

libdvbv5

Generated by Doxygen 1.9.3

1 The libdvbv5 API documentation	1
1.1 Introduction	2
1.2 Features provided by libdvbv5	2
1.3 Introduction to DVBv5 key value properties	3
1.3.1 DVBv5 and libdvbv5 properties	3
1.3.2 DVBv5 and libdvbv5 statistics	3
2 DVBv5 Tools	4
3 Todo List	4
4 Module Index	5
4.1 Modules	5
5 Data Structure Index	5
5.1 Data Structures	5
6 File Index	9
6.1 File List	9
7 Module Documentation	11
7.1 Digital TV device enumeration	11
7.1.1 Detailed Description	12
7.1.2 Enumeration Type Documentation	12
7.1.3 Function Documentation	13
7.2 Digital TV frontend control	20
7.2.1 Detailed Description	23
7.2.2 Macro Definition Documentation	23
7.2.3 Enumeration Type Documentation	28
7.2.4 Function Documentation	29
7.3 Digital TV frontend scan	42
7.3.1 Detailed Description	43
7.3.2 Typedef Documentation	43
7.3.3 Function Documentation	44
7.4 Satellite Equipment Control	49
7.4.1 Detailed Description	50
7.4.2 Enumeration Type Documentation	50
7.4.3 Function Documentation	50
7.5 Ancillary functions and macros	53
7.5.1 Detailed Description	55
7.5.2 Macro Definition Documentation	55
7.5.3 Typedef Documentation	55
7.5.4 Enumeration Type Documentation	55
7.5.5 Function Documentation	61

7.6 Digital TV table parsing	63
7.6.1 Detailed Description	72
7.6.2 Macro Definition Documentation	72
7.6.3 Typedef Documentation	82
7.6.4 Enumeration Type Documentation	85
7.6.5 Function Documentation	91
7.6.6 Variable Documentation	105
7.7 Parsers for several MPEG-TS descriptors	105
7.7.1 Detailed Description	110
7.7.2 Enumeration Type Documentation	110
7.7.3 Function Documentation	110
7.8 Digital TV demux	130
7.8.1 Detailed Description	130
7.8.2 Function Documentation	130
7.9 Channel and transponder file read/write	133
7.9.1 Detailed Description	134
7.9.2 Enumeration Type Documentation	134
7.9.3 Function Documentation	135
7.9.4 Variable Documentation	141
8 Data Structure Documentation	141
8.1 atsc_desc_service_location Struct Reference	141
8.1.1 Detailed Description	142
8.1.2 Field Documentation	142
8.2 atsc_desc_service_location_elementary Struct Reference	143
8.2.1 Detailed Description	143
8.2.2 Field Documentation	144
8.3 atsc_table_eit Struct Reference	145
8.3.1 Detailed Description	145
8.3.2 Field Documentation	145
8.4 atsc_table_eit_desc_length Union Reference	146
8.4.1 Detailed Description	146
8.4.2 Field Documentation	146
8.5 atsc_table_eit_event Struct Reference	147
8.5.1 Detailed Description	147
8.5.2 Field Documentation	148
8.6 atsc_table_mgt Struct Reference	150
8.6.1 Detailed Description	150
8.6.2 Field Documentation	151
8.7 atsc_table_mgt_table Struct Reference	152
8.7.1 Detailed Description	152
8.7.2 Field Documentation	153

8.8 atsc_table_vct Struct Reference	154
8.8.1 Detailed Description	154
8.8.2 Field Documentation	155
8.9 atsc_table_vct_channel Struct Reference	156
8.9.1 Detailed Description	156
8.9.2 Field Documentation	157
8.10 atsc_table_vct_descriptor_length Union Reference	161
8.10.1 Detailed Description	161
8.10.2 Field Documentation	161
8.11 dvb_desc Struct Reference	162
8.11.1 Detailed Description	162
8.11.2 Field Documentation	162
8.12 dvb_desc_ca Struct Reference	163
8.12.1 Detailed Description	163
8.12.2 Field Documentation	164
8.13 dvb_desc_ca_identifier Struct Reference	165
8.13.1 Detailed Description	165
8.13.2 Field Documentation	166
8.14 dvb_desc_cable_delivery Struct Reference	167
8.14.1 Detailed Description	167
8.14.2 Field Documentation	167
8.15 dvb_desc_event_extended Struct Reference	169
8.15.1 Detailed Description	169
8.15.2 Field Documentation	170
8.16 dvb_desc_event_extended_item Struct Reference	171
8.16.1 Detailed Description	172
8.16.2 Field Documentation	172
8.17 dvb_desc_event_short Struct Reference	172
8.17.1 Detailed Description	173
8.17.2 Field Documentation	174
8.18 dvb_desc_frequency_list Struct Reference	175
8.18.1 Detailed Description	175
8.18.2 Field Documentation	176
8.19 dvb_desc_hierarchy Struct Reference	177
8.19.1 Detailed Description	177
8.19.2 Field Documentation	178
8.20 dvb_desc_language Struct Reference	179
8.20.1 Detailed Description	179
8.20.2 Field Documentation	180
8.21 dvb_desc_logical_channel Struct Reference	180
8.21.1 Detailed Description	180
8.21.2 Field Documentation	181

8.22 dvb_desc_logical_channel_number Struct Reference	181
8.22.1 Detailed Description	182
8.22.2 Field Documentation	182
8.23 dvb_desc_network_name Struct Reference	183
8.23.1 Detailed Description	183
8.23.2 Field Documentation	184
8.24 dvb_desc_registration Struct Reference	184
8.24.1 Detailed Description	184
8.24.2 Field Documentation	185
8.25 dvb_desc_sat Struct Reference	186
8.25.1 Detailed Description	186
8.25.2 Field Documentation	187
8.26 dvb_desc_service Struct Reference	188
8.26.1 Detailed Description	188
8.26.2 Field Documentation	189
8.27 dvb_desc_t2_delivery Struct Reference	190
8.27.1 Detailed Description	190
8.27.2 Field Documentation	191
8.28 dvb_desc_t2_delivery_cell Struct Reference	193
8.28.1 Detailed Description	193
8.28.2 Field Documentation	194
8.29 dvb_desc_t2_delivery_subcell Struct Reference	195
8.29.1 Detailed Description	195
8.29.2 Field Documentation	196
8.30 dvb_desc_t2_delivery_subcell_old Struct Reference	196
8.30.1 Detailed Description	196
8.30.2 Field Documentation	197
8.31 dvb_desc_terrestrial_delivery Struct Reference	197
8.31.1 Detailed Description	197
8.31.2 Field Documentation	198
8.32 dvb_desc_ts_info Struct Reference	200
8.32.1 Detailed Description	200
8.32.2 Field Documentation	201
8.33 dvb_desc_ts_info_transmission_type Struct Reference	202
8.33.1 Detailed Description	202
8.33.2 Field Documentation	203
8.34 dvb_descriptor Struct Reference	203
8.34.1 Detailed Description	203
8.34.2 Field Documentation	204
8.35 dvb_dev_list Struct Reference	205
8.35.1 Detailed Description	205
8.35.2 Field Documentation	205

8.36 dvb_device Struct Reference	207
8.36.1 Detailed Description	207
8.36.2 Field Documentation	207
8.37 dvb_elementary_pid Struct Reference	208
8.37.1 Detailed Description	208
8.37.2 Field Documentation	208
8.38 dvb_entry Struct Reference	209
8.38.1 Detailed Description	209
8.38.2 Field Documentation	210
8.39 dvb_ext_descriptor Struct Reference	214
8.39.1 Detailed Description	214
8.39.2 Field Documentation	215
8.40 dvb_extension_descriptor Struct Reference	216
8.40.1 Detailed Description	216
8.40.2 Field Documentation	216
8.41 dvb_file Struct Reference	217
8.41.1 Detailed Description	217
8.41.2 Field Documentation	217
8.42 dvb_mpeg_es_pic_start Struct Reference	218
8.42.1 Detailed Description	218
8.42.2 Field Documentation	219
8.43 dvb_mpeg_es_seq_start Struct Reference	220
8.43.1 Detailed Description	221
8.43.2 Field Documentation	221
8.44 dvb_mpeg_pes Struct Reference	223
8.44.1 Detailed Description	224
8.44.2 Field Documentation	224
8.45 dvb_mpeg_pes_optional Struct Reference	225
8.45.1 Detailed Description	225
8.45.2 Field Documentation	226
8.46 dvb_mpeg_ts Struct Reference	228
8.46.1 Detailed Description	229
8.46.2 Field Documentation	229
8.47 dvb_mpeg_ts_adaption Struct Reference	231
8.47.1 Detailed Description	231
8.47.2 Field Documentation	232
8.48 dvb_open_descriptor Struct Reference	233
8.48.1 Detailed Description	234
8.49 dvb_parse_file Struct Reference	234
8.49.1 Detailed Description	234
8.49.2 Field Documentation	234
8.50 dvb_parse_struct Struct Reference	235

8.50.1 Detailed Description	235
8.50.2 Field Documentation	235
8.51 dvb_parse_table Struct Reference	236
8.51.1 Detailed Description	236
8.51.2 Field Documentation	237
8.52 dvb_sat_inb Struct Reference	237
8.52.1 Detailed Description	238
8.52.2 Field Documentation	238
8.53 dvb_table_cat Struct Reference	239
8.53.1 Detailed Description	239
8.53.2 Field Documentation	240
8.54 dvb_table_eit Struct Reference	240
8.54.1 Detailed Description	240
8.54.2 Field Documentation	241
8.55 dvb_table_eit_event Struct Reference	242
8.55.1 Detailed Description	242
8.55.2 Field Documentation	243
8.56 dvb_table_filter Struct Reference	245
8.56.1 Detailed Description	245
8.56.2 Field Documentation	245
8.57 dvb_table_header Struct Reference	246
8.57.1 Detailed Description	246
8.57.2 Field Documentation	247
8.58 dvb_table_nit Struct Reference	249
8.58.1 Detailed Description	249
8.58.2 Field Documentation	249
8.59 dvb_table_nit_transport Struct Reference	250
8.59.1 Detailed Description	250
8.59.2 Field Documentation	251
8.60 dvb_table_nit_transport_header Union Reference	252
8.60.1 Detailed Description	252
8.60.2 Field Documentation	253
8.61 dvb_table_pat Struct Reference	253
8.61.1 Detailed Description	253
8.61.2 Field Documentation	254
8.62 dvb_table_pat_program Struct Reference	254
8.62.1 Detailed Description	255
8.62.2 Field Documentation	255
8.63 dvb_table_pmt Struct Reference	256
8.63.1 Detailed Description	256
8.63.2 Field Documentation	257
8.64 dvb_table_pmt_stream Struct Reference	258

8.64.1 Detailed Description	259
8.64.2 Field Documentation	259
8.65 dvb_table_sdt Struct Reference	261
8.65.1 Detailed Description	261
8.65.2 Field Documentation	262
8.66 dvb_table_sdt_service Struct Reference	262
8.66.1 Detailed Description	263
8.66.2 Field Documentation	263
8.67 dvb_ts_packet_header Struct Reference	265
8.67.1 Detailed Description	265
8.67.2 Field Documentation	266
8.68 dvb_v5_descriptors Struct Reference	269
8.68.1 Detailed Description	269
8.68.2 Field Documentation	270
8.69 dvb_v5_descriptors_program Struct Reference	272
8.69.1 Detailed Description	272
8.69.2 Field Documentation	272
8.70 dvb_v5_fe_parms Struct Reference	273
8.70.1 Detailed Description	274
8.70.2 Field Documentation	274
8.71 dvb_sat_inb::dvbsat_freorange Struct Reference	277
8.71.1 Detailed Description	277
8.71.2 Field Documentation	277
8.72 isdb_desc_partial_reception Struct Reference	278
8.72.1 Detailed Description	278
8.72.2 Field Documentation	278
8.73 isdb_partial_reception_service_id Struct Reference	279
8.73.1 Detailed Description	279
8.73.2 Field Documentation	279
8.74 isdbt_desc_terrestrial_delivery_system Struct Reference	280
8.74.1 Detailed Description	280
8.74.2 Field Documentation	281
8.75 ts_t Struct Reference	282
8.75.1 Detailed Description	282
8.75.2 Field Documentation	283
9 File Documentation	284
9.1 doc/libdvbv5-index.doc File Reference	284
9.2 lib/include/libdvbv5/atsc_eit.h File Reference	284
9.2.1 Detailed Description	285
9.3 atsc_eit.h	286
9.4 lib/include/libdvbv5/atsc_header.h File Reference	287

9.4.1 Detailed Description	287
9.5 atsc_header.h	288
9.6 lib/include/libdvbv5/cat.h File Reference	288
9.6.1 Detailed Description	289
9.6.2 Function Documentation	289
9.7 cat.h	290
9.8 lib/include/libdvbv5/countries.h File Reference	291
9.8.1 Detailed Description	293
9.9 countries.h	293
9.10 lib/include/libdvbv5/crc32.h File Reference	296
9.10.1 Detailed Description	297
9.11 crc32.h	297
9.12 lib/include/libdvbv5/desc_atsc_service_location.h File Reference	298
9.12.1 Detailed Description	298
9.13 desc_atsc_service_location.h	299
9.14 lib/include/libdvbv5/desc_ca.h File Reference	299
9.14.1 Detailed Description	300
9.14.2 Macro Definition Documentation	301
9.15 desc_ca.h	301
9.16 lib/include/libdvbv5/desc_ca_identifier.h File Reference	302
9.16.1 Detailed Description	303
9.16.2 Macro Definition Documentation	303
9.17 desc_ca_identifier.h	304
9.18 lib/include/libdvbv5/desc_cable_delivery.h File Reference	304
9.18.1 Detailed Description	305
9.18.2 Variable Documentation	305
9.19 desc_cable_delivery.h	306
9.20 lib/include/libdvbv5/desc_event_extended.h File Reference	307
9.20.1 Detailed Description	307
9.21 desc_event_extended.h	308
9.22 lib/include/libdvbv5/desc_event_short.h File Reference	308
9.22.1 Detailed Description	309
9.23 desc_event_short.h	310
9.24 lib/include/libdvbv5/desc_extension.h File Reference	310
9.24.1 Detailed Description	311
9.25 desc_extension.h	312
9.26 lib/include/libdvbv5/desc_frequency_list.h File Reference	313
9.26.1 Detailed Description	313
9.27 desc_frequency_list.h	314
9.28 lib/include/libdvbv5/desc_hierarchy.h File Reference	314
9.28.1 Detailed Description	315
9.29 desc_hierarchy.h	315

9.30 lib/include/libdvbv5/desc_isdbt_delivery.h File Reference	316
9.30.1 Detailed Description	317
9.30.2 Variable Documentation	317
9.31 desc_isdbt_delivery.h	318
9.32 lib/include/libdvbv5/desc_language.h File Reference	318
9.32.1 Detailed Description	319
9.33 desc_language.h	320
9.34 lib/include/libdvbv5/desc_logical_channel.h File Reference	320
9.34.1 Detailed Description	321
9.35 desc_logical_channel.h	322
9.36 lib/include/libdvbv5/desc_network_name.h File Reference	322
9.36.1 Detailed Description	323
9.37 desc_network_name.h	324
9.38 lib/include/libdvbv5/desc_partial_reception.h File Reference	324
9.38.1 Detailed Description	325
9.39 desc_partial_reception.h	326
9.40 lib/include/libdvbv5/desc_registration_id.h File Reference	326
9.40.1 Detailed Description	327
9.41 desc_registration_id.h	327
9.42 lib/include/libdvbv5/desc_sat.h File Reference	328
9.42.1 Detailed Description	329
9.42.2 Variable Documentation	329
9.43 desc_sat.h	330
9.44 lib/include/libdvbv5/desc_service.h File Reference	331
9.44.1 Detailed Description	331
9.45 desc_service.h	332
9.46 lib/include/libdvbv5/desc_t2_delivery.h File Reference	332
9.46.1 Detailed Description	333
9.46.2 Variable Documentation	334
9.47 desc_t2_delivery.h	334
9.48 lib/include/libdvbv5/desc_terrestrial_delivery.h File Reference	335
9.48.1 Detailed Description	336
9.48.2 Function Documentation	337
9.48.3 Variable Documentation	337
9.49 desc_terrestrial_delivery.h	338
9.50 lib/include/libdvbv5/desc_ts_info.h File Reference	339
9.50.1 Detailed Description	339
9.51 desc_ts_info.h	340
9.52 lib/include/libdvbv5/descriptors.h File Reference	341
9.52.1 Detailed Description	343
9.52.2 Macro Definition Documentation	344
9.53 descriptors.h	344

9.54 lib/include/libdvbv5/dvb-demux.h File Reference	348
9.54.1 Detailed Description	348
9.55 dvb-demux.h	349
9.56 lib/include/libdvbv5/dvb-dev.h File Reference	349
9.56.1 Detailed Description	351
9.56.2 Typedef Documentation	351
9.56.3 Enumeration Type Documentation	351
9.56.4 Function Documentation	352
9.57 dvb-dev.h	353
9.58 lib/include/libdvbv5/dvb-fe.h File Reference	355
9.58.1 Detailed Description	357
9.59 dvb-fe.h	358
9.60 lib/include/libdvbv5/dvb-file.h File Reference	360
9.60.1 Detailed Description	362
9.61 dvb-file.h	362
9.62 lib/include/libdvbv5/dvb-log.h File Reference	365
9.62.1 Detailed Description	365
9.62.2 Typedef Documentation	365
9.63 dvb-log.h	366
9.64 lib/include/libdvbv5/dvb-sat.h File Reference	366
9.64.1 Detailed Description	367
9.65 dvb-sat.h	368
9.66 lib/include/libdvbv5/dvb-scan.h File Reference	368
9.66.1 Detailed Description	370
9.66.2 Macro Definition Documentation	370
9.67 dvb-scan.h	370
9.68 lib/include/libdvbv5/dvb-v5-std.h File Reference	372
9.68.1 Detailed Description	374
9.69 dvb-v5-std.h	374
9.70 lib/include/libdvbv5/eit.h File Reference	376
9.70.1 Detailed Description	377
9.70.2 Variable Documentation	377
9.71 eit.h	378
9.72 lib/include/libdvbv5/header.h File Reference	379
9.72.1 Detailed Description	379
9.73 header.h	380
9.74 lib/include/libdvbv5/mgt.h File Reference	381
9.74.1 Detailed Description	382
9.74.2 Macro Definition Documentation	382
9.75 mgt.h	383
9.76 lib/include/libdvbv5/mpeg_es.h File Reference	384
9.76.1 Detailed Description	385

9.77 mpeg_es.h	385
9.78 lib/include/libdvbv5/mpeg_pes.h File Reference	387
9.78.1 Detailed Description	388
9.79 mpeg_pes.h	388
9.80 lib/include/libdvbv5/mpeg_ts.h File Reference	390
9.80.1 Detailed Description	391
9.81 mpeg_ts.h	391
9.82 lib/include/libdvbv5/nit.h File Reference	392
9.82.1 Detailed Description	393
9.83 nit.h	394
9.84 lib/include/libdvbv5/pat.h File Reference	395
9.84.1 Detailed Description	396
9.85 pat.h	397
9.86 lib/include/libdvbv5/pmt.h File Reference	397
9.86.1 Detailed Description	399
9.86.2 Macro Definition Documentation	399
9.87 pmt.h	400
9.88 lib/include/libdvbv5/sdt.h File Reference	401
9.88.1 Detailed Description	402
9.89 sdt.h	403
9.90 lib/include/libdvbv5/vct.h File Reference	404
9.90.1 Detailed Description	404
9.91 vct.h	405
10 Example Documentation	407
10.1 dvbv5-scan.c	407
10.2 dvbv5-zap.c	412
10.3 dvb-fe-tool.c	428
10.4 dvb-format-convert.c	432
Index	435

1 The libdvbv5 API documentation

Copyright

GNU General Public License version 2 (GPLv2)

Author

Mauro Carvalho Chehab mchehab@kernel.org

Source code

The source code for libdvbv5 is available together with v4l-utils source code, at: <http://git.linuxtv.org/cgit.cgi/v4l-utils.git/>

See also

For DVBv5 Kernel API specification, see <http://linuxtv.org/downloads/v4l-dvb-apis/dvbapi.html>.

Bug Report

Please submit bug report and patches to linux-media@vger.kernel.org

1.1 Introduction

This is a library that provides access to DVB adapter cards using the Linux DVB API version 5, as defined at <http://linuxtv.org/downloads/v4l-dvb-apis/dvbapi.html>.

It also provides backward compatibility to a driver that supports only the legacy DVBv3 API.

Note

The DVBv3 API was deprecated at the Linux Kernel. Any Kernel since version 3.3 supports the DVBv5 API.

The DVBv3 API was replaced because its support is limited to just 4 standards, without covering their innovations: ATSC, DVB-C, DVB-T and DVB-S.

The DVBv5 API was originally introduced to support DVB-S2 (also called as DVB S2API, at the time it was merged), and were designed in a way that it can easily support any newer standards. So, extensions were added to support other standards, like ATSC-MH, DVB-T2, DVB-S2, ISDB, CMDB, etc.

Most of those standards are supported by libdvbv5.

As the libdvbv5 API is maintained by the same people that maintains the Linux DVB drivers, and it is used as the reference library, together with the dvbv5 applications, all new improvements at the Linux DVB API gets merged, the corresponding support at libdvbv5 is also included.

1.2 Features provided by libdvbv5

The libdvbv5 provides the following features:

- It uses the latest DVBv5 API speck to talk with the Digital TV devices on Linux, falling back to older versions of it, up to the latest version of the DVBv3 API;
- It supports several Satellite Equipment Control (SEC) types and systems;
- It supports SCR/Unicable setups for Satellite;
- It supports several DiSEqC satellite system configurations;
- It provides a standard way to scan for DVB channels for several different types of standards: DVB-S, DVB-S2, DVB-S Turbo, DVB-C DVB-T, DVB-T2, ATSC and ISDB-T;
- It is flexible enough to be extended to support newer standards and newer features;
- It provides a way to activate/deactivate the Low Noise Amplifier (LNA) on devices that support such feature;
- It parses the MPEG-TS main tables found on Digital TV systems, like PAT, PMT, SDT, TVCT, CVCT, NIT, EIT, MGT and CAT;
- Provides enhanced statistics indication about the Quality of Service provided by a tuned transponder.

1.3 Introduction to DVBv5 key value properties

1.3.1 DVBv5 and libdvbv5 properties

The deprecated DVBv3 frontend API used to declare an union that contains 4 structs inside, one for each of the supported standards (ATSC and DVB-T/C/S).

This gives no flexibility to extend, as adding more structs at the union would change the size of the struct, breaking the Kernel-space to userspace API, and causing all DVB applications to break.

So, instead of keeping using this approach, the DVBv5 API came with a different way: it passes a series of values using key/value properties.

Those values feed a in-Kernel cache. There are two special properties:

- DTV_CLEAR - clears the Kernel cache
- DTV_TUNE - sends the cache for the DTV driver to tune into a transponder.

See http://linuxtv.org/downloads/v4l-dvb-apis/FE_GET_SET_PROPERTY.html for more details.

The same way as DVBv5, the libdvbv5 API also works with a set of key/value properties.

Inside libdvbv5, there are two types of properties:

- The ones defined at the Kernel's frontend API, that are found at `/usr/include/linux/dvb/frontend.h` (actually, it uses a local copy of that file, stored at `./include/linux/dvb/frontend.h`);
- Some extra properties used by libdvbv5. Those can be found at [lib/include/libdvbv5/dvb-v5-std.h](#) and start at `DTV_USER_COMMAND_START`.

Those extra properties allow to control other parameters that are visible only on userspace, like the Service ID that will be used, and the corresponding audio and video program IDs.

1.3.2 DVBv5 and libdvbv5 statistics

Just like what happens with DVBv3 frontend setting, the statistics provided by DVBv3 has several issues.

On DVBv3, there are a number of special ioctls designed to get the statistics from a DTV device.

Those DVBv3 statistics are not flexible, and they lack the scales that are provided by each call. So, for example, a `FE_READ_SNR` ioctl returns a number from 0 to 65535, but there's no way to know if this number is a value in dB (or a submultiple) or if it is just a relative quality number related to the Signal/Noise ratio.

Also, some delivery systems like ISDB provide up to 4 statistics for each parameter, because it allows to set different modulation parameters to the several different layers of the stream.

Starting with DVBv5 version 5.10 (added on Kernel 3.8), there's now a new mechanism to retrieve the statistics. This mechanism provides a way to discover the scale used internally by the Kernel, allowing the userspace applications to properly present the statistics.

It also allows to obtain per-layer statistics, plus a global ponderated mean statistics for the transponder as a whole, on standards like ISDB.

Just like the DTV properties, the stats are cached. That warrants that all stats are got at the same time, when `dvb_fe_get_stats()` is called. The Kernel drivers internally also warrant that those stats are also obtained for the same period of time, making them more coherent.

The libdvbv5 automatically detects if the Digital TV driver in usage provides the DVBv5 version 5.10 statistics mechanism. If it doesn't, it falls back to DVBv3 way.

If DVB version 5.10 is supported, it also provides an extra Quality of service indicator that tells if a received transponder has Poor, OK or Good quality.

2 DVBv5 Tools

Set of DVBv5 tools, bundled together with libdvbv5.

DVBv5 Tools is a small set of command line utilities that was developed to be compliant with the newer features provided by version 5 of the DVB API. The tools should also be backward compatible with the older v3 DVB API. They were written using libdvbv5.

It is composed of 4 tools:

- dvbv5-scan - Scans the channel transponders and gets the services available there;
- dvbv5-zap - Locks into a channel (zap), allowing other applications to get the stream at the dvr devices or to monitor the stream;
- dvb-fe-tool - Lists frontend properties and allow to manually set the DVB frontend
- dvb-format-convert - Converts from/to other formats used by DVBv3 apps into the dvbv5 format.

The DVBv5 default file format is formed by a channel name, followed by a series of key/value properties. Those tools also support the legacy formats used by dvb-apps.

For example, this is a channel file with one DVB-C channel on it:

```
[CHANNEL]
DELIVERY_SYSTEM = DVBC/ANNEX_A
FREQUENCY = 573000000
SYMBOL_RATE = 5217000
INNER_FEC = NONE
MODULATION = QAM/256
INVERSION = AUTO
```

And this is (part of) a service (zap) file, produced from the above channel definition using dvbv5-scan:

```
[SBT]
SERVICE_ID = 4
VIDEO_PID = 42
AUDIO_PID = 257
PID_f1 = 768
FREQUENCY = 573000000
MODULATION = QAM/256
INVERSION = AUTO
SYMBOL_RATE = 5217000
INNER_FEC = NONE
DELIVERY_SYSTEM = DVBC/ANNEX_A

[TNT]
SERVICE_ID = 48
VIDEO_PID = 336
AUDIO_PID = 337 338 849
PID_86 = 816
FREQUENCY = 573000000
MODULATION = QAM/256
INVERSION = AUTO
SYMBOL_RATE = 5217000
INNER_FEC = NONE
DELIVERY_SYSTEM = DVBC/ANNEX_A
```

3 Todo List

Global `dvb_fe_open_flags` (int adapter, int frontend, unsigned verbose, unsigned use_legacy_call, dvb_↔
logfunc logfunc, int flags)

Add/check support for O_NONBLOCK at the scan routines.

4 Module Index

4.1 Modules

Here is a list of all modules:

Digital TV device enumeration	11
Digital TV frontend control	20
Digital TV frontend scan	42
Satellite Equipment Control	49
Ancillary functions and macros	53
Digital TV table parsing	63
Parsers for several MPEG-TS descriptors	105
Digital TV demux	130
Channel and transponder file read/write	133

5 Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

atsc_desc_service_location Describes the elementary streams inside a PAT table for ATSC	141
atsc_desc_service_location_elementary Service location elementary descriptors	143
atsc_table_eit ATSC EIT table	145
atsc_table_eit_desc_length ATSC EIT descriptor length	146
atsc_table_eit_event ATSC EIT event table	147
atsc_table_mgt ATSC MGT table	150
atsc_table_mgt_table ATSC tables description at MGT table	152
atsc_table_vct ATSC VCT table (covers both CVCT and TVCT)	154
atsc_table_vct_channel ATSC VCT channel table (covers both CVCT and TVCT)	156

atsc_table_vct_descriptor_length	
ATSC VCT descriptor length	161
dvb_desc	
Linked list containing the several descriptors found on a MPEG-TS table	162
dvb_desc_ca	
Contains the private data for Conditional Access	163
dvb_desc_ca_identifier	
Indicates if a particular bouquet, service or event is associated with a CA system	165
dvb_desc_cable_delivery	
Structure containing the cable delivery system descriptor	167
dvb_desc_event_extended	
Structure containing the extended event descriptor	169
dvb_desc_event_extended_item	
	171
dvb_desc_event_short	
Structure containing the short event descriptor	172
dvb_desc_frequency_list	
Struct containing the frequency list descriptor	175
dvb_desc_hierarchy	
Structure containing the hierarchy descriptor	177
dvb_desc_language	
Structure containing the ISO639 language descriptor	179
dvb_desc_logical_channel	
Structure containing the logical channel number descriptor	180
dvb_desc_logical_channel_number	
Structure containing the logical channel number entires	181
dvb_desc_network_name	
Struct containing the network name descriptor	183
dvb_desc_registration	
Struct containing the frequency list descriptor	184
dvb_desc_sat	
Structure containing the satellite delivery system descriptor	186
dvb_desc_service	
Structure containing the service descriptor	188
dvb_desc_t2_delivery	
Structure containing the T2 delivery system descriptor	190
dvb_desc_t2_delivery_cell	
Structure to describe transponder cells	193
dvb_desc_t2_delivery_subcell	
Structure to describe transponder subcell extension and frequencies	195
dvb_desc_t2_delivery_subcell_old	
Structure to describe transponder subcell extension and frequencies	196

dvb_desc_terrestrial_delivery	197
Structure containing the DVB-T terrestrial delivery system descriptor	
dvb_desc_ts_info	200
Structure describing the ISDB TS information descriptor	
dvb_desc_ts_info_transmission_type	202
ISDB TS information transmission type	
dvb_descriptor	203
Contains the parser information for the MPEG-TS parser code	
dvb_dev_list	205
Digital TV device node properties	
dvb_device	207
Digital TV list of devices	
dvb_elementary_pid	208
Associates an elementary stream type with its PID	
dvb_entry	209
Represents one entry on a DTV file	
dvb_ext_descriptor	214
Structure that describes the parser functions for the extended descriptors	
dvb_extension_descriptor	216
Structure containing the extended descriptors	
dvb_file	217
Describes an entire DVB file opened	
dvb_mpeg_es_pic_start	218
MPEG ES Picture start header	
dvb_mpeg_es_seq_start	220
MPEG ES Sequence header	
dvb_mpeg_pes	223
MPEG PES data structure	
dvb_mpeg_pes_optional	225
MPEG PES optional header	
dvb_mpeg_ts	228
MPEG TS header	
dvb_mpeg_ts_adaption	231
MPEG TS header adaption field	
dvb_open_descriptor	233
Opaque struct with a DVB open file descriptor	
dvb_parse_file	234
Describes an entire file format	
dvb_parse_struct	235
Describes the format to parse an specific delivery system	
dvb_parse_table	236
Describes the fields to parse on a file	

dvb_sat_lnb	Stores the information of a LNBf	237
dvb_table_cat	ATSC CAT table	239
dvb_table_eit	DVB EIT table	240
dvb_table_eit_event	DVB EIT event table	242
dvb_table_filter	Describes the PES filters used by DVB scan	245
dvb_table_header	Header of a MPEG-TS table	246
dvb_table_nit	MPEG-TS NIT table	249
dvb_table_nit_transport	MPEG-TS NIT transport table	250
dvb_table_nit_transport_header	MPEG-TS NIT transport header	252
dvb_table_pat	MPEG-TS PAT table	253
dvb_table_pat_program	MPEG-TS PAT program table	254
dvb_table_pmt	MPEG-TS PMT table	256
dvb_table_pmt_stream	MPEG-TS PMT stream table	258
dvb_table_sdt	MPEG-TS SDT table	261
dvb_table_sdt_service	MPEG-TS SDT service table	262
dvb_ts_packet_header	Header of a MPEG-TS transport packet	265
dvb_v5_descriptors	Contains the descriptors needed to scan the Service ID and other relevant info at a MPEG-TS Digital TV stream	269
dvb_v5_descriptors_program	Associates PMT with PAT tables	272
dvb_v5_fe_parms	Keeps data needed to handle the DVB frontend	273
dvb_sat_lnb::dvbsat_freqrange		277
isdb_desc_partial_reception	Structure containing the partial reception descriptor	278

isdb_partial_reception_service_id	Service ID that uses partial reception	279
isdbt_desc_terrestrial_delivery_system	Struct containing the ISDB-T terrestrial delivery system	280
ts_t	MPEG PES timestamp structure, used for dts and pts	282

6 File Index

6.1 File List

Here is a list of all files with brief descriptions:

lib/include/libdvbv5/atsc_eit.h	Provides the table parser for the ATSC EIT (Event Information Table)	284
lib/include/libdvbv5/atsc_header.h	Provides some common ATSC stuff	287
lib/include/libdvbv5/cat.h	Provides the table parser for the CAT (Conditional Access Table)	288
lib/include/libdvbv5/countries.h	Provides ancillary code to convert ISO 3166-1 country codes	291
lib/include/libdvbv5/crc32.h	Provides ancillary code to calculate DVB crc32 checksum	296
lib/include/libdvbv5/desc_atsc_service_location.h	Provides the descriptors for ATSC service location	298
lib/include/libdvbv5/desc_ca.h	Provides the descriptors for Conditional Access	299
lib/include/libdvbv5/desc_ca_identifier.h	Provides the descriptors for the Conditional Access identifier	302
lib/include/libdvbv5/desc_cable_delivery.h	Provides the descriptors for the cable delivery system descriptor	304
lib/include/libdvbv5/desc_event_extended.h	Provides the descriptors for the extended event descriptor	307
lib/include/libdvbv5/desc_event_short.h	Provides the descriptors for the short event descriptor	308
lib/include/libdvbv5/desc_extension.h	Provides the descriptors for the extension descriptor	310
lib/include/libdvbv5/desc_frequency_list.h	Provides the descriptors for the frequency list descriptor	313
lib/include/libdvbv5/desc_hierarchy.h	Provides the descriptors for the hierarchy descriptor	314

lib/include/libdvbv5/desc_isdbt_delivery.h	316
Provides the descriptors for the ISDB-T terrestrial delivery system	
lib/include/libdvbv5/desc_language.h	318
Provides the descriptors for the ISO639 language descriptor	
lib/include/libdvbv5/desc_logical_channel.h	320
Provides the descriptors for the LCN - Logican Channel Number	
lib/include/libdvbv5/desc_network_name.h	322
Provides the descriptors for the network name descriptor	
lib/include/libdvbv5/desc_partial_reception.h	324
Provides the descriptors for the ISDB partial reception descriptor	
lib/include/libdvbv5/desc_registration_id.h	326
Provides the descriptors for the registration descriptor	
lib/include/libdvbv5/desc_sat.h	328
Provides the descriptors for the satellite delivery system descriptor	
lib/include/libdvbv5/desc_service.h	331
Provides the descriptors for the service descriptor	
lib/include/libdvbv5/desc_t2_delivery.h	332
Provides the descriptors for the DVB-T2 delivery system descriptor	
lib/include/libdvbv5/desc_terrestrial_delivery.h	335
Provides the descriptors for the DVB-T terrestrial delivery system descriptor	
lib/include/libdvbv5/desc_ts_info.h	339
Provides the descriptors for the ISDB TS information descriptor	
lib/include/libdvbv5/descriptors.h	341
Provides a way to handle MPEG-TS descriptors found on Digital TV streams	
lib/include/libdvbv5/dvb-demux.h	348
Provides interfaces to deal with DVB demux	
lib/include/libdvbv5/dvb-dev.h	349
Provides interfaces to handle Digital TV devices	
lib/include/libdvbv5/dvb-fe.h	355
Provides interfaces to deal with DVB frontend	
lib/include/libdvbv5/dvb-file.h	360
Provides interfaces to deal with DVB channel and program files	
lib/include/libdvbv5/dvb-log.h	365
Provides interfaces to handle libdvbv5 log messages	
lib/include/libdvbv5/dvb-sat.h	366
Provides interfaces to deal with DVB Satellite systems	
lib/include/libdvbv5/dvb-scan.h	368
Provides interfaces to scan programs inside MPEG-TS digital TV streams	
lib/include/libdvbv5/dvb-v5-std.h	372
Provides libdvbv5 defined properties for the frontend	
lib/include/libdvbv5/eit.h	376
Provides the table parser for the DVB EIT (Event Information Table)	

lib/include/libdvbv5/header.h	Provides the MPEG TS table headers	379
lib/include/libdvbv5/mgt.h	Provides the table parser for the ATSC MGT (Master Guide Table)	381
lib/include/libdvbv5/mpeg_es.h	Provides the table parser for the MPEG-TS Elementary Stream	384
lib/include/libdvbv5/mpeg_pes.h	Provides the table parser for the MPEG-PES Elementary Stream	387
lib/include/libdvbv5/mpeg_ts.h	Provides the table parser for the MPEG-PES Elementary Stream	390
lib/include/libdvbv5/nit.h	Provides the descriptors for NIT MPEG-TS table	392
lib/include/libdvbv5/pat.h	Provides the descriptors for PAT MPEG-TS table	395
lib/include/libdvbv5/pmt.h	Provides the descriptors for PMT MPEG-TS table	397
lib/include/libdvbv5/sdt.h	Provides the descriptors for SDT MPEG-TS table	401
lib/include/libdvbv5/vct.h	Provides the descriptors for TVCT and CVCT tables	404

7 Module Documentation

7.1 Digital TV device enumeration

Files

- file [dvb-dev.h](#)
Provides interfaces to handle Digital TV devices.

Data Structures

- struct [dvb_dev_list](#)
Digital TV device node properties.
- struct [dvb_device](#)
Digital TV list of devices.

Enumerations

- enum [dvb_dev_type](#) {
[DVB_DEVICE_FRONTEND](#) , [DVB_DEVICE_DEMUX](#) , [DVB_DEVICE_DVR](#) , [DVB_DEVICE_NET](#) ,
[DVB_DEVICE_CA](#) , [DVB_DEVICE_CA_SEC](#) , [DVB_DEVICE_VIDEO](#) , [DVB_DEVICE_AUDIO](#) }
Type of a device entry to search.

Functions

- struct [dvb_device](#) * [dvb_dev_alloc](#) (void)
Allocate a struct [dvb_device](#).
- void [dvb_dev_free](#) (struct [dvb_device](#) *dvb)
free a struct [dvb_device](#)
- int [dvb_dev_find](#) (struct [dvb_device](#) *dvb, [dvb_dev_change_t](#) handler, void *user_priv)
finds all DVB devices on the local machine
- void [dvb_dev_stop_monitor](#) (struct [dvb_device](#) *dvb)
Stop the [dvb_dev_find](#) loop.
- void [dvb_dev_set_logpriv](#) (struct [dvb_device](#) *dvb, unsigned verbose, [dvb_logfunc_priv](#) logfunc, void *logpriv)
Sets the DVB verbosity and log function with context private data.
- void [dvb_dev_set_log](#) (struct [dvb_device](#) *dvb, unsigned verbose, [dvb_logfunc](#) logfunc)
Sets the DVB verbosity and log function.
- struct [dvb_open_descriptor](#) * [dvb_dev_open](#) (struct [dvb_device](#) *dvb, const char *sysname, int flags)
Opens a dvb device.
- void [dvb_dev_close](#) (struct [dvb_open_descriptor](#) *open_dev)
Closes a dvb device.
- int [dvb_dev_get_fd](#) (struct [dvb_open_descriptor](#) *open_dev)
returns fd from a local device This will not work for remote devices.
- ssize_t [dvb_dev_read](#) (struct [dvb_open_descriptor](#) *open_dev, void *buf, size_t count)
read from a dvb demux or dvr file
- void [dvb_dev_dmx_stop](#) (struct [dvb_open_descriptor](#) *open_dev)
Stops the demux filter for a given file descriptor.
- int [dvb_dev_set_bufsize](#) (struct [dvb_open_descriptor](#) *open_dev, int buffersize)
Start a demux or dvr buffer size.
- int [dvb_dev_dmx_set_pesfilter](#) (struct [dvb_open_descriptor](#) *open_dev, int pid, [dmx_pes_type_t](#) type, [dmx_output_t](#) output, int buffersize)
Start a filter for a MPEG-TS Packetized Elementary Stream (PES)
- int [dvb_dev_dmx_set_section_filter](#) (struct [dvb_open_descriptor](#) *open_dev, int pid, unsigned filtsize, unsigned char *filter, unsigned char *mask, unsigned char *mode, unsigned int flags)
Sets a MPEG-TS section filter.
- int [dvb_dev_dmx_get_pmt_pid](#) (struct [dvb_open_descriptor](#) *open_dev, int sid)
read the contents of the MPEG-TS PAT table, seeking for an specific service ID

7.1.1 Detailed Description

7.1.2 Enumeration Type Documentation

7.1.2.1 [dvb_dev_type](#) `enum dvb_dev_type`

Type of a device entry to search.

Parameters

DVB_DEVICE_FRONTEND	Digital TV frontend
DVB_DEVICE_DEMUX	Digital TV demux
DVB_DEVICE_DVR	Digital TV Digital Video Record
DVB_DEVICE_NET	Digital TV network interface control
DVB_DEVICE_CA	Digital TV Conditional Access
DVB_DEVICE_CA_SEC	Digital TV Conditional Access serial

Enumerator

DVB_DEVICE_FRONTEND	
DVB_DEVICE_DEMUX	
DVB_DEVICE_DVR	
DVB_DEVICE_NET	
DVB_DEVICE_CA	
DVB_DEVICE_CA_SEC	
DVB_DEVICE_VIDEO	
DVB_DEVICE_AUDIO	

Definition at line 58 of file [dvb-dev.h](#).

7.1.3 Function Documentation

7.1.3.1 [dvb_dev_alloc\(\)](#) `struct dvb_device * dvb_dev_alloc (`
`void)`

Allocate a struct [dvb_device](#).

Note

Before using the dvb device function calls, the struct [dvb_device](#) should be allocated via this function call.

Returns

on success, returns a pointer to the allocated struct [dvb_device](#) or NULL if not enough memory to allocate the struct.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.1.3.2 [dvb_dev_close\(\)](#) `void dvb_dev_close (`
`struct dvb_open_descriptor * open_dev)`

Closes a dvb device.

Parameters

<i>open_dev</i>	Points to the struct dvb_open_descriptor to be closed.
-----------------	--

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.1.3.3 dvb_dev_dmx_get_pmt_pid() `int dvb_dev_dmx_get_pmt_pid (`
 `struct dvb_open_descriptor * open_dev,`
 `int sid)`

read the contents of the MPEG-TS PAT table, seeking for an specific service ID

Parameters

<i>open_dev</i>	Points to the struct dvb_open_descriptor
<i>sid</i>	Session ID to seeking

Returns

At return, it returns a negative value if error or the PID associated with the desired Session ID.

Warning

This function currently assumes that the PAT fits into one session.

Note

valid only for DVB_DEVICE_DEMUX.

Examples

[dvbv5-zap.c](#).

7.1.3.4 dvb_dev_dmx_set_pesfilter() `int dvb_dev_dmx_set_pesfilter (`
 `struct dvb_open_descriptor * open_dev,`
 `int pid,`
 `dmx_pes_type_t type,`
 `dmx_output_t output,`
 `int buffersize)`

Start a filter for a MPEG-TS Packetized Elementary Stream (PES)

Parameters

<i>open_dev</i>	Points to the struct dvb_open_descriptor
<i>pid</i>	Program ID to filter. Use 0x2000 to select all PIDs
<i>type</i>	type of the PID (DMX_PES_VIDEO, DMX_PES_AUDIO, DMX_PES_OTHER, etc).
<i>output</i>	Where the data will be output (DMX_OUT_TS_TAP, DMX_OUT_DECODER, etc).
<i>buffersize</i>	Size of the buffer to be allocated to store the filtered data.

This is a wrapper function for DMX_SET_PES_FILTER and DMX_SET_BUFFER_SIZE ioctls.

See http://linuxtv.org/downloads/v4l-dvb-apis/dvb_demux.html for more details.

Returns

Returns zero on success, -1 otherwise.

Note

valid only for DVB_DEVICE_DEMUX.

Examples

[dvbv5-zap.c](#).

```
7.1.3.5 dvb_dev_dmx_set_section_filter() int dvb_dev_dmx_set_section_filter (
    struct dvb_open_descriptor * open_dev,
    int pid,
    unsigned filtsize,
    unsigned char * filter,
    unsigned char * mask,
    unsigned char * mode,
    unsigned int flags )
```

Sets a MPEG-TS section filter.

Parameters

<i>open_dev</i>	Points to the struct dvb_open_descriptor
<i>pid</i>	Program ID to filter. Use 0x2000 to select all PIDs
<i>filtsize</i>	Size of the filter (up to 18 bytes)
<i>filter</i>	data to filter. Can be NULL or should have filtsize length
<i>mask</i>	filter mask. Can be NULL or should have filtsize length
<i>mode</i>	mode mask. Can be NULL or should have filtsize length
<i>flags</i>	flags for set filter (DMX_CHECK_CRC, DMX_ONESHOT, DMX_IMMEDIATE_START).

This is a wrapper function for DMX_SET_FILTER ioctl.

See http://linuxtv.org/downloads/v4l-dvb-apis/dvb_demux.html for more details.

Returns

Returns zero on success, -1 otherwise.

Note

valid only for DVB_DEVICE_DEMUX.

```
7.1.3.6 dvb_dev_dmx_stop() void dvb_dev_dmx_stop (
    struct dvb_open_descriptor * open_dev )
```

Stops the demux filter for a given file descriptor.

Parameters

<i>open_dev</i>	Points to the struct dvb_open_descriptor
-----------------	--

This is a wrapper function for DMX_STOP ioctl.

See http://linuxtv.org/downloads/v4l-dvb-apis/dvb_demux.html for more details.

