D-ATV Transmitter Configuration File

Thomas M. Sailer, HB9JNX/AE4WA

6th February 2003

Contents

1 Introduction

This document describes the file format and the semantics of the D-ATV Transmitter Configuration File. This configuration file is parsed by the `fwtool` utility and then written to the D-ATV transmitter onboard flash memory through the serial interface.

A typical invocation of fwtool looks as follows:

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

The `-d` Parameter indicates the serial port to which the D-ATV transmitter is connected. Port names are listed in table 1.

Port	Windows Name	Linux Name
COM1	/dev/com1	/dev/ttyS0
COM2	/dev/com2	/dev/ttyS1

Table 1: Port Names

The following sections describe the configuration file format and the parameters. Parameters not documented herein are considered experimental and may there not work or be removed anytime.

1.1 Upgrading the Microcontroller Firmware with `fwtool`

`fwtool` can now be used to upgrade the microcontroller firmware as well. To do this, you must first jumper the board in firmware upgrade mode. Figure 1 illustrates this. Then you need to power the board and momentarily short the reset pins to reset the board. Invoking

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

should then check if an upgrade is necessary and do it if it is. Should downloading the flash programming firmware fail, recycle the board power and try again.

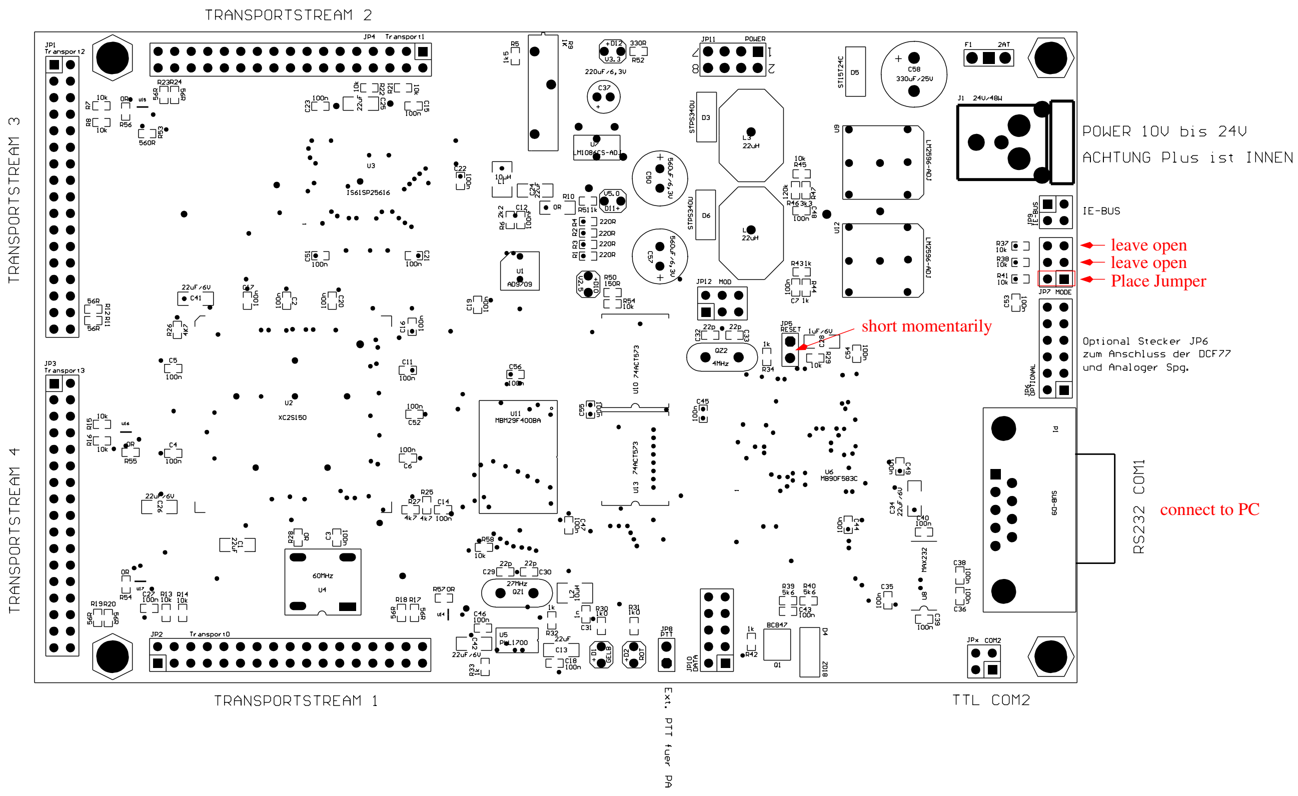

Figure 1: PCB Jumper Locations

1.2 Terminology

Transport Stream Data stream consisting of transport stream packets and containing any number of video, audio and/or data streams.

