Helix MP3-Dekoder

Digitale Audio-Übertragung

Um eine gute Qualität bei einer digitalen Audio-Übertragung zu erreichen, muss zum einen eine ausreichend hohe Abtastfrequenz verwendet werden und zum anderen muss die Quantisierung genügend klein sein, dass heißt zur Darstellung der Abtastwerte muss eine ausreichend hohe Anzahl von Bits verwendet werden. CD-Qualität erreicht man, wenn man 44100 mal pro Sekunde abtastet und für die Abtastwerte 16 Bit verwendet. Bei einer Stereoübertragung (2 Kanäle) ergibt sich daraus eine Bitrate von

```
44,1 kHz * 16 Bit * 2 Kanäle = 1,4 Mbit/sec
```

MP3 kodierte Daten werden verlustbehaftet komprimiert. Typischerweise resultiert dabei eine konstante Bitrate von 128 kBit/sec, das heißt, es wird eine Komprimierung von 1400 : 128 , also 11 : 1 erreicht. Der MP3-Standard lässt aber auch andere Kombinationen von Abtastraten und Komprimierungsfaktoren zu.

MP3-Dateien

Eine MP3-Datei ist in einzelne MP3-Rahmen aufgeteilt. Jeder Rahmen besteht aus einem Kopf und einem Datenblock. Das Dekodieren kann nur am Anfang eines Rahmens beginnen. Der Kopf enthält ein Synchronisationswort, welches zum Aufsuchen eines gültigen Rahmens benutzt werden kann. Die restlichen Informationen im Kopf kennzeichnen die Eigenschaften der MP3-Daten, wie z.B. Bitrate, Samplefrequenz, Anzahl der Kanäle, Art des Kompressionsverfahrens usw..

Helix MP3-Dekoder

Der Helix MP3-Dekoder ist Teil eines großen Open-Source Projektes, welches von der Firma RealNetworks [1] initiiert worden ist. Ziel des Projektes ist die Erstellung einer Software, die Audios und Videos in verschiedenen Formaten über das Netzwerk abspielen kann. Die Software soll auf unterschiedlichen Betriebssystemen und Prozessoren lauffähig sein. Insbesondere soll die Software auch auf Handys funktionieren. Zur Steigerung der Performance wurden einige Module in Assembler geschrieben, unter anderem auch für den ARM-Prozessor.

Für die Praktikumsaufgabe wurde die Bibliothek bereits übersetzt. Dabei wurde zur Steigerung der Leistungsfähigkeit eine hohe Optimierungsstufe eingestellt. Das Dekodieren der MP3-Daten geschieht mit Hilfe weniger Funktionsaufrufe:

MP3InitDecoder

```
HMP3Decoder MP3InitDecoder(void);
```

Initialisiert die Bibliothek. Reserviert für die internen Puffer Speicherplatz. Gibt einen Handle zurück, der in den nachfolgenden Aufrufen verwendet werden muss. Falls der Aufruf fehl schlägt, z.B. weil nicht genügend Speicher zur Verfügung steht, wird der Wert NULL zurückgegeben.

MP3FreeDecoder

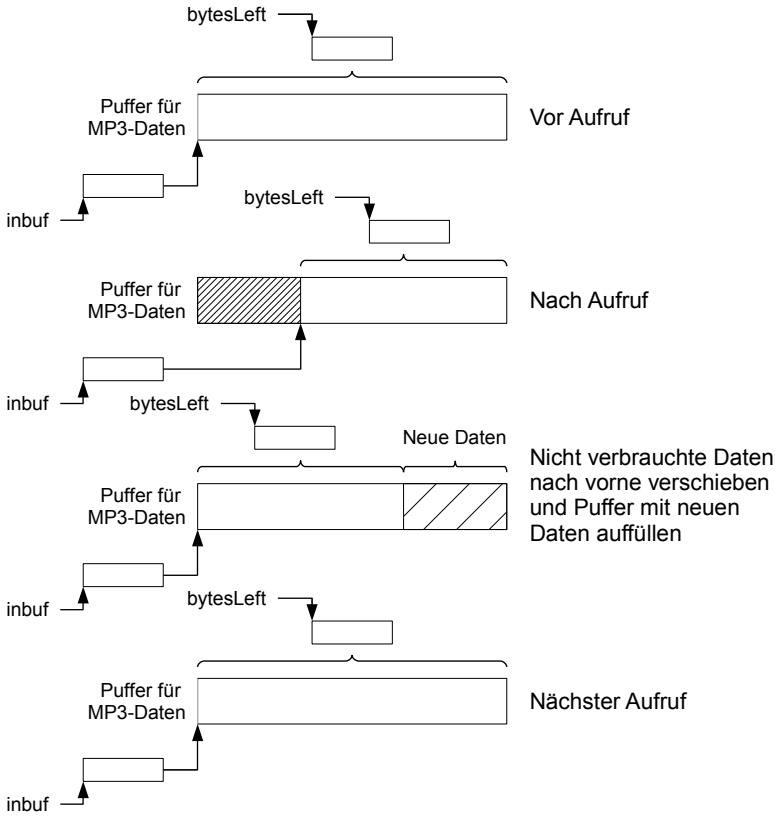
```
void MP3FreeDecoder(HMP3Decoder hMP3Decoder);
```

