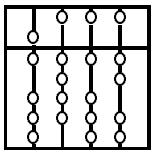
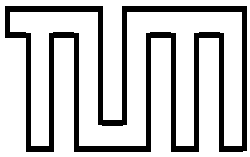


Technische Universität München
Institut für Informatik



Fortgeschrittenenpraktikum im SS 1994
bei
Prof. J. Schlichter

MPEG/SYSTEMS Multiplexer

Vermischen von MPEG-kodierten
Bild- und Toninformationen nach dem
ISO/IEC 11172-1 Verfahren

Christoph Moar
Betreuer: Tristan Daude

Abstract

In den letzten Jahren ist Bewegung in die Welt der digitalen Bild- und Tonübertragung gekommen. Neue Verfahren zur Komprimierung und Übertragung der Daten wurden geschaffen und ständig weiterentwickelt. Besonders stark haben sich seit 1988 die Verfahren der ISO/IEC Expert Groups profiliert. JPEG, MotionJPEG und zuletzt MPEG sind auf breite Akzeptanz in der Soft- und Hardwareindustrie gestoßen. Die psychooptischen und -akustischen Modelle, die von obigen Verfahren zur Kompression der digitalisierten Daten angesetzt werden, schlagen zur Zeit mit massivem Rechenbedarf zu Buche. Auf klassischen Workstations läßt sich mit derzeitigen Algorithmen, die nur auf Softwarelösungen basieren, allerhöchstens die Dekompression in Realzeit vornehmen. Zur gemeinsamen Dekompression von TV-Qualität Video- und CD-Qualität Toninformation kann man zu billigen Hardwarelösungen greifen, die für die verschiedensten Hardwareplattformen, auch für Industrie-Standard-PC's, zur Verfügung stehen. Hardware-erweiterungen für die (Echtzeit-) Kodierung solcher MPEG Filme sind noch relativ neu auf dem Markt und haben sich auf Preisklassen weit über denen einer mittleren Workstation angesiedelt.

Auf dem universitären Forschungsbereich hat sich bereits viel bewegt, es stehen getrennt Lösungen zur offline Kodierung von Video- bzw. Audiodaten zu MPEG/Video bzw. MPEG/Audio Strömen zur Verfügung. Was bisher fehlte, war eine Möglichkeit, solche Basisströme zu einem gemultiplexten MPEG/Systems Strom zu vermischen, wie sie von einfachen Hardwarelösungen oder Set-Top-Boxes dekodiert werden können. Die Eckdaten zu einem solchen Projekt kamen von der ISO/IEC Norm 11172-1, die sich ausschließlich mit diesem Verfahren auseinandersetzt.

In dieser Schrift werden die grundlegenden Verfahren erläutert, die getrennt hinter der Video- und Audiokodierung stehen; der Kern der Arbeit befaßt sich weiters mit den Strategien, die zur Vermischung der zwei - eigentlich grundlegend verschiedenen - Strömen in einer höheren Schicht befolgt werden müssen. Als Ergebnis kann das Programm mplex präsentiert werden, das als reine Softwarelösung zum Generieren von MPEG/Systems AV-Strömen verwendet werden kann.

Inhaltsverzeichnis

Abstract 2
Inhaltsverzeichnis 3
1. Motivation 5
2. Grundlagen 5
2.1. Der MPEG 1 International Standard 5
2.2. ISO/IEC 11172-2 MPEG 1/Video Grundlagen 6
Kompressionsverfahren 6
Videostrom 7
2.3. ISO/IEC 11172-3 MPEG 1/Audio Grundlagen 9
Kompressionsverfahren 9
Audiostrom 13
2.4. ISO/IEC 11172-1 MPEG 1/Systems Grundlagen 14
Multiplex Grundlagen 14
Systemstrom 15
2.5. Ausblick: Der MPEG 2 Draft International Standard 18
MPEG 2 Grundlagen 18
MPEG 2 Extensions 19
3. Multiplex Implementierung 20
3.1. Die Sourcen 20
3.2. Programmaufbau 21
3.3. Programmablauf 21
Titelschirm, Parameterüberprüfung und Input Stream Kontrolle 21
Informationen über MPEG-Grundstreams sammeln 22
MPEG/Systems Parameter eingeben 23
Die Multiplex-Hauptschleife, Vermischungsstrategie 27
3.4. Datenstrukturen 28
struct timecode_struc 28
struct vaunit_struc 28
struct aaunit_struc 29
struct video_struc 29
struct audio_struc 30
struct sector_struc 30
struct pack_struc 30
struct sys_header_struc 30
struct buffer_queue 31
struct buffer_struc 31

4. Eigenschaften von MPEG/Systems 32
4.1. Problemfälle 32
Positionieren im gemultiplexten Datenstrom 32
Bufferverhältnisse beim System Target Decoder 33
Standbildfunktionalität 34
Fast Playback / Rewind 34
4.2. Rückblick 35
Anhang A, kommentierter Videostrom 36
Anhang B, kommentierter Audiostrom 38
Anhang C, kommentierter Systemstrom 39
Literaturverzeichnis 40
Index 41

1. Motivation

Im Rahmen eines Werkstudentenprojektes bei Siemens Zentrale Forschung und Entwicklung ST SN 11 sollte eine Möglichkeit erarbeitet werden, beliebiges Video- und Audiomaterial zu einem MPEG System Strom zu kodieren. Es sei festgestellt, daß wir bereits über Mittel verfügen, um Videodaten aufzunehmen und daraus einen MPEG Video Strom herzustellen.[1] Weiter wurden Public-Domain Programme kompiliert -insbesondere der Berkeley-Coder-[2] die uns die Möglichkeit geben, auch Audio Daten zu einem MPEG Audio Strom zu kodieren. Nach Einarbeitung in den MPEG Standards

- **ISO/IEC 11172-1** *(Vermischen von Audio-/Videoströmen zu System Layer)*
- **ISO/IEC 11172-2** *(Kodieren von Video Daten zu Video Strom)*
- **ISO/IEC 11172-3** *(Kodieren von Audio Daten zu Audio Strom)*

konnte die mit dem Vermischen der beiden Ströme verbundene Problematik erst erkannt werden und mit einem Lösungsansatz begonnen werden. Es wurde auch festgestellt, daß es für dieses Problem noch keine Beispielimplementierung im Public-Domain gab.

2. Grundlagen

2.1. Der MPEG 1 International Standard

Im Mai 1988 entstanden im Rahmen einer neuen Organisation der ISO/IEC JTC1/SC2/WG8 vier Untergruppen:

- **JPEG** (*Joint Photographic Experts Group* zur Bearbeitung der Kodierungsalgorithmen von Einzelbildern)
- **JBIG** (*Joint Bi-Level Expert Group* zur Bearbeitung progressiv aufgebauter "bi-level"-Kodierungsalgorithmen)
- **CGEG** (*Computer Graphics Expert Group* zur Bearbeitung von Kodierungsprinzipien)
- **MPEG** (*Moving Picture Expert Group* zur Bearbeitung von Algorithmen zur Audio- und Bewegtbild-Kodierung)

MPEG wird heute in der ISO/IEC JTC1/SC29/WG11 entwickelt bzw. definiert. Zusammen mit Bewegtbildern wird hier auch die Audiokodierung bearbeitet. Unter Berücksichtigung des heutigen Entwicklungsstandes digitaler Massenspeicher aus der CD-Technologie wird hier eine Datenrate des komprimierten Stroms von ca. 1,4 Mbit/s angestrebt.[3] Hierin sind bereits zwei Audiokanäle mit einer Datenrate wahlweise zwischen 32 und 448 Kbits/s (in Schritten zu je 16 Kbit/s) enthalten. Diese Komprimierung ermöglicht eine VHS-Qualität für Video und annähernd HIFI-Qualität für Audio.

---

1 Die genaue Verfahrensweise ist in dem Bericht von Thomas Hahn, Werkstudent bei ZFE ST SN 11 dargestellt.

2 U.a. auf dem FTP-Server toe.cd.berkeley.edu (128.32.149.117) in pub/multimedia/mpeg/mpeg* zu finden.

3 MPEG-1 schreibt eine konstante Datenrate von maximal 1.856.000 Bit/s vor.

MPEG ist sowohl für symmetrische als auch für asymmetrische Kompression geeignet. Eine asymmetrische Kompression erfordert zum Kodieren einen wesentlich höheren Aufwand als zur Dekodierung. Asymmetrisch komprimierte Daten ermöglichen hierbei höhere Bild- und Tonqualität. Der wesentlich höhere Aufwand ist jedoch gegenwärtig nur bei Anwendungen, wo die Daten einmal kodiert und oft abgefragt und dekodiert werden, tragbar. Interaktive Dialoganwendungen werden eher symmetrische, weniger verzögerte Kodierungen bevorzugen, um gleich kurze Reaktionszeiten bei allen Teilnehmern zu ermöglichen.

Der MPEG-Standard basiert auf Ergebnissen der vorauseilenden JPEG-Gruppe und behält auch eine gewisse Kompatibilität (insbesondere unterstützte Bildgrößen und Frameraten) zum *H.261* Verfahren bei, da dieses bereits vorlag. Neben der Spezifikation der Video- und Audiokodierung besteht der MPEG-Standard auch aus einer *Systemdefinition*, die die Zusammenführung einzelner Datenströme zu einem gemeinsamen, synchronisierten Strom beschreibt. Um die Verhältnisse beim Multiplexen von Video und Audio Informationen in einen MPEG/Systems Strom zu verstehen, ist zuvor ein kurzer Einblick in die jeweiligen Kompressionsverfahren zu geben. Damit kann man den unterschiedlichen Aufbau beider Ursprungsströme sehen und die Schwierigkeiten, die beim Mischen der beiden entstehen, besser verstehen.

2.2. ISO/IEC 11172-2 MPEG 1/Video Grundlagen

Kompressionsverfahren

Für die MPEG/Video-Kodierung verweise ich auf die Literatur, zusätzlich gebe ich noch die für meinen Standpunkt relevanten Informationen wieder. Kurz gefaßt, wird ein einzelnes Bild in Luminanz und Chrominanz-Anteilen getrennt erfasst, die Chrominanz-Anteile werden aus physiologischen Gegebenheiten unterabgetastet.[4][5] Dies ergibt bereits eine nennenswerte Datenreduzierung (Faktor 2). Die bei MPEG spezifizierten Kompressionsverfahren ermöglichen jedoch eine zusätzliche, wesentlich stärkere Kompression des Videosignals (Faktor 20-100).

Die bei der Bilderfassung erhaltenen Bildschichten werden blockweise **DCT-Transformiert** (*Diskrete Cosinus Transformation*), das heißt die Bildinformation in einem Pixelblock bestimmter Größe wird als Überlagerung von Wellenmustern, von Frequenzanteilen, dargestellt. Dies ist nur eine Transformation und ergibt keine Datenkompression. Jetzt werden die Frequenzen quantisiert, das heißt, bestimmte Frequenzen, für die das menschliche Auge weniger empfindlich ist, werden entfernt und nicht übertragen. In den DCT-Koeffizienten treten jetzt viele Nullen auf. Die Koeffizienten werden jetzt so abgespeichert, daß möglichst viele Nullkoeffizienten hintereinander auftreten und in der Tupelform (Anzahl Nullen, nächster Koeffizient ungleich null) abgespeichert. Dies ergibt einen

---

4 Das menschliche Auge ist für etwas über 100 Helligkeitsunterschiede sensibel (8 Bit Auflösung völlig ausreichend); der Farbanteil wird viel ungenauer aufgenommen (weniger Bits ausreichend).

5 Zusätzlich ist das Verhältnis im Auge zwischen Stäbchen und Zäpfchen, die für das Grau- bzw. Farbempfinden zuständig sind, nicht gleich. Es gibt viel weniger Zellen, die Farbe wahrnehmen, und diese sind darum weiter voneinander entfernt. Entsprechend wird bei JPEG/MPEG auch nur jedes zweite oder vierte Bit auch auf Farbe abgetastet.

wesentlichen Kompressionsfaktor. Auf die Tupel wird zum Schluß dann noch das *Huffmann*-Verfahren zur Entropiekodierung verwendet.

Bei MPEG wird jetzt zusätzlich noch zeitliche Redundanz erfaßt. Es gibt *I-Frames (Intra Coded Frames)*, die einfach kodierte Bilder darstellen (nur DCT anwenden, Quantisierung, Huffmann), die selbstständig wieder dekodiert werden können. Dann gibt es *P-Frames (Predictive Coded Frames)*, zu deren Dekodierung auf das jeweils vorangegangene I- oder P-Frame zugegriffen werden muss. Es wird im Prinzip ein Verschiebungsvektor eines Bildblocks zum vorherigen I/P-Frame abgespeichert. *B-Frames (Bidirectionally-predicitive Coded Frames)* beziehen sich zusätzlich auf das vorhergehende und/oder nachfolgende I/P-Frame und erzielen dadurch die höchste Kompression. Eine eher rare Gattung sind *D-Frames*. Sie können bei schnellem Vorwärtsspulen eingesetzt werden, und sind im Prinzip I-Frames mit geringerer Qualität[6] und damit starker Komprimierung. Sie können ohne Bezug auf andere Frames dekodiert werden und sind sehr klein, müssen aber auch alleine in einer *Group of Pictures* auftreten. Bei schnellem Vorwärtsspulen brauchen somit nur wenig Daten übertragen zu werden und jedes Bild kann für sich dekodiert werden. Ein typisch kodierter Videostrom würde etwa folgendermaßen aussehen:

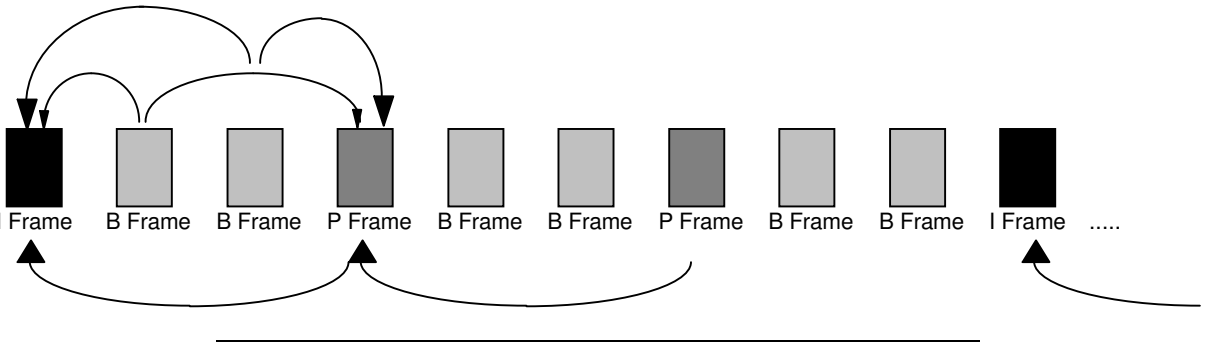


Abb. 1 Aufbau eines Videostromes

Die Länge eines kodierten Bildes kann sich je nach Kodierungsart, Komplexität des Bildes und Ähnlichkeit zu anderen Frames wesentlich unterscheiden. Wir können von keiner konstanten Bitrate mehr ausgehen, sondern nur eine durchschnittliche Bitrate errechnen, je nach Verhältnis und Größe von I-P-B-Frames. Dies wird bei dem System Layer noch ein wesentlicher Punkt werden.