Transport Stream Packet A fixed length 188 byte packet (4 bytes header, 184 bytes useable data) that carries the video, audio and other data. A PID indicates to the receiver to which stream the data belongs to.

PID The Packet IDentifier (PID) is a 13 bit number (0–8191) that identifies the stream to which a Transport Stream Packet belongs to. PID 8191 (hexadecimal 0x1fff) indicates a packet that does not contain useful data. It is used to fill transport streams when there is no useful data to transmit. PIDs 0–31 (0x00–0x1f) are reserved for system tables.

FEC Forward Error Correction (FEC) adds redundant bits at the transmitter to enable the receiver to correct a few transmission errors.

SI Tables System Information (SI) Tables are data structures present in every transport stream that enable the receiver to find the programs. They are essentially the table of contents of the transport stream.

2 The Configuration File

2.1 General Structure

Figure 2 shows the general structure of the configuration file. Lines starting with a hash mark `#` are comment lines and are not interpreted by `fwtool`. The `board` section groups parameters belonging to the D-ATV baseband board. The `modulator` section groups modulation parameters. The `transportstream` parameters group parameters belonging to the numbered transport stream input. Finally, the `teletext` section contains the parameters for the teletext encoder and the still picture transmitter.

```
# Sample D-ATV configuration file

board {

};

modulator {

};

transportstream 1 {

};

transportstream 2 {

};

transportstream 3 {

};

transportstream 4 {

};

teletext {

};
```

Figure 2: General Configuration File Structure

Parameters are specified using the syntax *parameter* `=` *value*`;`. Number values may be decimal integers or hexadecimal integers with a `0x` prefix. A `k` or `M` suffix multiplies the number with 1000 or 1000000 respectively. Text string values must be enclosed in double quotes `""`.

Figure 3 shows the signal paths through the D-ATV transmitter. The baseband board interfaces to the data sources (such as MPEG2 encoders) through the transport stream interfaces. The transport stream (TS) interfaces are byte parallel interfaces consisting of eight data lines, a clock line, and several optional synchronisation lines. Table 2 shows the transport stream connector signal assignment.

TS1 and TS2 both have a FIFO in the signal path. The direction of the TS clock signal (CK) is therefore configurable, that is, it can either be sourced by the D-ATV baseband board or the data source.

TS3 and TS4, on the other hand, do not have a FIFO. The TS clock signal must therefore be driven by the D-ATV baseband board to avoid data loss. These ports are therefore mainly for use with the D-ATV MPEG2 encoder.

