Thomas M. Sailer, HB9JNX/AE4WA

6. Februar 2002

Inhaltsverzeichnis

1 Einführung

Dieses Dokument beschreibt das Datenformat und die Semantik des D-ATV-Konfigurationsfiles. Dieses Konfigurationsfile wird von fwtool analysiert und über die serielle Schnittstelle in den D-ATV-Flash-Speicher geladen.

Ein typischer Aufruf von fwtool sieht wie folgt aus:

```
fwtool -d /dev/com1 -c sample.conf -W
```

Der -d Parameter gibt den seriellen Port an, an dem der D-ATV Sender angeschlossen ist. Portbezeichnungen sind in Tabelle 1 aufgelistet:

Port	Windowsname	Linuxname
COM1	/dev/com1	/dev/tty50
COM2	/dev/com2	/dev/tty51

Tabelle 1: Portbezeichnungen

Der folgende Teil beschreibt das Konfigurationsdateiformat und die Parameter. Nicht erwähnte Parameter werden als experimentell gesehen, sie funktionieren eventuell nicht oder können jederzeit entfernt werden.

1.1 Upgraden der Microcontroller-Firmware mit fwtool

`fwtool` kann jetzt auch zum Upgraden der Firmware eingesetzt werden. Um das zu erreichen, muß zuerst das Board in den Firmwareupgrade-Mode gejumpert werden (siehe Bild 1). Dann muß das Board an die Versorgungsspannung angeschlossen und ein Reset ausgelöst werden. Der Aufruf

```
fwtool -d /dev/com1 -fm
```

läßt fwtool überprüfen, ob ein Upgrade notwendig ist und erledigt selbigen dann. Sollte der Softwaredownload in das Flash fehlschlagen, trennen Sie kurz die Versorgungsspannung ab und probieren Sie es erneut.

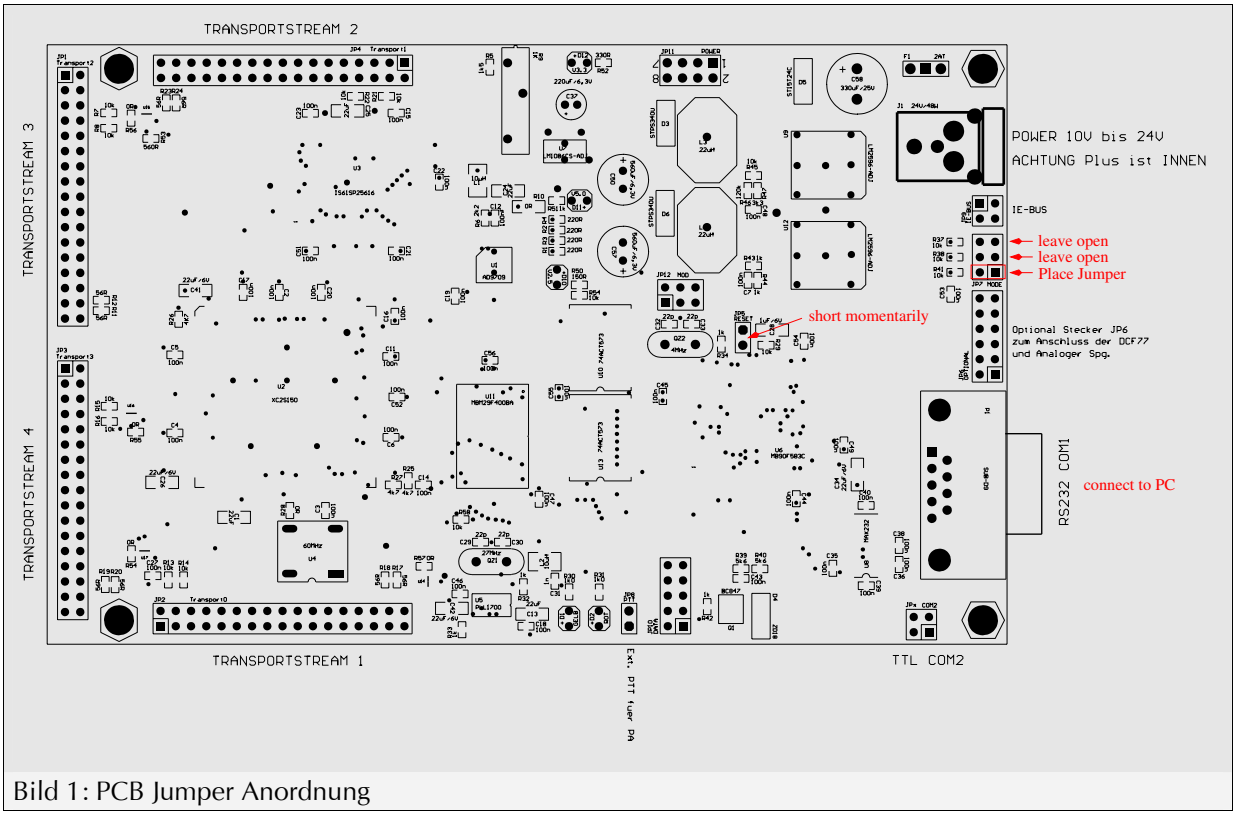

Bild 1: PCB Jumper Anordnung

1.2 Terminologie

Transport Stream

Datenstrom, bestehend aus Transport Stream-Paketen die eine beliebige Anzahl von Video-, Audio- und/oder Daten-Streams beinhalten.

Transport Stream Packet

Ein mit fester Länge von 188 Bytes definiertes Paket (4 Bytes Header, 184 Bytes Nutzdaten) welches die Video-, Audio- und andere Daten beinhaltet. Eine PID zeigt dem Empfänger an, zu welchem Daten-Stream die Information gehört.

PID

Der Packet Identifier (PID) ist ein 13-Bit-Zahlenwert (dez. 0–8191), der den Stream identifiziert, zu dem ein Transport Stream Packet gehört. PID 8191 (hex. 0x1fff) zeigt ein Packet an, welches keine sinnvollen Daten enthält. Er wird verwendet, um Transport Streams zu füllen, wenn keine sinnvollen Daten zum Senden verfügbar sind. Die PIDs 0–31 (0x00–0x1f) sind reserviert für Systemtabellen.

FEC

Forward Error Correction (FEC) fügt redundante Bits zum Sendestream hinzu, um dem Empfänger die Korrektur kleiner Übertragungsfehler zu ermöglichen.