Videostrom

Der Videostrom[7] teilt sich in 6 Schichten auf:

- Die oberste Schicht stellt den **Sequence Layer** dar. Hier wird die Zwischenspeicherung der Daten gesteuert und grundlegende Informationen für den Dekoder abgespeichert. Es ist nicht sinnvoll, einen Datenstrom zu generieren, der beim Dekodieren zu hohe Anforderungen an die benötigte Speicherkapazität stellt. Deshalb enthält der Sequence Layer zu Beginn mehre Einträge, die im *Sequence Header* zusammengefaßt sind und anhand derer der Dekoder

6 Es werden nur noch sehr niedrige Frequenzanteile, die für den Grundfarbton ausreichen, abgespeichert.
7 Ein kommentiertes Beispiel eines Videostromes ist in Anhang A zu finden.

entscheiden kann, ob der Strom für ihn überhaupt dekodierbar ist. Unter anderem enthält der Header notwendige Statusinformationen, Größe des Bildes in Pixeln, mittlere Datenrate, evtl. Quantisierungsmatrizen usw. Der Dekoder muß mindestens einen Sequence Header erhalten, bevor er mit dem Dekodieren des Stroms anfangen kann.[8]

- Der **Group of Pictures Layer** ist die nächste Ebene. Sie beinhaltet mindestens ein I-Frame, auf das ein wahlfreier Zugriff erfolgen kann. Eine Bildgruppe kann damit als solche gekennzeichnet werden und kann als abgeschlossene Einheit betrachtet werden, die gleichfalls einen definierten Startpunkt im Strom darstellt. Durch Setzen eines Bitflags (B*roken Link Flag*) kann dem Dekoder mitgeteilt werden, daß erste eventuell auftretende B- oder P-Frames, die sich auf Bilder vor dieser Gruppe beziehen, übersprungen werden sollen. Dies wäre eine günstige Gelegenheit z.B. beim Vorwärtsspulen eines Filmes, da der Dekoder bei einer so gekennzeichneten Gruppe problemlos mit der Verarbeitung beginnen könnte. Der Nachteil liegt aber daran, daß nicht festgelegt ist, wie groß eine solche GOP sein kann. Wenn das der Enkoder so will, kann er auch zig tausend Frames zu einer Group zusammenfassen. Damit kann man wohl keine exakte Positionierung mehr erzielen und muß gegebenenfalls ein I-Frame innerhalb der Gruppe finden.[9] Ein MPEG-Strom ist im Prinzip nur dafür optimal ausgelegt, sequentiell von Anfang zu Ende abgespielt zu werden. Es ist Aufgabe des Enkoders, den Strom so zu gestalten, daß ein Dekoder auch an anderen, definierten Einstiegspunkten (z.B. GOP's) mit dem Dekodieren beginnen kann.
- Der **Picture Layer** markiert den Beginn eines neuen Frames und damit das Ende des vorhergehenden. Er enthält Informationen darüber, um was für ein Bild es sich handelt (I/P/B/D) sowie auch zeitliche Verschiebungsinformationen.

  Bei einer Sequenz der Art

  ```
  (B B)I B B P B B P ...
  -2-1 0 1 2 3 4 5 6 ...
  ```

  müssen die Frames folgendermaßern abgespeichert werden

  ```
  I(B B)P B B P B B ...
  0-2-1 3 1 2 6 4 5
  ```

  Es muß somit notiert werden, in welcher Reihenfolge die Frames dekodiert und wann sie präsentiert werden sollen. Das wird hier bei MPEG/Video im Picture Header durch durchnumerierten Reihenfolgefelder lokal in einer Gruppe erreicht. Bei einem MPEG/Systems Multiplexer muß anhand der Bildrate, dieser Reihenfolgefelder und einer globalen Uhr zu jedem Bild eine *Decoding Time Stamp* und eine *Presentation Time Stamp* errechnet werden.

  Hier erkennt man auch die Problematik, wenn man irgendwo im Strom aufsetzt. Es befinden sich einige B's und evtl. P's in der Stromreihenfolge, die nicht darstellbar sind. Der Dekoder muß so implementiert sein, daß er solche nicht dekodierbaren Frames überspringen kann.

---

[8] Der Sequence Header kann aber auch beliebig oft im Strom wiederholt werden, damit sich z.B. ein Dekoder auch an anderen Stellen im Strom einklinken kann.

[9] In einem solchen Fall müssen natürlich sofort darauf folgende B-Frames überlesen werden, weil sie sich auf das noch vorhergehende I/P-Frame beziehen.

Da die einzelnen Frames variable Länge haben, waren die MPEG Gremien darauf bedacht, daß erstens die Synchronisationswörter eindeutig sind, und zweitens daß in dem Datenteil der Frames (also die Huffmann-kodierten DCT Koeffizienten oder die Verschiebungsvektoren vorangegangener Blöcke) sichergestellt ist, daß kein solches Syncword durch zufällige Daten entsteht. Das wurde durch *variable-length-Codes* und verschiedener Markierungsbits bei "gefährlichen" Situationen festgelegt. Es kann garantiert werden, daß keine Datenbytes als Syncword mißverstanden werden, der Dekoder dort aufsetzt, synchronisiert und völlig Sinnloses dekodiert. Das ist besonders für Broadcast-Applikationen sinnvoll, wo ein Dekoder zu einem beliebigem Zeitpunkt eingeschalten werden kann. Wir werden feststellen, daß das bei weitem nicht so selbstverständlich ist, wie man vielleicht denken mag. Bei MPEG/Audio wurde diese Möglichkeit nicht berücksichtigt, was Probleme beim Multiplexen des Stroms ergibt und im Extremfall beim Random-Einschalten eines Dekoders zu merkwürdigen Ergegbissen führen kann.

- **Slice Layer**, **Macroblock Layer** und **Block Layer** bilden die nächsten Schichten. Sie sind für uns in Bezug zum MPEG/SYSTEMS Multiplex Verfahren unwichtig und können ggf. in der einschlägigen Literatur nachgelesen werden.

2.3. ISO/IEC 11172-3 MPEG 1/Audio Grundlagen

Kompressionsverfahren

Es sei erstmal vorweggestellt, daß das menschliche Ohr viel genauer auflöst als das Auge, dafür aber eine höhere Reaktionszeit aufweist. Das MPEG-1 Komittee hat ein Kompressionsverfahren entwickelt und es in drei Varianten ausgereizt. Sie sind mit den Namen **MPEG Audio Layer I**, **II** und **III** benannt worden. Die Audiokodierung ist in MPEG kompatibel zu der Kodierung von Audiodaten für eine *Compact-Disc-Digital-Audio* (**CD-DA**) und für ein *Digital Audio Tape* (**DAT**). Das wesentliche Kriterium ist hier die Abtastrate von wahlweise 32 kHz (einige Kommunikationsgeräte), 44.1 kHz (CD-Audio) oder 48 kHz (Studiogeräte) und mit 16 Bit pro Abtastwert. Die Kompression erfolgt hier für je ein Audiosignal zu 32, 64, 96, 128 oder 192 Kbit/s. Es werden drei Qualitätsstufen (*Layer*) mit unterschiedlichem Kodierungs- und Dekodierungsaufwand definiert. Eine Realisierung einer höheren Stufe muß immer auch MPEG-Audio einer niedrigeren Stufe dekodieren können.

Ähnlich zum Verfahren der zweidimensionalen DCT für Video wird hier eine Frequenztransformation vorgenommen. Zu dieser Transformation eignet sich die **FFT** (*Fast Fourier Transformation*). Damit wird der jeweilige Anteil des Spektrums in jedem der 32 nichtüberlappenden Frequenzbändern bestimmt. Nach der Zerlegung des Tonsignals in 32 Frequenzbändern kann die Quantisierung der verschiedenen Spektralanteile hierfür unterschiedlich erfolgen. Es wird über ein *psychoakustisches Modell* parallel zur eigentlichen FFT der Rauschpegel in jedem dieser Frequenzbänder bestimmt. Bei einem hohen Rauschen kann auch eine gröbere Quantisierung erfolgen. Ein geringer Rauschpegel führt zu einer hohen Auflösung. Die so quantisierten Spektralanteile werden in Stufe eins und zwei

einfach **PCM** (*Pulse Code Modulation*) kodiert und in Stufe drei anschließend nach Huffmann entropiekodiert.

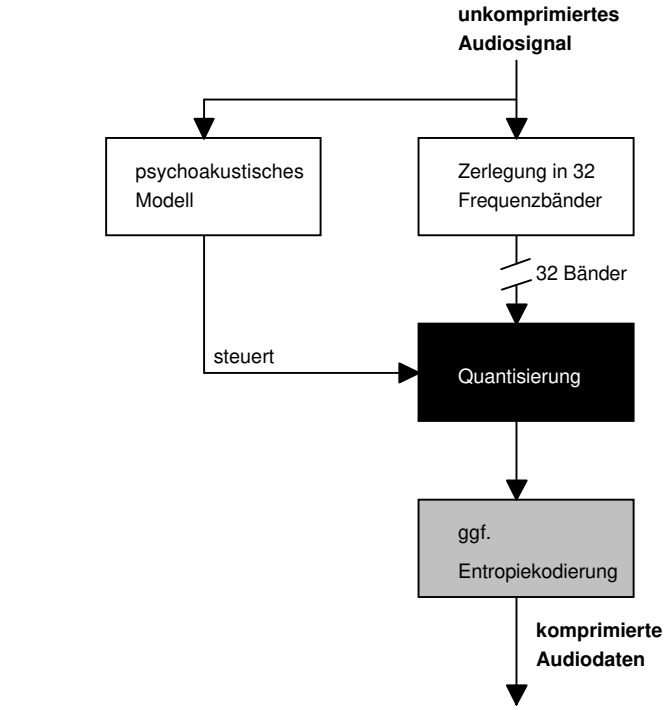


Abb.2 MPEG-Audiokodierung

Layer I ist die einfachste Variante des MPEG Modells, es wird in Frequenzenbänder mit einem psychoakustischen Modell quantisiert. Layer II fügt nocht raffinierte Bitallokations Techniken und größere Genauigkeit dazu, Layer III eine hybride Filterbank und eine nicht uniforme Quantisierung. Layer I, II und III[10] erhöhen jeweils das Verhältnis von Qualität zu Kompression und Rechenleistung.

Fuer CD-Audio Qualität ist eine sampling Rate von 44.1 kHz mit einer Auflösung von 16 Bit pro Kanal nötig. Das ergibt

$$\frac{16\,bit * 2\,Kanälen * 44100\,samples}{\sec} \approx \frac{1.4\,mBit}{\sec}$$ für CD 2kanal Stereo Qualität.

MPEG versucht, die unrelevanten oder redundanten Informationen aus dem Audio-Signal zu entfernen. Da ein Kanal (raw PCM ~ 705 Kbit/s) auf 32kBit/s bis zu 192 Kbit/s (einstellbar) reduziert werden kann, ergibt das eine maximale Kompressionsrate von ca. 22. Normale Kompression liegt in etwa zwischen 1:6 oder 1:7. Wer glaubt, das sei zu wenig (im Vergleich zu Video), dem sei erinnert,

[10] Layer III streams sind praktisch in der Szene nicht vorhanden. Alles basiert z.Zt. auf Layer I und II.

das wir hier mit keinen hörbaren Qualitätsverlust zu tun haben. 96 Kbit/s pro Kanal ist für fast alle Tonquellen mehr als genug.[11]

Die Kompression bei den MPEG Audio und Video Verfahren hat nichts mit *verlustlosen* digitalen *Datenkompressionen* im Huffmann-Stil zu tun, es handelt sich dabei im Wesentlichen um Weglassen von mit Sinnesorganen nicht wahrnehmbaren Daten, das ergibt erst den höchstmöglichen Kompressionsfaktor. Da sich die Menge der Audiodaten nach Abtastfrequenz und Abtastauflösung richten, hat man im Prinzip zwei Möglichkeiten zur Reduzierung der Information: Man vermindert entweder die Abtastfrequenz oder die Auflösung (z.B. weniger als 16 Bit pro Abtastwert verwenden). Der Mensch kann in etwa von 20Hz bis 20kHz hören. Laut dem Theorem von *Nyquist* muß man mit mindestens der doppelten Frequenz abtasten, als die höchste Frequenz, die man wiedergeben will. Mit Spielraum für schlechte Filter arbeitet man bei CD-Qualität mit min. 44.1 kHz für das dynamische Spektrum des menschlichen Gehörorgans. Die Abtastfrequenz kann man darum schon aus physikalischen Gründen nicht reduzieren. Also muß wohl die Auflösung irgendwie reduziert werden.

Der Grund, warum man mit 16 Bit arbeiten will, ist, um ein gutes *Signal/Rauschen Verhältnis* zu erhalten. Für das Ohr entsprechen 6dB einem Verdoppeln des Schalldrucks. CD-Audio erreicht ein Verhältnis von Signal zu Rauschen von über 90dB.[12] Das entspricht ziemlich genau der dynamischen Auflösung des menschlichen Gehörs, darum kann man theoretisch auch kein Rauschen aus dem CD-System selbst entlocken (nur in der Aufnahme). Was würde passieren, wenn man nur mit 8 bit Auflösung abtasten würde? Man erhält ein deutliches Rauschen, da das Verhältnis zwischen Signal und Grundrauschen deutlich reduziert wird. Das kann man in leisen Passagen hören. Der Trick besteht jetzt darin, daß offensichtlich laute Passagen dieses Rauschen, eigentlich alle leiseren Töne, maskieren. Genau da setzt das MPEG-Verfahren an.