Note

valid only for DVB_DEVICE_DEMUX.

```
7.1.3.7 dvb_dev_find() int dvb_dev_find (
    struct dvb\_device * dvb,
    dvb\_dev\_change\_t handler,
    void * user_priv )
```

finds all DVB devices on the local machine

Parameters

<i>dvb</i>	pointer to struct dvb_device to be filled
<i>enable_monitor</i>	if different than zero put the routine into monitor mode
<i>user_priv</i>	pointer to user private data

This routine can be called on two modes: normal or monitor mode

In normal mode, it will seek for the local Digital TV devices, store them at the struct [dvb_device](#) and return.

In monitor mode, it will not only enumerate all devices, but it will also keep waiting for device changes. The device seek loop will only be interrupted after calling [dvb_dev_stop_monitor\(\)](#).

Please notice that, in such mode, the function will wait forever. So, it is up to the application to put start a separate thread to handle it in monitor mode, and add the needed mutexes to make it thread safe.

Returns

returns 0 on success, a negative value otherwise.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

```
7.1.3.8 dvb_dev_free() void dvb_dev_free (
    struct dvb\_device * dvb )
```

free a struct [dvb_device](#)

Parameters

<i>dvb</i>	pointer to struct dvb_device to be freed
------------	--

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.1.3.9 `dvb_dev_get_fd()` `int dvb_dev_get_fd (`
`struct dvb_open_descriptor * open_dev)`

returns fd from a local device This will not work for remote devices.

Parameters

<i>open_dev</i>	Points to the struct dvb_open_descriptor
-----------------	--

Returns

On success, returns the fd. Returns -1 on error.

7.1.3.10 `dvb_dev_open()` `struct dvb_open_descriptor * dvb_dev_open (`
`struct dvb_device * dvb,`
`const char * sysname,`
`int flags)`

Opens a dvb device.

Parameters

<i>dvb</i>	pointer to struct dvb_device to be used
<i>sysname</i>	Kernel's name of the device to be opened, as obtained via dvb_dev_seek_by_adapter() or via dvb_dev_find() .
<i>flags</i>	Flags to be passed to open: O_RDONLY, O_RDWR and/or O_NONBLOCK

Note

Please notice that O_NONBLOCK is not supported for frontend devices, and will be silently ignored.
the sysname will only work if a previous call to [dvb_dev_find\(\)](#) is issued.

This function is equivalent to open(2) system call: it opens a Digital TV given by the dev parameter, using the flags.

Returns

returns a pointer to the `dvb_open_descriptor` that should be used on further calls if success. NULL otherwise.

Examples

`dvb-fe-tool.c`, `dvbv5-scan.c`, and `dvbv5-zap.c`.

7.1.3.11 `dvb_dev_read()` `ssize_t dvb_dev_read (`
 `struct dvb_open_descriptor * open_dev,`
 `void * buf,`
 `size_t count)`

read from a dvb demux or dvr file

Parameters

<i>open_dev</i>	Points to the struct <code>dvb_open_descriptor</code> to be closed.
<i>buf</i>	Buffer to store the data
<i>count</i>	number of bytes to read

Returns

On success, returns the number of bytes read. Returns -1 on error.

Examples

`dvbv5-zap.c`.

7.1.3.12 `dvb_dev_set_bufsize()` `int dvb_dev_set_bufsize (`
 `struct dvb_open_descriptor * open_dev,`
 `int buffersize)`

Start a demux or dvr buffer size.

Parameters

<i>open_dev</i>	Points to the struct <code>dvb_open_descriptor</code>
<i>buffersize</i>	Size of the buffer to be allocated to store the filtered data.

This is a wrapper function for `DMX_SET_BUFFER_SIZE` ioctl.

See http://linuxtv.org/downloads/v4l-dvb-apis/dvb_demux.html for more details.

Returns

Returns zero on success, -1 otherwise.

Note

valid only for DVB_DEVICE_DEMUX or DVB_DEVICE_DVR.

Examples

[dvbv5-zap.c](#).

7.1.3.13 dvb_dev_set_log() `void dvb_dev_set_log (`
 `struct dvb_device * dvb,`
 `unsigned verbose,`
 `dvb_logfunc logfunc)`

Sets the DVB verbosity and log function.

Parameters

<i>dvb</i>	pointer to struct dvb_device to be used
<i>verbose</i>	Verbosity level of the messages that will be printed
<i>logfunc</i>	Callback function to be called when a log event happens. Can either store the event into a file or to print it at the TUI/GUI. Can be null.

Sets the function to report log errors and to set the verbosity level of debug report messages. If not called, or if logfunc is NULL, the libdvbv5 will report error and debug messages via stderr, and will use colors for the debug messages.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.1.3.14 dvb_dev_set_logpriv() `void dvb_dev_set_logpriv (`
 `struct dvb_device * dvb,`
 `unsigned verbose,`
 `dvb_logfunc_priv logfunc,`
 `void * logpriv)`

Sets the DVB verbosity and log function with context private data.

Parameters

<i>dvb</i>	pointer to struct dvb_device to be used
<i>verbose</i>	Verbosity level of the messages that will be printed
<i>logfunc</i>	Callback function to be called when a log event happens. Can either store the event into a file or to print it at the TUI/GUI. Can be null.
<i>logpriv</i>	Private data for log function

Sets the function to report log errors and to set the verbosity level of debug report messages. If not called, or if

logfunc is NULL, the libdvbv5 will report error and debug messages via stderr, and will use colors for the debug messages.

7.1.3.15 dvb_dev_stop_monitor() `void dvb_dev_stop_monitor (`
`struct dvb_device * dvb)`

Stop the dvb_dev_find loop.

Parameters

<code>dvb</code>	pointer to struct <code>dvb_device</code> to be used
------------------	--

This function stops `dvb_dev_find()` if it is running in monitor mode. It does nothing on other modes. Can be called even if the monitor mode was already stopped.

7.2 Digital TV frontend control

Files

- file `dvb-fe.h`
Provides interfaces to deal with DVB frontend.
- file `dvb-v5-std.h`
Provides libdvbv5 defined properties for the frontend.

Data Structures

- struct `dvb_v5_fe_parms`
Keeps data needed to handle the DVB frontend.

Macros

- #define `MAX_DELIVERY_SYSTEMS`
Max number of delivery systems for a given frontend.
- #define `DTV_USER_COMMAND_START`
Start number for libdvbv5 user commands.
- #define `DTV_POLARIZATION`
Satellite polarization (for Satellite delivery systems)
- #define `DTV_AUDIO_PID`
Audio PID.
- #define `DTV_VIDEO_PID`
Video PID.
- #define `DTV_SERVICE_ID`
MPEG TS service ID.
- #define `DTV_CH_NAME`
Digital TV service name.
- #define `DTV_VCHANNEL`
Digital TV channel number.
- #define `DTV_SAT_NUMBER`

- *Number of the satellite (used on multi-dish Satellite systems)*
- `#define DTV_DISEQC_WAIT`
Extra time needed to wait for DiSeqC to complete, in ms.
- `#define DTV_DISEQC_LNB`
LNBf name.
- `#define DTV_FREQ_BPF`
SCR/Unicable band-pass filter frequency in kHz.
- `#define DTV_PLS_CODE`
DVB-T2 PLS code.
- `#define DTV_PLS_MODE`
DVB-T2 PLS mode.
- `#define DTV_COUNTRY_CODE`
Country variant of international delivery system standard.
- `#define DTV_MAX_USER_COMMAND`
Last user command.
- `#define DTV_USER_NAME_SIZE`
Number of user commands.
- `#define DTV_STAT_COMMAND_START`
Start number for libdvbv5 statistics commands.
- `#define DTV_STATUS`
Lock status of a DTV frontend.
- `#define DTV_BER`
Bit Error Rate.
- `#define DTV_PER`
Packet Error Rate.
- `#define DTV_QUALITY`
A quality indicator that represents if a locked channel provides a good, OK or poor signal.
- `#define DTV_PRE_BER`
Bit Error Rate before Viterbi.
- `#define DTV_MAX_STAT_COMMAND`
Last statistics command.
- `#define DTV_STAT_NAME_SIZE`
Number of statistics commands.
- `#define DTV_NUM_KERNEL_STATS`
Number of statistics commands provided by the Kernel.
- `#define DTV_NUM_STATS_PROPS`
Total number of statistics commands.

Enumerations

- `enum dvb_quality { DVB_QUAL_UNKNOWN , DVB_QUAL_POOR , DVB_QUAL_OK , DVB_QUAL_GOOD }`
Provides an estimation about the user's experience while watching to a given MPEG stream.

Functions

- struct `dvb_v5_fe_parms` * `dvb_fe_dummy` (void)
Allocates a dummy frontend structure.
- struct `dvb_v5_fe_parms` * `dvb_fe_open_flags` (int adapter, int frontend, unsigned verbose, unsigned use_↵ legacy_call, `dvb_logfunc` logfunc, int flags)
Opens a frontend and allocates a structure to work with.
- struct `dvb_v5_fe_parms` * `dvb_fe_open` (int adapter, int frontend, unsigned verbose, unsigned use_legacy_↵ _call)
Opens a frontend and allocates a structure to work with.
- struct `dvb_v5_fe_parms` * `dvb_fe_open2` (int adapter, int frontend, unsigned verbose, unsigned use_legacy_↵ _call, `dvb_logfunc` logfunc)
Opens a frontend and allocates a structure to work with.
- void `dvb_fe_close` (struct `dvb_v5_fe_parms` *parms)
Closes the frontend and frees allocated resources.
- const char * `dvb_cmd_name` (int cmd)
Returns the string name associated with a DVBv5 command.
- const char *const * `dvb_attr_names` (int cmd)
Returns an string array with the valid string values associated with a DVBv5 command.
- int `dvb_fe_retrieve_parm` (const struct `dvb_v5_fe_parms` *parms, unsigned cmd, uint32_t *value)
Retrieves the value of a DVBv5/libdvbv5 property.
- int `dvb_fe_store_parm` (struct `dvb_v5_fe_parms` *parms, unsigned cmd, uint32_t value)
Stores the value of a DVBv5/libdvbv5 property.
- int `dvb_set_sys` (struct `dvb_v5_fe_parms` *parms, fe_delivery_system_t sys)
Sets the delivery system.
- int `dvb_add_parms_for_sys` (struct `dvb_v5_fe_parms` *parms, fe_delivery_system_t sys)
Make dvb properties reflect the current standard.
- int `dvb_set_compat_delivery_system` (struct `dvb_v5_fe_parms` *parms, uint32_t desired_system)
Sets the delivery system.
- void `dvb_fe_prt_parms` (const struct `dvb_v5_fe_parms` *parms)
Prints all the properties at the cache.
- int `dvb_fe_set_parms` (struct `dvb_v5_fe_parms` *parms)
Prints all the properties at the cache.
- int `dvb_fe_get_parms` (struct `dvb_v5_fe_parms` *parms)
Prints all the properties at the cache.
- struct dtv_stats * `dvb_fe_retrieve_stats_layer` (struct `dvb_v5_fe_parms` *parms, unsigned cmd, unsigned layer)
Retrieve the stats for a DTV layer from cache.
- int `dvb_fe_retrieve_stats` (struct `dvb_v5_fe_parms` *parms, unsigned cmd, uint32_t *value)
Retrieve the stats for a DTV layer from cache.
- int `dvb_fe_get_stats` (struct `dvb_v5_fe_parms` *parms)
Retrieve the stats from the Kernel.
- float `dvb_fe_retrieve_ber` (struct `dvb_v5_fe_parms` *parms, unsigned layer, enum fecap_scale_params *scale)
Retrieve the BER stats from cache.
- float `dvb_fe_retrieve_per` (struct `dvb_v5_fe_parms` *parms, unsigned layer)
Retrieve the PER stats from cache.
- enum `dvb_quality` `dvb_fe_retrieve_quality` (struct `dvb_v5_fe_parms` *parms, unsigned layer)
Retrieve the quality stats from cache.
- int `dvb_fe_snprintf_eng` (char *buf, int len, float val)
Ancillary function to sprintf on ENG format.

- int [dvb_fe_snprintf_stat](#) (struct [dvb_v5_fe_parms](#) *parms, uint32_t cmd, char *display_name, int layer, char **buf, int *len, int *show_layer_name)
Ancillary function to sprintf on ENG format.
- int [dvb_fe_get_event](#) (struct [dvb_v5_fe_parms](#) *parms)
Get both status statistics and dvb parameters.
- int [dvb_fe_sec_voltage](#) (struct [dvb_v5_fe_parms](#) *parms, int on, int v18)
DVB ioctl wrapper for setting SEC voltage.
- int [dvb_fe_sec_tone](#) (struct [dvb_v5_fe_parms](#) *parms, fe_sec_tone_mode_t tone)
DVB ioctl wrapper for setting SEC tone.
- int [dvb_fe_lnb_high_voltage](#) (struct [dvb_v5_fe_parms](#) *parms, int on)
DVB ioctl wrapper for setting LNBf high voltage.
- int [dvb_fe_diseqc_burst](#) (struct [dvb_v5_fe_parms](#) *parms, int mini_b)
DVB ioctl wrapper for setting SEC DiSeqC tone burst to select between satellite A or B.
- int [dvb_fe_diseqc_cmd](#) (struct [dvb_v5_fe_parms](#) *parms, const unsigned len, const unsigned char *buf)
DVB ioctl wrapper for setting SEC DiSeqC command.
- int [dvb_fe_diseqc_reply](#) (struct [dvb_v5_fe_parms](#) *parms, unsigned *len, char *buf, int timeout)
DVB ioctl wrapper for getting SEC DiSeqC reply.
- int [dvb_fe_is_satellite](#) (uint32_t delivery_system)
DVB Ancillary routine to check if a given Delivery system is satellite.
- int [dvb_fe_set_default_country](#) (struct [dvb_v5_fe_parms](#) *parms, const char *country)
Set default country variant of delivery systems like ISDB-T.

7.2.1 Detailed Description

7.2.2 Macro Definition Documentation

7.2.2.1 DTV_AUDIO_PID `#define DTV_AUDIO_PID`

Audio PID.

Definition at line 111 of file [dvb-v5-std.h](#).

7.2.2.2 DTV_BER `#define DTV_BER`

Bit Error Rate.

This is a parameter that it is derivated from two counters at the Kernel side

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 200 of file [dvb-v5-std.h](#).

7.2.2.3 DTV_CH_NAME `#define DTV_CH_NAME`

Digital TV service name.

Definition at line 113 of file [dvb-v5-std.h](#).

7.2.2.4 DTV_COUNTRY_CODE `#define DTV_COUNTRY_CODE`

Country variant of international delivery system standard.

in ISO 3166-1 two letter code.

Definition at line 121 of file [dvb-v5-std.h](#).

7.2.2.5 DTV_DISEQC_LNB `#define DTV_DISEQC_LNB`

LNBf name.

Definition at line 117 of file [dvb-v5-std.h](#).

7.2.2.6 DTV_DISEQC_WAIT `#define DTV_DISEQC_WAIT`

Extra time needed to wait for DiSeqC to complete, in ms.

The minimal wait time is 15 ms. The time here will be added to the minimal time.

Definition at line 116 of file [dvb-v5-std.h](#).

7.2.2.7 DTV_FREQ_BPF `#define DTV_FREQ_BPF`

SCR/Unicable band-pass filter frequency in kHz.

Definition at line 118 of file [dvb-v5-std.h](#).

7.2.2.8 DTV_MAX_STAT_COMMAND `#define DTV_MAX_STAT_COMMAND`

Last statistics command.

Definition at line 205 of file [dvb-v5-std.h](#).

7.2.2.9 DTV_MAX_USER_COMMAND `#define DTV_MAX_USER_COMMAND`

Last user command.

Definition at line 123 of file [dvb-v5-std.h](#).

7.2.2.10 DTV_NUM_KERNEL_STATS `#define DTV_NUM_KERNEL_STATS`

Number of statistics commands provided by the Kernel.

Definition at line 210 of file [dvb-v5-std.h](#).

7.2.2.11 DTV_NUM_STATS_PROPS `#define DTV_NUM_STATS_PROPS`

Total number of statistics commands.

Definition at line 212 of file [dvb-v5-std.h](#).

7.2.2.12 DTV_PER `#define DTV_PER`

Packet Error Rate.

This is a parameter that it is derivated from two counters at the Kernel side

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 201 of file [dvb-v5-std.h](#).

7.2.2.13 DTV_PLS_CODE `#define DTV_PLS_CODE`

DVB-T2 PLS code.

Not used internally. It is needed only for file conversion.

Definition at line 119 of file [dvb-v5-std.h](#).

7.2.2.14 DTV_PLS_MODE `#define DTV_PLS_MODE`

DVB-T2 PLS mode.

Not used internally. It is needed only for file conversion.

Definition at line 120 of file [dvb-v5-std.h](#).

7.2.2.15 DTV_POLARIZATION `#define DTV_POLARIZATION`

Satellite polarization (for Satellite delivery systems)

Examples

[dvbv5-scan.c](#).

Definition at line 109 of file [dvb-v5-std.h](#).

7.2.2.16 DTV_PRE_BER `#define DTV_PRE_BER`

Bit Error Rate before Viterbi.

This is the error rate before applying the Forward Error Correction. This is a parameter that it is derivated from two counters at the Kernel side.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 203 of file [dvb-v5-std.h](#).

7.2.2.17 DTV_QUALITY `#define DTV_QUALITY`

A quality indicator that represents if a locked channel provides a good, OK or poor signal.

This is estimated considering the error rates, signal strength and/or S/N ratio of the carrier.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 202 of file [dvb-v5-std.h](#).

7.2.2.18 DTV_SAT_NUMBER `#define DTV_SAT_NUMBER`

Number of the satellite (used on multi-dish Satellite systems)

Definition at line 115 of file [dvb-v5-std.h](#).

7.2.2.19 DTV_SERVICE_ID `#define DTV_SERVICE_ID`

MPEG TS service ID.

Definition at line 112 of file [dvb-v5-std.h](#).

7.2.2.20 DTV_STAT_COMMAND_START `#define DTV_STAT_COMMAND_START`

Start number for libdvbv5 statistics commands.

Definition at line 197 of file [dvb-v5-std.h](#).

7.2.2.21 DTV_STAT_NAME_SIZE `#define DTV_STAT_NAME_SIZE`

Number of statistics commands.

Definition at line 207 of file [dvb-v5-std.h](#).

7.2.2.22 DTV_STATUS `#define DTV_STATUS`

Lock status of a DTV frontend.

This actually comes from the Kernel, but it uses a separate ioctl.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 199 of file [dvb-v5-std.h](#).

7.2.2.23 DTV_USER_COMMAND_START `#define DTV_USER_COMMAND_START`

Start number for libdvbv5 user commands.

Definition at line 107 of file [dvb-v5-std.h](#).

7.2.2.24 DTV_USER_NAME_SIZE `#define DTV_USER_NAME_SIZE`

Number of user commands.

Definition at line 125 of file [dvb-v5-std.h](#).

7.2.2.25 DTV_VCHANNEL `#define DTV_VCHANNEL`

Digital TV channel number.

May contain symbols

Definition at line 114 of file [dvb-v5-std.h](#).

7.2.2.26 DTV_VIDEO_PID `#define DTV_VIDEO_PID`

Video PID.

Definition at line 110 of file [dvb-v5-std.h](#).

7.2.2.27 MAX_DELIVERY_SYSTEMS `#define MAX_DELIVERY_SYSTEMS`

Max number of delivery systems for a given frontend.

Definition at line 71 of file [dvb-fe.h](#).

7.2.3 Enumeration Type Documentation**7.2.3.1 dvb_quality** `enum dvb_quality`

Provides an estimation about the user's experience while watching to a given MPEG stream.

Parameters

<i>DVB_QUAL_UNKNOWN</i>	Quality could not be estimated, as the Kernel driver doesn't provide enough statistics
<i>DVB_QUAL_POOR</i>	The signal reception is poor. Signal loss or packets can be lost too frequently.
<i>DVB_QUAL_OK</i>	The signal reception is ok. Eventual artifacts could be expected, but it should work.
<i>DVB_QUAL_GOOD</i>	The signal is good, and not many errors are happening. The user should have a good experience watching the stream.

Enumerator

DVB_QUAL_UNKNOWN	
DVB_QUAL_POOR	
DVB_QUAL_OK	
DVB_QUAL_GOOD	

Definition at line 231 of file [dvb-v5-std.h](#).

7.2.4 Function Documentation

7.2.4.1 `dvb_add_parms_for_sys()` `int dvb_add_parms_for_sys (`
 `struct dvb_v5_fe_parms * parms,`
 `fe_delivery_system_t sys)`

Make dvb properties reflect the current standard.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>sys</i>	delivery system to be selected

This function prepares the properties cache for a given delivery system.

It is automatically called by [dvb_set_sys\(\)](#), and should not be normally called, except when [dvb_fe_dummy\(\)](#) is used.

Returns

Return 0 if success, EINVAL otherwise.

7.2.4.2 `dvb_attr_names()` `const char *const * dvb_attr_names (`
 `int cmd)`

Returns an string array with the valid string values associated with a DVBv5 command.

Parameters

<i>cmd</i>	DVBv5 or libdvbv5 property
------------	----------------------------

Returns

it returns a string array that corresponds to the names associated with the possible values for that property, when available. For example: `dvb_cmd_name(DTV_CODE_RATE_HP)` would return an array with the possible values for the code rates: { "1/2", "2/3", ... NULL }

Note

The array always ends with NULL.

7.2.4.3 dvb_cmd_name() `const char * dvb_cmd_name (`
`int cmd)`

Returns the string name associated with a DVBv5 command.

Parameters

<i>cmd</i>	DVBv5 or libdvbv5 property
------------	----------------------------

This function gets an integer argument (*cmd*) and returns a string that corresponds to the name of that property.

Returns

it returns a string that corresponds to the property name. For example: `dvb_cmd_name(DTV_GUARD_INTERVAL)` would return "GUARD_INTERVAL". It also returns names for the properties used internally by libdvbv5.

7.2.4.4 dvb_fe_close() `void dvb_fe_close (`
`struct dvb_v5_fe_parms * parms)`

Closes the frontend and frees allocated resources.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
--------------	--

7.2.4.5 dvb_fe_diseqc_burst() `int dvb_fe_diseqc_burst (`
`struct dvb_v5_fe_parms * parms,`
`int mini_b)`

DVB ioctl wrapper for setting SEC DiSeqC tone burst to select between satellite A or B.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
<i>mini_b</i>	if different than zero, sends a 22 KHz tone burst to select satellite B. Otherwise, sends tone to select satellite A.

Valid only on certain DISEqC arrangements.

If `dvb_v5_fe_parms::lnb` is set, this is controlled automatically.

7.2.4.6 dvb_fe_diseqc_cmd() `int dvb_fe_diseqc_cmd (`
 `struct dvb_v5_fe_parms * parms,`
 `const unsigned len,`
 `const unsigned char * buf)`

DVB ioctl wrapper for setting SEC DiSeqC command.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
<i>len</i>	size of the DiSeqC command
<i>buf</i>	DiSeqC command to be sent

If `dvb_v5_fe_parms::lnb` is set, this is controlled automatically.

7.2.4.7 dvb_fe_diseqc_reply() `int dvb_fe_diseqc_reply (`
 `struct dvb_v5_fe_parms * parms,`
 `unsigned * len,`
 `char * buf,`
 `int timeout)`

DVB ioctl wrapper for getting SEC DiSeqC reply.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
<i>len</i>	size of the DiSeqC command
<i>buf</i>	DiSeqC command to be sent
<i>timeout</i>	maximum time to receive the command, in ms.

If `dvb_v5_fe_parms::lnb` is set, this is controlled automatically.

7.2.4.8 dvb_fe_dummy() `struct dvb_v5_fe_parms * dvb_fe_dummy (`
 `void)`

Allocates a dummy frontend structure.

This is useful for some applications that may want to just use the frontend structure internally, without associating it with a real hardware

Returns

Returns a pointer to a dummy struct, or NULL if no memory.

7.2.4.9 dvb_fe_get_event() `int dvb_fe_get_event (`
 `struct dvb_v5_fe_parms * parms)`

Get both status statistics and dvb parameters.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
--------------	---

That's similar of calling both [dvb_fe_get_parms\(\)](#) and [dvb_fe_get_stats\(\)](#).

Returns

It returns 0 if success or an errorno otherwise.

7.2.4.10 dvb_fe_get_parms() `int dvb_fe_get_parms (`
`struct dvb_v5_fe_parms * parms)`

Prints all the properties at the cache.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
--------------	---

Gets the properties from the DVB hardware. The values will only reflect what's set at the hardware if the frontend is locked.

Returns

Return 0 if success, EINVAL otherwise.

Examples

[dvb-fe-tool.c](#).

7.2.4.11 dvb_fe_get_stats() `int dvb_fe_get_stats (`
`struct dvb_v5_fe_parms * parms)`

Retrieve the stats from the Kernel.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
--------------	---

Updates the stats cache from the available stats at the Kernel.

Returns

The returned value is 0 if success, EINVAL otherwise.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.2.4.12 dvb_fe_is_satellite() `int dvb_fe_is_satellite (`
 `uint32_t delivery_system)`

DVB Ancillary routine to check if a given Delivery system is satellite.

Parameters

<i>delivery_system</i>	delivery system to be selected
------------------------	--------------------------------

7.2.4.13 dvb_fe_lnb_high_voltage() `int dvb_fe_lnb_high_voltage (`
 `struct dvb_v5_fe_parms * parms,`
 `int on)`

DVB ioctl wrapper for setting LNBf high voltage.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>on</i>	a value different than zero indicates to produce lightly higher voltages instead of 13/18V, in order to compensate for long cables.

7.2.4.14 dvb_fe_open() `struct dvb_v5_fe_parms * dvb_fe_open (`
 `int adapter,`
 `int frontend,`
 `unsigned verbose,`
 `unsigned use_legacy_call)`

Opens a frontend and allocates a structure to work with.

Parameters

<i>adapter</i>	Number of the adapter to open
<i>frontend</i>	Number of the frontend to open
<i>verbose</i>	Verbosity level of the messages that will be printed
<i>use_legacy_call</i>	Force to use the DVBv3 calls, instead of using the DVBv5 API

This function should be called before using any other function at the frontend library (or the other alternatives: [dvb_fe_open2\(\)](#) or [dvb_fe_dummy\(\)](#)).

Returns

Returns a pointer to an allocated data pointer or NULL on error.

7.2.4.15 dvb_fe_open2() struct [dvb_v5_fe_parms](#) * dvb_fe_open2 (
 int *adapter*,
 int *frontend*,
 unsigned *verbose*,
 unsigned *use_legacy_call*,
 [dvb_logfunc](#) *logfunc*)

Opens a frontend and allocates a structure to work with.

Parameters

<i>adapter</i>	Number of the adapter to open
<i>frontend</i>	Number of the frontend to open
<i>verbose</i>	Verbosity level of the messages that will be printed
<i>use_legacy_call</i>	Force to use the DVBv3 calls, instead of using the DVBv5 API
<i>logfunc</i>	Callback function to be called when a log event happens. Can either store the event into a file or to print it at the TUI/GUI.

This function should be called before using any other function at the frontend library (or the other alternatives: [dvb_fe_open\(\)](#) or [dvb_fe_dummy\(\)](#)).

Returns

Returns a pointer to an allocated data pointer or NULL on error.

7.2.4.16 dvb_fe_open_flags() struct [dvb_v5_fe_parms](#) * dvb_fe_open_flags (
 int *adapter*,
 int *frontend*,
 unsigned *verbose*,
 unsigned *use_legacy_call*,
 [dvb_logfunc](#) *logfunc*,
 int *flags*)

Opens a frontend and allocates a structure to work with.

Parameters

<i>adapter</i>	Number of the adapter to open
<i>frontend</i>	Number of the frontend to open
<i>verbose</i>	Verbosity level of the messages that will be printed
<i>use_legacy_call</i>	Force to use the DVBv3 calls, instead of using the DVBv5 API
<i>logfunc</i>	Callback function to be called when a log event happens. Can either store the event into a file or to print it at the TUI/GUI. If NULL, the library will use its internal handler.
<i>flags</i>	Flags to be passed to open. Currently only two flags are supported: O_RDONLY or O_RDWR. Using O_NONBLOCK may hit unexpected issues.

Todo Add/check support for O_NONBLOCK at the scan routines.

This function should be called before using any other function at the frontend library (or the other alternatives: `dvb_fe_open()` or `dvb_fe_dummy()`).

In general, this is called using O_RDWR, except if all that it is wanted is to check the DVB frontend statistics.

Returns

Returns a pointer to an allocated data pointer or NULL on error.

7.2.4.17 dvb_fe_prt_parms() `void dvb_fe_prt_parms (`
`const struct dvb_v5_fe_parms * parms)`

Prints all the properties at the cache.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
--------------	--

Used mostly for debugging issues.

Examples

`dvb-fe-tool.c`.

7.2.4.18 dvb_fe_retrieve_ber() `float dvb_fe_retrieve_ber (`
`struct dvb_v5_fe_parms * parms,`
`unsigned layer,`
`enum fecap_scale_params * scale)`

Retrieve the BER stats from cache.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
<i>layer</i>	DTV layer
<i>scale</i>	retrieves the scale

Gets the value for BER stats from stats cache, on a given layer. Layer 0 is always present. On DTV standards that doesn't have layers, it returns the same value as `dvb_fe_retrieve_stats()` for layer = 0.

For DTV standards with multiple layers, like ISDB, layer=1 is layer 'A', layer=2 is layer 'B' and layer=3 is layer 'C'. Please notice that not all frontends support per-layer stats. Also, the layer value is only valid if the layer exists at the original stream. Also, on such standards, layer 0 is typically a mean value of the layers, or a sum of events (if FE_SCALE_COUNTER).

For it to be valid, [dvb_fe_get_stats\(\)](#) should be called first.

Returns

It returns a float number for the BER value. If the statistics is not available for any reason, scale will be equal to FE_SCALE_NOT_AVAILABLE.

7.2.4.19 dvb_fe_retrieve_parm() `int dvb_fe_retrieve_parm (
 const struct dvb_v5_fe_parms * parms,
 unsigned cmd,
 uint32_t * value)`

Retrieves the value of a DVBv5/libdvbv5 property.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>cmd</i>	DVBv5 or libdvbv5 property
<i>value</i>	Pointer to an uint32_t where the value will be stored.

This reads the value of a property stored at the cache. Before using it, a [dvb_fe_get_parms\(\)](#) is likely required.

Returns

Return 0 if success, EINVAL otherwise.

Examples

[dvbv5-zap.c](#).

7.2.4.20 dvb_fe_retrieve_per() `float dvb_fe_retrieve_per (
 struct dvb_v5_fe_parms * parms,
 unsigned layer)`

Retrieve the PER stats from cache.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>layer</i>	DTV layer

Gets the value for BER stats from stats cache, on a given layer. Layer 0 is always present. On DTV standards that doesn't have layers, it returns the same value as [dvb_fe_retrieve_stats\(\)](#) for layer = 0.

For DTV standards with multiple layers, like ISDB, layer=1 is layer 'A', layer=2 is layer 'B' and layer=3 is layer 'C'. Please notice that not all frontends support per-layer stats. Also, the layer value is only valid if the layer exists at

the original stream. Also, on such standards, layer 0 is typically a mean value of the layers, or a sum of events (if FE_SCALE_COUNTER).

For it to be valid, [dvb_fe_get_stats\(\)](#) should be called first.

Returns

A negative value indicates error.

7.2.4.21 dvb_fe_retrieve_quality() enum [dvb_quality](#) dvb_fe_retrieve_quality (
 struct [dvb_v5_fe_parms](#) * *parms*,
 unsigned *layer*)

Retrieve the quality stats from cache.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>layer</i>	DTV layer

Gets a quality measure for a given layer. Layer 0 is always present. On DTV standards that doesn't have layers, it returns the same value as [dvb_fe_retrieve_stats\(\)](#) for layer = 0.

For DTV standards with multiple layers, like ISDB, layer=1 is layer 'A', layer=2 is layer 'B' and layer=3 is layer 'C'. Please notice that not all frontends support per-layer stats. Also, the layer value is only valid if the layer exists at the original stream. Also, on such standards, layer 0 is typically a mean value of the layers, or a sum of events (if FE_SCALE_COUNTER).

For it to be valid, [dvb_fe_get_stats\(\)](#) should be called first.

Returns

returns an enum [dvb_quantity](#), where DVB_QUAL_UNKNOWN means that the stat is not available.

Examples

[dvb-fe-tool.c](#).

7.2.4.22 dvb_fe_retrieve_stats() int dvb_fe_retrieve_stats (
 struct [dvb_v5_fe_parms](#) * *parms*,
 unsigned *cmd*,
 uint32_t * *value*)

Retrieve the stats for a DTV layer from cache.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>cmd</i>	DVBv5 or libdvb5 property
<i>value</i>	DTV value pointer

Gets the value for one stats property for layer = 0.

For it to be valid, [dvb_fe_get_stats\(\)](#) should be called first.

Returns

The returned value is 0 if success, EINVAL otherwise.

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.2.4.23 dvb_fe_retrieve_stats_layer() `struct dtv_stats * dvb_fe_retrieve_stats_layer (struct dvb_v5_fe_parms * parms, unsigned cmd, unsigned layer)`

Retrieve the stats for a DTV layer from cache.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>cmd</i>	DVBv5 or libdvbv5 property
<i>layer</i>	DTV layer

Gets the value for one stats cache, on a given layer. Layer 0 is always present. On DTV standards that doesn't have layers, it returns the same value as [dvb_fe_retrieve_stats\(\)](#) for layer = 0.

For DTV standards with multiple layers, like ISDB, layer=1 is layer 'A', layer=2 is layer 'B' and layer=3 is layer 'C'. Please notice that not all frontends support per-layer stats. Also, the layer value is only valid if the layer exists at the original stream. Also, on such standards, layer 0 is typically a mean value of the layers, or a sum of events (if FE_SCALE_COUNTER).

For it to be valid, [dvb_fe_get_stats\(\)](#) should be called first.

Returns

It returns a struct dtv_stats if succeed or NULL otherwise.

7.2.4.24 dvb_fe_sec_tone() `int dvb_fe_sec_tone (struct dvb_v5_fe_parms * parms, fe_sec_tone_mode_t tone)`

DVB ioctl wrapper for setting SEC tone.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>tone</i>	tone setting, as defined by DVB fe_sec_tone_mode_t type

If `dvb_v5_fe_parms::lnb` is set, this is controlled automatically.

7.2.4.25 dvb_fe_sec_voltage() `int dvb_fe_sec_voltage (`
 `struct dvb_v5_fe_parms * parms,`
 `int on,`
 `int v18)`

DVB ioctl wrapper for setting SEC voltage.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>on</i>	a value different than zero indicates to enable voltage on a Satellite Equipment Control (SEC)
<i>v18</i>	if on != 0, a value different than zero means 18 Volts; zero means 13 Volts.

If `dvb_v5_fe_parms::lnb` is set, this is controlled automatically.

7.2.4.26 dvb_fe_set_default_country() `int dvb_fe_set_default_country (`
 `struct dvb_v5_fe_parms * parms,`
 `const char * country)`

Set default country variant of delivery systems like ISDB-T.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>country</i>	default country, in ISO 3166-1 two letter code. If NULL, default charset is guessed from locale environment variables.

Returns

0 if success or an error otherwise.

"COUNTRY" property in `dvb_fe_set_parm()` overrides the setting.

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.2.4.27 dvb_fe_set_parms() `int dvb_fe_set_parms (`
 `struct dvb_v5_fe_parms * parms)`

Prints all the properties at the cache.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
--------------	---

Writes the properties stored at the DVB cache at the DVB hardware. At return, some properties could have a different value, as the frontend may not support the values set.

Returns

Return 0 if success, EINVAL otherwise.

Examples

[dvbv5-zap.c](#).

7.2.4.28 dvb_fe_snprintf_eng() `int dvb_fe_snprintf_eng (`
`char * buf,`
`int len,`
`float val)`

Ancillary function to sprintf on ENG format.

Parameters

<i>buf</i>	buffer to store the value
<i>len</i>	buffer length
<i>val</i>	value to be printed

On ENG notation, the exponential value should be multiple of 3. This is good to display some values, like BER.

Returns

At return, it shows the actual size of the print. A negative value indicates an error.

7.2.4.29 dvb_fe_snprintf_stat() `int dvb_fe_snprintf_stat (`
`struct dvb_v5_fe_parms * parms,`
`uint32_t cmd,`
`char * display_name,`
`int layer,`
`char ** buf,`
`int * len,`
`int * show_layer_name)`

Ancillary function to sprintf on ENG format.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>cmd</i>	DVBv5 or libdvbv5 property
<i>display_name</i>	String with the name of the property to be shown
<i>layer</i>	DTV Layer
<i>buf</i>	buffer to store the value
<i>len</i>	buffer length
<i>show_layer_name</i>	a value different than zero shows the layer name, if the layer is bigger than zero.

This function calls internally `dvb_fe_retrieve_stats_layer()`. It allows to print a DVBv5 statistics value into a string. An extra property is available (DTV_QUALITY) with prints either one of the values: Poor, Ok or Good, depending on the overall measures.

Returns

: It returns the length of the printed data. A negative value indicates an error.

Examples

`dvb-fe-tool.c`, `dvbv5-scan.c`, and `dvbv5-zap.c`.

7.2.4.30 `dvb_fe_store_parm()` `int dvb_fe_store_parm (`
 `struct dvb_v5_fe_parms * parms,`
 `unsigned cmd,`
 `uint32_t value)`

Stores the value of a DVBv5/libdvbv5 property.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
<i>cmd</i>	DVBv5 or libdvbv5 property
<i>value</i>	Pointer to an <code>uint32_t</code> where the value will be stored.

This stores the value of a property at the cache. The value will only be send to the hardware after calling `dvb_fe_set_parms()`.

Returns

Return 0 if success, EINVAL otherwise.

Examples

`dvbv5-zap.c`.

7.2.4.31 `dvb_set_compat_delivery_system()` `int dvb_set_compat_delivery_system (`
 `struct dvb_v5_fe_parms * parms,`
 `uint32_t desired_system)`

Sets the delivery system.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> pointer to the opened device
<i>desired_system</i>	delivery system to be selected

This function changes the delivery system of the frontend. By default, the libdvbv5 will use the first available delivery system. If another delivery system is desirable, this function should be called before being able to store the properties for the new delivery system via [dvb_fe_store_parm\(\)](#).

This function is an enhanced version of [dvb_set_sys\(\)](#). It has an special logic inside to work with Kernels that supports only DVBv3.

Returns

Return 0 if success, EINVAL otherwise.

Examples

[dvbv5-zap.c](#).

```
7.2.4.32 dvb_set_sys()  int dvb_set_sys (
    struct dvb\_v5\_fe\_parms * parms,
    fe_delivery_system_t sys )
```

Sets the delivery system.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>sys</i>	delivery system to be selected

This function changes the delivery system of the frontend. By default, the libdvbv5 will use the first available delivery system. If another delivery system is desirable, this function should be called before being able to store the properties for the new delivery system via [dvb_fe_store_parm\(\)](#).

Returns

Return 0 if success, EINVAL otherwise.

Examples

[dvb-fe-tool.c](#).

7.3 Digital TV frontend scan

Files

- file [dvb-scan.h](#)
Provides interfaces to scan programs inside MPEG-TS digital TV streams.

Data Structures

- struct [dvb_v5_descriptors_program](#)
Associates PMT with PAT tables.
- struct [dvb_v5_descriptors](#)
Contains the descriptors needed to scan the Service ID and other relevant info at a MPEG-TS Digital TV stream.
- struct [dvb_table_filter](#)
Describes the PES filters used by DVB scan.

Typedefs

- typedef int() [check_frontend_t](#)(void *args, struct [dvb_v5_fe_parms](#) *parms)

Callback for the application to show the frontend status.

Functions

- struct [dvb_v5_descriptors](#) * [dvb_dev_scan](#) (struct [dvb_open_descriptor](#) *open_dev, struct [dvb_entry](#) *entry, [check_frontend_t](#) *check_frontend, void *args, unsigned other_nit, unsigned timeout_multiply)
Scans a DVB dvb_add_scanned_transponder.
- void [dvb_table_filter_free](#) (struct [dvb_table_filter](#) *sect)
deallocates all data associated with a table filter
- int [dvb_read_section](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, unsigned char tid, uint16_t pid, void **table, unsigned timeout)
read MPEG-TS tables that comes from a DTV card
- int [dvb_read_section_with_id](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, unsigned char tid, uint16_t pid, int ts_id, void **table, unsigned timeout)
read MPEG-TS tables that comes from a DTV card with an specific table section ID
- int [dvb_read_sections](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, struct [dvb_table_filter](#) *sect, unsigned timeout)
read MPEG-TS tables that comes from a DTV card
- struct [dvb_v5_descriptors](#) * [dvb_scan_alloc_handler_table](#) (uint32_t delivery_system)
allocates a struct dvb_v5_descriptors
- void [dvb_scan_free_handler_table](#) (struct [dvb_v5_descriptors](#) *dvb_scan_handler)
frees a struct dvb_v5_descriptors
- struct [dvb_v5_descriptors](#) * [dvb_get_ts_tables](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, uint32_t delivery_system, unsigned other_nit, unsigned timeout_multiply)
Scans a DVB stream, looking for the tables needed to identify the programs inside a MPEG-TS.
- void [dvb_free_ts_tables](#) (struct [dvb_v5_descriptors](#) *dvb_desc)
frees a struct dvb_v5_descriptors
- struct [dvb_v5_descriptors](#) * [dvb_scan_transponder](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_entry](#) *entry, int dmxfid, [check_frontend_t](#) *check_frontend, void *args, unsigned other_nit, unsigned timeout_multiply)
Scans a DVB dvb_add_scanned_transponder.
- void [dvb_add_scanned_transponders](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_v5_descriptors](#) *dvb_scan_handler, struct [dvb_entry](#) *first_entry, struct [dvb_entry](#) *entry)
Add new transponders to a dvb_file.

7.3.1 Detailed Description

7.3.2 Typedef Documentation

7.3.2.1 [check_frontend_t](#) typedef int() [check_frontend_t](#)(void *args, struct [dvb_v5_fe_parms](#) *parms)

Callback for the application to show the frontend status.

Parameters

<i>args</i>	a pointer, opaque to libdvbv5, to be used by the application if needed.
<i>parms</i>	pointer to struct dvb_v5_fe_parms created when the frontend is opened

Definition at line 293 of file [dvb-scan.h](#).

7.3.3 Function Documentation

7.3.3.1 [dvb_add_scanned_transponders\(\)](#) void [dvb_add_scanned_transponders](#) (

```

    struct dvb\_v5\_fe\_parms * parms,
    struct dvb\_v5\_descriptors * dvb_scan_handler,
    struct dvb\_entry * first_entry,
    struct dvb\_entry * entry )

```

Add new transponders to a [dvb_file](#).

Parameters

<i>parms</i>	pointer to struct dvb_v5_fe_parms created when the frontend is opened
<i>dvb_scan_handler</i>	pointer to a struct dvb_v5_descriptors containing scanned MPEG-TS
<i>first_entry</i>	first entry of a DVB file struct
<i>entry</i>	current entry on a DVB file struct

When the NIT table is parsed, some new transponders could be described inside. This function adds new entries to a [dvb_file](#) struct, pointing to those new transponders. It is used inside the scan loop, as shown at the [dvb_scan_transponder\(\)](#), to add new channels.

Example:

```

for (entry = dvb_file->first_entry; entry != NULL; entry = entry->next) {
    struct dvb\_v5\_descriptors *dvb_scan_handler = NULL;
    dvb_scan_handler = dvb\_scan\_transponder(parms, entry, dmxf_fd,
                                           &check_frontend, args,
                                           args->other_nit,
                                           args->timeout_multiply);

    if (parms->abort) {
        dvb\_scan\_free\_handler\_table(dvb_scan_handler);
        break;
    }
    if (dvb_scan_handler) {
        dvb\_store\_channel(&dvb_file_new, parms, dvb_scan_handler,
                        args->get_detected, args->get_nit);
        dvb\_scan\_free\_handler\_table(dvb_scan_handler);
        dvb\_add\_scanned\_transponders(parms, dvb_scan_handler,
                                   dvb_file->first_entry, entry);
        dvb\_scan\_free\_handler\_table(dvb_scan_handler);
    }
}

```

Examples

[dvbv5-scan.c](#).

7.3.3.2 dvb_dev_scan() `struct dvb_v5_descriptors * dvb_dev_scan (`
`struct dvb_open_descriptor * open_dev,`
`struct dvb_entry * entry,`
`check_frontend_t * check_frontend,`
`void * args,`
`unsigned other_nit,`
`unsigned timeout_multiply)`

Scans a DVB `dvb_add_scanned_transponder`.

Parameters

<i>entry</i>	DVB file entry that corresponds to a transponder to be tuned
<i>open_dev</i>	Points to the struct <code>dvb_open_descriptor</code>
<i>check_frontend</i>	a pointer to a function that will show the frontend status while tuning into a transponder
<i>args</i>	a pointer, opaque to libdvbv5, that will be used when calling <code>check_frontend</code> . It should contain any parameters that could be needed by <code>check_frontend</code> .
<i>other_nit</i>	Use alternate table IDs for NIT and other tables
<i>timeout_multiply</i>	Improves the timeout for each table reception, by

This is the function that applications should use when doing a transponders scan. It does everything needed to fill the entries with DVB programs (virtual channels) and detect the PIDs associated with them.

This is the `dvb_device` variant of `dvb_scan_transponder()`.

Examples

`dvbv5-scan.c`.

7.3.3.3 dvb_free_ts_tables() `void dvb_free_ts_tables (`
`struct dvb_v5_descriptors * dvb_desc)`

frees a struct `dvb_v5_descriptors`

Parameters

<i>dvb_desc</i>	pointed to the structure to be freed.
-----------------	---------------------------------------

This function recursively frees everything that is allocated by `dvb_get_ts_tables()` and stored at `dvb_desc`, including `dvb_desc` itself.

7.3.3.4 dvb_get_ts_tables() `struct dvb_v5_descriptors * dvb_get_ts_tables (`
`struct dvb_v5_fe_parms * parms,`
`int dmx_fd,`
`uint32_t delivery_system,`
`unsigned other_nit,`
`unsigned timeout_multiply)`

Scans a DVB stream, looking for the tables needed to identify the programs inside a MPEG-TS.