SI Tables

System Information Tables sind Datenstrukturen, die in jedem Transport Stream vorhanden sind und dem Empfänger ermöglichen, die Programme zu finden. Sie stellen sozusagen das Inhaltsverzeichnis des Transport Streams dar.

2 Die Konfigurationsdatei

2.1 Generelle Struktur

Abb. 2 zeigt die generelle Struktur der Konfigurationsdatei. Zeilen, die mit # beginnen, sind Kommentare und werden von fwtool nicht interpretiert. Der `board`-Teil gruppiert die Parameter der D-ATV Base Band-Platine. Der `modulator`-Teil gruppiert die Modulationsparameter. Der `transportstream`-Parameter faßt eine Gruppe von Parametern zusammen, die zu dem durch die Nummer angegebenen Transport Stream-Eingang gehören. Der `teletext`-Teil schließlich beinhaltet die Parameter für den Teletext Decoder sowie die Standbild Parameter.

```
# Sample D-ATV configuration file
board {
};
modulator {
};
transportstream 1 {
};
transportstream 2 {
};
transportstream 3 {
};
transportstream 4 {
};
teletext {
};
```

Abb. 2: Generelle Struktur der Konfigurationsdatei

Die Parameter sind spezifiziert und verwenden die Syntax `Parameter = Wert;`. Zahlenwerte können dezimal integer oder hexadezimal integer mit einem *0x* Präfix sein. Ein *k*- oder *M*-Suffix multipliziert den Zahlenwert mit 1.000 bzw. 1.000.000. Textwerte müssen in doppelten Anführungszeichen eingeschlossen werden.

Abbildung 3 zeigt den Signalweg durch den D-ATV Sender. Die Base Band-Platine verbindet die Datenquellen (z.B. MPEG2-Encoder oder DVB-S-Receiver) durch die Transport Stream-Schnittstellen mit dem IQ-Modulator.

Die Transport Stream-Schnittstellen sind parallele Schnittstellen und beinhalten acht Daten-, eine Takt- und mehrere optionalen Synchronisationsleitungen.

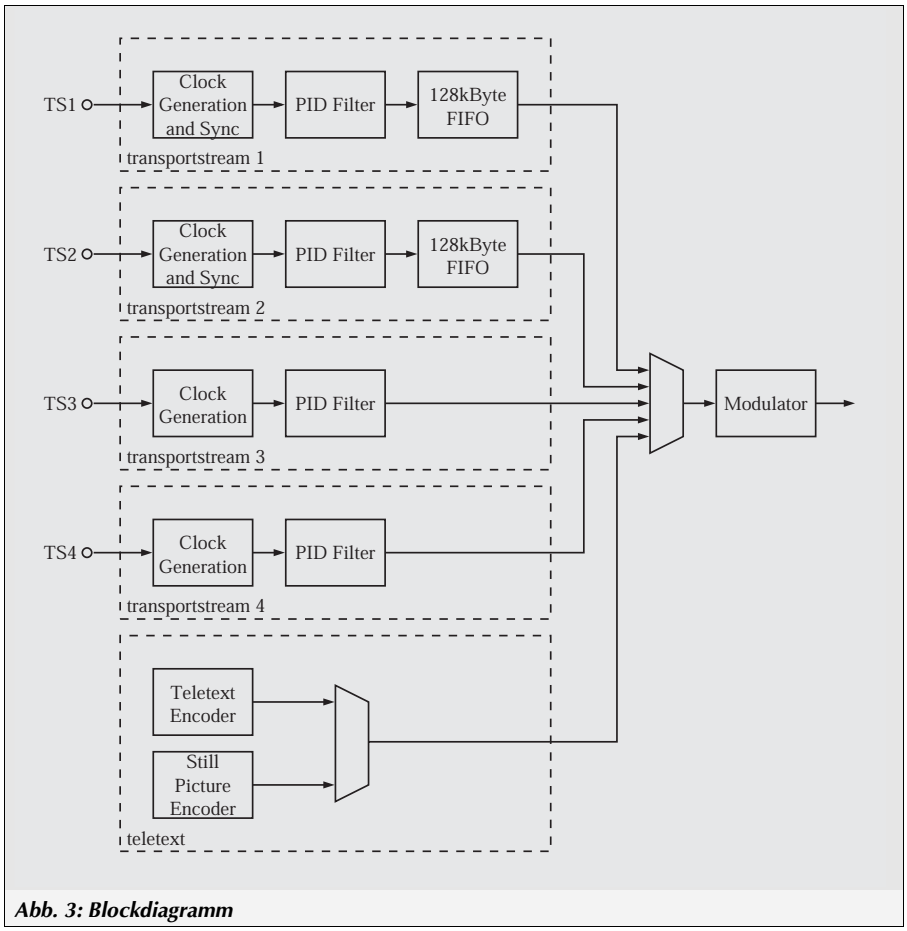

Abb. 3: Blockdiagramm

Tabelle 2 zeigt die Transport Stream Anschluss Zuordnung:

TS1 und TS2 haben beide einen FIFO im Signalweg. Die Richtung des TS Taktsignals (CK) ist deshalb konfigurierbar. Es kann entweder vom D-ATV-Board oder aus der Datenstruktur kommen.

TS3 and TS4 haben keinen FIFO. Das TS Taktsignal muss deshalb vom D-ATV Baseband-Board erzeugt werden, um Datenverlust zu vermeiden. Diese Ports werden deshalb hauptsächlich für die MPEG2 Encoder verwendet.

Signal	Pins		Signal
V5.0	1	2	V5.0
V5.0	3	4	V5.0
SDA	5	6	XERROR or IRQ VC
SCL	7	8	XRESET
GND	9	10	GND
CK	11	12	SY
VL	13	14	EN
D6	15	16	D7
D4	17	18	D5
D2	19	20	D3
D0	21	22	D1
GND	23	24	GND
SDOUT	25	26	PLLTHR
SCLK	27	28	SDIN
GND	29	30	GND
MCLKI	31	32	ASCLK
RSTDA	33	34	BCLK

Tabelle 2: Transport Stream-Anschluss

2.2 Platinenspezifischer Teil

2.2.1 Taktfrequenz

Der Taktparameter `clock` gibt die Frequenz des Quarzoszillators auf dem D-ATV Base Band-Board an. Alle Timings werden aus diesem Quarz abgeleitet. Das D-ATV-Board wird normalerweise mit 60 MHz ausgeliefert, aber für extrem niedrige Übertragungsraten kann die Quarzfrequenz reduziert werden. Das Maximum ist 62MHz.