Nehmen wir an, wir haben ein starkes Tonsignal mit einer Frequenz von 1000Hz. Gleichzeitig haben wir einen schwächeren, 18 dB leisereren Ton mit einer Frequenz von 1100Hz. Ein solcher schwacher Ton wird von einem im Frequenzspektrum nahegelegenen starken Ton einfach überlagert und maskiert. Unser Gehörorgan kann diese leise Frequenz nicht mehr wahrnehmen. Dieser Maskierungseffekt ist abhängig von Schalldruck und Entfernung benachbarter Töne. Zum Vergleich, eine Frequenz um die 2000Hz würde erst ab ca. 45dB maskiert werden. Dieser Effekt bedeutet, daß man in der Umgebung einer starken Frequenz das Grundrauschen erhöhen kann, weil man es sowieso nicht wahrnehmen wird. Das Rauschen erhöhen entspricht aber einer Reduzierung der Auflösung, weil man einfach jeder Frequenz einen bestimmten Grundpegel gibt und nun weniger Bits für die restliche Amplitude verwendet. Diese Reduzierung der Auflösung ergibt eine Kompression.

---

[11] Rock'n'Roll oder Pop können ruhigen Gewissens so gehandhabt werden. Für klassische Musik sollte bis zu 128 Kbit/s pro Kanal gegangen werden.

[12] Bei einer dynamischen Auflösung von 90dB und binärer Kodierung der Amplitude entspricht je 1 Bit einer Verdoppelung der Amplitude und damit 6dB. Mit 16 Bit können wir damit 16*6=96dB darstellen, ausreichend für die physiologische Auflösung unseres Gehörorgans.

Im Prinzip wird ein Digitalisiergerät gebaut, daß gleich viel Frequenzen wahrnehmen und abspeichern kann, wie das menschliche Ohr. Das Gerät wird diese Frequenzen immer wahrnehmen, unabhängig davon, ob bestimmte Frequenzkombinationen auftreten, die wir so nicht hören können. Beim obigen Beispiel würde das Gerät die Frequenz um die 1000Hz genauso digitalisieren, wie auch die bei 1100Hz. Unser Ohr würde die bei 1100Hz nur dann hören, wenn die bei 1000Hz aber nicht vorhanden wäre (oder weniger laut wäre). Hier wird die Datenreduzierung eingesetzt. Durch Weglassen von für den Gehörsinn redundanten Daten wird so die Technik an den Gehörsinn angepaßt und dadurch eine bessere Nutzung der Bandbreite eines Übertragungs- oder Speichermediums erreicht.

Der MPEG Audio Coder unterteil das Frequenzspektrum (20Hz bis 20kHz) in 32 Schichten (*subbands*).[13] Der Coder berechnet jetzt den Maskierungseffekt von jeder Frequenz in jeder Schicht im Verhältnis zu jeder anderen Frequenz. In jeder Schicht erhält er somit einen Maskierungswert, das heißt einen dB-Wert, unter dem alle Töne maskiert werden. Anhand von dem Wert kann dann für jede Schicht die Auflösung, das Verhältnis zu Signal und Rauschpegel, eingestellt werden. Zusätzlich verfügt der Coder über das Wissen, daß unser Gehörorgan für hohe und niedere Frequenzen nicht gleich sensibel ist.[14] Auch das wird im psychoakustischen Modell berücksichtigt und dort, wo das Ohr empfindlicher ist, wird eine geringerere Quantisierung der Spektralkomponenten aus der Fouriertransformation eingesetzt. Vor und nach einem starken Ton wird auch ein weiteres Maskierungsphänomen (*premasking*, *postmasking*) festgestellt, dessen Auswirkungen ebenfalls genutzt werden. Es gibt eine deutliche Signalstärkenverschiebung (30 - 40 db). Man glaubt, das Gehirn brauche etwas Zeit zum Verarbeiten.

Die Audiokodierung kann sich zusätzlich auf einen Kanal, zwei unabhängige Kanäle und den Stereoton beziehen. Der Stereoton kann nach MPEG in zwei Arten vorliegen: Zum einen lassen sich beide Kanäle völlig unabhängig voneinander verarbeiten, zum anderen gibt es den Modus des *Joint Stereo Mode*. In diesem Fall berücksichtigt MPEG die zwischen beiden Kanälen vorhandenen Redundanzen und erreicht somit einen noch höheren Kompressionsgrad.

Wir haben festgestellt: die Datenreduzierung wird durch Reduzierung der Auflösung erreicht. Sie macht den Hauptteil der Kompressionstechnik aus. Zusätzlich wird in Layer III *Huffmann*-Kodierung nachträglich angewendet.

Die Nachteile des MPEG-Verfahrens sind, daß der Coder, insbesondere bei Echtzeitkomprimierung, iterativ den Maskierungseffekt nachgeht, bis er keine Rechenzeit mehr zur Verfügung hat. Wo jetzt die Zeit (und Bits) verloren gehen, ist implementierungsabhängig. Der Maskierungseffekt wird auch nicht von einer Frequenz weg links und rechts auf der Zeitachse gleich verfolgt. Für Layer I und II bearbeitet der Coder die Tondaten in einem Zeitintervall (*time slice*) von 23ms gleichzeitig. Für bestimmt aufgebautes Tonmaterial kann dies ein Problem sein, weil zum Beispiel der

---

[13] Bei Layer I und II sind diese Frequenzgruppen gleich breit (625 Hz), Layer III arbeitet mit verschiedenen Spannweiten.

[14] Der Peak liegt zwischen 2 und 4kHz, wo auch die menschliche Stimme angesiedelt ist.

Maskierungseffekt einer am Ende einer Zeitscheibe auftretenden Frequenz immer erst am Anfang der nächsten eintreten könnte. MPEG Audio Layer III löst auch dieses Problem und kann flexibler vorgehen.

Audiostrom

Das Prinzip bei MPEG-Audio Kodierung ist der Videokodierung sehr ähnlich, aber doch wurden teilweise andere Resultate erwünscht. Der MPEG Standard schreibt eine *konstante Datenausgangsrate* vor. Die Daten werden, unabhängig davon, wie kompliziert das Tonmaterial ist, tatsächlich immer so komprimiert, daß die Datenrate (z.B. 64 Kbit/s) eingehalten wird.

Ein Audiostrom gliedert sich in **Frames**[15], diese in **Audio Access Units** und diese wiederum in **Slots**. Ein Slot besteht bei der niedrigsten Komplexität der Kodierung (Ebene 1) aus vier Byte, sonst jeweils aus einem Byte. Ein Frame besteht immer aus einer festen Anzahl von Abtastwerten. Am wichtigsten ist hier der Begriff der Audio Access Unit. Darunter ist die kleinste Audiosequenz komprimierter Daten zu verstehen, die unabhängig von allen übrigen Daten vollständig dekodiert werden kann. Eine Audio Access Unit enthält Informationen über 384 (MPEG Layer I) bzw. 1152 (MPEG Layer II) Samples. Je nach Bitrate, Digitalisierfrequenz und Layer läßt sich damit die Größe einer solchen Access Unit berechnen, sie beträgt

$$\frac{384\ bytes * bitrate}{8\ bits * Frequenz} \text{ bzw } \frac{1152\ bytes * bitrate}{8\ bits * Frequenz}.$$[16]

Auch die Spieldauer einer solchen Access Unit kann in der folgenden Tabelle festgehalten werden.

	Layer I			**Layer II**		
Anzahl Samples pro AU	384			1152		
Mögliche Frequenzen kHz	32	44.1	48	32	44.1	48
Spieldauer ms	12	8.7	8	36	26.1	24

Es besteht die Möglichkeit, zusätzliche Informationen in den AU's abzuspeichern. So kann auch eine *CRC-Checksumme* eingefügt werden. Es sei erwähnt, daß sich dadurch die Größe der AU's nicht vergrößert. Es wird dafür einfach Toninformation höher komprimiert. Pro 1152 originalen Samples würde man beim Beispiel mit 44,1 kHz, 64kBit/s, Layer II = 208,98 Bytes immer eine AU Größe von 208-209 Bytes erhalten, egal, ob die 1152 Original Samples komplizierte Musik oder nur völlige Stille enthalten.

---

[15] Ein kommentiertes Beispiel eines Audio Frame Headers ist in Anhang B zu finden.

[16] z.B. bei 44,1 kHz, 64 Kbit/s, Layer II = 208,98 Bytes. Die meisten Access Units sind hier demnach 209 Byte lang, einige (gesondert markierte) nur 208 Byte.

Die MPEG Gremien waren bei der Definition des Audiostroms offensichtlich nicht so vorsichtig wie beim Videoteil. Audiodaten können hier sehr wohl als Syncwörter mißverstanden werden. Das heißt, der Dekoder muß vom Anfang des Streams ausgehen, um den ersten Header zu lesen. Dort kann er dann die Größe der Unit errechnen und somit die nächsten x Bytes als kodierte Audio Daten einlesen. Die Position des nächsten Headers kann exakt berechnet werden. Wegen dieser Eigenschaft, die beim Videoteil nicht vorhanden ist, wird die Toninformation selbst ohne Rücksicht darauf, daß eventuell ein Syncword entsteht, erstellt. Wenn der Dekoder demnach irgendwo im Stream aufsetzt, besteht die Möglichkeit, daß er fälschlicherweise Audiodaten als Header-Syncword erkennt. Besonders beim Dekodieren eines Multiplexed Stream, wie es z.B. in Broadcast Applikationen irgendwo im Strom der Fall sein kann, muß der System Dekoder überprüfen, ob er jetzt ein echtes Syncword erhalten hat, oder ob es nur Audiodaten sind. Das könnte realisiert werden, indem z.B. die Position des nächsten Syncs errechnet wird, dort nachgeschaut und festgestellt wird, ob es wieder ein gültiges Syncwort ist. Das könnte einige Male wiederholt werden, und die Wahrscheinlichkeit, das es sich um ein Audio-Datum handelt, jeweils quadratisch reduziert werden. Wenn der Dekoder glaubt, es handle sich um korrekte Syncwörter, kann er mit dem Dekodieren anfangen. Im schlimmsten Fall würde er Schrott erzeugen, ein Reset seitens des Endanwenders müßte das System mit einem neuen Start versehen. Etwas mehr Vorsicht bei der Realisierung des MPEG-Audio Teils hätte diesen Mißstand ersparen können. Ein Beispiel sei der Video-Teil, wo dies nicht vorkommen kann.

2.4. ISO/IEC 11172-1 MPEG 1/Systems Grundlagen

Multiplex Grundlagen

Zusätzlich wird im MPEG International Standard über die *Systemdefinition* die Zusammenfassung dieser beiden Datenströme zu einem einzigen definiert. Bei dieser Zusammenfassung wird als wichtigste Aufgabe das eigentliche Mischen definiert. Hierzu zählt auch die *Koordination* beim Datentransfer zwischen den einkommenden Datenströmen und den herausgehenden, die *Justage* der Uhren und das *Buffermanagement*. Als grobe Einheit wird der nach ISO 11172 definierte Datenstrom in einzelne *Packs* unterteilt. Der Dekoder erhält für diesen aus Packs bestehenden, vermischten Datenstrom die notwendigen Startdaten. Als Beispiel sei hier die maximal auftretende Datenrate genannt, die zu Beginn eines jeden ISO-11172-Datenstroms in dem ersten Pack enthalten sein muß. Bei der Definition dieses Datenstromes werden implizite Annahmen getroffen: Eine solche Headerinformation kann sehr gut bei den auf einem Sekundärspeichermedium abgelegten Daten vor einem ersten, ggf. wahlfreien Zugriff, gelesen werden. Auch in einem dem Telefon oder Videofon entsprechenden Dialogdienst über Kommunikationsnetze wird ein Teilnehmer immer zuerst diese erhalten. Als kritisch ist hier jedoch eine Verteilungsanwendung anzusehen. Ein Teilnehmer kann sich zu einem beliebigen Zeitpunkt hier aufschalten. Ihm steht dann diese benötigte Headerinformation nicht direkt zur Verfügung, weil ein ISO-11172-Datenstrom die benötigte Information nur am Anfang oder in einigen Abstanden überträgt. Es kann Schwierigkeiten geben, diese zeitlich auftretenden Informationen auszufiltern. Über ein zu definierendes Protokoll könnte diese Information jedoch zusätzlich bei Bedarf übertragen werden.

Ein MPEG/Systems Stream besteht aus Datenblöcken aus mehreren MPEG Audio/Video/Padding/ User/Private- Streams. Es können bis zu 15 Video und 31 Audio Streams verknüpft werden. Die vom MPEG/Systems Standard definierte übergeordnete Schicht unterstützt folgende Aufgaben:

- **Synchronisation** (mittels **TimeStamp**) von mehreren komprimierten Streams bei der Ausgabe. Diese Zeitstempel beziehen sich immer relativ auf eine Beziehung zwischen den gemultiplexten Strömen und nicht absolut zu anderen gemäß ISO 11172 existierenden Datenströmen.
- **Gruppierung** von Daten aus den Ursprungsstreams in Einheiten, die im System Stream aneinandergereiht werden
- **Buffermanagement** für die einzelnen Datenstreams

Die MPEG Gremien haben außer dieser Schicht nicht einen MPEG/Systems Enkoder definiert, sondern das Modell eines einfachsten Dekoder, den **STD** (*System Target Dekoder*). Bei der Generierung eines MPEG/Systems Stream sind einem zahlreiche Freiheiten gesetzt, jedoch muß der erzeugte Stream von diesem einfachen STD dekodierbar sein. Dies liefert i.d.R. die größten Einschränkungen. Der STD hat keine Kontrolle über den einkommenden Stream. Er kriegt die Daten kontinuierlich, mit der immer konstanten durchschnittlichen Bitrate geliefert und muß hoffen, daß der Strom so gebaut ist, daß seine Buffer nicht über- bzw. unterlaufen.

