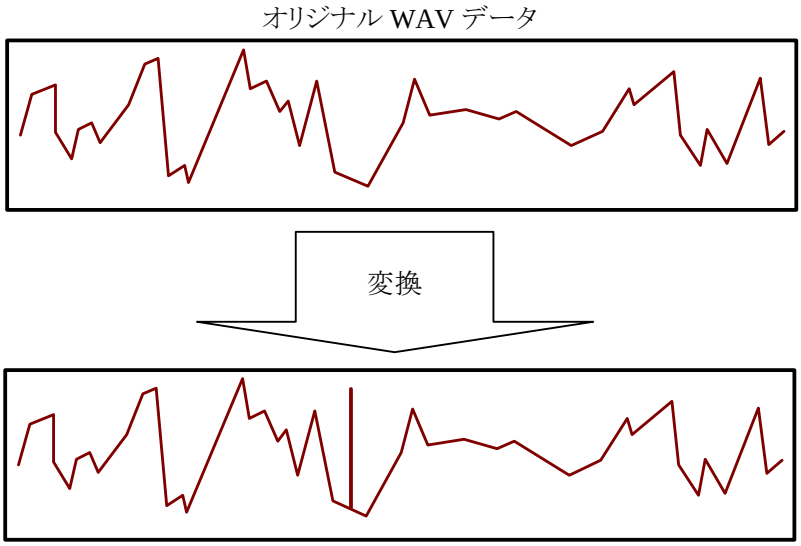
いくつかの音響プログラムを紹介します.

グリッチを乗せる

WAV ファイルのある部分にグリッチノイズを乗せるプログラムです. グリッチのような急峻な変化にどのような反応をするかなどのテスト波形を生成するのに使えるでしょう. 以降にプログラムの機能をイメージ図で示します.



通常四つの引数を要求します. 最初の引数は入力の WAV ファイル名, 二番目の引数は出力用の WAV ファイル名です. 三番目の引数はグリッチが始まる位置を秒で, 次の引数で, グリッチのレベルを -1.0 ~ 1.0 の範囲で与えます. これは波形のピーク値に対するレベルをです. 引数指定方法を以降に示します.

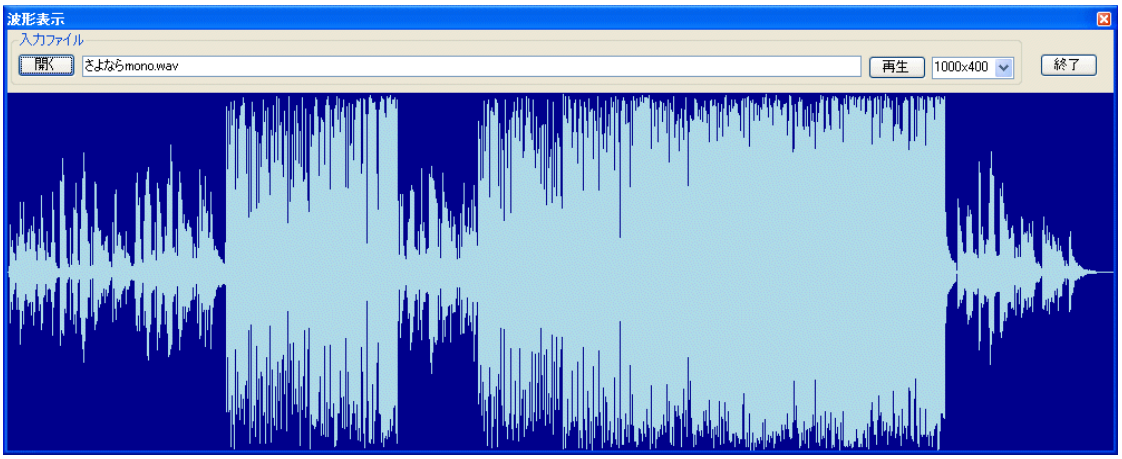
glitch <入力ファイル名1> <出力ファイル名> [<開始時間:秒> <レベル:-1.0～1.0>]

「開始時間:秒」は, グリッチを乗せる開始時間を指定します.