Beispiel:
```
clock = 60000000;
```

2.3 Modulatorteil

2.3.1 Modulation

Dieser Parameter wählt den Modulationsstandard. Mögliche Werte sind *dvb-s* und *dvb-c.*

Beispiel:
```
modulation = dvb-s;
```

2.3.2 Konstellation

Dieser Parameter wählt die Konstellation. Mögliche Werte sind *qpsk* im DVB-S Modus und *qam16*, *qam32*, *qam64* im DVB-C Modus.

Beispiel:
```
constellation = qpsk;
```

2.3.3 FEC Rate, DVB-S

Dieser Parameter spezifiziert die "inner Forward Error Correction code" Rate. Er erlaubt eine Anpassung des Verhältnisses zwischen übertragener Bitrate und der Robustheit des modulierten Signals. Mögliche Werte sind: *1/2, 2/3, 3/4, 5/6, 7/8*. Dieser Parameter hat nur Bedeutung im DVB-S Modus.

Beispiel:
```
fec = 5/6;
```

2.3.4 Sendefrequenz

Der `frequency` Parameter spezifiziert die Sendefrequenz. Er muss im 70-, 23- oder 13 cm-Amateurband liegen und das HF-Modul muss das gewählte Band unterstützen.

Beispiel:
```
frequency = 1275M;
```

2.3.5 Symbol Rate, DVB-S

Der Symbolraten-Parameter spezifiziert die Bandbreite des Modulatorsignals (Gl. 1) und die Anwender-Bitrate (Gl. 2)

$$BW \approx \frac{4}{3} SR \quad (1)$$

$$BR = 2 \cdot SR \cdot R_{inner} \cdot R_{outer} \quad (2)$$

SR	Symbol Rate (Symbols/s) (siehe 2.3.5)
BW	Signal Bandwidth (Hz)
BR	User Bitrate (Bits/s)
R_{inner}	Inner FEC Rate (siehe 2.3.3)
R_{outer}	Outer FEC RATE, fixed at 188/204
F_{clk}	Crystal Oscillator Frequency (siehe 2.2.1)

Das Verhältnis F_{clk}/SR muss eines aus 4, 4⅓, 4½, 4⅔, 5, 5⅓, 5½, 6, 6½, 7, 7½, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 20,22, 24, 26, 28, 30 oder 32 sein.

fwtool rundet die Symbolrate zu dem nächsten möglichen Verhältnis.

Beispiel:
```
symbol rate = 15000k;
```

2.3.6 Symbol Rate, DVB-C

Der Symbolraten Parameter spezifiziert die Bandbreite des Modulator Signals (Gl. 3) und die Anwender-Bitrate (Gl.4)

$$BW \approx 1{,}15 SR \quad (3)$$

$$BR = \log 2q \cdot SR \cdot R_{outer} \quad (4)$$

SR	Symbol Rate (Symbols/s) (siehe 2.3.6)
BW	Signal Bandwidth (Hz)
BR	User Bitrate (Bits/s)
Q	Constellation QAMq
R_{outer}	FEC Rate, fixed at 188/204
F_{clk}	Quarzoszillatorfrequenz (siehe 2.2.1)

Das Verhältnis F_{clk}/SR muß eines von 8, 9, 10, 11, 12, 13, 14, 15 oder 16 sein. fwtool rundet die Symbol Rate zu dem nächstmöglichen Verhältnis. Es ist zu empfehlen, eine Bandbreite von 8 MHz oder weniger zu verwenden sowie eine Symbolrate von 6,9 MSymbole/sec oder weniger, da einige Empfänger auf 8 MHz begrenzt sind.

Beispiel:
```
symbol rate = 6000k;
```

2.3.7 Inversion

Dieser Parameter bestimmt, ob die I&Q Signale vertauscht werden sollen. Vertauschen der Signale erzeugt einen Effekt gleich dem Empfang eines USB Signals mit einem LSB Empfänger. Viele, aber nicht alle Empfänger[1] detektieren automatisch, ob Inversion benutzt wird. In „off“ Programmierung wird das D-ATV Signal direkt gesendet. Wenn ein spektruminvertierender Transverter benutzt wird, auf „on“ stellen.

Beispiel:
```
inversion = off;
```

[1] Die WinTV DVB-S Nova Karte mit convergence.de-Firmware scheint Inversion-Betrieb nicht automatisch zu erkennen

2.3.8 PTT

Dieser Parameter wählt aus, ob der Sender beim Einschalten der Spannungsversorgung ein- oder ausgeschaltet wird. Die PTT kann danach über das Menü geschaltet werden. Der Parameter ist typischerweise auf `on` gesetzt für D-ATV-Relais und `off` für Endstellen.

Beispiel:
```
ptt = off;
```

2.3.9 Rufzeichen des Senders

Der Netzwerkname-Parameter muss auf das Rufzeichen des Senders eingestellt werden.

Beispiel:
```
network name = "HB9W";
```

2.4 Transportstream-Teil

2.4.1 Auswahl der Eingänge

Dieser Parameter spezifiziert den Transport Stream-Portmodus. Die folgenden Modi existieren:

`off`	Port abgeschaltet
`datvencoder`	D-ATV MPEG2-Encoder an Port angeschlossen
`fujitsueval`	Fujitsu MPEG2-Encoder Evaluation Board an Port angeschlossen
`extclock`	Fremdgetaktetes Signal an Port angeschlossen.

Das "Fujitsu MPEG2 Encoder Evaluation Board" ist vergleichbar dem D-ATV Encoder-Board. Der Unterschied ist, daß das Base Band-Board nicht versucht, die MPEG2-Firmware in den Encoder zu laden. Die „`extclock`" Option kann nur an TS1 und TS2 verwendet werden.

Beispiel:
```
mode = datvencoder;
```

2.4.2 Auswahl der aktiven Taktflanke

Dieser Parameter hat nur einen Effekt wenn der Eingang im `extclock`-Modus ist. Er spezifiziert die aktive Flanke des TS Taktes (CK). Gültige Werte sind: `falling`, `rising` und `both`.