Systemstrom

Der MPEG Strom[17] sieht im Prinzip folgendermaßen aus:

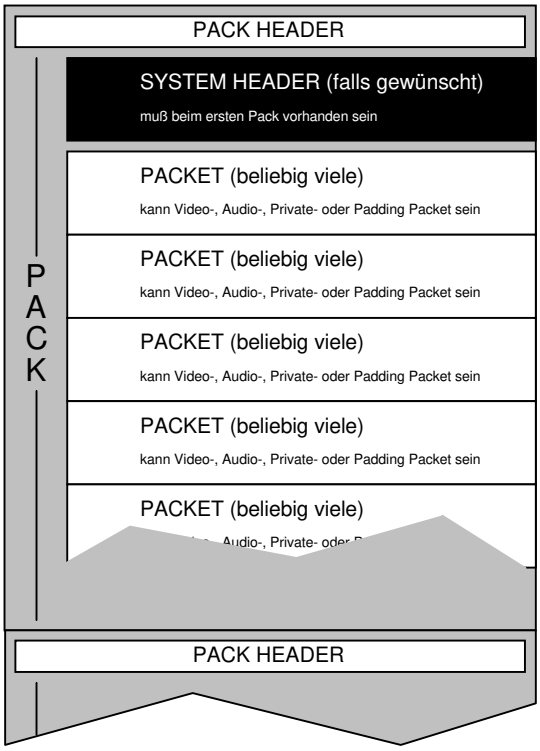


Abb.3 MPEG/SYSTEM Strom

[17] Ein Dump eines MPEG/System Stromes ist in Anhang C zu finden.

- **Packets** enthalten Daten aus den ursprünglichen Streams. Die Wahl der Packetgröße ist dem Enkoder überlassen, muß jedoch vom STD verarbeitbar sein.[18] Am Anfang des Packets finden wir Headerinformationen über Art des Packets (Audio, Video, Padding, Private und User), Größe (das impliziert auch, daß sich Packet-Größen verändern können, dies wird jedoch i.d.R. nicht empfohlen) sowie *Decoding Time Stamps* (**DTS**) und *Presentation Time Stamps* (**PTS**)[19]. Man muß dem Dekoder mitteilen, wann er mit dem Dekodieren anfangen soll und wann er das Dekodierte wiedergeben soll. Wir erinnern uns: Die bei MPEG/Video im Picture Layer enthaltene *Präsentationsreihenfolge* wird nicht mehr berücksichtigt. Eine exakte Berechnung der DTS und PTS ist Voraussetzung dafür, daß der Film überhaupt korrekt abläuft. Der Multiplexer wird anhand der Bildreihenfolge im Picture Layer feststellen, um wieviel *Picture Periods*, also Bilddauerverzögerungen, die Anzeige im Vergleich zur Dekodierung verzögert werden soll und geeignete TimeStamps errechnen. Zu diesen TimeStamps kann der STD eine immer konstante Dekodierverzögerung addieren. Diese spielt aber beim Vorbereiten des Stroms keine Rolle, es wird davon ausgegangen, der Dekoder würde null Zeit fürs Dekodieren brauchen. Bei Audio-Packets ist demnach die berechnete DTS immer gleich PTS, bei Video wird bei I- und P-Frames lediglich eine PTS Verzögerung einberechnet. So müßten z.B. im folgenden Stück einer Videosequenz die Timestamps folgende Größen aufweisen:

Typ	DTS	PTS
	TS = (Anzahl * Dauer Pict.Period)	
I	0 * PP	1 * PP
P	1 * PP	4 * PP
B	2 * PP	2 * PP
B	3 * PP	3 * PP

Der MPEG-Standard schreibt vor, daß DTS/PTS nur dann in einem Packet Header stehen können, wenn auch wirklich eine Access Unit, also ein Bild oder Audioframe, im folgenden Packet anfängt. Bei Audio beginnt in der Regel immer eine Access Unit in jedem Packet,[20] somit bezieht sich die DTS/PTS nur auf die erste Access Unit, die im Packet vorkommt. Der STD kann für im gleichen Packet vorkommende Units entsprechende TimeStamps interpolieren. Bei Video Streams sind I-Pictures teilweise sehr lang (z.B. 15 kByte), werden demnach in mehreren Packets unterteilt. PTS/DTS wird nur dann berechnet und mitkodiert, wenn in dem Packet tatsächlich auch eine solche Unit anfängt.

---

[18] Ein STD hat z.B. einen Audio Buffer von 4 kByte. Man wird demnach eine Packetgröße kleiner gleich der Buffergröße wählen, sagen wir in etwa die Hälfte, weil ein Packet im Dekoder ankommen muß, bevor auch die erste darin vorkommende Access Unit dekodiert werden kann.

[19] Die TimeStamps beziehen sich immer auf Zyklen einer 90kHz globalen Clock.

[20] Eine Audio Access Unit ist einige hundert Bytes lang, ein Packet deutlich größer.

- Beliebig viele Packets werden in **Packs** gruppiert. Es ist dem Multiplexer viel Freiheit gegeben, wie viele Packets pro Pack usw., doch auch dafür gibt es Eckdaten an die man sich halten kann. Z.B könnte für ein spezielles Medium mit eigenen Sektorgrößen das Verhältnis Packetgröße/ Packetanzahl/ Packgröße/ Packanzahl optimiert werden, so daß physikalische Sektoren mit logischen Packets übereinstimmen. Am Anfang jedes Packs wird eine weitere Art von Time-Stamp berechnet. Die **SCR** (*System Clock Reference*) gibt an, zu welchem Zeitpunkt das letzte Byte des SCR Feldes des Pack Headers im Dekoder eintreffen sollte. Damit wird dem Dekoder letztendlich immer wieder mitgeteilt, zu welchem Zeitpunkt dieses Byte eingelesen werden sollte, der Dekoder kann somit feststellen, ob das Medium zu schnell/zu langsam ist und ob es Probleme mit der Datenrate gibt. SCR ist wichtig im Kontext mit der Synchronisation mit der DTS/PTS, da der Enddekoder anhand dieses SCR Wertes auch seine inneren Uhren immer wieder resynchronisiert.

Betrachten wir, wie der Dekoder möglicherweise auftretende *Time Outs*, Verzögertes Eintreffen von Daten, die bereits präsentiert werden sollten, feststellt. Dazu gibt uns auch folgende Skizze Einblick in den Beziehungen zwischen den einzelnen TimeStamps.

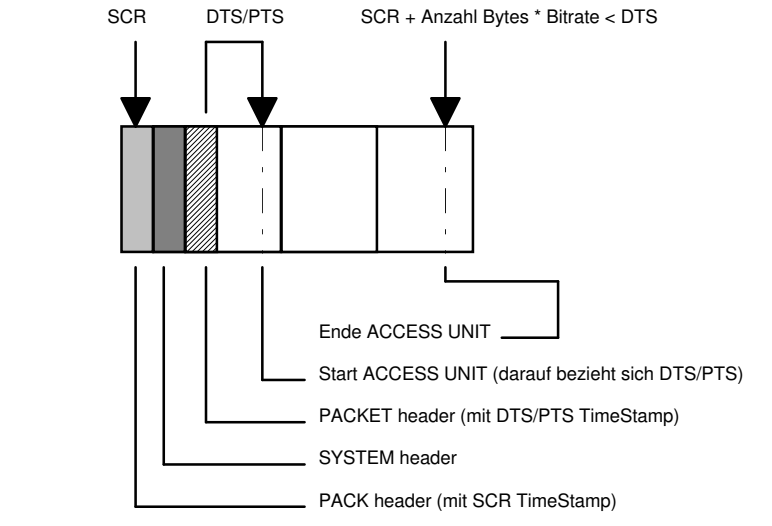


Abb. 4 Beziehungen zwischen den TimeStamps

Falls die *aktuelle SCR + Anzahl Daten Access Unit * Datenrate > DTS* ist, gab es einen *Bufferunterlauf*. Der Dekoder sollte mit dem Dekodieren anfangen, hat aber noch nicht die ganze Access Unit erhalten. Falls hingegen bestimmte Audiodaten zu früh geschickt werden und die DTS der letzten im Audio Buffer enthaltenen Access Unit noch nicht eingetroffen ist, das heißt der Buffer sich nicht genügend geleert hat, gibts es ggf. einen *Bufferüberlauf*. Diese Problematik ist beim Erzeugen des Stroms immer im Auge zu behalten, und es passiert tatsächlich, daß manchmal sowohl der Video- als auch der Audiobuffer keinen Platz für das nächste Packet enthalten. Das Medium müßte jetzt irgendwie eine Pause machen, da es

wahrscheinlich zu schnell war.[21] Da vom Standard die Möglichkeit zur Interaktion mit dem Medium voraussichtlich erst mit MPEG 2 gegeben wird, muß man zur Zeit einfach davon ausgehen, daß das Medium (Beispiel: CD ROM) konstante Datenraten liefert. Falls weder Video- noch Audiobuffer Daten aufnehmen können, muß ein sogenanntes **Padding Paket** geschickt werden. Das ist ein ganzes Packet (also soundsoviel Kilobyte) an *Padding Bytes*, die im STD Dekoder einfach im Sink landen. Dadurch kann das Medium seine zu hohe Datenrate wieder ausgleichen, während der Dekoder seine Buffer räumen kann. Tests bei gelieferten MPEG/System Streams weisen auf ca. 14%-15% Padding Bytes hin. Was an Komprimierung in den einzeilen Streams vorhanden war, wird hier gewissermaßen geschenkt, um billige, nicht intelligente Medien bzw. Dekoder zu ermöglichen. Wird die im System Header angegebene Datenrate ziemlich genau berechnet (kann nur in Schritten zu 50 Bytes/sec angegeben werden), so reduziert sich der Prozentsatz an solchen Padding Bytes erheblich.

- Ein **System Header** kann nach Beginn eines beliebigen Packs geschickt werden, muß aber mindestens einmal, zu Beginn des Streams, übertragen werden. Es enthält grundsätzliche Systeminformationen, so z.B. die Datenrate und Info über benötigte Buffergrößen. Ein Dekoder kann nur dann mit dem Dekodieren anfangen, wenn er einen korrekten System Header eingelesen hat. Sollte geplant sein, den Dekoder irgendwo mitten im Strom starten zu lassen, so ist es zweckmäßig, entsprechend oft einen System Header zu generieren.

2.5. Ausblick: Der MPEG 2 Draft International Standard

MPEG 2 Grundlagen

MPEG definiert eine vollkommen ausreichende Audioqualität. Hier konnte auch schon auf wesentliche Entwicklungen aus anderen Bereichen zurückgegriffen werden. Die Qualität von MPEG-Video bei etwa 1,2 Mbit/s (ohne Ton) ist vergleichbar mit VHS-Aufzeichnungen. Der Algorithmus ist im Zusammenhang mit der erforderlichen Kompression sehr gut für eine Auflösung von ca. 360x240 Pixel. Selbstverständlich sind auch höhere Auflösungen kodierbar, bei beispielsweise 625 Zeilen geht dies jedoch auf Kosten der Qualität. MPEG2 ist eine datenstromkompatible Erweiterung zur Reduktion auf eine Datenrate von ca. 4 bis 10 Mbit/s.

Generell definiert MPEG nur den Prozeß der Dekompression, nicht den Dekoder selbst. Hier entstehen heute unterschiedliche Systeme, von denen bisher nur wenige Produkte auf dem Markt sind; viele wurden jedoch angekündigt. Als Beispiel sei hier die CD-I Variante von Philips genannt. MPEG-Chips oder Chipsets werden von AT&T, C-Cube, Cypress, CGT und anderen in einer großen Stückzahl auf dem Markt eingeführt werden.

Man geht heute davon aus, daß sich die Qualität einer nach MPEG optimal komprimierten Videosequenz nicht mehr wesentlich bei der vorgegebenen maximalen Datenrate von ca. 1,5 Mbit/s verbessern läßt. Hier soll nur das Ergebnis (der Kompressionsfaktor und die Qualität) zählen, der

[21] Die im System Header angegebene Datenrate war wohl etwas zu hoch angerechnet; auch kleine Abweichungen sammeln sich in der Zeit auf.

Aufwand, insbesondere ob die *Komprimierung in Echtzeit* oder *Offline* stattfinden, wird nicht berücksichtigt. Im Bereich der Bewegtbild-Kodierung wird deshalb an Verfahren der Kompression für einen Bereich von ca. 4-10 Mbit/s gearbeitet. Dies wird mit MPEG2 bezeichnet. Während MPEG1 die bisher etablierten Verfahren kennzeichnet, zielt MPEG2 hier auch auf eine höhere Bildauflösung, ähnlich der digitalen Videostudionorm *CCIR 601*. MPEG2 betrachtet, ähnlich dem hierarchischen Mode in JPEG, eine Skalierungsmöglichkeit der komprimierten Bewegtbilder. Hier werden bei der Kodierung Bewegtbilder in verschiedenen Qualitäten komprimiert, so daß bei einer Dekompression derselben Daten verschiedene Alternativen zur Verfügung stehen.

- Eine **räumliche Skalierung** ermöglicht die Dekompression von Bildsequenzen mit unterschiedlichen Auflösungen.
- Eine **Skalierung der Datenrate** ermöglicht bei der Wiedergabe die Darstellung einer unterschiedlichen Anzahl von Vollbildern pro Sekunde. Dies wird heute in MPEG 1 über die *D-Frames* definiert.
- Eine **Skalierung der Bittiefe**, Auflösung der Quantisierung der DCT-Koeffizienten, bietet die Möglichkeit eines progressiv genaueren Bildaufbaus.

Bei MPEG2 wird beispielweise auch der Verlust einzelner *ATM-Zellen* eines kodierten Datenstromes betrachtet. Es sollen definierte Sequenzen verschiedener Bildarten (I,P,B) bestimmt werden, die eine minimale Ende-zu-Ende-Verzögerung bei gegebener Datenrate ermöglichen.

Laut Pressebericht der MPEG-Gremien vom 25. März 1994 hat die MPEG2 Spezifikation den Status von *Draft International Standard* erreicht. Die Arbeit im Bereich der Bildkomprimierung wird jetzt mit subjektiven Testreihen fortgesetzt, dasselbe gilt für den Multikanal-Audio Teil. Während MPEG1 nur bis zu zwei Audiokanäle zur Verfügung stellt, arbeitet MPEG2 mit bis zu 5, genauer gesagt 5.1 Kanälen.

