# IB FT-BS-2017-133

## Dokumentation eines Verfahrens zur Übertragung von Videodaten mittels PCM

**Institutsbericht**

Rainer Holland
Institut für Flugsystemtechnik

DLR Deutsches Zentrum für Luft- und Raumfahrt

# Inhalt

# 1. Einleitung

Das hier vorgestellte Verfahren dient zur Übertragung von Videodaten mittels PCM.

Das Ziel war die Einbindung von Videodaten aus dem Hubschrauber in den PCM-Rahmen für die relevanten Flugzustandsgrößen, um so eine einfache zeitliche Zuordnung zwischen beiden Datentypen zu haben. Damit wird weiterhin auch nur ein PCM-moduliertes Hochfrequenzsignal vom Hubschrauber zum Boden übertragen, welches mit der bestehenden Hardwarekonfiguration verarbeitet werden kann. Andere Lösungen sehen zum Beispiel die getrennte Übertragung (Modulierung) von Flugzustandsgrößen und Videodaten mittels Burst-PCM vor.

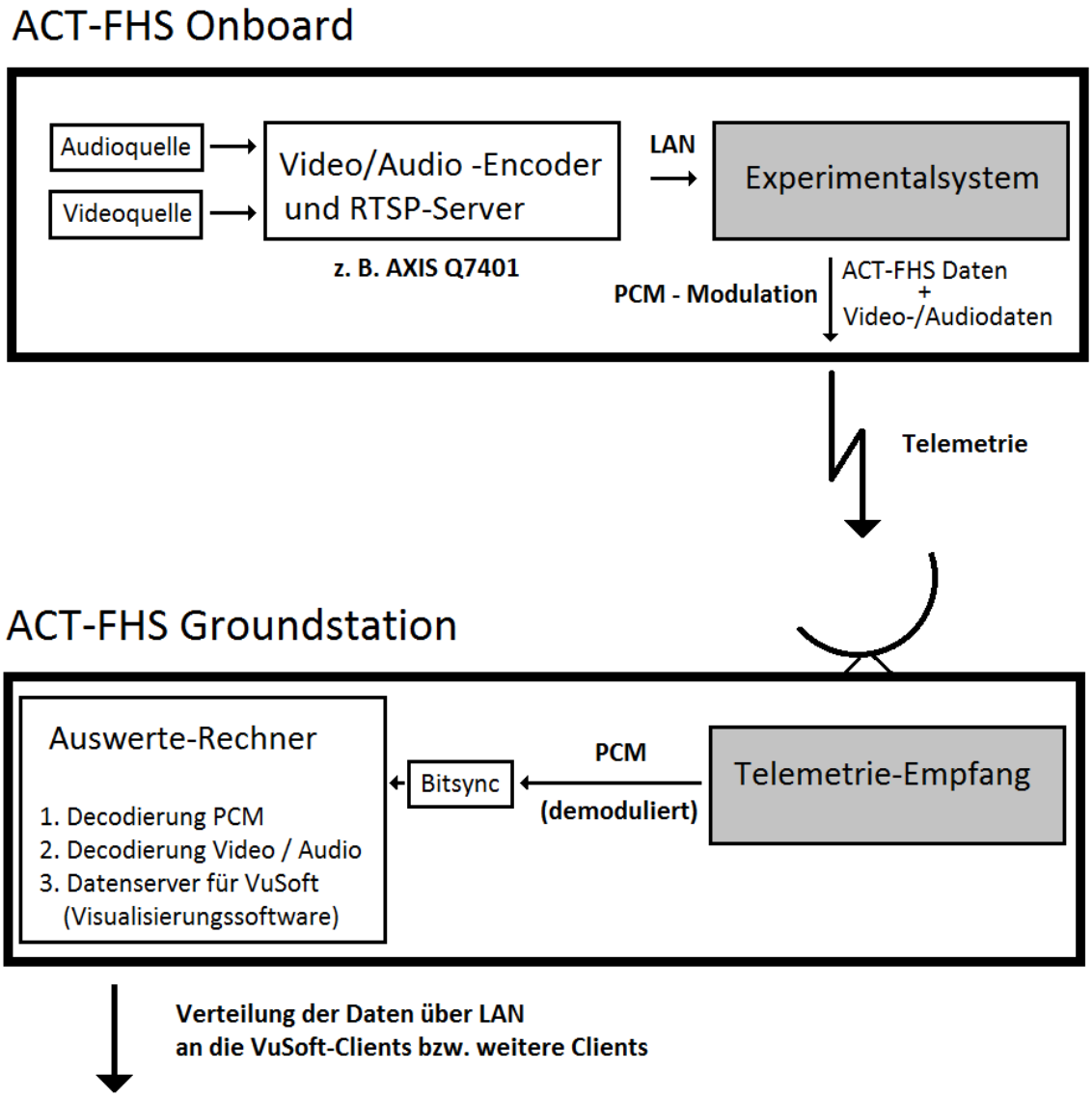


**Bild 1. Schematische Darstellung der Datenübertragung vom Hubschrauber zum Boden**

Bild 1 skizziert die realisierte Datenübertragung.

## 2. Vorüberlegungen zu den Videodaten

Unkomprimierte Videodaten haben ein großes Datenvolumen und beanspruchen für sich eine große Anzahl von Worten im PCM-Rahmen bzw. eine hohe Übertragungsbandbreite.

Als Beispiel ist ein $352 * 288$ Pixel großes Farbbild gewählt. Jedes Pixel benötigt 3 Bytes (R-G-B) für die Farbdarstellung. Damit benötigt jedes Bild $352 * 288 * 3 = 304128$ Bytes. Soll das Geschehen im Hubschrauber mit 10 Bildern/sec aufgelöst werden, so beträgt die dafür notwendige Bandbreite $304128 * 10$ Bytes/sec = $304128 * 10 * 8$ Bits/sec.

Für eine praktische Nutzung muß also eine Kompression der Videodaten erfolgen. Ein bekanntes Verfahren ist der JPEG-Algorithmus für Einzelbilder bzw. MOTIONJPEG (MJPEG) für eine Folge von Einzelbildern. Trotz der erreichbaren Kompressionsfaktoren von bis 12-14, bei relativ guter Bildqualität ist die Datenmenge immer noch zu hoch.

Eine noch höhere Kompression für eine Abfolge von Einzelbildern läßt sich über den H264-Algortihmus erreichen. Hierbei werden Einzelbilder in sogenannten GOPs (**G**roup **o**f **P**ictures) zusammengefaßt. Ein GOP beginnt mit einem I-Frame, der den gesamten Inhalt des Bildes in komprimierter Form enthält. Es folgen dann P-Frames, die nur Änderungen gegenüber dem vorherigen Bild enthalten. Die Größe eines GOPs sollte so gewählt werden, daß bei Übertragungsstörungen durch die Telemetrie in der Bildfolge häufig ein I-Frame erscheint. Beim I-Frame wird zur Bildrückgewinnung keine Information von vorherigen Bildern benötigt. Da I-Frames aber mehr Daten benötigen als P-Frames, ist wegen der Forderung nach kleiner Videobandbreite hier ein Kompromiß zu finden

Für die Wandlung und Komprimierung von Bilddaten aus einer Kamera wird ein Encoder der Firma AXIS - *AXIS Q7401* (s. Bild 2) – verwandt. Die Beschreibung der vollen Funktionalität des AXIS-Encoders ist den Datenblättern der Firma AXIS (*www.axis.com*) zu entnehmen. Hier wird nur die Funktion als RTSP-Server für Videodaten betrachtet.


**Bild 2. AXIS-Encoder AXIS Q7401**

# 3. Realisierung

## *3.1. Übersicht*

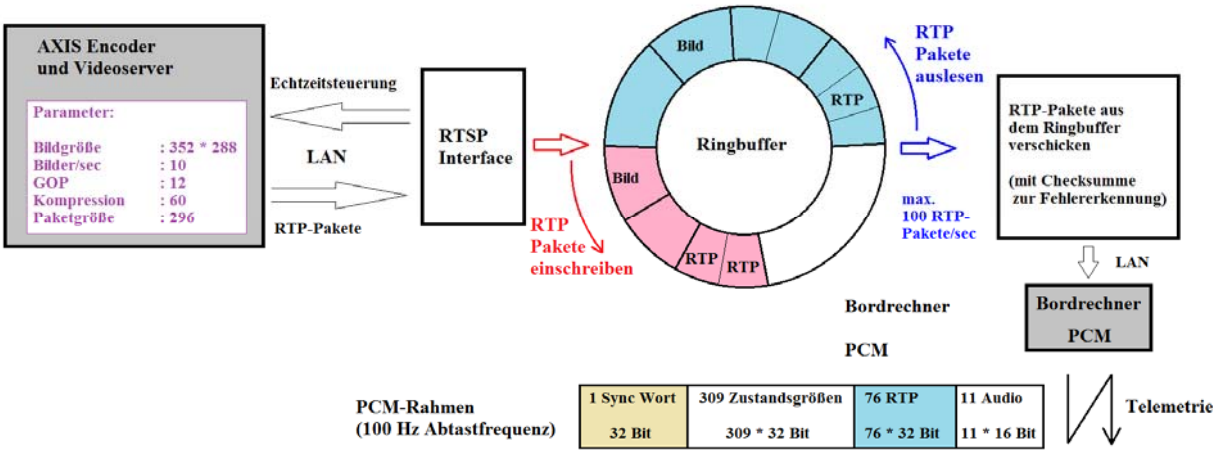


**Bild 3. Bordseitige Implementierung**

Bild 3 zeigt die Implementierung an Bord des Hubschraubers. In einem dort erzeugten PCM-Rahmen werden 304 Bytes (76 * 32 Bit-Worte) zur Übertragung der Videoinformation genutzt. Die Abtastfrequenz beträgt 100 Hz. Damit lassen sich maximal 100 Videodaten-Pakete zu je 304 Bytes pro Sekunde übertragen.

Die RTP (**R**ealtime **T**ransport **P**rotocol)-Pakete (=H264 komprimierte Videodaten) liefert ein Encoder der Firma AXIS - *AXIS Q7401*. Dazu wird ein RTSP (**R**eal-**T**ime **S**treaming **P**rotocol) verwendet. Hierüber teilt die steuernde Client-Software dem Encoder die gewünschten Parameter, wie Bildgröße, Bildfrequenz, GOP-Länge Kompression und Größe eines RTP-Paketes mit. Des weiteren ist über das RTSP-Protokoll auch das Echtzeitstreaming spezifiziert.

Die RTP-Pakete werden fortlaufend in einen Zwischenspeicher (Ringbuffer) geschrieben, der alle 10 msec ausgelesen wird. Die ausgelesenen Daten werden mit einer Kennung und einer Checksumme versehen und in den PCM-Rahmen eingefügt.

Die verlustfreie Übertragung der Videodaten ist solange gewährleistet, wie das Auslesen schneller erfolgt als das Einschreiben. Anders formuliert, solange weniger als 100 RTP-Pakete pro Sekunde vom Encoder kommen.

Da der PCM-Rahmen alle 10 msec übertragen wird, kommt es zum mehrfachen Versenden identischer RTP-Pakete, wenn der Encoder weniger als 100 pro Sekunde von diesen verschickt. Dies ist aber kein Fehler, sondern muß nur bei der Auswertung berücksichtigt werden.

Bild 4. zeigt die Softwareverarbeitung nach dem Empfang des PCM-Rahmens am Boden.