Parameters

<i>parms</i>	pointer to struct dvb_v5_fe_parms created when the frontend is opened
<i>dmx_fd</i>	an opened demux file descriptor
<i>delivery_system</i>	delivery system to be scanned
<i>other_nit</i>	use alternate table IDs for NIT and other tables
<i>timeout_multiply</i>	improves the timeout for each table reception by using a value that will multiply the wait time.

Given an opened frontend and demux, this function seeks for all programs available at the transport stream, and parses the following tables: PAT, PMT, NIT, SDT (and VCT, if the delivery system is ATSC).

On success, it returns a pointer to a struct [dvb_v5_descriptors](#), that can either be used to tune into a service or to be stored inside a file.

```
7.3.3.5  dvb_read_section()  int dvb_read_section (
    struct dvb_v5_fe_parms * parms,
    int dmx_fd,
    unsigned char tid,
    uint16_t pid,
    void ** table,
    unsigned timeout )
```

read MPEG-TS tables that comes from a DTV card

Parameters

<i>parms</i>	pointer to struct dvb_v5_fe_parms created when the frontend is opened
<i>dmx_fd</i>	an opened demux file descriptor
<i>tid</i>	Table ID
<i>pid</i>	Program ID
<i>table</i>	pointer to a pointer for the table struct to be filled
<i>timeout</i>	Limit, in seconds, to read a MPEG-TS table

This function is used to read the DVB tables by specifying a table ID and a program ID. The libdvbv5 should have a parser for the descriptors of the table type that should be parsed. The table will be automatically allocated on success. The function will read on the specified demux and return when reading is done or an error has occurred. If table is not NULL after the call, it has to be freed with the appropriate free table function (even if an error has occurred).

If the application wants to abort the read operation, it can change the value of `parms->p.abort` to 1.

Returns 0 on success or a negative error code.

Example usage:

```
struct dvb_table_pat *pat;
int r = dvb_read_section( parms, dmx_fd, DVB_TABLE_PAT, DVB_TABLE_PAT_PID,
    (void **) &pat, 5 );

if (r < 0)
    dvb_logerr("error reading PAT table");
else {
    // do something with pat
}
if (pat)
    dvb_table_pat_free( pat );
```

```

7.3.3.6 dvb_read_section_with_id() int dvb_read_section_with_id (
    struct dvb_v5_fe_parms * parms,
    int dmxf↔fd,
    unsigned char tid,
    uint16_t pid,
    int ts_id,
    void ** table,
    unsigned timeout )

```

read MPEG-TS tables that comes from a DTV card with an specific table section ID

Parameters

<i>parms</i>	pointer to struct dvb_v5_fe_parms created when the frontend is opened
<i>dmxf_↔fd</i>	an opened demux file descriptor
<i>tid</i>	Table ID
<i>pid</i>	Program ID
<i>ts_id</i>	Table section ID (for multisession filtering). If no specific table section is needed, -1 should be used
<i>table</i>	pointer to a pointer for the table struct to be filled
<i>timeout</i>	limit, in seconds, to read a MPEG-TS table

This is a variant of [dvb_read_section\(\)](#) that also seeks for an specific table section ID given by *ts_id*.

```

7.3.3.7 dvb_read_sections() int dvb_read_sections (
    struct dvb_v5_fe_parms * parms,
    int dmxf↔fd,
    struct dvb_table_filter * sect,
    unsigned timeout )

```

read MPEG-TS tables that comes from a DTV card

Parameters

<i>parms</i>	pointer to struct dvb_v5_fe_parms created when the frontend is opened
<i>dmxf_↔fd</i>	an opened demux file descriptor
<i>sect</i>	section filter pointer
<i>timeout</i>	limit, in seconds, to read a MPEG-TS table

This is a variant of [dvb_read_section\(\)](#) that uses a struct [dvb_table_filter](#) to specify the filter to use.

```

7.3.3.8 dvb_scan_alloc_handler_table() struct dvb_v5_descriptors * dvb_scan_alloc_handler_table
(
    uint32_t delivery_system )

```

allocates a struct [dvb_v5_descriptors](#)

Parameters

<i>delivery_system</i>	Delivery system to be used on the table
------------------------	---

At success, returns a pointer. NULL otherwise.

7.3.3.9 dvb_scan_free_handler_table() void dvb_scan_free_handler_table (struct dvb_v5_descriptors * dvb_scan_handler)

frees a struct `dvb_v5_descriptors`

Parameters

<code>dvb_scan_handler</code>	pointer to the struct to be freed.
-------------------------------	------------------------------------

Examples

[dvbv5-scan.c](#).

7.3.3.10 dvb_scan_transponder() struct dvb_v5_descriptors * dvb_scan_transponder (struct dvb_v5_fe_parms * parms, struct dvb_entry * entry, int dmx_fd, check_frontend_t * check_frontend, void * args, unsigned other_nit, unsigned timeout_multiply)

Scans a DVB `dvb_add_scanned_transponder`.

Parameters

<code>parms</code>	pointer to struct <code>dvb_v5_fe_parms</code> created when the frontend is opened
<code>entry</code>	DVB file entry that corresponds to a transponder to be tuned
<code>dmx_fd</code>	an opened demux file descriptor
<code>check_frontend</code>	a pointer to a function that will show the frontend status while tuning into a transponder
<code>args</code>	a pointer, opaque to libdvbv5, that will be used when calling <code>check_frontend</code> . It should contain any parameters that could be needed by <code>check_frontend</code> .
<code>other_nit</code>	Use alternate table IDs for NIT and other tables
<code>timeout_multiply</code>	Improves the timeout for each table reception, by

This is the function that applications should use when doing a transponders scan. It does everything needed to fill the entries with DVB programs (virtual channels) and detect the PIDs associated with them.

A typical usage is to after open a channel file, open a `dmx_fd` and open a frontend. Then, seek for the MPEG tables on all the transponder frequencies with:

```
for (entry = dvb_file->first_entry; entry != NULL; entry = entry->next) {
    struct dvb_v5_descriptors *dvb_scan_handler = NULL;
    dvb_scan_handler = dvb_scan_transponder(parms, entry, dmx_fd,
                                           &check_frontend, args,
                                           args->other_nit,
                                           args->timeout_multiply);

    if (parms->abort) {
        dvb_scan_free_handler_table(dvb_scan_handler);
        break;
    }
}
```

```

if (dvb_scan_handler) {
    dvb_store_channel(&dvb_file_new, parms, dvb_scan_handler,
                    args->get_detected, args->get_nit);
    dvb_scan_free_handler_table(dvb_scan_handler);
}

```

7.3.3.11 dvb_table_filter_free() void dvb_table_filter_free (struct dvb_table_filter * sect)

deallocates all data associated with a table filter

Parameters

sect	table filter pointer
------	----------------------

7.4 Satellite Equipment Control

Files

- file [dvb-sat.h](#)
Provides interfaces to deal with DVB Satellite systems.

Data Structures

- struct [dvb_sat_lnb](#)
Stores the information of a LNBf.

Enumerations

- enum [dvb_sat_polarization](#) {
POLARIZATION_OFF , POLARIZATION_H , POLARIZATION_V , POLARIZATION_L ,
POLARIZATION_R }
Polarization types for Satellite systems.

Functions

- int [dvb_sat_search_lnb](#) (const char *name)
search for a LNBf entry
- int [dvb_print_lnb](#) (int index)
prints the contents of a LNBf entry at STDOUT.
- void [dvb_print_all_lnb](#) (void)
Prints all LNBf entries at STDOUT.
- const struct [dvb_sat_lnb](#) * [dvb_sat_get_lnb](#) (int index)
gets a LNBf entry at its internal database
- const char * [dvb_sat_get_lnb_name](#) (int index)
gets a LNBf entry at its internal database
- int [dvb_sat_set_parms](#) (struct [dvb_v5_fe_parms](#) *parms)
sets the satellite parameters
- int [dvb_sat_real_freq](#) (struct [dvb_v5_fe_parms](#) *p, int freq)
return the real satellite frequency

7.4.1 Detailed Description

7.4.2 Enumeration Type Documentation

7.4.2.1 `dvb_sat_polarization` enum `dvb_sat_polarization`

Polarization types for Satellite systems.

Parameters

<i>POLARIZATION_OFF</i>	Polarization disabled/unused.
<i>POLARIZATION_H</i>	Horizontal polarization
<i>POLARIZATION_V</i>	Vertical polarization
<i>POLARIZATION_L</i>	Left circular polarization (C-band)
<i>POLARIZATION_R</i>	Right circular polarization (C-band)

Enumerator

POLARIZATION_OFF	
POLARIZATION_H	
POLARIZATION_V	
POLARIZATION_L	
POLARIZATION_R	

Definition at line 138 of file `dvb-v5-std.h`.

7.4.3 Function Documentation

7.4.3.1 `dvb_print_all_lnb()` `void dvb_print_all_lnb (void)`

Prints all LNBf entries at STDOUT.

This function doesn't return anything. Internally, it calls `dvb_print_lnb()` for all entries inside its LNBf database.

Examples

`dvbv5-scan.c`, and `dvbv5-zap.c`.

7.4.3.2 `dvb_print_lnb()` `int dvb_print_lnb (int index)`

prints the contents of a LNBf entry at STDOUT.

Parameters

<i>index</i>	index for the entry
--------------	---------------------

Returns

returns -1 if the index is out of range, zero otherwise.

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.4.3.3 dvb_sat_get_lnb() `const struct dvb_sat_lnb * dvb_sat_get_lnb (`
`int index)`

gets a LNBf entry at its internal database

Parameters

<i>index</i>	index for the entry.
--------------	----------------------

Returns

returns NULL if not found, of a struct [dvb_sat_lnb](#) pointer otherwise.

NOTE: none of the strings are i18n translated. In order to get the translated name, you should use [dvb_sat_get_lnb_name\(\)](#)

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.4.3.4 dvb_sat_get_lnb_name() `const char * dvb_sat_get_lnb_name (`
`int index)`

gets a LNBf entry at its internal database

Parameters

<i>index</i>	index for the entry.
--------------	----------------------

Returns

returns NULL if not found, of the name of a LNBf, translated to the user language, if translation is available.

7.4.3.5 dvb_sat_real_freq() `int dvb_sat_real_freq (`
 `struct dvb_v5_fe_parms * p,`
 `int freq)`

return the real satellite frequency

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer.
--------------	---

This function is called internally by the library to get the satellite frequency, considering the LO frequency.

Returns

frequency.

7.4.3.6 dvb_sat_search_lnb() `int dvb_sat_search_lnb (`
 `const char * name)`

search for a LNBf entry

Parameters

<i>name</i>	name of the LNBf entry to seek.
-------------	---------------------------------

On success, it returns a non-negative number which corresponds to the LNBf entry inside the LNBf structure at `dvb-sat.c`.

Returns

A -1 return code indicates that the LNBf was not found.

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.4.3.7 dvb_sat_set_parms() `int dvb_sat_set_parms (`
 `struct dvb_v5_fe_parms * parms)`

sets the satellite parameters

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer.
--------------	---

This function is called internally by the library to set the LNBf parameters, if the `dvb_v5_fe_parms::lnb` field is filled.

Returns

0 on success.

7.5 Ancillary functions and macros

Files

- file [dvb-log.h](#)
Provides interfaces to handle libdvbv5 log messages.
- file [crc32.h](#)
Provides ancillary code to calculate DVB crc32 checksum.
- file [countries.h](#)
Provides ancillary code to convert ISO 3166-1 country codes.

Macros

- `#define ARRAY_SIZE(x)`
Calculates the number of elements of an array.

Typedefs

- `typedef void(* dvb_logfunc) (int level, const char *fmt,...)`
typedef used by `dvb_fe_open2` for the log function

Enumerations

- `enum dvb_country_t {
 COUNTRY_UNKNOWN , AD , AE , AF ,
 AG , AI , AL , AM ,
 AO , AQ , AR , AS ,
 AT , AU , AW , AX ,
 AZ , BA , BB , BD ,
 BE , BF , BG , BH ,
 BI , BJ , BL , BM ,
 BN , BO , BQ , BR ,
 BS , BT , BV , BW ,
 BY , BZ , CA , CC ,
 CD , CF , CG , CH ,
 CI , CK , CL , CM ,
 CN , CO , CR , CU ,
 CV , CW , CX , CY ,
 CZ , DE , DJ , DK ,
 DM , DO , DZ , EC ,
 EE , EG , EH , ER ,
 ES , ET , FI , FJ ,
 FK , FM , FO , FR ,
 GA , GB , GD , GE ,
 GF , GG , GH , GI ,
 GL , GM , GN , GP ,
 GQ , GR , GS , GT ,
 GU , GW , GY , HK ,`


```
HM, HN, HR, HT,
HU, ID, IE, IL,
IM, IN, IO, IQ,
IR, IS, IT, JE,
JM, JO, JP, KE,
KG, KH, KI, KM,
KN, KP, KR, KW,
KY, KZ, LA, LB,
LC, LI, LK, LR,
LS, LT, LU, LV,
LY, MA, MC, MD,
ME, MF, MG, MH,
MK, ML, MM, MN,
MO, MP, MQ, MR,
MS, MT, MU, MV,
MW, MX, MY, MZ,
NA, NC, NE, NF,
NG, NI, NL, NO,
NP, NR, NU, NZ,
OM, PA, PE, PF,
PG, PH, PK, PL,
PM, PN, PR, PS,
PT, PW, PY, QA,
RE, RO, RS, RU,
RW, SA, SB, SC,
SD, SE, SG, SH,
SI, SJ, SK, SL,
SM, SN, SO, SR,
SS, ST, SV, SX,
SY, SZ, TC, TD,
TF, TG, TH, TJ,
TK, TL, TM, TN,
TO, TR, TT, TV,
TW, TZ, UA, UG,
UM, US, UY, UZ,
VA, VC, VE, VG,
VI, VN, VU, WF,
WS, YE, YT, ZA,
ZM, ZW }
```

ISO-3166-1 alpha-2 country code.

Functions

- void [dvb_default_log](#) (int level, const char *fmt,...)
This is the prototype of the internal log function that it is used, if the library client doesn't desire to override with something else.
- void [atsc_time](#) (const uint32_t start_time, struct tm *tm)
Converts an ATSC EIT formatted timestamp into struct tm.
- uint32_t [dvb_crc32](#) (uint8_t *data, size_t datalen, uint32_t crc)
Calculates the crc-32 as defined at the MPEG-TS specs.
- enum [dvb_country_t](#) [dvb_country_a2_to_id](#) (const char *name)
Converts an Unix-like 2-letter Country code into enum dvb_country_t.
- enum [dvb_country_t](#) [dvb_country_a3_to_id](#) (const char *name)
Converts a 3-letter Country code as used by MPEG-TS tables into enum dvb_country_t.
- const char * [dvb_country_to_2letters](#) (int id)

Converts an enum `dvb_country_t` into Unix-like 2-letter Country code.

- `const char * dvb_country_to_3letters` (int id)

Converts an enum `dvb_country_t` into a 3-letter Country code as used by MPEG-TS tables.

- `const char * dvb_country_to_name` (int id)

Converts an enum `dvb_country_t` into a Country name as used by MPEG-TS tables.

- `enum dvb_country_t dvb_guess_user_country` (void)

Guess the country code from the Unix environment variables.

7.5.1 Detailed Description

7.5.2 Macro Definition Documentation

7.5.2.1 `ARRAY_SIZE` `#define ARRAY_SIZE(` `x)`

Calculates the number of elements of an array.

Examples

`dvb-fe-tool.c`.

Definition at line 64 of file `dvb-fe.h`.

7.5.3 Typedef Documentation

7.5.3.1 `dvb_logfunc` `void(* dvb_logfunc)(void *logpriv, int level, const char *fmt,...)`

typedef used by `dvb_fe_open2` for the log function

typedef used by `dvb_fe_open2` for the log function with private context

Definition at line 44 of file `dvb-log.h`.

7.5.4 Enumeration Type Documentation

7.5.4.1 `dvb_country_t` `enum dvb_country_t`

ISO-3166-1 alpha-2 country code.

Enumerator

COUNTRY_UNKNOWN	(Unknown Country)
AD	Andorra.
AE	United Arab Emirates.
AF	Afghanistan.
AG	Antigua and Barbuda.
AI	Anguilla.
AL	Albania.
AM	Armenia.
AO	Angola.
AQ	Antarctica.
AR	Argentina.
AS	American Samoa.
AT	Austria.
AU	Australia.
AW	Aruba.
AX	Aland Islands.
AZ	Azerbaijan.
BA	Bosnia and Herzegovina.
BB	Barbados.
BD	Bangladesh.
BE	Belgium.
BF	Burkina Faso.
BG	Bulgaria.
BH	Bahrain.
BI	Burundi.
BJ	Benin.
BL	Saint Barthelemy.
BM	Bermuda.
BN	Brunei Darussalam.
BO	Plurinational State of Bolivia.
BQ	Bonaire, Saint Eustatius and Saba.
BR	Brazil.
BS	Bahamas.
BT	Bhutan.
BV	Bouvet Island.
BW	Botswana.
BY	Belarus.
BZ	Belize.
CA	Canada.
CC	Cocos (Keeling) Islands.
CD	The Democratic Republic of the Congo.
CF	Central African Republic.
CG	Congo.
CH	Switzerland.
CI	Cote d'Ivoire.
CK	Cook Islands.
CL	Chile.
CM	Cameroon.
CN	China.

Enumerator

CO	Colombia.
CR	Costa Rica.
CU	Cuba.
CV	Cape Verde.
CW	Curacao.
CX	Christmas Island.
CY	Cyprus.
CZ	Czech Republic.
DE	Germany.
DJ	Djibouti.
DK	Denmark.
DM	Dominica.
DO	Dominican Republic.
DZ	Algeria.
EC	Ecuador.
EE	Estonia.
EG	Egypt.
EH	Western Sahara.
ER	Eritrea.
ES	Spain.
ET	Ethiopia.
FI	Finland.
FJ	Fiji.
FK	Falkland Islands (Malvinas)
FM	Federated States of Micronesia.
FO	Faroe Islands.
FR	France.
GA	Gabon.
GB	United Kingdom.
GD	Grenada.
GE	Georgia.
GF	French Guiana.
GG	Guernsey.
GH	Ghana.
GI	Gibraltar.
GL	Greenland.
GM	Gambia.
GN	Guinea.
GP	Guadeloupe.
GQ	Equatorial Guinea.
GR	Greece.
GS	South Georgia and the South Sandwich Islands.
GT	Guatemala.
GU	Guam.
GW	Guinea-Bissau.
GY	Guyana.
HK	Hong Kong.
HM	Heard Island and McDonald Islands.
HN	Honduras.

Enumerator

HR	Croatia.
HT	Haiti.
HU	Hungary.
ID	Indonesia.
IE	Ireland.
IL	Israel.
IM	Isle of Man.
IN	India.
IO	British Indian Ocean Territory.
IQ	Iraq.
IR	Islamic Republic of Iran.
IS	Iceland.
IT	Italy.
JE	Jersey.
JM	Jamaica.
JO	Jordan.
JP	Japan.
KE	Kenya.
KG	Kyrgyzstan.
KH	Cambodia.
KI	Kiribati.
KM	Comoros.
KN	Saint Kitts and Nevis.
KP	Democratic People's Republic of Korea.
KR	Republic of Korea.
KW	Kuwait.
KY	Cayman Islands.
KZ	Kazakhstan.
LA	Lao People's Democratic Republic.
LB	Lebanon.
LC	Saint Lucia.
LI	Liechtenstein.
LK	Sri Lanka.
LR	Liberia.
LS	Lesotho.
LT	Lithuania.
LU	Luxembourg.
LV	Latvia.
LY	Libyan Arab Jamahiriya.
MA	Morocco.
MC	Monaco.
MD	Republic of Moldova.
ME	Montenegro.
MF	Saint Martin (French part)
MG	Madagascar.
MH	Marshall Islands.
MK	The Former Yugoslav Republic of Macedonia.
ML	Mali.
MM	Myanmar.

Enumerator

MN	Mongolia.
MO	Macao.
MP	Northern Mariana Islands.
MQ	Martinique.
MR	Mauritania.
MS	Montserrat.
MT	Malta.
MU	Mauritius.
MV	Maldives.
MW	Malawi.
MX	Mexico.
MY	Malaysia.
MZ	Mozambique.
NA	Namibia.
NC	New Caledonia.
NE	Niger.
NF	Norfolk Island.
NG	Nigeria.
NI	Nicaragua.
NL	Netherlands.
NO	Norway.
NP	Nepal.
NR	Nauru.
NU	Niue.
NZ	New Zealand.
OM	Oman.
PA	Panama.
PE	Peru.
PF	French Polynesia.
PG	Papua New Guinea.
PH	Philippines.
PK	Pakistan.
PL	Poland.
PM	Saint Pierre and Miquelon.
PN	Pitcairn.
PR	Puerto Rico.
PS	Occupied Palestinian Territory.
PT	Portugal.
PW	Palau.
PY	Paraguay.
QA	Qatar.
RE	Reunion.
RO	Romania.
RS	Serbia.
RU	Russian Federation.
RW	Rwanda.
SA	Saudi Arabia.
SB	Solomon Islands.
SC	Seychelles.

Enumerator

SD	Sudan.
SE	Sweden.
SG	Singapore.
SH	Saint Helena, Ascension and Tristan da Cunha.
SI	Slovenia.
SJ	Svalbard and Jan Mayen.
SK	Slovakia.
SL	Sierra Leone.
SM	San Marino.
SN	Senegal.
SO	Somalia.
SR	Suriname.
SS	South Sudan.
ST	Sao Tome and Principe.
SV	El Salvador.
SX	Sint Maarten (Dutch part)
SY	Syrian Arab Republic.
SZ	Swaziland.
TC	Turks and Caicos Islands.
TD	Chad.
TF	French Southern Territories.
TG	Togo.
TH	Thailand.
TJ	Tajikistan.
TK	Tokelau.
TL	Timor-Leste.
TM	Turkmenistan.
TN	Tunisia.
TO	Tonga.
TR	Turkey.
TT	Trinidad and Tobago.
TV	Tuvalu.
TW	Taiwan, Province of China.
TZ	United Republic of Tanzania.
UA	Ukraine.
UG	Uganda.
UM	United States Minor Outlying Islands.
US	United States.
UY	Uruguay.
UZ	Uzbekistan.
VA	Holy See (Vatican City State)
VC	Saint Vincent and The Grenadines.
VE	Bolivarian Republic of Venezuela.
VG	British Virgin Islands.
VI	U.S. Virgin Islands
VN	Viet Nam.
VU	Vanuatu.
WF	Wallis and Futuna.
WS	Samoa.

Enumerator

YE	Yemen.
YT	Mayotte.
ZA	South Africa.
ZM	Zambia.
ZW	Zimbabwe.

Definition at line 544 of file [countries.h](#).

7.5.5 Function Documentation

7.5.5.1 atsc_time() `void atsc_time (`
`const uint32_t start_time,`
`struct tm * tm)`

Converts an ATSC EIT formatted timestamp into struct tm.

Parameters

<i>start_time</i>	event on ATSC EIT time format
<i>tm</i>	pointer to struct tm where the converted timestamp will be stored.

7.5.5.2 dvb_country_a2_to_id() `enum dvb_country_t dvb_country_a2_to_id (`
`const char * name)`

Converts an Unix-like 2-letter Country code into enum `dvb_country_t`.

Parameters

<i>name</i>	two-letter Country code.
-------------	--------------------------

Returns

It returns the corresponding enum `dvb_country_t` ID. If not found, returns `COUNTRY_UNKNOWN`.

7.5.5.3 dvb_country_a3_to_id() `enum dvb_country_t dvb_country_a3_to_id (`
`const char * name)`

Converts a 3-letter Country code as used by MPEG-TS tables into enum `dvb_country_t`.

Parameters

<i>name</i>	three-letter Country code.
-------------	----------------------------

Returns

It returns the corresponding enum `dvb_country_t` ID. If not found, returns `COUNTRY_UNKNOWN`.

7.5.5.4 `dvb_country_to_2letters()` `const char * dvb_country_to_2letters (`
`int id)`

Converts an enum `dvb_country_t` into Unix-like 2-letter Country code.

Parameters

<i>id</i>	enum <code>dvb_country_t</code> ID.
-----------	-------------------------------------

Returns

It returns the 2-letter country code string that corresponds to the Country. If not found, returns `NULL`.

7.5.5.5 `dvb_country_to_3letters()` `const char * dvb_country_to_3letters (`
`int id)`

Converts an enum `dvb_country_t` into a 3-letter Country code as used by MPEG-TS tables.

Parameters

<i>id</i>	enum <code>dvb_country_t</code> ID.
-----------	-------------------------------------

Returns

It returns the 3-letter country code string that corresponds to the Country. If not found, returns `NULL`.

7.5.5.6 `dvb_country_to_name()` `const char * dvb_country_to_name (`
`int id)`

Converts an enum `dvb_country_t` into a Country name as used by MPEG-TS tables.

Parameters

<i>id</i>	enum <code>dvb_country_t</code> ID.
-----------	-------------------------------------

Returns

It returns a string with the Country name that corresponds to the country. If not found, returns NULL.

7.5.5.7 dvb_crc32() `uint32_t dvb_crc32 (`
 `uint8_t * data,`
 `size_t datalen,`
 `uint32_t crc)`

Calculates the crc-32 as defined at the MPEG-TS specs.

Parameters

<i>data</i>	Pointer to the buffer to be checked
<i>datalen</i>	Length of the buffer
<i>crc</i>	Initial value for the crc checksum. To calculate the checksum of the entire packet at once, use 0xFFFFFFFF

7.5.5.8 dvb_default_log() `void dvb_default_log (`
 `int level,`
 `const char * fmt,`
 `...)`

This is the prototype of the internal log function that it is used, if the library client doesn't desire to override with something else.

Parameters

<i>level</i>	level of the message, as defined at syslog.h
<i>fmt</i>	format string (same as format string on sprintf)

7.5.5.9 dvb_guess_user_country() `enum dvb_country_t dvb_guess_user_country (`
 `void)`

Guess the country code from the Unix environment variables.

Returns

It returns the corresponding enum `dvb_country_t` ID. If not found, returns `COUNTRY_UNKNOWN`.

7.6 Digital TV table parsing

Files

- file [descriptors.h](#)

- *Provides a way to handle MPEG-TS descriptors found on Digital TV streams.*
- file [header.h](#)
Provides the MPEG TS table headers.
- file [atsc_header.h](#)
Provides some common ATSC stuff.
- file [atsc_eit.h](#)
Provides the table parser for the ATSC EIT (Event Information Table)
- file [cat.h](#)
Provides the table parser for the CAT (Conditional Access Table)
- file [eit.h](#)
Provides the table parser for the DVB EIT (Event Information Table)
- file [mgt.h](#)
Provides the table parser for the ATSC MGT (Master Guide Table)
- file [nit.h](#)
Provides the descriptors for NIT MPEG-TS table.
- file [pat.h](#)
Provides the descriptors for PAT MPEG-TS table.
- file [pmt.h](#)
Provides the descriptors for PMT MPEG-TS table.
- file [sdt.h](#)
Provides the descriptors for SDT MPEG-TS table.
- file [vct.h](#)
Provides the descriptors for TVCT and CVCT tables.
- file [mpeg_es.h](#)
Provides the table parser for the MPEG-TS Elementary Stream.
- file [mpeg_pes.h](#)
Provides the table parser for the MPEG-PES Elementary Stream.
- file [mpeg_ts.h](#)
Provides the table parser for the MPEG-PES Elementary Stream.

Data Structures

- struct [dvb_desc](#)
Linked list containing the several descriptors found on a MPEG-TS table.
- struct [dvb_descriptor](#)
Contains the parser information for the MPEG-TS parser code.
- struct [dvb_ts_packet_header](#)
Header of a MPEG-TS transport packet.
- struct [dvb_table_header](#)
Header of a MPEG-TS table.
- struct [atsc_table_eit_event](#)
ATSC EIT event table.
- union [atsc_table_eit_desc_length](#)
ATSC EIT descriptor length.
- struct [atsc_table_eit](#)
ATSC EIT table.
- struct [dvb_table_eit_event](#)
DVB EIT event table.
- struct [dvb_table_eit](#)
DVB EIT table.

- struct [atsc_table_mgt_table](#)
ATSC tables description at MGT table.
- struct [atsc_table_mgt](#)
ATSC MGT table.
- union [dvb_table_nit_transport_header](#)
MPEG-TS NIT transport header.
- struct [dvb_table_nit_transport](#)
MPEG-TS NIT transport table.
- struct [dvb_table_nit](#)
MPEG-TS NIT table.
- struct [dvb_table_pat_program](#)
MPEG-TS PAT program table.
- struct [dvb_table_pat](#)
MPEG-TS PAT table.
- struct [dvb_table_pmt_stream](#)
MPEG-TS PMT stream table.
- struct [dvb_table_pmt](#)
MPEG-TS PMT table.
- struct [dvb_table_sdt_service](#)
MPEG-TS SDT service table.
- struct [dvb_table_sdt](#)
MPEG-TS SDT table.
- struct [atsc_table_vct_channel](#)
ATSC VCT channel table (covers both CVCT and TVCT)
- struct [atsc_table_vct](#)
ATSC VCT table (covers both CVCT and TVCT)
- union [atsc_table_vct_descriptor_length](#)
ATSC VCT descriptor length.
- struct [dvb_mpeg_es_seq_start](#)
MPEG ES Sequence header.
- struct [dvb_mpeg_es_pic_start](#)
MPEG ES Picture start header.
- struct [ts_t](#)
MPEG PES timestamp structure, used for dts and pts.
- struct [dvb_mpeg_pes_optional](#)
MPEG PES optional header.
- struct [dvb_mpeg_pes](#)
MPEG PES data structure.
- struct [dvb_mpeg_ts_adaption](#)
MPEG TS header adaption field.
- struct [dvb_mpeg_ts](#)
MPEG TS header.

Macros

- #define [DVB_MAX_PAYLOAD_PACKET_SIZE](#)
Maximum size of a table session to be parsed.
- #define [DVB_CRC_SIZE](#)
number of bytes for the descriptor's CRC check
- #define [ATSC_BASE_PID](#)

- ATSC PID for the Program and System Information Protocol.*

 - #define [ATSC_TABLE_EIT](#)
ATSC EIT table ID.
 - #define [atsc_eit_event_foreach](#)(_event, _eit)
Macro used to find event on an ATSC EIT table.
 - #define [DVB_TABLE_CAT](#)
ATSC CAT table ID.
 - #define [DVB_TABLE_CAT_PID](#)
ATSC PID table ID.
 - #define [DVB_TABLE_EIT](#)
DVB EIT table ID for the actual TS.
 - #define [DVB_TABLE_EIT_OTHER](#)
DVB EIT table ID for other TS.
 - #define [DVB_TABLE_EIT_PID](#)
DVB EIT Program ID.
 - #define [DVB_TABLE_EIT_SCHEDULE](#)
Start table ID for the DVB EIT schedule data on the actual TS The range has 0x0f elements (0x50 to 0x5F).
 - #define [DVB_TABLE_EIT_SCHEDULE_OTHER](#)
Start table ID for the DVB EIT schedule data on other TS The range has 0x0f elements (0x60 to 0x6F).
 - #define [dvb_eit_event_foreach](#)(_event, _eit)
Macro used to find event on a DVB EIT table.
 - #define [ATSC_TABLE_MGT](#)
ATSC MGT table ID.
 - #define [DVB_TABLE_NIT](#)
NIT table ID.
 - #define [DVB_TABLE_NIT2](#)
NIT table ID (alternative table ID)
 - #define [DVB_TABLE_NIT_PID](#)
NIT Program ID.
 - #define [dvb_nit_transport_foreach](#)(_tran, _nit)
Macro used to find a transport inside a NIT table.
 - #define [DVB_TABLE_PAT](#)
PAT table ID.
 - #define [DVB_TABLE_PAT_PID](#)
PAT Program ID.
 - #define [dvb_pat_program_foreach](#)(_pgm, _pat)
Macro used to find programs on a PAT table.
 - #define [DVB_TABLE_PMT](#)
PMT table ID.
 - #define [dvb_pmt_stream_foreach](#)(_stream, _pmt)
Macro used to find streams on a PMT table.
 - #define [DVB_TABLE_SDT](#)
SDT table ID.
 - #define [DVB_TABLE_SDT2](#)
SDT table ID (alternative table ID)
 - #define [DVB_TABLE_SDT_PID](#)
SDT Program ID.
 - #define [dvb_sdt_service_foreach](#)(_service, _sdt)
Macro used to find services on a SDT table.
 - #define [ATSC_TABLE_TVCT](#)
TVCT table ID.

- `#define ATSC_TABLE_CVCT`
CVCT table ID.
- `#define ATSC_TABLE_VCT_PID`
Program ID with the VCT tables on it.
- `#define atsc_vct_channel_foreach(_channel, _vct)`
Macro used to find channels on a VCT table.
- `#define DVB_MPEG_ES_PIC_START`
Picture Start.
- `#define DVB_MPEG_ES_USER_DATA`
User Data.
- `#define DVB_MPEG_ES_SEQ_START`
Sequence Start.
- `#define DVB_MPEG_ES_SEQ_EXT`
Extension.
- `#define DVB_MPEG_ES_GOP`
Group Of Pictures.
- `#define DVB_MPEG_ES_SLICES`
Slices.
- `#define DVB_MPEG_PES`
MPEG Packetized Elementary Stream magic.
- `#define DVB_MPEG_PES_AUDIO`
PES Audio.
- `#define DVB_MPEG_PES_VIDEO`
PES Video.
- `#define DVB_MPEG_STREAM_MAP`
PES Stream map.
- `#define DVB_MPEG_STREAM_PADDING`
PES padding.
- `#define DVB_MPEG_STREAM_PRIVATE_2`
PES private.
- `#define DVB_MPEG_STREAM_ECM`
PES ECM Stream.
- `#define DVB_MPEG_STREAM_EMM`
PES EMM Stream.
- `#define DVB_MPEG_STREAM_DIRECTORY`
PES Stream directory.
- `#define DVB_MPEG_STREAM_DSMCC`
PES DSMCC.
- `#define DVB_MPEG_STREAM_H222E`
PES H.222.1 type E.
- `#define DVB_MPEG_TS`
MPEG Transport Stream magic.
- `#define DVB_MPEG_TS_PACKET_SIZE`
Size of an MPEG packet.

Typedefs

- typedef void(* [dvb_table_init_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, void **table)
Function prototype for a function that initializes the descriptors parsing on a table.
- typedef int(* [dvb_desc_init_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Function prototype for the descriptors parsing init code.
- typedef void(* [dvb_desc_print_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Function prototype for the descriptors parsing print code.
- typedef void(* [dvb_desc_free_func](#)) (struct [dvb_desc](#) *desc)
Function prototype for the descriptors memory free code.
- typedef void [nit_handler_callback_t](#)(struct [dvb_table_nit](#) *nit, struct [dvb_desc](#) *desc, void *priv)
typedef for a callback used when a NIT table entry is found
- typedef void [nit_tran_handler_callback_t](#)(struct [dvb_table_nit](#) *nit, struct [dvb_table_nit_transport](#) *tran, struct [dvb_desc](#) *desc, void *priv)
typedef for a callback used when a NIT transport table entry is found
- typedef int(* [dvb_desc_ext_init_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_extension_descriptor](#) *ext, void *desc)
Function prototype for the extended descriptors parsing init code.
- typedef void(* [dvb_desc_ext_print_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_extension_descriptor](#) *ext, const void *desc)
Function prototype for the extended descriptors parsing print code.
- typedef void(* [dvb_desc_ext_free_func](#)) (const void *desc)
Function prototype for the extended descriptors parsing free code.

Enumerations

- enum [descriptors](#) {
[video_stream_descriptor](#) , [audio_stream_descriptor](#) , [hierarchy_descriptor](#) , [registration_descriptor](#) ,
[ds_alignment_descriptor](#) , [target_background_grid_descriptor](#) , [video_window_descriptor](#) , [conditional_access_descriptor](#)
, [iso639_language_descriptor](#) , [system_clock_descriptor](#) , [multiplex_buffer_utilization_descriptor](#) , [copyright_descriptor](#)
, [maximum_bitrate_descriptor](#) , [private_data_indicator_descriptor](#) , [smoothing_buffer_descriptor](#) , [std_descriptor](#)
, [ibp_descriptor](#) , [mpeg4_video_descriptor](#) , [mpeg4_audio_descriptor](#) , [iod_descriptor](#) ,
[sl_descriptor](#) , [fmc_descriptor](#) , [external_es_id_descriptor](#) , [muxcode_descriptor](#) ,
[fmxbuffer_size_descriptor](#) , [multiplexbuffer_descriptor](#) , [content_labeling_descriptor](#) , [metadata_pointer_descriptor](#)
, [metadata_descriptor](#) , [metadata_std_descriptor](#) , [AVC_video_descriptor](#) , [ipmp_descriptor](#) ,
[AVC_timing_and_HRD_descriptor](#) , [mpeg2_aac_audio_descriptor](#) , [flexmux_timing_descriptor](#) , [network_name_descriptor](#)
, [service_list_descriptor](#) , [stuffing_descriptor](#) , [satellite_delivery_system_descriptor](#) , [cable_delivery_system_descriptor](#)
, [VBI_data_descriptor](#) , [VBI_teletext_descriptor](#) , [bouquet_name_descriptor](#) , [service_descriptor](#) ,
[country_availability_descriptor](#) , [linkage_descriptor](#) , [NVOD_reference_descriptor](#) , [time_shifted_service_descriptor](#)
, [short_event_descriptor](#) , [extended_event_descriptor](#) , [time_shifted_event_descriptor](#) , [component_descriptor](#)
, [mosaic_descriptor](#) , [stream_identifier_descriptor](#) , [CA_identifier_descriptor](#) , [content_descriptor](#) ,
[parental_rating_descriptor](#) , [teletext_descriptor](#) , [telephone_descriptor](#) , [local_time_offset_descriptor](#) ,
[subtitling_descriptor](#) , [terrestrial_delivery_system_descriptor](#) , [multilingual_network_name_descriptor](#) ,
[multilingual_bouquet_name_descriptor](#) ,

```

multilingual_service_name_descriptor , multilingual_component_descriptor , private_data_specifier_descriptor
, service_move_descriptor ,
short_smoothing_buffer_descriptor , frequency_list_descriptor , partial_transport_stream_descriptor ,
data_broadcast_descriptor ,
scrambling_descriptor , data_broadcast_id_descriptor , transport_stream_descriptor , DSNG_descriptor ,
PDC_descriptor , AC_3_descriptor , ancillary_data_descriptor , cell_list_descriptor ,
cell_frequency_link_descriptor , announcement_support_descriptor , application_signalling_descriptor ,
adaptation_field_data_descriptor ,
service_identifier_descriptor , service_availability_descriptor , default_authority_descriptor , related_content_descriptor
,
TVA_id_descriptor , content_identifier_descriptor , time_slice_fec_identifier_descriptor , ECM_repetition_rate_descriptor
,
S2_satellite_delivery_system_descriptor , enhanced_AC_3_descriptor , DTS_descriptor , AAC_descriptor ,
XAIT_location_descriptor , FTA_content_management_descriptor , extension_descriptor , CUE_identifier_descriptor
,
extended_channel_name , service_location , component_name_descriptor , logical_channel_number_descriptor
,
carousel_id_descriptor , association_tag_descriptor , deferred_association_tags_descriptor , hierarchical_transmission_descrip
,
digital_copy_control_descriptor , network_identifier_descriptor , partial_transport_stream_time_descriptor ,
audio_component_descriptor ,
hyperlink_descriptor , target_area_descriptor , data_contents_descriptor , video_decode_control_descriptor ,
download_content_descriptor , CA_EMM_TS_descriptor , CA_contract_information_descriptor , CA_service_descriptor
,
TS_Information_descriptor , extended_broadcaster_descriptor , logo_transmission_descriptor , basic_local_event_descriptor
,
reference_descriptor , node_relation_descriptor , short_node_information_descriptor , STC_reference_descriptor
,
series_descriptor , event_group_descriptor , SI_parameter_descriptor , broadcaster_Name_Descriptor ,
component_group_descriptor , SI_prime_TS_descriptor , board_information_descriptor , LDT_linkage_descriptor
,
connected_transmission_descriptor , content_availability_descriptor , service_group_descriptor , carousel_compatible_compos
,
conditional_playback_descriptor , ISDBT_delivery_system_descriptor , partial_reception_descriptor ,
emergency_information_descriptor ,
data_component_descriptor , system_management_descriptor , atsc_stuffing_descriptor , atsc_ac3_audio_descriptor
,
atsc_caption_service_descriptor , atsc_content_advisory_descriptor , atsc_extended_channel_descriptor ,
atsc_service_location_descriptor ,
atsc_time_shifted_service_descriptor , atsc_component_name_descriptor , atsc_DCC_departing_request_descriptor
, atsc_DCC_arriving_request_descriptor ,
atsc_redistribution_control_descriptor , atsc_ATSC_private_information_descriptor , atsc_genre_descriptor }

```

List containing all descriptors used by Digital TV MPEG-TS.

- enum `dvb_streams` {


```

stream_video , stream_video_h262 , stream_audio , stream_audio_13818_3 ,
stream_private_sections , stream_private_data , stream_mheg , stream_h222 ,
stream_h222_1 , stream_13818_6_A , stream_13818_6_B , stream_13818_6_C ,
stream_13818_6_D , stream_h222_aux , stream_audio_adts , stream_video_14496_2 ,
stream_audio_latm , stream_14496_1_pes , stream_14496_1_iso , stream_download ,
stream_video_h264 , stream_audio_14496_3 , stream_video_hevc , stream_video_cavs ,
stream_video_moto , stream_audio_a52 , stream_scte_27 , stream_audio_sdds ,
stream_audio_dts_hdmv , stream_audio_e_ac3 , stream_audio_dts , stream_audio_a52_vls ,
stream_spu_vls , stream_audio_sdds2 }

```

Add support for MPEG-TS Stream types.

- enum `dvb_mpeg_es_frame_t` {


```

DVB_MPEG_ES_FRAME_UNKNOWN , DVB_MPEG_ES_FRAME_I , DVB_MPEG_ES_FRAME_P ,
DVB_MPEG_ES_FRAME_B ,
DVB_MPEG_ES_FRAME_D }

```


MPEG frame types.

Functions

- `uint32_t dvb_bcd (uint32_t bcd)`
Converts from BCD to CPU integer internal representation.
- `void dvb_hexdump (struct dvb_v5_fe_parms *parms, const char *prefix, const unsigned char *buf, int len)`
dumps data into the logs in hexadecimal format
- `int dvb_desc_parse (struct dvb_v5_fe_parms *parms, const uint8_t *buf, uint16_t buflen, struct dvb_desc **head_desc)`
parse MPEG-TS descriptors
- `void dvb_desc_free (struct dvb_desc **list)`
frees a dvb_desc linked list
- `void dvb_desc_print (struct dvb_v5_fe_parms *parms, struct dvb_desc *desc)`
prints the contents of a struct dvb_desc linked list
- `void dvb_table_header_init (struct dvb_table_header *header)`
Initializes and parses MPEG-TS table header.
- `void dvb_table_header_print (struct dvb_v5_fe_parms *parms, const struct dvb_table_header *header)`
Prints the content of the MPEG-TS table header.
- `ssize_t atsc_table_eit_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf, ssize_t buflen, struct atsc_table_eit **table)`
Initializes and parses ATSC EIT table.
- `void atsc_table_eit_free (struct atsc_table_eit *table)`
Frees all data allocated by the ATSC EIT table parser.
- `void atsc_table_eit_print (struct dvb_v5_fe_parms *parms, struct atsc_table_eit *table)`
Prints the content of the ATSC EIT table.
- `ssize_t dvb_table_eit_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf, ssize_t buflen, struct dvb_table_eit **table)`
Initializes and parses EIT table.
- `void dvb_table_eit_free (struct dvb_table_eit *table)`
Frees all data allocated by the DVB EIT table parser.
- `void dvb_table_eit_print (struct dvb_v5_fe_parms *parms, struct dvb_table_eit *table)`
Prints the content of the DVB EIT table.
- `void dvb_time (const uint8_t data[5], struct tm *tm)`
Converts a DVB EIT formatted timestamp into struct tm.
- `ssize_t atsc_table_mgt_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf, ssize_t buflen, struct atsc_table_mgt **table)`
Initializes and parses MGT table.
- `void atsc_table_mgt_free (struct atsc_table_mgt *table)`
Frees all data allocated by the MGT table parser.
- `void atsc_table_mgt_print (struct dvb_v5_fe_parms *parms, struct atsc_table_mgt *table)`
Prints the content of the MGT table.
- `ssize_t dvb_table_nit_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf, ssize_t buflen, struct dvb_table_nit **table)`
Initializes and parses NIT table.
- `void dvb_table_nit_free (struct dvb_table_nit *table)`
Frees all data allocated by the NIT table parser.
- `void dvb_table_nit_print (struct dvb_v5_fe_parms *parms, struct dvb_table_nit *table)`
Prints the content of the NIT table.
- `void dvb_table_nit_descriptor_handler (struct dvb_v5_fe_parms *parms, struct dvb_table_nit *table, enum descriptors descriptor, nit_handler_callback_t *call_nit, nit_tran_handler_callback_t *call_tran, void *priv)`

- For each entry at NIT and NIT transport tables, call a callback.*
- `ssize_t dvb_table_pat_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, struct `dvb_table_pat` **table)
Initializes and parses PAT table.
 - void `dvb_table_pat_free` (struct `dvb_table_pat` *table)
Frees all data allocated by the PAT table parser.
 - void `dvb_table_pat_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_table_pat` *table)
Prints the content of the PAT table.
 - `ssize_t dvb_table_pmt_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, struct `dvb_table_pmt` **table)
Initializes and parses PMT table.
 - void `dvb_table_pmt_free` (struct `dvb_table_pmt` *table)
Frees all data allocated by the PMT table parser.
 - void `dvb_table_pmt_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_table_pmt` *table)
Prints the content of the PAT table.
 - `ssize_t dvb_table_sdt_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, struct `dvb_table_sdt` **table)
Initializes and parses SDT table.
 - void `dvb_table_sdt_free` (struct `dvb_table_sdt` *table)
Frees all data allocated by the SDT table parser.
 - void `dvb_table_sdt_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_table_sdt` *table)
Prints the content of the SDT table.
 - `ssize_t atsc_table_vct_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, struct `atsc_table_vct` **table)
Initializes and parses VCT table.
 - void `atsc_table_vct_free` (struct `atsc_table_vct` *table)
Frees all data allocated by the VCT table parser.
 - void `atsc_table_vct_print` (struct `dvb_v5_fe_parms` *parms, struct `atsc_table_vct` *table)
Prints the content of the VCT table.
 - int `dvb_mpeg_es_seq_start_init` (const uint8_t *buf, ssize_t buflen, struct `dvb_mpeg_es_seq_start` *seq_start)
Initialize a struct `dvb_mpeg_es_seq_start` from buffer.
 - void `dvb_mpeg_es_seq_start_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_mpeg_es_seq_start` *seq_start)
Print details of struct `dvb_mpeg_es_seq_start`.
 - int `dvb_mpeg_es_pic_start_init` (const uint8_t *buf, ssize_t buflen, struct `dvb_mpeg_es_pic_start` *pic_start)
Initialize a struct `dvb_mpeg_es_pic_start` from buffer.
 - void `dvb_mpeg_es_pic_start_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_mpeg_es_pic_start` *pic_start)
Print details of struct `dvb_mpeg_es_pic_start`.
 - `ssize_t dvb_mpeg_pes_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, uint8_t *table)
Initialize a struct `dvb_mpeg_pes` from buffer.
 - void `dvb_mpeg_pes_free` (struct `dvb_mpeg_pes` *pes)
Deallocate memory associated with a struct `dvb_mpeg_pes`.
 - void `dvb_mpeg_pes_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_mpeg_pes` *pes)
Print details of struct `dvb_mpeg_pes`.
 - `ssize_t dvb_mpeg_ts_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, uint8_t *table, ssize_t *table_length)
Initialize a struct `dvb_mpeg_ts` from buffer.
 - void `dvb_mpeg_ts_free` (struct `dvb_mpeg_ts` *ts)
Deallocate memory associated with a struct `dvb_mpeg_ts`.
 - void `dvb_mpeg_ts_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_mpeg_ts` *ts)
Print details of struct `dvb_mpeg_ts`.