- main left
- main right
- surround side
- surround rear
- center
- 100 Hz special effects

Für Bitraten unter 64 Kbits/s versteht MPEG2 auch halbe Datenraten von 16kHz, 22.05kHz oder 24kHz.

MPEG 2 Extensions

Es wird auch heftig an den sogenannten *MPEG2 Extensions* gearbeitet. Es handelt sich dabei um zahlreiche Features, die den gegenwärtigen Draft Standard ergänzen werden. Insbesondere erhält MPEG2 einen *Digital Storage Media Command and Control* (**DSM-CC**), ein Protokoll der die Interaktion zwischen Terminals und Videoservern, Video Datenbanken, CD-gestützten Spielen und

ähnliches unterstützen wird. Auch sind zahlreiche Erweiterungen im Bereich der Datenbank- und Netzwerkapplikationen geplant. Weiters soll eine Unterstützung für 10-bit Video Kodierung für Spezialanwendungen eingearbeitet werden. Die dritte Erweiterung soll eine **NBC** (*Non Backwards Compatible*) Audiokodierung implementieren, da diese Kompatibilität zu MPEG1 für Dekoder einige Performanceeinbußungen produziert.

Gegen Ende 1994 soll MPEG2 den *Working Draft Status* und bis Mitte 1996 soll der Status von *International Standard* erreicht werden.

3. Multiplex Implementierung

3.1. Die Sourcen

Die geforderte Aufgabenstellung war die Implementierung eines *offline MPEG/Systems Muliplexer*. Von bereits vorhandenen MPEG Grundstreams sollte ein von einem Standard Dekoder abspielbarer System Strom erzeugt werden. Die zum Kompilieren benötigten Sourcen sind

- `main.c` (Main)
- `main.h` (Daten- und Typdefinitionen, Funktionsprototypen)
- `inputstream.c` (Verifikation der Inputstreams und Sammeln der Informationen über Access Units)
- `multiplex.c` (Multiplex Hauptschleife)
- `systems.c` (Generierung von Packs, Sysheaders, Packets)
- `interactive.c` (Interaktion mit dem Benutzer, Ein-/Ausgabe von Informationen)
- `timecode.c` (Verwaltung der ISO-Timecode Formate)
- `buffer.c` (Verwaltung der verketteten Liste für die STD Buffer)
- `inits.c` (Initialisierung mancher Strukturen)

- `bitstream.c` (Von der ISO MPEG Software Simulation Group zur Bearbeitung von Bitstreams)
- `bitstream.h` (entsprechendes Header File)

- `multiplex.mak` (Standard Unix Style Makefile)

Als Compiler wurde sowohl der Sun-Standard K&R-Style C Compiler verwendet, wie auch der GNU GCC Compiler. Da die Sourcen selbsterklärend sein sollten, gebe ich im Folgenden lediglich einen groben Überblick über die Strategie, die beim Vermischen der Ursprungsstreams verfolgt wird und über die vom Multiplexprogramm verarbeiteten Daten. Falls Interesse an dem genauen Vorgang besteht, möchte ich auf die Dokumentation in den Quellcodes verweisen.

3.2. Programmaufbau

In erster Ebene betrachtet besteht das Programm aus vier Arbeitsschritten:

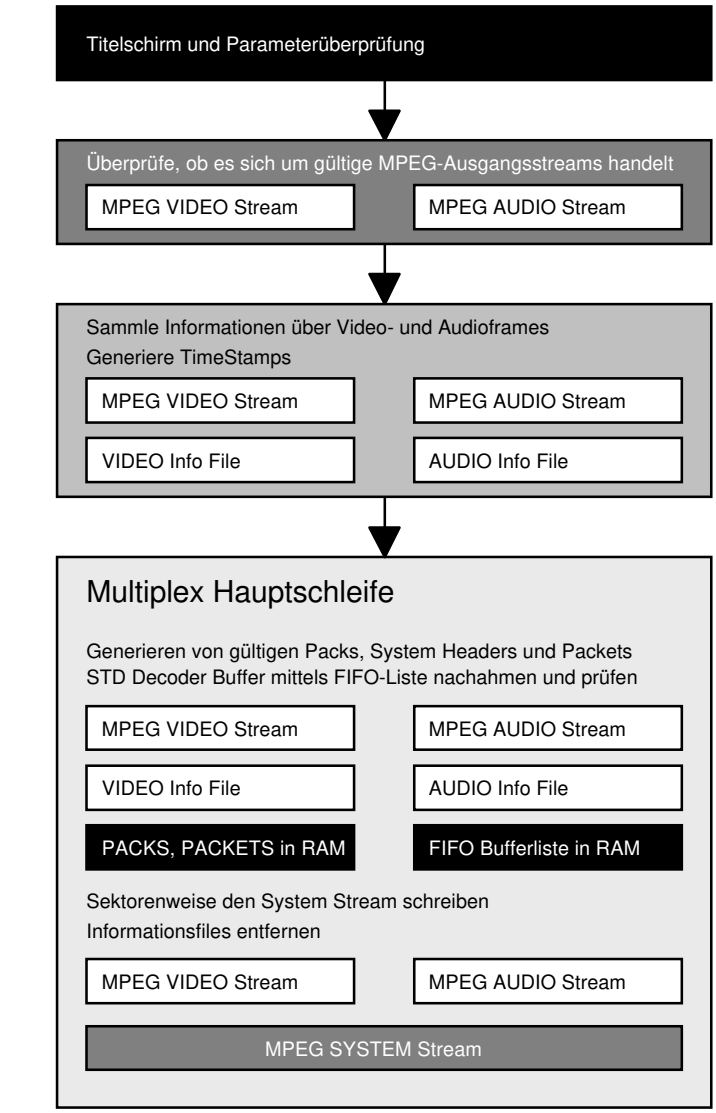


Abb. 5 Grundstruktur Multiplexprogramm

3.3. Programmablauf

Titelschirm, Parameterüberprüfung und Input Stream Kontrolle

Aufruf des Programms mit

```
mplex name_instream1 name_instream2 name_outstream

*************************************************************
*                MPEG   SYSTEMS   Multiplexer                *
*                Christoph Moar,  April  1994                *
*                SIEMENS   ZFE   ST   SN   11                *
*************************************************************

File name_instream1.mpg is a 11172-2 Video stream.
File name_instream2.mpg is a 11172-3 Audio stream.
```

Das Programm hat bereits die ersten zwei groben Arbeitsschritte getätigt. Die Eingabezeile wurde gescannt, die Existenz der Input Streams überprüft sowie deren MPEG-Art und Gültigkeit verifiziert. Falls eine dieser Kontrollen einen Fehler entdeckt, steigt das Programm mit detaillierter Fehlermeldung und Exit-Code 1 aus.

Informationen über MPEG-Grundstreams sammeln

```
Scanning Video stream for access units information.
Got 71 picture headers. 99%
Done, stream bit offset 3270160.

+------------------ VIDEO STREAM INFORMATION -----------------+

Stream length  :   408770
Sequence start :        1
Sequence end   :        0
No. Pictures   :       71
No. Groups     :        7
No. I Frames   :        7 avg. size 11452 bytes
No. P Frames   :       18 avg. size  8979 bytes
No. B Frames   :       46 avg. size  3630 bytes
No. D Frames   :        0 avg. size     0 bytes
Horizontal size:      352
Vertical size  :      240
Aspect ratio   :   1.0000 VGA etc
Picture rate   :   24.000 frames/sec
Bit rate       :   143750 bytes/sec (1150000 bits/sec)
Computed rate  :   138750 bytes/sec
Vbv buffer size:    40960 bytes
CSPF           :        1

Continue processing (y/n) : y
```

Das Programm arbeitet sich jetzt durch den MPEG Video Stream. Er sammelt sowohl globale Informationen über den Stream (werden in der Struktur `Video_struc video_info` abgespeichert), als auch detaillierte Informationen über jedes einzelne Frame. Frame-Type und Länge des kodierten Frames in Bytes werden zusammen mit errechneten DTS und PTS in einer vorher im aktuellen Verzeichnis generierten temporären Datei abgespeichert. Es handelt sich um vitale Informationen für das später kommende Multiplexverfahren. Es werden grundlegende Informationen angezeigt, und der Anwender kann bestimmen, ob er die Berechnung fortführen will oder nicht. Sollten wir gerade einen nicht weiter verarbeitbaren MPEG Video Stream begegnet sein (z.B. *variable Bit* oder *Picture Rate*), so kann hier schon abgebrochen werden. Besonders die Angabe **CSPF** (C*onstrained System Parameter File*) sollte unsere Aufmerksamkeit erhalten. Streams, die dieses Bit gesetzt haben, lassen sich i.d.R. besonders problemlos verarbeiten.

```
Scanning Audio stream for access units information.
Got 106 frame headers. 99%
Done, stream bit offset 531648.

+------------------ AUDIO STREAM INFORMATION ------------------+

Stream length  :    66456
Syncwords      :      106
Frames         :        6 size    626 bytes
Frames         :      100 size    627 bytes
Layer          :        2
CRC checksums  :       no
Bit rate       :    24576 bytes/sec (192 kbit/sec)
Frequency      :     44.1 kHz
Mode           :        0 stereo
Mode extension :        0
Copyright bit  :        0 no copyright
Original/Copy  :        0 copy
Emphasis       :        0 none

Continue processing (y/n) : y
```

Auch hier ist der Audio Stream fertig verarbeitet worden. Globale Parameter werden gesammelt (und in der Struktur `Audio_struc audio_info` abgespeichert) und einzelne Frameinformationen in der schon vorher erstellten temporären Datei abgespeichert. Auch hier kann bei proprietären Eigenschaften des Quellstreams entschieden werden, die Verarbeitung abzubrechen. Fehler im Aufbau eines der beiden Urpsrungsstreams werden von den Routinen entdeckt und gemeldet.

Bei "normalen" Quellstreams sind in dieser Phase insgesamt ca 50-100 kByte temporärer Plattenspeicher pro Minute Film nötig.

MPEG/Systems Parameter eingeben

```
Merging elementary streams to MPEG/SYSTEMS multiplexed stream.

+------------------ MPEG/SYSTEMS INFORMATION ------------------+

sector size (CD-ROM 2324 bytes)          : 2324
packs to packets ratio                1 : 3

STD video buffer in kB (CSPS: max 46 kB) : 40
STD audio buffer in kB (CSPS: max  4 kB) : 4

computed multiplexed stream data rate    : 170609.000
target data rate (e.g. 170650)           : 170650

startup sectors_delay                    : 2

Very verbose mode (y/n) : y
```

Es werden vorerst für das Verfahren einstellbare Eckwerte abgefragt.

- Der Anwender kann die Größe der Target-**Sektoren** angeben. Ein Sektor ist die für den Systemstream gewählte Speichereinheit. Durch Angabe der Sektorgröße werden implizit auch alle anderen Größen, wie Pack oder Packet Größen eingestellt. In der Regel wird ein Packet einem Sektor entsprechen, außer wenn an dessen Anfang noch ein System- oder Pack Header mit abgespeichert wird. Dann reduziert sich die Packetgröße um diesen Pack/System Overhead. Mit diesem Verfahren kann der Ausgabestream ideal an den Eigenschaften eines Speichermediums angepaßt werden. Bei CD ROM empfiehlt sich eine Sektorgröße von 2324 Bytes, da dies der natürlichen Sektorgröße entspricht. Bei Festplatten könnte z.B. mit 2048 Bytes o.ä. gearbeitet werden.

- Die **packs-to-packets-ratio** gibt an, jede wieviel Packets ein Pack- und Systemheader abgespeichert werden soll. Das kann beliebig eingestellt werden, es sollte jedoch beachtet werden, daß ein Pack Header mindestens jede 0,7 sek eintreffen sollte. Bei einer Datenrate von max. ca. 1,4 Mbit/s (183.500 Kbyte/s), wie sie bei MPEG i.d.R. verarbeitet wird, und einer Sektorgröße um die 2 Kbyte, ist ein Verhältnis unter 1:60 anzugeben. Deutlich kleinere Verhältnisse sind jedoch zu empfehlen. Das hilft dem STD, sich der Geschwindigkeit seines Mediums anzupassen und seine internen Uhren exakt zu justieren.

- Die **Buffergrößen** beziehen sich auf die im System Target Decoder implementierten Buffer. Für Dekoder, die sich an dem **CSPS** (*Constrained System Parameter Set*) halten, sind Eckwerte angegeben. Die notwendige Größe für den Videobuffer kann man auch der Videostrominformation dem Feld `Vbv buffer size` entnehmen, sie ist dort aber nicht immer vorhanden. Besonders beim Video-Buffer scheint die uns zur Verfügung stehende Karte nicht ganz 46 Kbyte zu besitzen. Mit 40 Kbyte hat die Karte keine Probleme.

- Das System errechnet je nach Sektorgröße, packs-to-packet-ratio und Bitraten der Ursprungsstreams eine gesamte **Datenrate**. Diese kann vom Anwender eventuell noch erhöht oder erniedrigt werden, ist aber i.d.R. bereits großzügig bemessen. Das System erhöht nach MPEG-Standard die Datenrate zu einem Vielfachen von 50 Bytes/s.

- Die vom System errechneten TimeStamps sind so bemessen, daß das Abspielen der ersten Audioinformation immer synchron mit der Darstellung des ersten Frames beginnt. Zusätzlich wird eine knapp bemessene Verzögerung dazugerechnet, damit die in Packets abgeschickten Audio- und Videodaten auch vollständig im Dekoder ankommen.