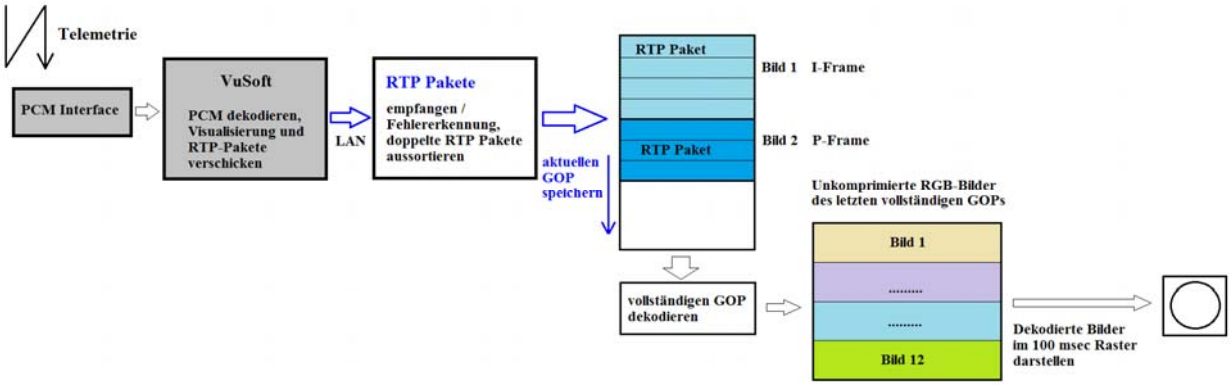


**Bild 4. Bodenseitige Implementierung**

Die Decodierung des PCM-Rahmens erfolgt durch die Software „VuSoft“, welche auch die empfangenen RTP-Pakete per LAN an einen Rechner bzw. eine Software zur weiteren Verarbeitung verschickt.
Die empfangenen RTP-Pakete werden bei der Weiterverarbeitung gesammelt und zu Frames (=Bilder) zusammengefaßt. 12 Frames repräsentieren hier eine GOP. Ist eine vollständige GOP zwischengespeichert, so werden mittels eines H264-Decoders (hier verwendet *FFMpeg*) die unkomprimierten Bilder erzeugt. Diese werden danach im Abstand von 100 msec – entsprechend 10 Hz - ausgegeben, um einen kontinuierlichen Bewegungseindruck zu erzeugen. In dieser Zeit sammelt die Software weitere RTP-Pakete für die nächste GOP.
Dieses Verfahren führt allerdings zu einer Zeitverzögerung zwischen den Bilddaten an Bord des Hubschraubers und den am Boden dargestellten.

## *3.2. Client Software zur Kommunikation mittels RTSP-Protokoll*

Für die Kommunikation mit dem AXIS-Encoder über LAN werden von der steuernden Client-Software drei Sockets geöffnet. Der erste Socket für das RTSP-Protokoll dient zur Initialisierung und Steuerung des Videostreams mittels vordefinierter Kommandos, sowie für Rückmeldungen vom AXIS-Encoder. Detaillierte Informationen zum RTSP-Protokoll können im Internet gefunden werden. Über den zweiten Socket (RTCP = **R**ealtime **C**ontrol **P**rotocol) werden Informationen zur Einhaltung der Datenqualität bzw. Dienstqualität ausgetauscht.
Die H264-Videodaten werden über den dritten Socket (RTP) empfangen.

*Betrieb unter Windows im privaten Netz (nicht geroutet / IP=192.168.xxx.xxx)*
**1.** Start -> Netzwerke -> <rechte Maus Taste> -> Eigenschaften
-> Adapter Einstellungen ändern -> LAN Verbindung -> Eigenschaften
-> Internetprotokoll Version 4
( ) IP Adresse automatisch beziehen
(*) Folgende IP Adresse verwenden
IP-Adresse 192.168.zzz.zzz IP-Adresse des eigenen Rechners
Subnetzmaske 255.255.255.255
Standardgateway 192.168.aaa.aaa (IP-Adresse des AXIS-Encoders)
Da die Firewall auf dem Adapter im Modus „unbekannte Zone“ läuft, wird die Adresse des AXIS-Encoders als Gateway eintragen und die Arbeitsplatz-Zone dann ausgewählt.
**2.** Systemsteuerung -> Windows Firewall -> erweiterte Einstellungen ->
eingehende Regeln -> rechte Maustaste -> neue Regel
-> Programm -> Programmpfad -> .. -> Verbindung zulassen
(eingehende Regel bedeutet, das gewählte Programm baut die Verbindung nach außen auf / hier zum AXIS-Enc.)
(ausgehende Regel bedeutet, ein Programm von außen baut die Verbindung zu dem gewählten Programm auf)

### 3.2.1. RTP-Paket-Definition

Entnommen aus *https://de.wikipedia.org/wiki/Real-Time_Transport_Protocol*

**RTP Header**

| Byte 0 | | | | | | | | Byte 1 | | | | | | | | Byte 2 | | | | | | | | Byte 3 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Bit 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Bit 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Bit 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| V=2 | | P | X | CC | | | | M | PT | | | | | | | Sequence Number | | | | | | | | | | | | | | | |
| Timestamp (in sample rate units) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Synchronization Source (SSRC) identifier | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Contributing Source (CSRC) identifiers (optional) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Header Extension (optional) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

*Version (V), 2 bit*
: Versionsstand des RTP-Protokolls

*Padding (P), 1 bit*
: Das Füll-Bit ist gesetzt, wenn ein oder mehrere Füll-Bytes am Ende des Pakets angehängt sind, die nicht zum eigentlichen Dateninhalt (Payload) gehören. Das letzte Füll-Byte gibt die Anzahl der hinzugefügten Füll-Byte an. Füll-Byte werden nur dann benötigt, wenn nachfolgende Protokolle eine vorgegebene Blockgröße benötigen, z.B. Verschlüsselungsalgorithmen.

*Extension (X), 1 bit*
: Das Erweiterungs-Bit ist gesetzt, wenn der Header um genau einen Erweiterungs-Header ergänzt wird

*CSRC Count (CC), 4 bit*
: Der CSRC-Zähler gibt die Anzahl der CSRC-Identifier an.

*Marker (M), 1 bit*
: Das Marker-Bit ist für anwendungsspezifische Verwendungen reserviert. Es wird genutzt zur Kennzeichnung von Ereignissen, z.B. dem Auftreten des Endes eines Einzelbildes einer Videosequenz.

*Payload Type (PT), 7 bit*
: Dieses Feld beschreibt das Format des zu transportierenden RTP-Inhalts, also der Nutzdaten (Payload) – Hier verwendeter Codec: **H264-Videodaten**.

*Sequence Number, 16 bit*
: Die Sequenznummer wird für jedes weitere RTP-Datenpaket erhöht. Die Startnummer wird zufällig ausgewählt und ist nicht vorherbestimmbar. Der Empfänger kann mit Hilfe der Sequenznummer die Paketreihenfolge wiederherstellen und den Verlust von Paketen erkennen.

*Timestamp, 32 bit*
: Der Zeitstempel gibt den Zeitpunkt des ersten Bytes des RTP-Datenpakets an. Der Zeitpunkt muss sich an einem Takt orientieren, der kontinuierlich und linear ist, damit die Synchronität des Streams sichergestellt und Laufzeitunterschiede der Übertragungsstrecke (Jitter) ermittelt werden können. Der Startwert sollte wie die Sequenznummer ein zufälliger Wert sein. Aufeinanderfolgende Pakete können den gleichen Zeitstempel haben, wenn die transportierten Daten z.B. zum selben Videoframe gehören. Pakete mit aufeinanderfolgenden Sequenznummern können aber auch nicht aufeinanderfolgende Zeitstempel enthalten, wenn wie z.B. bei komprimiertem Video Übertragungs- und Wiedergabereihenfolge nicht übereinstimmen.

*SSRC, 32 bit*
: Dieses Feld dient zur Identifikation der Synchronisationsquelle. Der Wert wird zufällig ermittelt, damit nicht zwei Quellen innerhalb der RTP-Session die gleiche Identifikationsnummer besitzen.

*Header Extension für H264-Videodaten, 16 bit*

```
Bit     0 1 2   3 4 5 6 7   0   1   2   3 4 5 6 7   0 1 2 4 5 6 7 0 1 2 3 4 5 6 7
        | NAL   | Fragment  | S | E | R |   NAL     |   RTP Payload                 |
        | Unit  |   Type    |   |   |   |   Unit    |                               |
```

Für die Zuordnung der H264-Daten (Payload) ist die Header Extension wie folgt zu interpretieren:

| | |
|---|---|
| *Fragment Typ=0x01*: | Frame ist unfragmentiert und besteht nur aus diesem RTP-Paket |
| *Fragment Typ=0x1C=28* | Frame ist fragmentiert und besteht aus mehreren RTP-Paketen |
| *S (Start-Bit) = 1* | Frame startet mit diesem RTP Paket |
| *E (End-Bit) = 1* | fragmentierter Frame endet mit diesem RTP Paket |
| *NAL Unit = 0x65* | RTP-Paket gehört zu einem I-Frame |
| (NAL=**N**etwork **A**bstraction **L**ayer) | |
| *R* | Reserved |

Das nachfolgende Beispiel einer Sequenz von RTP-Headern (s. Tabelle 1) illustriert die obigen Spezifikationen (0x bedeutet hexadezimaler Wert):

| *Seq. Number* | *Timestamp* | *SSRC* | *Header-Ext. H264* | **Typ** | **S** | **E** | **NAL** |
|---|---|---|---|---|---|---|---|
| 0x80600BF2 | 0x0037A585 | 0x11C9B849 | 0x7C85 (**0111 1100 1000 0101**) | **0x1C** | **1** | **0** | **0x65** |
| 0x80600BF3 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BF4 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BF5 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BF6 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BF7 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BF8 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BF9 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BFA | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BFB | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BFC | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BFD | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BFE | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600BFF | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600C00 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600C01 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600C02 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600C03 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80600C04 | 0x0037A585 | 0x11C9B849 | 0x7C05 (**0111 1100 0000 0101**) | **0x1C** | **0** | **0** | **0x65** |
| 0x80E00C05 | 0x0037A585 | 0x11C9B849 | 0x7C45 (**0111 1100 0100 0101**) | **0x1C** | **0** | **1** | **0x65** |
| 0x80600C06 | 0x0037CFB5 | 0x11C9B849 | 0x5C81 (**0101 1100 1000 0001**) | **0x1C** | **1** | **0** | **0x41** |
| 0x80600C07 | 0x0037CFB5 | 0x11C9B849 | 0x5C01 (**0101 1100 0000 0001**) | **0x1C** | **0** | **0** | **0x41** |
| 0x80600C08 | 0x0037CFB5 | 0x11C9B849 | 0x5C01 (**0101 1100 0000 0001**) | **0x1C** | **0** | **0** | **0x41** |
| 0x80E00C09 | 0x0037CFB5 | 0x11C9B849 | 0x5C41 (**0101 1100 0100 0001**) | **0x1C** | **0** | **1** | **0x41** |
| 0x80600C0A | 0x0037F9E2 | 0x11C9B849 | 0x5C81 (**0101 1100 1000 0001**) | **0x1C** | **1** | **0** | **0x41** |
| 0x80600C0B | 0x0037F9E2 | 0x11C9B849 | 0x5C01 (**0101 1100 0000 0001**) | **0x1C** | **0** | **0** | **0x41** |
| 0x80600C0C | 0x0037F9E2 | 0x11C9B849 | 0x5C01 (**0101 1100 0000 0001**) | **0x1C** | **0** | **0** | **0x41** |
| 0x80600C0D | 0x0037F9E2 | 0x11C9B849 | 0x5C01 (**0101 1100 0000 0001**) | **0x1C** | **0** | **0** | **0x41** |
| 0x80600C0E | 0x0037F9E2 | 0x11C9B849 | 0x5C01 (**0101 1100 0000 0001**) | **0x1C** | **0** | **0** | **0x41** |
| 0x80E00C0F | 0x0037F9E2 | 0x11C9B849 | 0x5C41 (**0101 1100 0100 0001**) | **0x1C** | **0** | **1** | **0x41** |
| 0x80600C10 | 0x00381602 | 0x11C9B849 | 0x5C81 (**0101 1100 1000 0001**) | **0x1C** | **1** | **0** | **0x41** |

*Tabelle 1.Beispiel für eine Folge von RTP-Headern*

### 3.2.2. H264 Decodierung

Um die, aus den RTP Paketen wieder zusammengesetzten Bilder decodieren zu können, müssen diese, als zusammenhängende GOPs abgespeichert und in einen H264-Raw-Stream umgewandelt werden. Dazu bekommt jedes Bild, das aus fragmentierten Paketen zusammengesetzt wird einen Prefix bestehend aus den Bytes **0x00, 0x00, 0x01** und **NAL-UNIT** vorangestellt. Bei unfragmentierten Bildern – *Fragment Typ* = 1 – werden nur die Bytes **0x00, 0x00** und **0x01** vorangestellt. Außerdem wird an den Anfang des GOPs, je nach Bildgröße ein sogenannter „Header" eingefügt.