Variables

- const [dvb_table_init_func](#) [dvb_table_initializers](#) [256]
Table with all possible descriptors.
- const struct [dvb_descriptor](#) [dvb_descriptors](#) []
Contains the parsers for the several descriptors.
- const char * [pmt_stream_name](#) []
Converts from enum [dvb_streams](#) into a string.
- const char * [dvb_mpeg_es_frame_names](#) [5]
Vector that translates from enum [dvb_mpeg_es_frame_t](#) to string.

7.6.1 Detailed Description

7.6.2 Macro Definition Documentation

7.6.2.1 ATSC_BASE_PID `#define ATSC_BASE_PID`

ATSC PID for the Program and System Information Protocol.

Definition at line 44 of file [atsc_header.h](#).

7.6.2.2 `atsc_eit_event_foreach` `#define atsc_eit_event_foreach(` `_event,` `_eit)`

Macro used to find event on an ATSC EIT table.

Parameters

<code>_event</code>	event to seek
<code>_eit</code>	pointer to struct atsc_table_eit_event

Definition at line 160 of file [atsc_eit.h](#).

7.6.2.3 ATSC_TABLE_CVCT `#define ATSC_TABLE_CVCT`

CVCT table ID.

Definition at line 60 of file [vct.h](#).

7.6.2.4 ATSC_TABLE_EIT `#define ATSC_TABLE_EIT`

ATSC EIT table ID.

Definition at line 53 of file [atsc_eit.h](#).

7.6.2.5 ATSC_TABLE_MGT `#define ATSC_TABLE_MGT`

ATSC MGT table ID.

Definition at line 51 of file [mgt.h](#).

7.6.2.6 ATSC_TABLE_TVCT `#define ATSC_TABLE_TVCT`

TVCT table ID.

Definition at line 59 of file [vct.h](#).

7.6.2.7 ATSC_TABLE_VCT_PID `#define ATSC_TABLE_VCT_PID`

Program ID with the VCT tables on it.

Definition at line 61 of file [vct.h](#).

7.6.2.8 atsc_vct_channel_foreach `#define atsc_vct_channel_foreach(
 _channel,
 _vct)`

Macro used to find channels on a VCT table.

Parameters

<code>_channel</code>	channel to seek
<code>_vct</code>	pointer to struct atsc_table_vct_channel

Definition at line 202 of file [vct.h](#).

7.6.2.9 DVB_CRC_SIZE `#define DVB_CRC_SIZE`

number of bytes for the descriptor's CRC check

Definition at line 61 of file [descriptors.h](#).

7.6.2.10 dvb_eit_event_foreach `#define dvb_eit_event_foreach(
 _event,
 _eit)`

Macro used to find event on a DVB EIT table.

Parameters

<code>_event</code>	event to seek
<code>_eit</code>	pointer to struct dvb_table_eit_event

Definition at line 162 of file [eit.h](#).

7.6.2.11 DVB_MAX_PAYLOAD_PACKET_SIZE `#define DVB_MAX_PAYLOAD_PACKET_SIZE`

Maximum size of a table session to be parsed.

Definition at line 55 of file [descriptors.h](#).

7.6.2.12 DVB_MPEG_ES_GOP `#define DVB_MPEG_ES_GOP`

Group Of Pictures.

Definition at line 67 of file [mpeg_es.h](#).

7.6.2.13 DVB_MPEG_ES_PIC_START `#define DVB_MPEG_ES_PIC_START`

Picture Start.

Definition at line 63 of file [mpeg_es.h](#).

7.6.2.14 DVB_MPEG_ES_SEQ_EXT `#define DVB_MPEG_ES_SEQ_EXT`

Extension.

Definition at line 66 of file [mpeg_es.h](#).

7.6.2.15 DVB_MPEG_ES_SEQ_START `#define DVB_MPEG_ES_SEQ_START`

Sequence Start.

Definition at line 65 of file [mpeg_es.h](#).

7.6.2.16 DVB_MPEG_ES_SLICES `#define DVB_MPEG_ES_SLICES`

Slices.

Definition at line 68 of file [mpeg_es.h](#).

7.6.2.17 DVB_MPEG_ES_USER_DATA `#define DVB_MPEG_ES_USER_DATA`

User Data.

Definition at line 64 of file [mpeg_es.h](#).

7.6.2.18 DVB_MPEG_PES `#define DVB_MPEG_PES`

MPEG Packetized Elementary Stream magic.

Definition at line 80 of file [mpeg_pes.h](#).

7.6.2.19 DVB_MPEG_PES_AUDIO `#define DVB_MPEG_PES_AUDIO`

PES Audio.

Definition at line 82 of file [mpeg_pes.h](#).

7.6.2.20 DVB_MPEG_PES_VIDEO `#define DVB_MPEG_PES_VIDEO`

PES Video.

Definition at line 83 of file [mpeg_pes.h](#).

7.6.2.21 DVB_MPEG_STREAM_DIRECTORY `#define DVB_MPEG_STREAM_DIRECTORY`

PES Stream directory.

Definition at line 90 of file [mpeg_pes.h](#).

7.6.2.22 DVB_MPEG_STREAM_DSMCC `#define DVB_MPEG_STREAM_DSMCC`

PES DSMCC.

Definition at line 91 of file [mpeg_pes.h](#).

7.6.2.23 DVB_MPEG_STREAM_ECM `#define DVB_MPEG_STREAM_ECM`

PES ECM Stream.

Definition at line 88 of file [mpeg_pes.h](#).

7.6.2.24 DVB_MPEG_STREAM_EMM `#define DVB_MPEG_STREAM_EMM`

PES EMM Stream.

Definition at line 89 of file [mpeg_pes.h](#).

7.6.2.25 DVB_MPEG_STREAM_H222E `#define DVB_MPEG_STREAM_H222E`

PES H.222.1 type E.

Definition at line 92 of file [mpeg_pes.h](#).

7.6.2.26 DVB_MPEG_STREAM_MAP `#define DVB_MPEG_STREAM_MAP`

PES Stream map.

Definition at line 85 of file [mpeg_pes.h](#).

7.6.2.27 DVB_MPEG_STREAM_PADDING `#define DVB_MPEG_STREAM_PADDING`

PES padding.

Definition at line 86 of file [mpeg_pes.h](#).

7.6.2.28 DVB_MPEG_STREAM_PRIVATE_2 `#define DVB_MPEG_STREAM_PRIVATE_2`

PES private.

Definition at line 87 of file [mpeg_pes.h](#).

7.6.2.29 DVB_MPEG_TS `#define DVB_MPEG_TS`

MPEG Transport Stream magic.

Definition at line 50 of file [mpeg_ts.h](#).

7.6.2.30 DVB_MPEG_TS_PACKET_SIZE `#define DVB_MPEG_TS_PACKET_SIZE`

Size of an MPEG packet.

Definition at line 51 of file [mpeg_ts.h](#).

7.6.2.31 dvb_nit_transport_foreach `#define dvb_nit_transport_foreach(
 _tran,
 _nit)`

Macro used to find a transport inside a NIT table.

Parameters

<code>_tran</code>	transport to seek
<code>_nit</code>	pointer to struct dvb_table_nit_transport

Definition at line 189 of file [nit.h](#).

7.6.2.32 dvb_pat_program_foreach `#define dvb_pat_program_foreach(
 _pgm,
 _pat)`

Macro used to find programs on a PAT table.

Parameters

<i>_pgm</i>	program to seek
<i>_pat</i>	pointer to struct dvb_table_pat_program

Definition at line [121](#) of file [pat.h](#).

7.6.2.33 dvb_pmt_stream_foreach `#define dvb_pmt_stream_foreach(
 _stream,
 _pmt)`

Macro used to find streams on a PMT table.

Parameters

<i>_stream</i>	stream to seek
<i>_pmt</i>	pointer to struct dvb_table_pmt_stream

Definition at line [249](#) of file [pmt.h](#).

7.6.2.34 dvb_sdt_service_foreach `#define dvb_sdt_service_foreach(
 _service,
 _sdt)`

Macro used to find services on a SDT table.

Parameters

<i>_service</i>	service to seek
<i>_sdt</i>	pointer to struct dvb_table_sdt_service

Definition at line [137](#) of file [sdt.h](#).

7.6.2.35 DVB_TABLE_CAT `#define DVB_TABLE_CAT`

ATSC CAT table ID.

Definition at line [47](#) of file [cat.h](#).

7.6.2.36 DVB_TABLE_CAT_PID `#define DVB_TABLE_CAT_PID`

ATSC PID table ID.

Definition at line [48](#) of file [cat.h](#).

7.6.2.37 DVB_TABLE_EIT `#define DVB_TABLE_EIT`

DVB EIT table ID for the actual TS.

Definition at line [68](#) of file [eit.h](#).

7.6.2.38 DVB_TABLE_EIT_OTHER `#define DVB_TABLE_EIT_OTHER`

DVB EIT table ID for other TS.

Definition at line [69](#) of file [eit.h](#).

7.6.2.39 DVB_TABLE_EIT_PID `#define DVB_TABLE_EIT_PID`

DVB EIT Program ID.

Definition at line [70](#) of file [eit.h](#).

7.6.2.40 DVB_TABLE_EIT_SCHEDULE `#define DVB_TABLE_EIT_SCHEDULE`

Start table ID for the DVB EIT schedule data on the actual TS The range has 0x0f elements (0x50 to 0x5F).

Definition at line [72](#) of file [eit.h](#).

7.6.2.41 DVB_TABLE_EIT_SCHEDULE_OTHER `#define DVB_TABLE_EIT_SCHEDULE_OTHER`

Start table ID for the DVB EIT schedule data on other TS The range has 0x0f elements (0x60 to 0x6F).

Definition at line [73](#) of file [eit.h](#).

7.6.2.42 DVB_TABLE_NIT `#define DVB_TABLE_NIT`

NIT table ID.

Definition at line 61 of file [nit.h](#).

7.6.2.43 DVB_TABLE_NIT2 `#define DVB_TABLE_NIT2`

NIT table ID (alternative table ID)

Definition at line 62 of file [nit.h](#).

7.6.2.44 DVB_TABLE_NIT_PID `#define DVB_TABLE_NIT_PID`

NIT Program ID.

Definition at line 63 of file [nit.h](#).

7.6.2.45 DVB_TABLE_PAT `#define DVB_TABLE_PAT`

PAT table ID.

Definition at line 55 of file [pat.h](#).

7.6.2.46 DVB_TABLE_PAT_PID `#define DVB_TABLE_PAT_PID`

PAT Program ID.

Definition at line 56 of file [pat.h](#).

7.6.2.47 DVB_TABLE_PMT `#define DVB_TABLE_PMT`

PMT table ID.

Definition at line 52 of file [pmt.h](#).

7.6.2.48 DVB_TABLE_SDT `#define DVB_TABLE_SDT`

SDT table ID.

Definition at line 58 of file [sdt.h](#).

7.6.2.49 DVB_TABLE_SDT2 `#define DVB_TABLE_SDT2`

SDT table ID (alternative table ID)

Definition at line 59 of file [sdt.h](#).

7.6.2.50 DVB_TABLE_SDT_PID `#define DVB_TABLE_SDT_PID`

SDT Program ID.

Definition at line 60 of file [sdt.h](#).

7.6.3 Typedef Documentation**7.6.3.1 dvb_desc_ext_free_func** `typedef void(* dvb_desc_ext_free_func) (const void *desc)`

Function prototype for the extended descriptors parsing free code.

Parameters

<i>desc</i>	struct dvb_desc pointer
-------------	---

Definition at line 157 of file [desc_extension.h](#).

7.6.3.2 dvb_desc_ext_init_func `typedef int(* dvb_desc_ext_init_func) (struct dvb_v5_fe_parms *parms, const uint8_t *buf, struct dvb_extension_descriptor *ext, void *desc)`

Function prototype for the extended descriptors parsing init code.

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>buf</i>	buffer with the content of the descriptor
<i>ext</i>	struct dvb_extension_descriptor pointer
<i>desc</i>	struct dvb_desc pointer

Definition at line 135 of file [desc_extension.h](#).

7.6.3.3 dvb_desc_ext_print_func `typedef void(* dvb_desc_ext_print_func) (struct dvb_v5_fe_parms *parms, const struct dvb_extension_descriptor *ext, const void *desc)`

Function prototype for the extended descriptors parsing print code.

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>buf</i>	buffer with the content of the descriptor
<i>ext</i>	struct dvb_extension_descriptor pointer
<i>desc</i>	struct dvb_desc pointer

Definition at line 148 of file [desc_extension.h](#).

7.6.3.4 dvb_desc_free_func `typedef void(* dvb_desc_free_func) (struct dvb_desc *desc)`

Function prototype for the descriptors memory free code.

Parameters

<i>desc</i>	pointer to struct dvb_desc pointer to be freed
-------------	--

Definition at line 234 of file [descriptors.h](#).

7.6.3.5 dvb_desc_init_func `typedef int(* dvb_desc_init_func) (struct dvb_v5_fe_parms *parms, const uint8_t *buf, struct dvb_desc *desc)`

Function prototype for the descriptors parsing init code.

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>buf</i>	buffer with the content of the descriptor
<i>desc</i>	struct dvb_desc pointer

Definition at line 215 of file [descriptors.h](#).

7.6.3.6 dvb_desc_print_func `typedef void(* dvb_desc_print_func) (struct dvb_v5_fe_parms *parms, const struct dvb_desc *desc)`

Function prototype for the descriptors parsing print code.

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>desc</i>	struct dvb_desc pointer

Definition at line [225](#) of file [descriptors.h](#).

7.6.3.7 dvb_table_init_func typedef void(* dvb_table_init_func) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, void **table)

Function prototype for a function that initializes the descriptors parsing on a table.

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>buf</i>	Buffer with data to be parsed
<i>buflen</i>	Size of the buffer to be parsed
<i>table</i>	pointer to a place where the allocated memory with the table structure will be stored.

Definition at line [79](#) of file [descriptors.h](#).

7.6.3.8 nit_handler_callback_t typedef void nit_handler_callback_t(struct [dvb_table_nit](#) *nit, struct [dvb_desc](#) *desc, void *priv)

typedef for a callback used when a NIT table entry is found

Parameters

<i>nit</i>	a struct dvb_table_nit pointer
<i>desc</i>	a struct dvb_desc pointer
<i>priv</i>	an opaque optional pointer

Definition at line [164](#) of file [nit.h](#).

7.6.3.9 nit_tran_handler_callback_t typedef void nit_tran_handler_callback_t(struct [dvb_table_nit](#) *nit, struct [dvb_table_nit_transport](#) *tran, struct [dvb_desc](#) *desc, void *priv)

typedef for a callback used when a NIT transport table entry is found

Parameters

<i>nit</i>	a struct dvb_table_nit pointer
<i>tran</i>	a struct dvb_table_nit_transport pointer
<i>desc</i>	a struct dvb_desc pointer
<i>priv</i>	an opaque optional pointer

Definition at line 177 of file [nit.h](#).

7.6.4 Enumeration Type Documentation

7.6.4.1 descriptors `enum descriptors`

List containing all descriptors used by Digital TV MPEG-TS.

Enumerator

<code>video_stream_descriptor</code>	video_stream descriptor - ISO/IEC 13818-1
<code>audio_stream_descriptor</code>	audio_stream descriptor - ISO/IEC 13818-1
<code>hierarchy_descriptor</code>	hierarchy descriptor - ISO/IEC 13818-1
<code>registration_descriptor</code>	registration descriptor - ISO/IEC 13818-1
<code>ds_alignment_descriptor</code>	ds_alignment descriptor - ISO/IEC 13818-1
<code>target_background_grid_descriptor</code>	target_background_grid descriptor - ISO/IEC 13818-1
<code>video_window_descriptor</code>	video_window descriptor - ISO/IEC 13818-1
<code>conditional_access_descriptor</code>	conditional_access descriptor - ISO/IEC 13818-1
<code>iso639_language_descriptor</code>	iso639_language descriptor - ISO/IEC 13818-1
<code>system_clock_descriptor</code>	system_clock descriptor - ISO/IEC 13818-1
<code>multiplex_buffer_utilization_descriptor</code>	multiplex_buffer_utilization descriptor - ISO/IEC 13818-1
<code>copyright_descriptor</code>	copyright descriptor - ISO/IEC 13818-1
<code>maximum_bitrate_descriptor</code>	maximum_bitrate descriptor - ISO/IEC 13818-1
<code>private_data_indicator_descriptor</code>	private_data_indicator descriptor - ISO/IEC 13818-1
<code>smoothing_buffer_descriptor</code>	smoothing_buffer descriptor - ISO/IEC 13818-1
<code>std_descriptor</code>	std descriptor - ISO/IEC 13818-1
<code>ibp_descriptor</code>	ibp descriptor - ISO/IEC 13818-1
<code>mpeg4_video_descriptor</code>	mpeg4_video descriptor - ISO/IEC 13818-1
<code>mpeg4_audio_descriptor</code>	mpeg4_audio descriptor - ISO/IEC 13818-1
<code>iod_descriptor</code>	iod descriptor - ISO/IEC 13818-1
<code>sl_descriptor</code>	sl descriptor - ISO/IEC 13818-1
<code>fmc_descriptor</code>	fmc descriptor - ISO/IEC 13818-1
<code>external_es_id_descriptor</code>	external_es_id descriptor - ISO/IEC 13818-1
<code>muxcode_descriptor</code>	muxcode descriptor - ISO/IEC 13818-1
<code>fmxbuffer_size_descriptor</code>	fmxbuffer_size descriptor - ISO/IEC 13818-1
<code>multiplexbuffer_descriptor</code>	multiplexbuffer descriptor - ISO/IEC 13818-1
<code>content_labeling_descriptor</code>	content_labeling descriptor - ISO/IEC 13818-1
<code>metadata_pointer_descriptor</code>	metadata_pointer descriptor - ISO/IEC 13818-1

Enumerator

metadata_descriptor	metadata descriptor - ISO/IEC 13818-1
metadata_std_descriptor	metadata_std descriptor - ISO/IEC 13818-1
AVC_video_descriptor	AVC_video descriptor - ISO/IEC 13818-1.
ipmp_descriptor	ipmp descriptor - ISO/IEC 13818-1
AVC_timing_and_HRD_descriptor	AVC_timing_and_HRD descriptor - ISO/IEC 13818-1.
mpeg2_aac_audio_descriptor	mpeg2_aac_audio descriptor - ISO/IEC 13818-1
flexmux_timing_descriptor	flexmux_timing descriptor - ISO/IEC 13818-1
network_name_descriptor	network_name descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
service_list_descriptor	service_list descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
stuffing_descriptor	stuffing descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
satellite_delivery_system_descriptor	satellite_delivery_system descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
cable_delivery_system_descriptor	cable_delivery_system descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
VBI_data_descriptor	VBI_data descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
VBI_teletext_descriptor	VBI_teletext descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
bouquet_name_descriptor	bouquet_name descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
service_descriptor	service descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
country_availability_descriptor	country_availability descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
linkage_descriptor	linkage descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
NVOD_reference_descriptor	NVOD_reference descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
time_shifted_service_descriptor	time_shifted_service descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
short_event_descriptor	short_event descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
extended_event_descriptor	extended_event descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
time_shifted_event_descriptor	time_shifted_event descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
component_descriptor	component descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
mosaic_descriptor	mosaic descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
stream_identifier_descriptor	stream_identifier descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
CA_identifier_descriptor	CA_identifier descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
content_descriptor	content descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
parental_rating_descriptor	parental_rating descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
teletext_descriptor	teletext descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
telephone_descriptor	telephone descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
local_time_offset_descriptor	local_time_offset descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
subtitling_descriptor	subtitling descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
terrestrial_delivery_system_descriptor	terrestrial_delivery_system descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
multilingual_network_name_descriptor	multilingual_network_name descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
multilingual_bouquet_name_descriptor	multilingual_bouquet_name descriptor - ETSI EN 300 468 V1.11.1 (2010-04)

Enumerator

multilingual_service_name_descriptor	multilingual_service_name descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
multilingual_component_descriptor	multilingual_component descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
private_data_specifier_descriptor	private_data_specifier descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
service_move_descriptor	service_move descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
short_smoothing_buffer_descriptor	short_smoothing_buffer descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
frequency_list_descriptor	frequency_list descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
partial_transport_stream_descriptor	partial_transport_stream descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
data_broadcast_descriptor	data_broadcast descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
scrambling_descriptor	scrambling descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
data_broadcast_id_descriptor	data_broadcast_id descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
transport_stream_descriptor	transport_stream descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
DSNG_descriptor	DSNG descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
PDC_descriptor	PDC descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
AC_3_descriptor	AC_3 descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
ancillary_data_descriptor	ancillary_data descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
cell_list_descriptor	cell_list descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
cell_frequency_link_descriptor	cell_frequency_link descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
announcement_support_descriptor	announcement_support descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
application_signalling_descriptor	application_signalling descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
adaptation_field_data_descriptor	adaptation_field_data descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
service_identifier_descriptor	service_identifier descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
service_availability_descriptor	service_availability descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
default_authority_descriptor	default_authority descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
related_content_descriptor	related_content descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
TVA_id_descriptor	TVA_id descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
content_identifier_descriptor	content_identifier descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
time_slice_fec_identifier_descriptor	time_slice_fec_identifier descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
ECM_repetition_rate_descriptor	ECM_repetition_rate descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
S2_satellite_delivery_system_descriptor	S2_satellite_delivery_system descriptor - ETSI EN 300 468 V1.11.1 (2010-04)

Enumerator

enhanced_AC_3_descriptor	enhanced_AC_3 descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
DTS_descriptor	DTS descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
AAC_descriptor	AAC descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
XAIT_location_descriptor	XAIT_location descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
FTA_content_management_descriptor	FTA_content_management descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
extension_descriptor	extension descriptor - ETSI EN 300 468 V1.11.1 (2010-04)
CUE_identifier_descriptor	CUE_identifier descriptor - SCTE 35 2004.
extended_channel_name	extended_channel_name descriptor - SCTE 35 2004
service_location	service_location descriptor - SCTE 35 2004
component_name_descriptor	component_name descriptor - SCTE 35 2004 See also http://www.etherguidesystems.com/Help/SDOs/ATSC/Semantics/Descriptors/Default.aspx
logical_channel_number_descriptor	logical_channel_number descriptor - SCTE 35 2004 See also http://www.coolstf.com/tsreader/descriptors.html
carousel_id_descriptor	carousel_id descriptor - ABNT NBR 15603-1 2007
association_tag_descriptor	association_tag descriptor - ABNT NBR 15603-1 2007
deferred_association_tags_descriptor	deferred_association_tags descriptor - ABNT NBR 15603-1 2007
hierarchical_transmission_descriptor	hierarchical_transmission descriptor - ABNT NBR 15603-1 2007
digital_copy_control_descriptor	digital_copy_control descriptor - ABNT NBR 15603-1 2007
network_identifier_descriptor	network_identifier descriptor - ABNT NBR 15603-1 2007
partial_transport_stream_time_descriptor	partial_transport_stream_time descriptor - ABNT NBR 15603-1 2007
audio_component_descriptor	audio_component descriptor - ABNT NBR 15603-1 2007
hyperlink_descriptor	hyperlink descriptor - ABNT NBR 15603-1 2007
target_area_descriptor	target_area descriptor - ABNT NBR 15603-1 2007
data_contents_descriptor	data_contents descriptor - ABNT NBR 15603-1 2007
video_decode_control_descriptor	video_decode_control descriptor - ABNT NBR 15603-1 2007
download_content_descriptor	download_content descriptor - ABNT NBR 15603-1 2007
CA_EMM_TS_descriptor	CA_EMM_TS descriptor - ABNT NBR 15603-1 2007.
CA_contract_information_descriptor	CA_contract_information descriptor - ABNT NBR 15603-1 2007.
CA_service_descriptor	CA_service descriptor - ABNT NBR 15603-1 2007.
TS_Information_descriptor	transport_stream_information descriptor - ABNT NBR 15603-1 2007
extended_broadcaster_descriptor	extended_broadcaster descriptor - ABNT NBR 15603-1 2007
logo_transmission_descriptor	logo_transmission descriptor - ABNT NBR 15603-1 2007
basic_local_event_descriptor	basic_local_event descriptor - ABNT NBR 15603-1 2007
reference_descriptor	reference descriptor - ABNT NBR 15603-1 2007
node_relation_descriptor	node_relation descriptor - ABNT NBR 15603-1 2007
short_node_information_descriptor	short_node_information descriptor - ABNT NBR 15603-1 2007

Enumerator

STC_reference_descriptor	STC_reference descriptor - ABNT NBR 15603-1 2007.
series_descriptor	series descriptor - ABNT NBR 15603-1 2007
event_group_descriptor	event_group descriptor - ABNT NBR 15603-1 2007
SI_parameter_descriptor	SI_parameter descriptor - ABNT NBR 15603-1 2007.
broadcaster_Name_Descriptor	broadcaster_Name descriptor - ABNT NBR 15603-1 2007
component_group_descriptor	component_group descriptor - ABNT NBR 15603-1 2007
SI_prime_TS_descriptor	SI_prime_transport_stream descriptor - ABNT NBR 15603-1 2007.
board_information_descriptor	board_information descriptor - ABNT NBR 15603-1 2007
LDT_linkage_descriptor	LDT_linkage descriptor - ABNT NBR 15603-1 2007.
connected_transmission_descriptor	connected_transmission descriptor - ABNT NBR 15603-1 2007
content_availability_descriptor	content_availability descriptor - ABNT NBR 15603-1 2007
service_group_descriptor	service_group descriptor - ABNT NBR 15603-1 2007
carousel_compatible_composite_descriptor	carousel_compatible_composite descriptor - ABNT NBR 15603-1 2007
conditional_playback_descriptor	conditional_playback descriptor - ABNT NBR 15603-1 2007
ISDBT_delivery_system_descriptor	ISDBT_terrestrial_delivery_system descriptor - ABNT NBR 15603-1 2007.
partial_reception_descriptor	partial_reception descriptor - ABNT NBR 15603-1 2007
emergency_information_descriptor	emergency_information descriptor - ABNT NBR 15603-1 2007
data_component_descriptor	data_component descriptor - ABNT NBR 15603-1 2007
system_management_descriptor	system_management descriptor - ABNT NBR 15603-1 2007
atsc_stuffing_descriptor	atsc_stuffing descriptor - ATSC A/65:2009
atsc_ac3_audio_descriptor	atsc_ac3_audio descriptor - ATSC A/65:2009
atsc_caption_service_descriptor	atsc_caption_service descriptor - ATSC A/65:2009
atsc_content_advisory_descriptor	atsc_content_advisory descriptor - ATSC A/65:2009
atsc_extended_channel_descriptor	atsc_extended_channel descriptor - ATSC A/65:2009
atsc_service_location_descriptor	atsc_service_location descriptor - ATSC A/65:2009
atsc_time_shifted_service_descriptor	atsc_time_shifted_service descriptor - ATSC A/65:2009
atsc_component_name_descriptor	atsc_component_name descriptor - ATSC A/65:2009
atsc_DCC_departing_request_descriptor	atsc_DCC_departing_request descriptor - ATSC A/65:2009
atsc_DCC_arriving_request_descriptor	atsc_DCC_arriving_request descriptor - ATSC A/65:2009
atsc_redistribution_control_descriptor	atsc_redistribution_control descriptor - ATSC A/65:2009
atsc_ATSC_private_information_descriptor	atsc_ATSC_private_information descriptor - ATSC A/65:2009
atsc_genre_descriptor	atsc_genre descriptor - ATSC A/65:2009

Definition at line 592 of file [descriptors.h](#).

7.6.4.2 dvb_mpeg_es_frame_t enum dvb_mpeg_es_frame_t

MPEG frame types.

Enumerator

DVB_MPEG_ES_FRAME_UNKNOWN	Unknown frame.
DVB_MPEG_ES_FRAME_I	I frame.

Enumerator

DVB_MPEG_ES_FRAME_P	P frame.
DVB_MPEG_ES_FRAME_B	B frame.
DVB_MPEG_ES_FRAME_D	D frame.

Definition at line 165 of file [mpeg_es.h](#).

7.6.4.3 dvb_streams enum dvb_streams

Add support for MPEG-TS Stream types.

Enumerator

stream_video	ISO/IEC 11172 Video.
stream_video_h262	ITU-T Rec. H.262 ISO/IEC 13818-2 Video or ISO/IEC 11172-2 constrained parameter video stream
stream_audio	ISO/IEC 11172 Audio.
stream_audio_13818_3	ISO/IEC 13818-3 Audio.
stream_private_sections	ITU-T Rec. H.222.0 ISO/IEC 13818-1 private_sections
stream_private_data	ITU-T Rec. H.222.0 ISO/IEC 13818-1 PES packets containing private data
stream_mheg	ISO/IEC 13522 MHEG.
stream_h222	ITU-T Rec. H.222.0 ISO/IEC 13818-1 Annex A DSM-CC
stream_h222_1	ITU-T Rec. H.222.1
stream_13818_6_A	ISO/IEC 13818-6 type A.
stream_13818_6_B	ISO/IEC 13818-6 type B.
stream_13818_6_C	ISO/IEC 13818-6 type C.
stream_13818_6_D	ISO/IEC 13818-6 type D.
stream_h222_aux	ITU-T Rec. H.222.0 ISO/IEC 13818-1 auxiliary
stream_audio_adts	ISO/IEC 13818-7 Audio with ADTS transport syntax.
stream_video_14496_2	ISO/IEC 14496-2 Visual.
stream_audio_latm	ISO/IEC 14496-3 Audio with the LATM transport syntax as defined in ISO/IEC 14496-3 / AMD 1.
stream_14496_1_pes	ISO/IEC 14496-1 SL-packetized stream or FlexMux stream carried in PES packets.
stream_14496_1_iso	ISO/IEC 14496-1 SL-packetized stream or FlexMux stream carried in ISO/IEC14496_sections.
stream_download	ISO/IEC 13818-6 Synchronized Download Protocol.
stream_video_h264	
stream_audio_14496_3	
stream_video_hevc	
stream_video_cavs	
stream_video_moto	
stream_audio_a52	
stream_scte_27	
stream_audio_sdds	
stream_audio_dts_hdmv	
stream_audio_e_ac3	
stream_audio_dts	
stream_audio_a52_vls	
stream_spu_vls	
stream_audio_sdds2	

Definition at line 106 of file [pmt.h](#).

7.6.5 Function Documentation

7.6.5.1 atsc_table_eit_free() void atsc_table_eit_free (
 struct [atsc_table_eit](#) * *table*)

Frees all data allocated by the ATSC EIT table parser.

Parameters

<i>table</i>	pointer to struct atsc_table_eit to be freed
--------------	--

7.6.5.2 atsc_table_eit_init() ssize_t atsc_table_eit_init (
 struct [dvb_v5_fe_parms](#) * *parms*,
 const uint8_t * *buf*,
 ssize_t *buflen*,
 struct [atsc_table_eit](#) ** *table*)

Initializes and parses ATSC EIT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the EIT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct atsc_table_eit to be allocated and filled

This function allocates an ATSC EIT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.3 atsc_table_eit_print() void atsc_table_eit_print (
 struct [dvb_v5_fe_parms](#) * *parms*,
 struct [atsc_table_eit](#) * *table*)

Prints the content of the ATSC EIT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct atsc_table_eit

7.6.5.4 atsc_table_mgt_free() `void atsc_table_mgt_free (struct atsc_table_mgt * table)`

Frees all data allocated by the MGT table parser.

Parameters

<i>table</i>	pointer to struct atsc_table_mgt to be freed
--------------	--

7.6.5.5 atsc_table_mgt_init() `ssize_t atsc_table_mgt_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, ssize_t buflen, struct atsc_table_mgt ** table)`

Initializes and parses MGT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the MGT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct atsc_table_mgt to be allocated and filled

This function allocates an ATSC MGT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.6 atsc_table_mgt_print() `void atsc_table_mgt_print (struct dvb_v5_fe_parms * parms, struct atsc_table_mgt * table)`

Prints the content of the MGT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct atsc_table_mgt

7.6.5.7 atsc_table_vct_free() `void atsc_table_vct_free (struct atsc_table_vct * table)`

Frees all data allocated by the VCT table parser.

Parameters

<i>table</i>	pointer to struct atsc_table_vct to be freed
--------------	--

7.6.5.8 atsc_table_vct_init() `ssize_t atsc_table_vct_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, ssize_t buflen, struct atsc_table_vct ** table)`

Initializes and parses VCT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the VCT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct atsc_table_vct to be allocated and filled

This function allocates an ATSC VCT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.9 atsc_table_vct_print() `void atsc_table_vct_print (struct dvb_v5_fe_parms * parms, struct atsc_table_vct * table)`

Prints the content of the VCT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct atsc_table_vct

7.6.5.10 dvb_bcd() `uint32_t dvb_bcd (`
`uint32_t bcd)`

Converts from BCD to CPU integer internal representation.

Parameters

<i>bcd</i>	value in BCD encoding
------------	-----------------------

7.6.5.11 dvb_desc_free() `void dvb_desc_free (`
`struct dvb_desc ** list)`

frees a [dvb_desc](#) linked list

Parameters

<i>list</i>	struct dvb_desc pointer.
-------------	--

7.6.5.12 dvb_desc_parse() `int dvb_desc_parse (`
`struct dvb_v5_fe_parms * parms,`
`const uint8_t * buf,`
`uint16_t buflen,`
`struct dvb_desc ** head_desc)`

parse MPEG-TS descriptors

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>buf</i>	Buffer with data to be parsed
<i>buflen</i>	Size of the buffer to be parsed
<i>head_desc</i>	pointer to the place to store the parsed data

This function takes a buf as argument and parses it to find the MPEG-TS descriptors inside it, creating a linked list. On success, head_desc will be allocated and filled with a linked list with the descriptors found inside the buffer.

This function is used by the several MPEG-TS table handlers to parse the entire table that got read by [dvb_read_sessions](#) and other similar functions.

Returns

Returns 0 on success, a negative value otherwise.

7.6.5.13 dvb_desc_print() void dvb_desc_print (
 struct dvb_v5_fe_parms * parms,
 struct dvb_desc * desc)

prints the contents of a struct [dvb_desc](#) linked list

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>desc</i>	struct dvb_desc pointer.

7.6.5.14 dvb_hexdump() void dvb_hexdump (
 struct dvb_v5_fe_parms * parms,
 const char * prefix,
 const unsigned char * buf,
 int len)

dumps data into the logs in hexadecimal format

Parameters

<i>parms</i>	Struct dvb_v5_fe_parms pointer
<i>prefix</i>	String to be printed before the dvb_hexdump
<i>buf</i>	Buffer to hex dump
<i>len</i>	Number of bytes to show

7.6.5.15 dvb_mpeg_es_pic_start_init() int dvb_mpeg_es_pic_start_init (
 const uint8_t * buf,
 ssize_t buflen,
 struct dvb_mpeg_es_pic_start * pic_start)

Initialize a struct [dvb_mpeg_es_pic_start](#) from buffer.

Parameters

<i>buf</i>	Buffer
<i>buflen</i>	Length of buffer
<i>pic_start</i>	Pointer to allocated struct dvb_mpeg_es_pic_start

Returns

If buflen too small, return -1, 0 otherwise.

This function copies the length of struct `dvb_mpeg_es_pic_start` to `pic_start` and fixes endianness. `seq_start` has to be allocated with malloc.

7.6.5.16 dvb_mpeg_es_pic_start_print() `void dvb_mpeg_es_pic_start_print (`
 struct `dvb_v5_fe_parms` * `parms`,
 struct `dvb_mpeg_es_pic_start` * `pic_start`)

Print details of struct `dvb_mpeg_es_pic_start`.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> for log functions
<i>pic_start</i>	Pointer to struct <code>dvb_mpeg_es_pic_start</code> to print

This function prints the fields of struct `dvb_mpeg_es_pic_start`

7.6.5.17 dvb_mpeg_es_seq_start_init() `int dvb_mpeg_es_seq_start_init (`
 const uint8_t * `buf`,
 ssize_t `buflen`,
 struct `dvb_mpeg_es_seq_start` * `seq_start`)

Initialize a struct `dvb_mpeg_es_seq_start` from buffer.

Parameters

<i>buf</i>	Buffer
<i>buflen</i>	Length of buffer
<i>seq_start</i>	Pointer to allocated struct <code>dvb_mpeg_es_seq_start</code>

Returns

If buflen too small, return -1, 0 otherwise.

This function copies the length of struct `dvb_mpeg_es_seq_start` to `seq_start` and fixes endianness. `seq_start` has to be allocated with malloc.

7.6.5.18 dvb_mpeg_es_seq_start_print() `void dvb_mpeg_es_seq_start_print (`
 struct `dvb_v5_fe_parms` * `parms`,
 struct `dvb_mpeg_es_seq_start` * `seq_start`)

Print details of struct `dvb_mpeg_es_seq_start`.

Parameters

<i>parms</i>	struct <code>dvb_v5_fe_parms</code> for log functions
<i>seq_start</i>	Pointer to struct <code>dvb_mpeg_es_seq_start</code> to print

This function prints the fields of struct [dvb_mpeg_es_seq_start](#)

7.6.5.19 dvb_mpeg_pes_free() void dvb_mpeg_pes_free (
 struct [dvb_mpeg_pes](#) * pes)

Deallocate memory associated with a struct [dvb_mpeg_pes](#).

Parameters

<i>pes</i>	struct dvb_mpeg_pes to be deallocated
------------	---

If the pointer *pes* was allocated dynamically, this function can be used to free the memory.

7.6.5.20 dvb_mpeg_pes_init() ssize_t dvb_mpeg_pes_init (
 struct [dvb_v5_fe_parms](#) * parms,
 const uint8_t * buf,
 ssize_t buflen,
 uint8_t * table)

Initialize a struct [dvb_mpeg_pes](#) from buffer.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms for log functions
<i>buf</i>	Buffer
<i>buflen</i>	Length of buffer
<i>table</i>	Pointer to allocated struct dvb_mpeg_pes

Returns

Length of data in table

This function copies the length of struct [dvb_mpeg_pes](#) to table and fixes endianness. The pointer table has to be allocated on stack or dynamically.

7.6.5.21 dvb_mpeg_pes_print() void dvb_mpeg_pes_print (
 struct [dvb_v5_fe_parms](#) * parms,
 struct [dvb_mpeg_pes](#) * pes)

Print details of struct [dvb_mpeg_pes](#).

Parameters

<i>parms</i>	struct dvb_v5_fe_parms for log functions
<i>pes</i>	Pointer to struct dvb_mpeg_pes to print

This function prints the fields of struct [dvb_mpeg_pes](#)

7.6.5.22 dvb_mpeg_ts_free() void dvb_mpeg_ts_free (struct dvb_mpeg_ts * ts)

Deallocate memory associated with a struct [dvb_mpeg_ts](#).

Parameters

<i>ts</i>	struct dvb_mpeg_ts to be deallocated
-----------	--

If *ts* was allocated dynamically, this function can be used to free the memory.

7.6.5.23 dvb_mpeg_ts_init() ssize_t dvb_mpeg_ts_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, ssize_t buflen, uint8_t * table, ssize_t * table_length)

Initialize a struct [dvb_mpeg_ts](#) from buffer.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms for log functions
<i>buf</i>	Buffer
<i>buflen</i>	Length of buffer
<i>table</i>	Pointer to allocated struct dvb_mpeg_ts
<i>table_length</i>	Pointer to size_t where length will be written to

Returns

Length of data in table

This function copies the length of struct [dvb_mpeg_ts](#) to *table* and fixes endianness. The pointer *table* has to be allocated on stack or dynamically.

7.6.5.24 dvb_mpeg_ts_print() void dvb_mpeg_ts_print (struct dvb_v5_fe_parms * parms, struct dvb_mpeg_ts * ts)

Print details of struct [dvb_mpeg_ts](#).

Parameters

<i>parms</i>	struct dvb_v5_fe_parms for log functions
<i>ts</i>	Pointer to struct dvb_mpeg_ts to print

This function prints the fields of struct [dvb_mpeg_ts](#)

7.6.5.25 dvb_table_eit_free() void dvb_table_eit_free (struct dvb_table_eit * table)

Frees all data allocated by the DVB EIT table parser.

Parameters

<i>table</i>	pointer to struct dvb_table_eit to be freed
--------------	---

7.6.5.26 dvb_table_eit_init() `ssize_t dvb_table_eit_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, ssize_t buflen, struct dvb_table_eit ** table)`

Initializes and parses EIT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the EIT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct dvb_table_eit to be allocated and filled

This function allocates an EIT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.27 dvb_table_eit_print() `void dvb_table_eit_print (struct dvb_v5_fe_parms * parms, struct dvb_table_eit * table)`

Prints the content of the DVB EIT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct dvb_table_eit

7.6.5.28 dvb_table_header_init() `void dvb_table_header_init (struct dvb_table_header * header)`

Initializes and parses MPEG-TS table header.

Parameters

<i>header</i>	pointer to struct dvb_table_header to be parsed
---------------	---

7.6.5.29 dvb_table_header_print() void dvb_table_header_print (
 struct [dvb_v5_fe_parms](#) * *parms*,
 const struct [dvb_table_header](#) * *header*)

Prints the content of the MPEG-TS table header.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>header</i>	pointer to struct dvb_table_header to be printed

7.6.5.30 dvb_table_nit_descriptor_handler() void dvb_table_nit_descriptor_handler (
 struct [dvb_v5_fe_parms](#) * *parms*,
 struct [dvb_table_nit](#) * *table*,
 enum [descriptors](#) *descriptor*,
 [nit_handler_callback_t](#) * *call_nit*,
 [nit_tran_handler_callback_t](#) * *call_tran*,
 void * *priv*)

For each entry at NIT and NIT transport tables, call a callback.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct dvb_table_nit
<i>descriptor</i>	indicates the NIT table descriptor to seek
<i>call_nit</i>	a nit_handler_callback_t function to be called when a new entry at the NIT table is found (or NULL).
<i>call_tran</i>	a nit_tran_handler_callback_t function to be called when a new entry at the NIT transport table is found (or NULL).
<i>priv</i>	an opaque pointer to be optionally used by the callbacks. The function won't touch on it, just use as an argument for the callback functions.

When parsing a NIT entry, we need to call some code to properly handle when a given descriptor in the table is found. This is used, for example, to create newer transponders to seek during scan.

For example, to seek for the CATV delivery system descriptor and call a function that would add a new transponder to a scan procedure:

```
dvb_table_nit_descriptor_handler(
    &parms->p, dvb_scan_handler->nit,
    cable_delivery_system_descriptor,
    NULL, add_update_nit_dvbc, &tr);
```

7.6.5.31 dvb_table_nit_free() void dvb_table_nit_free (
 struct dvb_table_nit * table)

Frees all data allocated by the NIT table parser.

Parameters

<i>table</i>	pointer to struct dvb_table_nit to be freed
--------------	---

7.6.5.32 dvb_table_nit_init() ssize_t dvb_table_nit_init (
 struct dvb_v5_fe_parms * parms,
 const uint8_t * buf,
 ssize_t buflen,
 struct dvb_table_nit ** table)

Initializes and parses NIT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the NIT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct dvb_table_nit to be allocated and filled

This function allocates a NIT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.33 dvb_table_nit_print() void dvb_table_nit_print (
 struct dvb_v5_fe_parms * parms,
 struct dvb_table_nit * table)

Prints the content of the NIT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct dvb_table_nit

7.6.5.34 dvb_table_pat_free() void dvb_table_pat_free (
 struct dvb_table_pat * table)

Frees all data allocated by the PAT table parser.

Parameters

<i>table</i>	pointer to struct dvb_table_pat to be freed
--------------	---

7.6.5.35 dvb_table_pat_init() `ssize_t dvb_table_pat_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `ssize_t buflen,`
 `struct dvb_table_pat ** table)`

Initializes and parses PAT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the PAT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct dvb_table_pat to be allocated and filled

This function allocates a PAT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.36 dvb_table_pat_print() `void dvb_table_pat_print (`
 `struct dvb_v5_fe_parms * parms,`
 `struct dvb_table_pat * table)`

Prints the content of the PAT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct dvb_table_pat

7.6.5.37 dvb_table_pmt_free() `void dvb_table_pmt_free (`
 `struct dvb_table_pmt * table)`

Frees all data allocated by the PMT table parser.