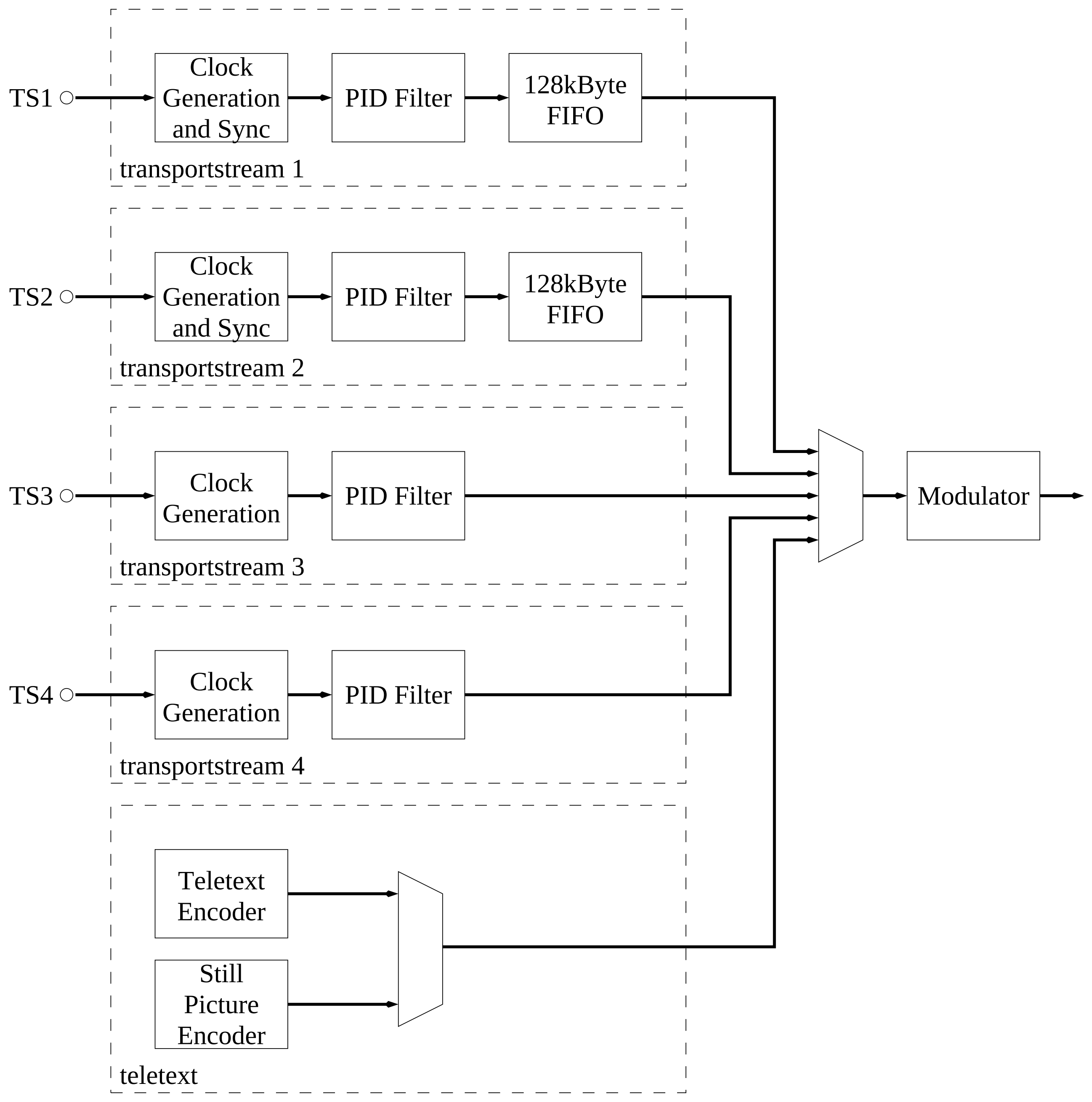

Figure 3: Blockdiagramm

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

Table 2: Transportstream-Connector

2.2 Board section

2.2.1 Clock Frequency

The `clock` parameter indicates the frequency of the crystal oscillator on the D-ATV baseband board. All timings are derived from this crystal. The D-ATV board is normally shipped with a 60 MHz crystal, but for very low bitrate applications, this crystal frequency may be reduced. The maximum is 62 MHz.

Example:

```
clock = 60000000;
```

2.3 Modulator section

2.3.1 Modulation

This parameter selects the modulation standard. Possible values are `dvb-s` and `dvb-c`.

Example:

```
modulation = dvb-s;
```

2.3.2 Constellation

This parameter selects the constellation. Possible values are `qpsk` in DVB-S mode and `qam16`, `qam32`, `qam64` in DVB-C mode.

Example:

```
constellation = qpsk;
```

2.3.3 FEC Rate, DVB-S

This parameter specifies the inner Forward Error Correction code rate. It allows a trade-off between user bit rate and robustness of the modulation signal. Possible values are `1/2`, `2/3`, `3/4`, `5/6` and `7/8`. This parameter is only meaningful in DVB-S mode.

Example:

```
fec = 5/6;
```

2.3.4 Transmitter Frequency

The frequency parameter specifies the transmission frequency. It must lie either within the 70cm, the 23cm or the 13cm amateur radio band, and the HF module must support the selected band.

Example:

```
frequency = 1275M;
```

2.3.5 Symbol Rate, DVB-S

The symbol rate parameter specifies bandwidth of the modulator signal (Eq. 1), and also the user bitrate (Eq. 2).

$$BW \approx \frac{4}{3} SR \tag{1}$$

$$BR = 2 \cdot SR \cdot R_{inner} \cdot R_{outer} \tag{2}$$

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

The ratio $\frac{2 \cdot F_{clk}}{SR}$ must be one of 4, $4\frac{1}{3}$, $4\frac{1}{2}$, $4\frac{2}{3}$, 5, $5\frac{1}{3}$, $5\frac{1}{2}$, 6, $6\frac{1}{2}$, 7, $7\frac{1}{2}$, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 20, 22, 24, 26, 28, 30, 32. `fwtool` rounds the symbol rate to the nearest ratio.