Beispiel:
```
clock edge = rising;
```

2.4.3 Clock Debounce Filter

Dieser Parameter bestimmt das Verhalten des Clock Debounce Filters. Der Wert muss auf das größte N gestellt werden, welches folgende Bedingungen erfüllt:

$$1 \le N \le 4$$

$$F_{TSCK} \le \frac{F_{clk}}{2N}$$

F_{TSCK}	Transportstreamtaktfrequenz
N	Clock Debounce Filter Parameter (siehe 2.4.3)
F_{clk}	Quarzoszillatorfrequenz (siehe 2.2.1)

Beispiel:
```
clock filter = 4;
```

2.4.4 Bitrate

Dieser Parameter bestimmt die gesamt sinnvolle Bitrate in den Port.

Beispiel:
```
bitrate = 4500k;
```

2.4.5 Auswahl des Video Einganges

Dieser Parameter hat nur einen Effekt im "`datvencoder`"-Modus. Er spezifiziert die Charakteristik des Video-Eingangssignals. Er besteht aus einem oder mehreren Schlüsselworten aus der Liste, separiert durch Kommata:

`d1`	D1-Auflösung (752x576 px.)
`hd1`	HD1-Auflösung (384x576 px.)
`sif`	SIF-Auflösung (384x288 px.)
`qsif`	QSIF-Auflösung (192x144 px.)
`ntsc`	Eingangssignal ist NTSC
`pal`	Eingangssignal ist PAL
`composite`	Benutze Composite-Eingang
`svideo`	Benutze S-Video-Eingang

Beispiel:
```
video input = d1, pal, svideo;
```

2.4.6 Video GOP-Konfiguration

Dieser Parameter spezifiziert die Bild-Encodier-Sequenz des Encoders. Der voreingestellte Modus liefert gute Encoder-Effektivität zum Preis einer höheren Encoder-Latenz (Zeitverzögerung zwischen Eingangsbild und encodiertem Ausgangsbild). Die Latenz kann durch Verkleinerung der GOP-Größe reduziert werden. Dieser Parameter sollte nur von Experten, die MPEG2-Encoderierung wirklich gut verstehen, geändert werden.

Beispiel:
```
video gop = "IBBPBBPBBPBBPBB";
```

2.4.7 Spatial Filter

Dieser Parameter stellt die Cut-Off-Frequenz des Spatialfilters sowie die Bildschärfe ein. Mögliche Werte sind `soft`, `standard` and `sharp`.

Beispiel:
```
spatial filter = standard;
```

2.4.8 Audio Encoder Bitrate

Dieser Parameter hat nur einen Effekt im "`datvencoder`"-Modus. Er spezifiziert die Bitrate des MPEG2 Layer2 Audio Encoders. Gültige Bitraten sind abhängig vom Encoder Modus (siehe 2.4.8):

Bitrate	Stereo	Joint Stereo	Dual Channel	Single Channel
`32k`	–	–	–	✓
`48k`	–	–	–	✓
`56k`	–	–	–	✓
`64k`	✓	✓	✓	✓
`80k`	–	–	–	✓
`96k`	✓	✓	✓	✓
`112k`	✓	✓	✓	✓
`128k`	✓	✓	✓	✓
`160k`	✓	✓	✓	✓
`192k`	✓	✓	✓	✓
`224k`	✓	✓	✓	–
`256k`	✓	✓	✓	–
`320k`	✓	✓	✓	–
`384k`	✓	✓	✓	–

Beispiel:
```
audio bitrate = 384k;
```

2.4.9 Audio Encoder Encoding Mode

Dieser Parameter hat nur einen Effekt im "`datvencoder`"-Modus. Er spezifiziert den Encodiermodus des

`stereo`	Eingangssignal ist ein Stereosignal
`joint stereo`	Eingangssignal ist ein Stereosignal. Der Encoder soll Redundanzen zwischen beiden Kanälen benutzen.
`dual channel`	Beide Kanäle sind unabhängig voneinander
`single channel`	Nur ein Kanal vorhanden

MPEG2 Layer2 Audio Encoders. Der Wert muss eins dieser Schlüsselwörter sein:

Beispiel:
```
audio mode = joint stereo;
```

2.4.10 Audio Encoder Sampling Rate

Dieser Parameter hat nur einen Effekt im "`datvencoder`"-Modus. Er spezifiziert die Samplerate des MPEG2 Layer2 Audio Encoders. Gültige Werte sind: `48000`, `44100`, oder `32000`.

Beispiel:
```
audio sample rate = 44100;
```

2.4.11 Program Clock Reference (PCR) PID

Dieser Parameter setzt den PID der „Program Clock Reference" (PCR). Er ist normalerweise auf den PID gesetzt, welcher den Video-Stream enthält.

Beispiel:
```
pcr pid = 0x20;
```

2.4.12 Video PID

Dieser Parameter setzt den PID des Video-Streams.

Beispiel:
```
video pid = 0x20;
```

2.4.13 Audio PID

Dieser Parameter setzt den PID des Audio-Streams.

Beispiel:
```
audio pid = 0x21;
```

2.4.14 Program Map Table (PMT) PID

Dieser Parameter setzt den PID der „program map table" (PMT). der PMT sagt dem Empfänger, welcher PID zu dem TV-Kanal gehört und verlangt seinen eigenen PID.

Beispiel:
```
pmt pid = 0x22;
```

2.4.15 Rufzeichen des Programms

Dieser Parameter setzt das Rufzeichen des TV-Kanals. Das Rufzeichen ist codiert in der SI Tabelle, welche eine Kanal-Identifizierung im Empfänger ermöglicht.

Beispiel:
```
callsign = "HB9JNX";
```

2.4.16 Sprache

Dieser Parameter identifiziert die Sprache de TV-Kanals. Er sollte auf “eng” für Englisch oder “DEU” für Deutsch gesetzt werden.

Beispiel:
```
language = "eng";
```

2.4.17 PID filter

Der PID Filter erlaubt dem Anwender, selektiv bestimmte PIDs auf dem entsprechenden TS Eingang zu blockieren. Zwei Strategien sind möglich:

1. Alle PIDs erlauben, Auflisten der zu blockierenden PIDs
2. Blockiere alle PIDs als Voreinstellung, Auflisten der erlaubten PIDs

Voreinstellung ist “alle” oder “keine”, die Ausnahmen sind mit `minus pid/mask` und `plus pid/mask` markiert. `pid` spezifiziert das passende Zahlwort und `mask` spezifiziert das zu vergleichende Bitmuster. Der resultierende PID Filter muss den PID 8191 (0x1fff), (das NullPacket PID) blockieren.