Gibt die reservierten Ressourcen wieder frei.

MP3Decode

```
int MP3Decode(HMP3Decoder hMP3Decoder, unsigned char **inbuf,
                        int *bytesLeft, short *outbuf, int useSize);
```

Dekodiert einen Rahmen der MP3-Daten. Der Anfang des Rahmens wird mittels eines Zeigers auf eine Zeigervariable übergeben (Parameter *inbuf*). Der Aufruf sorgt dafür, dass nach erfolgreichem Dekodieren die Zeigervariable auf den Anfang des nächsten Rahmens gesetzt gesetzt wird. Auch die Länge der insgesamt zur Verfügung stehenden Daten muss mittels einer Variablen übergeben werden, *bytesLeft* enthält die Adresse dieser Variablen. Die Variable wird ebenfalls automatisch aktualisiert und enthält nach dem Aufruf die Angabe, wie viele Daten nicht verwendet worden sind.
Der Puffer muss so ausgelegt werden, dass ein MP3-Rahmen mit maximaler Größe vollständig hineinpasst. Diese Maximalgröße in Bytes ist durch die Konstante *MAINBUF_SIZE* definiert.
Ein möglicher Ablauf des Aufrufs ist in der nebenstehenden Zeichnung gezeigt.



Der Aufruf legt die dekodierten Daten in einem Puffer ab, dessen Anfangsadresse mit dem Parameter *outbuf* übergeben werden muss. Die maximale Anzahl 16-Bit Werte, die in den Ausgabepuffer passen müssen, ergibt sich aus *MAX_NCHAN* * *MAX_NGRAN* * *MAX_NSAMP* (alles in mp3dec.h definiert).
Das Flag *useSize* sollte immer auf 0 gesetzt werden.

MP3GetLastFrameInfo

```
void MP3GetLastFrameInfo(HMP3Decoder hMP3Decoder,
                                MP3FrameInfo *mp3FrameInfo);
```

Nach erfolgreichem Aufruf von *MP3Decode* können mit diesem Aufruf die Eigenschaften der MP3-Daten abgefragt werden. *MP3FrameInfo* enthält folgende Informationen:

bitrate	Rate der kodierten Daten in Bit/sec.
nChans	Anzahl Kanäle.
samprate	Ausgabegeschwindigkeit in Samples/sec.
bitsPerSample	Anzahl Bit pro Sample.
outputSamps	Anzahl Samples im Ausgabepuffer.
layer	MPEG Layer.

version MPEG Version (0: MPEG1, 1: MPEG2, 2: MPEG2.5).

MP3FindSyncWord

```
int MP3FindSyncWord(unsigned char *buf, int nBytes);
```

Mit diesem Aufruf wird das nächste Synchronisationsword im MP3-Stream gesucht. Insbesondere dient dieser Aufruf dazu, die sogenannten ID3-Tags[2] zu überspringen. Diese stellen eine Art Inhaltsangabe dar. Der Rückgabewert des Aufrufs gibt an, wie viele Bytes bis zum Erreichen des nächsten Synchronisationsword übersprungen werden müssen.

Referenzen

1: http://www.realnetworks.com/
2: http://www.id3.org/Introduction