Header bei einer Bildgröße von

*704x576* Pixel

0x00,0x00,0x00,0x01,0x67,0x42,0x00,0x29,0xE2,0x90,0x16,0x02,0x4D,0x81,0x27,0x05,
0x01,0x05,0xE1,0xE2,0x44,0x54,0x00,0x00,0x01,0x68,0xCE,0x3C,0x80,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x81,0x02,0x00,0x00,0x00,0x00,0x00,0x0A,0x46,0x66,0x00,0x00,
0x35,0x04,0x00,0x00,0x28,0x00,0xD0,0x22

*352x288* Pixel

0x00,0x00,0x00,0x01,0x67,0x42,0x00,0x29,0xE2,0x90,0x2C,0x12,0xD8,0x12,0x70,0x50,
0x10,0x5E,0x1E,0x24,0x45,0x40,0x00,0x00,0x01,0x68,0xCE,0x3C,0x80,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x81,0x02,0x00,0x00,0x00,0x00,0x00,0x0A,0x6C,0x29,0x00,0x00,
0xE9,0x03,0x00,0x00,0x3F,0x00,0xD0,0x22);

Diese, im H264-Format vorliegende GOP wird durch Nutzung der, auf *FFMpeg* basierenden Bibliothek *Acinerella.dll* decodiert.

Die folgende Beschreibung entstammt einem Internetbeitrag:
*Acinerella* ist ein in C geschriebener Wrapper für die Decodierfähigkeiten der *FFMpeg* Bibliotheken *libavformat und libavcodec*. Acinerella exportiert gerade mal 10 API-Funktionen - daher ist es sehr einfach die Headerdatei zu übersetzen und Acinerella mit anderen Programmiersprachen (zum Beispiel Pascal) zu verwenden. Würde man FFMpeg direkt verwenden wollen, so muß man drei 1000 Zeilen lange Header übertragen.
Zwar gibt es eine Pascal Version dafür, doch diese ist nicht Up-To-Date und dazu höchst wahrscheinlich fehlerhaft.
Acinerella ist einfach zu verwenden. Somit ist die größte Hürde auf dem Weg zum eigenen Video- und Audioplayer - das Decodieren der Daten - genommen. Die Datenströme werden direkt aus dem Speicher geladen, wodurch sich größtmögliche Freiheit bei der Entwicklung ergibt.
Alle FFMpeg Bibliotheken sind statisch mit Acinerella verbunden, wodurch sich eine einzelne, 2 MB große, DLL ergibt.
Wichtig:
Die Bibliothek ist unter der GPL veröffentlicht! Also wer damit geschriebene Programme veröffentlicht, muß (entsprechend der Lizenz) den Sourcecode davon ebenfalls offen legen.

### 3.2.3. Abhängigkeit der RTP-Paketanzahl vom Bildinhalt

Die Anzahl der versendeten RTP-Pakete hängt vom Bildinhalt ab. Ändern sich die Bildinhalte nur geringfügig, so ist die Differenz zwischen zwei Bildern gering und die entsprechenden P-Frames enthalten nur wenige Daten. Da die I-Frames jeweils den gesamten Bildinhalt codieren, bleibt deren Größe relativ unabhängig von der Veränderung der Bildinhalte. Allerdings hängt sie von den Bilddetails ab.

Die folgenden Beispiele illustrieren diesen Zusammenhang:

| *Nr* | *Zeit/[ms]* | *Paketgröße* | *\|Seq.Nr* | *Timestamp* | *SSRC* | *Header-Ext. H264* | |
|---|---|---|---|---|---|---|---|
| 1 | 0,0 | 296 | 0x80603679 | 0x2B0BC2FB | 0x13942BF4 | 0x7C85 | Start I-Frame |
| 2 | 0,5 | 296 | 0x8060367A | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 3 | 0,8 | 296 | 0x8060367B | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| ..4 | 1,0 | 296 | 0x8060367C | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 5 | 1,3 | 296 | 0x8060367D | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 6 | 1,6 | 296 | 0x8060367E | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 7 | 1,8 | 296 | 0x8060367F | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 8 | 2,1 | 296 | 0x80603680 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 9 | 2,3 | 296 | 0x80603681 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 10 | 2,6 | 296 | 0x80603682 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 11 | 2,9 | 296 | 0x80603683 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 12 | 3,1 | 296 | 0x80603684 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 13 | 3,4 | 296 | 0x80603685 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 14 | 3,6 | 296 | 0x80603686 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 15 | 3,9 | 296 | 0x80603687 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 16 | 4,2 | 296 | 0x80603688 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 17 | 4,4 | 296 | 0x80603689 | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 18 | 4,7 | 296 | 0x8060368A | 0x2B0BC2FB | 0x13942BF4 | 0x7C05 | Fragment I-Frame |
| 19 | 4,9 | 55 | 0x80E0368B | 0x2B0BC2FB | 0x13942BF4 | 0x7C45 | Ende I-Frame |
| 20 | 5,1 | 205 | 0x80E0368C | 0x2B0BFB39 | 0x13942BF4 | 0x419A | P-Frame (vollständig) |
| 21 | 5,4 | 148 | 0x80E0368D | 0x2B0C1759 | 0x13942BF4 | 0x419A | P-Frame (vollständig) |
| 22 | 5,7 | 94 | 0x80E0368E | 0x2B0C3377 | 0x13942BF4 | 0x419A | P-Frame (vollständig) |
| 23 | 109,9 | 107 | 0x80E0368F | 0x2B0C5DA6 | 0x13942BF4 | 0x419A | P-Frame (vollständig) |
| 24 | 188,3 | 128 | 0x80E03690 | 0x2B0C79C6 | 0x13942BF4 | 0x419A | P-Frame (vollständig) |
| 25 | 314,1 | 93 | 0x80E03691 | 0x2B0CA3F4 | 0x13942BF4 | 0x419A | P-Frame (vollständig) |
| 26 | 395,2 | 71 | 0x80E03692 | 0x2B0CC015 | 0x13942BF4 | 0x419A | P-Frame (vollständig) |
| 27 | 514,2 | 55 | 0x80E03693 | 0x2B0CEA44 | 0x13942BF4 | 0x419B | P-Frame (vollständig) |
| 28 | 606,4 | 68 | 0x80E03694 | 0x2B0D0664 | 0x13942BF4 | 0x419B | P-Frame (vollständig) |
| 29 | 718,4 | 57 | 0x80E03695 | 0x2B0D3094 | 0x13942BF4 | 0x419B | P-Frame (vollständig) |
| 30 | 799,4 | 58 | 0x80E03696 | 0x2B0D4CB2 | 0x13942BF4 | 0x419B | P-Frame (vollständig) |
| 31 | 925,4 | 296 | 0x80603697 | 0x2B0D76DF | 0x13942BF4 | 0x7C85 | Start I-Frame |

*Tabelle 2. Bildfolge mit geringfügigen Änderungen im Bildinhalt*

Bei dem obigen Beispiel in Tabelle 2 wird ein GOP, entsprechend 12 Frames, mittels 30 RTP-Paketen übertragen. Die gewählte Paketgröße lag bei 296 Bytes. Bei einer Abtastfrequenz des PCM-Rahmens von 100 Hz, welches einer maximal möglichen Übertragungsrate von 100 RTP-Paketen pro sec entspricht, können die H264-Videodaten verlustfrei übertragen werden.
Ab einer bestimmten Zeit nach dem Starten der RTP-Übertragung beträgt die Zeit zwischen zwei P-Frames ungefähr 100 msec, welches der eingestellten Bildwiederholfrequenz von 10 Hz entsprechen sollte.
Solange die Größe eines Frames kleiner 296 Bytes ist, werden die Daten vollständig, also unfragmentiert übertragen.

| *Nr* | *Zeit/[ms]* | *Paketgröße* | *\|Seq.Nr* | *Timestamp* | *SSRC* | *Header-Ext. H264* | |
|---|---|---|---|---|---|---|---|
| 1 | 0,0 | 296 | 0x80600BF2 | 0x0037A585 | 0x11C9B849 | 0x7C85 | Start I-Frame |
| 2 | 0,4 | 296 | 0x80600BF3 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 3 | 0,7 | 296 | 0x80600BF4 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 4 | 0,9 | 296 | 0x80600BF5 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 5 | 1,1 | 296 | 0x80600BF6 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 6 | 1,4 | 296 | 0x80600BF7 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 7 | 1,6 | 296 | 0x80600BF8 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 8 | 1,8 | 296 | 0x80600BF9 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 9 | 2,1 | 296 | 0x80600BFA | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 10 | 2,4 | 296 | 0x80600BFB | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 11 | 2,6 | 296 | 0x80600BFC | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 12 | 2,8 | 296 | 0x80600BFD | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 13 | 3,1 | 296 | 0x80600BFE | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 14 | 3,3 | 296 | 0x80600BFF | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 15 | 3,6 | 296 | 0x80600C00 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 16 | 3,8 | 296 | 0x80600C01 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 17 | 4,0 | 296 | 0x80600C02 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 18 | 4,3 | 296 | 0x80600C03 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 19 | 4,5 | 296 | 0x80600C04 | 0x0037A585 | 0x11C9B849 | 0x7C05 | Fragment I-Frame |
| 20 | 4,7 | 20 | 0x80E00C05 | 0x0037A585 | 0x11C9B849 | 0x7C45 | Ende I-Frame |
| 21 | 49,3 | 296 | 0x80600C06 | 0x0037CFB5 | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 22 | 49,9 | 296 | 0x80600C07 | 0x0037CFB5 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 23 | 50,2 | 296 | 0x80600C08 | 0x0037CFB5 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 24 | 50,7 | 146 | 0x80E00C09 | 0x0037CFB5 | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 25 | 170,2 | 296 | 0x80600C0A | 0x0037F9E2 | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 26 | 171,0 | 296 | 0x80600C0B | 0x0037F9E2 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 27 | 171,2 | 296 | 0x80600C0C | 0x0037F9E2 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 28 | 171,6 | 296 | 0x80600C0D | 0x0037F9E2 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 29 | 172,1 | 296 | 0x80600C0E | 0x0037F9E2 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 30 | 172,6 | 76 | 0x80E00C0F | 0x0037F9E2 | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 31 | 251,8 | 296 | 0x80600C10 | 0x00381602 | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 32 | 252,2 | 296 | 0x80600C11 | 0x00381602 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 33 | 252,7 | 296 | 0x80600C12 | 0x00381602 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 34 | 253,2 | 296 | 0x80600C13 | 0x00381602 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 35 | 253,7 | 61 | 0x80E00C14 | 0x00381602 | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 36 | 370,8 | 296 | 0x80600C15 | 0x00384030 | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 37 | 371,2 | 296 | 0x80600C16 | 0x00384030 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 38 | 371,6 | 296 | 0x80600C17 | 0x00384030 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 39 | 372,1 | 296 | 0x80600C18 | 0x00384030 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 40 | 372,6 | 243 | 0x80E00C19 | 0x00384030 | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 41 | 456,0 | 296 | 0x80600C1A | 0x00385C50 | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 42 | 456,5 | 296 | 0x80600C1B | 0x00385C50 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 43 | 456,8 | 296 | 0x80600C1C | 0x00385C50 | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 44 | 457,3 | 176 | 0x80E00C1D | 0x00385C50 | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 45 | 580,8 | 296 | 0x80600C1E | 0x0038867F | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 46 | 581,3 | 296 | 0x80600C1F | 0x0038867F | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 47 | 581,6 | 296 | 0x80600C20 | 0x0038867F | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 48 | 582,1 | 296 | 0x80600C21 | 0x0038867F | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 49 | 582,6 | 296 | 0x80600C22 | 0x0038867F | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 50 | 583,1 | 296 | 0x80600C23 | 0x0038867F | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 51 | 583,5 | 31 | 0x80E00C24 | 0x0038867F | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 52 | 659,2 | 296 | 0x80600C25 | 0x0038A29E | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 53 | 671,9 | 296 | 0x80600C26 | 0x0038A29E | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 54 | 672,4 | 296 | 0x80600C27 | 0x0038A29E | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 55 | 672,7 | 296 | 0x80600C28 | 0x0038A29E | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 56 | 673,2 | 296 | 0x80600C29 | 0x0038A29E | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 57 | 673,7 | 51 | 0x80E00C2A | 0x0038A29E | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 58 | 778,4 | 296 | 0x80600C2B | 0x0038CCCD | 0x11C9B849 | 0x5C81 | Start P-Frame |