Example:

```
symbol rate = 15000k;
```

2.3.6 Symbol Rate, DVB-C

The symbol rate parameter specifies bandwidth of the modulator signal (Eq. 3), and also the user bitrate (Eq. 4).

$$BW \approx 1.15 SR \tag{3}$$

$$BR = log_2 q \cdot SR \cdot R_{outer} \tag{4}$$

SR	Symbol Rate (Symbols/s) (see section 2.3.6)
BW	Signal Bandwidth (Hz)
BR	User Bitrate (Bits/s)
q	Constellation QAMq
R_{outer}	FEC Rate, fixed at $\frac{188}{204}$
F_{clk}	Crystal Oscillator Frequency (see section 2.2.1)

The ratio $\frac{F_{clk}}{SR}$ must be one of 8, 9, 10, 11, 12, 13, 14, 15, 16. `fwtool` rounds the symbol rate to the nearest ratio. It is advisable to use a bandwidth of 8MHz or less, and thus a symbol rate of 6.9MSymbols/s or less, since many receivers are limited to 8MHz bandwidth.

Example:

```
symbol rate = 6000k;
```

2.3.7 Inversion

This parameter determines whether the I & Q signals should be swapped. Swapping the I & Q signals produces an effect similar to receiving an upper sideband (USB) signal with a lower sideband (LSB) receiver. Most, but not all[1] receivers automatically detect whether inversion is in use. Set to off if the D-ATV transmitter signal is directly transmitted. Set to on if a spectrum-inverting transverter is used.

Example:

```
inversion = off;
```

[1]The WinTV DVB-S Nova with the convergence.de firmware does not seem to automatically detect inversion

2.3.8 PTT

This parameter selects whether the transmitter is turned on or off at power-up. The PTT may subsequently be toggled via the menu. This parameter is typically set to on for D-ATV repeaters, and off for end users.

Example:

```
ptt = off;
```

2.3.9 Transmitter Callsign

The network name parameter must be set to the callsign of the transmitter.

Example:

```
network name = "HB9W";
```

2.4 Transportstream section

2.4.1 Port Mode

This parameter specifies the transport stream port mode. The following modes exist:

`off`	Port switched off
`datvencoder`	D-ATV MPEG2 encoder connected to port
`fujitsueval`	Fujitsu MPEG2 encoder board connected to port
`extclock`	Device driving the TS clock signal connected to port

The Fujitsu MPEG2 encoder evaluation board is related to the D-ATV encoder board. The difference is that if the former is connected, the baseband board does not try to download the MPEG2 encoder firmware to the encoder.

The `extclock` option can only be used on TS1 and TS2.

Example:

```
mode = datvencoder;
```

2.4.2 Active Clock Edge

This parameter only has an effect if the port is in `extclock` mode. It specifies the active edge(s) of the TS clock (CK) signal. Valid values are `falling`, `rising` and `both`.

Example:

```
clock edge = rising;
```

2.4.3 Clock Debounce Filter

This parameter specifies the action of the clock debounce filter. Set the value to the largest N that satisfies the following conditions: $1 \leq N \leq 4$ and $F_{TSCK} < \frac{F_{clk}}{2N}$.

F_{TSCK}	Frequency of the Transport Stream Clock Signal
N	Clock Debounce Filter Parameter (see section 2.4.3)
F_{clk}	Crystal Oscillator Frequency (see section 2.2.1)

Example:

```
clock filter = 4;
```

2.4.4 Bitrate

This parameter specifies the total useful bitrate sent into the port.

Example:

```
bitrate = 4500k;
```

2.4.5 Video Input Selection

This parameter only has an effect in `datvencoder` mode. It specifies the characteristics of the video input signal. It consists of one or more keywords from the list below, separated by commas.

`d1`	D1 resolution
`hd1`	HD1 resolution
`sif`	SIF resolution
`qsif`	QSIF resolution
`ntsc`	Input signal is NTSC
`pal`	Input signal is PAL
`composite`	Use Composite port
`svideo`	Use S-Video port

Example:

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

2.4.6 Video GOP Configuration

This parameter specifies the picture encoding sequence of the Encoder. The default mode provides good encoding efficiency at the expense of a higher encoding latency. Tuning this parameter, eg. by shortening the GOP size, latency may be reduced at the expense of the bitrate. This parameter should only be changed by experts who understand MPEG2 encoding.