Beispiele:
```
pidfilter = all minus 0x1ffe/0x1ffe;
pidfilter = none plus 0x0020/0x1ffe;
```

2.4.18 Tunermodus

Dieser Parameter spezifiziert, ob ein Empfänger an dem Port angeschlossen ist. Folgende Tunermodi bestehen:

`off`	Kein Tuner angeschlossen
`dfm`	DFM-Analog-Tuner angeschlossen am D-ATV MPEG2-Encoder
`mb86a15`	Fujitsu DVB-S-Tuner direkt am TS-Port angeschlossen

Beispiel:
```
tuner mode = off;
```

2.4.19 Tunerfrequenz

Dieser Parameter hat nur einen Effekt, wenn der Tuner Modus nicht auf `off` gesetzt ist. Er gibt die Frequenz an, auf die der Tuner eingestellt werden soll.

Beispiel:

```
tuner frequency = 1260M;
```

2.4.20 Tuner FEC Modus

Dieser Parameter hat nur einen Effekt, wenn der Tuner Modus auf (siehe 2.4.18) mb86a15 gesetzt ist. Er spezifiziert, welche inneren FEC Einstellungen getestet werden sollen, bis ein gültiges Signal gefunden wird. Er kann entweder `auto` oder auf eine innere FEC Rate gesetzt werden.

Beispiele:
```
tuner fec = auto;
tuner fec = 1/2;
```

2.4.21 Tuner Symbol Rate

Dieser Parameter hat nur einen Effekt, wenn der Tuner Modus auf (Siehe 2.4.17) mb86a15 gesetzt ist. Er spezifiziert, welche Symbolrate erwartet werden soll.

Beispiel:
```
tuner symrate = 3000k;
```

2.5 Teletext-Teil

2.5.1 Program Clock Reference (PCR) PID

Dieser Parameter setzt den PID des "Program Clock Reference (PCR)". Er ist normalerweise auf den PID, welcher den Video-Stream beinhaltet, gesetzt.

Beispiel:
```
pcr pid = 0x20;
```

2.5.2 Video PID

Dieser Parameter wählt den PID des Video-Streams, welcher das Standbild enthält.

Beispiel:
```
video pid = 0x20;
```

2.5.3 Teletext PID

Dieser Parameter wählt den PID des Teletext-Streams.

Beispiel:
```
teletext pid = 0x21;
```

2.5.4 Program Map Table (PMT) PID

Dieser Parameter setzt den PID der "program map table (PMT)". Der PMT sagt dem Empfänger, welcher PID zu dem TV-Kanal gehört und verlangt seinen eigenen PID.

Beispiel:
```
pmt pid = 0x22;
```

2.5.5 Program Callsign

Dieser Parameter setzt das Rufzeichen des Teletext/Standbild-Kanals. Das Rufzeichen ist codiert in der SI-Tabelle welche eine Kanalidentifizierung im Empfänger ermöglicht.

Beispiel:
```
callsign = "HB9JNX";
```

2.5.6 Spracheinstellung

Dieser Parameter identifiziert die Sprache des Teletext/Standbild-Kanals . Er sollte auf "eng" für Englisch oder "DEU" für Deutsch gesetzt werden.

Beispiel:
```
language = "eng";
```

2.5.7 Bild-Dateien

Dieser Parameter spezifiziert die Datei, welche das Bild zum Übertragen im Standbild-Kanal beinhaltet. Der Kanal ist dazu bestimmt z.B. ein Logo des Operators anzuzeigen. Beachten Sie, daß dieser Kanal nicht 100% DVB-kompatibel ist, so daß keine Garantie gegeben werden kann, daß alle Empfänger diesen Kanal anzeigen können. Die Datei muß entweder vom JPEG-Typ sein oder einen MPEG2 „elementary stream" beinhalten. MPEG2-Softwareencoder erzeugen üblicherweise Stream-Dateien, die nicht kompatibel sind. Wenn die Datei ein JPEG Bild (muss 704x576 Pixel groß sein) enthält, kann mittels des mpeg2enc-Programms von mjptools ein kompatibler „elementary stream" erzeugt werden. Die Binärdatei muss im selben Verzeichnis wie fwtools unter Windows oder im usr/bin/ Verzeichnis unter Linux zu finden sein.

Beispiel:
```
picture file = "mylogo.jpg";
```

2.5.8 VM Code

Dieser Parameter spezifiziert die Datei, welche den teletext encoder virtual machine bytecode enthält. Nähere Informationen zum Teletext-Encoder VM-Bytecode siehe Kapitel 5.

Beispiel:
```
vm code = "teletext.o";
```

2.6 Extern-Programm-Teil

Der D-ATV-Sender kann nicht nur lokal encodierte Programme senden, sondern auch Programme von anderen Quellen wie z.B. DVB-S-Receivern oder PCs. Um Receivern das Auffinden solcher Programme zu ermöglichen, muß der D-ATV-Sender die Program Map Tables (PMT) dieser Programme ebenfalls mitsenden und sie in der Program Association Table (PAT) aufführen

2.6.1 Program Clock Reference (PCR) PID

Dieser Parameter enthält die PID der Program Clock Reference (PCR). Er wird normalerweise auf die PID gesetzt, die den Video-Stream enthält.

Beispiel:
```
pcr pid = 0x420;
```

2.6.2 Program Map Table (PMT) PID

Dieser Parameter enthält die PID der Program Map Table (PMT). Die PMT enthält Informationen für den Receiver, welche PID welchen TV-Kanal enthält und besitzt selbst eine PID.

Beispiel:
```
pmt pid = 0x422;
```

2.6.3 Sprache

Dieser Parameter gibt die Sprache des TV-Kanals an. Er sollte auf „eng“ für englisch und „DEU“ für deutsch gesetzt werden.

Beispiel:
```
language = „eng“;
```

2.6.4 Stream subsections

Die Stream subsections (etwa: Streamunterteilungen) (`video stream`, `audio stream`, `teletext stream`, `stream`) entsprechen den individuellen Strömen, die zusammen ein Programm ergeben.

2.6.5 PID

Dieser Parameter gibt die Stream-PID an.

Beispiel:
```
pid = 0x440;
```

2.6.6 Stream Typ

Dieser Parameter gibt den Typ des Streams an (siehe [1, Tabelle 2-36]). Dieser Parameter kann nicht manuell für Video-, Audio- und Teletext-Streams gesetzt (dies geschieht automatisch).

Beispiel:
```
stream type = 0x80;
```

2.6.7 Stream ID

Dieser Parameter setzt die ID des Streams, es handelt sich um die Nummer des Streams.