Parameters

<i>table</i>	pointer to struct dvb_table_pmt to be freed
--------------	---

7.6.5.38 dvb_table_pmt_init() `ssize_t dvb_table_pmt_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `ssize_t buflen,`
 `struct dvb_table_pmt ** table)`

Initializes and parses PMT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the PMT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct dvb_table_pmt to be allocated and filled

This function allocates a PMT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.39 dvb_table_pmt_print() `void dvb_table_pmt_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_table_pmt * table)`

Prints the content of the PAT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct dvb_table_pmt

7.6.5.40 dvb_table_sdt_free() `void dvb_table_sdt_free (`
 `struct dvb_table_sdt * table)`

Frees all data allocated by the SDT table parser.

Parameters

<i>table</i>	pointer to struct dvb_table_sdt to be freed
--------------	---

7.6.5.41 dvb_table_sdt_init() `ssize_t dvb_table_sdt_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `ssize_t buflen,`
 `struct dvb_table_sdt ** table)`

Initializes and parses SDT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the SDT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct dvb_table_sdt to be allocated and filled

This function allocates a SDT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.6.5.42 dvb_table_sdt_print() `void dvb_table_sdt_print (`
 `struct dvb_v5_fe_parms * parms,`
 `struct dvb_table_sdt * table)`

Prints the content of the SDT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct dvb_table_sdt

7.6.5.43 dvb_time() `void dvb_time (`
 `const uint8_t data[5],`
 `struct tm * tm)`

Converts a DVB EIT formatted timestamp into struct tm.

Parameters

<i>data</i>	event on DVB EIT time format
<i>tm</i>	pointer to struct tm where the converted timestamp will be stored.

7.6.6 Variable Documentation

7.6.6.1 dvb_descriptors `const struct dvb_descriptor dvb_descriptors[] [extern]`

Contains the parsers for the several descriptors.

7.6.6.2 dvb_mpeg_es_frame_names `const char* dvb_mpeg_es_frame_names[5] [extern]`

Vector that translates from enum dvb_mpeg_es_frame_t to string.

7.6.6.3 dvb_table_initializers `const dvb_table_init_func dvb_table_initializers[256] [extern]`

Table with all possible descriptors.

7.6.6.4 pmt_stream_name `const char* pmt_stream_name[] [extern]`

Converts from enum dvb_streams into a string.

7.7 Parsers for several MPEG-TS descriptors

Files

- file [desc_atsc_service_location.h](#)
Provides the descriptors for ATSC service location.
- file [desc_ca.h](#)
Provides the descriptors for Conditional Access.
- file [desc_ca_identifier.h](#)
Provides the descriptors for the Conditional Access identifier.
- file [desc_cable_delivery.h](#)
Provides the descriptors for the cable delivery system descriptor.
- file [desc_event_extended.h](#)
Provides the descriptors for the extended event descriptor.
- file [desc_event_short.h](#)
Provides the descriptors for the short event descriptor.

- file [desc_extension.h](#)
Provides the descriptors for the extension descriptor.
- file [desc_frequency_list.h](#)
Provides the descriptors for the frequency list descriptor.
- file [desc_hierarchy.h](#)
Provides the descriptors for the hierarchy descriptor.
- file [desc_isdbt_delivery.h](#)
Provides the descriptors for the ISDB-T terrestrial delivery system.
- file [desc_language.h](#)
Provides the descriptors for the ISO639 language descriptor.
- file [desc_logical_channel.h](#)
Provides the descriptors for the LCN - Logican Channel Number.
- file [desc_network_name.h](#)
Provides the descriptors for the network name descriptor.
- file [desc_partial_reception.h](#)
Provides the descriptors for the ISDB partial reception descriptor.
- file [desc_registration_id.h](#)
Provides the descriptors for the registration descriptor.
- file [desc_sat.h](#)
Provides the descriptors for the satellite delivery system descriptor.
- file [desc_service.h](#)
Provides the descriptors for the service descriptor.
- file [desc_t2_delivery.h](#)
Provides the descriptors for the DVB-T2 delivery system descriptor.
- file [desc_terrestrial_delivery.h](#)
Provides the descriptors for the DVB-T terrestrial delivery system descriptor.
- file [desc_ts_info.h](#)
Provides the descriptors for the ISDB TS information descriptor.

Data Structures

- struct [atsc_desc_service_location_elementary](#)
service location elementary descriptors
- struct [atsc_desc_service_location](#)
Describes the elementary streams inside a PAT table for ATSC.
- struct [dvb_desc_ca](#)
Contains the private data for Conditional Access.
- struct [dvb_desc_ca_identifier](#)
Indicates if a particular bouquet, service or event is associated with a CA system.
- struct [dvb_desc_cable_delivery](#)
Structure containing the cable delivery system descriptor.
- struct [dvb_desc_event_extended](#)
Structure containing the extended event descriptor.
- struct [dvb_desc_event_short](#)
Structure containing the short event descriptor.
- struct [dvb_extension_descriptor](#)
Structure containing the extended descriptors.
- struct [dvb_ext_descriptor](#)
Structure that describes the parser functions for the extended descriptors.
- struct [dvb_desc_frequency_list](#)

- Struct containing the frequency list descriptor.*

 - struct [dvb_desc_hierarchy](#)

Structure containing the hierarchy descriptor.
- struct [isdbt_desc_terrestrial_delivery_system](#)

Struct containing the ISDB-T terrestrial delivery system.
- struct [dvb_desc_language](#)

Structure containing the ISO639 language descriptor.
- struct [dvb_desc_logical_channel_number](#)

Structure containing the logical channel number entires.
- struct [dvb_desc_logical_channel](#)

Structure containing the logical channel number descriptor.
- struct [dvb_desc_network_name](#)

Struct containing the network name descriptor.
- struct [isdb_partial_reception_service_id](#)

Service ID that uses partial reception.
- struct [isdb_desc_partial_reception](#)

Structure containing the partial reception descriptor.
- struct [dvb_desc_registration](#)

Struct containing the frequency list descriptor.
- struct [dvb_desc_sat](#)

Structure containing the satellite delivery system descriptor.
- struct [dvb_desc_service](#)

Structure containing the service descriptor.
- struct [dvb_desc_t2_delivery_subcell_old](#)

Structure to describe transponder subcell extension and frequencies.
- struct [dvb_desc_t2_delivery_subcell](#)

Structure to describe transponder subcell extension and frequencies.
- struct [dvb_desc_t2_delivery_cell](#)

Structure to describe transponder cells.
- struct [dvb_desc_t2_delivery](#)

Structure containing the T2 delivery system descriptor.
- struct [dvb_desc_terrestrial_delivery](#)

Structure containing the DVB-T terrestrial delivery system descriptor.
- struct [dvb_desc_ts_info_transmission_type](#)

ISDB TS information transmission type.
- struct [dvb_desc_ts_info](#)

Structure describing the ISDB TS information descriptor.

Enumerations

- enum [extension_descriptors](#) {
[image_icon_descriptor](#) , [cpdm_delivery_signalling_descriptor](#) , [CP_descriptor](#) , [CP_identifier_descriptor](#) ,
[T2_delivery_system_descriptor](#) , [SH_delivery_system_descriptor](#) , [supplementary_audio_descriptor](#) ,
[network_change_notify_descriptor](#) ,
[message_descriptor](#) , [target_region_descriptor](#) , [target_region_name_descriptor](#) , [service_relocated_descriptor](#)
}

List containing all extended descriptors used by Digital TV MPEG-TS as defined at ETSI EN 300 468 V1.11.1.

Functions

- int `atsc_desc_service_location_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the service location descriptor.
- void `atsc_desc_service_location_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the service location descriptor.
- void `atsc_desc_service_location_free` (struct `dvb_desc` *desc)
Frees all data allocated by the service location descriptor.
- int `dvb_desc_ca_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the CA descriptor.
- void `dvb_desc_ca_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the CA descriptor.
- void `dvb_desc_ca_free` (struct `dvb_desc` *desc)
Frees all data allocated by the CA descriptor.
- int `dvb_desc_ca_identifier_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the CA identifier descriptor.
- void `dvb_desc_ca_identifier_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the CA identifier descriptor.
- void `dvb_desc_ca_identifier_free` (struct `dvb_desc` *desc)
Frees all data allocated by the CA identifier descriptor.
- int `dvb_desc_cable_delivery_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the service location descriptor.
- void `dvb_desc_cable_delivery_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the service location descriptor.
- int `dvb_desc_event_extended_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the extended event descriptor.
- void `dvb_desc_event_extended_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the extended event descriptor.
- void `dvb_desc_event_extended_free` (struct `dvb_desc` *desc)
Frees all data allocated by the extended event descriptor.
- int `dvb_desc_event_short_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the short event descriptor.
- void `dvb_desc_event_short_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the short event descriptor.
- void `dvb_desc_event_short_free` (struct `dvb_desc` *desc)
Frees all data allocated by the short event descriptor.
- int `dvb_extension_descriptor_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the extended descriptor.
- void `dvb_extension_descriptor_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the extended descriptor.
- void `dvb_extension_descriptor_free` (struct `dvb_desc` *desc)
Frees all data allocated by the extended descriptor.
- int `dvb_desc_frequency_list_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the frequency list descriptor.
- void `dvb_desc_frequency_list_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the frequency list descriptor.
- int `dvb_desc_hierarchy_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)

- Initializes and parses the hierarchy descriptor.*
- void `dvb_desc_hierarchy_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the hierarchy descriptor.*
- int `isdbt_desc_delivery_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the ISDB-T terrestrial delivery system descriptor.*
- void `isdbt_desc_delivery_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the ISDB-T terrestrial delivery system descriptor.*
- void `isdbt_desc_delivery_free` (struct `dvb_desc` *desc)
- Frees all data allocated by the ISDB-T terrestrial delivery system descriptor.*
- int `dvb_desc_language_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the language descriptor.*
- void `dvb_desc_language_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the language descriptor.*
- int `dvb_desc_logical_channel_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the logical channel number descriptor.*
- void `dvb_desc_logical_channel_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the logical channel number descriptor.*
- void `dvb_desc_logical_channel_free` (struct `dvb_desc` *desc)
- Frees all data allocated by the logical channel number descriptor.*
- int `dvb_desc_network_name_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the network name descriptor.*
- void `dvb_desc_network_name_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the network name descriptor.*
- void `dvb_desc_network_name_free` (struct `dvb_desc` *desc)
- Frees all data allocated by the network name descriptor.*
- int `isdb_desc_partial_reception_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the ISDB-T partial reception descriptor.*
- void `isdb_desc_partial_reception_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the ISDB-T partial reception descriptor.*
- void `isdb_desc_partial_reception_free` (struct `dvb_desc` *desc)
- Frees all data allocated by the ISDB-T partial reception descriptor.*
- int `dvb_desc_registration_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the registration descriptor.*
- void `dvb_desc_registration_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the registration descriptor.*
- void `dvb_desc_registration_free` (struct `dvb_desc` *desc)
- Frees all data allocated by the registration descriptor.*
- int `dvb_desc_sat_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the satellite delivery system descriptor.*
- void `dvb_desc_sat_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the satellite delivery system descriptor.*
- int `dvb_desc_service_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
- Initializes and parses the service descriptor.*
- void `dvb_desc_service_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
- Prints the content of the service descriptor.*
- void `dvb_desc_service_free` (struct `dvb_desc` *desc)
- Frees all data allocated by the service descriptor.*

- `int dvb_desc_t2_delivery_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_extension_descriptor` *ext, void *desc)
Initializes and parses the T2 delivery system descriptor.
- `void dvb_desc_t2_delivery_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_extension_descriptor` *ext, const void *desc)
Prints the content of the T2 delivery system descriptor.
- `void dvb_desc_t2_delivery_free` (const void *desc)
Frees all data allocated by the T2 delivery system descriptor.
- `void dvb_desc_terrestrial_delivery_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the DVB-T terrestrial delivery system descriptor.
- `int dvb_desc_ts_info_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the ISDB TS information descriptor.
- `void dvb_desc_ts_info_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the ISDB TS information descriptor.
- `void dvb_desc_ts_info_free` (struct `dvb_desc` *desc)
Frees all data allocated by the ISDB TS information descriptor.

7.7.1 Detailed Description

7.7.2 Enumeration Type Documentation

7.7.2.1 extension_descriptors `enum extension_descriptors`

List containing all extended descriptors used by Digital TV MPEG-TS as defined at ETSI EN 300 468 V1.11.1.

Enumerator

<code>image_icon_descriptor</code>	image icon descriptor
<code>cpcm_delivery_signalling_descriptor</code>	Content Protection/Copy Management (CPCM) delivery signalling descriptor.
<code>CP_descriptor</code>	Content Protection descriptor.
<code>CP_identifier_descriptor</code>	Content Protection identifier descriptor.
<code>T2_delivery_system_descriptor</code>	DVB-T2 delivery system descriptor.
<code>SH_delivery_system_descriptor</code>	DVB-SH delivery system descriptor.
<code>supplementary_audio_descriptor</code>	supplementary audio descriptor
<code>network_change_notify_descriptor</code>	network change notify descriptor
<code>message_descriptor</code>	message descriptor
<code>target_region_descriptor</code>	target region descriptor
<code>target_region_name_descriptor</code>	target region name descriptor
<code>service_relocated_descriptor</code>	service relocated descriptor

Definition at line 89 of file `desc_extension.h`.

7.7.3 Function Documentation

7.7.3.1 atsc_desc_service_location_free() void atsc_desc_service_location_free (struct dvb_desc * desc)

Frees all data allocated by the service location descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.2 atsc_desc_service_location_init() int atsc_desc_service_location_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, struct dvb_desc * desc)

Initializes and parses the service location descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.3 atsc_desc_service_location_print() void atsc_desc_service_location_print (struct dvb_v5_fe_parms * parms, const struct dvb_desc * desc)

Prints the content of the service location descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.4 dvb_desc_ca_free() void dvb_desc_ca_free (struct dvb_desc * desc)

Frees all data allocated by the CA descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.5 dvb_desc_ca_identifier_free() `void dvb_desc_ca_identifier_free (struct dvb_desc * desc)`

Frees all data allocated by the CA identifier descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.6 dvb_desc_ca_identifier_init() `int dvb_desc_ca_identifier_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, struct dvb_desc * desc)`

Initializes and parses the CA identifier descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.7 dvb_desc_ca_identifier_print() `void dvb_desc_ca_identifier_print (struct dvb_v5_fe_parms * parms, const struct dvb_desc * desc)`

Prints the content of the CA identifier descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.8 dvb_desc_ca_init() `int dvb_desc_ca_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `struct dvb_desc * desc)`

Initializes and parses the CA descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.9 dvb_desc_ca_print() `void dvb_desc_ca_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the CA descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.10 dvb_desc_cable_delivery_init() `int dvb_desc_cable_delivery_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `struct dvb_desc * desc)`

Initializes and parses the service location descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function initializes and makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Currently, no memory is allocated internally.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.11 dvb_desc_cable_delivery_print() `void dvb_desc_cable_delivery_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the service location descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.12 dvb_desc_event_extended_free() `void dvb_desc_event_extended_free (`
 `struct dvb_desc * desc)`

Frees all data allocated by the extended event descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.13 dvb_desc_event_extended_init() `int dvb_desc_event_extended_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `struct dvb_desc * desc)`

Initializes and parses the extended event descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow

the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.14 dvb_desc_event_extended_print() `void dvb_desc_event_extended_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the extended event descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.15 dvb_desc_event_short_free() `void dvb_desc_event_short_free (`
 `struct dvb_desc * desc)`

Frees all data allocated by the short event descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.16 dvb_desc_event_short_init() `int dvb_desc_event_short_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `struct dvb_desc * desc)`

Initializes and parses the short event descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.17 dvb_desc_event_short_print() `void dvb_desc_event_short_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the short event descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.18 dvb_desc_frequency_list_init() `int dvb_desc_frequency_list_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `struct dvb_desc * desc)`

Initializes and parses the frequency list descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function initializes and makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Currently, no memory is allocated internally.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.19 dvb_desc_frequency_list_print() `void dvb_desc_frequency_list_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the frequency list descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.20 dvb_desc_hierarchy_init() `int dvb_desc_hierarchy_init (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const uint8_t * *buf*,
 struct [dvb_desc](#) * *desc*)

Initializes and parses the hierarchy descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function initializes and makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Currently, no memory is allocated internally.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.21 dvb_desc_hierarchy_print() `void dvb_desc_hierarchy_print (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const struct [dvb_desc](#) * *desc*)

Prints the content of the hierarchy descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.22 dvb_desc_language_init() `int dvb_desc_language_init (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const uint8_t * *buf*,
 struct [dvb_desc](#) * *desc*)

Initializes and parses the language descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function initializes and makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Currently, no memory is allocated internally.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.23 [dvb_desc_language_print\(\)](#) `void dvb_desc_language_print (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const struct [dvb_desc](#) * *desc*)

Prints the content of the language descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.24 [dvb_desc_logical_channel_free\(\)](#) `void dvb_desc_logical_channel_free (`
 struct [dvb_desc](#) * *desc*)

Frees all data allocated by the logical channel number descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.25 [dvb_desc_logical_channel_init\(\)](#) `int dvb_desc_logical_channel_init (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const uint8_t * *buf*,
 struct [dvb_desc](#) * *desc*)

Initializes and parses the logical channel number descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.26 dvb_desc_logical_channel_print() `void dvb_desc_logical_channel_print (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const struct [dvb_desc](#) * *desc*)

Prints the content of the logical channel number descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.27 dvb_desc_network_name_free() `void dvb_desc_network_name_free (`
 struct [dvb_desc](#) * *desc*)

Frees all data allocated by the network name descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.28 dvb_desc_network_name_init() `int dvb_desc_network_name_init (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const uint8_t * *buf*,
 struct [dvb_desc](#) * *desc*)

Initializes and parses the network name descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
--------------	---

Parameters

<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.29 [dvb_desc_network_name_print\(\)](#) `void dvb_desc_network_name_print (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const struct [dvb_desc](#) * *desc*)

Prints the content of the network name descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.30 [dvb_desc_registration_free\(\)](#) `void dvb_desc_registration_free (`
 struct [dvb_desc](#) * *desc*)

Frees all data allocated by the registration descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.31 [dvb_desc_registration_init\(\)](#) `int dvb_desc_registration_init (`
 struct [dvb_v5_fe_parms](#) * *parms*,
 const uint8_t * *buf*,
 struct [dvb_desc](#) * *desc*)

Initializes and parses the registration descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function initializes and makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Currently, no memory is allocated internally.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.32 dvb_desc_registration_print() `void dvb_desc_registration_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the registration descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.33 dvb_desc_sat_init() `int dvb_desc_sat_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `struct dvb_desc * desc)`

Initializes and parses the satellite delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function initializes and makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Currently, no memory is allocated internally.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.34 dvb_desc_sat_print() `void dvb_desc_sat_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the satellite delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.35 dvb_desc_service_free() void dvb_desc_service_free (
 struct [dvb_desc](#) * *desc*)

Frees all data allocated by the service descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.36 dvb_desc_service_init() int dvb_desc_service_init (
 struct [dvb_v5_fe_parms](#) * *parms*,
 const uint8_t * *buf*,
 struct [dvb_desc](#) * *desc*)

Initializes and parses the service descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.37 dvb_desc_service_print() void dvb_desc_service_print (
 struct [dvb_v5_fe_parms](#) * *parms*,
 const struct [dvb_desc](#) * *desc*)

Prints the content of the service descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.38 dvb_desc_t2_delivery_free() `void dvb_desc_t2_delivery_free (`
`const void * desc)`

Frees all data allocated by the T2 delivery system descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.39 dvb_desc_t2_delivery_init() `int dvb_desc_t2_delivery_init (`
`struct dvb_v5_fe_parms * parms,`
`const uint8_t * buf,`
`struct dvb_extension_descriptor * ext,`
`void * desc)`

Initializes and parses the T2 delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>ext</i>	struct dvb_extension_descriptor pointer
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.40 dvb_desc_t2_delivery_print() `void dvb_desc_t2_delivery_print (`
`struct dvb_v5_fe_parms * parms,`
`const struct dvb_extension_descriptor * ext,`
`const void * desc)`

Prints the content of the T2 delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>ext</i>	struct dvb_extension_descriptor pointer
<i>desc</i>	pointer to struct dvb_desc

7.7.3.41 dvb_desc_terrestrial_delivery_print() void dvb_desc_terrestrial_delivery_print (
 struct dvb_v5_fe_parms * parms,
 const struct dvb_desc * desc)

Prints the content of the DVB-T terrestrial delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.42 dvb_desc_ts_info_free() void dvb_desc_ts_info_free (
 struct dvb_desc * desc)

Frees all data allocated by the ISDB TS information descriptor.

descriptor

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.43 dvb_desc_ts_info_init() int dvb_desc_ts_info_init (
 struct dvb_v5_fe_parms * parms,
 const uint8_t * buf,
 struct dvb_desc * desc)

Initializes and parses the ISDB TS information descriptor.

descriptor

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.44 dvb_desc_ts_info_print() `void dvb_desc_ts_info_print (`
 `struct dvb_v5_fe_parms * parms,`
 `const struct dvb_desc * desc)`

Prints the content of the ISDB TS information descriptor.

descriptor

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.45 dvb_extension_descriptor_free() `void dvb_extension_descriptor_free (`
 `struct dvb_desc * desc)`

Frees all data allocated by the extended descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.46 dvb_extension_descriptor_init() `int dvb_extension_descriptor_init (`
 `struct dvb_v5_fe_parms * parms,`
 `const uint8_t * buf,`
 `struct dvb_desc * desc)`

Initializes and parses the extended descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.47 dvb_extension_descriptor_print() void dvb_extension_descriptor_print (
 struct dvb_v5_fe_parms * parms,
 const struct dvb_desc * desc)

Prints the content of the extended descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.48 isdb_desc_partial_reception_free() `void isdb_desc_partial_reception_free (struct dvb_desc * desc)`

Frees all data allocated by the ISDB-T partial reception descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.49 isdb_desc_partial_reception_init() `int isdb_desc_partial_reception_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, struct dvb_desc * desc)`

Initializes and parses the ISDB-T partial reception descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.50 isdb_desc_partial_reception_print() `void isdb_desc_partial_reception_print (struct dvb_v5_fe_parms * parms, const struct dvb_desc * desc)`

Prints the content of the ISDB-T partial reception descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.7.3.51 isdbt_desc_delivery_free() `void isdbt_desc_delivery_free (`
`struct dvb_desc * desc)`

Frees all data allocated by the ISDB-T terrestrial delivery system descriptor.

Parameters

<i>desc</i>	pointer to struct dvb_desc to be freed
-------------	--

7.7.3.52 isdbt_desc_delivery_init() `int isdbt_desc_delivery_init (`
`struct dvb_v5_fe_parms * parms,`
`const uint8_t * buf,`
`struct dvb_desc * desc)`

Initializes and parses the ISDB-T terrestrial delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function allocates a the descriptor and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

7.7.3.53 isdbt_desc_delivery_print() `void isdbt_desc_delivery_print (`
`struct dvb_v5_fe_parms * parms,`
`const struct dvb_desc * desc)`

Prints the content of the ISDB-T terrestrial delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>desc</i>	pointer to struct dvb_desc

7.8 Digital TV demux

Files

- file [dvb-demux.h](#)
Provides interfaces to deal with DVB demux.

Functions

- int [dvb_dmx_open](#) (int adapter, int demux)
Opens a DVB demux in read/write mode.
- void [dvb_dmx_close](#) (int dmx_fd)
Stops the DMX filter for the file descriptor and closes.
- void [dvb_dmx_stop](#) (int dmx_fd)
Stops the DMX filter for a given file descriptor.
- int [dvb_set_pesfilter](#) (int dmxfd, int pid, dmx_pes_type_t type, dmx_output_t output, int buffersize)
Start a filter for a MPEG-TS Packetized Elementary Stream (PES)
- int [dvb_set_section_filter](#) (int dmxfd, int pid, unsigned filtsize, unsigned char *filter, unsigned char *mask, unsigned char *mode, unsigned int flags)
Sets a MPEG-TS section filter.
- int [dvb_get_pmt_pid](#) (int dmxfd, int sid)
read the contents of the MPEG-TS PAT table, seeking for an specific service ID

7.8.1 Detailed Description

7.8.2 Function Documentation

7.8.2.1 [dvb_dmx_close\(\)](#) void [dvb_dmx_close](#) (
int *dmx_fd*)

Stops the DMX filter for the file descriptor and closes.

Parameters

<i>dmx</i> ↔ <i>_fd</i>	File descriptor to close
----------------------------	--------------------------

This is a wrapper function to [close\(\)](#).

Warning

Deprecated. Please use [dvb_dev_close\(\)](#) instead.

7.8.2.2 dvb_dmx_open() `int dvb_dmx_open (`
 `int adapter,`
 `int demux)`

Opens a DVB demux in read/write mode.

Parameters

<i>adapter</i>	DVB adapter number to open
<i>demux</i>	DVB demux number to open

This is a wrapper function to `open()`. File is always opened in blocking mode.

Returns

Returns a file descriptor on success, -1 otherwise.

Warning

Deprecated. Please use [dvb_dev_open\(\)](#) instead.

7.8.2.3 dvb_dmx_stop() `void dvb_dmx_stop (`
 `int dmxfid)`

Stops the DMX filter for a given file descriptor.

Parameters

<i>dmxfid</i>	File descriptor to close
---------------	--------------------------

This is a wrapper function to `DMX_STOP` ioctl. See <http://linuxtv.org/downloads/v4l-dvb-apis/dvb-demux.html> for more details.

Warning

Deprecated. Please use [dvb_dev_dmx_stop\(\)](#) instead.

7.8.2.4 dvb_get_pmt_pid() `int dvb_get_pmt_pid (`
 `int dmxfid,`
 `int sid)`

read the contents of the MPEG-TS PAT table, seeking for an specific service ID

Parameters

<i>dmxfd</i>	File descriptor for the demux device
<i>sid</i>	Session ID to seeking

Warning

Deprecated. Please use [dvb_get_pmt_pid\(\)](#) instead.

Returns

At return, it returns a negative value if error or the PID associated with the desired Session ID.

Warning

This function currently assumes that the PAT fits into one session.

7.8.2.5 dvb_set_pesfilter() `int dvb_set_pesfilter (`
 `int dmxfd,`
 `int pid,`
 `dmx_pes_type_t type,`
 `dmx_output_t output,`
 `int buffersize)`

Start a filter for a MPEG-TS Packetized Elementary Stream (PES)

Parameters

<i>dmxfd</i>	File descriptor for the demux device
<i>pid</i>	Program ID to filter. Use 0x2000 to select all PIDs
<i>type</i>	type of the PID (DMX_PES_VIDEO, DMX_PES_AUDIO, DMX_PES_OTHER, etc).
<i>output</i>	Where the data will be output (DMX_OUT_TS_TAP, DMX_OUT_DECODER, etc).
<i>buffersize</i>	Size of the buffer to be allocated to store the filtered data.

This is a wrapper function for DMX_SET_PES_FILTER ioctl. See <http://linuxtv.org/downloads/v4l-dvb-apis/dv-demux.html> for more details.

Returns

Retuns zero on success, -1 otherwise.

Warning

Deprecated. Please use [dvb_dev_dmx_set_pesfilter\(\)](#) instead.

```

7.8.2.6 dvb_set_section_filter() int dvb_set_section_filter (
    int dmxfid,
    int pid,
    unsigned filtsize,
    unsigned char * filter,
    unsigned char * mask,
    unsigned char * mode,
    unsigned int flags )

```

Sets a MPEG-TS section filter.

Parameters

<i>dmxfid</i>	File descriptor for the demux device
<i>pid</i>	Program ID to filter. Use 0x2000 to select all PIDs
<i>filtsize</i>	Size of the filter (up to 18 bytes)
<i>filter</i>	data to filter. Can be NULL or should have filtsize length
<i>mask</i>	filter mask. Can be NULL or should have filtsize length
<i>mode</i>	mode mask. Can be NULL or should have filtsize length
<i>flags</i>	flags for set filter (DMX_CHECK_CRC, DMX_ONESHOT, DMX_IMMEDIATE_START).

This is a wrapper function for DMX_SET_FILTER ioctl. See <http://linuxtv.org/downloads/v4l-dvb-apis/dvb-demux.html> for more details.

Warning

Deprecated. Please use [dvb_dev_dmx_set_pesfilter\(\)](#) instead.

Returns

Returns zero on success, -1 otherwise.

7.9 Channel and transponder file read/write

Files

- file [dvb-file.h](#)
Provides interfaces to deal with DVB channel and program files.

Data Structures

- struct [dvb_elementary_pid](#)
associates an elementary stream type with its PID
- struct [dvb_entry](#)
Represents one entry on a DTV file.
- struct [dvb_parse_table](#)
Describes the fields to parse on a file.
- struct [dvb_parse_struct](#)
Describes the format to parse an specific delivery system.

Enumerations

- enum [dvb_file_formats](#) {
 [FILE_UNKNOWN](#) , [FILE_ZAP](#) , [FILE_CHANNEL](#) , [FILE_DVBV5](#) ,
 [FILE_VDR](#) }

Known file formats.

Functions

- static void [dvb_file_free](#) (struct [dvb_file](#) *[dvb_file](#))
Deallocates memory associated with a struct [dvb_file](#).
- struct [dvb_file](#) * [dvb_read_file](#) (const char *[fname](#))
Read a file at libdvbv5 format.
- int [dvb_write_file](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#))
Write a file at libdvbv5 format.
- struct [dvb_file](#) * [dvb_read_file_format](#) (const char *[fname](#), uint32_t [delsys](#), enum [dvb_file_formats](#) [format](#))
Read a file on any format natively supported by the library.
- int [dvb_write_file_format](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#), uint32_t [delsys](#), enum [dvb_file_formats](#) [format](#))
Write a file on any format natively supported by the library.
- int [dvb_store_entry_prop](#) (struct [dvb_entry](#) *[entry](#), uint32_t [cmd](#), uint32_t [value](#))
Stores a key/value pair on a DVB file entry.
- int [dvb_retrieve_entry_prop](#) (struct [dvb_entry](#) *[entry](#), uint32_t [cmd](#), uint32_t *[value](#))
Retrieves the value associated with a key on a DVB file entry.
- int [dvb_store_channel](#) (struct [dvb_file](#) **[dvb_file](#), struct [dvb_v5_fe_parms](#) *[parms](#), struct [dvb_v5_descriptors](#) *[dvb_desc](#), int [get_detected](#), int [get_nit](#))
Stores a new scanned channel into a [dvb_file](#) struct
- int [dvb_parse_delsys](#) (const char *[name](#))
Ancillary function that seeks for a delivery system.
- enum [dvb_file_formats](#) [dvb_parse_format](#) (const char *[name](#))
Ancillary function that parses the name of a file format.
- struct [dvb_file](#) * [dvb_parse_format_online](#) (const char *[fname](#), uint32_t [delsys](#), const struct [dvb_parse_file](#) *[parse_file](#))
Read and parses a one line file format.
- int [dvb_write_format_online](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#), uint32_t [delsys](#), const struct [dvb_parse_file](#) *[parse_file](#))
Writes a file into an one line file format.
- int [dvb_write_format_vdr](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#))
Writes a file into vdr format (compatible up to version 2.1)

Variables

- const struct [dvb_parse_file](#) [channel_file_format](#)
File format definitions for dvb-apps channel format.
- const struct [dvb_parse_file](#) [channel_file_zap_format](#)
File format definitions for dvb-apps zap format.

7.9.1 Detailed Description

7.9.2 Enumeration Type Documentation

7.9.2.1 dvb_file_formats `enum dvb_file_formats`

Known file formats.

Please notice that the channel format defined here has a few optional fields that aren't part of the dvb-apps format, for DVB-S2 and for DVB-T2. They're there to match the formats found at dtv-scan-tables package up to September, 5 2014.

Enumerator

FILE_UNKNOWN	File format is unknown.
FILE_ZAP	File is at the dvb-apps "dvbzap" format.
FILE_CHANNEL	File is at the dvb-apps output format for dvb-zap.
FILE_DVBV5	File is at libdvbv5 format.
FILE_VDR	File is at DVR format (as supported on version 2.1.6). Note: this is only supported as an output format.

Definition at line 232 of file [dvb-file.h](#).

7.9.3 Function Documentation**7.9.3.1 dvb_file_free()** `static void dvb_file_free (struct dvb_file * dvb_file) [inline], [static]`

Deallocates memory associated with a struct [dvb_file](#).

Parameters

dvb_file	dvb_file struct to be deallocated
--------------------------	---

This function assumes that several functions were dynamically allocated by the library file functions.

Examples

[dvb-format-convert.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 255 of file [dvb-file.h](#).

References [dvb_entry::audio_pid](#), [dvb_entry::channel](#), [dvb_file::first_entry](#), [dvb_entry::lnb](#), [dvb_entry::location](#), [dvb_entry::next](#), [dvb_entry::other_el_pid](#), [dvb_entry::vchannel](#), and [dvb_entry::video_pid](#).

7.9.3.2 dvb_parse_delsys() `int dvb_parse_delsys (const char * name)`

Ancillary function that seeks for a delivery system.

Parameters

<i>name</i>	string containing the name of the Delivery System to seek
-------------	---

If the name is found, this function returns the DVBv5 property that corresponds to the string given. The function is case-insensitive, and it can check for alternate ways to write the name of a Delivery System. Currently, it supports: DVB-C, DVB-H, DVB-S, DVB-S2, DVB-T, DVB-T2, ISDB-C, ISDB-S, ISDB-T, ATSC-MH, DVBC/ANNEX_A, DVBC/ANNEX_B, DVBT, DSS, DVBS, DVBS2, DVBH, ISDBT, ISDBS, ISDBC, ATSC, ATSCMH, DTMB, CMMB, DAB, DVBT2, TURBO, DVBC/ANNEX_C. Please notice that this doesn't mean that all those standards are properly supported by the library.

Returns

Returns the Delivery System property number if success, -1 if error.

Examples

[dvb-fe-tool.c](#), and [dvb-format-convert.c](#).

7.9.3.3 dvb_parse_format() `enum dvb_file_formats dvb_parse_format (const char * name)`

Ancillary function that parses the name of a file format.

Parameters

<i>name</i>	string containing the name of the format Current valid names are: ZAP, CHANNEL, VDR and DVBV5. The name is case-insensitive.
-------------	--

Returns

It returns FILE_ZAP, FILE_CHANNEL, FILE_VDR or FILE_DVBV5 if the name was translated. FILE_UNKNOWN otherwise.

Examples

[dvb-format-convert.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

7.9.3.4 dvb_parse_format_online() `struct dvb_file * dvb_parse_format_online (const char * fname, uint32_t delsys, const struct dvb_parse_file * parse_file)`

Read and parses a one line file format.

Parameters

<i>fname</i>	file name
<i>delsys</i>	delivery system
<i>parse_file</i>	pointer struct dvb_parse_file

Returns

It a pointer to struct [dvb_file](#) on success, NULL otherwise.

This function is called internally by `dvb_read_file_format`.

7.9.3.5 `dvb_read_file()` `struct dvb_file * dvb_read_file (`
`const char * fname)`

Read a file at libdvbv5 format.

Parameters

<i>fname</i>	file name
--------------	-----------

Returns

It returns a pointer to struct [dvb_file](#) describing the entries that were read from the file. If it fails, NULL is returned.

7.9.3.6 `dvb_read_file_format()` `struct dvb_file * dvb_read_file_format (`
`const char * fname,`
`uint32_t delsys,`
`enum dvb_file_formats format)`

Read a file on any format natively supported by the library.

Parameters

<i>fname</i>	file name
<i>delsys</i>	Delivery system, as specified by enum <code>fe_delivery_system</code>
<i>format</i>	Name of the format to be read

Returns

It returns a pointer to struct [dvb_file](#) describing the entries that were read from the file. If it fails, NULL is returned.

Examples

[dvb-format-convert.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

```

7.9.3.7 dvb_retrieve_entry_prop() int dvb_retrieve_entry_prop (
    struct dvb\_entry * entry,
    uint32_t cmd,
    uint32_t * value )

```

Retrieves the value associated with a key on a DVB file entry.

Parameters

<i>entry</i>	entry to be used
<i>cmd</i>	key for the property to be found. It be one of the DVBv5 properties, plus the libdvbv5 ones, as defined at dvb-v5-std.h
<i>value</i>	pointer to store the value associated with the property.

This function seeks for a property with the name specified by *cmd* and fills *value* with its contents.

Returns

Returns 0 if success, or, -1 if the entry doesn't exist.

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

```

7.9.3.8 dvb_store_channel() int dvb_store_channel (
    struct dvb\_file ** dvb_file,
    struct dvb\_v5\_fe\_parms * parms,
    struct dvb\_v5\_descriptors * dvb_desc,
    int get_detected,
    int get_nit )

```

stored a new scanned channel into a [dvb_file](#) struct

Parameters

<i>dvb_file</i>	file struct to be filled
<i>parms</i>	struct dvb_v5_fe_parms used by libdvbv5 frontend
<i>dvb_desc</i>	struct dvb_desc as described at descriptors.h , filled with the descriptors associated with a DVB channel. those descriptors can be filled by calling one of the scan functions defined at dvb-sat.h .
<i>get_detected</i>	if different than zero, uses the frontend parameters obtained from the device driver (such as modulation, FEC, etc)
<i>get_nit</i>	if true, uses the parameters obtained from the MPEG-TS NIT table to add newly detected transponders.

This function should be used to store the services found on a scanned transponder. Initially, it copies the same parameters used to set the frontend, that came from a file where the Service ID and Elementary Stream PIDs are unknown. At tuning time, it is common to set the device to tune on auto-detection mode (e. g. using QAM/AUTO, for example, to autodetect the QAM modulation). The libdvbv5's logic will be to check the detected values. So, the modulation might, for example, have changed to QAM/256. In such case, if *get_detected* is 0, it will store QAM/↵ AUTO at the struct. If *get_detected* is different than zero, it will store QAM/256. If *get_nit* is different than zero, and if

the MPEG-TS has info about other physical channels/transponders, this function will add newer entries to [dvb_file](#), for it to seek for new transponders. This is very useful especially for DVB-C, where all transponders belong to the same operator. Knowing one frequency is generally enough to get all DVB-C transponders.

Returns

Returns 0 if success, or, -1 if error.

Examples

[dvbv5-scan.c](#).

```
7.9.3.9 dvb_store_entry_prop() int dvb_store_entry_prop (
    struct dvb\_entry * entry,
    uint32_t cmd,
    uint32_t value )
```

Stores a key/value pair on a DVB file entry.

Parameters

<i>entry</i>	entry to be filled
<i>cmd</i>	key for the property to be used. It be one of the DVBv5 properties, plus the libdvbv5 ones, as defined at dvb-v5-std.h
<i>value</i>	value for the property.

This function seeks for a property with the name specified by cmd and fills it with value. If the entry doesn't exist, it creates a new key.

Returns

Returns 0 if success, or, if the entry has already DTV_MAX_COMMAND properties, it returns -1.

```
7.9.3.10 dvb_write_file() int dvb_write_file (
    const char * fname,
    struct dvb\_file * dvb_file )
```

Write a file at libdvbv5 format.

Parameters

<i>fname</i>	file name
<i>dvb_file</i>	contents of the file to be written

Returns

It returns zero if success, or a positive error number if it fails.

7.9.3.11 `dvb_write_file_format()` `int dvb_write_file_format (`
 `const char * fname,`
 `struct dvb_file * dvb_file,`
 `uint32_t delsys,`
 `enum dvb_file_formats format)`

Write a file on any format natively supported by the library.

Parameters

<i>fname</i>	file name
dvb_file	contents of the file to be written
<i>delsys</i>	Delivery system, as specified by enum <code>fe_delivery_system</code>
<i>format</i>	Name of the format to be read

Returns

It a pointer to struct [dvb_file](#) on success, NULL otherwise.

Examples

[dvb-format-convert.c](#), and [dvbv5-scan.c](#).

7.9.3.12 `dvb_write_format_online()` `int dvb_write_format_online (`
 `const char * fname,`
 `struct dvb_file * dvb_file,`
 `uint32_t delsys,`
 `const struct dvb_parse_file * parse_file)`

Writes a file into an one line file format.

Parameters

<i>fname</i>	file name
dvb_file	contents of the file to be written
<i>delsys</i>	delivery system
<i>parse_file</i>	pointer struct dvb_parse_file

Returns

It returns zero if success, or a positive error number if it fails.

This function is called internally by `dvb_write_file_format`.

7.9.3.13 dvb_write_format_vdr() `int dvb_write_format_vdr (`
`const char * fname,`
`struct dvb_file * dvb_file)`

Writes a file into vdr format (compatible up to version 2.1)

Parameters

<i>fname</i>	file name
<i>dvb_file</i>	contents of the file to be written

Returns

It returns zero if success, or a positive error number if it fails.

This function is called internally by dvb_write_file_format.

7.9.4 Variable Documentation

7.9.4.1 channel_file_format `const struct dvb_parse_file channel_file_format [extern]`

File format definitions for dvb-apps channel format.

7.9.4.2 channel_file_zap_format `const struct dvb_parse_file channel_file_zap_format [extern]`

File format definitions for dvb-apps zap format.

8 Data Structure Documentation

8.1 atsc_desc_service_location Struct Reference

Describes the elementary streams inside a PAT table for ATSC.

```
#include <desc_atsc_service_location.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `struct atsc_desc_service_location_elementary * elementary`
- `union {`
 - `uint16_t bitfield`
 - `struct {`
 - `uint16_t pcr_pid:13`
 - `uint16_t reserved:3`
 - `}`
- `};`
- `uint8_t number_elements`

8.1.1 Detailed Description

Describes the elementary streams inside a PAT table for ATSC.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>elementary</i>	pointer to struct atsc_desc_service_location_elementary
<i>pcr_pid</i>	PCR pid
<i>number_elements</i>	number elements

Definition at line 75 of file [desc_atsc_service_location.h](#).

8.1.2 Field Documentation

8.1.2.1 `union { ... } atsc_desc_service_location::@131`

8.1.2.2 bitfield `uint16_t atsc_desc_service_location::bitfield`

Definition at line 83 of file [desc_atsc_service_location.h](#).

8.1.2.3 elementary `struct atsc_desc_service_location_elementary* atsc_desc_service_location↔
::elementary`

Definition at line 80 of file [desc_atsc_service_location.h](#).

8.1.2.4 length `uint8_t atsc_desc_service_location::length`

Definition at line 77 of file [desc_atsc_service_location.h](#).

8.1.2.5 next `struct dvb_desc* atsc_desc_service_location::next`

Definition at line 78 of file [desc_atsc_service_location.h](#).

8.1.2.6 number_elements `uint8_t atsc_desc_service_location::number_elements`

Definition at line 90 of file [desc_atsc_service_location.h](#).

8.1.2.7 pcr_pid `uint16_t atsc_desc_service_location::pcr_pid`

Definition at line 85 of file [desc_atsc_service_location.h](#).

8.1.2.8 reserved `uint16_t atsc_desc_service_location::reserved`

Definition at line 86 of file [desc_atsc_service_location.h](#).

8.1.2.9 type `uint8_t atsc_desc_service_location::type`

Definition at line 76 of file [desc_atsc_service_location.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_atsc_service_location.h](#)

8.2 atsc_desc_service_location_elementary Struct Reference

service location elementary descriptors

```
#include <desc_atsc_service_location.h>
```

Data Fields

- `uint8_t stream_type`
- `union {`
 - `uint16_t bitfield`
`struct {`
 - `uint16_t elementary_pid:13`
 - `uint16_t reserved:3``}``};`
- `unsigned char ISO_639_language_code [3]`

8.2.1 Detailed Description

service location elementary descriptors

Parameters

<i>stream_type</i>	stream type
<i>elementary_pid</i>	elementary pid
<i>ISO_639_language_code</i>	ISO 639 language code

Definition at line 51 of file [desc_atsc_service_location.h](#).

8.2.2 Field Documentation

8.2.2.1 `union { ... } atsc_desc_service_location_elementary::@127`

8.2.2.2 bitfield `uint16_t atsc_desc_service_location_elementary::bitfield`

Definition at line 54 of file [desc_atsc_service_location.h](#).

8.2.2.3 elementary_pid `uint16_t atsc_desc_service_location_elementary::elementary_pid`

Definition at line 56 of file [desc_atsc_service_location.h](#).

8.2.2.4 ISO_639_language_code `unsigned char atsc_desc_service_location_elementary::ISO_639_↵
language_code[3]`

Definition at line 60 of file [desc_atsc_service_location.h](#).

8.2.2.5 reserved `uint16_t atsc_desc_service_location_elementary::reserved`

Definition at line 57 of file [desc_atsc_service_location.h](#).

8.2.2.6 stream_type `uint8_t atsc_desc_service_location_elementary::stream_type`

Definition at line 52 of file [desc_atsc_service_location.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_atsc_service_location.h](#)

8.3 atsc_table_eit Struct Reference

ATSC EIT table.

```
#include <atsc_eit.h>
```

Data Fields

- struct [dvb_table_header](#) header
- uint8_t [protocol_version](#)
- uint8_t [events](#)
- struct [atsc_table_eit_event](#) * event

8.3.1 Detailed Description

ATSC EIT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>protocol_version</i>	protocol version
<i>events</i>	events
<i>event</i>	pointer to struct atsc_table_eit_event

This structure is used to store the original ATSC EIT table, converting the integer fields to the CPU endianness.

Everything after [atsc_table_eit::event](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 146 of file [atsc_eit.h](#).

8.3.2 Field Documentation

8.3.2.1 event struct [atsc_table_eit_event](#)* atsc_table_eit::event

Definition at line 150 of file [atsc_eit.h](#).

8.3.2.2 events uint8_t atsc_table_eit::events

Definition at line 149 of file [atsc_eit.h](#).

8.3.2.3 header `struct dvb_table_header atsc_table_eit::header`

Definition at line 147 of file [atsc_eit.h](#).

8.3.2.4 protocol_version `uint8_t atsc_table_eit::protocol_version`

Definition at line 148 of file [atsc_eit.h](#).

The documentation for this struct was generated from the following file:

- `lib/include/libdvbv5/atsc_eit.h`

8.4 atsc_table_eit_desc_length Union Reference

ATSC EIT descriptor length.