以降に, 実行例を示します. 音源にグリッチを乗せてみます. 16ビット・モノラルの音源に, WAV データの開始位置 30.0 [msec] から-0.9 レベルのグリッチを乗せます.

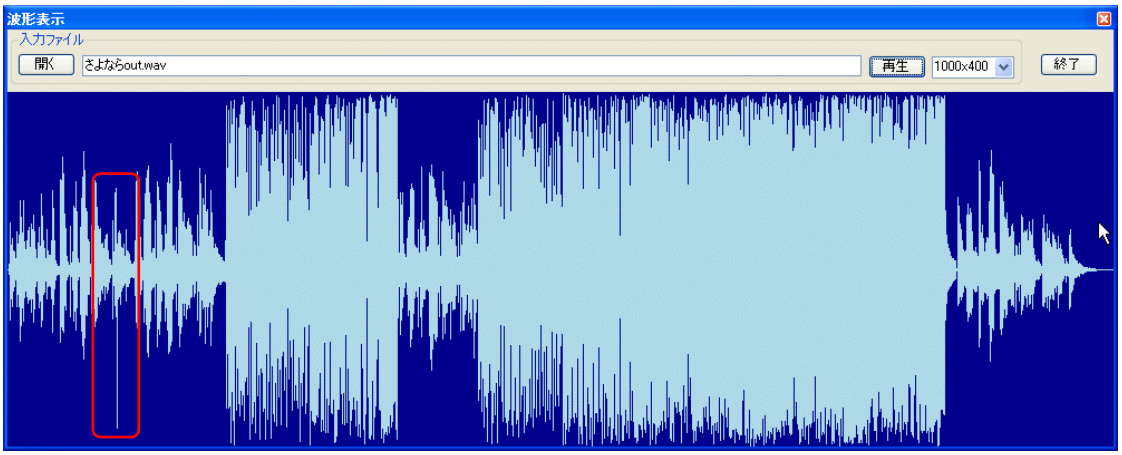
```
C:¥>glitch さよなら mono.wav さよなら out.wav 30 -0.9

ファイル名[さよなら mono.wav]
 "fmt "の長さ: 16 [bytes]
              データ形式: 1 (1 = PCM)
            チャンネル数: 1
      サンプリング周波数: 44100 [Hz]
          バイト数 / 秒: 88200 [bytes/sec]
 バイト数×チャンネル数: 2 [bytes]
     ビット数 / サンプル: 16 [bits/sample]

 "data" の長さ: 26672856 [bytes]

 時間=302.413

[さよなら mono.wav] を [さよなら out.wav] へ変換しました.
```

入力波形:



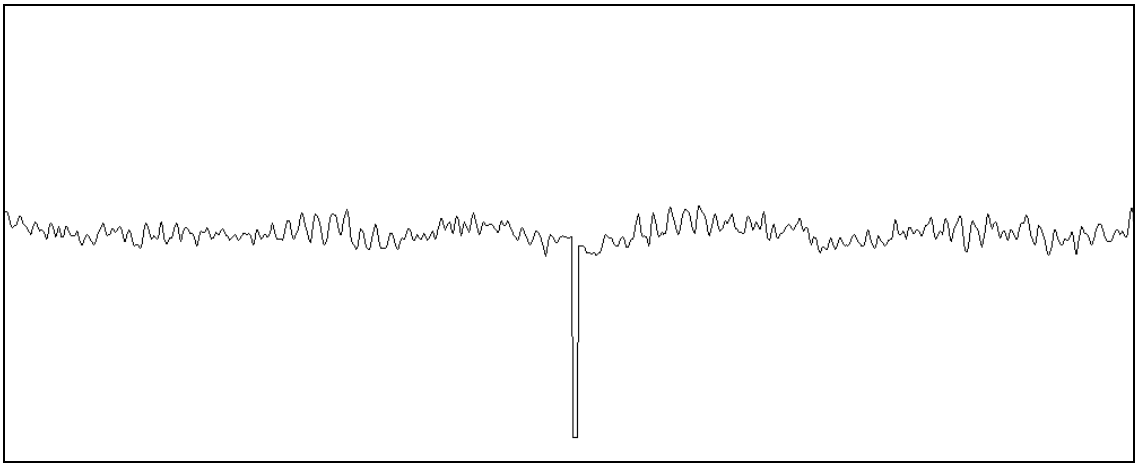
入力 WAV ファイル（さよなら mono.wav）

出力波形：



出力 WAV ファイル（さよなら out.wav）

グリッチが乗っています．ヘッドフォンを付けて，聞いてみると 30 秒近辺でプチッと音が聞こえます．分かり難いので，29.990 秒目から 0.020 秒間を拡大してみます．