Beispiel:
```
stream id = 1;
```

2.6.8 Component Type

Dieser Parameter setzt den Component Type des Streams (siehe [2, Tabelle 24]).

Beispiel:

```
component type = 1;
```

2.6.9 Language

Dieser Parameter gibt die Sprache des Streams an. Er sollte auf „eng“ für englisch oder „DEU“ für deutsch gesetzt werden.

Beispiel:

```
language = „eng“;
```

3 Der alte Teletext-Encoder

Der alte Teletext-Encoder besteht aus einer statischen Tabelle von Teletextseiten mit ihren Zeilen. Er erlaubt wenig dynamische Inhalte und keine Kontrolle über den Encodierprozess. Er wird vermutlich in Zukunft aus der Konfiguration entfernt. Abb. 3 zeigt Konfigurationsanweisungen einer beispielhaft zu encodierenden Teletextseite.

```
teletext {
   page header = "www.D-ATV.de \x92\x20\x08";
   page {
      number = 100;
      line 1 = "";
      line 2 = "\x01 www.D-ATV.de";
      line 3 = "";
      line 4 = "Digital Baseband:";
      line 5 = " Thomas Sailer, HB9JNX/AE4WA";
      line 6 = "";
      line 7 = "RF";
      line 8 = " Wolf-Henning Rech, DF9IC/N1EOW";
      line 9 = " Jens Geisler, DL8SDL";
      line 10 = "";
      line 11 = "Schematics, Boards &";
      line 12 = " Connections to Fujitsu";
      line 13 = " Stefan Reimann, DG8FAC";
      line 14 = "";
      line 15 = "\x03adacom e.V.";
   };
};
```

Abb. 4 Alter Teletext Encoder Konfigurationsdatei

3.1 Der Teletext-Teil

3.1.1 Teletext Seitenkopf

Dieser Parameter setzt die Inhalte des Seitenkopfes (oberste Textzeile), der rechts von der Seitenzahl angezeigt wird.

Beispiel:
```
page header = "www.D-ATV.de \x92\x20\x08";
```

3.2 Teletextseiten

Der `teletext`-Teil enthält Seitenuntergliederungen, startend mit einer öffnenden Klammer `{` und endend mit der Zeichenfolge `};` Jede dieser Bereiche beschreibt jeweils eine einzelne Teletextseite.

3.2.1 Page Number (Seitenzahl)

Dieser Parameter spezifiziert die Teletextseitenzahl. Der Wert muss zwischen 100 und 899 (jeweils inklusive) sein. Teletextdecoders beginnen automatisch mit Seite 100, deshalb sollte Seite 100 vorhanden und mit Einführungsinformation versehen sein.

Beispiel:
```
page number = 100;
```

3.2.2 Teletext Lines (Teletextzeilen)

Dieser Parameter spezifiziert die einzelnen Zeilen einer Teletextseite. Die Zeilen sind von 1-24 durchnummeriert. Teletextzeilen können bis zu 40 Zeichen lang sein. Kürzere Zeilen werden mit Leerzeichen aufgefüllt. Nichtdruckbare Zeichen können durch Eingabe eines „Backslash“, gefolgt von “x”, und einer zwei Digit langen Hexadezimalzahl, welche den Zeichencode beinhaltet. Um zum Beispiel das Zeichen 1 (0x01) einzugeben, muß im File „`\x01`“ stehen. Die Zeichencodes 0–31 (0x00–0x1f) werden für *ETSI Teletext Attribute markup* verwendet (z.B. Farben), die Zeichencodes 128–255 (0x80–0xff) um dynamische Daten einzugeben, wie etwa Packet Counters.

Beispiel:

```
line 2 = " \x01 www.D-ATV.de";
```

4 Beispielkonfiguration

Abb. 4 zeigt eine einfache, minimale Konfigurationsdatei. Es wird angenommen, daß ein MPEG2 Encoder an TS1angeschlossen ist und TS2-TS4 nicht benutzt werden.

```
# Minimal D-ATV configuration file
board {
      clock = 60000000;
};

modulator {
      fec = 2/3;
      frequency = 1275M;
      symbol rate = 3750k;
      network name = "HB9JNX";
};

transportstream 1 {
      mode = datvencoder;
      bitrate = 4500k;
      callsign = "HB9JNX";
      language = "eng";
};

transportstream 2 {
      mode = off;
};

transportstream 3 {
      mode = off;
};

transportstream 4 {
      mode = off;
};

teletext {
      callsign = "HB9JNX";
      language = "eng";
      picture file = "mylogo.jpg";
      vm code = "teletext.o";
};
```

Abb. 5: Einfache Konfigurationsdatei

5 Der neue Teletext-Encoder

Der neue Teletext-Encoder erlaubt die volle Kontrolle des Encoding-Prozesses und dynamische Inhalte. Er wird ausgeführt mittels eines Bytecode-Programms, das in einer stack-basierenden, virtuellen Maschine interpretiert wird. Telecode-Teletextprogramme müssen nicht in der stack-basierten Assembler-Sprache der virtuellen Maschine geschrieben werden, sondern können in C programmiert und dann in einen Bytecode kompiliert werden. Die folgende Tabelle zeigt die Executables des Bytecode-Entwicklungssystems:

`cpp`	C Preprocessor
`rcc`	C Compiler proper
`vm`	VM Simulator
`vmar`	Bytecode Archiver
`vmas`	Bytecode Assembler
`vmdisass`	Bytecode Disassembler
`vmld`	Bytecode Linker
`atv2txtvm`	Konvertierungstool von DG9MHZ-ATV-Files zu VM Teletext Source Code.

