# Aufgabenblatt 10

*Julian Bertol*

## Aufgabe 2

Die erzeugte Foo.Wav Datei befindet sich im Projekt-Ordner.

Ausgabe hexer:

```
000007f0:  76 7c 89 7d 6c 7e 1c 7f  9a 7f e6 7f ff 7f e6 7f  v|.}l~..........
00000800:  9a 7f 1c 7f 6c 7e 89 7d  76 7c 31 7b bb 79 16 78  ....l~.}v|1{.y.x
00000810:  41 76 3d 74 0c 72 ad 6f  22 6d 6d 6a 8d 67 84 64  Av=t.r.o"mmj.g.d
00000820:  54 61 fe 5d 82 5a e2 56  20 53 3e 4f 3c 4b 1c 47  Ta.].Z.V S>O<K.G
00000830:  e1 42 8b 3e 1c 3a 96 35  fb 30 4d 2c 8e 27 be 22  .B.>.:.5.0M,.'."
00000840:  e1 1d f9 18 06 14 0b 0f  0b 0a 06 05 00 00 fa fa  ................
00000850:  f5 f5 f5 f0 fa eb 07 e7  1f e2 42 dd 72 d8 b3 d3  ..........B.r...
00000860:  05 cf 6a ca e4 c5 75 c1  1f bd e4 b8 c4 b4 c2 b0  ..j...u.........
00000870:  e0 ac 1e a9 7e a5 02 a2  ac 9e 7c 9b 73 98 93 95  ....~.....|.s...
00000880:  de 92 53 90 f4 8d c3 8b  bf 89 ea 87 45 86 cf 84  ..S.........E...
00000890:  8a 83 77 82 94 81 e4 80  66 80 1a 80 01 80 1a 80  ..w.....f.......
000008a0:  66 80 e4 80 94 81 77 82  8a 83 cf 84 45 86 ea 87  f.....w.....E...
000008b0:  bf 89 c3 8b f4 8d 53 90  de 92 93 95 73 98 7c 9b  ......S.....s.|.
000008c0:  ac 9e 02 a2 7e a5 1e a9  e0 ac c2 b0 c4 b4 e4 b8  ....~...........
000008d0:  1f bd 75 c1 e4 c5 6a ca  05 cf b3 d3 72 d8 42 dd  ..u...j.....r.B.
000008e0:  1f e2 07 e7 fa eb f5 f0  f5 f5 fa fa 00 00 06 05  ................
000008f0:  0b 0a 0b 0f 06 14 f9 18  e1 1d be 22 8e 27 4d 2c  ..........."'M,
00000900:  fb 30 96 35 1c 3a 8b 3e  e1 42 1c 47 3c 4b 3e 4f  .0.5.:.>.B.G<K>O
00000910:  21 53 e2 56 82 5a fe 5d  54 61 84 64 8d 67 6d 6a  !S.V.Z.]Ta.d.gmj
00000920:  22 6d ad 6f 0c 72 3d 74  41 76 16 78 bb 79 31 7b  "m.o.r=tAv.x.y1{
00000930:  76 7c 89 7d 6c 7e 1c 7f  9a 7f e6 7f ff 7f e6 7f  v|.}l~..........
00000940:  9a 7f 1c 7f 6c 7e 89 7d  76 7c 31 7b bb 79 16 78  ....l~.}v|1{.y.x
00000950:  41 76 3d 74 0c 72 ad 6f  22 6d 6d 6a 8d 67 84 64  Av=t.r.o"mmj.g.d
00000960:  54 61 fe 5d 82 5a e2 56  20 53 3e 4f 3c 4b 1c 47  Ta.].Z.V S>O<K.G
00000970:  e1 42 8b 3e 1c 3a 96 35  fb 30 4d 2c 8e 27 be 22  .B.>.:.5.0M,.'."
00000980:  e1 1d f9 18 06 14 0b 0f  0b 0a 06 05 00 00 fa fa  ................
00000990:  f5 f5 f5 f0 fa eb 08 e7  1f e2 42 dd 72 d8 b3 d3  ..........B.r...
000009a0:  05 cf 6a ca e4 c5 75 c1  1f bd e4 b8 c4 b4 c2 b0  ..j...u.........
000009b0:  df ac 1e a9 7e a5 02 a2  ac 9e 7b 9b 73 98 93 95  ....~.....{.s...
"foo.wav" 0x7532c (480044) bytes
```