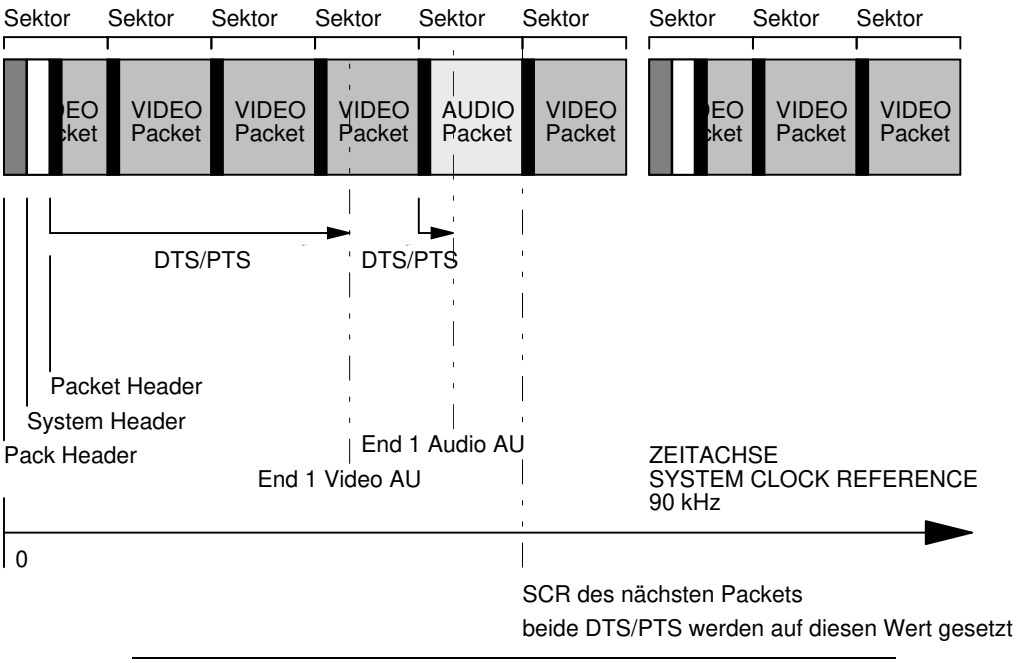


Abb. 6 Offsetberechnung für die ersten DTS/PTS

Wir erinnern uns, daß wir gerade eine Datenrate für den fertigen System Stream angegeben haben. Diese errechnet sich aus den Datenraten der Ursprungsstreams und dem Overhead, den die verschiedenen Pack-System-Packet-Header produzieren. Aber besonders während der ersten 10-15 Pakete kann diese Datenrate aus der Sicht des Dekoders nicht eingehalten werden. Überlegen wir uns warum. Wenn wir jetzt für das erste Bild und die ersten paar Audio-Informationen insgesamt jeweils 15000 und 500 Bytes übertragen müßten, so stellen wir fest, daß wir das nicht können. Wir müssen alles Packetweise abschicken, das ergibt uns, sagen wir mal, 16384 Bytes (davon vielleicht 1200 schon des nächsten Frames) für Video und 2048 (davon ca. 1500 der nächsten Frames) für Audio. Anhand dieser übertragenen Bytes und der allgemeinen Datenrate errechnen wir die ersten sinnvollen TimeStamps, das wäre nach Eintreffen des Audiopackets. Das reicht nicht immer für ein fehlerfreies Abspielen aus. Wir haben es hier mit ziemlich komplexen Bufferverhältnissen zu tun, ab diesem Zeitpunkt werden nämlich alle folgenden TimeStamps nur mehr durch Picture Rate bzw. Sample Rate gegeben. Die SCR, die Eintreffzeit der Pack Headers nach der allgemeinen Datenrate, wird nur mehr nach Anzahl der übertragenen Bytes errechnet. Es passiert jetzt, daß der Dekoder die Daten für das jeweils nächste Video/Audio Frame erwartet. Es würde nach dem von der allg. Datenrate errechnet SCR auch klappen, wenn jetzt wirklich nur die nötigen Video- bzw. Audiodaten übertragen würden. Da aber wie oben wieder etwas zuviel Video übertragen werden muß, bis das nächste Audiopacket dran kommen kann, halten wir die mittlere Datenrate nicht ein. Die Audiodaten müßten sofort nach Ende des Videopackets eintreten, wir übertragen praktisch Bytes, die der Videodekoder noch gar nicht braucht, aber dafür Verzögern wir das Eintreffen der Audiodaten. Von der Sicht des Audiodekoders aus haben wir die lokale Datenrate nicht eingehalten. Die Audiodaten kommen für ihn zu spät an.

Eine Möglichkeit, diesem Mißstand zu beugen, ist der **`startup sectors_delay`**. Damit teilen wir dem Multiplexer mit, er möge die ganz erste DTS/PTS nach obigem Schema errechnen und zusätzlich diese und alle weiteren TimeStamps um die Zeit verzögern, die soundsoviele zusätzliche Packets zum Eintreffen brauchen. Damit erreichen wir, daß zuallererst sowohl der Video-, als auch der Audio-Buffer ziemlich gefüllt werden, sodaß der Dekoder beim Abspielen der Daten keinen Bufferunterlauf kriegt, wenn Audio-/Videodaten packetweise verschickt werden.[22] [23]

- Die Abfrage nach einem **Very-verbose-mode** gibt an, wie detailliert beim Multiplexen Statusinformationen in der folgenden Tabelle dargestellt werden sollen.

Audio Sectors	Video Sectors	Padding Sectors	Bytes MPEG System File	Audio Buffer	Video Buffer
0	1	0	2324	4096	38687
0	2	0	4648	4096	36384
0	3	0	6972	4096	34081
0	4	0	9296	4096	31808
0	5	0	11620	4096	29505
0	6	0	13944	4096	27202
0	7	0	16268	4096	24929
0	8	0	18592	4096	22626
0	9	0	20916	4096	20323
0	10	0	23240	4096	18050
0	11	0	25564	4096	15747
0	12	0	27888	4096	13444
0	13	0	30212	4096	26173
0	14	0	32536	4096	23870
1	14	0	34860	1793	23870
1	15	0	37184	1793	21597
1	16	0	39508	2420	28502
1	17	0	41832	2420	26199
1	18	0	44156	3047	23926
1	19	0	46480	3047	23952
2	19	0	48804	1371	23952
2	20	0	51128	1998	21679
2	21	0	53452	1998	22067
2	22	0	55776	2625	19764

...

[22] Je kleiner die Sektorgröße gewählt wird, desto weniger notwendig ist diese Startverzögerung, desto mehr erhöht sich aber der Overhead, den die Pack/System/Packet-Header erzeugen.

[23] Eine einfache Möglichkeit ist, die Anzahl der Verzögerungspaketen ungefähr etwas weniger als die Video+Audio Buffergröße/Sektorgröße anzugeben. 8 bis 10 startup delay hat sich als sehr sinnvoll erwiesen.

```
|      27 |     174 |       3 |      474096 |   3059 |   8321 |
|      27 |     175 |       3 |      476420 |   3059 |   6048 |
|      27 |     176 |       3 |      478744 |   3685 |   7774 |
|      28 |     176 |       3 |      481068 |   1382 |   7774 |
|      28 |     177 |       3 |      483392 |   2009 |   5501 |
|      28 |     178 |       3 |      485716 |   2009 |   7383 |
|         |file  end|         |             |        |        |
|      28 |     179 |       3 |      488040 |   2636 |   6717 |
|file  end|         |         |             |        |        |
|      29 |     179 |       3 |      490364 |    334 |   6717 |
|      29 |     179 |       3 |      490368 |    334 |   6717 |
+---------+---------+---------+-------------+--------+--------+

Done, tmp files removed.
```

Mit dieser Tabelle kann genau überprüft werden, in welchem Verhältnis die einzelnen Datensektoren abgespeichert werden und wie sich im Target Decoder die Buffergrößen entsprechend modifizieren. Eine so detaillierte Ausgabe erreicht man mit Setzen des very verbose mode. Ansonsten wird immer nur die aktuelle Zeile ausgegeben und überschrieben. Damit kann man bei langen MPEG-Filmen überprüfen, wieviel Bytes und Sektoren ausgegeben wurden, ohne daß einem der Bildschirm unentwegt scrollt. Nur Informationen über *Time Out* (wenn Daten zu spät ankamen) oder *File End* werden in separaten Zeilen angegeben. Die Tabelle würde in der ersten Datenzeile während der Berechnung ständig aktualisiert werden und am Ende im Vergleich zur vorhergehenden etwas angenehmer folgendermaßen erscheinen:

```
+---------+---------+---------+-------------+--------+--------+
|  Audio  |  Video  | Padding | Bytes  MPEG | Audio  | Video  |
| Sectors | Sectors | Sectors | System File | Buffer | Buffer |
+---------+---------+---------+-------------+--------+--------+
|      28 |     178 |       3 |      485716 |   2009 |   7383 |
|         |file  end|         |             |        |        |
|      28 |     179 |       3 |      488040 |   2636 |   6717 |
|file  end|         |         |             |        |        |
|      29 |     179 |       3 |      490368 |    334 |   6717 |
+---------+---------+---------+-------------+--------+--------+
```

Die Multiplex-Hauptschleife, Vermischungsstrategie

Das Programm geht in der Multiplex-Hauptschleife folgendermaßen vor: Er verwaltet ständig eine **FIFO** (*First In - First Out*) verkettete Liste für beide Buffer des Dekoders. Jetzt wird anhand der Datenrate und der geschriebenen Bytes die momentane SCR Systemzeit errechnet. Der in den Buffer vorhandene Restplatz wird berechnet. Da davon ausgegangen wird, daß sowieso mehr Video- als Audiodaten übertragen werden müssen, wird in folgender Reihenfolge entschieden, ob Video, Audio oder Padding Packet geschickt werden soll.

- Es wird überprüft, ob im Videobuffer Platz für ein gesamtes Packet ist und ob das nächste Audiopacket auch erst im nächsten Packet übertragen werden kann, ohne daß es ein Time Out gibt.

- Falls nicht, ob im Audiobuffer Platz für ein Packet ist und ob das nächste Videopacket auch erst später übertragen werden kann, ohne daß es ein Time Out gibt.
- Falls nicht, ob im Audiobuffer Platz für ein Packet ist, obwohl die Audiodaten bereits Time Out verursachen (dieser Fall trifft bei schlecht gewählter Startverzögerung auf).
- Falls nicht, ob im Videobuffer Platz für ein Packet ist, obwohl die Videodaten bereits Time Out verursachen (dieser Fall trifft bei schlecht gewählter Startverzögerung auf).
- Schließlich, beide Buffer sind voll belegt, es muß ein Padding Packet geschickt werden, damit sich der Dekoder erholen, das Medium jedoch konstant Daten liefern kann.

Nachdem entschlossen worden ist, was für ein Sektor geschrieben werden muß, wird ein entsprechendes Packet, im richtigen Verhältnis auch Pack und System Header im Hauptspeicher generiert, die Daten aus den Grundstreams im Sektorbuffer eingelesen, die TimeStamps und sonstige Packetinformationen nach dem MPEG-Verfahren bitkodiert und der fertige Sektor ausgeschrieben. Das ganze wird iteriert, bis beide Grundströme zu Ende sind. Bevor das Programm nun beendet wird, werden noch die temporären Files, die Frame Typ, Länge und Time Informationen beinhalten, gelöscht und im System Stream noch die letzten vier Byte *ISO_11172_END* Code geschrieben.

3.4. Datenstrukturen

Ich werde hier nur einen groben Überblick über die wesentlichen, vom Programm verwendeten Typdefinitionen, Datenstrukturen und Konstantenvereinbarungen geben. Sie alle können in der Headerdatei `main.h` nachgelesen werden.

struct timecode_struc

```
typedef struct timecode_struc
{   unsigned long msb;
    unsigned long lsb;
} Timecode_struc;
```

Diese Struktur bildet die Speichereinheit für eine Timestamp. Die ISO Norm schriebt vor, daß SCR, DTS und PTS in einem 33 Bit Feld abgelegt werden. Funktionen, die für Generierung, Vergleich und Speicherung von Timestamps sorgen, finden wir in der Datei `timecode.c`.

struct vaunit_struc

```
typedef struct vaunit_struc
{   unsigned int length;
    unsigned int type;
    Timecode_struc DTS;
    Timecode_struc PTS;
} Vaunit_struc;
```

Mit der Funktion `get_info_video (inputstream.c)` arbeitet sich das Programm durch den MPEG Video Basis Strom und sammelt bzw. berechnet dabei Informationen (Länge in Bytes, Bildart sowie DTS und PTS) über jede einzelne Access Unit (ein Video Frame) in dieser Struktur.

struct aaunit_struc

```
typedef struct aaunit_struc
{   unsigned int length;
    Timecode_struc PTS;
} Aaunit_struc;
```

Eine weitere Funktion, `get_info_audeo (inputstream.c)`, arbeitet sich durch den MPEG Audio Basis Strom und sammelt bzw. berechnet dabei Informationen (Länge in Bytes, PTS) über jede einzelne Access Unit (ein Audio Frame) in dieser Struktur. Es sei erwähnt, daß hier, im Gegensatz zu einer Video Access Unit, die Unterscheidung zwischen DTS und PTS nicht mehr nötig ist. Es wird von der ISO Norm idealerweise davon ausgegangen, daß die Dekodierverzögerung für ein Audio Unit im System Target Decoder stets konstant und vernachlässigbar ist. Nur bei Video Frames muß dem Dekoder mitgeteilt werden, mit wieviel Vorsprung zur Präsentationszeit ein Frame dekodiert werden soll, damit dessen dekodierten Blöcke für andere P- oder B-Frames zur Verfügung stehen können.

struct video_struc

```
typedef struct video_struc
{   unsigned int stream_length  ;    /* Länge in Bytes */
    unsigned int num_sequence   ;    /* Anzahl Sequence Start Syncwords */
    unsigned int num_seq_end    ;    /* Anzahl Sequence End   Syncwords */
    unsigned int num_pictures   ;    /* Anzahl Picture Start  Syncwords */
    unsigned int num_groups     ;    /* Anzahl Group Start    Syncwords */
    unsigned int num_frames[4]  ;    /* Anzahl Frames [I,P,B,D] */
    unsigned int avg_frames[4]  ;    /* Länge  Frames [I,P,B,D] */

    unsigned int horizontal_size;    /* Horizontalgröße Video */
    unsigned int vertical_size  ;    /* Vertikalgröße Video */
    unsigned int aspect_ratio   ;    /* Breite/Höhe Verhältnis */
    unsigned int picture_rate   ;    /* Bildrate */
    unsigned int bit_rate       ;    /* Bitrate */
    unsigned int comp_bit_rate  ;    /* Berechnete Bitrate */
    unsigned int vbv_buffer_size;    /* vbv Buffer Größe */
    unsigned int CSPF           ;    /* handelt es sich um ein Constrained System
                                        Parameter File? */
} Video_struc;
```