```
#include <atsc_eit.h>
```

Data Fields

- `uint16_t bitfield`
- `struct {`
 `uint16_t desc_length:12`
 `uint16_t reserved:4`
 `};`

8.4.1 Detailed Description

ATSC EIT descriptor length.

Parameters

<i>desc_length</i>	descriptor length
--------------------	-------------------

This structure is used to store the original ATSC EIT event table, converting the integer fields to the CPU endianness, and converting the timestamps to a way that it is better handled on Linux.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Definition at line 121 of file [atsc_eit.h](#).

8.4.2 Field Documentation

8.4.2.1 `struct { ... } atsc_table_eit_desc_length::@19`

8.4.2.2 bitfield `uint16_t atsc_table_eit_desc_length::bitfield`

Definition at line 122 of file [atsc_eit.h](#).

8.4.2.3 desc_length `uint16_t atsc_table_eit_desc_length::desc_length`

Definition at line 124 of file [atsc_eit.h](#).

8.4.2.4 reserved `uint16_t atsc_table_eit_desc_length::reserved`

Definition at line 125 of file [atsc_eit.h](#).

The documentation for this union was generated from the following file:

- [lib/include/libdvbv5/atsc_eit.h](#)

8.5 atsc_table_eit_event Struct Reference

ATSC EIT event table.

```
#include <atsc_eit.h>
```

Data Fields

- union {
 - `uint16_t bitfield`
 - struct {
 - `uint16_t event_id:14`
 - `uint16_t one:2`
- };
- `uint32_t start_time`
- union {
 - `uint32_t bitfield2`
 - struct {
 - `uint32_t title_length:8`
 - `uint32_t duration:20`
 - `uint32_t etm:2`
 - `uint32_t one2:2`
 - `uint32_t __pad0__:2`
- };
- struct `dvb_desc` * `descriptor`
- struct `atsc_table_eit_event` * `next`
- struct tm `start`
- `uint16_t source_id`

8.5.1 Detailed Description

ATSC EIT event table.

Parameters

<i>event_id</i>	an uniquely (inside a service ID) event ID
<i>title_length</i>	title length. Zero means no title
<i>duration</i>	duration in seconds
<i>etm</i>	Extended Text Message location
<i>descriptor</i>	pointer to struct dvb_desc
<i>next</i>	pointer to struct atsc_table_eit_event
<i>start</i>	event start (in struct tm format)
<i>source_id</i>	source id (obtained from ATSC header)

This structure is used to store the original ATSC EIT event table, converting the integer fields to the CPU endianness, and converting the timestamps to a way that it is better handled on Linux.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [atsc_table_eit_event::descriptor](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 81 of file [atsc_eit.h](#).

8.5.2 Field Documentation

8.5.2.1 `union { ... } atsc_table_eit_event::@11`

8.5.2.2 `union { ... } atsc_table_eit_event::@13`

8.5.2.3 `__pad0__ uint32_t atsc_table_eit_event::__pad0__`

Definition at line 97 of file [atsc_eit.h](#).

8.5.2.4 **bitfield** `uint16_t atsc_table_eit_event::bitfield`

Definition at line 83 of file [atsc_eit.h](#).

8.5.2.5 bitfield2 `uint32_t atsc_table_eit_event::bitfield2`

Definition at line 91 of file [atsc_eit.h](#).

8.5.2.6 descriptor `struct dvb_desc* atsc_table_eit_event::descriptor`

Definition at line 100 of file [atsc_eit.h](#).

8.5.2.7 duration `uint32_t atsc_table_eit_event::duration`

Definition at line 94 of file [atsc_eit.h](#).

8.5.2.8 etm `uint32_t atsc_table_eit_event::etm`

Definition at line 95 of file [atsc_eit.h](#).

8.5.2.9 event_id `uint16_t atsc_table_eit_event::event_id`

Definition at line 85 of file [atsc_eit.h](#).

8.5.2.10 next `struct atsc_table_eit_event* atsc_table_eit_event::next`

Definition at line 101 of file [atsc_eit.h](#).

8.5.2.11 one `uint16_t atsc_table_eit_event::one`

Definition at line 86 of file [atsc_eit.h](#).

8.5.2.12 one2 `uint32_t atsc_table_eit_event::one2`

Definition at line 96 of file [atsc_eit.h](#).

8.5.2.13 source_id `uint16_t atsc_table_eit_event::source_id`

Definition at line 103 of file [atsc_eit.h](#).

8.5.2.14 start `struct tm atsc_table_eit_event::start`

Definition at line 102 of file [atsc_eit.h](#).

8.5.2.15 start_time `uint32_t atsc_table_eit_event::start_time`

Definition at line 89 of file [atsc_eit.h](#).

8.5.2.16 title_length `uint32_t atsc_table_eit_event::title_length`

Definition at line 93 of file [atsc_eit.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/atsc_eit.h](#)

8.6 atsc_table_mgt Struct Reference

ATSC MGT table.

```
#include <mgt.h>
```

Data Fields

- struct [dvb_table_header](#) `header`
- `uint8_t` `protocol_version`
- `uint16_t` `tables`
- struct [atsc_table_mgt_table](#) * `table`
- struct [dvb_desc](#) * `descriptor`

8.6.1 Detailed Description

ATSC MGT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>protocol_version</i>	protocol version
<i>tables</i>	tables_defined Number of tables defined
<i>table</i>	pointer to struct atsc_table_mgt_table
<i>descriptor</i>	pointer to struct dvb_desc

This structure is used to store the original MGT channel table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [atsc_table_mgt::table](#) (including it) won't be bit-mapped * to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 122 of file [mgt.h](#).

8.6.2 Field Documentation

8.6.2.1 descriptor `struct dvb_desc* atsc_table_mgt::descriptor`

Definition at line 127 of file [mgt.h](#).

8.6.2.2 header `struct dvb_table_header atsc_table_mgt::header`

Definition at line 123 of file [mgt.h](#).

8.6.2.3 protocol_version `uint8_t atsc_table_mgt::protocol_version`

Definition at line 124 of file [mgt.h](#).

8.6.2.4 table `struct atsc_table_mgt_table* atsc_table_mgt::table`

Definition at line 126 of file [mgt.h](#).

8.6.2.5 tables `uint16_t atsc_table_mgt::tables`

Definition at line 125 of file [mgt.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mgt.h](#)

8.7 atsc_table_mgt_table Struct Reference

ATSC tables description at MGT table.

```
#include <mgt.h>
```

Data Fields

- `uint16_t type`
- `union {`
 - `uint16_t bitfield`
 - `struct {`
 - `uint16_t pid:13`
 - `uint16_t one:3`
 - `}`
- `};`
- `uint8_t type_version:5`
- `uint8_t one2:3`
- `uint32_t size`
- `union {`
 - `uint16_t bitfield2`
 - `struct {`
 - `uint16_t desc_length:12`
 - `uint16_t one3:4`
 - `}`
- `};`
- `struct dvb_desc * descriptor`
- `struct atsc_table_mgt_table * next`

8.7.1 Detailed Description

ATSC tables description at MGT table.

Parameters

<i>type</i>	table type
<i>pid</i>	table type pid
<i>type_version</i>	type type version number
<i>size</i>	number of bytes for the table entry
<i>desc_length</i>	table type descriptors length
<i>descriptor</i>	pointer to struct dvb_desc
<i>next</i>	pointer to struct atsc_table_mgt_table

This structure is used to store the original VCT channel table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [atsc_table_mgt_table::descriptor](#) (including it) won't be bit-mapped * to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 77 of file [mgt.h](#).

8.7.2 Field Documentation

8.7.2.1 `union { ... } atsc_table_mgt_table::@27`

8.7.2.2 `union { ... } atsc_table_mgt_table::@29`

8.7.2.3 bitfield `uint16_t atsc_table_mgt_table::bitfield`

Definition at line 80 of file [mgt.h](#).

8.7.2.4 bitfield2 `uint16_t atsc_table_mgt_table::bitfield2`

Definition at line 90 of file [mgt.h](#).

8.7.2.5 desc_length `uint16_t atsc_table_mgt_table::desc_length`

Definition at line 92 of file [mgt.h](#).

8.7.2.6 descriptor `struct dvb_desc* atsc_table_mgt_table::descriptor`

Definition at line 96 of file [mgt.h](#).

8.7.2.7 next `struct atsc_table_mgt_table* atsc_table_mgt_table::next`

Definition at line 97 of file [mgt.h](#).

8.7.2.8 one `uint16_t atsc_table_mgt_table::one`

Definition at line 83 of file [mgt.h](#).

8.7.2.9 one2 `uint8_t atsc_table_mgt_table::one2`

Definition at line 87 of file [mgt.h](#).

8.7.2.10 one3 `uint16_t atsc_table_mgt_table::one3`

Definition at line 93 of file [mgt.h](#).

8.7.2.11 pid `uint16_t atsc_table_mgt_table::pid`

Definition at line 82 of file [mgt.h](#).

8.7.2.12 size `uint32_t atsc_table_mgt_table::size`

Definition at line 88 of file [mgt.h](#).

8.7.2.13 type `uint16_t atsc_table_mgt_table::type`

Definition at line 78 of file [mgt.h](#).

8.7.2.14 type_version `uint8_t atsc_table_mgt_table::type_version`

Definition at line 86 of file [mgt.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mgt.h](#)

8.8 atsc_table_vct Struct Reference

ATSC VCT table (covers both CVCT and TVCT)

```
#include <vct.h>
```

Data Fields

- struct [dvb_table_header](#) header
- `uint8_t` protocol_version
- `uint8_t` num_channels_in_section
- struct [atsc_table_vct_channel](#) * channel
- struct [dvb_desc](#) * descriptor

8.8.1 Detailed Description

ATSC VCT table (covers both CVCT and TVCT)

Parameters

<i>header</i>	struct dvb_table_header content
<i>protocol_version</i>	protocol version
<i>num_channels_in_section</i>	num channels in section
<i>channel</i>	pointer to struct channel
<i>descriptor</i>	pointer to struct descriptor

Everything after [atsc_table_vct::channel](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there

Definition at line [168](#) of file [vct.h](#).

8.8.2 Field Documentation

8.8.2.1 channel struct [atsc_table_vct_channel*](#) atsc_table_vct::channel

Definition at line [174](#) of file [vct.h](#).

8.8.2.2 descriptor struct [dvb_desc*](#) atsc_table_vct::descriptor

Definition at line [175](#) of file [vct.h](#).

8.8.2.3 header struct [dvb_table_header](#) atsc_table_vct::header

Definition at line [169](#) of file [vct.h](#).

8.8.2.4 num_channels_in_section uint8_t atsc_table_vct::num_channels_in_section

Definition at line [172](#) of file [vct.h](#).

8.8.2.5 protocol_version uint8_t atsc_table_vct::protocol_version

Definition at line [170](#) of file [vct.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/vct.h](#)

8.9 atsc_table_vct_channel Struct Reference

ATSC VCT channel table (covers both CVCT and TVCT)

```
#include <vct.h>
```

Data Fields

- uint16_t [__short_name](#) [7]
- union {
 - uint32_t [bitfield1](#)
 - struct {
 - uint32_t [modulation_mode](#):8
 - uint32_t [minor_channel_number](#):10
 - uint32_t [major_channel_number](#):10
 - uint32_t [reserved1](#):4
- };
- uint32_t [carrier_frequency](#)
- uint16_t [channel_tsid](#)
- uint16_t [program_number](#)
- union {
 - uint16_t [bitfield2](#)
 - struct {
 - uint16_t [service_type](#):6
 - uint16_t [reserved2](#):3
 - uint16_t [hide_guide](#):1
 - uint16_t [out_of_band](#):1
 - uint16_t [path_select](#):1
 - uint16_t [hidden](#):1
 - uint16_t [access_controlled](#):1
 - uint16_t [ETM_location](#):2
- };
- uint16_t [source_id](#)
- union {
 - uint16_t [bitfield3](#)
 - struct {
 - uint16_t [descriptors_length](#):10
 - uint16_t [reserved3](#):6
- };
- struct [dvb_desc](#) * [descriptor](#)
- struct [atsc_table_vct_channel](#) * [next](#)
- char [short_name](#) [32]

8.9.1 Detailed Description

ATSC VCT channel table (covers both CVCT and TVCT)

Parameters

<i>modulation_mode</i>	modulation mode
<i>minor_channel_number</i>	minor channel number
<i>major_channel_number</i>	major channel number
<i>carrier_frequency</i>	carrier frequency
<i>channel_tsid</i>	channel tsid
<i>program_number</i>	program number
<i>service_type</i>	service type
<i>hide_guide</i>	hide guide
<i>out_of_band</i>	out of band (CVCT only)
<i>path_select</i>	path select (CVCT only)
<i>hidden</i>	hidden
<i>access_controlled</i>	access controlled
<i>ETM_location</i>	ETM location
<i>source_id</i>	source ID
<i>descriptors_length</i>	length of the descriptors
<i>descriptor</i>	pointer to struct dvb_desc
<i>next</i>	pointer to another struct atsc_table_vct_channel
<i>descriptors_length</i>	length of the descriptors
<i>short_name</i>	short name. The __short_name is converted from UTF-16 to locale charset when parsed

This structure is used to store the original VCT channel table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [atsc_table_vct_channel::descriptor](#) (including it) won't be bit-mapped * to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 101 of file [vct.h](#).

8.9.2 Field Documentation

8.9.2.1 `union { ... } atsc_table_vct_channel::@69`

8.9.2.2 `union { ... } atsc_table_vct_channel::@71`

8.9.2.3 `union { ... } atsc_table_vct_channel::@73`

8.9.2.4 __short_name `uint16_t atsc_table_vct_channel::__short_name[7]`

Definition at line 102 of file [vct.h](#).

8.9.2.5 access_controlled `uint16_t atsc_table_vct_channel::access_controlled`

Definition at line 126 of file [vct.h](#).

8.9.2.6 bitfield1 `uint32_t atsc_table_vct_channel::bitfield1`

Definition at line 105 of file [vct.h](#).

8.9.2.7 bitfield2 `uint16_t atsc_table_vct_channel::bitfield2`

Definition at line 118 of file [vct.h](#).

8.9.2.8 bitfield3 `uint16_t atsc_table_vct_channel::bitfield3`

Definition at line 134 of file [vct.h](#).

8.9.2.9 carrier_frequency `uint32_t atsc_table_vct_channel::carrier_frequency`

Definition at line 114 of file [vct.h](#).

8.9.2.10 channel_tsid `uint16_t atsc_table_vct_channel::channel_tsid`

Definition at line 115 of file [vct.h](#).

8.9.2.11 descriptor `struct dvb_desc* atsc_table_vct_channel::descriptor`

Definition at line 146 of file [vct.h](#).

8.9.2.12 descriptors_length `uint16_t atsc_table_vct_channel::descriptors_length`

Definition at line 136 of file [vct.h](#).

8.9.2.13 ETM_location `uint16_t atsc_table_vct_channel::ETM_location`

Definition at line 127 of file [vct.h](#).

8.9.2.14 hidden `uint16_t atsc_table_vct_channel::hidden`

Definition at line 125 of file [vct.h](#).

8.9.2.15 hide_guide `uint16_t atsc_table_vct_channel::hide_guide`

Definition at line 122 of file [vct.h](#).

8.9.2.16 major_channel_number `uint32_t atsc_table_vct_channel::major_channel_number`

Definition at line 109 of file [vct.h](#).

8.9.2.17 minor_channel_number `uint32_t atsc_table_vct_channel::minor_channel_number`

Definition at line 108 of file [vct.h](#).

8.9.2.18 modulation_mode `uint32_t atsc_table_vct_channel::modulation_mode`

Definition at line 107 of file [vct.h](#).

8.9.2.19 next `struct atsc_table_vct_channel* atsc_table_vct_channel::next`

Definition at line 147 of file [vct.h](#).

8.9.2.20 out_of_band `uint16_t atsc_table_vct_channel::out_of_band`

Definition at line 123 of file [vct.h](#).

8.9.2.21 path_select `uint16_t atsc_table_vct_channel::path_select`

Definition at line 124 of file [vct.h](#).

8.9.2.22 program_number `uint16_t atsc_table_vct_channel::program_number`

Definition at line 116 of file [vct.h](#).

8.9.2.23 reserved1 `uint32_t atsc_table_vct_channel::reserved1`

Definition at line 110 of file [vct.h](#).

8.9.2.24 reserved2 `uint16_t atsc_table_vct_channel::reserved2`

Definition at line 121 of file [vct.h](#).

8.9.2.25 reserved3 `uint16_t atsc_table_vct_channel::reserved3`

Definition at line 137 of file [vct.h](#).

8.9.2.26 service_type `uint16_t atsc_table_vct_channel::service_type`

Definition at line 120 of file [vct.h](#).

8.9.2.27 short_name `char atsc_table_vct_channel::short_name[32]`

Definition at line 151 of file [vct.h](#).

8.9.2.28 source_id uint16_t atsc_table_vct_channel::source_id

Definition at line 132 of file [vct.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[vct.h](#)

8.10 atsc_table_vct_descriptor_length Union Reference

ATSC VCT descriptor length.

```
#include <vct.h>
```

Data Fields

- uint16_t [bitfield](#)
 - struct {
 - uint16_t [descriptor_length](#):10
 - uint16_t [reserved](#):6
- ```
};
```

### 8.10.1 Detailed Description

ATSC VCT descriptor length.

#### Parameters

|                          |                   |
|--------------------------|-------------------|
| <i>descriptor_length</i> | descriptor length |
|--------------------------|-------------------|

Used internally by the library to parse the descriptor length endianness.

Definition at line 187 of file [vct.h](#).

### 8.10.2 Field Documentation

**8.10.2.1** struct { ... } atsc\_table\_vct\_descriptor\_length::@81

**8.10.2.2 bitfield** uint16\_t atsc\_table\_vct\_descriptor\_length::bitfield

Definition at line 188 of file [vct.h](#).

### 8.10.2.3 **descriptor\_length** `uint16_t atsc_table_vct_descriptor_length::descriptor_length`

Definition at line 190 of file [vct.h](#).

### 8.10.2.4 **reserved** `uint16_t atsc_table_vct_descriptor_length::reserved`

Definition at line 191 of file [vct.h](#).

The documentation for this union was generated from the following file:

- [lib/include/libdvbv5/vct.h](#)

## 8.11 **dvb\_desc Struct Reference**

Linked list containing the several descriptors found on a MPEG-TS table.

```
#include <descriptors.h>
```

### Data Fields

- `uint8_t` [type](#)
- `uint8_t` [length](#)
- `struct dvb\_desc *` [next](#)
- `uint8_t` [data](#) []

#### 8.11.1 Detailed Description

Linked list containing the several descriptors found on a MPEG-TS table.

##### Parameters

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>type</i>   | Descriptor type                                    |
| <i>length</i> | Length of the descriptor                           |
| <i>next</i>   | pointer to the <a href="#">dvb_desc</a> descriptor |
| <i>data</i>   | Descriptor data                                    |

Definition at line 117 of file [descriptors.h](#).

#### 8.11.2 Field Documentation

##### 8.11.2.1 **data** `uint8_t dvb_desc::data[]`

Definition at line 122 of file [descriptors.h](#).

**8.11.2.2 length** `uint8_t dvb_desc::length`

Definition at line 119 of file [descriptors.h](#).

**8.11.2.3 next** `struct dvb_desc* dvb_desc::next`

Definition at line 120 of file [descriptors.h](#).

**8.11.2.4 type** `uint8_t dvb_desc::type`

Definition at line 118 of file [descriptors.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/descriptors.h](#)

## 8.12 dvb\_desc\_ca Struct Reference

Contains the private data for Conditional Access.

```
#include <desc_ca.h>
```

### Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint16_t ca_id`
- `union {  
    uint16_t bitfield1  
    struct {  
        uint16_t ca_pid:13  
        uint16_t reserved:3  
    }  
};`
- `uint8_t * privdata`
- `uint8_t privdata_len`

### 8.12.1 Detailed Description

Contains the private data for Conditional Access.

**Parameters**

|                     |                                            |
|---------------------|--------------------------------------------|
| <i>type</i>         | descriptor tag                             |
| <i>length</i>       | descriptor length                          |
| <i>next</i>         | pointer to struct <a href="#">dvb_desc</a> |
| <i>ca_id</i>        | Conditional Access ID                      |
| <i>ca_pid</i>       | Conditional Access ID                      |
| <i>privdata</i>     | pointer to private data buffer             |
| <i>privdata_len</i> | length of the private data                 |

Definition at line [56](#) of file [desc\\_ca.h](#).

**8.12.2 Field Documentation**

**8.12.2.1**    `union { ... } dvb_desc_ca::@135`

**8.12.2.2 bitfield1**    `uint16_t dvb_desc_ca::bitfield1`

Definition at line [63](#) of file [desc\\_ca.h](#).

**8.12.2.3 ca\_id**    `uint16_t dvb_desc_ca::ca_id`

Definition at line [61](#) of file [desc\\_ca.h](#).

**8.12.2.4 ca\_pid**    `uint16_t dvb_desc_ca::ca_pid`

Definition at line [65](#) of file [desc\\_ca.h](#).

**8.12.2.5 length**    `uint8_t dvb_desc_ca::length`

Definition at line [58](#) of file [desc\\_ca.h](#).

**8.12.2.6 next** `struct dvb_desc* dvb_desc_ca::next`

Definition at line 59 of file [desc\\_ca.h](#).

**8.12.2.7 privdata** `uint8_t* dvb_desc_ca::privdata`

Definition at line 70 of file [desc\\_ca.h](#).

**8.12.2.8 privdata\_len** `uint8_t dvb_desc_ca::privdata_len`

Definition at line 71 of file [desc\\_ca.h](#).

**8.12.2.9 reserved** `uint16_t dvb_desc_ca::reserved`

Definition at line 66 of file [desc\\_ca.h](#).

**8.12.2.10 type** `uint8_t dvb_desc_ca::type`

Definition at line 57 of file [desc\\_ca.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc\\_ca.h](#)

## 8.13 dvb\_desc\_ca\_identifier Struct Reference

Indicates if a particular bouquet, service or event is associated with a CA system.

```
#include <desc_ca_identifier.h>
```

### Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint8_t caid_count`
- `uint16_t * caids`

### 8.13.1 Detailed Description

Indicates if a particular bouquet, service or event is associated with a CA system.



**Parameters**

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>type</i>       | descriptor tag                             |
| <i>length</i>     | descriptor length                          |
| <i>next</i>       | pointer to struct <a href="#">dvb_desc</a> |
| <i>caid_count</i> | Number of CA IDs                           |
| <i>caids</i>      | CA Identifier IDs                          |

Definition at line [53](#) of file [desc\\_ca\\_identifier.h](#).

**8.13.2 Field Documentation**

**8.13.2.1 caid\_count** `uint8_t dvb_desc_ca_identifier::caid_count`

Definition at line [58](#) of file [desc\\_ca\\_identifier.h](#).

**8.13.2.2 caids** `uint16_t* dvb_desc_ca_identifier::caids`

Definition at line [59](#) of file [desc\\_ca\\_identifier.h](#).

**8.13.2.3 length** `uint8_t dvb_desc_ca_identifier::length`

Definition at line [55](#) of file [desc\\_ca\\_identifier.h](#).

**8.13.2.4 next** `struct dvb\_desc* dvb_desc_ca_identifier::next`

Definition at line [56](#) of file [desc\\_ca\\_identifier.h](#).

**8.13.2.5 type** `uint8_t dvb_desc_ca_identifier::type`

Definition at line [54](#) of file [desc\\_ca\\_identifier.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc\\_ca\\_identifier.h](#)

## 8.14 dvb\_desc\_cable\_delivery Struct Reference

Structure containing the cable delivery system descriptor.

```
#include <desc_cable_delivery.h>
```

### Data Fields

- uint8\_t [type](#)
  - uint8\_t [length](#)
  - struct [dvb\\_desc](#) \* [next](#)
  - uint32\_t [frequency](#)
  - union {
    - uint16\_t [bitfield1](#)
    - struct {
      - uint16\_t [fec\\_outer](#):4
      - uint16\_t [reserved\\_future\\_use](#):12
- };
- uint8\_t [modulation](#)
- union {
  - uint32\_t [bitfield2](#)
  - struct {
    - uint32\_t [fec\\_inner](#):4
    - uint32\_t [symbol\\_rate](#):28
- };

### 8.14.1 Detailed Description

Structure containing the cable delivery system descriptor.

#### Parameters

|                    |                                               |
|--------------------|-----------------------------------------------|
| <i>type</i>        | descriptor tag                                |
| <i>length</i>      | descriptor length                             |
| <i>next</i>        | pointer to struct <a href="#">dvb_desc</a>    |
| <i>frequency</i>   | frequency, converted to Hz.                   |
| <i>fec_outer</i>   | FEC outer (typically, Viterbi)                |
| <i>modulation</i>  | modulation                                    |
| <i>fec_inner</i>   | FEC inner (convolutional code)                |
| <i>symbol_rate</i> | symbol rate, converted to symbols/sec (bauds) |

Definition at line 57 of file [desc\\_cable\\_delivery.h](#).

### 8.14.2 Field Documentation

**8.14.2.1**     `union { ... } dvb_desc_cable_delivery::@139`

**8.14.2.2**     `union { ... } dvb_desc_cable_delivery::@141`

**8.14.2.3 bitfield1**   `uint16_t dvb_desc_cable_delivery::bitfield1`

Definition at line 64 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.4 bitfield2**   `uint32_t dvb_desc_cable_delivery::bitfield2`

Definition at line 72 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.5 fec\_inner**   `uint32_t dvb_desc_cable_delivery::fec_inner`

Definition at line 74 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.6 fec\_outer**   `uint16_t dvb_desc_cable_delivery::fec_outer`

Definition at line 66 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.7 frequency**   `uint32_t dvb_desc_cable_delivery::frequency`

Definition at line 62 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.8 length**   `uint8_t dvb_desc_cable_delivery::length`

Definition at line 59 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.9 modulation**   `uint8_t dvb_desc_cable_delivery::modulation`

Definition at line 70 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.10 next** struct [dvb\\_desc](#)\* dvb\_desc\_cable\_delivery::next

Definition at line 60 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.11 reserved\_future\_use** uint16\_t dvb\_desc\_cable\_delivery::reserved\_future\_use

Definition at line 67 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.12 symbol\_rate** uint32\_t dvb\_desc\_cable\_delivery::symbol\_rate

Definition at line 75 of file [desc\\_cable\\_delivery.h](#).

**8.14.2.13 type** uint8\_t dvb\_desc\_cable\_delivery::type

Definition at line 58 of file [desc\\_cable\\_delivery.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[desc\\_cable\\_delivery.h](#)

## 8.15 dvb\_desc\_event\_extended Struct Reference

Structure containing the extended event descriptor.

```
#include <desc_event_extended.h>
```

### Data Fields

- uint8\_t [type](#)
  - uint8\_t [length](#)
  - struct [dvb\\_desc](#) \* [next](#)
  - union {
    - struct {
      - uint8\_t [last\\_id](#):4
      - uint8\_t [id](#):4
    - uint8\_t [ids](#)
- unsigned char [language](#) [4]
- char \* [text](#)
- char \* [text\\_emph](#)
- struct [dvb\\_desc\\_event\\_extended\\_item](#) \* [items](#)
- int [num\\_items](#)

### 8.15.1 Detailed Description

Structure containing the extended event descriptor.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <i>type</i>      | descriptor tag                             |
| <i>length</i>    | descriptor length                          |
| <i>next</i>      | pointer to struct <a href="#">dvb_desc</a> |
| <i>id</i>        | descriptor number                          |
| <i>last_id</i>   | last descriptor number                     |
| <i>language</i>  | ISO 639 language code                      |
| <i>text</i>      | text string                                |
| <i>text_emph</i> | text emphasis string                       |

The emphasis text is the one that uses asterisks. For example, in the text: "the quick *\*fox\** jumps over the lazy table" the emphasis would be "fox".

Definition at line 68 of file [desc\\_event\\_extended.h](#).

**8.15.2 Field Documentation**

**8.15.2.1**     `union { ... } dvb_desc_event_extended::@147`

**8.15.2.2**   **id**   `uint8_t dvb_desc_event_extended::id`

Definition at line 76 of file [desc\\_event\\_extended.h](#).

**8.15.2.3**   **ids**   `uint8_t dvb_desc_event_extended::ids`

Definition at line 78 of file [desc\\_event\\_extended.h](#).

**8.15.2.4**   **items**   `struct dvb\_desc\_event\_extended\_item* dvb_desc_event_extended::items`

Definition at line 84 of file [desc\\_event\\_extended.h](#).

**8.15.2.5**   **language**   `unsigned char dvb_desc_event_extended::language[4]`

Definition at line 81 of file [desc\\_event\\_extended.h](#).

**8.15.2.6 last\_id** `uint8_t dvb_desc_event_extended::last_id`

Definition at line 75 of file [desc\\_event\\_extended.h](#).

**8.15.2.7 length** `uint8_t dvb_desc_event_extended::length`

Definition at line 70 of file [desc\\_event\\_extended.h](#).

**8.15.2.8 next** `struct dvb_desc* dvb_desc_event_extended::next`

Definition at line 71 of file [desc\\_event\\_extended.h](#).

**8.15.2.9 num\_items** `int dvb_desc_event_extended::num_items`

Definition at line 85 of file [desc\\_event\\_extended.h](#).

**8.15.2.10 text** `char* dvb_desc_event_extended::text`

Definition at line 82 of file [desc\\_event\\_extended.h](#).

**8.15.2.11 text\_emph** `char* dvb_desc_event_extended::text_emph`

Definition at line 83 of file [desc\\_event\\_extended.h](#).

**8.15.2.12 type** `uint8_t dvb_desc_event_extended::type`

Definition at line 69 of file [desc\\_event\\_extended.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc\\_event\\_extended.h](#)

## 8.16 dvb\_desc\_event\_extended\_item Struct Reference

```
#include <desc_event_extended.h>
```

## Data Fields

- char \* [description](#)
- char \* [description\\_emph](#)
- char \* [item](#)
- char \* [item\\_emph](#)

### 8.16.1 Detailed Description

Definition at line [42](#) of file [desc\\_event\\_extended.h](#).

### 8.16.2 Field Documentation

#### 8.16.2.1 **description** `char* dvb_desc_event_extended_item::description`

Definition at line [43](#) of file [desc\\_event\\_extended.h](#).

#### 8.16.2.2 **description\_emph** `char* dvb_desc_event_extended_item::description_emph`

Definition at line [44](#) of file [desc\\_event\\_extended.h](#).

#### 8.16.2.3 **item** `char* dvb_desc_event_extended_item::item`

Definition at line [45](#) of file [desc\\_event\\_extended.h](#).

#### 8.16.2.4 **item\_emph** `char* dvb_desc_event_extended_item::item_emph`

Definition at line [46](#) of file [desc\\_event\\_extended.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc\\_event\\_extended.h](#)

## 8.17 dvb\_desc\_event\_short Struct Reference

Structure containing the short event descriptor.

```
#include <desc_event_short.h>
```

### Data Fields

- uint8\_t [type](#)
- uint8\_t [length](#)
- struct [dvb\\_desc](#) \* [next](#)
- unsigned char [language](#) [4]
- char \* [name](#)
- char \* [name\\_emph](#)
- char \* [text](#)
- char \* [text\\_emph](#)

#### 8.17.1 Detailed Description

Structure containing the short event descriptor.



**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <i>type</i>      | descriptor tag                             |
| <i>length</i>    | descriptor length                          |
| <i>next</i>      | pointer to struct <a href="#">dvb_desc</a> |
| <i>language</i>  | ISO 639 language code                      |
| <i>name</i>      | event name string                          |
| <i>name_emph</i> | event name emphasis string                 |
| <i>text</i>      | event text string                          |
| <i>text_emph</i> | event text emphasis string                 |

The emphasis text is the one that uses asterisks. For example, in the text: "the quick *\*fox\** jumps over the lazy table" the emphasis would be "fox".

Definition at line 60 of file [desc\\_event\\_short.h](#).

**8.17.2 Field Documentation**

**8.17.2.1 language** `unsigned char dvb_desc_event_short::language[4]`

Definition at line 65 of file [desc\\_event\\_short.h](#).

**8.17.2.2 length** `uint8_t dvb_desc_event_short::length`

Definition at line 62 of file [desc\\_event\\_short.h](#).

**8.17.2.3 name** `char* dvb_desc_event_short::name`

Definition at line 66 of file [desc\\_event\\_short.h](#).

**8.17.2.4 name\_emph** `char* dvb_desc_event_short::name_emph`

Definition at line 67 of file [desc\\_event\\_short.h](#).

**8.17.2.5 next** `struct dvb\_desc* dvb_desc_event_short::next`

Definition at line 63 of file [desc\\_event\\_short.h](#).

**8.17.2.6 text** `char* dvb_desc_event_short::text`

Definition at line 68 of file [desc\\_event\\_short.h](#).

**8.17.2.7 text\_emph** `char* dvb_desc_event_short::text_emph`

Definition at line 69 of file [desc\\_event\\_short.h](#).

**8.17.2.8 type** `uint8_t dvb_desc_event_short::type`

Definition at line 61 of file [desc\\_event\\_short.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc\\_event\\_short.h](#)

## 8.18 dvb\_desc\_frequency\_list Struct Reference

Struct containing the frequency list descriptor.

```
#include <desc_frequency_list.h>
```

### Data Fields

- `uint8_t type`
  - `uint8_t length`
  - `struct dvb_desc * next`
  - `uint8_t frequencies`
  - `uint32_t * frequency`
  - union {
    - `uint8_t bitfield`
    - struct {
      - `uint8_t freq_type:2`
      - `uint8_t reserved:6`
- ```
};
```

8.18.1 Detailed Description

Struct containing the frequency list descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>frequencies</i>	number of frequencies in the frequency vector
<i>frequency</i>	vector with the centre frequency
<i>freq_type</i>	freq type, being: 0 = undefined, 1 = satellite, 2 = cable or 3 = terrestrial.

Definition at line 56 of file [desc_frequency_list.h](#).

8.18.2 Field Documentation

8.18.2.1 `union { ... } dvb_desc_frequency_list::@151`

8.18.2.2 **bitfield** `uint8_t dvb_desc_frequency_list::bitfield`

Definition at line 65 of file [desc_frequency_list.h](#).

8.18.2.3 **freq_type** `uint8_t dvb_desc_frequency_list::freq_type`

Definition at line 67 of file [desc_frequency_list.h](#).

8.18.2.4 **frequencies** `uint8_t dvb_desc_frequency_list::frequencies`

Definition at line 61 of file [desc_frequency_list.h](#).

8.18.2.5 **frequency** `uint32_t* dvb_desc_frequency_list::frequency`

Definition at line 62 of file [desc_frequency_list.h](#).

8.18.2.6 **length** `uint8_t dvb_desc_frequency_list::length`

Definition at line 58 of file [desc_frequency_list.h](#).

8.18.2.7 **next** `struct dvb_desc* dvb_desc_frequency_list::next`

Definition at line 59 of file [desc_frequency_list.h](#).

8.18.2.8 reserved `uint8_t dvb_desc_frequency_list::reserved`

Definition at line 68 of file [desc_frequency_list.h](#).

8.18.2.9 type `uint8_t dvb_desc_frequency_list::type`

Definition at line 57 of file [desc_frequency_list.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_frequency_list.h](#)

8.19 dvb_desc_hierarchy Struct Reference

Structure containing the hierarchy descriptor.

```
#include <desc_hierarchy.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint8_t hierarchy_type:4`
- `uint8_t reserved:4`
- `uint8_t layer:6`
- `uint8_t reserved2:2`
- `uint8_t embedded_layer:6`
- `uint8_t reserved3:2`
- `uint8_t channel:6`
- `uint8_t reserved4:2`

8.19.1 Detailed Description

Structure containing the hierarchy descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>hierarchy_type</i>	hierarchy type
<i>layer</i>	hierarchy layer index
<i>embedded_layer</i>	hierarchy embedded layer index
<i>channel</i>	hierarchy channel

Definition at line 57 of file [desc_hierarchy.h](#).

8.19.2 Field Documentation

8.19.2.1 channel `uint8_t dvb_desc_hierarchy::channel`

Definition at line 71 of file [desc_hierarchy.h](#).

8.19.2.2 embedded_layer `uint8_t dvb_desc_hierarchy::embedded_layer`

Definition at line 68 of file [desc_hierarchy.h](#).

8.19.2.3 hierarchy_type `uint8_t dvb_desc_hierarchy::hierarchy_type`

Definition at line 62 of file [desc_hierarchy.h](#).

8.19.2.4 layer `uint8_t dvb_desc_hierarchy::layer`

Definition at line 65 of file [desc_hierarchy.h](#).

8.19.2.5 length `uint8_t dvb_desc_hierarchy::length`

Definition at line 59 of file [desc_hierarchy.h](#).

8.19.2.6 next `struct dvb_desc* dvb_desc_hierarchy::next`

Definition at line 60 of file [desc_hierarchy.h](#).

8.19.2.7 reserved `uint8_t dvb_desc_hierarchy::reserved`

Definition at line 63 of file [desc_hierarchy.h](#).

8.19.2.8 reserved2 `uint8_t dvb_desc_hierarchy::reserved2`

Definition at line 66 of file [desc_hierarchy.h](#).

8.19.2.9 reserved3 `uint8_t dvb_desc_hierarchy::reserved3`

Definition at line 69 of file [desc_hierarchy.h](#).

8.19.2.10 reserved4 `uint8_t dvb_desc_hierarchy::reserved4`

Definition at line 72 of file [desc_hierarchy.h](#).

8.19.2.11 type `uint8_t dvb_desc_hierarchy::type`

Definition at line 58 of file [desc_hierarchy.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_hierarchy.h](#)

8.20 dvb_desc_language Struct Reference

Structure containing the ISO639 language descriptor.

```
#include <desc_language.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `unsigned char language [4]`
- `uint8_t audio_type`

8.20.1 Detailed Description

Structure containing the ISO639 language descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>language</i>	ISO639 language string
<i>audio_type</i>	audio type: 0 = undefined, 1 = clean effects, 2 = hearing impaired, 3 = visual impaired comentary, other values are reserved.

Definition at line 55 of file [desc_language.h](#).

8.20.2 Field Documentation

8.20.2.1 audio_type `uint8_t dvb_desc_language::audio_type`

Definition at line 61 of file [desc_language.h](#).

8.20.2.2 language `unsigned char dvb_desc_language::language[4]`

Definition at line 60 of file [desc_language.h](#).

8.20.2.3 length `uint8_t dvb_desc_language::length`

Definition at line 57 of file [desc_language.h](#).

8.20.2.4 next `struct dvb_desc* dvb_desc_language::next`

Definition at line 58 of file [desc_language.h](#).

8.20.2.5 type `uint8_t dvb_desc_language::type`

Definition at line 56 of file [desc_language.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_language.h](#)

8.21 dvb_desc_logical_channel Struct Reference

Structure containing the logical channel number descriptor.

```
#include <desc_logical_channel.h>
```

Data Fields

- `uint8_t` [type](#)
- `uint8_t` [length](#)
- `struct dvb_desc *` [next](#)
- `struct dvb_desc_logical_channel_number *` [lcn](#)

8.21.1 Detailed Description

Structure containing the logical channel number descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>lcn</i>	pointer to struct dvb_desc_logical_channel_number

Definition at line 78 of file [desc_logical_channel.h](#).

8.21.2 Field Documentation

8.21.2.1 lcn struct [dvb_desc_logical_channel_number](#)* dvb_desc_logical_channel::lcn

Definition at line 83 of file [desc_logical_channel.h](#).

8.21.2.2 length uint8_t dvb_desc_logical_channel::length

Definition at line 80 of file [desc_logical_channel.h](#).

8.21.2.3 next struct [dvb_desc](#)* dvb_desc_logical_channel::next

Definition at line 81 of file [desc_logical_channel.h](#).

8.21.2.4 type uint8_t dvb_desc_logical_channel::type

Definition at line 79 of file [desc_logical_channel.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[desc_logical_channel.h](#)

8.22 dvb_desc_logical_channel_number Struct Reference

Structure containing the logical channel number entires.

```
#include <desc_logical_channel.h>
```


Data Fields

- uint16_t [service_id](#)
 - union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [logical_channel_number](#):10
 - uint16_t [reserved](#):5
 - uint16_t [visible_service_flag](#):1
- ```
}
```
- ```
};
```

8.22.1 Detailed Description

Structure containing the logical channel number entires.

Parameters

<i>service_id</i>	service id
<i>visible_service_flag</i>	visible service flag
<i>logical_channel_number</i>	logical channel number

Definition at line [56](#) of file [desc_logical_channel.h](#).

8.22.2 Field Documentation

8.22.2.1 union { ... } dvb_desc_logical_channel_number::@159

8.22.2.2 **bitfield** uint16_t dvb_desc_logical_channel_number::bitfield

Definition at line [59](#) of file [desc_logical_channel.h](#).

8.22.2.3 **logical_channel_number** uint16_t dvb_desc_logical_channel_number::logical_channel_↵
number

Definition at line [61](#) of file [desc_logical_channel.h](#).

8.22.2.4 reserved uint16_t dvb_desc_logical_channel_number::reserved

Definition at line 62 of file [desc_logical_channel.h](#).

8.22.2.5 service_id uint16_t dvb_desc_logical_channel_number::service_id

Definition at line 57 of file [desc_logical_channel.h](#).

8.22.2.6 visible_service_flag uint16_t dvb_desc_logical_channel_number::visible_service_flag

Definition at line 63 of file [desc_logical_channel.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_logical_channel.h](#)

8.23 dvb_desc_network_name Struct Reference

Struct containing the network name descriptor.

```
#include <desc_network_name.h>
```

Data Fields

- uint8_t [type](#)
- uint8_t [length](#)
- struct [dvb_desc](#) * [next](#)
- char * [network_name](#)
- char * [network_name_emph](#)

8.23.1 Detailed Description

Struct containing the network name descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>network_name</i>	network name string
<i>network_name_emph</i>	network name emphasis string

The emphasis text is the one that uses asterisks. For example, in the text: "the quick *fox* jumps over the lazy

table" the emphasis would be "fox".

Definition at line 57 of file [desc_network_name.h](#).

8.23.2 Field Documentation

8.23.2.1 length `uint8_t dvb_desc_network_name::length`

Definition at line 59 of file [desc_network_name.h](#).

8.23.2.2 network_name `char* dvb_desc_network_name::network_name`

Definition at line 62 of file [desc_network_name.h](#).

8.23.2.3 network_name_emph `char* dvb_desc_network_name::network_name_emph`

Definition at line 63 of file [desc_network_name.h](#).

8.23.2.4 next `struct dvb_desc* dvb_desc_network_name::next`

Definition at line 60 of file [desc_network_name.h](#).

8.23.2.5 type `uint8_t dvb_desc_network_name::type`

Definition at line 58 of file [desc_network_name.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_network_name.h](#)

8.24 dvb_desc_registration Struct Reference

Struct containing the frequency list descriptor.

```
#include <desc_registration_id.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint8_t format_identifier [4]`
- `uint8_t * additional_identification_info`

8.24.1 Detailed Description

Struct containing the frequency list descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>format_identifier</i>	32-bit value obtained from ISO/IEC JTC 1/SC 29 which describes the format of the ES The length of the vector is given by: length - 4.

Definition at line 53 of file [desc_registration_id.h](#).

8.24.2 Field Documentation

8.24.2.1 additional_identification_info `uint8_t* dvb_desc_registration::additional_identification←_info`

Definition at line 59 of file [desc_registration_id.h](#).

8.24.2.2 format_identifier `uint8_t dvb_desc_registration::format_identifier[4]`

Definition at line 58 of file [desc_registration_id.h](#).

8.24.2.3 length `uint8_t dvb_desc_registration::length`

Definition at line 55 of file [desc_registration_id.h](#).

8.24.2.4 next `struct dvb_desc* dvb_desc_registration::next`

Definition at line 56 of file [desc_registration_id.h](#).

8.24.2.5 type `uint8_t dvb_desc_registration::type`

Definition at line 54 of file [desc_registration_id.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_registration_id.h](#)

8.25 dvb_desc_sat Struct Reference

Structure containing the satellite delivery system descriptor.

```
#include <desc_sat.h>
```

Data Fields

- uint8_t [type](#)
- uint8_t [length](#)
- struct [dvb_desc](#) * [next](#)
- uint32_t [frequency](#)
- uint16_t [orbit](#)
- uint8_t [modulation_type](#):2
- uint8_t [modulation_system](#):1
- uint8_t [roll_off](#):2
- uint8_t [polarization](#):2
- uint8_t [west_east](#):1
- union {
 - uint32_t [bitfield](#)
 - struct {
 - uint32_t [fec](#):4
 - uint32_t [symbol_rate](#):28

```
};
```

8.25.1 Detailed Description

Structure containing the satellite delivery system descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>frequency</i>	frequency in kHz
<i>orbit</i>	orbital position in degrees (multiplied by 10)
<i>west_east</i>	west east flag. 0 = west, 1 = east
<i>polarization</i>	polarization. 0 = horizontal, 1 = vertical, 2 = left, 3 = right.
<i>roll_off</i>	roll off alpha factor. 0 = 0.35, 1 = 0.25, 2 = 0.20, 3 = reserved.
<i>modulation_system</i>	modulation system. 0 = DVB-S, 1 = DVB-S2.
<i>modulation_type</i>	modulation type. 0 = auto, 1 = QPSK, 2 = 8PSK, 3 = 16-QAM (only for DVB-S2).
<i>symbol_rate</i>	symbol rate in Kbauds.
<i>fec</i>	inner FEC (convolutional code)

Definition at line 63 of file [desc_sat.h](#).

8.25.2 Field Documentation

8.25.2.1 `union { ... } dvb_desc_sat::@163`

8.25.2.2 bitfield `uint32_t dvb_desc_sat::bitfield`

Definition at line 76 of file [desc_sat.h](#).