Verständnisfrage:

Die Daten in der WAV-Datei werden sowohl auf dem Host (little-endian Architektur) als auch auf dem Target (big-endian Architektur) korrekt interpretiert, weil das WAV-Format standardmäßig little-endian ist. Das bedeutet, dass die Byte-Reihenfolge bei der Speicherung der Daten im little-endian Format erfolgt. Die meisten Audio-Bibliotheken und -Programme sind so implementiert, dass sie dies berücksichtigen und die Daten korrekt interpretieren können, unabhängig von der Endianess der zugrunde liegenden Hardware. Beim Erzeugen der PCM-Daten mit dem Format PCM_16, also 16-Bit Integer, werden die Daten explizit in der little-endian Reihenfolge gespeichert, was eine korrekte Interpretation auf beiden Architekturen ermöglicht.

## Aufgabe 3:

```
// prepare a 5 seconds buffer and write it
const int size = sampleRate*5;
float sample[size], factor=1.0;
for (int i=0; i<size; i++) {
    float ex_factor = sin(M_PI * i / sampleRate);
    sample[i]=sin(float(i)/size*M_PI*3000) * factor * ex_factor;
}
outfile.write(&sample[0], size);
```

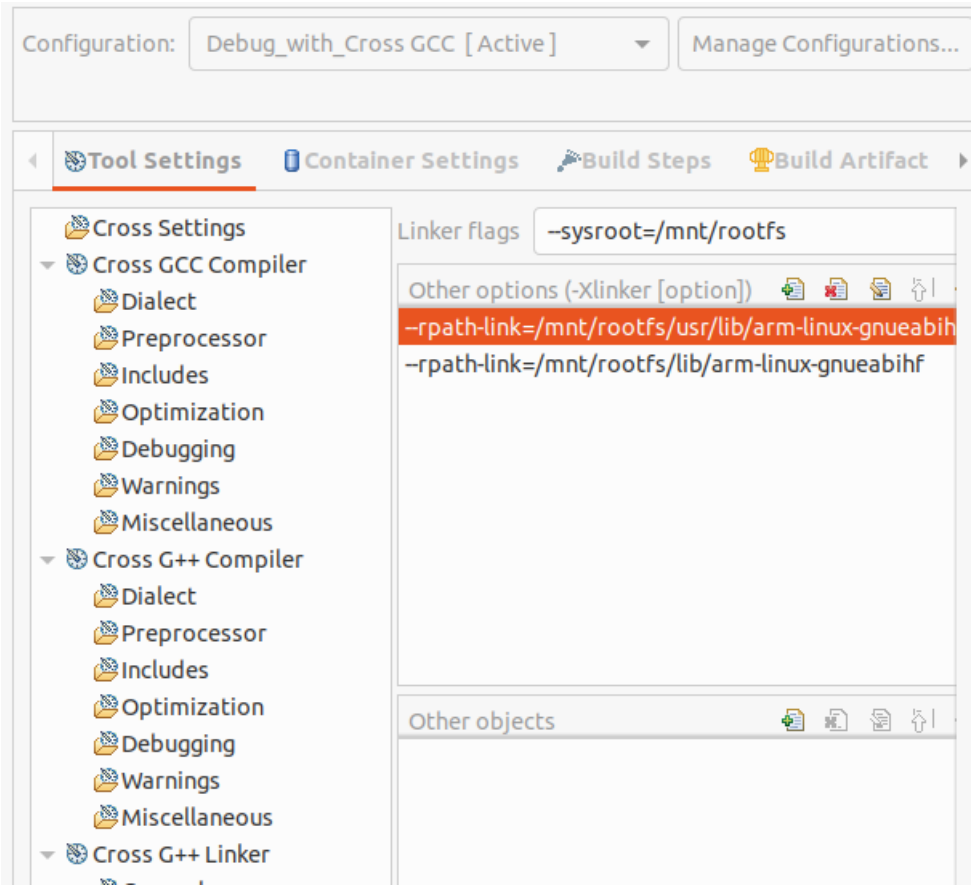
## Aufgabe 4:

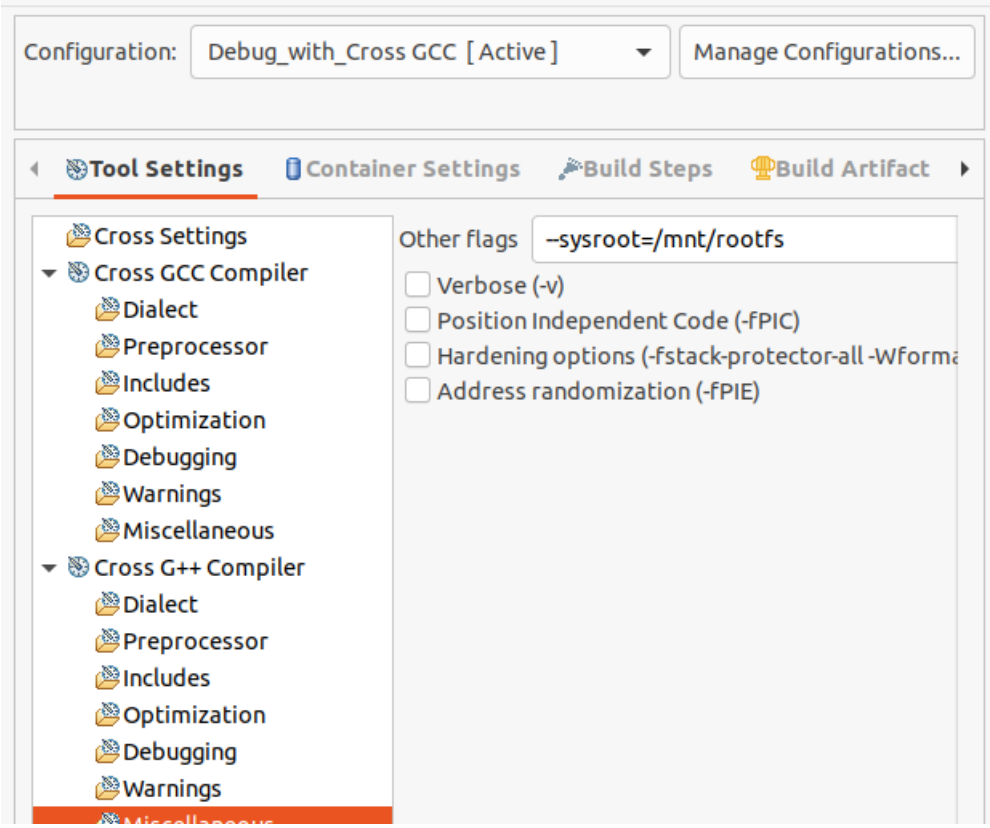
```
const int size = sampleRate * 5 * channels; // Buffergröße für Stereo
float sample[size], factor = 1.0;
for (int i = 0; i < size; i += 2) {
    // Kanal 1
    float volume1 = (sin(2 * M_PI * (i / 2) / sampleRate * 0.1) + 1) / 2; // Lautstärke modulieren
    sample[i] = sin(float(i / 2) / sampleRate * M_PI * 3000) * factor * volume1;

    // Kanal 2 mit leicht unterschiedlicher Frequenz
    float volume2 = (sin(2 * M_PI * (i / 2) / sampleRate * 0.1 + M_PI) + 1) / 2; // Lautstärke modulieren, phasenversc
    sample[i + 1] = sin(float(i / 2) / sampleRate * M_PI * 3100) * factor * volume2;
}
outfile.write(&sample[0], size);

cout << "done!" << endl;
return 0;
```