In dieser Struktur werden alle weiteren Informationen, die einen MPEG Video Stream charakterisieren, festgehalten. Man beachte, wie auch die durchschnittliche Länge eines jeden Frametyps festgehalten wird, und wie auch eine anhand der Daten berechnete Bitrate notiert wird. Es befinden sich nämlich durchaus Video Streams im Umlauf, die dafür gedacht sind, auf einem Rechner mit Software-MPEG-Dekomprimierung dargestellt zu werden. Da übliche Workstations Fullsizevideos z.Zt. softwaremäßig nicht in Echtzeit Dekomprimieren und Anzeigen können, werden dort geringe Frameraten des öfteren in Kauf genommen. Solche Videos haben dann auch oft falsche Bitraten in den Headers notiert, auf die sich der Multiplexer jedoch verlassen müßte. Zur Verifikation wird darum anhand von Streamlänge, Bildrate und Anzahl von Frames eine Bitrate berechnet.

struct audio_struc

```
typedef struct audio_struc
{   unsigned int stream_length  ;    /* Länge in Bytes */
    unsigned int num_syncword   ;    /* Anzahl Syncwords */
    unsigned int num_frames[2]  ;    /* Anzahl Frames [Bytelänge x, x+1] */
    unsigned int size_frames[2] ;    /* Länge  Frames [x, x+1] */

    unsigned int layer          ;    /* Layer 1,2,3 */
    unsigned int protection     ;    /* befinden sich CRC Summen im Stream ? */
    unsigned int bit_rate       ;    /* Bitrate */
    unsigned int frequency      ;    /* Sampling Frequenz */
    unsigned int mode           ;    /* Mode [mono,stereo,joint stereo...] */
    unsigned int mode_extension ;    /* Mode Extension */
    unsigned int copyright      ;    /* Copyright-geschützt ? */
    unsigned int original_copy  ;    /* ist es Original oder eine erste Kopie ? */
    unsigned int emphasis       ;    /* Emphasis-Modell */
} Audio_struc;
```

In dieser Struktur werden alle weiteren Informationen, die einen MPEG Audio Stream charakterisieren, festgehalten. Wie wir bereits in Kapitel 2.3 gelernt haben, sind die einzelnen Audio Access Units immer gleich lang, sie haben entweder eine berechenbare Länge x oder x+1. Die Anzahl der Frames und deren Länge wird auch hier getrennt festgehalten.

struct sector_struc

```
typedef struct sector_struc
{   unsigned char  buf [MAX_SECTOR_SIZE]  ;    /* Sektorbuffer in RAM */
    unsigned int   length_of_sector       ;    /* Effektive Länge Sektors */
    unsigned int   length_of_packet_data  ;    /* Eigentliche Packetdaten */
    Timecode_struc TS                     ;    /* Früheste TimeStamp (DTS, PTS) */
} Sector_struc;
```

Mit dieser Struktur kann im Speicher ein kompletter Sektor aufgebaut werden, der dann auf Hintergrundspeicher dem System Stream angehängt werden kann. Je nachdem, ob im Sektor ein Video- oder Audiopacket vorkommt, wird die jeweils früheste TimeStamp (DTS oder PTS) notiert. Ein Sektor kann entweder nur aus einem Packet bestehen, oder aus Pack und Packet, oder aus Pack, Sys Header und Packet.

struct pack_struc

```
typedef struct pack_struc
{   unsigned char  buf [PACK_HEADER_SIZE]  ;   /* Pack Header Buffer in RAM */
    Timecode_struc SCR                     ;   /* SCR des Packs */
} Pack_struc;
```

struct sys_header_struc

```
typedef struct sys_header_struc
{   unsigned char  buf [SYS_HEADER_SIZE]   ;   /* Sys Header Buffer in RAM */
} Sys_header_struc;
```

Diese zwei Strukturen sind eigentlich nur Kurzzeitbuffer für die Sektorstruktur. Falls in einem Sektor Pack Header oder Sys Header eingeschlossen werden sollen, werden aus diesen Buffern die fertigen Daten kopiert.

struct buffer_queue

```
typedef struct buffer_queue
{   unsigned int    size ;
    Timecode_struc  DTS  ;
    struct buffer_queue *next ;
} Buffer_queue;
```

Mit dieser Struktur erhalten wir einen Eintrag für die verkettete FIFO-Liste, die die Buffer des System Target Decoders simulieren soll. Jede einzelne Access Unit wird hier festgehalten und mit der DTS notiert. Den Anker der verketteten Liste erhalten wir mit der Struktur

struct buffer_struc

```
typedef struct buffer_queue
{   unsigned int    max_size ;
    Buffer_queue    *first   ;
} Buffer_struc;
```

Alle Funktionen, die sich um das Ein-/Austragen aus der FIFO-Liste, Restplatzberechnung im STD Buffer, Initialisieren der Listen kümmern, können in der Datei `buffer.c` überprüft werden.

4. Eigenschaften von MPEG/Systems

4.1. Problemfälle

Positionieren im gemultiplexten Datenstrom

Es gibt einige Probleme im Zusammenhang mit MPEG/Systems und dem *Video On Demand Service*. So sind im Prinzip der System Target Decoder und der MPEG Stream als solcher so konzipiert, daß die Abarbeitung des Stroms immer vom Anfang an beginnt. Will der User z.B. die erste halbe Stunde des Films überspringen, so muß es - sofern das implementiert ist - dem Dekoder mitgeteilt werden, der sich darum kümmert, vom Anfang des Streams an die einzelnen Headerinfo's einzulesen, um seine Buffergrößen so anzupassen, bis ein geeigneter Startpunkt gefunden ist. Der Dekoder hat somit immer Zugriff auf die gesamte Datei. Bei *Video On Demand* und *Broadcast* Applikationen entsteht die Problematik, daß man wirklich mitten in einer gigabytegroßen Datei aufsetzen will. Wollen wir jetzt z.B. auf eine bestimmte Sekunde positionieren, so müssen wir dem Dekoder einen korrekten MPEG-Anfang simulieren. Das kann theoretischerweise folgendermaßen geschehen:

1. Beginne mit Dateianfang. Lese Header ein, suche dann anhand vom PTS in den einzelnen Packets ein geeignetes Startpacket aus. Wurde ein geeignetes PTS gefunden, muß festgestellt werden, ob der Dekoder die Access Unit auch dekodieren kann, sprich ob es ein I oder P picture ist, oder - noch besser - eine Group of Pictures, die dann durch Setzen des *Broken Link Flags* als nicht mehr mit der vorherigen zusammenhängend markiert werden kann.
2. Anhand des hoffentlich gefundenen, passenden Frames, muß jetzt ein System Header generiert werden, der vorher geschickt werden muß. Dieser beginnt wiederum mit einem Pack Header, wo auch eine entsprechende SCR TimeStamp berechnet werden muß. Die Clock Reference muß vom nächsten Pack aus nach Anzahl der zu übermittelnden Daten und Datenrate zurückgerechnet werden, damit für den Dekoder ein Korrelieren mit dem DTS des ersten Frames eintritt. Es muss jedoch beachtet werden, daß i.d.R. nur das ganz erste Packet mit einem Picture Start übereinstimmt. Bei allen anderen Packets haben wir immer noch die Restbytes des vorhergehenden Frames. Der Beginn des Frames, auf dem sich die DTS auch bezieht, kann irgendwo mitten im Packet eintreten. Diese vorher stehenden Bytes machen nur dann Sinn, wenn der Dekoder auch das vorhergehende Packet erhalten hat und dessen Bytes im Buffer hat. Dieser jetzt sinnlose Overhead müß demnach entfernt werden, die Packetgröße neu berechnet werden, SCR Felder anhand des nächsten Packets zurückgerechnet werden, damit die Clock Reference Timestamps mit der geänderten Packetlänge übereinstimmen. Das alles erfordert doch einen erheblichen Aufwand für den Videoserver, da dieser bei einem ihm vorliegenden MPEG/System Stream ja gar nicht mehr weiß, wo innerhalb des Packets der Frameanfang ist. Es müßte entweder eine Tabelle für jeden Film abgelegt werden, oder der Server muß sich durch das gewählte Packet scannen und den Picture Header identifizieren.

Die folgende Skizzen erläutert diesen Sachverhalt:

MPEG/SYSTEM Stream, auf Medium vorliegend

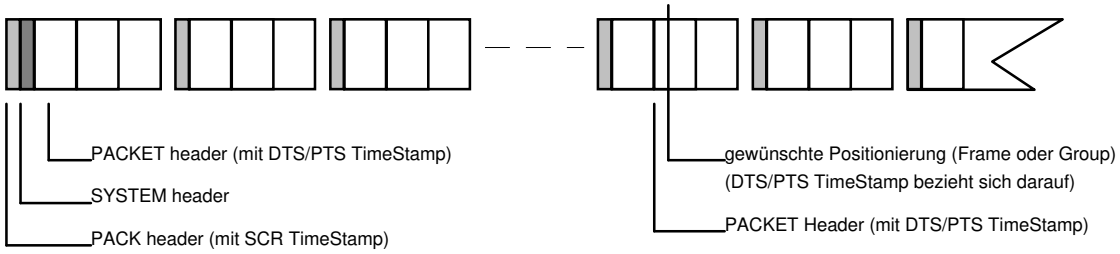
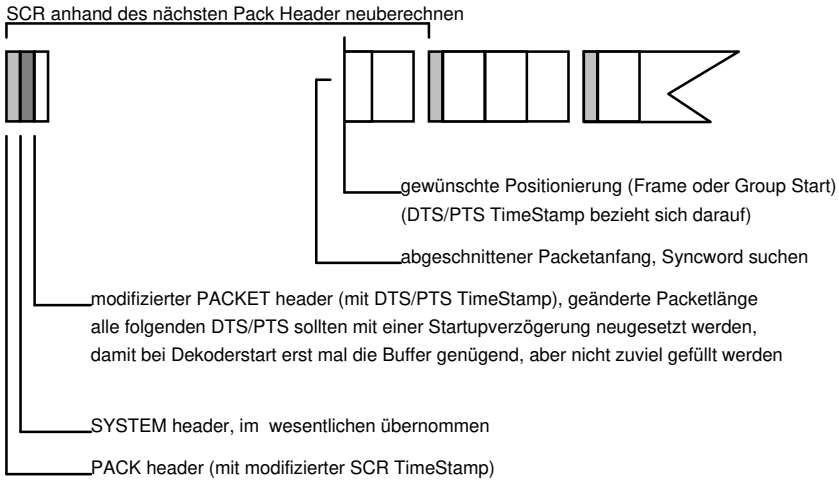


Abb. 7 Auf dem Medium bereits vorliegender System Stream

MPEG/SYSTEM Stream, vom Server zu erzeugen



Das alles garantiert immer noch nicht, daß kein Bufferunterlauf oder Bufferüberlauf im neuen Stream passiert!

Abb.8 Generierung eines neuen korrekten System Stream

Bufferverhältnisse beim System Target Decoder

Eine weitere Schwierigkeit ist, daß der erzeugte Stream vom Koder nach iterativ berechneten Bufferverhältnissen erzeugt wird. So könnten z.B. erst mal 13 Video-Packets geschickt werden, dann 1 Audio Packet, dann 6 Video, 1 Audio, 7 Video usw. Dazwischen immer wieder Padding Packets, um Bufferüberläufe zu verhindern. Wird jetzt irgendwo mitten im Strom aufgesetzt, so wird der Dekoder mit größter Wahrscheinlichkeit Schwierigkeiten mit den Bufferverhältnissen kriegen.

Die Zusammensetzung des gesamten System Streams wird vom Enkoder oder Multiplexer vom Anfang an gestaltet. Jedes Packet im Stream hat genau eine beim Multiplexen ihm implizit zugewiesene Reihenfolge und ist auf die Verhältnisse im Stream und in den Buffern des STD vom Anfang an angewiesen. Wenn beim Schreiben des Packets davon ausgegangen wurde, daß bereits x Bytes im Buffer waren, die zu einer bestimmten Zeit wieder freigegeben werden, so dürfen dort tatsächlich weder weniger noch mehr Bytes vorhanden sein. Auch kleine Abweichungen werden unweigerlich Schwierigkeiten bringen, da es zu einem beliebigen Zeitpunkt durchaus vorkommen

kann, daß ein Packet ursprünglich den Buffer genau füllen oder genau leeren würde. Waren jetzt mehr oder weniger Bytes darin vorhanden, so entsteht ein Bufferunter- /-überlauf.

Was passiert, wenn der Enddekoder einfach mitten im Stream eingeschalten wird, ist auch nicht definiert. Ein simpler STD, wie in MPEG definiert, wäre nicht in der Lage zu arbeiten. Er ist auf einen korrekten MPEG Strom samt Headern vom Anfang an angewiesen, es ist nicht feststellbar, was bei einem fehlerhaften Stromanfang passiert. Diese Feature wird von MPEG nicht vorgeschrieben, muß gegebenenfalls vom Dekoderhersteller vorgesehen und entsprechend implementiert werden. Vorkehrungen, wie das Sicherstellen, daß Audiodaten nicht als Syncwörter erkannt werden, müssen gegeben werden; außerdem muß der Dekoder mit fehlenden Audio- bzw. Videodaten in den Buffern, eventuell nicht dekodierbaren Frames und Ähnlichem fertig werden können.

Falls aber vom Hardwarehersteller ein solcher erweiterter Dekoder geliefert wird (was ja durchaus denkbar ist), entfällt auch die ganze Vorarbeit, die sonst der *Video On Demand* Server liefern müßte. Er kann anhand der mittleren Datenrate und der Pack/Packetgröße schon mal ein grobes *Seek* auf die Datei machen und einfach anfangen, Daten zu schaufeln. Der Enddekoder muß dann nur so intelligent implementiert sein, daß er selbstständig so schnell wie möglich einen mehr oder weniger definierten gültigen Anfangsstatus ausfindig macht und sich selber um das Verwerfen der davor auftretenden Daten kümmert.