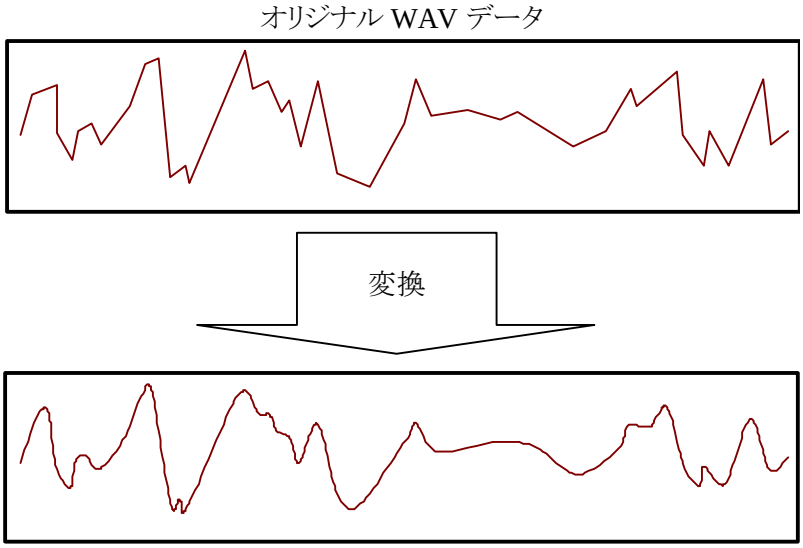


出力 WAV ファイルの一部拡大（さよなら out.wav の一部）

綺麗にグリッチが乗っています．29.990 秒目から 20 [msec] を拡大しているので，グリッチは丁度真ん中に現れます．グリッチレベルとして-0.9 を与えていますので，下の方へ 9 割方伸びています．

移動平均

WAV ファイルに移動平均をかけます．音が滑らかになりますが，籠もった感じも受けます．また，ダイナミックレンジも狭まりますので，音のダイナミック性が失われます．代わりに滑らかな感じになります．



通常二つの引数を要求します．最初の引数は入力の WAV ファイル名で，二番目の引数は出力用の WAV ファイル名です．さらに，移動平均の範囲を秒で指定します．この引数を指定しないと，デフォルトの値が使われます．

soft　<入力ファイル名>　<出力ファイル名>　[<間隔:秒>]

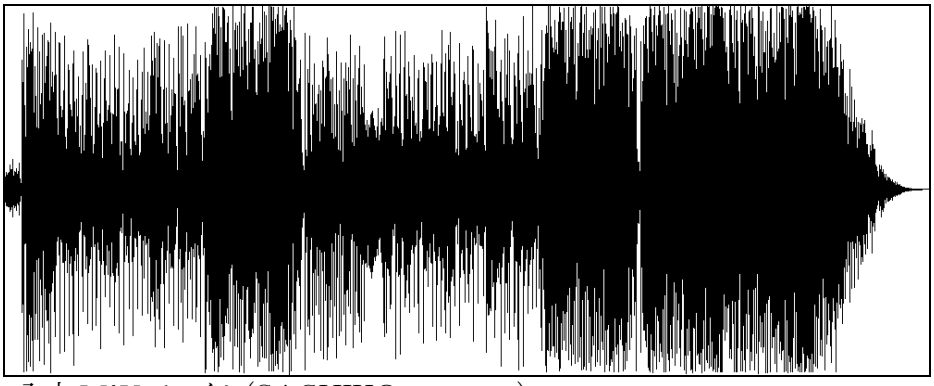
「間隔:秒」は，移動平均をとる範囲です．あまり大きな値を与えると，音が籠もってしまい，ダイナミックレンジも狭くなります．