## Aufgabe 5






## Aufgabe 6

Host:

```
hfu@host085:~$ gst-inspect-1.0 | grep mp3
mpg123:  mpg123audiodec: mpg123 mp3 decoder
lame:  lamemp3enc: L.A.M.E. mp3 encoder
typefindfunctions: application/x-id3v2: mp3, mp2, mp1, mpga, ogg, flac, tta
typefindfunctions: application/x-id3v1: mp3, mp2, mp1, mpga, ogg, flac, tta
typefindfunctions: application/x-apetag: mp3, ape, mpc, wv
typefindfunctions: audio/mpeg: mp3, mp2, mp1, mpga
hfu@host085:~$
```

Target:

```
pi@target085:~ $ gst-inspect-1.0 | grep mp3
typefindfunctions: application/x-id3v2: mp3, mp2, mp1, mpga, ogg, flac, tta
typefindfunctions: application/x-id3v1: mp3, mp2, mp1, mpga, ogg, flac, tta
typefindfunctions: application/x-apetag: mp3, ape, mpc, wv
typefindfunctions: audio/mpeg: mp3, mp2, mp1, mpga
mpg123:  mpg123audiodec: mpg123 mp3 decoder
libav:  avdec_mp3float: libav MP3 (MPEG audio layer 3) decoder
libav:  avdec_mp3: libav MP3 (MPEG audio layer 3) decoder
libav:  avdec_mp3adufloat: libav ADU (Application Data Unit) MP3 (MPEG audio layer 3) decoder
libav:  avdec_mp3adu: libav ADU (Application Data Unit) MP3 (MPEG audio layer 3) decoder
libav:  avdec_mp3on4float: libav MP3onMP4 decoder
libav:  avdec_mp3on4: libav MP3onMP4 decoder
libav:  avmux_mp3: libav MP3 (MPEG audio layer 3) formatter (not recommended, use id3v2mux instead)
lame:  lamemp3enc: L.A.M.E. mp3 encoder
pi@target085:~ $
```

Test:

Target:

```
pi@target085:~ $ gst-launch-1.0 audiotestsrc ! audioconvert ! audioresample ! alsasink
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Redistribute latency...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstAudioSinkClock
```

Host:

```
hfu@host085:~$ gst-launch-1.0 audiotestsrc ! audioconvert ! audioresample ! alsasink
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Redistribute latency...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstAudioSinkClock
```

Auf beiden Geräten wird ein Ton abgespielt

**Aufgabe 7**

Host:

```
hfu@host085:~/A15_Audio/Test$ gst-launch-1.0 filesrc location=cp.ogg ! oggdemux ! vorbisdec ! audioconvert ! audioresamp
le ! alsasink
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Redistribute latency...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstAudioSinkClock
```

Target:

```
pi@target085:~/A15_Audio/A15_Audio/Test $ gst-launch-1.0 filesrc location=cp.ogg ! oggdemux ! vorbisdec ! audioconvert !
 audioresample ! alsasink
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Redistribute latency...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstAudioSinkClock
```

Auch hier läuft auf beiden Geräten Musik

**Aufgabe 8**

gst-launch-1.0 filesrc location=hirsch.mp3 ! decodebin ! audioconvert ! tee name=t ! queue ! vorbisenc ! oggmux ! filesink location=cp.ogg t. ! queue ! audioresample ! alsasink

**Aufgabe 9**

Befehl auf dem Host

gst-launch-1.0 -v alsasrc ! audioconvert ! audioresample ! 'audio/x-raw,rate=16000,width=16,channels=1' ! speexenc ! rtpspeexpay ! udpsink host=<target-IP> port=6666

Befehl auf dem target:

gst-launch-1.0 udpsrc port=6666 caps="application/x-rtp,media=(string)audio,clock-rate=(int)16000,encoding-name=(string)SPEEX,payload=(int)110" ! rtpjitterbuffer ! rtpspeexdepay ! speexdec ! audioconvert ! audioresample ! alsasink

**Aufgabe 10**

Neuer Befehl:
gst-launch-1.0 udpsrc port=6666 caps="application/x-rtp,media=(string)audio,clock-rate=(int)16000,encoding-name=(string)SPEEX,payload=(int)110" ! rtpjitterbuffer ! rtpspeexdepay ! speexdec ! tee name=t ! queue ! audioconvert ! audioresample ! alsasink t. ! queue ! audioconvert ! audioresample ! wavenc ! filesink location=recorded_audio.wav