*Tabelle 3.1. Bildfolge mit markanten Änderungen im Bildinhalt*

| *Nr* | *Zeit/[ms]* | *Paketgröße* | *\|Seq.Nr* | *Timestamp* | *SSRC* | *Header-Ext. H264* | |
|---|---|---|---|---|---|---|---|
| 59 | 778,9 | 296 | 0x80600C2C | 0x0038CCCD | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 60 | 779,3 | 296 | 0x80600C2D | 0x0038CCCD | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 61 | 779,7 | 296 | 0x80600C2E | 0x0038CCCD | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 62 | 780,2 | 296 | 0x80600C2F | 0x0038CCCD | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 63 | 780,7 | 251 | 0x80E00C30 | 0x0038CCCD | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 64 | 867,5 | 296 | 0x80600C31 | 0x0038E8ED | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 65 | 868,1 | 296 | 0x80600C32 | 0x0038E8ED | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 66 | 868,5 | 296 | 0x80600C33 | 0x0038E8ED | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 67 | 869,3 | 296 | 0x80600C34 | 0x0038E8ED | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 68 | 869,9 | 175 | 0x80E00C35 | 0x0038E8ED | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 69 | 985,2 | 296 | 0x80600C36 | 0x0039131B | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 70 | 997,5 | 296 | 0x80600C37 | 0x0039131B | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 71 | 998,0 | 296 | 0x80600C38 | 0x0039131B | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 72 | 998,5 | 296 | 0x80600C39 | 0x0039131B | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 73 | 998,8 | 197 | 0x80E00C3A | 0x0039131B | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 74 | 1067,3 | 296 | 0x80600C3B | 0x00392F3B | 0x11C9B849 | 0x5C81 | Start P-Frame |
| 75 | 1067,7 | 296 | 0x80600C3C | 0x00392F3B | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 76 | 1068,1 | 296 | 0x80600C3D | 0x00392F3B | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 77 | 1068,6 | 296 | 0x80600C3E | 0x00392F3B | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 78 | 1069,0 | 296 | 0x80600C3F | 0x00392F3B | 0x11C9B849 | 0x5C01 | Fragment P-Frame |
| 79 | 1069,5 | 278 | 0x80E00C40 | 0x00392F3B | 0x11C9B849 | 0x5C41 | Ende P-Frame |
| 80 | 1193,5 | 296 | 0x80600C41 | 0x0039596A | 0x11C9B849 | 0x7C85 | Start I-Frame |

*Tabelle 3.2. Bildfolge mit markanten Änderungen im Bildinhalt*

Bei dem gewählten Beispiel in Tabelle 3.1/2 wird ein GOP, mittels 79 RTP-Paketen übertragen. Auch hier können noch alle H264-Videodaten verlustfrei per PCM übertragen werden.

## *3.3. Datenformat für den PCM-Transfer*

Die, vom AXIS-Encoder gesendeten RTP-Pakete werden gesammelt und mit einer Frequenz von 100 Hz über eine UDP-Verbindung an einen Bordrechner verschickt. Dieser Rechner ist für die Verarbeitung von Flugzustandsgrößen zuständig. Zu seinen Aufgaben gehört auch das Versenden von gemessenen und berechneten Daten über einen PCM-Encoder. Den RTP-Datenpaketen wird ein 16 Byte großer Datenblock vorangestellt. Diese Daten-Pakete werden an eine feste Stelle im PCM-Rahmen eingefügt und dieser per Funk zur Bodenstation übertragen.

| | UDP zum Bordrechner für PCM | UDP nach PCM von **VuSoft** zum Bodenrechner |
|---|---|---|
| Buffer[0] | Identifier / Byte4 **0xFF** | PCM-Sync / Byte4 / **Bit 31..24** |
| Buffer[1] | Identifier / Byte3 **0x56** | PCM-Sync / Byte3 / **Bit 23..16** |
| Buffer[1] | Identifier / Byte2 **0x49** | PCM-Sync / Byte2 / **Bit 15.. 8** |
| Buffer[3] | Identifier / Byte1 **0xFF** | PCM-Sync / Byte1 / **Bit 7.. 0** |
| Buffer[4] | Transfer-Counter / Byte4 / **Bit 31..24** | PCM- Counter / Byte4 / **Bit 31..24** |
| Buffer[5] | Transfer-Counter / Byte3 / **Bit 23..16** | PCM- Counter / Byte3 / **Bit 23..16** |
| Buffer[6] | Transfer-Counter / Byte2 / **Bit 15.. 8** | PCM- Counter / Byte2 / **Bit 15.. 8** |
| Buffer[7] | Transfer-Counter / Byte1 / **Bit 7.. 0** | PCM- Counter / Byte1 / **Bit 7.. 0** |
| Buffer[8] | Transfer-Counter / Byte4 / **Bit 31..24** | Transfer-Counter / Byte4 |
| Buffer[9] | Transfer-Counter / Byte3 / **Bit 23..16** | Transfer-Counter / Byte3 |
| Buffer[10] | Transfer-Counter / Byte2 / **Bit 15.. 8** | Transfer-Counter / Byte2 |
| Buffer[11] | Transfer-Counter / Byte1 / **Bit 7.. 0** | Transfer-Counter / Byte1 |
| Buffer[12] | RTP-Paketgröße / Byte2 / **Bit 15.. 8** | RTP-Paketgröße / Byte2 |
| Buffer[13] | RTP-Paketgröße / Byte1 / **Bit 7.. 0** | RTP-Paketgröße / Byte1 |
| Buffer[14] | RTP-Checksumme / Byte2 / **Bit 15.. 8** | RTP-Checksumme / Byte2 |
| Buffer[15] | RTP-Checksumme / Byte1 / **Bit 7.. 0** | RTP-Checksumme / Byte1 |
| Buffer[16] | RTP- Datenpaket / Byte1 (bis max. 296) | RTP- Datenpaket / Byte1 |

*Tabelle 4. Datenformat der Videodaten vor und nach dem Versenden per PCM*

Die Anzahl der Bytes des jeweiligen RTP-Datenpaketes, beginnend ab Buffer[16], ist unter den Bits 13..0 des Buffers[12] und Buffer[13] abgelegt. Bit15 von Buffer[12] enthält die Information über die gewählte Bildauflösung:
Bit15 = 1: Bildauflösung = 704 * 576 Bit15 = 0: Bildauflösung = 352 * 288

Buffer[14] und Buffer[15] enthalten die Checksumme, die beim Aufaddieren von allen Bytes des RTP-Datenpakets entsteht.

Der Bodenrechner vergleicht die *Sequence Number* vom Header des empfangenen RTP-Paketes mit der des vorherigen RTP-Paketes, um doppelt gesendete Pakete auszusortieren oder fehlende zu erkennen.

### *3.4. Verlustfreie Echtzeitübertragung der empfangenen Videodaten mittels VuSoft*

Um eine kontinuierliche Wiedergabe des bordseitig erzeugten Videos zu erreichen, muß gewährleistet sein, daß jedes per PCM-Telemetrie versendetes Videodatenpaket auch zur Auswertung bereit gestellt wird. Der Verlust von einzelnen Videopaketen stört oder verhindert die Bilderzeugung. Da der Bildinhalt komprimiert in den Paketen gespeichert ist, werden nicht nur bestimmte Bildbereiche gestört, sondern die gesamte Bildfolge.
Der Verlust von Videodatenpaketen kann durch Störungen auf der Telemetriestrecke entstehen. Eine weitere Ursache kann der unvollständige Transfer der empfangenen Datenpakete an den Videodecoder sein.
Die Software *VuSoft* (detaillierte Informationen siehe Benutzerhandbuch) decodiert die PCM-Daten und versendet eine Auswahl aus diesen an die H264-Decodersoftware. Realisiert ist dies in der Library „Embedded.dll“. Durch die Nutzung dieser Library ist gewährleistet, daß jedes empfangene Videodatenpaket per UDP an eine vorher festgelegte IP-Adresse mit Port verschickt wird.

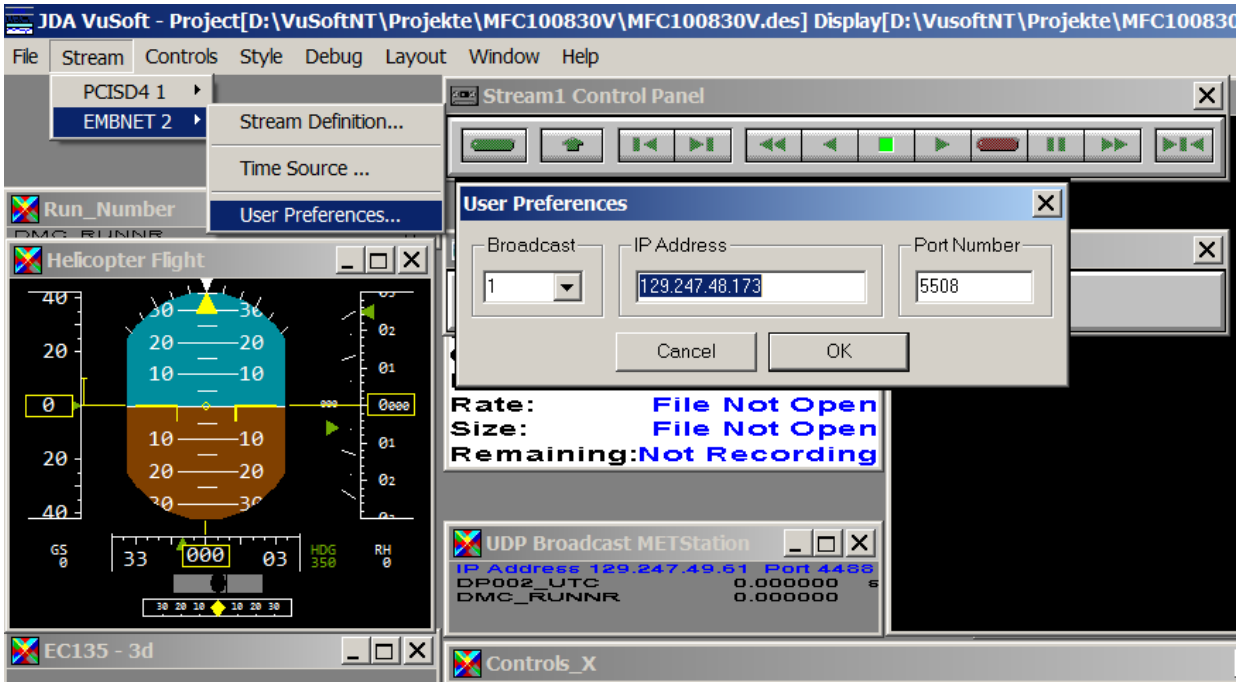


**Bild 5. Auswahl von Port und IP-Adresse für den „Embedded-Transfer“ in VuSoft**

Der „Embedded-Transfer“ wird durch ein separates Control Panel gestartet (s. Bild 6).

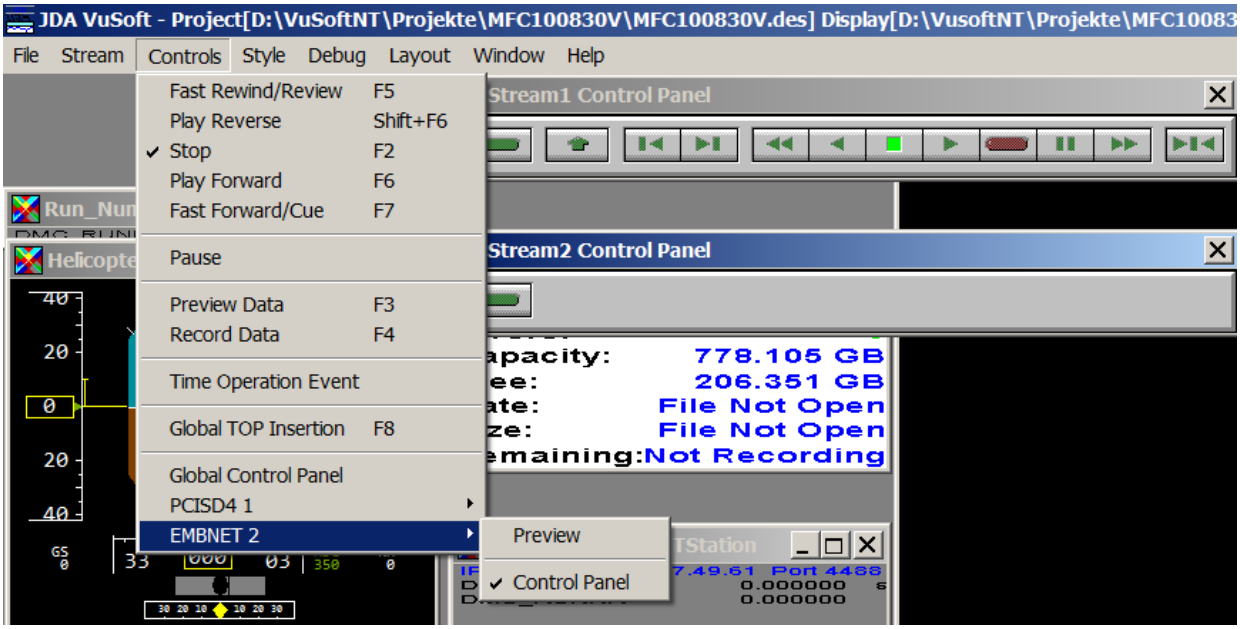


**Bild 6. Auswahl eines Control Panels für den „Embedded-Transfer“ in VuSoft**

Die Auswahl, der zu transferierenden Daten findet im Parameter-Menu von VuSoft statt (s. Bild7).