Example:

```
video gop = "IBBPBBPBBPBBPBB";
```

2.4.7 Spatial Filter

This parameter specifies the cut-off frequency of the spatial filter and thus the sharpness of the picture. Possible values are soft, standard and sharp.

Example:

```
spatial filter = standard;
```

2.4.8 Audio Encoder Bitrate

This parameter only has an effect in `datvencoder` mode. It specifies the bitrate of the MPEG2 Layer 2 audio encoder. Valid bitrates depend on the encoding mode (subsection 2.4.9):

	Audio Encoding Mode			
Bit Rate	stereo	joint stereo	dual channel	single channel
32k	–	–	–	√
48k	–	–	–	√
56k	–	–	–	√
64k	√	√	√	√
80k	–	–	–	√
96k	√	√	√	√
112k	√	√	√	√
128k	√	√	√	√
160k	√	√	√	√
192k	√	√	√	√
224k	√	√	√	–
256k	√	√	√	–
320k	√	√	√	–
384k	√	√	√	–

Example:

```
audio bitrate = 384k;
```

2.4.9 Audio Encoder Encoding Mode

This parameter only has an effect in `datvencoder` mode. It specifies the encoding mode of the MPEG2 Layer 2 audio encoder. The value must be one of the following keywords:

`stereo`	Input is a stereo signal.
`joint stereo`	Input is a stereo signal. Encoder shall use redundancies between both channels.
`dual channel`	Both channels are unrelated.
`single channel`	Only a single channel present.

Example:

```
audio mode = joint stereo;
```

2.4.10 Audio Encoder Sampling Rate

This parameter only has an effect in `datvencoder` mode. It specifies the sampling rate of the MPEG2 Layer 2 audio encoder. The value must be either 48000, 44100, or 32000.

Example:

```
audio sample rate = 44100;
```

2.4.11 Program Clock Reference (PCR) PID

This parameter sets the PID of the Program Clock Reference (PCR). It is normally set to the PID that contains the video stream.

Example:

```
pcr pid = 0x20;
```

2.4.12 Video PID

This parameter sets the PID of the video stream.

Example:

```
video pid = 0x20;
```

2.4.13 Audio PID

This parameter sets the PID of the audio stream.

Example:

```
audio pid = 0x21;
```

2.4.14 Program Map Table (PMT) PID

This parameter sets the PID of the program map table (PMT). The PMT tells the receiver which PIDs belong to the TV channel and requires its own distinct PID.

Example:

```
pmt pid = 0x22;
```

2.4.15 Program Callsign

This parameter sets the callsign of the TV channel. The callsign is encoded into the SI tables, which allows the receiver to display the channel identification.

Example:

```
callsign = "HB9JNX";
```

2.4.16 Language

This parameter identifies the language of the TV channel. It should be set to `"eng"` for english or to `"DEU"` for german.

Example:

```
language = "eng";
```

2.4.17 PID filter

The PID filter allows the user to selectively reject PIDs on the corresponding TS input. Two strategies are possible:

1. pass all PIDs by default, list exceptions to reject
2. reject all PIDs by default, list exceptions to pass

The default is indicated by `all` or `none`, the exception terms are `minus` *pid/mask* and `plus` *pid/mask*. *pid* specifies the PID number to match, and *mask* specifies which bits shall be compared.

The resulting PID filter must reject the PID 8191 (0x1fff), the Null Packet PID.

Examples:

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

2.4.18 Tuner Mode

This parameter specifies whether a tuner is connected to the port. The following tuner modes exist:

`off`	No tuner connected to the port
`dfm`	DFM analog tuner connected to D-ATV MPEG2 encoder
`mb86a15`	Fujitsu DVB-S tuner connected directly to TS port

Example:

```
tuner mode = off;
```

2.4.19 Tuner Frequency

This parameter only has an effect if the tuner mode (section 2.4.18) is not set to `off`. It specifies the frequency to which the tuner shall be tuned to.

Example:

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

2.4.20 Tuner FEC Mode

This parameter only has an effect if the tuner mode (section 2.4.18) is set to `mb86a15`. It specifies which inner FEC settings should be tried until a valid signal is found. Its value can either be `auto` or an inner FEC rate.

Examples:

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

2.4.21 Tuner Symbol Rate

This parameter only has an effect if the tuner mode (section 2.4.18) is set to `mb86a15`. It specifies which symbol rate should be expected..

Example:

```
tuner symrate = 3000k;
```

2.5 Teletext section

2.5.1 Program Clock Reference (PCR) PID

This parameter sets the PID of the Program Clock Reference (PCR). It is normally set to the PID that contains the video stream.