Angenommen, der Teletext Encoder-C-Code ist in der Datei `teletext.c` beinhaltet, kann der C-Code durch folgendes Kommando kompiliert und assembliert werden in die Objectcode-Datei `Teletext.o`:

```
vmas -c -o teletext.o teletext.c
```

Der Objektcode kann disassembliert werden durch:

```
vmdisass teletext.o
```

Der Objektcode kann simuliert werden durch:

```
vm -c -1 -m teletext teletext.o
```

Abb. 6 zeigt ein Beispiel des Sourcecodes eines Teletext Encoders:

```
/* sample teletext encoder */
#include "dvbs.h"
static const char pg_header[] = TXT_ARG0 " www.D-ATV.de " TXT_ARG1;

static const char *pg_100[] = {
      pg_header,
      NULL,
      TXTATTR_ALPHA_RED " www.D-ATV.de",
      NULL,
      "Digital Baseband:",
      " Thomas Sailer, HB9JNX/AE4WA",
      NULL,
      "RF",
      " Wolf-Henning Rech, DF9IC/N1EOW",
      " Jens Geisler, DL8SDL",
      NULL,
      "Schematics, Boards &",
      " Connections to Fujitsu",
      " Stefan Reimann, DG8FAC",
      NULL,
      TXTATTR_ALPHA_YELLOW "adacom e.V.",
      NULL,
      NULL,
```

```
	NULL,
	NULL,
	NULL,
	NULL,
	NULL,
	NULL,
	NULL
};

void teletext(void)
{
	char t[9];
	for (;;) {
		timedec(t, NULL, gettime());
		teletext_encodepage(0, 24, 0x100, 0, 0, pg_100, "100", t);
		teletext_encodepage(0, 0, 0x1ff, 0, 0, pg_100, "100", t);
	}
}
```

Abb 6: Beispiel Teletext Encoder Source Code

DG9MHZ ATV Dateien[2] können in VM Teletext Objektcode umgewandelt werden durch:

```
atv2txtvm -c -o teletext.o -i "D-ATV" -p 10 100_0000.ATV 101_0000.ATV
```

ATV Dateien können mit vtedit geschrieben werden[4].

5.1 C-Code

Die im Header angegebene Datei `dvbs.h` beinhaltet Prototypen für die eingebauten Libraryfunktionen. Die VM startet den Teletext Encoder durch Aufruf der Funktion `Teletext` mit dem prototype `void teletext(void)`

5.2 VM Built-In Library Functions

5.2.1 C Typ-Größen

Typ	Bits
char	8
short	16
int	32
long	32

5.2.2 C99 Standard Makros

`NULL, offsetof`

5.2.3 C99 Standard Typen

`ptrdiff_t, size_t, int8_t, u_int8_t, int16_t, u_int16_t, int32_t, u_int32_t`

5.2.4 C99 Standard Funktionen

`memcpy, memmove, strcpy, strncpy, strcat, strncat, memcmp, strcmp, strncmp, memchr, strchr, strcspn, strpbrk, strrchr, strspn, strstr, memset, strlen, exit`

5.2.5 Event-Log-Funktionen

`void logreadinit(unsigned int *p);`	
Setzt den Event-Log-Zeiger auf die älteste Nachricht im Buffer.	
`p`	a pointer to an opaque cookie of type `unsigned int`

`unsigned int logreadline(unsigned int *p, char *buf, unsigned int bufsz);`	
Liest die nächste Event-Log-Nachricht ein.	
`p`	Zeiger auf einen opaque cookie des Typs `unsigned int`
`buf`	Zeiger auf einen Buffer ausreichender Größe
`bufsz`	die Größe des Buffers
`logreadline`	Gibt die Anzahl der Zeichen zurück, die im Buffer gespeichert sind und nicht Null sind. Ein Rückgabewert von Null bedeutet, dass keine weiteren Ereignisse im Event-Log-Buffer mehr vorhanden sind.

5.2.6 Zeit- und Datumsfunktionen

struct time	
`day`	Modifiziertes julianisches Datum (Anzahl der Tage seit dem 17.11.1858.
`sec`	Anzahl der Sekunden seit Mitternacht
`msec`	Anzahl der Millisekunden der zuletzt angefangenen Sekunde
`valid`	wenn valid gesetzt ist, wurden Zeit und Datum per serieller Schnittstelle oder DCF77 eingestellt.

struct timehms	
`h`	Stunden
`m`	Minuten
`s`	Sekunden

struct date	
`d`	Tag
`m`	Monat
`y`	Jahr

`struct time gettime(void);`
gibt die aktuelle Zeit zurück

`u_int32_t getjiffies(void);`
gibt eine monoton steigende Nummer zurück. Sie erhöht sich HZ mal pro Sekunde.

`struct date mjdtodate(u_int16_t mjd);`
konvertiert ein modifiziertes julianisches Datum in ein Standard-gregorianisches Datum

`u_int16_t datetomjd(u_int16_t d, u_int16_t m, u_int16_t y);`
konvertiert ein Standard-gregorianisches Datum in ein modifiziertes julianisches Datum.

`char *timedec(char *buf, struct timehms *hms, u_int32_t tm);`
nimmt die Sekunden seit Mitternacht und konvertiert sie in Stunden, Minuten und Sekunden und in eine menschenlesbare Form 01:23:45 *hms* und *buf* sollten NULL sein. Die Funktion gibt einen Zeiger auf *buf[0]* zurück.

5.2.7 Parameter und Statistik-Funktionen

```
u_int16_t getadc(unsigned int n);
```
Gibt den Wert des A/D-Wandler n zurück. n reicht von 0 bis 3, und der Rückgabewert des 10-Bit-A/D-Wandlers bewegt sich zwischen 0 und 1023, entsprechend einer Eingangsspannung von 0 bis 5 Volt.

```
u_int32_t readcounter(unsigned int n);
```
Gibt den Wert des Zählers n zurück

n	Wert
0	Local PCR (Program Clock Reference)
1	Total packet count
2	Mux-generated NULL packets
3	Table/Teletext packets
4	Transport Stream 1 packets
5	Transport Stream 2 packets
6	Transport Stream 3 packets
7	Transport Stream 4 packets
8-15	unused

```
u_int8_t get_inversion(void);
```
Gibt die “spectral inversion“-Einstellung zurück

```
u_int8_t get_fecmode(void);
```
Gibt den FEC-Modus zurück.

Rückgabewert	FEC-Modus
0	1/2
1	2/3
2	3/4
3	5/6
4	7/8

```
u_int32_t get_frequency(void);
```
Gibt die Übertragungsmittenfrequenz in kHz.

```
u_int8_t get_ptt(void);
```
returns whether the PTT is keyed.

5.2.8 Numeric to String conversion

flags	
INTCONV_SIGN	number is signed
INTCONV_PLUS	write an explicit + if a signed number is positive
INTCONV_PADZERO	pad buffer to the left with Zeros
INTCONV_PADSPACE	pad buffer to the left with spaces
INTCONV_LOWERCASE	use lower case hexadecimal characters

`char *int2hex(char *buf, u_int16_t len, u_int32_t val, u_int16_t flags);`
konvertiert `val` in einen Dezimalstring, der in den Buffer `buf` gespeichert wird. Bis zu `len` Zeichen werden gespeichert, `buf` muß mindestens `len+1` Zeichen groß sein. Die Funktion gibt einen Zeiger auf den Nummernstring zurück, der sich in `buf` befindet, jedoch nicht notwendigerweise auf dessen Anfang.

