Audio

Florian Zeitz

Chaos Computer Club Cologne e.V.
http://koeln.ccc.de

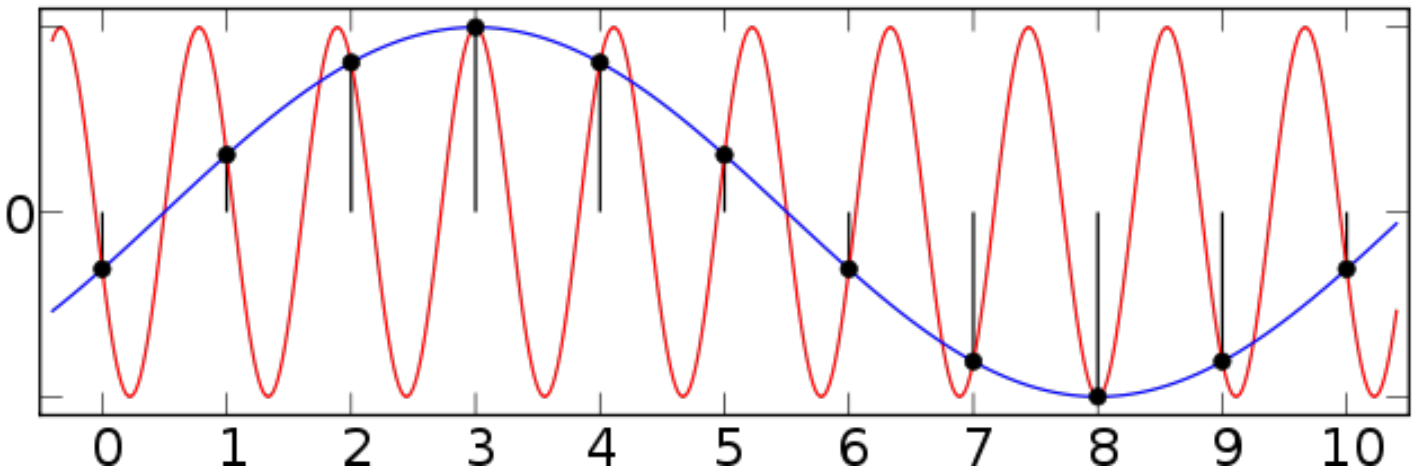
2012-11-19

A/D Wandlung

- Audiosignale werden periodisch abgetastet $\Rightarrow$ Abtastwerte (Sample)
- Signalpegel wird diskretisiert
- Bei uns: 16bit Sample, Abtastrate konfiguierbar

Nyquist-Shannon-Sampling-Theorem

$$f_s > 2 \cdot f_g$$



D/A Wandlung

- Wert der Sample wird bis das nächste ansteht gehalten
- Tiefpassfilterung

Konzept

- ähnlich VGA
- 2 Buffer werden abwechselnd bei Bedarf gefüllt
- ausgabe erfolgt im Hintergrund über DMA
- Buffer beinhalten 128 interleaved Stereo Sample
- Demo-Code: `firmware/guitar/`

API

```
1 void InitializeAudio(uint32_t freq);
2
3 typedef void AudioCallbackFunction(void *context, int16_t buffer[256]);
4 void PlayAudioWithCallback(AudioCallbackFunction *callback, void *context);
5 void StopAudio();
```

API: Beispiel

```
1  #include <game/Audio.h>
2  void AudioCallback(void *context, int16_t ↩
       buffer[256])
3  {
4    static int v = 1;
5    for (int i = 0; i < 128; i++) {
6      buffer[2*i] = buffer[2*i+1] = v;
7      v *= -1;
8    }
9  }
10
11 void Init(struct Gamestate* state)
12 {
13   InitializeAudio(Gaming_AudioFreq_11k);
14   PlayAudioWithCallback(AudioCallback, NULL);
15 }
```

Synth

- Angelehnt an klassische Synthesizer
- Drei Wellenformen: Rechteck, Sägezahn, Dreieck
- Demo-Code: `firmware/synthtest/`

Synth API

```c
typedef enum {
  SynthRect,
  SynthSaw,
  SynthTri
} SynthWave;

typedef struct {
  uint16_t freq;
  uint16_t duration;
  uint8_t volume;
} SynthNote;
```

```c
typedef struct {
  SynthWave instrument;
  uint32_t length;
  SynthNote *notes;
  uint32_t note;
} SynthChannel;

typedef struct {
  uint32_t samplingFrequency;
  uint32_t channelNum;
  SynthChannel *channels;
  uint32_t pos;
} SynthSong;
```

```c
int16_t SynthGetSample(SynthSong *song);
```

Synth API Beispiel

```
1  SynthSong song = { Gaming_AudioFreq_22k, 1,
2    &(SynthChannel){ SynthRect, 2,
3      &(SynthNote[]){
4        {Note_g1, 250, 0xff},
5        {Note_Pause, 250, 0x00},
6        {Note_d1, 250, 0xff}
7        {Note_Pause, 250, 0x00},
8      }
9    }
10 };
11 void AudioCallback(void *context, int16_t ↩
       buffer[256]) {
12   for (int i = 0; i < 128; i++)
13     buffer[2*i] = buffer[2*i+1] = ↩
         SynthGetSample(song);
14 }
```

Schwächen

- Neu, schlecht getestestet, unvollständig
- Noch kein support für mehr als einen Channel
- Noch nicht Sampling Frequenz Invariant
- Lautstärke wird noch ignoriert
- Filter wären cool
- Es gibt alternativ BitBin: vollständiger, aber strange™