Standbildfunktionalität

Gleiches gilt für Standbild: Bei *Video On Demand* ist es denkbar, daß der Anwender nur mit dem Server kommuniziert und dem mitteilt, daß jetzt Pause sei. Pause ist in einem MPEG-Strom natürlich nicht definiert, es ist eine Eigenschaft, die ausschließlich vom Dekoder zur Verfügung gestellt werden kann. Der Server könnte zwar ein *ISO_11172_END*-Syncword schicken, damit weiß der Dekoder, daß sein Job fertig ist. Würden einfach keine Daten mehr geschickt werden, würde der Dekoder einen Fehler im Medium vermuten und einen Bufferunterlauf melden. *ISO_11172_END* schicken ist an und für sich kein Problem, aber sobald die Pause aufgehoben werden soll, also ein Neustart simuliert werden muß, treten exakt die selben Probleme auf wie oben. Der Enddekoder hat seine Puffer geleert und hat keine Ahnung, was er mit den ankommenden Daten anfangen soll.

Fast Playback / Rewind

Fast Playback/Rewind ist ebenfalls keine Funktion, die der MPEG-Server als solcher liefern kann. Er müßte sonst den Strom dekodieren, einzelne Bilder und Tondaten auswählen und diese neu zu einem MPEG/System Strom kodieren und schicken. Die Möglichkeit für Fast Forward ist im Medium/Dekoder zu implementieren. Der Dekoder könnte ein schnelleres Anliefern der Daten verlangen, soweit dies vom Medium unterstützt werden kann, und nur jedes x-te Frame verarbeiten. Auch das ist aber vom MPEG Standard nicht festgelegt und eine reine Dekodereigenschaft. Ähnliches gilt für Rücklauf, wobei dort zusätzlich die Schwierigkeiten beim Dekodieren des Videostroms eintreten. Servermäßig ist hier kaum eine Möglichkeit denkbar.

4.2. Rückblick

Besonders in Beziehung zu Applikationen im Rahmen eines *Video On Demand* Servers kann eigentlich der *ISO 11172-1 SYSTEM* Stream nicht als praktikables Systemformat empfohlen werden. Er bringt für die Interaktion mit dem Client mehr Nachteile als Vorteile. Getrennte MPEG/Audio und MPEG/Video Files sind schon deutlich angebrachter, da sie ihre zugrundeliegende Struktur, die Bild- und Framereihenfolge nicht verheimlichen. Es kann viel einfacher mitten in einem Stream aufgesetzt werden, ohne daß Packetdaten aus einer anderen Unit entfernt werden müssen, und die Schwierigkeiten mit Buffer- und Multiplexmanagement entfallen völlig. Zu Demonstrationszwecken und zur Realisierung eines geschlossenen Systems aus einem einfachen Medium und Dekoder ist das Verfahren aber sehr wohl geeignet. Das Hauptanwendungsgebiet liegt wohl bei fertigen MPEG-Filmen auf CD, die mit einem Endgerät abgespielt werden können, das die alleinige Kontrolle über das Medium hat.

Anhang A, kommentierter Videostrom

Listing file: flower.hex.

```
------------------------------------------------------------------------
Offset: 0000000
------------------------------------------------------------------------
00000000|00000000|00000001|10110011|                  seq.header code
                                 00010110|0000     hor. size
ver. size                                      0000|11110000
pel.aspect ratio                                           |0010
picture rate                                                    0100|
00000010|01110001|00                                  bit rate
                    1                                marker bit
                     00000|01000                     vlb buffer size
                                1                    CSPF
                                 1                   load IQM
                                  0|00010000|00100000|00100110|00101100|
------------------------------------------------------------------------
Offset: 0000020 .. (Intra Quantizer Matr.) ..     Offset: 0000100
------------------------------------------------------------------------
01001100|01011100|01110000|10001010|00110110|00111010|01000110|01001100|
01011100|01110000|10001010|1010011                   IQ Matrix
                                 1                   load NIQM
                                  |00010000|00010001|00010010|00010011|
------------------------------------------------------------------------
Offset: 0000120 .. (Non Intra Quantizer Matr.) .. Offset: 0000200
------------------------------------------------------------------------
00011011|00011100|00011110|00011111|00010111|00011000|00011001|00011011|
00011100|00011110|00011111|00100001|                  NIQM
group start code                   00000000|00000000|00000001|10111000|
------------------------------------------------------------------------
Offset: 0000220
------------------------------------------------------------------------
0                                                    drop_frame_flag
 00000                                               time_code_hours
      00|0000                                        time_code_minute
             1                                       marker_bit
              000|000                                time_code_second
                     00000|0                         time_code_pictur
                            1                        closed GOP
                             0                       broken link
                              00000|                 next_start_code
picture start code                 00000000|00000000|00000001|00000000|
00000000|00                                         temporal referer
           001                                       picture coding I
              000|00000000|00000                     VBV delay
                                0                    extra bit pictur
                                 00|                 next_start_code
slice start codes                  00000000|00000000|00000001|00000001|
+ slice_vertical_position                                    ^^^^^^^^
------------------------------------------------------------------------
Offset: 0000240
------------------------------------------------------------------------
01000                                                quantizer scale
     0                                               no extra bit sli
      1                                              macro adr incr
```

```
       1|                                                macro type
         1110                                            dct  etc...
            1110|01110100|10100000|00001001|01100000|10111111|10001000|

... quantizer scale ...

-------------------------------------------------------------------------
Offset: 0002220
-------------------------------------------------------------------------
11010100|10010010|01000010|01000100|01110010|00110100|
slice start code                                        00000000|00000000|
00000001|00000010                                      slice start code
                |00111011|11110100|01111011|01110100|11111101|10110110|
... quantizer scale ...

-------------------------------------------------------------------------
Offset: 0004060
-------------------------------------------------------------------------
01011110|00100011|11010100|10011110|00010000|10010001|00011100|10001101|
00000000|
         00000000|00000000|00000001|00000011|         slice start code
                                              00101011|11110100|11100101|
... quantizer scale ...

... usw ...

-------------------------------------------------------------------------
Offset: 0042560
-------------------------------------------------------------------------
00000000|00000000|00000001|00000000|                   picture start code
                                  00000000|11         temporal refere
                                            011       picture coding B
VBV delay                                      000|00000000|00000
extra bit picture                                                 0
next start code...                                                 00|
10010000|
         00000000|00000000|00000001|00000001|         slice start code
                                              01001010|00111110|11001111|

... usw ...
```

Anhang B, kommentierter Audiostrom

Listing file: test.hex.

```
--------------------------------------------------------------------------------
Offset: 0000000
--------------------------------------------------------------------------------
1111 1111|1111                                      syncword
              1                                     ID
               10                                   Layer
                 0                                  protection_bit
                  |0100                            bitrate_index
                        01                          sampling_frequency
                          0                         padding_bit
                           0                        private_bit
                            |00                     mode
                               00                   mode_extension
                                  0                 copyright
                                   0                original/copy
                                    00              emphasis
                                       |0100 1111   protection=0->CRC Sum

Audio PCM Daten                                  |0101 1111|0100 0100|0011 0011|
0010 0100|1001 0010|0100 1001|0010 0100|1001 1111|1111 1010|1010 1010|0000 1010|

...
```

Anhang C, kommentierter Systemstrom

```
************************************
*    MPEG  Multiplex  Information    *
*    Christoph Moar,   March 1994    *
*    SIEMENS   ZFE   ST   SN   11    *
************************************

Scanning MPEG multiplexed stream nfl.mpg for header informations.
Pack / Packet ratio  1 : 1.

pack_start_code with SCR TimeStamp : 000003142
     mux_rate field  : 3486
system_header_start_code
     header length   : 12
     rate_bound      : 3486
     audio_bound     : 1
     fixed_flag      : 1
     CSPS_flag       : 1
     audio_lock      : 1
     video_lock      : 1
     video_bound     : 1
     STREAM ID       : 224
     STD_buffer_scale: 0
     STD_buffer_size : 40
     STREAM ID       : 192
     STD_buffer_scale: 0
     STD_buffer_size : 32
     VIDEO packet_start_code stream: 0      packet_length: 2288
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     STD_buffer_scale: 0
     STD_buffer_size : 40
     PTS: 000009d2c
     DTS: 000008e86

pack_start_code with SCR TimeStamp : 0000035f2
     mux_rate field  : 3486
     VIDEO packet_start_code stream: 0      packet_length: 2306
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     stuffing Byte detected
     STD_buffer_scale: 0
     STD_buffer_size : 40

next pack start code...
```

Literaturverzeichnis

Index

A
Abtastrate, 7; 23
Access Unit, 11; 14; 15; 20; 21; 26
AT&T, 16
ATM, 17
Audio Access Units, 11
Audio_struc, 21
Audiokodierung, 7; 10; 18
Audioqualität, 16
Auflösung, 8; 9; 10; 16; 17

B
B-Frames, 5
Berkeley-Coder, 3
Bitrate, 5; 13; 20
bitstream.c, 18
bitstream.h, 18
Block Layer, 7
Broadcast, 7; 12; 26
Broken Link Flag, 6; 26
Buffer, 13; 15; 16; 20; 21; 22; 24; 25; 26; 27; 28; 29
Buffergrößen, 22
Buffermanagement, 12; 13
Bufferüberlauf, 15; 28
Bufferunterlauf, 15; 28
Bufferverhältnisse, 23; 28

C
C-Cube, 16
CCIR 601, 17
CD, 29
CD Audio, 8
CD ROM, 16; 22
CD-DA, 7
CD-I, 16
CGEG, 3
CGT, 16
Compact Disc, 7
Compiler, 18
Constrained System Parameter File, 20
Constrained System Parameter Set, 22
CRC-Checksumme, 11
CSPF, 20
CSPS, 22
Cypress, 16

D
D-Frames, 5; 17
DAT, 7
Datenrate, 3; 11; 12; 15; 16; 17; 22; 23; 25; 28
Datenreduzierung, 10
DCT, 4; 7; 17
Decoding Time Stamp, 6; 14; 15; 20; 23; 24; 27
Dekodierverzögerung, 14
Dialogdienst, 12
Digital Audio Tape, 7
Digital Storage Media Command and Control, 17
Digitalisiergerät, 10
Diskrete Cosinus Transformation, 4; 7; 17
Draft International Standard, 17
DSM-CC, 17
DTS, 14; 15; 20; 23; 27

E
Echtzeitkompression, 17
Echtzeitkomprimierung, 10
Entropiekodierung, 5; 8

F
Fast Fourier Transformation, 7; 10
Fast Playback, 29
FFT, 7; 10
FIFO, 25
File End, 25
Filterbank, 8
First In - First Out, 25
Frames, 11
Frequenzbänder, 7
Frequenzenbänder, 8
Frequenzspektrum, 10
Frequenztransformation, 7

G
Group of Pictures, 5; 6; 26
Group of Pictures Layer, 6

H
H.261, 4
Headerinformation, 12
Huffmann, 5; 7; 8; 9; 10

I
I-Frames, 5; 6
International Standard, 18
ISO/IEC
  11172-1, 3; 12
  11172-2, 3
  11172-3, 3
  JTC1/SC2/WG8, 3
  JTC1/SC29/WG11, 3
ISO_11172_END, 26; 28

J
JBIG, 3
Joint Stereo Mode, 10
JPEG, 3; 4

K
Kommunikationsnetze, 12
Kompression
  asymmetrische, 4
  symmetrische, 4
Kompressionsfaktor, 16
Kompressionsrate, 8
Koordination, 12

L
Layer, 3; 5; 6; 7; 8; 10; 11; 14; 21

M
Macroblock Layer, 7
Markierungsbits, 7
Maskierungseffekt, 9; 10
Maskierungsphänomen, 10
Maskierungswert, 10
Medium, 16
MPEG, 3
  Audio, 3; 7; 10
  Standards, 3
  Systems, 7
  Video, 3; 4; 6
MPEG Strom, 13
MPEG/System, 1; 3; 4; 6; 7; 13; 16; 18; 21; 26; 27; 29
MPEG2, 16
MPEG2 Extensions, 17
mplex, 19
Multiplex-Hauptschleife, 25
multiplex.c, 18
multiplex.h, 18

N
NBC, 18
Non Backwards Compatible, 18
Nyquist, 9

O
Offlinekompression, 17
Overhead, 22; 27

P
P-Frames, 5
Pack, 22
Pack Header, 26
Packet, 14; 22; 25
Packets pro Pack, 15
Packs, 12; 15; 16
packs-to-packets-ratio, 22
Padding Bytes, 16
Padding Packet, 16; 25
Pause, 28
PCM, 8
Philips, 16
Picture Header, 6
Picture Layer, 6
Picture Periods, 14
Picture Rate, 20; 23
postmasking, 10
Präsentationsreihenfolge, 14
premasking, 10
Presentation Time Stamp, 6; 14; 15; 20; 23; 24; 26
psychoakustischen Modell, 8; 10
psychoakustisches Modell, 7
PTS, 14; 20; 23; 26
Pulse Code Modulation, 8

Q
Quantisierung, 7; 8; 10; 17

R
Rauschen, 9
Rauschpegel, 7
Rewind, 29

S
Sample Rate, 23
SCR, 15; 23; 25; 26
Seek, 28
Sektor, 22
Sektoren, 25
Sekundärspeichermedium, 12
Sequence Header, 6
Sequence Layer, 5
Skalierung
  Bittiefe, 17
  Datenrate, 17
  räumliche, 17
Slice Layer, 7
Slot, 11
Sourcen, 18
Speichereinheit, 22
Spektralanteile, 7
Spektralkomponenten, 10
Standbild, 28
startup sectors_delay, 24
Statusinformationen, 24
STD, 13; 14; 16; 22; 26; 28
Stereo, 10
Strategie, 18
subbands, 10
Synchronisation, 13; 15
Synchronisationswörter, 7
Syncword, 7; 12
System Clock Reference, 15
System Header, 16; 26
System Target Decoder, 22; 26
Systemdefinition, 4; 12

T
temporäre Datei, 20; 21
temporärer Plattenspeicher, 21
Thomas Hahn, 4
Time Out, 25
time slice, 10
TimeStamp, 13; 14; 15; 22; 23; 24; 26; 27
Transformation, 4

V
variable-length-Codes, 7
Vbv buffer size, 22
verkettete Liste, 25
verlustlose Datenkompression, 9
Vermischungsstrategie, 18
Verteilungsanwendung, 12
Very-verbose-mode, 24
VHS, 16
Video On Demand, 26; 28; 29
Video_struc, 20
Videoserver, 17

W
Working Draft, 18

Z
Zeitscheibe, 10