Example:

```
pcr pid = 0x20;
```

2.5.2 Video PID

This parameter sets the PID of the video stream that contains the still picture.

Example:

```
video pid = 0x20;
```

2.5.3 Teletext PID

This parameter sets the PID of the teletext stream.

Example:

```
teletext pid = 0x21;
```

2.5.4 Program Map Table (PMT) PID

This parameter sets the PID of the program map table (PMT). The PMT tells the receiver which PIDs belong to the TV channel and requires its own distinct PID.

Example:

```
pmt pid = 0x22;
```

2.5.5 Program Callsign

This parameter sets the callsign of the teletext/still picture channel. The callsign is encoded into the SI tables, which allows the receiver to display the channel identification.

Example:

```
callsign = "HB9JNX";
```

2.5.6 Language

This parameter identifies the language of the teletext/still picture channel. It should be set to `"eng"` for english or to `"DEU"` for german.

Example:

```
language = "eng";
```

2.5.7 Picture File

This parameter specifies the file containing the picture that shall be transmitted on the still picture channel. This channel is intended to display eg. the logo of the operator. Note that this channel is not strictly DVB compliant, so there is no guarantee that every receiver is able to display the channel.

The file must either be a JPEG file or contain an MPEG2 elementary stream. Note that software MPEG2 encoders normally produce program stream files, which are incompatible. If the file contains a JPEG image (which must have 704×576 pixels), the `mpeg2enc` program from the mjpegtools package is used to convert the image to an MPEG2 elementary stream. The `mpeg2enc` binary must be in the same directory as `fwtool` on Windows or in the `/usr/bin` directory under Linux.

Example:

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

2.5.8 VM Code

This parameter specifies the file containing the teletext encoder virtual machine bytecode. See section 5 for additional information.

Example:

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

2.6 External Program section

The D-ATV transmitter can not only transmit locally encoded programs, but also programs from other sources such as DVB-S receivers or personal computers. In order to enable receivers to find these programs also, the D-ATV transmitter must transmit Program Map Tables (PMT) for these programs too and must properly list them in the Program Association Table (PAT). This is the purpose of the external program sections.

2.6.1 Program Clock Reference (PCR) PID

This parameter sets the PID of the Program Clock Reference (PCR). It is normally set to the PID that contains the video stream.

Example:

```
pcr pid = 0x420;
```

2.6.2 Program Map Table (PMT) PID

This parameter sets the PID of the program map table (PMT). The PMT tells the receiver which PIDs belong to the TV channel and requires its own distinct PID.

Example:

```
pmt pid = 0x422;
```

2.6.3 Language

This parameter identifies the language of the TV channel. It should be set to `"eng"` for english or to `"DEU"` for german.

Example:

```
language = "eng";
```

2.6.4 Stream subsections

The stream subsections (`video stream`, `audio stream`, `teletext stream`, `stream`) list the individual streams that compose a program.

2.6.5 PID

This parameter sets the PID of the stream.

Example:

```
pid = 0x440;
```

2.6.6 Stream Type

This parameter sets the Stream Type of the stream. See [1, Table 2-36]. This parameter can not be set manually for video, audio and teletext streams (it is set automatically).

Example:

```
stream type = 0x80;
```

2.6.7 Stream ID

This parameter sets the Stream ID of the stream. This is the number of the stream.

Example:

```
stream id = 1;
```

2.6.8 Component Type

This parameter sets the Component Type of the stream. See [2, Table 24].

Example:

```
component type = 1;
```

2.6.9 Language

This parameter identifies the language of the stream. It should be set to "eng" for english or to "DEU" for german.

Example:

```
language = "eng";
```

3 The Old Teletext Encoder

The old teletext encoder is driven by a static table of teletext pages and their lines. It allows little dynamic content and no control over the encoding process itself. It will likely be removed in the future. Figure 4 shows the configuration file instructions to encode an example teletext page.

```
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.";
        };
};
```

Figure 4: Old Teletext Encoder Configuration Lines

3.1 Teletext section

3.1.1 Teletext Page Header

This parameter sets the contents of the topmost teletext line that is displayed right of the page number.

Example:

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

3.2 Teletext Page section

The teletext section may contain `page { };` subsections, each specifying a single teletext page.

3.2.1 Page Number

This parameter specifies the teletext page number. Its value must be between 100 and 899 inclusive. Teletext decoders start with page 100, so a page with number 100 should be present and display introductory material.