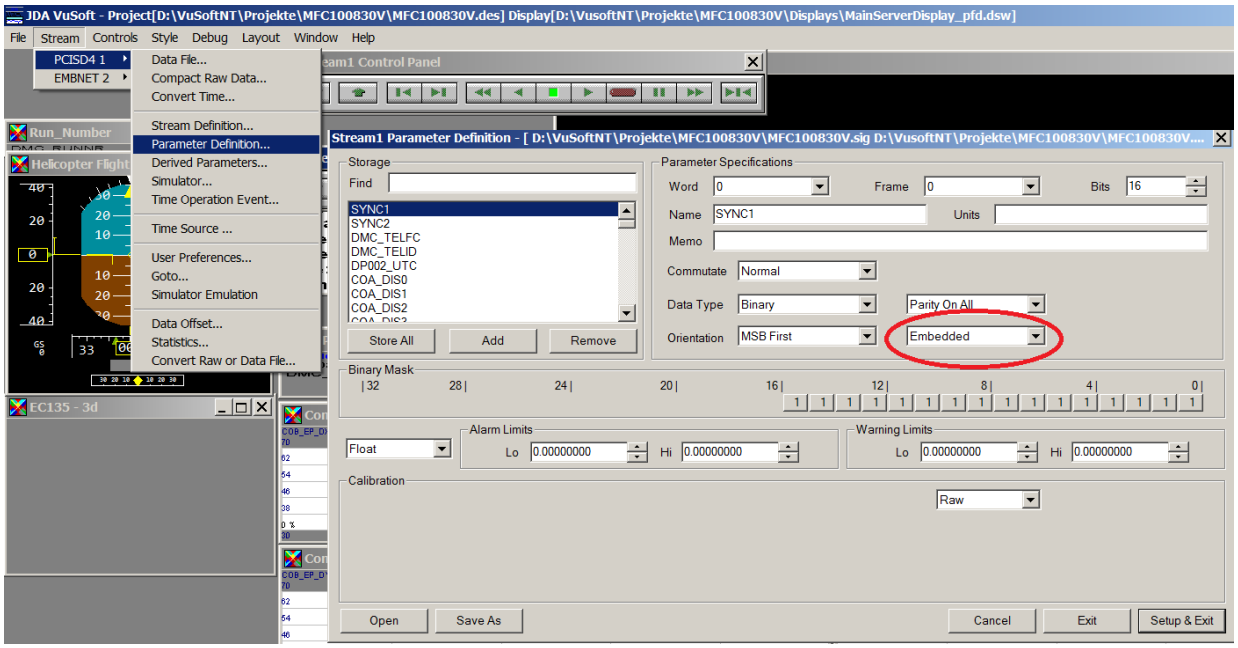


**Bild 7. Kennzeichnung von Parametern für den „Embedded-Transfer“**

Die Kennzeichnung kann aber auch schneller in der korrespondierenden Signal-Datei – Endung **.sig* – durchgeführt werden (s. Tabelle 5).

| *Word.Nr* | | *Bezeichnung* | | | *Bits* | *Embedded* |
|---|---|---|---|---|---|---|
| 1 | * | SYNC1 | ;;0;0;2; | ;0;;;;;0;0000100;;; | 16 | ;1;; |
| 2 | * | SYNC2 | ;;1;0;2; | ;0;;;;;0;0000100;;; | 16 | ;1;; |
| 3 | * | DMC_TELFC | ;;2;0;2; | ;0;;;;;0;0000100;;; | 32 | ;1;; |
| 4 | * | DMC_TELID | ;;3;0;2; | ;0;;;;;0;0000100;;; | 32 | ;0;; |
| 5 | * | DP002_UTC | ;;4;0;2;s | ;0;;;;;0;3000100;;; | 32 | ;0;; |
| …….. | | | | | | |
| …….. | | | | | | |
| 310 | * | FSS_ST_TRA | ;;309;0;2; | ;0;;;;;0;3000100;;; | 32 | ;0;; |
| 311 | * | FSS_ST_ROC | ;;310;0;2; | ;0;;;;;0;3000100;;; | 32 | ;0;; |
| 312 | * | SCC_VID3200 | ;;311;0;2; | ;0;;;;;0;0000100;;; | 32 | ;1;; |
| 313 | * | SCC_VID3201 | ;;312;0;2; | ;0;;;;;0;0000100;;; | 32 | ;1;; |
| 314 | * | SCC_VID3202 | ;;313;0;2; | ;0;;;;;0;0000100;;; | 32 | ;1;; |
| 315 | * | SCC_VID3203 | ;;314;0;2; | ;0;;;;;0;0000100;;; | 32 | ;1;; |
| …….. | | | | | | |

*Tabelle 5. Auszug aus der VuSoft-Signaldatei für das verwendete Projekt zur Videoübertragung*

Wie der obigen Tabelle 5 zu entnehmen ist, werden außer den Videodaten SCC_VIDxxxx auch noch das PCM-SYNC-Wort (SYNC1 und SYNC2), sowie der PCM-Rahmenzähler DMC_TELFC versendet. Dies dient dazu, den Anfang und die Anzahl der gesendeten Videopakete zu identifizieren.

67 26 35 FE 6B 28 40 00 03 ED 67 00 00 8F EA 00 8B 45 A6 **80 E0 0C 3E 4F 48 64 D2 3F 3C FF E4 41**
**9B 60 16 03 B8 E9 7F FF 11 92 B2 4A E2 FF 26 59 0F CE E1 5D F2 79 14 2D 12 E1 5C BC A6 85 32 DC**
**9B AF BA EF 0C 02 A4 2A C6 81 7B FD 59 3B D4 B8 8A 55 A9 77 5B F5 D3 EB 15 2E 7F 5A EF 3D B8 C9**
**5F BD 44 D5 64 E5 25 6A 4D 4D CF 5B CA 6B 8C CA 2D 93 D7 7E FC C5 B9 75 92 2F 2B 88 18 7A 19 2F**
**94 BF E9 F9 26 C9 17 21 36 2E 1E 21 7C 5E C3 67 66 E6 57 29 61 2B C9 28 9F 24 F4 58 EF D7** 77 CF
62 79 44 F5 ED 76 E6 CB 15 BE 43 13 8B 88 E6 21 BE 2F E6 92 23 3D 6A B3 7A 89 F6 AF D5 38 EF 25
FF D0 CA 89 2E 97 54 EE 97 4E E4 F4 FA 2F BE 5F 4F A7 36 22 6F 25 BA DF 4F 89 C9 10 E4 CB 56 A4
F2 7A B5 D2 8E 90 B2 48 E2 A0 E4 94 6E 4C 34 27 D7 7C DD 6A 13 50 40 33 A9 D3 FF FE B5 0A AA D4
D8 D4 E4 7F FF E8 2B BB BB BE 3D 4B 2F FA 7F FF FF D0 9D 6A BA D7 FC FF FB E7 00 FD DA F2 31 1A
B5 AE AB CF F5 86 A7 7E 55 FA 87 1A 5D D9 C8 F6 A6 93 50 C6 25 97 FF E8 67 26 35 FE 6B 28 40 00
03 ED 68 00 00 8F EB 01 28 90 F7 **80 60 0C 3F 4F 48 88 02 3F 3C FF E4 7C 85 88 84 00 00 AF 8E 23**
**90 42 01 E0 64 00 18 F2 DE A4 00 C7 96 01 F9 8D 88 50 F7 71 81 04 41 EE C1 7F 88 6C EA 06 95 7C**
**86 4A 50 72 00 F1 6B 29 24 A9 D3 9B D3 0B 91 A8 23 A4 38 3E F4 FC 68 18 E3 0A B2 CE 4C D4 ED 31**
**98 E2 6E 23 88 EC 10 0E BC 44 24 9D B1 5B 91 AD 3D E9 80 1F 1B 25 24 6A 27 FF 2D 40 A6 9F DD 48**
**3E 54 30 37 9B 98 2C 43 3C 8A 30 E4 62 C9 85 79 47 43 CB FD 24 A2 38 8E 10 0C 01 0F E3 10 23 CB**
**7C 71 DB 01 8A 68 DD 4B DB 6D 34 FC 7A 83 AA 9F FF D2 88 E2 3F FA 17 D7 51 9F 3A 99 AC 03 B5 3C**
**22 38 8F AF D2 EB EA B0 3B 39 AB 84 79 6C BF 63 11 C4 7F F4 33 6A AB F3 60 9E 44 71 1F F1 A1 1E**
**AB 5D E2 3D 79 D5 0E B9 A0 14 23 29 2E 5D 1E A1 FC B7 FF 5B 7F 11 C4 7F F4 17 EC D5 7F 62 38 8F**
**FE 85 FA AF D7 11 C4 7F F4 2F CD 8B F5 A4 2D 42 F6 85 A7 FC 47 11 FF D0 4F AE A8 F1 1C 47 FF 41**
**3E B5 D2 88 E2 3F FA 5D 7D 57 D5 03 5E 8F 50 02 C4 8A 7A** FE 6B 28 40 00 03 ED 69 00 00 8F EB 01
……..……(hexadezimale Darstellung)…….
PCM-Sync-Wort=0xFE 6B 28 40| PCM-Rahmenzähler DMC_TELFC| Transfer-Counter| **RTP-Paket/Videodaten**

*Tabelle 6. Anfang eines mit „Embedded.DLL“ aus VuSoft versendeten Datenbuffers*

Aus Tabelle 6 ist zu entnehmen, daß der übertragene Datenbuffer nicht mit dem PCM-SYNC-WORT, dem zuerst ausgewählten Wort für den „Embedded-Transfer“ beginnt. Außerdem werden pro UDP-Transfer auch mehrere Datenpakete verschickt, deren Anzahl von Transfer zu Transfer variiert.

In dem empfangenen Datenbuffer muß also der Anfang des Videodatenpaketes gesucht werden, welcher eindeutig durch das unveränderliche PCM-SYNC-Wort gegeben ist. Die enthaltene Anzahl der Videopakete bestimmt sich aus der Anzahl der, im Buffer gefundenen SYNC-WORTE.
Anhand des PCM-Rahmenzählers DMC_TELFC läßt sich erkennen, ob die Videodatenpakete aus fortlaufenden PCM-Rahmen kommen, also keine Daten durch die Übertragung verloren gegangen sind.
Über den *Transfer-Counter*, der bordseitig jedem Videodatenpaket zugeordnet wird und ebenso über die *Sequence Number* des RTP-Headers können doppelt gesendete Pakete erkannt werden.