以降に，実行例を示します．音源にグリッチを乗せてみます．16 ビット・モノラルのある曲に移動平均範囲として，0.0001 秒(0.1 [msec])を指定します．

```
C:¥>soft ＳＡＣＨＩＫＯmono.wav soft01.wav 0.0001

ファイル名[ＳＡＣＨＩＫＯmono.wav]
 "fmt "の長さ: 16 [bytes]
            データ形式: 1 (1 = PCM)
          チャンネル数: 1
     サンプリング周波数: 44100 [Hz]
         バイト数 / 秒: 88200 [bytes/sec]
 バイト数×チャンネル数: 2 [bytes]
    ビット数 / サンプル: 16 [bits/sample]

 "data" の長さ: 23654064 [bytes]

 時間=268.187

ファイル名[ＳＡＣＨＩＫＯmono.wav]
 "fmt "の長さ: 16 [bytes]
            データ形式: 1 (1 = PCM)
          チャンネル数: 1
     サンプリング周波数: 44100 [Hz]
```

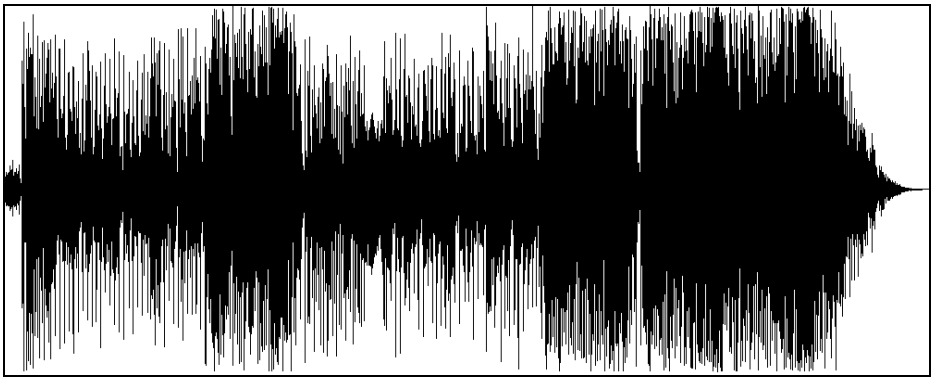
```
          バイト数 / 秒: 88200 [bytes/sec]
 バイト数×チャンネル数: 2 [bytes]
    ビット数 / サンプル: 16 [bits/sample]

 "data" の長さ: 23654064 [bytes]

 時間=268.187
 サンプル数=4

ファイル [ＳＡＣＨＩＫＯmono.wav] を [soft01.wav]へ変換しました.
```

入力波形:



入力 WAV ファイル(SACHIKOmono.wav)

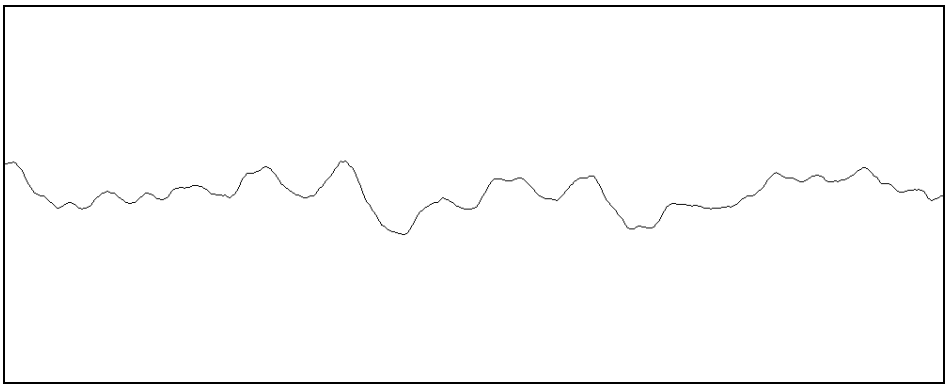
出力波形:



出力 WAV ファイルの一部(soft01.wav)