Example:

```
page number = 100;
```

3.2.2 Teletext Lines

These parameters specify the teletext page lines. The lines are numbered from 1 to 24 inclusive. Teletext lines can be up to 40 characters long. Shorter lines are padded with space characters. Nonprinting characters can be entered by a slash, followed by an “x”, and a two digit hexadecimal number that encodes the character code. For example, to enter character 1 (0x01), type “`\x01`”. Character codes 0–31 (0x00–0x1f) are used for ETSI Teletext Attribute markup (eg. colours), and character codes 128–255 (0x80–0xff) are used to insert dynamic data, such as packet counters. TODO: describe dynamic data insertion in more detail.

Example:

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

4 Sample Configuration

Figure 5 shows a simple minimalistic configuration file. It assumes that there is one MPEG2 encoder connected to TS1, and that TS2–TS4 are left unconnected.

```
# 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";
};
```

Figure 5: Simple Configuration File

5 The New Teletext Encoder

The new teletext encoder allows full control over the encoding process and arbitrary dynamic content. It is driven by a user bytecode program that is interpreted by a stack-based virtual machine.

Bytecode teletext programs need not be written in the stack-based assembly language of the virtual machine (VM). They can be written in the C programming language and then compiled to bytecode.

The following table shows the executables that constitute the bytecode development system:

cpp	C Preprocessor
rcc	C Compiler proper
vm	VM Simulator
vmar	Bytecode Archiver
vmas	Bytecode Assembler
vmdisass	Bytecode Disassembler
vmld	Bytecode Linker
atv2txtvm	Conversion utility from DG9MHZ ATV files to VM Teletext source code

Assuming the teletext encoder C code is contained in a file named `teletext.c`, the C code can be compiled and assembled into the object code file `teletext.o` using the following command:

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

The object code file can be disassembled with:

```
vmdisass teletext.o
```

The object code file can be simulated with:

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

Figure 6 shows an example source code of a teletext encoder.

DG9MHZ ATV files [?] can be converted to VM teletext object code using:

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

ATV files can be written by `vtedit` from [?].

5.1 C-Code

The header file `dvbs.h` contains prototypes for the built-in library functions.

The VM starts the teletext encoder by calling the function `teletext`, with the prototype `void teletext(void)`.

5.2 VM Built-In Library Functions

5.2.1 C type sizes

Type	Bits
`char`	8
`short`	16
`int`	32
`long`	32

5.2.2 C99 standard macros

`NULL`, `offsetof`

5.2.3 C99 standard types

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

```
/* 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);
        }
}
```

Figure 6: Example Teletext Encoder Source Code

5.2.4 C99 standard functions

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

5.2.5 Event log functions

`void logreadinit(unsigned int *p);` rewinds the event log to the oldest log message still in the circular buffer. `p` is a pointer to an opaque cookie of type `unsigned int`.

`unsigned int logreadline(unsigned int *p, char *buf, unsigned int bufsz);` reads the next event log message. `p` is a pointer to the opaque cookie initialized by `logreadinit`, `buf` a pointer to a buffer of suitable size, and `bufsz` the size of the buffer. `logreadline` returns the number of characters stored in the buffer, which may not be zero terminated. A zero return value means that there are no more events in the event log buffer.

5.2.6 Time and Date functions

struct time

day	modified julian date (number of days since november 17th 1858)
msec	milliseconds
sec	seconds since midnight (UTC)
valid	if set, time and date have been set via the serial interface or DCF77

struct timehms

h	hours
m	minutes
s	seconds

struct date

d	day
m	month
s	year

`struct time gettime(void);` returns the current time.

`u_int32_t getjiffies(void);` returns a monotonically increasing number. It increments `HZ` times per second.

`struct date mjdtodate(u_int16_t mjd);` converts a modified julian date to a standard gregorian date.

`u_int16_t datetomjd(u_int16_t d, u_int16_t m, u_int16_t y);` converts a standard gregorian date to a modified julian date.

`char *timedec(char *buf, struct timehms *hms, u_int32_t tm);` takes the number of seconds since midnight and converts it to hours, minutes and seconds and to a human readable string of the form 01:23:45. Both `hms` and `buf` may be `NULL`. The function returns a pointer to `buf[0]`.

5.2.7 Parameters and Statistics functions

`u_int16_t getadc(unsigned int n);` returns the value of A/D converter `n`. `n` ranges from 0 to 3, and the return value of the 10 bit A/D converter is between 0 and 1023, corresponding to 0 and 5V at the input.