8.25.2.3 fec `uint32_t dvb_desc_sat::fec`

Definition at line 78 of file [desc_sat.h](#).

8.25.2.4 frequency `uint32_t dvb_desc_sat::frequency`

Definition at line 68 of file [desc_sat.h](#).

8.25.2.5 length `uint8_t dvb_desc_sat::length`

Definition at line 65 of file [desc_sat.h](#).

8.25.2.6 modulation_system `uint8_t dvb_desc_sat::modulation_system`

Definition at line 71 of file [desc_sat.h](#).

8.25.2.7 modulation_type `uint8_t dvb_desc_sat::modulation_type`

Definition at line 70 of file [desc_sat.h](#).

8.25.2.8 next `struct dvb_desc* dvb_desc_sat::next`

Definition at line 66 of file [desc_sat.h](#).

8.25.2.9 orbit `uint16_t dvb_desc_sat::orbit`

Definition at line 69 of file [desc_sat.h](#).

8.25.2.10 polarization `uint8_t dvb_desc_sat::polarization`

Definition at line 73 of file [desc_sat.h](#).

8.25.2.11 roll_off `uint8_t dvb_desc_sat::roll_off`

Definition at line 72 of file [desc_sat.h](#).

8.25.2.12 symbol_rate `uint32_t dvb_desc_sat::symbol_rate`

Definition at line 79 of file [desc_sat.h](#).

8.25.2.13 type `uint8_t dvb_desc_sat::type`

Definition at line 64 of file [desc_sat.h](#).

8.25.2.14 west_east `uint8_t dvb_desc_sat::west_east`

Definition at line 74 of file [desc_sat.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_sat.h](#)

8.26 dvb_desc_service Struct Reference

Structure containing the service descriptor.

```
#include <desc_service.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint8_t service_type`
- `char * name`
- `char * name_emph`
- `char * provider`
- `char * provider_emph`

8.26.1 Detailed Description

Structure containing the service descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>service_type</i>	service type
<i>name</i>	name string
<i>name_emph</i>	name emphasis string
<i>provider</i>	provider string
<i>provider_emph</i>	provider emphasis string

The emphasis text is the one that uses asterisks. For example, in the text: "the quick **fox** jumps over the lazy table" the emphasis would be "fox".

Definition at line 60 of file [desc_service.h](#).

8.26.2 Field Documentation

8.26.2.1 **length** `uint8_t dvb_desc_service::length`

Definition at line 62 of file [desc_service.h](#).

8.26.2.2 **name** `char* dvb_desc_service::name`

Definition at line 66 of file [desc_service.h](#).

8.26.2.3 **name_emph** `char* dvb_desc_service::name_emph`

Definition at line 67 of file [desc_service.h](#).

8.26.2.4 **next** `struct dvb_desc* dvb_desc_service::next`

Definition at line 63 of file [desc_service.h](#).

8.26.2.5 **provider** `char* dvb_desc_service::provider`

Definition at line 68 of file [desc_service.h](#).

8.26.2.6 provider_emph `char* dvb_desc_service::provider_emph`

Definition at line 69 of file [desc_service.h](#).

8.26.2.7 service_type `uint8_t dvb_desc_service::service_type`

Definition at line 65 of file [desc_service.h](#).

8.26.2.8 type `uint8_t dvb_desc_service::type`

Definition at line 61 of file [desc_service.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_service.h](#)

8.27 dvb_desc_t2_delivery Struct Reference

Structure containing the T2 delivery system descriptor.

```
#include <desc_t2_delivery.h>
```

Data Fields

- `uint8_t` [plp_id](#)
 - `uint16_t` [system_id](#)
 - `union` {
 - `uint16_t` [bitfield](#)
 - `struct` {
 - `uint16_t` [tfs_flag](#):1
 - `uint16_t` [other_frequency_flag](#):1
 - `uint16_t` [transmission_mode](#):3
 - `uint16_t` [guard_interval](#):3
 - `uint16_t` [reserved](#):2
 - `uint16_t` [bandwidth](#):4
 - `uint16_t` [SISO_MISO](#):2
- `uint32_t` * [centre_frequency](#)
- `uint8_t` [frequency_loop_length](#)
- `uint8_t` [subcel_info_loop_length](#)
- `struct` [dvb_desc_t2_delivery_subcell_old](#) * [subcell](#)
- `unsigned int` [num_cell](#)
- `struct` [dvb_desc_t2_delivery_cell](#) * [cell](#)

8.27.1 Detailed Description

Structure containing the T2 delivery system descriptor.

Parameters

<i>plp_id</i>	data PLP id
<i>system_id</i>	T2 system id
<i>SISO_MISO</i>	SISO MISO
<i>bandwidth</i>	bandwidth
<i>guard_interval</i>	guard interval
<i>transmission_mode</i>	transmission mode
<i>other_frequency_flag</i>	other frequency flag
<i>tfs_flag</i>	tfs flag
<i>centre_frequency</i>	centre frequency vector. It contains the full frequencies for all cells and subcells.
<i>frequency_loop_length</i>	size of the dvb_desc_t2_delivery::centre_frequency vector
<i>subcel_info_loop_length</i>	unused. Always 0
<i>subcell</i>	unused. Always NULL
<i>num_cell</i>	number of cells
<i>cell</i>	cell array. Contains per-cell and per-subcell pointers to the frequencies parsed.

Definition at line 119 of file [desc_t2_delivery.h](#).

8.27.2 Field Documentation

8.27.2.1 `union { ... } dvb_desc_t2_delivery::@167`

8.27.2.2 **bandwidth** `uint16_t dvb_desc_t2_delivery::bandwidth`

Definition at line 132 of file [desc_t2_delivery.h](#).

8.27.2.3 **bitfield** `uint16_t dvb_desc_t2_delivery::bitfield`

Definition at line 125 of file [desc_t2_delivery.h](#).

8.27.2.4 **cell** `struct dvb_desc_t2_delivery_cell* dvb_desc_t2_delivery::cell`

Definition at line 146 of file [desc_t2_delivery.h](#).

8.27.2.5 centre_frequency uint32_t* dvb_desc_t2_delivery::centre_frequency

Definition at line 137 of file [desc_t2_delivery.h](#).

8.27.2.6 frequency_loop_length uint8_t dvb_desc_t2_delivery::frequency_loop_length

Definition at line 138 of file [desc_t2_delivery.h](#).

8.27.2.7 guard_interval uint16_t dvb_desc_t2_delivery::guard_interval

Definition at line 130 of file [desc_t2_delivery.h](#).

8.27.2.8 num_cell unsigned int dvb_desc_t2_delivery::num_cell

Definition at line 145 of file [desc_t2_delivery.h](#).

8.27.2.9 other_frequency_flag uint16_t dvb_desc_t2_delivery::other_frequency_flag

Definition at line 128 of file [desc_t2_delivery.h](#).

8.27.2.10 plp_id uint8_t dvb_desc_t2_delivery::plp_id

Definition at line 122 of file [desc_t2_delivery.h](#).

8.27.2.11 reserved uint16_t dvb_desc_t2_delivery::reserved

Definition at line 131 of file [desc_t2_delivery.h](#).

8.27.2.12 SISO_MISO uint16_t dvb_desc_t2_delivery::SISO_MISO

Definition at line 133 of file [desc_t2_delivery.h](#).

8.27.2.13 subcel_info_loop_length uint8_t dvb_desc_t2_delivery::subcel_info_loop_length

Definition at line 141 of file [desc_t2_delivery.h](#).

8.27.2.14 subcell struct dvb_desc_t2_delivery_subcell_old* dvb_desc_t2_delivery::subcell

Definition at line 142 of file [desc_t2_delivery.h](#).

8.27.2.15 system_id uint16_t dvb_desc_t2_delivery::system_id

Definition at line 123 of file [desc_t2_delivery.h](#).

8.27.2.16 tfs_flag uint16_t dvb_desc_t2_delivery::tfs_flag

Definition at line 127 of file [desc_t2_delivery.h](#).

8.27.2.17 transmission_mode uint16_t dvb_desc_t2_delivery::transmission_mode

Definition at line 129 of file [desc_t2_delivery.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[desc_t2_delivery.h](#)

8.28 dvb_desc_t2_delivery_cell Struct Reference

Structure to describe transponder cells.

```
#include <desc_t2_delivery.h>
```

Data Fields

- uint16_t [cell_id](#)
- int [num_freqs](#)
- uint32_t * [centre_frequency](#)
- uint8_t [subcel_length](#)
- struct [dvb_desc_t2_delivery_subcell](#) * [subcel](#)

8.28.1 Detailed Description

Structure to describe transponder cells.

Parameters

<i>cell_id</i>	cell id extension
<i>num_freqs</i>	number of cell frequencies
<i>centre_frequency</i>	pointer to centre frequencies
<i>subcel_length</i>	number of subcells. May be zero
<i>subcell</i>	pointer to subcell array. May be NULL

Definition at line 86 of file [desc_t2_delivery.h](#).

8.28.2 Field Documentation

8.28.2.1 cell_id `uint16_t dvb_desc_t2_delivery_cell::cell_id`

Definition at line 87 of file [desc_t2_delivery.h](#).

8.28.2.2 centre_frequency `uint32_t* dvb_desc_t2_delivery_cell::centre_frequency`

Definition at line 89 of file [desc_t2_delivery.h](#).

8.28.2.3 num_freqs `int dvb_desc_t2_delivery_cell::num_freqs`

Definition at line 88 of file [desc_t2_delivery.h](#).

8.28.2.4 subcel `struct dvb_desc_t2_delivery_subcell* dvb_desc_t2_delivery_cell::subcel`

Definition at line 91 of file [desc_t2_delivery.h](#).

8.28.2.5 subcel_length `uint8_t dvb_desc_t2_delivery_cell::subcel_length`

Definition at line 90 of file [desc_t2_delivery.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_t2_delivery.h](#)

8.29 dvb_desc_t2_delivery_subcell Struct Reference

Structure to describe transponder subcell extension and frequencies.

```
#include <desc_t2_delivery.h>
```

Data Fields

- uint8_t [cell_id_extension](#)
- uint32_t [transposer_frequency](#)

8.29.1 Detailed Description

Structure to describe transponder subcell extension and frequencies.

Parameters

<i>cell_id_extension</i>	cell id extension
<i>transposer_frequency</i>	pointer to transposer frequency

Definition at line 69 of file [desc_t2_delivery.h](#).

8.29.2 Field Documentation

8.29.2.1 cell_id_extension `uint8_t dvb_desc_t2_delivery_subcell::cell_id_extension`

Definition at line 70 of file [desc_t2_delivery.h](#).

8.29.2.2 transposer_frequency `uint32_t dvb_desc_t2_delivery_subcell::transposer_frequency`

Definition at line 71 of file [desc_t2_delivery.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_t2_delivery.h](#)

8.30 dvb_desc_t2_delivery_subcell_old Struct Reference

Structure to describe transponder subcell extension and frequencies.

```
#include <desc_t2_delivery.h>
```

Data Fields

- `uint8_t` [cell_id_extension](#)
- `uint16_t` [transposer_frequency](#)

8.30.1 Detailed Description

Structure to describe transponder subcell extension and frequencies.

Parameters

<i>cell_id_extension</i>	cell id extension
<i>transposer_frequency</i>	transposer frequency

NOTE: This struct is deprecated and will never be filled. It is kept here just to avoid breaking ABI.

All subcell transposer frequencies will be added to `dvb_desc_t2_delivery::centre_frequency` array.

Definition at line 55 of file `desc_t2_delivery.h`.

8.30.2 Field Documentation

8.30.2.1 cell_id_extension `uint8_t dvb_desc_t2_delivery_subcell_old::cell_id_extension`

Definition at line 56 of file `desc_t2_delivery.h`.

8.30.2.2 transposer_frequency `uint16_t dvb_desc_t2_delivery_subcell_old::transposer_frequency`

Definition at line 57 of file `desc_t2_delivery.h`.

The documentation for this struct was generated from the following file:

- `lib/include/libdvbv5/desc_t2_delivery.h`

8.31 dvb_desc_terrestrial_delivery Struct Reference

Structure containing the DVB-T terrestrial delivery system descriptor.

```
#include <desc_terrestrial_delivery.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint32_t centre_frequency`
- `uint8_t reserved_future_use1:2`
- `uint8_t mpe_fec_indicator:1`
- `uint8_t time_slice_indicator:1`
- `uint8_t priority:1`
- `uint8_t bandwidth:3`
- `uint8_t code_rate_hp_stream:3`
- `uint8_t hierarchy_information:3`
- `uint8_t constellation:2`
- `uint8_t other_frequency_flag:1`
- `uint8_t transmission_mode:2`
- `uint8_t guard_interval:2`
- `uint8_t code_rate_lp_stream:3`
- `uint32_t reserved_future_use2`

8.31.1 Detailed Description

Structure containing the DVB-T terrestrial delivery system descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>centre_frequency</i>	centre frequency, multiplied by 10 Hz
<i>bandwidth</i>	bandwidth
<i>priority</i>	priority (0 = LP, 1 = HP)
<i>time_slice_indicator</i>	time slicing indicator
<i>mpe_fec_indicator</i>	mpe fec indicator. If 1, MPE-FEC is not used.
<i>constellation</i>	constellation
<i>hierarchy_information</i>	hierarchy information
<i>code_rate_hp_stream</i>	code rate hp stream
<i>code_rate_lp_stream</i>	code rate lp stream
<i>guard_interval</i>	guard interval
<i>transmission_mode</i>	transmission mode
<i>other_frequency_flag</i>	other frequency flag

Definition at line 65 of file [desc_terrestrial_delivery.h](#).

8.31.2 Field Documentation

8.31.2.1 bandwidth `uint8_t dvb_desc_terrestrial_delivery::bandwidth`

Definition at line 75 of file [desc_terrestrial_delivery.h](#).

8.31.2.2 centre_frequency `uint32_t dvb_desc_terrestrial_delivery::centre_frequency`

Definition at line 70 of file [desc_terrestrial_delivery.h](#).

8.31.2.3 code_rate_hp_stream `uint8_t dvb_desc_terrestrial_delivery::code_rate_hp_stream`

Definition at line 76 of file [desc_terrestrial_delivery.h](#).

8.31.2.4 code_rate_lp_stream `uint8_t dvb_desc_terrestrial_delivery::code_rate_lp_stream`

Definition at line 82 of file [desc_terrestrial_delivery.h](#).

8.31.2.5 constellation `uint8_t dvb_desc_terrestrial_delivery::constellation`

Definition at line 78 of file [desc_terrestrial_delivery.h](#).

8.31.2.6 guard_interval `uint8_t dvb_desc_terrestrial_delivery::guard_interval`

Definition at line 81 of file [desc_terrestrial_delivery.h](#).

8.31.2.7 hierarchy_information `uint8_t dvb_desc_terrestrial_delivery::hierarchy_information`

Definition at line 77 of file [desc_terrestrial_delivery.h](#).

8.31.2.8 length `uint8_t dvb_desc_terrestrial_delivery::length`

Definition at line 67 of file [desc_terrestrial_delivery.h](#).

8.31.2.9 mpe_fec_indicator `uint8_t dvb_desc_terrestrial_delivery::mpe_fec_indicator`

Definition at line 72 of file [desc_terrestrial_delivery.h](#).

8.31.2.10 next `struct dvb_desc* dvb_desc_terrestrial_delivery::next`

Definition at line 68 of file [desc_terrestrial_delivery.h](#).

8.31.2.11 other_frequency_flag `uint8_t dvb_desc_terrestrial_delivery::other_frequency_flag`

Definition at line 79 of file [desc_terrestrial_delivery.h](#).

8.31.2.12 priority `uint8_t dvb_desc_terrestrial_delivery::priority`

Definition at line 74 of file [desc_terrestrial_delivery.h](#).

8.31.2.13 reserved_future_use1 `uint8_t dvb_desc_terrestrial_delivery::reserved_future_use1`

Definition at line 71 of file [desc_terrestrial_delivery.h](#).

8.31.2.14 reserved_future_use2 `uint32_t dvb_desc_terrestrial_delivery::reserved_future_use2`

Definition at line 83 of file [desc_terrestrial_delivery.h](#).

8.31.2.15 time_slice_indicator `uint8_t dvb_desc_terrestrial_delivery::time_slice_indicator`

Definition at line 73 of file [desc_terrestrial_delivery.h](#).

8.31.2.16 transmission_mode `uint8_t dvb_desc_terrestrial_delivery::transmission_mode`

Definition at line 80 of file [desc_terrestrial_delivery.h](#).

8.31.2.17 type `uint8_t dvb_desc_terrestrial_delivery::type`

Definition at line 66 of file [desc_terrestrial_delivery.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_terrestrial_delivery.h](#)

8.32 dvb_desc_ts_info Struct Reference

Structure describing the ISDB TS information descriptor.

```
#include <desc_ts_info.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `char * ts_name`
- `char * ts_name_emph`
- `struct dvb_desc_ts_info_transmission_type transmission_type`
- `uint16_t * service_id`
- `union {
 uint16_t bitfield
 struct {
 uint8_t transmission_type_count:2
 uint8_t length_of_ts_name:6
 uint8_t remote_control_key_id:8
 }
};`

8.32.1 Detailed Description

Structure describing the ISDB TS information descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>remote_control_key_id</i>	remote control key id
<i>length_of_ts_name</i>	length of ts name
<i>transmission_type_count</i>	transmission type count
<i>ts_name</i>	ts name string
<i>ts_name_emph</i>	ts name emphasis string
<i>transmission_type</i>	struct dvb_desc_ts_info_transmission_type content
<i>service_id</i>	service id vector

Definition at line 76 of file [desc_ts_info.h](#).

8.32.2 Field Documentation

8.32.2.1 `union { ... } dvb_desc_ts_info::@171`

8.32.2.2 **bitfield** `uint16_t dvb_desc_ts_info::bitfield`

Definition at line 86 of file [desc_ts_info.h](#).

8.32.2.3 **length** `uint8_t dvb_desc_ts_info::length`

Definition at line 78 of file [desc_ts_info.h](#).

8.32.2.4 **length_of_ts_name** `uint8_t dvb_desc_ts_info::length_of_ts_name`

Definition at line 89 of file [desc_ts_info.h](#).

8.32.2.5 **next** `struct dvb_desc* dvb_desc_ts_info::next`

Definition at line 79 of file [desc_ts_info.h](#).

8.32.2.6 remote_control_key_id uint8_t dvb_desc_ts_info::remote_control_key_id

Definition at line 90 of file [desc_ts_info.h](#).

8.32.2.7 service_id uint16_t* dvb_desc_ts_info::service_id

Definition at line 83 of file [desc_ts_info.h](#).

8.32.2.8 transmission_type struct [dvb_desc_ts_info_transmission_type](#) dvb_desc_ts_info::transmission_type

Definition at line 82 of file [desc_ts_info.h](#).

8.32.2.9 transmission_type_count uint8_t dvb_desc_ts_info::transmission_type_count

Definition at line 88 of file [desc_ts_info.h](#).

8.32.2.10 ts_name char* dvb_desc_ts_info::ts_name

Definition at line 81 of file [desc_ts_info.h](#).

8.32.2.11 ts_name_emph char * dvb_desc_ts_info::ts_name_emph

Definition at line 81 of file [desc_ts_info.h](#).

8.32.2.12 type uint8_t dvb_desc_ts_info::type

Definition at line 77 of file [desc_ts_info.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[desc_ts_info.h](#)

8.33 dvb_desc_ts_info_transmission_type Struct Reference

ISDB TS information transmission type.

```
#include <desc_ts_info.h>
```

Data Fields

- uint8_t [transmission_type_info](#)
- uint8_t [num_of_service](#)

8.33.1 Detailed Description

ISDB TS information transmission type.

Parameters

<i>transmission_type_info</i>	transmission type info
<i>num_of_service</i>	num of service

Definition at line 54 of file [desc_ts_info.h](#).

8.33.2 Field Documentation

8.33.2.1 num_of_service `uint8_t dvb_desc_ts_info_transmission_type::num_of_service`

Definition at line 56 of file [desc_ts_info.h](#).

8.33.2.2 transmission_type_info `uint8_t dvb_desc_ts_info_transmission_type::transmission_type←_info`

Definition at line 55 of file [desc_ts_info.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_ts_info.h](#)

8.34 dvb_descriptor Struct Reference

Contains the parser information for the MPEG-TS parser code.

```
#include <descriptors.h>
```

Data Fields

- `const char *` [name](#)
- [dvb_desc_init_func](#) init
- [dvb_desc_print_func](#) print
- [dvb_desc_free_func](#) free
- `ssize_t` [size](#)

8.34.1 Detailed Description

Contains the parser information for the MPEG-TS parser code.

Parameters

<i>name</i>	String containing the name of the descriptor
<i>init</i>	Pointer to a function to initialize the descriptor parser. This function fills the descriptor-specific internal structures
<i>print</i>	Prints the content of the descriptor
<i>free</i>	Frees all memory blocks allocated by the init function
<i>size</i>	Descriptor's size, in bytes.

Definition at line 249 of file [descriptors.h](#).

8.34.2 Field Documentation

8.34.2.1 free `dvb_desc_free_func` `dvb_descriptor::free`

Definition at line 253 of file [descriptors.h](#).

8.34.2.2 init `dvb_desc_init_func` `dvb_descriptor::init`

Definition at line 251 of file [descriptors.h](#).

8.34.2.3 name `const char*` `dvb_descriptor::name`

Definition at line 250 of file [descriptors.h](#).

8.34.2.4 print `dvb_desc_print_func` `dvb_descriptor::print`

Definition at line 252 of file [descriptors.h](#).

8.34.2.5 size `ssize_t` `dvb_descriptor::size`

Definition at line 254 of file [descriptors.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/descriptors.h](#)

8.35 dvb_dev_list Struct Reference

Digital TV device node properties.

```
#include <dvb-dev.h>
```

Data Fields

- char * [syspath](#)
- char * [path](#)
- char * [sysname](#)
- enum [dvb_dev_type](#) [dvb_type](#)
- char * [bus_addr](#)
- char * [bus_id](#)
- char * [manufacturer](#)
- char * [product](#)
- char * [serial](#)

8.35.1 Detailed Description

Digital TV device node properties.

Parameters

<i>path</i>	path for the /dev file handler
<i>sysname</i>	Kernel's system name for the device (dvb?.frontend?, for example)
<i>dvb_type</i>	type of the DVB device, as defined by enum dvb_dev_type
<i>bus_addr</i>	address of the device at the bus. For USB devices, it will be like: usb:3-1.1.4; for PCI devices: pci:0000:01:00.0)
<i>bus_id</i>	Id of the device at the bus (optional, PCI ID or USB ID)
<i>manufacturer</i>	Device's manufacturer name (optional, only on USB)
<i>product</i>	Device's product name (optional, only on USB)
<i>serial</i>	Device's serial name (optional, only on USB)

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line [86](#) of file [dvb-dev.h](#).

8.35.2 Field Documentation

8.35.2.1 bus_addr char* dvb_dev_list::bus_addr

Definition at line [91](#) of file [dvb-dev.h](#).

8.35.2.2 bus_id `char* dvb_dev_list::bus_id`

Definition at line 92 of file [dvb-dev.h](#).

8.35.2.3 dvb_type `enum dvb_dev_type dvb_dev_list::dvb_type`

Definition at line 90 of file [dvb-dev.h](#).

8.35.2.4 manufacturer `char* dvb_dev_list::manufacturer`

Definition at line 93 of file [dvb-dev.h](#).

8.35.2.5 path `char* dvb_dev_list::path`

Examples

[dvbv5-zap.c](#).

Definition at line 88 of file [dvb-dev.h](#).

8.35.2.6 product `char* dvb_dev_list::product`

Definition at line 94 of file [dvb-dev.h](#).

8.35.2.7 serial `char* dvb_dev_list::serial`

Definition at line 95 of file [dvb-dev.h](#).

8.35.2.8 sysname `char* dvb_dev_list::sysname`

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 89 of file [dvb-dev.h](#).

8.35.2.9 syspath `char* dvb_dev_list::syspath`

Definition at line 87 of file [dvb-dev.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-dev.h](#)

8.36 dvb_device Struct Reference

Digital TV list of devices.

```
#include <dvb-dev.h>
```

Data Fields

- struct [dvb_dev_list](#) * [devices](#)
- int [num_devices](#)
- struct [dvb_v5_fe_parms](#) * [fe_parms](#)

8.36.1 Detailed Description

Digital TV list of devices.

Parameters

<i>devices</i>	Array with a dvb_dev_list of devices. Each device node is a different entry at the list.
<i>num_devices</i>	number of elements at the devices array.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 140 of file [dvb-dev.h](#).

8.36.2 Field Documentation

8.36.2.1 devices `struct dvb_dev_list* dvb_device::devices`

Definition at line 142 of file [dvb-dev.h](#).

8.36.2.2 fe_parms `struct dvb_v5_fe_parms* dvb_device::fe_parms`

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line [146](#) of file [dvb-dev.h](#).

8.36.2.3 num_devices `int dvb_device::num_devices`

Definition at line [143](#) of file [dvb-dev.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-dev.h](#)

8.37 dvb_elementary_pid Struct Reference

associates an elementary stream type with its PID

```
#include <dvb-file.h>
```

Data Fields

- [uint8_t type](#)
- [uint16_t pid](#)

8.37.1 Detailed Description

associates an elementary stream type with its PID

Parameters

<i>type</i>	Elementary stream type
<i>pid</i>	Elementary stream Program ID

Definition at line [58](#) of file [dvb-file.h](#).

8.37.2 Field Documentation

8.37.2.1 pid uint16_t dvb_elementary_pid::pid

Examples

[dvbv5-zap.c](#).

Definition at line 60 of file [dvb-file.h](#).

8.37.2.2 type uint8_t dvb_elementary_pid::type

Examples

[dvbv5-zap.c](#).

Definition at line 59 of file [dvb-file.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-file.h](#)

8.38 dvb_entry Struct Reference

Represents one entry on a DTV file.

```
#include <dvb-file.h>
```

Data Fields

- struct dtv_property [props](#) [DTV_MAX_COMMAND]
- unsigned int [n_props](#)
- struct [dvb_entry](#) * [next](#)
- uint16_t [service_id](#)
- uint16_t * [video_pid](#)
- uint16_t * [audio_pid](#)
- struct [dvb_elementary_pid](#) * [other_el_pid](#)
- unsigned [video_pid_len](#)
- unsigned [audio_pid_len](#)
- unsigned [other_el_pid_len](#)
- char * [channel](#)
- char * [vchannel](#)
- char * [location](#)
- int [sat_number](#)
- unsigned [freq_bpf](#)
- unsigned [diseqc_wait](#)
- char * [lnb](#)
- uint16_t [network_id](#)
- uint16_t [transport_id](#)

8.38.1 Detailed Description

Represents one entry on a DTV file.

Parameters

<i>props</i>	A property key/value pair. The keys are the ones specified at the DVB API, plus the ones defined internally by libdvbv5, at the dvb-v5-std.h header file.
<i>next</i>	a pointer to the next entry. NULL if this is the last one.
<i>service_id</i>	Service ID associated with a program inside a transponder. Please note that pure "channel" files will have this field filled with 0.
<i>video_pid</i>	Array with the video program IDs inside a service
<i>audio_pid</i>	Array with the audio program IDs inside a service
<i>other_el_pid</i>	Array with all non-audio/video program IDs inside a service
<i>video_pid_len</i>	Size of the video_pid array
<i>audio_pid_len</i>	Size of the audio_pid array
<i>other_el_pid_len</i>	Size of the other_el_pid array
<i>channel</i>	String containing the name of the channel
<i>vchannel</i>	String representing the Number of the channel
<i>location</i>	String representing the location of the channel
<i>sat_number</i>	For satellite streams, this represents the number of the satellite dish on a DiSeqC arrangement. Should be zero on arrangements without DiSeqC.
<i>freq_bpf</i>	SCR/Unicable band-pass filter frequency to use, in kHz. For non SRC/Unicable arrangements, it should be zero.
<i>diseqc_wait</i>	Extra time to wait for DiSeqC commands to complete, in ms. The library will use 15 ms as the minimal time, plus the time specified on this field.
<i>lnb</i>	String with the name of the LNBf to be used for satellite tuning. The names should match the names provided by dvb_sat_get_lnb() call (see dvb-sat.h).

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 104 of file [dvb-file.h](#).

8.38.2 Field Documentation

8.38.2.1 **audio_pid** `uint16_t * dvb_entry::audio_pid`

Examples

[dvbv5-zap.c](#).

Definition at line 109 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.2 audio_pid_len unsigned dvb_entry::audio_pid_len

Examples

[dvbv5-zap.c](#).

Definition at line 111 of file [dvb-file.h](#).

8.38.2.3 channel char* dvb_entry::channel

Examples

[dvbv5-zap.c](#).

Definition at line 112 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.4 diseqc_wait unsigned dvb_entry::diseqc_wait

Definition at line 119 of file [dvb-file.h](#).

8.38.2.5 freq_bpf unsigned dvb_entry::freq_bpf

Definition at line 118 of file [dvb-file.h](#).

8.38.2.6 lnb char* dvb_entry::lnb

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 120 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.7 location `char* dvb_entry::location`

Definition at line 115 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.8 n_props `unsigned int dvb_entry::n_props`

Examples

[dvbv5-zap.c](#).

Definition at line 106 of file [dvb-file.h](#).

8.38.2.9 network_id `uint16_t dvb_entry::network_id`

Definition at line 122 of file [dvb-file.h](#).

8.38.2.10 next `struct dvb_entry* dvb_entry::next`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 107 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.11 other_el_pid `struct dvb_elementary_pid* dvb_entry::other_el_pid`

Examples

[dvbv5-zap.c](#).

Definition at line 110 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.12 other_el_pid_len `unsigned dvb_entry::other_el_pid_len`

Examples

[dvbv5-zap.c](#).

Definition at line 111 of file [dvb-file.h](#).

8.38.2.13 props `struct dtv_property dvb_entry::props[DTV_MAX_COMMAND]`

Examples

[dvbv5-zap.c](#).

Definition at line 105 of file [dvb-file.h](#).

8.38.2.14 sat_number `int dvb_entry::sat_number`

Examples

[dvbv5-zap.c](#).

Definition at line 117 of file [dvb-file.h](#).

8.38.2.15 service_id `uint16_t dvb_entry::service_id`

Examples

[dvbv5-zap.c](#).

Definition at line 108 of file [dvb-file.h](#).

8.38.2.16 transport_id `uint16_t dvb_entry::transport_id`

Definition at line 123 of file [dvb-file.h](#).

8.38.2.17 vchannel `char* dvb_entry::vchannel`

Examples

[dvbv5-zap.c](#).

Definition at line 113 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.18 video_pid `uint16_t* dvb_entry::video_pid`

Examples

[dvbv5-zap.c](#).

Definition at line 109 of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.38.2.19 video_pid_len `unsigned dvb_entry::video_pid_len`

Examples

[dvbv5-zap.c](#).

Definition at line 111 of file [dvb-file.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-file.h](#)

8.39 dvb_ext_descriptor Struct Reference

Structure that describes the parser functions for the extended descriptors.

```
#include <desc_extension.h>
```

Data Fields

- `const char *` [name](#)
- [dvb_desc_ext_init_func](#) `init`
- [dvb_desc_ext_print_func](#) `print`
- [dvb_desc_ext_free_func](#) `free`
- `ssize_t` [size](#)

8.39.1 Detailed Description

Structure that describes the parser functions for the extended descriptors.

Works on a similar way as struct [dvb_descriptor](#).

Parameters

<i>name</i>	name of the descriptor
<i>init</i>	init dvb_desc_ext_init_func pointer
<i>print</i>	print dvb_desc_ext_print_func pointer
<i>free</i>	free dvb_desc_ext_free_func pointer
<i>size</i>	size of the descriptor

Definition at line 171 of file [desc_extension.h](#).

8.39.2 Field Documentation

8.39.2.1 free [dvb_desc_ext_free_func](#) dvb_ext_descriptor::free

Definition at line 175 of file [desc_extension.h](#).

8.39.2.2 init [dvb_desc_ext_init_func](#) dvb_ext_descriptor::init

Definition at line 173 of file [desc_extension.h](#).

8.39.2.3 name const char* dvb_ext_descriptor::name

Definition at line 172 of file [desc_extension.h](#).

8.39.2.4 print [dvb_desc_ext_print_func](#) dvb_ext_descriptor::print

Definition at line 174 of file [desc_extension.h](#).

8.39.2.5 size ssize_t dvb_ext_descriptor::size

Definition at line 176 of file [desc_extension.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_extension.h](#)

8.40 dvb_extension_descriptor Struct Reference

Structure containing the extended descriptors.

```
#include <desc_extension.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint8_t extension_code`
- `struct dvb_desc * descriptor`

8.40.1 Detailed Description

Structure containing the extended descriptors.

Parameters

<i>type</i>	Descriptor type
<i>length</i>	Length of the descriptor
<i>next</i>	pointer to the <code>dvb_desc</code> descriptor
<i>extension_code</i>	extension code
<i>descriptor</i>	pointer to struct <code>dvb_desc</code>

Definition at line 115 of file `desc_extension.h`.

8.40.2 Field Documentation

8.40.2.1 descriptor `struct dvb_desc* dvb_extension_descriptor::descriptor`

Definition at line 122 of file `desc_extension.h`.

8.40.2.2 extension_code `uint8_t dvb_extension_descriptor::extension_code`

Definition at line 120 of file `desc_extension.h`.

8.40.2.3 length `uint8_t dvb_extension_descriptor::length`

Definition at line 117 of file `desc_extension.h`.

8.40.2.4 next struct [dvb_desc](#)* dvb_extension_descriptor::next

Definition at line 118 of file [desc_extension.h](#).

8.40.2.5 type uint8_t dvb_extension_descriptor::type

Definition at line 116 of file [desc_extension.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[desc_extension.h](#)

8.41 dvb_file Struct Reference

Describes an entire DVB file opened.

```
#include <dvb-file.h>
```

Data Fields

- char * [fname](#)
- int [n_entries](#)
- struct [dvb_entry](#) * [first_entry](#)

8.41.1 Detailed Description

Describes an entire DVB file opened.

Parameters

<i>fname</i>	name of the file
<i>n_entries</i>	number of the entries read
<i>first_entry</i>	entry for the first entry. NULL if the file is empty.

Examples

[dvb-format-convert.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 135 of file [dvb-file.h](#).

8.41.2 Field Documentation

8.41.2.1 first_entry `struct dvb_entry* dvb_file::first_entry`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line [138](#) of file [dvb-file.h](#).

Referenced by [dvb_file_free\(\)](#).

8.41.2.2 fname `char* dvb_file::fname`

Definition at line [136](#) of file [dvb-file.h](#).

8.41.2.3 n_entries `int dvb_file::n_entries`

Definition at line [137](#) of file [dvb-file.h](#).

The documentation for this struct was generated from the following file:

- `lib/include/libdvbv5/dvb-file.h`

8.42 dvb_mpeg_es_pic_start Struct Reference

MPEG ES Picture start header.

```
#include <mpeg_es.h>
```

Data Fields

- union {
 uint32_t [bitfield](#)
 struct {
 uint32_t [type](#):8
 uint32_t [sync](#):24
 }
};
- union {
 uint32_t [bitfield2](#)
 struct {
 uint32_t [dummy](#):3
 uint32_t [vbv_delay](#):16
 uint32_t [coding_type](#):3
 uint32_t [temporal_ref](#):10
 }
};

8.42.1 Detailed Description

MPEG ES Picture start header.

Parameters

<i>type</i>	DVB_MPEG_ES_PIC_START
<i>sync</i>	Sync bytes
<i>dummy</i>	Unused
<i>vbv_delay</i>	VBV delay
<i>coding_type</i>	Frame type (enum dvb_mpeg_es_frame_t)
<i>temporal_ref</i>	Temporal sequence number

Definition at line 130 of file [mpeg_es.h](#).

8.42.2 Field Documentation

8.42.2.1 `union { ... } dvb_mpeg_es_pic_start::@95`

8.42.2.2 `union { ... } dvb_mpeg_es_pic_start::@97`

8.42.2.3 bitfield `uint32_t dvb_mpeg_es_pic_start::bitfield`

Definition at line 132 of file [mpeg_es.h](#).

8.42.2.4 bitfield2 `uint32_t dvb_mpeg_es_pic_start::bitfield2`

Definition at line 139 of file [mpeg_es.h](#).

8.42.2.5 coding_type `uint32_t dvb_mpeg_es_pic_start::coding_type`

Definition at line 143 of file [mpeg_es.h](#).

8.42.2.6 dummy `uint32_t dvb_mpeg_es_pic_start::dummy`

Definition at line 141 of file [mpeg_es.h](#).

8.42.2.7 sync uint32_t dvb_mpeg_es_pic_start::sync

Definition at line 135 of file [mpeg_es.h](#).

8.42.2.8 temporal_ref uint32_t dvb_mpeg_es_pic_start::temporal_ref

Definition at line 144 of file [mpeg_es.h](#).

8.42.2.9 type uint32_t dvb_mpeg_es_pic_start::type

Definition at line 134 of file [mpeg_es.h](#).

8.42.2.10 vbv_delay uint32_t dvb_mpeg_es_pic_start::vbv_delay

Definition at line 142 of file [mpeg_es.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mpeg_es.h](#)

8.43 dvb_mpeg_es_seq_start Struct Reference

MPEG ES Sequence header.

```
#include <mpeg_es.h>
```

Data Fields

- union {
 - uint32_t [bitfield](#)
 - struct {
 - uint32_t [type](#):8
 - uint32_t [sync](#):24
- ```
}
```
- ```
};
```
- union {
 - uint32_t [bitfield2](#)
 - struct {
 - uint32_t [framerate](#):4
 - uint32_t [aspect](#):4
 - uint32_t [height](#):12
 - uint32_t [width](#):12
- ```
}
```
- ```
};
```

```

• union {
    uint32_t bitfield3
    struct {
        uint32_t qm_nonintra:1
        uint32_t qm_intra:1
        uint32_t constrained:1
        uint32_t vbv:10
        uint32_t one:1
        uint32_t bitrate:18
    }
};

```

8.43.1 Detailed Description

MPEG ES Sequence header.

Parameters

<i>type</i>	DVB_MPEG_ES_SEQ_START
<i>sync</i>	Sync bytes
<i>framerate</i>	Framerate
<i>aspect</i>	Aspect ratio
<i>height</i>	Height
<i>width</i>	Width
<i>qm_nonintra</i>	Load non-intra quantizer matrix
<i>qm_intra</i>	Load intra quantizer matrix
<i>constrained</i>	Constrained parameters flag
<i>vbv</i>	VBV buffer size
<i>one</i>	Should be 1
<i>bitrate</i>	Bitrate

Definition at line 88 of file [mpeg_es.h](#).

8.43.2 Field Documentation

8.43.2.1 union { ... } dvb_mpeg_es_seq_start::@83

8.43.2.2 union { ... } dvb_mpeg_es_seq_start::@85

8.43.2.3 union { ... } dvb_mpeg_es_seq_start::@87

8.43.2.4 aspect `uint32_t dvb_mpeg_es_seq_start::aspect`

Definition at line 100 of file [mpeg_es.h](#).

8.43.2.5 bitfield `uint32_t dvb_mpeg_es_seq_start::bitfield`

Definition at line 90 of file [mpeg_es.h](#).

8.43.2.6 bitfield2 `uint32_t dvb_mpeg_es_seq_start::bitfield2`

Definition at line 97 of file [mpeg_es.h](#).

8.43.2.7 bitfield3 `uint32_t dvb_mpeg_es_seq_start::bitfield3`

Definition at line 106 of file [mpeg_es.h](#).

8.43.2.8 bitrate `uint32_t dvb_mpeg_es_seq_start::bitrate`

Definition at line 113 of file [mpeg_es.h](#).

8.43.2.9 constrained `uint32_t dvb_mpeg_es_seq_start::constrained`

Definition at line 110 of file [mpeg_es.h](#).

8.43.2.10 framerate `uint32_t dvb_mpeg_es_seq_start::framerate`

Definition at line 99 of file [mpeg_es.h](#).

8.43.2.11 height `uint32_t dvb_mpeg_es_seq_start::height`

Definition at line 101 of file [mpeg_es.h](#).

8.43.2.12 one `uint32_t dvb_mpeg_es_seq_start::one`

Definition at line 112 of file [mpeg_es.h](#).

8.43.2.13 qm_intra `uint32_t dvb_mpeg_es_seq_start::qm_intra`

Definition at line 109 of file [mpeg_es.h](#).

8.43.2.14 qm_nonintra `uint32_t dvb_mpeg_es_seq_start::qm_nonintra`

Definition at line 108 of file [mpeg_es.h](#).

8.43.2.15 sync `uint32_t dvb_mpeg_es_seq_start::sync`

Definition at line 93 of file [mpeg_es.h](#).

8.43.2.16 type `uint32_t dvb_mpeg_es_seq_start::type`

Definition at line 92 of file [mpeg_es.h](#).

8.43.2.17 vbv `uint32_t dvb_mpeg_es_seq_start::vbv`

Definition at line 111 of file [mpeg_es.h](#).

8.43.2.18 width `uint32_t dvb_mpeg_es_seq_start::width`

Definition at line 102 of file [mpeg_es.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mpeg_es.h](#)

8.44 dvb_mpeg_pes Struct Reference

MPEG PES data structure.

```
#include <mpeg_pes.h>
```

Data Fields

- union {
 - uint32_t [bitfield](#)
 - struct {
 - uint32_t [stream_id](#):8
 - uint32_t [sync](#):24
- };
- uint16_t [length](#)
- struct [dvb_mpeg_pes_optional](#) [optional](#) []

8.44.1 Detailed Description

MPEG PES data structure.

Parameters

<i>sync</i>	24 bits DVB_MPEG_PES
<i>stream_id</i>	8 bits PES Stream ID
<i>length</i>	16 bits PES packet length
<i>optional</i>	Pointer to optional PES header

Definition at line [186](#) of file [mpeg_pes.h](#).

8.44.2 Field Documentation

8.44.2.1 union { ... } [dvb_mpeg_pes::@115](#)

8.44.2.2 **bitfield** uint32_t [dvb_mpeg_pes::bitfield](#)

Definition at line [188](#) of file [mpeg_pes.h](#).

8.44.2.3 **length** uint16_t [dvb_mpeg_pes::length](#)

Definition at line [194](#) of file [mpeg_pes.h](#).

8.44.2.4 optional struct [dvb_mpeg_pes_optional](#) dvb_mpeg_pes::optional[]

Definition at line 195 of file [mpeg_pes.h](#).

8.44.2.5 stream_id uint32_t dvb_mpeg_pes::stream_id

Definition at line 190 of file [mpeg_pes.h](#).

8.44.2.6 sync uint32_t dvb_mpeg_pes::sync

Definition at line 191 of file [mpeg_pes.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mpeg_pes.h](#)

8.45 dvb_mpeg_pes_optional Struct Reference

MPEG PES optional header.

```
#include <mpeg_pes.h>
```

Data Fields

- union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [PES_extension](#):1
 - uint16_t [PES_CRC](#):1
 - uint16_t [additional_copy_info](#):1
 - uint16_t [DSM_trick_mode](#):1
 - uint16_t [ES_rate](#):1
 - uint16_t [ESCR](#):1
 - uint16_t [PTS_DTS](#):2
 - uint16_t [original_or_copy](#):1
 - uint16_t [copyright](#):1
 - uint16_t [data_alignment_indicator](#):1
 - uint16_t [PES_priority](#):1
 - uint16_t [PES_scrambling_control](#):2
 - uint16_t [two](#):2
- };
- uint8_t [length](#)
- uint64_t [pts](#)
- uint64_t [dts](#)

8.45.1 Detailed Description

MPEG PES optional header.

Parameters

<i>two</i>	2 bits Should be 10
<i>PES_scrambling_control</i>	2 bits PES Scrambling Control (Not Scrambled=00, otherwise scrambled)
<i>PES_priority</i>	1 bit PES Priority
<i>data_alignment_indicator</i>	1 bit PES data alignment
<i>copyright</i>	1 bit PES content protected by copyright
<i>original_or_copy</i>	1 bit PES content is original (=1) or copied (=0)
<i>PTS_DTS</i>	2 bit PES header contains PTS (=10, =11) and/or DTS (=01, =11)
<i>ESCR</i>	1 bit PES header contains ESCR fields
<i>ES_rate</i>	1 bit PES header contains ES_rate field
<i>DSM_trick_mode</i>	1 bit PES header contains DSM_trick_mode field
<i>additional_copy_info</i>	1 bit PES header contains additional_copy_info field
<i>PES_CRC</i>	1 bit PES header contains CRC field
<i>PES_extension</i>	1 bit PES header contains extension field
<i>length</i>	8 bit PES header data length
<i>pts</i>	64 bit PES PTS timestamp
<i>dts</i>	64 bit PES DTS timestamp

Definition at line 152 of file [mpeg_pes.h](#).

8.45.2 Field Documentation

8.45.2.1 `union { ... } dvb_mpeg_pes_optional::@111`

8.45.2.2 **additional_copy_info** `uint16_t dvb_mpeg_pes_optional::additional_copy_info`

Definition at line 158 of file [mpeg_pes.h](#).

8.45.2.3 **bitfield** `uint16_t dvb_mpeg_pes_optional::bitfield`

Definition at line 154 of file [mpeg_pes.h](#).

8.45.2.4 **copyright** `uint16_t dvb_mpeg_pes_optional::copyright`

Definition at line 164 of file [mpeg_pes.h](#).

8.45.2.5 data_alignment_indicator uint16_t dvb_mpeg_pes_optional::data_alignment_indicator

Definition at line 165 of file [mpeg_pes.h](#).

8.45.2.6 DSM_trick_mode uint16_t dvb_mpeg_pes_optional::DSM_trick_mode

Definition at line 159 of file [mpeg_pes.h](#).

8.45.2.7 dts uint64_t dvb_mpeg_pes_optional::dts

Definition at line 173 of file [mpeg_pes.h](#).

8.45.2.8 ES_rate uint16_t dvb_mpeg_pes_optional::ES_rate

Definition at line 160 of file [mpeg_pes.h](#).

8.45.2.9 ESCR uint16_t dvb_mpeg_pes_optional::ESCR

Definition at line 161 of file [mpeg_pes.h](#).