## *3.5 Entwickelte Programme*

### 3.5.1. Client - Bordseitig

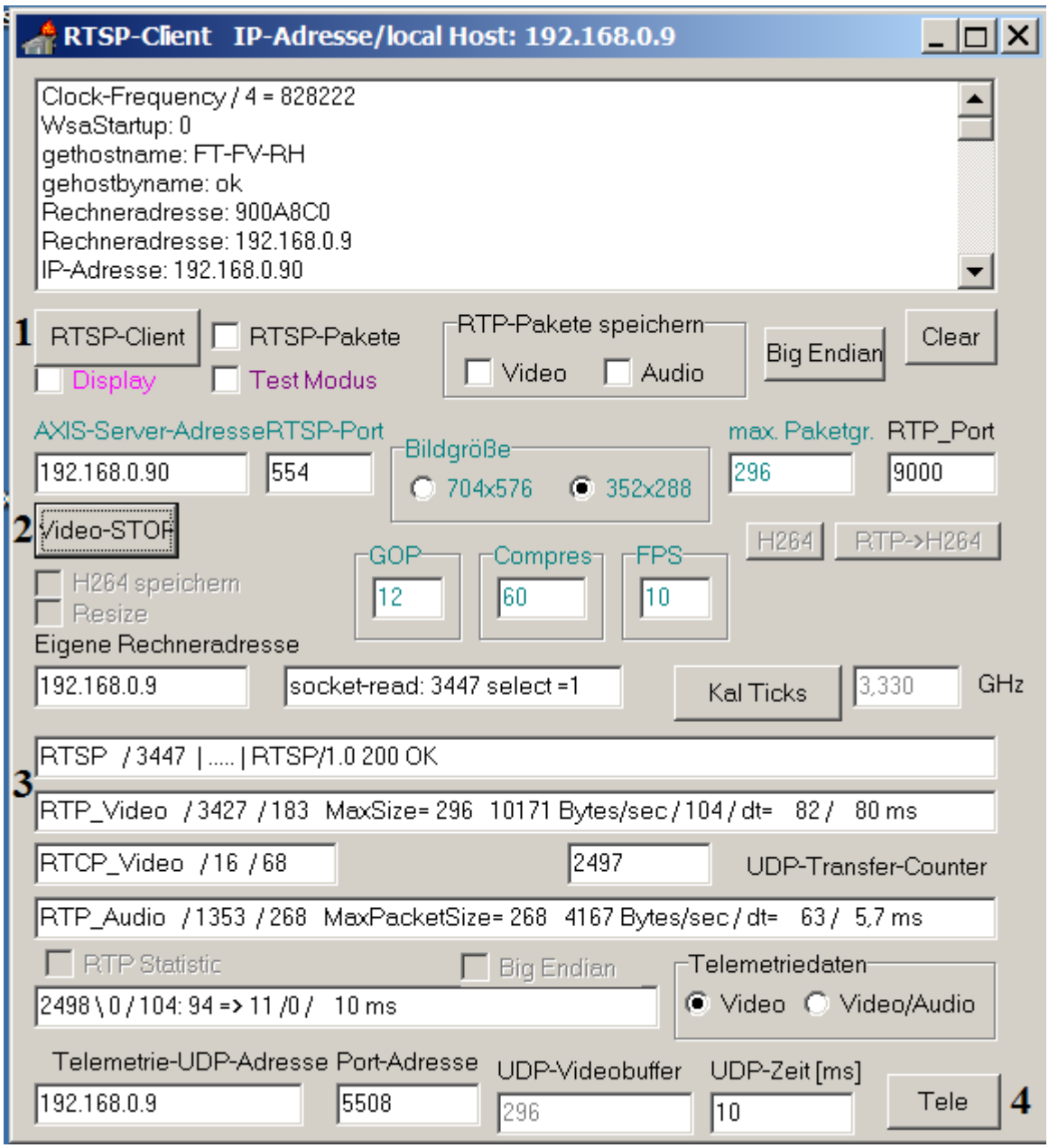


**Bild 8. Bordseitiges Programm für den Videodaten-Transfer vom AXIS-Encoder**

Bild 8 zeigt die Bedienoberfläche des Client-Programms für den Transfer der Videodaten vom AXIS-Encoder zum Bordrechner, um dort in den PCM-Rahmen einfügt zu werden.
Unter *„RTSP-Client“* **(1)** wird der RTSP-Datentransfer gestartet. Die Netzverbindung wird mit den, unter *AXIS-Server-Adresse* und unter *RTSP-Port*, eingegebenen Werten aufgebaut. Die Parameter für die Videobearbeitung durch den AXIS-Encoder werden unter *Bildgröße, GOP, Compres* und *FPS* vor dem Verbindungsaufbau eingestellt. Sobald die Verbindung steht, abzulesen an den Statusanzeigen **(3)**, kann die Decodierung und Darstellung der Videobilder mittels Button **(2)** initiiert bzw. gestoppt werden. Dies ist nur eine Option, um die Funktionalität zu überprüfen, nicht aber zum Betrieb notwendig
Die Verbindung und der UDP-Transfer von RTP-Paketen zum Bordrechner werden durch Betätigung des Buttons *Tele* **(4)** aufgebaut. Die, dafür notwendigen Netzparameter werden links vom Button *Tele*. eingegeben

Das Client-Programm läßt sich mit Angabe der Parameter für *GOP*, *Compres* und *FPS* über die Kommandozeile starten. Ohne Angabe von Parametern werden die, in Bild 8 angezeigten Werte genommen. 10 Sekunden nach dem Start werden die Verbindungen zum AXIS-Encoder und dem Bordrechner hergestellt.

### 3.5.2. Server - Bodenseitig

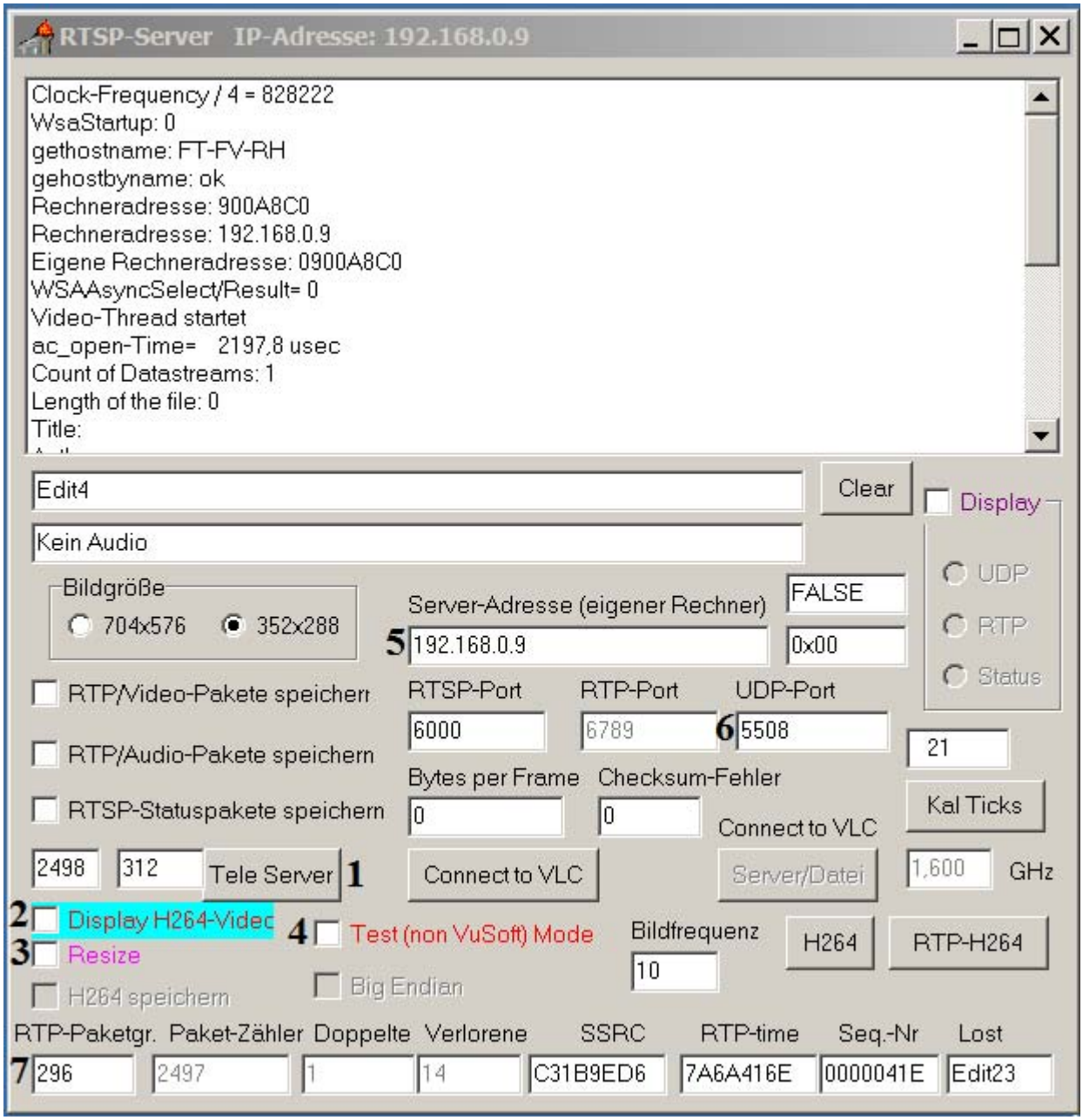


**Bild 9. Bodenseitiges Programm für den Empfang und die Visualisierung von Videodaten**

Der Button *„Tele Server“* **(1)** (s. Bild 9) versetzt das Programm in Empfangsbereitschaft. Durch die Wahl von *„Display H264-Video“* **(2)** wird die Darstellung der empfangenen Videobilder auf dem Display aktiviert. *„Resize“* **(3)** verdoppelt die Spalten und Zeilen des empfangenen Videobildes. Beide Optionen müssen vor dem Auslösen des Buttons *„Tele Server“* gewählt werden. UDP-Port **(6)** und IP-Adresse **(5)** für den „Embedded-Transfer“ korrespondieren mit denen im entsprechenden Menü des Programms VuSoft (s. Bild 5).
Sollten die Videodaten für Testzwecke direkt durch das Client-Programm (s. Bild 8) und nicht durch einen Embedded-Transfer via VuSoft verschickt werden, so muß die Option **(4)** *„Test (non VuSoft) Mode“* gewählt werden.
Der Status der empfangenen Videodaten wird in Zeile **(7)** bzw. unter Bildgröße dargestellt.

### 3.5.3 Bildvergrößerung – „Resize“

Die Größe des dargestellten Videobildes kann durch einen einfachen Algorithmus verändert werden. Durch die Option „*Resize*“ (s. Bild 9) werden Spalten und Zeilen des empfangenen Videobildes verdoppelt. Damit wird aus jedem einzelnen Pixel ein Block aus 4 Pixeln (s. Bild 10).