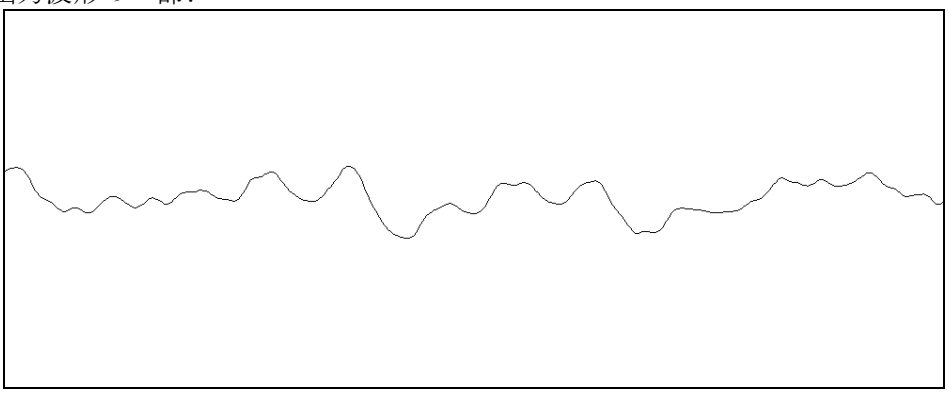
耳で聞くとあきらかに違いが分かりますが, 長い波形を狭い範囲に表示しても違いが分かりません. そこで, 両方の波形の 100 秒目から, 10 [msec]を拡大してみましょう.

入力波形の一部:



入力 WAV ファイル（SACHIKOmono.wav）

出力波形の一部:



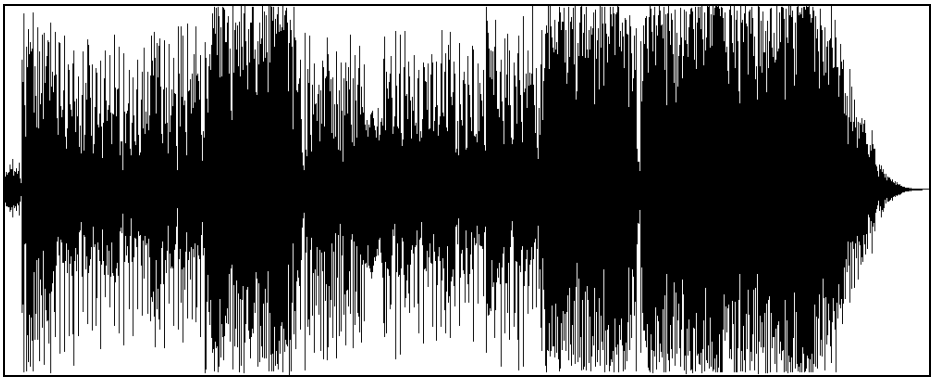
出力 WAV ファイルの一部（soft01.wav）

これでも分かり難いと人もいるでしょうから，移動平均の幅を 10 倍にしてみましょう．同じ音源に移動平均範囲として，0.001 秒(1.0 [msec])を指定します．

```
C:¥>soft ＳＡＣＨＩＫＯmono.wav soft02.wav 0.001

ファイル名[ＳＡＣＨＩＫＯmono.wav]
 "fmt "の長さ: 16 [bytes]
            データ形式: 1 (1 = PCM)
          チャンネル数: 1
     サンプリング周波数: 44100 [Hz]
         バイト数 / 秒: 88200 [bytes/sec]
 バイト数×チャンネル数: 2 [bytes]
    ビット数 / サンプル: 16 [bits/sample]

 "data" の長さ: 23654064 [bytes]

 時間=268.187

ファイル名[ＳＡＣＨＩＫＯmono.wav]
 "fmt "の長さ: 16 [bytes]
            データ形式: 1 (1 = PCM)
          チャンネル数: 1
     サンプリング周波数: 44100 [Hz]
         バイト数 / 秒: 88200 [bytes/sec]
 バイト数×チャンネル数: 2 [bytes]
    ビット数 / サンプル: 16 [bits/sample]

 "data" の長さ: 23654064 [bytes]

 時間=268.187
```