`u_int32_t readcounter(unsigned int n);` returns the value of counter `n`.

n	Value
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);` returns the spectral inversion setting.

`u_int8_t get_fecmode(void);` returns the FEC mode.

return value	FEC mode
0	1/2
1	2/3
2	3/4
3	5/6
4	7/8

`u_int32_t get_frequency(void);` returns the transmitter center frequency 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);` converts `val` to a decimal string, which is stored into the buffer `buf`. Up to `len` characters are stored, and thus `buf` must be at least `len` + 1 characters big. The function returns a pointer to the beginning of the number string, which is anywhere within `buf`, but not necessarily at the beginning.

`char *int2dec(char *buf, u_int16_t len, u_int32_t val, u_int16_t flags);` converts `val` to a hexadecimal string, which is stored into the buffer `buf`. Up to `len` characters are stored, and thus `buf` must be at least `len` + 1 characters big. The function returns a pointer to the beginning of the number string, which is anywhere within `buf`, but not necessarily at the beginning.

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	incremented, whenever service descriptor table information is updated
transport_stream_id	Transport Stream ID
nit_pid	PID where the Network Information Table is transmitted
service_id	Service ID
network_id	Network ID
service_provider_name	Service Provider Name
service_name	Service Name

For more detailed information about DVB System Information (SI) tables, see [2].

`struct portcapture getcapture(unsigned int port);` returns SI table data for transport stream port `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, ...);` encodes multiple teletext lines, from line number `startline` to `endline`. `startline` is usually 0, and `endline` 24. `pagenr` specifies the page number, and should be between `0x100` and `0x8ff`. Page numbers containing the hexadecimal characters A–F are not directly accessible by the receiver. `subnr` specifies the subpage number (normally 0). `flags` can be zero or multiple `TXTPAGECTRL` macros ored. `TXTPAGECTRL` are described in detail in [3, 9.3.1.3, p. 27]. `lines` contains a pointer to an array of `endline-startline+1` strings. Each string specifies the contents of the corresponding line. A `NULL` pointer suppresses the encoding of the corresponding line. Teletext lines may contain `TXTATTR` macros (see [3, 12.2, p. 76–80]) or `TXT_ARG`*n* to reference one of the optional arguments. `...` can contain up to 64 pointers to strings that can be referenced using the `TXT_ARG`*n* macro.

5.2.11 Lowlevel Teletext Enconding functions

`void teletext_oddparity(u_int8_t *buf, const u_int8_t *src, unsigned int len);` encodes a data buffer pointed to by `src` of size `len` using the teletext odd parity code and stores it into `buf`.

`void teletext_hamming84(u_int8_t *buf, const u_int8_t *src, unsigned int nibblelen);` encodes a data buffer pointed to by `src` containing `len` nibbles using the teletext 8/4 hamming code and stores it into `buf`. It first encodes the low nibble of `src[0]`, then the high nibble of `src[0]`, then the low nibble of `src[1]`, and so on.

`void teletext_hamming2418(u_int8_t *buf, const u_int8_t *src, unsigned int len);` encodes a data buffer pointed to by `src` containing `len` triples using the teletext 24/18 hamming code and stores it into `buf`. The `src[0]` contains the low 6 bits, `src[1]` the middle 6 bits, and `src[2]` the high 6 bits.

`u_int8_t *teletext_currentline(void);` returns a pointer to the current line buffer. The line buffer size is 42 bytes, and it contains a complete teletext line without the clock run-in and the framing code [3, 7.1, p. 17ff].

`u_int8_t *teletext_waitline(void);` transmits the current line and returns a pointer to the line buffer of the next line.

6 Connecting the PC Parallel Port to a Transport Stream input

The PC Parallel port may be used as a simple means to inject low rate data into the transport stream. Up to about 2MBit/s are possible. Table 3 shows how the parallel port signals should be wired to TS1 or TS2. The input port must be set to `extclock` mode, and the
tt clock filter should be set to 4, the maximum.

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

Table 3: Parallel Port Connector Wiring

7 Acknowledgements

The Microcontroller Firmware contains uIP, Copyright (c) 2001, Adam Dunkels.

The VM bytecode C compiler is based on LCC, written by Chris Fraser and David Hanson. LCC sources can be obtained free of charge from the LCC homepage [?].

References

[1] ISO/IEC 13818-1 Generic Coding of Moving Pictures and Associated Audio: Systems Recommendation H.222.0, 04 1995.

[2] 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.

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