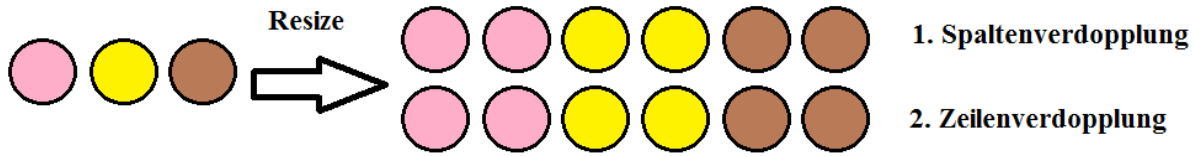


**Bild 10. Bildvergrößerung durch Verdopplung von Spalten und Zeilen**

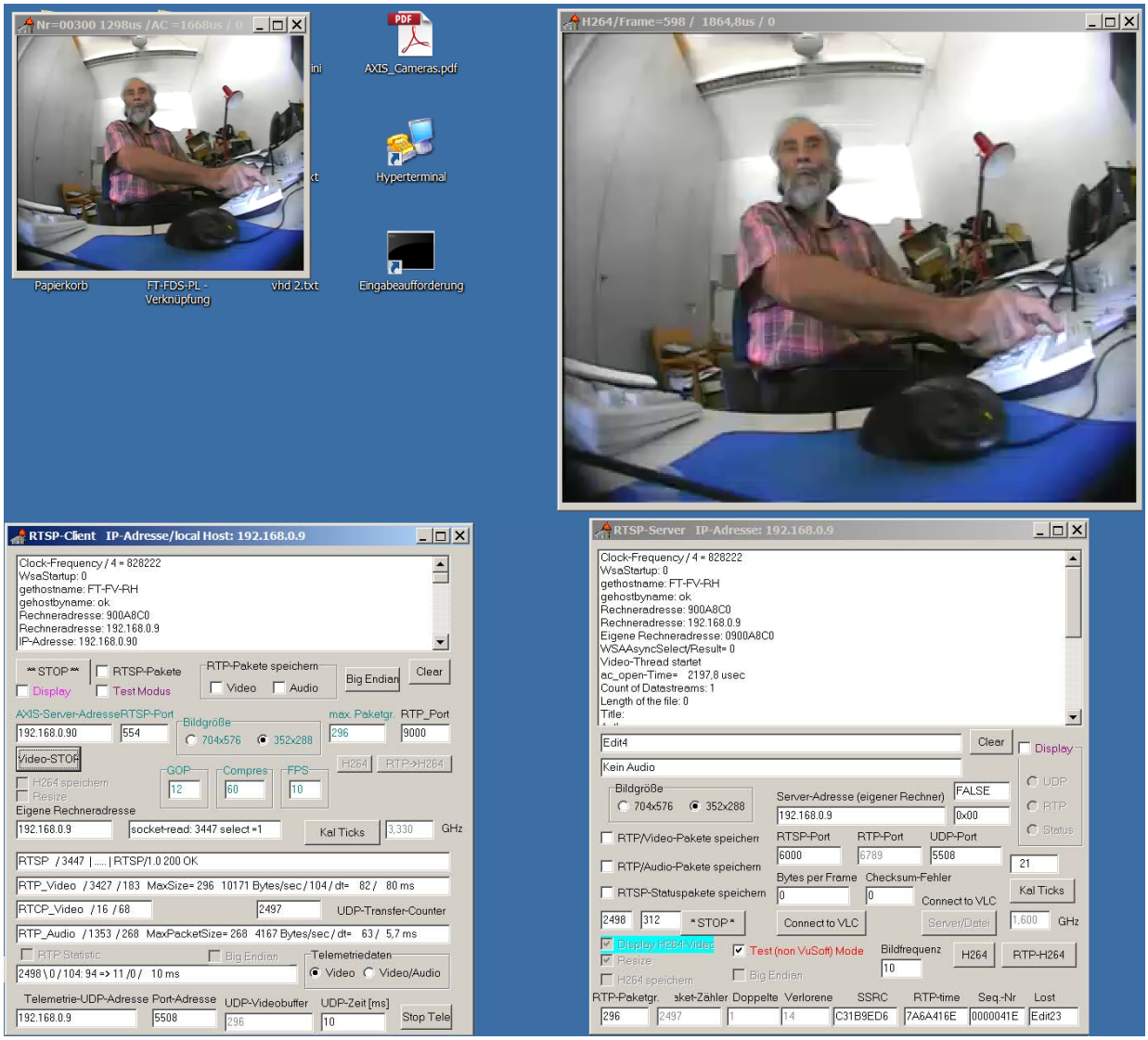


**Bild 11. Beispiel für den Resize-Algorithmus**

Links in Bild 11 ist das Video in Originalgröße mit $352 * 288$ Pixeln durch das Programm **Client** dargestellt. Rechts in Bild 11 ist das, mittels *„Resize“* vergrößerte Videobild mit $704 * 576$ Pixeln durch das Programm **Server** dargestellt.
Die Bildvergrößerung durch Spalten- und Zeilenverdopplung ist als Algorithmus einfach zu implementieren und benötigt weniger Zeit als andere, mit linearer bzw. nichtlinearer Pixelinterpolation arbeitende Algorithmen.
Der Nachteil der gröberen Rasterung des Bildes fällt erst beim Unterschreiten eines bestimmten Betrachtungsabstandes auf (s. Bild 12)

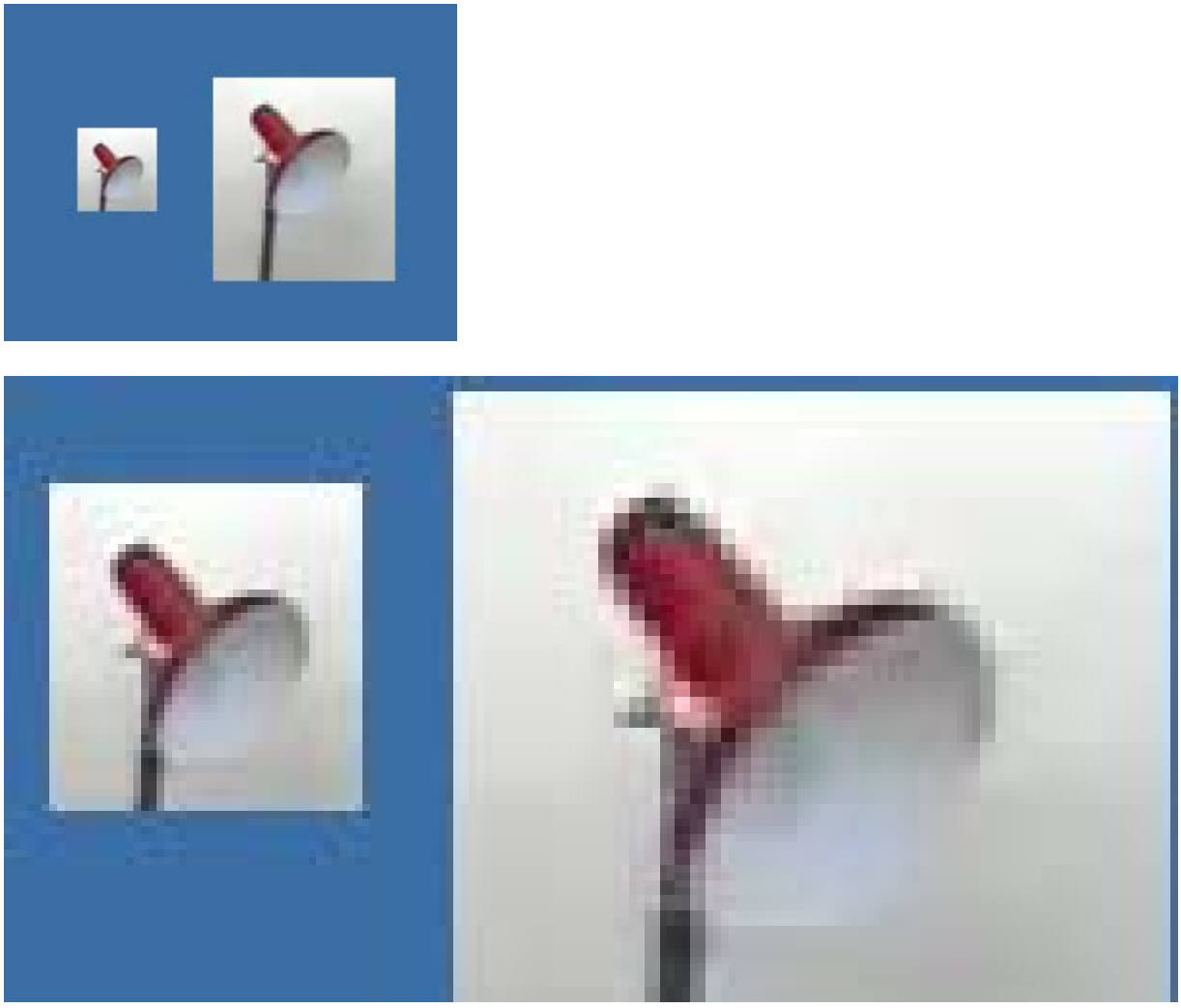

**Bild 12. Abhängigkeit der wahrgenommenen Bildqualität vom Betrachtungsabstand**

# 4. Ergebnisse

## *4.1. Zeitverzögerung*

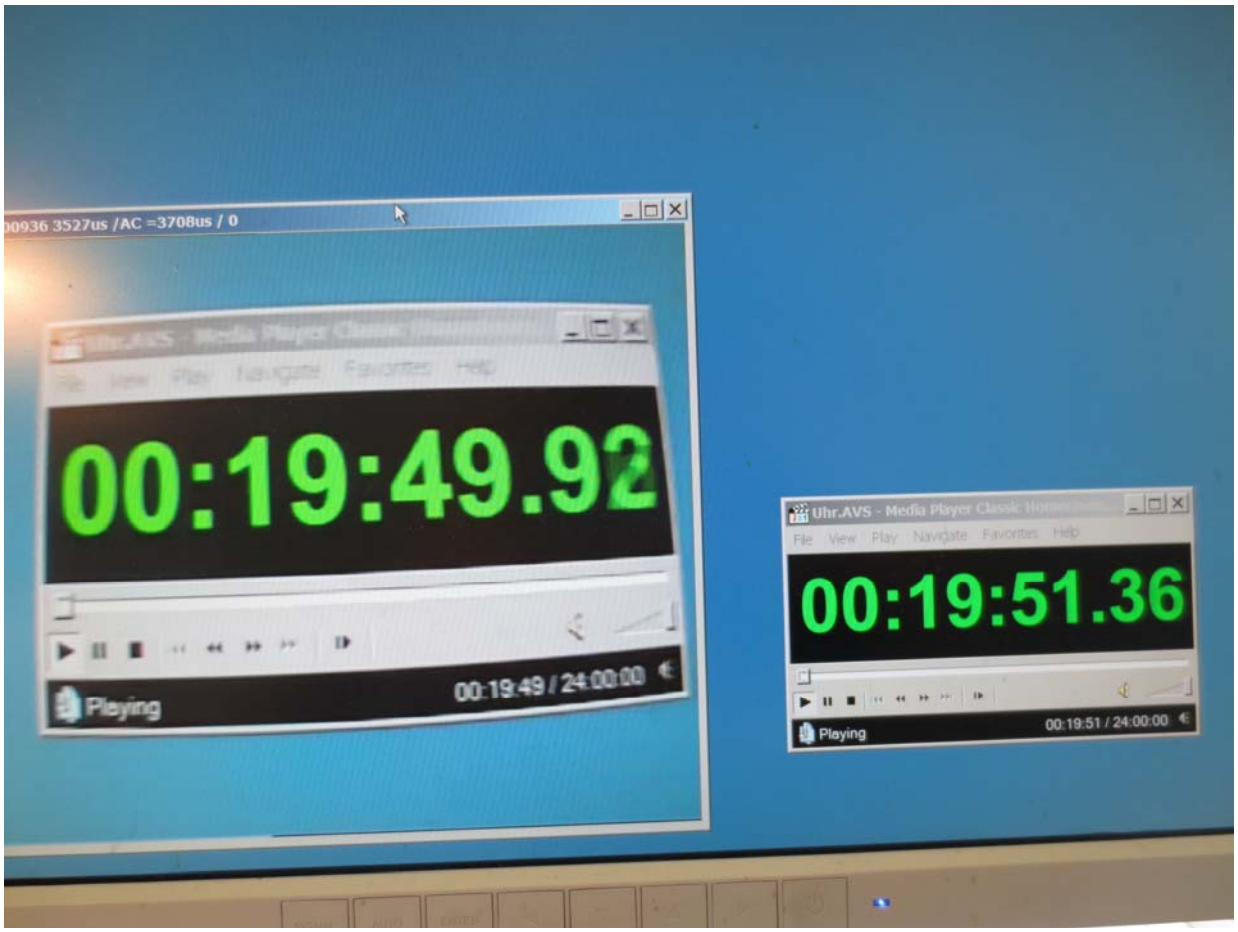


**Bild13. Messung der Zeitverzögerung zwischen Kamera- und decodiertem Videobild**

Rechts in Bild 13 ist eine, in Echtzeit laufende Uhr realisiert. Eine Kamera, die diese Uhr abfilmt, ist mit dem AXIS-Encoder verbunden. Das **Client**-Programm (s. Bild 3 und 8) übernimmt vom AXIS-Encoder die codierten Videodaten und stellt das decodierte Videobild dar (links in Bild 13). Die Differenz zwischen den beiden Uhrzeiten stellt die Zeitverzögerung dar, hier ungefähr 1,5 Sekunden.
Bei einer, im AXIS-Encoder eingestellten Bildfrequenz von 10 Hz, entsprechend 100 ms und einer GOP-Länge von 12 Bildern, entstehen mindestens $12 * 100ms = 1{,}2$ Sekunden Verzögerung durch den Videodecoder im **Client**-Programm. Da der H264-Decoder immer eine GOP sammelt bevor er decodiert (s. 3.2.2. H264 Decodierung), läßt sich diese Verzögerung nicht vermeiden.
Die restliche Verzögerung von ca. 300 ms entsteht im AXIS-Encoder, da dort vermutlich auch 2-3 Bilder gesammelt werden, bevor der Encoder codiert.
Zu dieser Verzögerung addieren sich weitere, durch die PCM-Übertragung und das Versenden der Videodatenpakete durch VuSoft hinzu.

## *4.2. Zeitverzögerung unter Einbeziehung der realen Telemetriestrecke*

Um die Zeitverzögerung des übertragenen Videos messen zu können, wurde auf einem Display im Hubschraubercockpit die GPS-Zeit dargestellt, mit einer Kamera abgefilmt und mittels Telemetrie zur Bodenstation übertragen. Die im Cockpit und der Bodenstation dargestellte Zeit ist mit einem Zeitnormal, realisiert durch eine GPS-gesteuerte Digitaluhr, verglichen worden. Die beiden Zeitdarstellungen sind in einem Video (Auflösung 30 Hz, entsprechen 33,33 ms – Bild 14 Und Bild 15) aufgezeichnet worden.
Zum Bestimmen der Zeitverzögerung betrachtet man den Zeitpunkt eines Sekundenwechsels auf der Digitaluhr.

Zuerst ist die Verzögerung der Displaydarstellung im Cockpit bezogen auf die der Digitaluhr ermittelt worden (s. Bild 14).