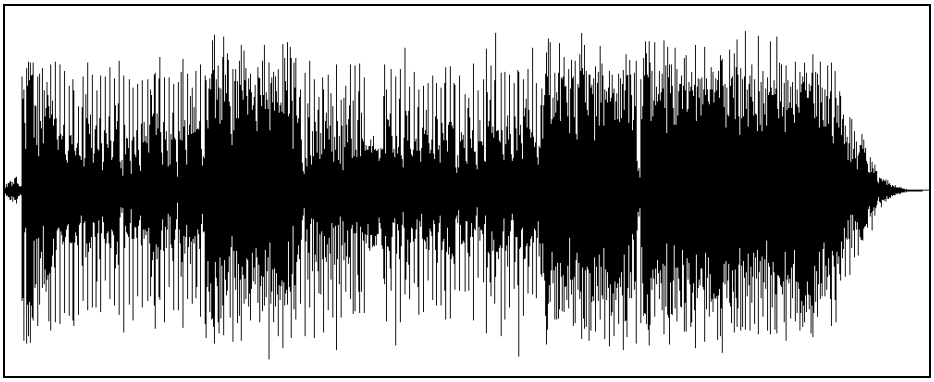
```
サンプル数=44

ファイル［ＳＡＣＨＩＫＯmono.wav］を［soft02.wav]へ変換しました.
```

入力波形:



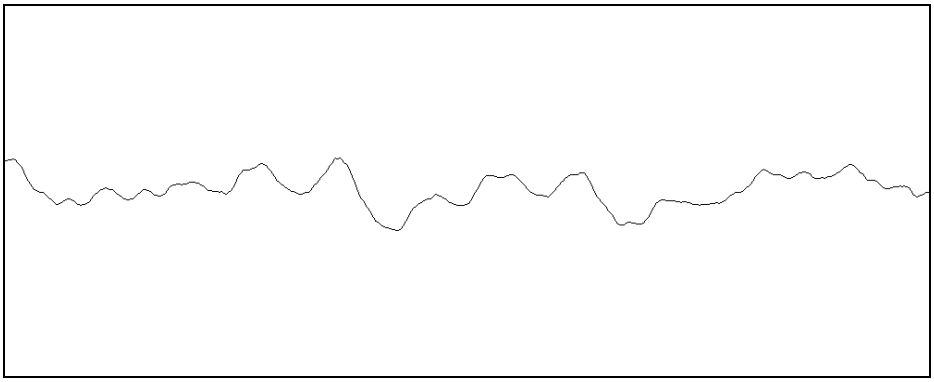
入力 WAV ファイル（SACHIKOmono.wav）

出力波形:



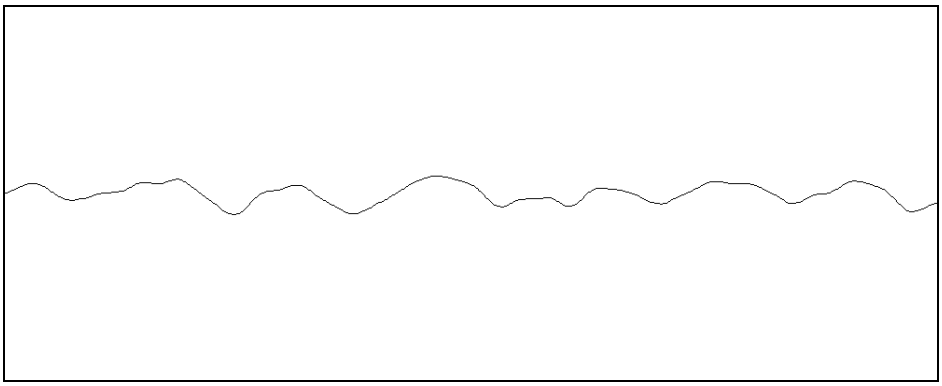
出力 WAV ファイルの一部（soft02.wav）

入力波形の一部:



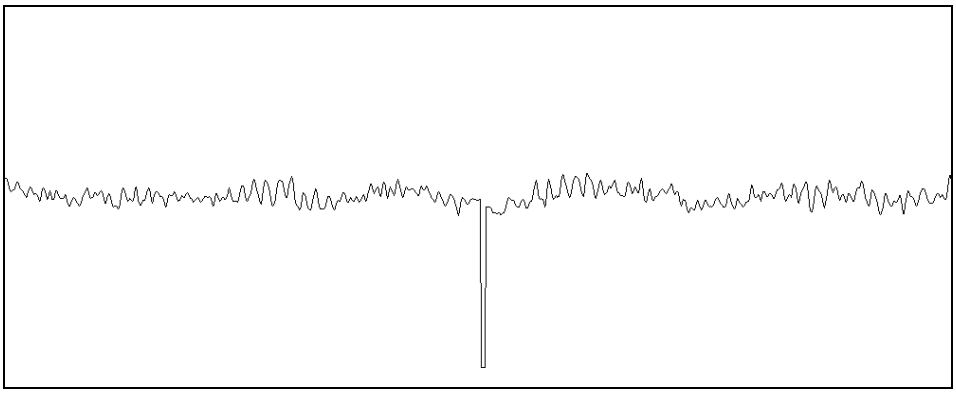
入力 WAV ファイル（SACHIKOmono.wav の一部）

出力波形の一部:



出力 WAV ファイルの一部(soft02.wav の一部)

さらに, 前節で乗せたグリッチを消せるか試してみましょう. 以降に前節のグリッチを示します. まずグリッチを乗せた波形の 29.990 秒目から 0.020 秒間を以降に示します.



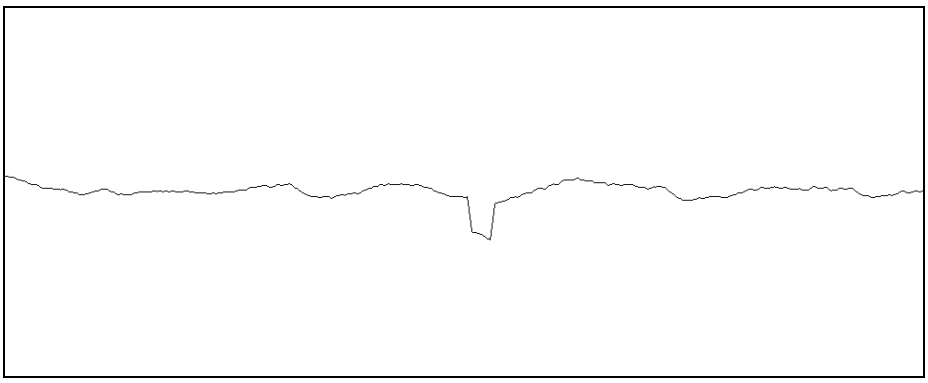
入力 WAV ファイルの一部拡大(さよなら out.wav の一部)

この波形に移動平均の範囲を, 0.0005 秒目(0.5 [msec])指定してみましょう.

```
C:¥>soft さよなら out.wav soft03.wav 0.0005

ファイル名[さよなら out.wav]
 "fmt "の長さ: 16 [bytes]
            データ形式: 1 (1 = PCM)
          チャンネル数: 1
     サンプリング周波数: 44100 [Hz]
         バイト数 / 秒: 88200 [bytes/sec]
 バイト数×チャンネル数: 2 [bytes]
    ビット数 / サンプル: 16 [bits/sample]

 "data" の長さ: 26672856 [bytes]

 時間=302.413

ファイル名[さよなら out.wav]
 "fmt "の長さ: 16 [bytes]
            データ形式: 1 (1 = PCM)
          チャンネル数: 1
     サンプリング周波数: 44100 [Hz]
         バイト数 / 秒: 88200 [bytes/sec]
 バイト数×チャンネル数: 2 [bytes]
    ビット数 / サンプル: 16 [bits/sample]

 "data" の長さ: 26672856 [bytes]
```

```
時間=302.413
サンプル数=22

ファイル［さよなら out.wav］を［soft03.wav］へ変換しました.
```

出力波形:



出力 WAV ファイルの一部（soft03.wav **の一部**）

グリッチは小さくなっていますが, ある程度残っています. ノイズ除去と考えると, あまり効率は良くないようです. 音をマイルドに変更するのに使用するのが良さそうです.

WAV ファイルチェッカー

WAV ファイルの整合性をチェックします． WAV ファイルを操作するプログラムを開発したときに，生成したファイルの矛盾がないかチェックするのに便利です．

本プログラムには，調べたい WAV ファイル名を指定します．引数指定方法を以降に示します．

checkWav　<入力ファイル名>