`char *int2dec(char *buf, u_int16_t len, u_int32_t val, u_int16_t flags);`
konvertiert `val` in einen Hexadezimalstring, der in den Buffer `buf` gespeichert wird. Bis zu `len` Zeichen werden gespeichert, `buf` muß mindestens `len+1` Zeichen groß sein. Die Funktion gibt einen Zeiger auf den Nummernstring zurück, der sich in `buf` befindet, jedoch nicht notwendigerweise auf dessen Anfang.

5.2.9 TS1/TS2 table decoder

The TS1/TS2 table decoder tries to extract data from the System Information tables received on transport stream ports 1 and 2.

struct portcapture	
`event_id`	wird inkrementiert, wenn die Service Descriptor Information Table ein Update erfährt.
`transport_stream_id`	Transport Stream ID
`nit_pid`	PID, mit der die Network Information Table übertragen wird.
`service_id`	Service ID
`network_id`	Network ID
`service_provider_name`	Service Provider Name
`service_name`	Service Name
Für nähere Informationen über die DVB System Information (SI) tables, siehe [2].	

`struct portcapture getcapture(unsigned int port);`
returns SI table data for transport stream port.

port	Transport Stream
0	TS1
1	TS2

5.2.10 Highlevel Teletext Enconding functions

`void teletext encodepage(u_int16_t startline, u_int16_t endline, u_int16_t pagenr, u_int16_t subnr, u_int32_t flags, const char **lines, ...);`	
encodiert mehrere Teletextzeilen, von `startline` bis `endline`	
`startline`	ist normalerweise 0
`endline`	ist normalerweise 24
`pagnr`	spezifiert eine Seitenzahl und sollte zwischen 0x100 und 0x8ff liegen. Seitenzahlen, die Hexadezimalziffern A-F enthalten, sind über den Receiver normalerweise nicht direkt erreichbar.
`subnr`	spezifiert eine Unterseitenzahl (normalerweise 0)
`flags`	can be zero or multiple TXTPAGECTRL macros ored.
`TXTPAGECTRL`	sind detailliert beschrieben bei [3, 9.3.13, p.27].
`lines`	enthält einen Zeiger auf ein Array von `endline-startline+1` Strings. Jeder String spezifiziert den Inhalt jeweils einer Zeile. Ein `NULL`-Pointer unterdrückt das encodieren der zugehörigen Zeile. Teletextzeilen können auch `TXTATTR`-Makros oder `TXT_ARGn`-Argumente enthalten.
`TXTATTR`	Makros (siehe [3, 12.2, p. 76-80])
`TXT_ARGn`	referenziert auf optionale Argumente. Kann bis zu 64 Zeiger beinhalten, die auf Strings verweisen.

5.2.11 Lowlevel Teletext Enconding functions

```
void teletext oddparity(u_int8_t *buf, const u_int8_t *src, unsigned int
len);
```

encodiert einen Datenbuffer, beginnend bei `src` mit der Länge `len`, Teletext-ungleich-Parität benutzend, und speichert es in `buf`.

```
void teletext hamming84(u_int8_t *buf, const u_int8_t *src, unsigned
int_nibblelen);
```

encodiert einen Datenbuffer, beginnend bei `src`, der `len` Nibbles mit Teletext-8/4-Hamming-Code in `buf` speichert. Zuerst wird das niederwertige Nibble von `src[0]` codiert, dann das höherwertige Nibble von `src[0]`, dann das niederwertige von `src[1]`, und so weiter.

```
void teletext hamming2418(u_int8_t *buf, const u_int8_t *src, unsigned int
len);
```

Encodiert einen Datenbuffer, beginnend bei `src`, der `len` Triples mit Teletext-24/18-Hamming-Code in `buf` speichert. `src[0]` enthält dabei die niederwertigen 6 Bits, `src[1]` die mittleren 6 Bits und `src[2]` die höherwertigen 6 Bits.

```
u_int8_t *teletext currentline(void);
```

Gibt einen Zeiger auf den Buffer der aktuellen Zeile zurück. Der Zeilenbuffer ist 42 Bytes groß und enthält eine komplette Teletext-Zeile ohne clock run-in und framing-code [3, 7.1, p.17ff].

```
u_int8_t *teletext waitline(void);
```

Sendet die aktuelle Zeile und gibt einen Zeiger auf den Buffer der nächsten Zeile zurück.

6 Anschluss eines PC Parallel Port an einen Transport Stream input

Der Parallelport eines PC kann als eine einfache Möglichkeit für einen langsamen Datentransfer in den Transport Stream benutzt werden. Bis zu 2 MBit/s sind möglich. Tabelle 3 zeigt, wie der Parallelport an TS1 oder TS2 angeschlossen werden muß. Der Eingangsport muss auf `extclock` Modus eingestellt und der TT clock filter sollte auf das Maximum 4 eingestellt werden.

Pin	Parport-Signal	TS-Signal
1	nStrobe	CK
2	D0	D0
3	D1	D1
4	D2	D2
5	D3	D3
6	D4	D4
7	D5	D5
8	D6	D6
9	D7	D7
10	nAck	nStrobe
11	Busy	ASCLK
12	Perror	SCLK
13	Select	SDIN
14	nAutoFd	SY
15	nFault	XRESET
16	nInit	VL
17	nSelectIn	EN
18…25	GND	GND

Tabelle 3: Parallelport Anschlußschema

7 Danksagung

Die Microcontroller Firmware enthält uIP, copyright (c) 2001, Adam Dunkels.

Der VM Bytecode C compiler basiert auf LCC, geschrieben von Chris Fraser und David Hanson. LCC Sourcen sind auf der LCC Homepage kostenlos erhältlich[4].

7.1 Referenzen

[1] ETSI EN 300 468 V1.4.1 European Standard (Telecommunications series) Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems, 07 2000.

[2] Detlef Fliegl, DG9MHZ. VTGEN, der Teletextencoder für IBM-kompatible PCs. *http://www.baycom.org/ftp/local/vt/ { vtpack.exe,vtgendoc.zip }* , November 1995.

[3] European Telecommunications Standards Institute (ETSI). ETS 300 706: Enhanced Teletext Specification, May 1997.

[4] Chris Fraser and David Hanson. lcc, A Retargetable Compiler for ANSI C. *http://www.cs.princeton.edu/software/lcc/*