**Bild 14. Bestimmung der Displayverzögerung im Cockpit gegenüber der Digitaluhr**

Beim Wechseln der Sekundenanzeige von 56 sec auf 57sec auf der Digitaluhr (s. Bild 14) hat das Cockpit-Display eine Anzeige von 56,896 sec. Das Cockpit-Display eilt also zirka 0,1 sec gegenüber der Digitaluhr nach.

Bild 15 zeigt die Verhältnisse beim dekodierten Videobild in der Bodenstation

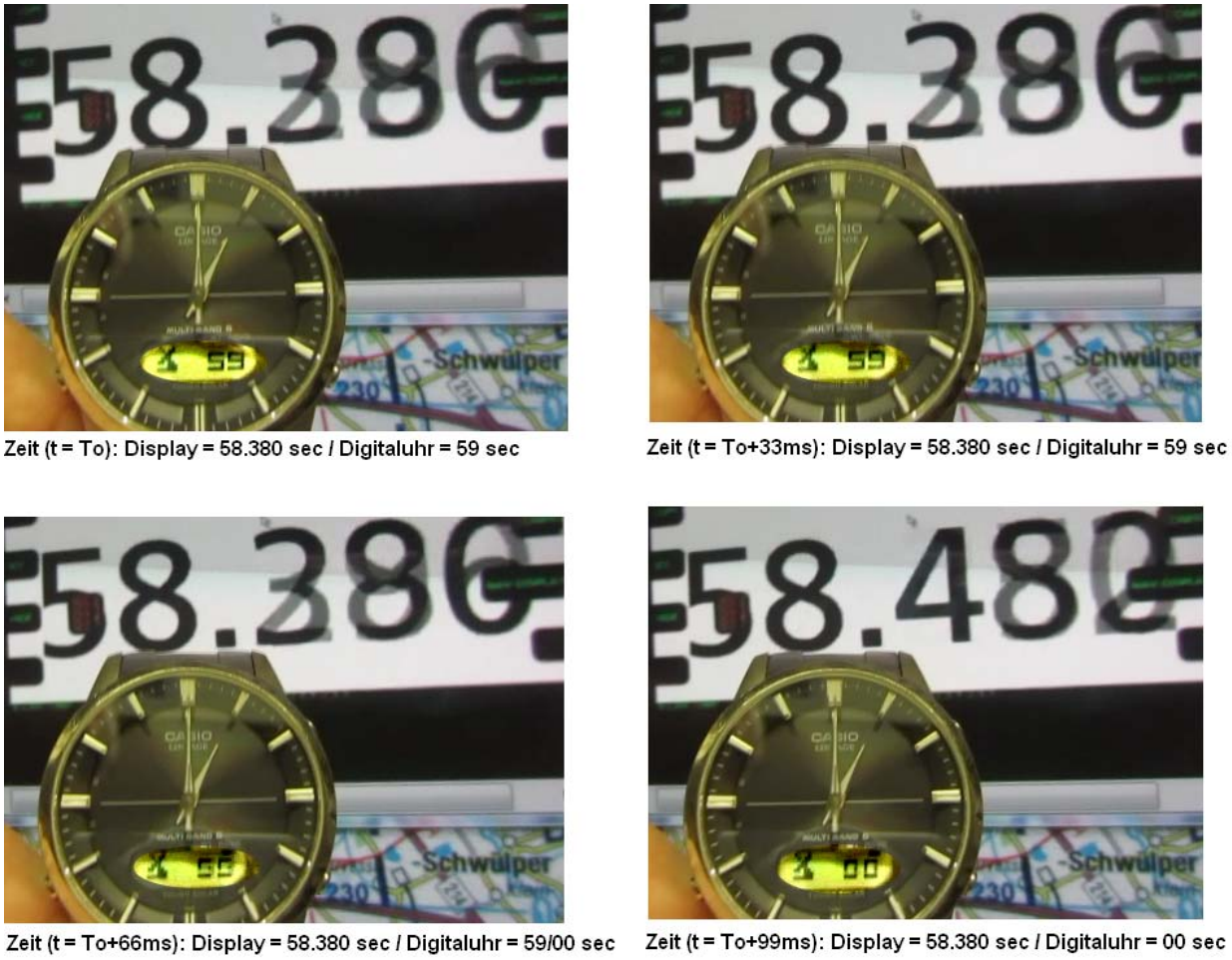


**Bild 15. Bestimmung der Verzögerung des dekodierten Videos gegenüber der Digitaluhr**

Betrachtet wird der Wechsel von 59 sec zu 00 sec auf der Digitaluhr (s. Bild 15). Das dekodierte Videobild hat zu diesem Zeitpunkt eine Anzeige von 58,480 sec. Es eilt also um zirka 1,5 sec nach, wenn man die Verzögerung von 0,1 sec aus der Cockpitmessung (Bild 14) und den Einfluß der zeitlichen Quantisierung (33,33 ms) der aufgenommenen Bilder mit berücksichtigt. Dies entspricht der im Labor gemessenen Verzögerung (s. 4.1.). Die Telemetrie-Übertragung mittels PCM hat also keinen, hier meßbaren Einfluß auf die Zeitverzögerung.

Die aufgenommen Bilder zeigen auch, daß der Wechsel der, im Display dargestellten Uhrzeit zirka alle 100 ms erfolgt. Dies entspricht der, im ASXIS-Encoder eingestellten Bildfrequenz von 10 Hz.

## *4.3. Verhalten bei Störungen*

Es ließ sich eine stabile Arbeitsweise des Videodecoders beobachten. Im Falle von Störungen kam es zwar zu Störbildern, jedoch ohne einem Beenden („Absturz“) der Decoder-Software. Am Ende der zeitlich begrenzten Störungen lieferte die Software, nach dem Empfang des nächsten vollständigen I-Frames wieder ungestörte Videobilder aus dem Cockpit (s. Bild 16). Ist beispielsweise das 1. Bild eines GOPs, also ein I-Frame gestört, endet die Störung erst mit dem Beginn der nächsten GOP. Bei den hier gewählten Parametern wären es 12 Bilder.

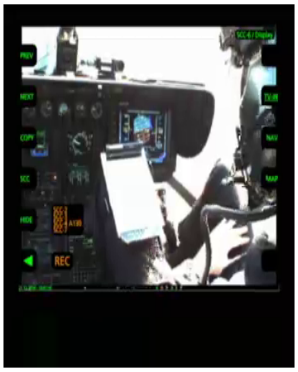

Bild 1 / T=To - ungestört

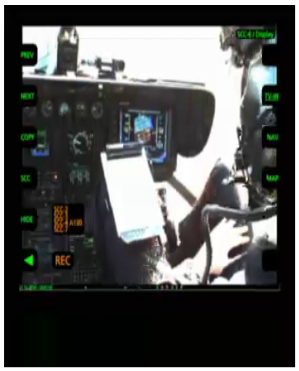

Bild 2 / T=To+100ms - ungestört

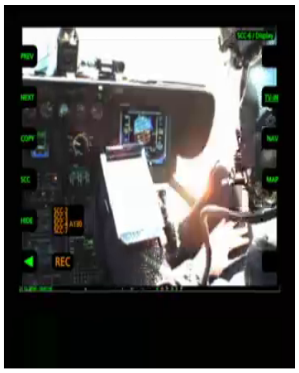

Bild 3 / T=To+200ms - ungestört

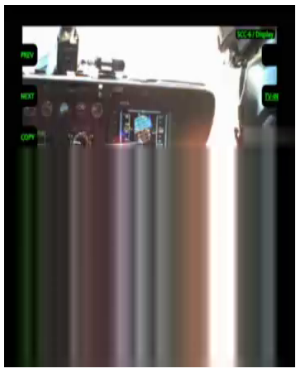

Bild 4 / T=To+300ms - gestört

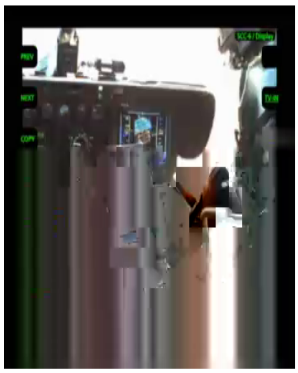

Bild 5 / T=To+400ms - gestört

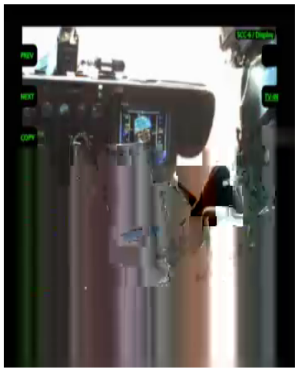

Bild 6 / T=To+500ms - gestört

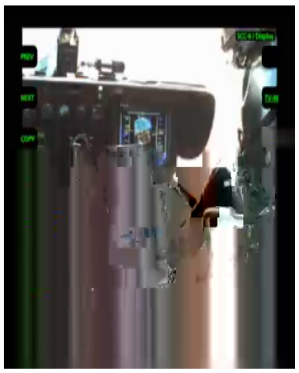

Bild 7 / T=To+600ms - gestört

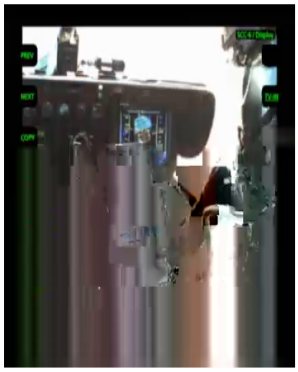

Bild 8 / T=To+700ms - gestört

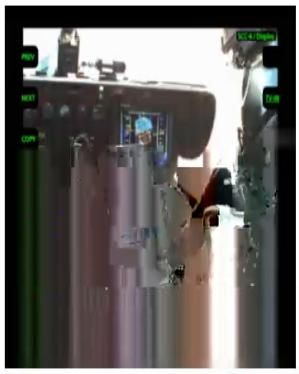

Bild 9 / T=To+800ms - gestört

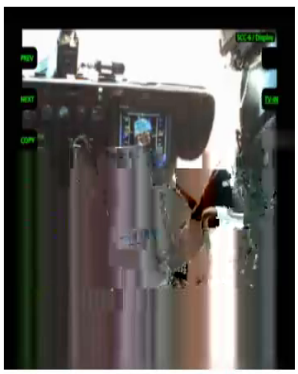

Bild 10 / T=To+900ms - gestört

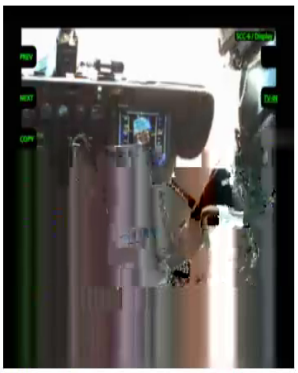

Bild 11 / T=To+1000ms - gestört

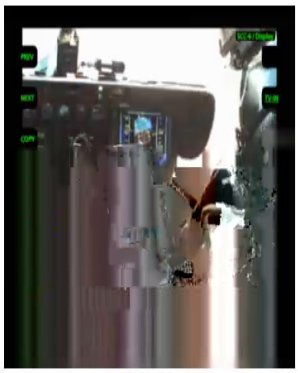

Bild 12 / T=To+1100ms - gestört

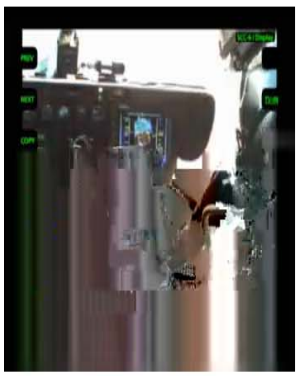

Bild 13 / T=To+1200ms - gestört

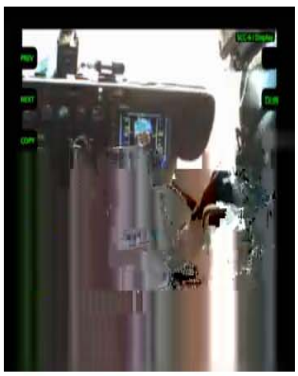

Bild 14 / T=To+1300ms - gestört

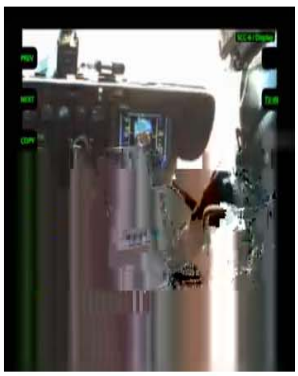

Bild 15 / T=To+1400ms - gestört

Bild 16 / T=To+1500ms - ungestört

**Bild 16. Einfluß von Störungen auf die Bilddarstellung**