8.45.2.10 length uint8_t dvb_mpeg_pes_optional::length

Definition at line 171 of file [mpeg_pes.h](#).

8.45.2.11 original_or_copy uint16_t dvb_mpeg_pes_optional::original_or_copy

Definition at line 163 of file [mpeg_pes.h](#).

8.45.2.12 PES_CRC uint16_t dvb_mpeg_pes_optional::PES_CRC

Definition at line 157 of file [mpeg_pes.h](#).

8.45.2.13 PES_extension `uint16_t dvb_mpeg_pes_optional::PES_extension`

Definition at line 156 of file [mpeg_pes.h](#).

8.45.2.14 PES_priority `uint16_t dvb_mpeg_pes_optional::PES_priority`

Definition at line 166 of file [mpeg_pes.h](#).

8.45.2.15 PES_scrambling_control `uint16_t dvb_mpeg_pes_optional::PES_scrambling_control`

Definition at line 167 of file [mpeg_pes.h](#).

8.45.2.16 pts `uint64_t dvb_mpeg_pes_optional::pts`

Definition at line 172 of file [mpeg_pes.h](#).

8.45.2.17 PTS_DTS `uint16_t dvb_mpeg_pes_optional::PTS_DTS`

Definition at line 162 of file [mpeg_pes.h](#).

8.45.2.18 two `uint16_t dvb_mpeg_pes_optional::two`

Definition at line 168 of file [mpeg_pes.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mpeg_pes.h](#)

8.46 dvb_mpeg_ts Struct Reference

MPEG TS header.

```
#include <mpeg_ts.h>
```

Data Fields

- uint8_t [sync_byte](#)
- union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [pid](#):13
 - uint16_t [priority](#):1
 - uint16_t [payload_start](#):1
 - uint16_t [tei](#):1
- };
- struct {
 - uint8_t [continuity_counter](#):4
 - uint8_t [payload](#):1
 - uint8_t [adaptation_field](#):1
 - uint8_t [scrambling](#):2
- };
- struct [dvb_mpeg_ts_adaption](#) [adaption](#) []

8.46.1 Detailed Description

MPEG TS header.

Parameters

<i>sync_byte</i>	DVB_MPEG_TS
<i>tei</i>	1 bit Transport Error Indicator
<i>payload_start</i>	1 bit Payload Unit Start Indicator
<i>priority</i>	1 bit Transport Priority
<i>pid</i>	13 bits Packet Identifier
<i>scrambling</i>	2 bits Scrambling control
<i>adaptation_field</i>	1 bit Adaptation field exist
<i>payload</i>	1 bit Contains payload
<i>continuity_counter</i>	4 bits Continuity counter
<i>adaption</i>	Pointer to optional adaption fiels (struct dvb_mpeg_ts_adaption)

Definition at line 101 of file [mpeg_ts.h](#).

8.46.2 Field Documentation**8.46.2.1** union { ... } dvb_mpeg_ts::@121

8.46.2.2 `struct { ... } dvb_mpeg_ts::@123`

8.46.2.3 adaptation_field `uint8_t dvb_mpeg_ts::adaptation_field`

Definition at line [115](#) of file [mpeg_ts.h](#).

8.46.2.4 adaption `struct dvb_mpeg_ts_adaption dvb_mpeg_ts::adaption[]`

Definition at line [118](#) of file [mpeg_ts.h](#).

8.46.2.5 bitfield `uint16_t dvb_mpeg_ts::bitfield`

Definition at line [104](#) of file [mpeg_ts.h](#).

8.46.2.6 continuity_counter `uint8_t dvb_mpeg_ts::continuity_counter`

Definition at line [113](#) of file [mpeg_ts.h](#).

8.46.2.7 payload `uint8_t dvb_mpeg_ts::payload`

Definition at line [114](#) of file [mpeg_ts.h](#).

8.46.2.8 payload_start `uint16_t dvb_mpeg_ts::payload_start`

Definition at line [108](#) of file [mpeg_ts.h](#).

8.46.2.9 pid `uint16_t dvb_mpeg_ts::pid`

Definition at line [106](#) of file [mpeg_ts.h](#).

8.46.2.10 priority uint16_t dvb_mpeg_ts::priority

Definition at line 107 of file [mpeg_ts.h](#).

8.46.2.11 scrambling uint8_t dvb_mpeg_ts::scrambling

Definition at line 116 of file [mpeg_ts.h](#).

8.46.2.12 sync_byte uint8_t dvb_mpeg_ts::sync_byte

Definition at line 102 of file [mpeg_ts.h](#).

8.46.2.13 tei uint16_t dvb_mpeg_ts::tei

Definition at line 109 of file [mpeg_ts.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mpeg_ts.h](#)

8.47 dvb_mpeg_ts_adaption Struct Reference

MPEG TS header adaption field.

```
#include <mpeg_ts.h>
```

Data Fields

- uint8_t [length](#)
- struct {
 - uint8_t [extension](#):1
 - uint8_t [private_data](#):1
 - uint8_t [splicing_point](#):1
 - uint8_t [OPCR](#):1
 - uint8_t [PCR](#):1
 - uint8_t [priority](#):1
 - uint8_t [random_access](#):1
 - uint8_t [discontinued](#):1
- };
- uint8_t [data](#) []

8.47.1 Detailed Description

MPEG TS header adaption field.

Parameters

<i>type</i>	DVB_MPEG_ES_SEQ_START
<i>length</i>	1 bit Adaptation Field Length
<i>discontinued</i>	1 bit Discontinuity indicator
<i>random_access</i>	1 bit Random Access indicator
<i>priority</i>	1 bit Elementary stream priority indicator
<i>PCR</i>	1 bit PCR flag
<i>OPCR</i>	1 bit OPCR flag
<i>splicing_point</i>	1 bit Splicing point flag
<i>private_data</i>	1 bit Transport private data flag
<i>extension</i>	1 bit Adaptation field extension flag
<i>data</i>	Pointer to data

Definition at line 70 of file [mpeg_ts.h](#).

8.47.2 Field Documentation

8.47.2.1 `struct { ... } dvb_mpeg_ts_adaption::@119`

8.47.2.2 data `uint8_t dvb_mpeg_ts_adaption::data[]`

Definition at line 82 of file [mpeg_ts.h](#).

8.47.2.3 discontinued `uint8_t dvb_mpeg_ts_adaption::discontinued`

Definition at line 80 of file [mpeg_ts.h](#).

8.47.2.4 extension `uint8_t dvb_mpeg_ts_adaption::extension`

Definition at line 73 of file [mpeg_ts.h](#).

8.47.2.5 length `uint8_t dvb_mpeg_ts_adaption::length`

Definition at line 71 of file [mpeg_ts.h](#).

8.47.2.6 OPCR `uint8_t dvb_mpeg_ts_adaption::OPCR`

Definition at line 76 of file [mpeg_ts.h](#).

8.47.2.7 PCR `uint8_t dvb_mpeg_ts_adaption::PCR`

Definition at line 77 of file [mpeg_ts.h](#).

8.47.2.8 priority `uint8_t dvb_mpeg_ts_adaption::priority`

Definition at line 78 of file [mpeg_ts.h](#).

8.47.2.9 private_data `uint8_t dvb_mpeg_ts_adaption::private_data`

Definition at line 74 of file [mpeg_ts.h](#).

8.47.2.10 random_access `uint8_t dvb_mpeg_ts_adaption::random_access`

Definition at line 79 of file [mpeg_ts.h](#).

8.47.2.11 splicing_point `uint8_t dvb_mpeg_ts_adaption::splicing_point`

Definition at line 75 of file [mpeg_ts.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mpeg_ts.h](#)

8.48 dvb_open_descriptor Struct Reference

Opaque struct with a DVB open file descriptor.

```
#include <dvb-dev.h>
```

8.48.1 Detailed Description

Opaque struct with a DVB open file descriptor.

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-dev.h](#)

8.49 dvb_parse_file Struct Reference

Describes an entire file format.

```
#include <dvb-file.h>
```

Data Fields

- int [has_delsys_id](#)
- char * [delimiter](#)
- struct [dvb_parse_struct](#) [formats](#) []

8.49.1 Detailed Description

Describes an entire file format.

Parameters

<i>has_delsys_id</i>	A non-zero value indicates that the id field at the formats vector should be used
<i>delimiter</i>	Delimiters to split entries on the format
<i>formats</i>	A struct dvb_parse_struct vector with the per delivery system parsers. This table should terminate with an empty entry.

Definition at line [203](#) of file [dvb-file.h](#).

8.49.2 Field Documentation

8.49.2.1 **delimiter** char* dvb_parse_file::delimiter

Definition at line [205](#) of file [dvb-file.h](#).

8.49.2.2 formats `struct dvb_parse_struct dvb_parse_file::formats[]`

Definition at line 206 of file [dvb-file.h](#).

8.49.2.3 has_delsys_id `int dvb_parse_file::has_delsys_id`

Definition at line 204 of file [dvb-file.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-file.h](#)

8.50 dvb_parse_struct Struct Reference

Describes the format to parse an specific delivery system.

```
#include <dvb-file.h>
```

Data Fields

- `char * id`
- `uint32_t delsys`
- `const struct dvb_parse_table * table`
- `unsigned int size`

8.50.1 Detailed Description

Describes the format to parse an specific delivery system.

Parameters

<i>id</i>	String that identifies the delivery system on the file to be parsed
<i>delsys</i>	Delivery system
<i>table</i>	the struct dvb_parse_table used to parse for this specific delivery system
<i>size</i>	Size of the table

Definition at line 185 of file [dvb-file.h](#).

8.50.2 Field Documentation

8.50.2.1 delsys `uint32_t dvb_parse_struct::delsys`

Definition at line 187 of file [dvb-file.h](#).

8.50.2.2 id `char* dvb_parse_struct::id`

Definition at line 186 of file [dvb-file.h](#).

8.50.2.3 size `unsigned int dvb_parse_struct::size`

Definition at line 189 of file [dvb-file.h](#).

8.50.2.4 table `const struct dvb_parse_table* dvb_parse_struct::table`

Definition at line 188 of file [dvb-file.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-file.h](#)

8.51 dvb_parse_table Struct Reference

Describes the fields to parse on a file.

```
#include <dvb-file.h>
```

Data Fields

- unsigned int [prop](#)
- const char ** [table](#)
- unsigned int [size](#)
- int [mult_factor](#)
- int [has_default_value](#)
- int [default_value](#)

8.51.1 Detailed Description

Describes the fields to parse on a file.

Parameters

<i>prop</i>	Name of the DVBv5 or libdvbv5 property field
<i>table</i>	Name of a translation table for string to int conversion
<i>size</i>	Size of the translation table
<i>mult_factor</i>	Multiply factor - Used, for example, to multiply the symbol rate read from a DVB-S table by 1000.
<i>has_default_value</i>	It is different than zero when the property can be optional. In this case, the next field should be present
<i>default_value</i>	Default value for the optional field

Definition at line 165 of file [dvb-file.h](#).

8.51.2 Field Documentation

8.51.2.1 default_value `int dvb_parse_table::default_value`

Definition at line 171 of file [dvb-file.h](#).

8.51.2.2 has_default_value `int dvb_parse_table::has_default_value`

Definition at line 170 of file [dvb-file.h](#).

8.51.2.3 mult_factor `int dvb_parse_table::mult_factor`

Definition at line 169 of file [dvb-file.h](#).

8.51.2.4 prop `unsigned int dvb_parse_table::prop`

Definition at line 166 of file [dvb-file.h](#).

8.51.2.5 size `unsigned int dvb_parse_table::size`

Definition at line 168 of file [dvb-file.h](#).

8.51.2.6 table `const char** dvb_parse_table::table`

Definition at line 167 of file [dvb-file.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-file.h](#)

8.52 dvb_sat_Inb Struct Reference

Stores the information of a LNBf.

```
#include <dvb-sat.h>
```


Data Structures

- struct [dvbsat_freqrange](#)

Data Fields

- const char * [name](#)
- const char * [alias](#)
- unsigned [lowfreq](#)
- unsigned [highfreq](#)
- unsigned [rangeswitch](#)
- struct [dvb_sat_lnb::dvbsat_freqrange](#) [freqrange](#) [2]

8.52.1 Detailed Description

Stores the information of a LNBf.

Parameters

<i>name</i>	long name of the LNBf type
<i>alias</i>	short name for the LNBf type

The LNBf (low-noise block downconverter) is a type of amplifier that is installed inside the parabolic dishes. It converts the antenna signal to an Intermediate Frequency. Several Ku-band LNBf have more than one IF. The lower IF is stored at [lowfreq](#), the higher IF at [highfreq](#). The exact setup for those structs actually depend on the model of the LNBf, and its usage.

Definition at line [53](#) of file [dvb-sat.h](#).

8.52.2 Field Documentation

8.52.2.1 [alias](#) `const char* dvb_sat_lnb::alias`

Examples

[dvbv5-scan.c](#).

Definition at line [55](#) of file [dvb-sat.h](#).

8.52.2.2 [freqrange](#) `struct dvb_sat_lnb::dvbsat_freqrange dvb_sat_lnb::freqrange[2]`

8.52.2.3 highfreq unsigned dvb_sat_lnb::highfreq

Definition at line 61 of file [dvb-sat.h](#).

8.52.2.4 lowfreq unsigned dvb_sat_lnb::lowfreq

Definition at line 61 of file [dvb-sat.h](#).

8.52.2.5 name const char* dvb_sat_lnb::name

Definition at line 54 of file [dvb-sat.h](#).

8.52.2.6 rangeswitch unsigned dvb_sat_lnb::rangeswitch

Definition at line 62 of file [dvb-sat.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-sat.h](#)

8.53 dvb_table_cat Struct Reference

ATSC CAT table.

```
#include <cat.h>
```

Data Fields

- struct [dvb_table_header](#) header
- struct [dvb_desc](#) * descriptor

8.53.1 Detailed Description

ATSC CAT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>descriptor</i>	pointer to struct dvb_desc

Definition at line 57 of file [cat.h](#).

8.53.2 Field Documentation

8.53.2.1 descriptor `struct dvb_desc* dvb_table_cat::descriptor`

Definition at line 59 of file [cat.h](#).

8.53.2.2 header `struct dvb_table_header dvb_table_cat::header`

Definition at line 58 of file [cat.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/cat.h](#)

8.54 dvb_table_eit Struct Reference

DVB EIT table.

```
#include <eit.h>
```

Data Fields

- struct [dvb_table_header](#) [header](#)
- uint16_t [transport_id](#)
- uint16_t [network_id](#)
- uint8_t [last_segment](#)
- uint8_t [last_table_id](#)
- struct [dvb_table_eit_event](#) * [event](#)

8.54.1 Detailed Description

DVB EIT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>transport_id</i>	transport id
<i>network_id</i>	network id
<i>last_segment</i>	last segment
<i>last_table_id</i>	last table id
<i>event</i>	pointer to struct dvb_table_eit_event

This structure is used to store the original EIT table, converting the integer fields to the CPU endianness.

Everything after `dvb_table_eit::event` (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 146 of file [eit.h](#).

8.54.2 Field Documentation

8.54.2.1 event `struct dvb_table_eit_event* dvb_table_eit::event`

Definition at line 152 of file [eit.h](#).

8.54.2.2 header `struct dvb_table_header dvb_table_eit::header`

Definition at line 147 of file [eit.h](#).

8.54.2.3 last_segment `uint8_t dvb_table_eit::last_segment`

Definition at line 150 of file [eit.h](#).

8.54.2.4 last_table_id `uint8_t dvb_table_eit::last_table_id`

Definition at line 151 of file [eit.h](#).

8.54.2.5 network_id `uint16_t dvb_table_eit::network_id`

Definition at line 149 of file [eit.h](#).

8.54.2.6 transport_id `uint16_t dvb_table_eit::transport_id`

Definition at line 148 of file [eit.h](#).

The documentation for this struct was generated from the following file:

- `lib/include/libdvbv5/eit.h`

8.55 dvb_table_eit_event Struct Reference

DVB EIT event table.

```
#include <eit.h>
```

Data Fields

- uint16_t [event_id](#)
- union {
 - uint16_t [bitfield1](#)
 - uint8_t [dvbstart](#) [5]
- };
- uint8_t [dvbduration](#) [3]
- union {
 - uint16_t [bitfield2](#)
 - struct {
 - uint16_t [desc_length](#):12
 - uint16_t [free_CA_mode](#):1
 - uint16_t [running_status](#):3
- };
- struct [dvb_desc](#) * [descriptor](#)
- struct [dvb_table_eit_event](#) * [next](#)
- struct tm [start](#)
- uint32_t [duration](#)
- uint16_t [service_id](#)

8.55.1 Detailed Description

DVB EIT event table.

Parameters

<i>event_id</i>	an uniquely (inside a service ID) event ID
<i>desc_length</i>	descriptor's length
<i>free_CA_mode</i>	free CA mode. 0 indicates that the event is not scrambled
<i>running_status</i>	running status of the event. The status can be translated to string via dvb_eit_running_status_name string table.
<i>descriptor</i>	pointer to struct dvb_desc
<i>next</i>	pointer to struct dvb_table_eit_event
<i>tm_start</i>	event start (in struct tm format)
<i>duration</i>	duration in seconds
<i>service_id</i>	service ID

This structure is used to store the original EIT event table, converting the integer fields to the CPU endianness, and converting the timestamps to a way that it is better handled on Linux.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after `dvb_table_eit_event::descriptor` (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 105 of file [eit.h](#).

8.55.2 Field Documentation

8.55.2.1 `union { ... } dvb_table_eit_event::@21`

8.55.2.2 `union { ... } dvb_table_eit_event::@23`

8.55.2.3 bitfield1 `uint16_t dvb_table_eit_event::bitfield1`

Definition at line 108 of file [eit.h](#).

8.55.2.4 bitfield2 `uint16_t dvb_table_eit_event::bitfield2`

Definition at line 113 of file [eit.h](#).

8.55.2.5 desc_length `uint16_t dvb_table_eit_event::desc_length`

Definition at line 115 of file [eit.h](#).

8.55.2.6 descriptor `struct dvb_desc* dvb_table_eit_event::descriptor`

Definition at line 120 of file [eit.h](#).

8.55.2.7 duration `uint32_t dvb_table_eit_event::duration`

Definition at line 123 of file [eit.h](#).

8.55.2.8 dvbduration `uint8_t dvb_table_eit_event::dvbduration[3]`

Definition at line 111 of file [eit.h](#).

8.55.2.9 dvbstart `uint8_t dvb_table_eit_event::dvbstart[5]`

Definition at line 109 of file [eit.h](#).

8.55.2.10 event_id `uint16_t dvb_table_eit_event::event_id`

Definition at line 106 of file [eit.h](#).

8.55.2.11 free_CA_mode `uint16_t dvb_table_eit_event::free_CA_mode`

Definition at line 116 of file [eit.h](#).

8.55.2.12 next `struct dvb_table_eit_event* dvb_table_eit_event::next`

Definition at line 121 of file [eit.h](#).

8.55.2.13 running_status `uint16_t dvb_table_eit_event::running_status`

Definition at line 117 of file [eit.h](#).

8.55.2.14 service_id `uint16_t dvb_table_eit_event::service_id`

Definition at line 124 of file [eit.h](#).

8.55.2.15 start `struct tm dvb_table_eit_event::start`

Definition at line 122 of file [eit.h](#).

The documentation for this struct was generated from the following file:

- `lib/include/libdvbv5/eit.h`

8.56 dvb_table_filter Struct Reference

Describes the PES filters used by DVB scan.

```
#include <dvb-scan.h>
```

Data Fields

- unsigned char [tid](#)
- uint16_t [pid](#)
- int [ts_id](#)
- void ** [table](#)
- int [allow_section_gaps](#)
- void * [priv](#)

8.56.1 Detailed Description

Describes the PES filters used by DVB scan.

Parameters

<i>tid</i>	Table ID
<i>pid</i>	Program ID
<i>ts_id</i>	Table section ID (for multisession filtering). If no specific table section is needed, -1 should be used
<i>table</i>	pointer to a pointer for the table struct to be filled
<i>allow_section_gaps</i>	Allow non-continuous section numbering
<i>priv</i>	Internal structure used inside the DVB core. shouldn't be touched externally.

Definition at line [122](#) of file [dvb-scan.h](#).

8.56.2 Field Documentation

8.56.2.1 `allow_section_gaps` `int dvb_table_filter::allow_section_gaps`

Definition at line [129](#) of file [dvb-scan.h](#).

8.56.2.2 `pid` `uint16_t dvb_table_filter::pid`

Definition at line [125](#) of file [dvb-scan.h](#).

8.56.2.3 priv void* dvb_table_filter::priv

Definition at line 136 of file [dvb-scan.h](#).

8.56.2.4 table void** dvb_table_filter::table

Definition at line 127 of file [dvb-scan.h](#).

8.56.2.5 tid unsigned char dvb_table_filter::tid

Definition at line 124 of file [dvb-scan.h](#).

8.56.2.6 ts_id int dvb_table_filter::ts_id

Definition at line 126 of file [dvb-scan.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-scan.h](#)

8.57 dvb_table_header Struct Reference

Header of a MPEG-TS table.

```
#include <header.h>
```

Data Fields

- uint8_t [table_id](#)
 - union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [section_length](#):12
 - uint8_t [one](#):2
 - uint8_t [zero](#):1
 - uint8_t [syntax](#):1
- uint16_t [id](#)
- uint8_t [current_next](#):1
- uint8_t [version](#):5
- uint8_t [one2](#):2
- uint8_t [section_id](#)
- uint8_t [last_section](#)

8.57.1 Detailed Description

Header of a MPEG-TS table.

Parameters

<i>table_id</i>	table id
<i>section_length</i>	section length
<i>syntax</i>	syntax
<i>id</i>	Table ID extension
<i>current_next</i>	current next
<i>version</i>	version
<i>section_id</i>	section number
<i>last_section</i>	last section number

All MPEG-TS tables start with this header.

Definition at line 103 of file [header.h](#).

8.57.2 Field Documentation

8.57.2.1 `union { ... } dvb_table_header::@7`

8.57.2.2 bitfield `uint16_t dvb_table_header::bitfield`

Definition at line 106 of file [header.h](#).

8.57.2.3 current_next `uint8_t dvb_table_header::current_next`

Definition at line 115 of file [header.h](#).

8.57.2.4 id `uint16_t dvb_table_header::id`

Definition at line 114 of file [header.h](#).

8.57.2.5 last_section `uint8_t dvb_table_header::last_section`

Definition at line 120 of file [header.h](#).

8.57.2.6 one `uint8_t dvb_table_header::one`

Definition at line 109 of file [header.h](#).

8.57.2.7 one2 `uint8_t dvb_table_header::one2`

Definition at line 117 of file [header.h](#).

8.57.2.8 section_id `uint8_t dvb_table_header::section_id`

Definition at line 119 of file [header.h](#).

8.57.2.9 section_length `uint16_t dvb_table_header::section_length`

Definition at line 108 of file [header.h](#).

8.57.2.10 syntax `uint8_t dvb_table_header::syntax`

Definition at line 111 of file [header.h](#).

8.57.2.11 table_id `uint8_t dvb_table_header::table_id`

Definition at line 104 of file [header.h](#).

8.57.2.12 version `uint8_t dvb_table_header::version`

Definition at line 116 of file [header.h](#).

8.57.2.13 zero `uint8_t dvb_table_header::zero`

Definition at line 110 of file [header.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/header.h](#)

8.58 dvb_table_nit Struct Reference

MPEG-TS NIT table.

```
#include <nit.h>
```

Data Fields

- struct [dvb_table_header](#) [header](#)
 - union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [desc_length](#):12
 - uint16_t [reserved](#):4
- };
- struct [dvb_desc](#) * [descriptor](#)
- struct [dvb_table_nit_transport](#) * [transport](#)

8.58.1 Detailed Description

MPEG-TS NIT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>desc_length</i>	descriptor length
<i>descriptor</i>	pointer to struct dvb_desc
<i>transport</i>	pointer to struct dvb_table_nit_transport

This structure is used to store the original NIT table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [dvb_table_nit::descriptor](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 143 of file [nit.h](#).

8.58.2 Field Documentation

8.58.2.1 union { ... } dvb_table_nit::@41

8.58.2.2 bitfield `uint16_t dvb_table_nit::bitfield`

Definition at line 146 of file [nit.h](#).

8.58.2.3 desc_length `uint16_t dvb_table_nit::desc_length`

Definition at line 148 of file [nit.h](#).

8.58.2.4 descriptor `struct dvb_desc* dvb_table_nit::descriptor`

Definition at line 152 of file [nit.h](#).

8.58.2.5 header `struct dvb_table_header dvb_table_nit::header`

Definition at line 144 of file [nit.h](#).

8.58.2.6 reserved `uint16_t dvb_table_nit::reserved`

Definition at line 149 of file [nit.h](#).

8.58.2.7 transport `struct dvb_table_nit_transport* dvb_table_nit::transport`

Definition at line 153 of file [nit.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/nit.h](#)

8.59 dvb_table_nit_transport Struct Reference

MPEG-TS NIT transport table.

```
#include <nit.h>
```

Data Fields

- `uint16_t transport_id`
- `uint16_t network_id`
- `union {`
 - `uint16_t bitfield`
 - `struct {`
 - `uint16_t desc_length:12`
 - `uint16_t reserved:4`
- `};`
- `struct dvb_desc * descriptor`
- `struct dvb_table_nit_transport * next`

8.59.1 Detailed Description

MPEG-TS NIT transport table.

Parameters

<i>transport_id</i>	transport id
<i>network_id</i>	network id
<i>desc_length</i>	desc length
<i>descriptor</i>	pointer to struct dvb_desc
<i>next</i>	pointer to struct dvb_table_nit_transport

This structure is used to store the original NIT transport table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [dvb_table_nit_transport::descriptor](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 109 of file [nit.h](#).

8.59.2 Field Documentation

8.59.2.1 `union { ... } dvb_table_nit_transport::@37`

8.59.2.2 bitfield `uint16_t dvb_table_nit_transport::bitfield`

Definition at line 113 of file [nit.h](#).

8.59.2.3 desc_length `uint16_t dvb_table_nit_transport::desc_length`

Definition at line 115 of file [nit.h](#).

8.59.2.4 descriptor `struct dvb_desc* dvb_table_nit_transport::descriptor`

Definition at line 119 of file [nit.h](#).

8.59.2.5 network_id `uint16_t dvb_table_nit_transport::network_id`

Definition at line 111 of file [nit.h](#).

8.59.2.6 next struct [dvb_table_nit_transport*](#) [dvb_table_nit_transport::next](#)

Definition at line [120](#) of file [nit.h](#).

8.59.2.7 reserved [uint16_t](#) [dvb_table_nit_transport::reserved](#)

Definition at line [116](#) of file [nit.h](#).

8.59.2.8 transport_id [uint16_t](#) [dvb_table_nit_transport::transport_id](#)

Definition at line [110](#) of file [nit.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/nit.h](#)

8.60 dvb_table_nit_transport_header Union Reference

MPEG-TS NIT transport header.

```
#include <nit.h>
```

Data Fields

- [uint16_t](#) [bitfield](#)
- struct {
 - [uint16_t](#) [transport_length](#):12
 - [uint16_t](#) [reserved](#):4

8.60.1 Detailed Description

MPEG-TS NIT transport header.

Parameters

<i>transport_length</i>	transport length
-------------------------	------------------

This structure is used to store the original NIT transport header, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Definition at line [79](#) of file [nit.h](#).

8.60.2 Field Documentation

8.60.2.1 `struct { ... } dvb_table_nit_transport_header::@35`

8.60.2.2 bitfield `uint16_t dvb_table_nit_transport_header::bitfield`

Definition at line 80 of file [nit.h](#).

8.60.2.3 reserved `uint16_t dvb_table_nit_transport_header::reserved`

Definition at line 83 of file [nit.h](#).

8.60.2.4 transport_length `uint16_t dvb_table_nit_transport_header::transport_length`

Definition at line 82 of file [nit.h](#).

The documentation for this union was generated from the following file:

- [lib/include/libdvbv5/nit.h](#)

8.61 dvb_table_pat Struct Reference

MPEG-TS PAT table.

```
#include <pat.h>
```

Data Fields

- struct [dvb_table_header](#) header
- `uint16_t` programs
- struct [dvb_table_pat_program](#) * program

8.61.1 Detailed Description

MPEG-TS PAT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>programs</i>	number of programs
<i>program</i>	pointer to struct dvb_table_pat_program

This structure is used to store the original PAT table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after `dvb_table_pat_program::program` (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line [108](#) of file [pat.h](#).

8.61.2 Field Documentation

8.61.2.1 header struct [dvb_table_header](#) `dvb_table_pat::header`

Definition at line [109](#) of file [pat.h](#).

8.61.2.2 program struct [dvb_table_pat_program*](#) `dvb_table_pat::program`

Definition at line [111](#) of file [pat.h](#).

8.61.2.3 programs uint16_t `dvb_table_pat::programs`

Definition at line [110](#) of file [pat.h](#).

The documentation for this struct was generated from the following file:

- `lib/include/libdvbv5/pat.h`

8.62 dvb_table_pat_program Struct Reference

MPEG-TS PAT program table.

```
#include <pat.h>
```

Data Fields

- uint16_t [service_id](#)
 - union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [pid](#):13
 - uint8_t [reserved](#):3
- };
- struct [dvb_table_pat_program](#) * [next](#)

8.62.1 Detailed Description

MPEG-TS PAT program table.

Parameters

<i>service↔ _id</i>	service id
<i>pid</i>	pid
<i>next</i>	pointer to struct dvb_table_pat_program

This structure is used to store the original PAT program table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [dvb_table_pat_program::next](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 77 of file [pat.h](#).

8.62.2 Field Documentation

8.62.2.1 union { ... } [dvb_table_pat_program::@45](#)

8.62.2.2 **bitfield** uint16_t [dvb_table_pat_program::bitfield](#)

Definition at line 80 of file [pat.h](#).

8.62.2.3 next struct [dvb_table_pat_program*](#) dvb_table_pat_program::next

Definition at line 86 of file [pat.h](#).

8.62.2.4 pid uint16_t dvb_table_pat_program::pid

Definition at line 82 of file [pat.h](#).

8.62.2.5 reserved uint8_t dvb_table_pat_program::reserved

Definition at line 83 of file [pat.h](#).

8.62.2.6 service_id uint16_t dvb_table_pat_program::service_id

Definition at line 78 of file [pat.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/pat.h](#)

8.63 dvb_table_pmt Struct Reference

MPEG-TS PMT table.

```
#include <pmt.h>
```

Data Fields

- struct [dvb_table_header](#) header
 - union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [pcr_pid](#):13
 - uint16_t [reserved2](#):3
- };
- union {
 - uint16_t [bitfield2](#)
 - struct {
 - uint16_t [desc_length](#):10
 - uint16_t [zero3](#):2
 - uint16_t [reserved3](#):4
- };
- struct [dvb_desc](#) * [descriptor](#)
 - struct [dvb_table_pmt_stream](#) * [stream](#)

8.63.1 Detailed Description

MPEG-TS PMT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>pcr_pid</i>	PCR PID
<i>desc_length</i>	descriptor length
<i>descriptor</i>	pointer to struct dvb_desc
<i>stream</i>	pointer to struct dvb_table_pmt_stream

This structure is used to store the original PMT stream table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [dvb_table_pmt::descriptor](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 214 of file [pmt.h](#).

8.63.2 Field Documentation

8.63.2.1 `union { ... } dvb_table_pmt::@57`

8.63.2.2 `union { ... } dvb_table_pmt::@59`

8.63.2.3 bitfield `uint16_t dvb_table_pmt::bitfield`

Definition at line 217 of file [pmt.h](#).

8.63.2.4 bitfield2 `uint16_t dvb_table_pmt::bitfield2`

Definition at line 225 of file [pmt.h](#).

8.63.2.5 desc_length `uint16_t dvb_table_pmt::desc_length`

Definition at line 227 of file [pmt.h](#).

8.63.2.6 descriptor `struct dvb_desc* dvb_table_pmt::descriptor`

Definition at line [232](#) of file [pmt.h](#).

8.63.2.7 header `struct dvb_table_header dvb_table_pmt::header`

Definition at line [215](#) of file [pmt.h](#).

8.63.2.8 pcr_pid `uint16_t dvb_table_pmt::pcr_pid`

Definition at line [219](#) of file [pmt.h](#).

8.63.2.9 reserved2 `uint16_t dvb_table_pmt::reserved2`

Definition at line [220](#) of file [pmt.h](#).

8.63.2.10 reserved3 `uint16_t dvb_table_pmt::reserved3`

Definition at line [229](#) of file [pmt.h](#).

8.63.2.11 stream `struct dvb_table_pmt_stream* dvb_table_pmt::stream`

Definition at line [233](#) of file [pmt.h](#).

8.63.2.12 zero3 `uint16_t dvb_table_pmt::zero3`

Definition at line [228](#) of file [pmt.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/pmt.h](#)

8.64 dvb_table_pmt_stream Struct Reference

MPEG-TS PMT stream table.

```
#include <pmt.h>
```

Data Fields

- [uint8_t type](#)
- union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [elementary_pid](#):13
 - uint16_t [reserved](#):3
- };
- union {
 - uint16_t [bitfield2](#)
 - struct {
 - uint16_t [desc_length](#):10
 - uint16_t [zero](#):2
 - uint16_t [reserved2](#):4
- };
- struct [dvb_desc](#) * [descriptor](#)
- struct [dvb_table_pmt_stream](#) * [next](#)

8.64.1 Detailed Description

MPEG-TS PMT stream table.

Parameters

<i>type</i>	stream type
<i>elementary_pid</i>	elementary pid
<i>desc_length</i>	descriptor length
<i>zero</i>	zero
<i>descriptor</i>	pointer to struct dvb_desc
<i>next</i>	pointer to struct dvb_table_pmt_stream

This structure is used to store the original PMT stream table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [dvb_table_pmt_stream::descriptor](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 172 of file [pmt.h](#).

8.64.2 Field Documentation**8.64.2.1** union { ... } dvb_table_pmt_stream::@49

8.64.2.2 `union { ... } dvb_table_pmt_stream::@51`

8.64.2.3 bitfield `uint16_t dvb_table_pmt_stream::bitfield`

Definition at line 175 of file [pmt.h](#).

8.64.2.4 bitfield2 `uint16_t dvb_table_pmt_stream::bitfield2`

Definition at line 182 of file [pmt.h](#).

8.64.2.5 desc_length `uint16_t dvb_table_pmt_stream::desc_length`

Definition at line 184 of file [pmt.h](#).

8.64.2.6 descriptor `struct dvb_desc* dvb_table_pmt_stream::descriptor`

Definition at line 189 of file [pmt.h](#).

8.64.2.7 elementary_pid `uint16_t dvb_table_pmt_stream::elementary_pid`

Definition at line 177 of file [pmt.h](#).

8.64.2.8 next `struct dvb_table_pmt_stream* dvb_table_pmt_stream::next`

Definition at line 190 of file [pmt.h](#).

8.64.2.9 reserved `uint16_t dvb_table_pmt_stream::reserved`

Definition at line 178 of file [pmt.h](#).

8.64.2.10 reserved2 uint16_t dvb_table_pmt_stream::reserved2

Definition at line 186 of file [pmt.h](#).

8.64.2.11 type uint8_t dvb_table_pmt_stream::type

Definition at line 173 of file [pmt.h](#).

8.64.2.12 zero uint16_t dvb_table_pmt_stream::zero

Definition at line 185 of file [pmt.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/pmt.h](#)

8.65 dvb_table_sdt Struct Reference

MPEG-TS SDT table.

```
#include <sdt.h>
```

Data Fields

- struct [dvb_table_header](#) header
- uint16_t [network_id](#)
- uint8_t [reserved](#)
- struct [dvb_table_sdt_service](#) * service

8.65.1 Detailed Description

MPEG-TS SDT table.

Parameters

<i>header</i>	struct dvb_table_header content
<i>network↔ _id</i>	network id
<i>service</i>	pointer to struct dvb_table_sdt_service

This structure is used to store the original SDT table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after `dvb_table_sdt::service` (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 123 of file `sdt.h`.

8.65.2 Field Documentation

8.65.2.1 header `struct dvb_table_header dvb_table_sdt::header`

Definition at line 124 of file `sdt.h`.

8.65.2.2 network_id `uint16_t dvb_table_sdt::network_id`

Definition at line 125 of file `sdt.h`.

8.65.2.3 reserved `uint8_t dvb_table_sdt::reserved`

Definition at line 126 of file `sdt.h`.

8.65.2.4 service `struct dvb_table_sdt_service* dvb_table_sdt::service`

Definition at line 127 of file `sdt.h`.

The documentation for this struct was generated from the following file:

- `lib/include/libdvbv5/sdt.h`

8.66 dvb_table_sdt_service Struct Reference

MPEG-TS SDT service table.

```
#include <sdt.h>
```

Data Fields

- uint16_t [service_id](#)
 - uint8_t [EIT_present_following](#):1
 - uint8_t [EIT_schedule](#):1
 - uint8_t [reserved](#):6
 - union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [desc_length](#):12
 - uint16_t [free_CA_mode](#):1
 - uint16_t [running_status](#):3
- struct [dvb_desc](#) * [descriptor](#)
- struct [dvb_table_sdt_service](#) * [next](#)

8.66.1 Detailed Description

MPEG-TS SDT service table.

Parameters

<i>service_id</i>	service id
<i>EIT_present_following</i>	EIT present following
<i>EIT_schedule</i>	EIT schedule
<i>desc_length</i>	desc length
<i>free_CA_mode</i>	free CA mode
<i>running_status</i>	running status
<i>descriptor</i>	pointer to struct dvb_desc
<i>next</i>	pointer to struct dvb_table_sdt_service

This structure is used to store the original SDT service table, converting the integer fields to the CPU endianness.

The undocumented parameters are used only internally by the API and/or are fields that are reserved. They shouldn't be used, as they may change on future API releases.

Everything after [dvb_table_sdt_service::descriptor](#) (including it) won't be bit-mapped to the data parsed from the MPEG TS. So, metadata are added there.

Definition at line 87 of file [sdt.h](#).

8.66.2 Field Documentation**8.66.2.1** union { ... } dvb_table_sdt_service::@65

8.66.2.2 bitfield `uint16_t dvb_table_sdt_service::bitfield`

Definition at line 93 of file [sdt.h](#).

8.66.2.3 desc_length `uint16_t dvb_table_sdt_service::desc_length`

Definition at line 95 of file [sdt.h](#).

8.66.2.4 descriptor `struct dvb_desc* dvb_table_sdt_service::descriptor`

Definition at line 100 of file [sdt.h](#).

8.66.2.5 EIT_present_following `uint8_t dvb_table_sdt_service::EIT_present_following`

Definition at line 89 of file [sdt.h](#).

8.66.2.6 EIT_schedule `uint8_t dvb_table_sdt_service::EIT_schedule`

Definition at line 90 of file [sdt.h](#).

8.66.2.7 free_CA_mode `uint16_t dvb_table_sdt_service::free_CA_mode`

Definition at line 96 of file [sdt.h](#).

8.66.2.8 next `struct dvb_table_sdt_service* dvb_table_sdt_service::next`

Definition at line 101 of file [sdt.h](#).

8.66.2.9 reserved `uint8_t dvb_table_sdt_service::reserved`

Definition at line 91 of file [sdt.h](#).

8.66.2.10 running_status uint16_t dvb_table_sdt_service::running_status

Definition at line 97 of file [sdt.h](#).

8.66.2.11 service_id uint16_t dvb_table_sdt_service::service_id

Definition at line 88 of file [sdt.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/sdt.h](#)

8.67 dvb_ts_packet_header Struct Reference

Header of a MPEG-TS transport packet.

```
#include <header.h>
```

Data Fields

- uint8_t [sync_byte](#)
 - union {
 - uint16_t [bitfield](#)
 - struct {
 - uint16_t [pid](#):13
 - uint16_t [transport_priority](#):1
 - uint16_t [payload_unit_start_indicator](#):1
 - uint16_t [transport_error_indicator](#):1
- };
- uint8_t [continuity_counter](#):4
- uint8_t [adaptation_field_control](#):2
- uint8_t [transport_scrambling_control](#):2
- uint8_t [adaptation_field_length](#)
- struct {
 - uint8_t [extension](#):1
 - uint8_t [private_data](#):1
 - uint8_t [splicing_point](#):1
 - uint8_t [OPCR](#):1
 - uint8_t [PCR](#):1
 - uint8_t [priority](#):1
 - uint8_t [random_access](#):1
 - uint8_t [discontinued](#):1
- };

8.67.1 Detailed Description

Header of a MPEG-TS transport packet.

Parameters

<i>sync_byte</i>	sync byte
<i>pid</i>	Program ID
<i>transport_priority</i>	transport priority
<i>payload_unit_start_indicator</i>	payload unit start indicator
<i>transport_error_indicator</i>	transport error indicator
<i>continuity_counter</i>	continuity counter
<i>adaptation_field_control</i>	adaptation field control
<i>transport_scrambling_control</i>	transport scrambling control
<i>adaptation_field_length</i>	adaptation field length

See also

[http://www.etherguidesystems.com/Help/SDOs/MPEG/Semantics/MPEG-2/transport←
_packet.aspx](http://www.etherguidesystems.com/Help/SDOs/MPEG/Semantics/MPEG-2/transport_packet.aspx)

Examples

[dvbv5-zap.c](#).

Definition at line 57 of file [header.h](#).

8.67.2 Field Documentation

8.67.2.1 `union { ... } dvb_ts_packet_header::@1`

8.67.2.2 `struct { ... } dvb_ts_packet_header::@3`

8.67.2.3 adaptation_field_control `uint8_t dvb_ts_packet_header::adaptation_field_control`

Examples

[dvbv5-zap.c](#).

Definition at line 69 of file [header.h](#).

8.67.2.4 adaptation_field_length `uint8_t dvb_ts_packet_header::adaptation_field_length`

Examples

[dvbv5-zap.c](#).Definition at line 73 of file [header.h](#).**8.67.2.5 bitfield** `uint16_t dvb_ts_packet_header::bitfield`

Examples

[dvbv5-zap.c](#).Definition at line 60 of file [header.h](#).**8.67.2.6 continuity_counter** `uint8_t dvb_ts_packet_header::continuity_counter`

Examples

[dvbv5-zap.c](#).Definition at line 68 of file [header.h](#).**8.67.2.7 discontinued** `uint8_t dvb_ts_packet_header::discontinued`

Examples

[dvbv5-zap.c](#).Definition at line 83 of file [header.h](#).**8.67.2.8 extension** `uint8_t dvb_ts_packet_header::extension`Definition at line 76 of file [header.h](#).**8.67.2.9 OPCR** `uint8_t dvb_ts_packet_header::OPCR`Definition at line 79 of file [header.h](#).

8.67.2.10 payload_unit_start_indicator `uint16_t dvb_ts_packet_header::payload_unit_start_↵
indicator`

Definition at line 64 of file [header.h](#).

8.67.2.11 PCR `uint8_t dvb_ts_packet_header::PCR`

Definition at line 80 of file [header.h](#).

8.67.2.12 pid `uint16_t dvb_ts_packet_header::pid`

Examples

[dvbv5-zap.c](#).

Definition at line 62 of file [header.h](#).

8.67.2.13 priority `uint8_t dvb_ts_packet_header::priority`

Definition at line 81 of file [header.h](#).

8.67.2.14 private_data `uint8_t dvb_ts_packet_header::private_data`

Definition at line 77 of file [header.h](#).

8.67.2.15 random_access `uint8_t dvb_ts_packet_header::random_access`

Definition at line 82 of file [header.h](#).

8.67.2.16 splicing_point `uint8_t dvb_ts_packet_header::splicing_point`

Definition at line 78 of file [header.h](#).

8.67.2.17 sync_byte `uint8_t dvb_ts_packet_header::sync_byte`

Examples

[dvbv5-zap.c](#).

Definition at line 58 of file [header.h](#).

8.67.2.18 transport_error_indicator `uint16_t dvb_ts_packet_header::transport_error_indicator`

Definition at line 65 of file [header.h](#).

8.67.2.19 transport_priority `uint16_t dvb_ts_packet_header::transport_priority`

Definition at line 63 of file [header.h](#).

8.67.2.20 transport_scrambling_control `uint8_t dvb_ts_packet_header::transport_scrambling_↵
control`

Definition at line 70 of file [header.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/header.h](#)

8.68 dvb_v5_descriptors Struct Reference

Contains the descriptors needed to scan the Service ID and other relevant info at a MPEG-TS Digital TV stream.

```
#include <dvb-scan.h>
```

Data Fields

- `uint32_t` [delivery_system](#)
- `struct dvb_entry *` [entry](#)
- `unsigned` [num_entry](#)
- `struct dvb_table_pat *` [pat](#)
- `struct atsc_table_vct *` [vct](#)
- `struct dvb_v5_descriptors_program *` [program](#)
- `struct dvb_table_nit *` [nit](#)
- `struct dvb_table_sdt *` [sdt](#)
- `unsigned` [num_program](#)
- `struct dvb_table_nit **` [other_nits](#)
- `unsigned` [num_other_nits](#)
- `struct dvb_table_sdt **` [other_sdts](#)
- `unsigned` [num_other_sdts](#)

8.68.1 Detailed Description

Contains the descriptors needed to scan the Service ID and other relevant info at a MPEG-TS Digital TV stream.

Parameters

<i>delivery_system</i>	Delivery system of the parsed MPEG-TS
<i>entry</i>	struct dvb_entry pointer (see dvb-file.h)
<i>pat</i>	PAT table descriptor pointer (table ID 0x00).
<i>vct</i>	VCT table descriptor pointer (either table ID 0xc8, for TVCT or table ID 0xc9, for CVCT)
<i>program</i>	PAT/PMT array associated programs found at MPEG-TS
<i>num_program</i>	Number of program entries at program array.
<i>nit</i>	NIT table descriptor pointer for table ID 0x40.
<i>sdt</i>	SDT table descriptor pointer for table ID 0x42.
<i>other_nits</i>	Contains an array of pointers to the other NIT extension tables identified by table ID 0x41.
<i>num_other_nits</i>	Number of NIT tables at other_nits array.
<i>other_sdt</i>	Contains an array of pointers to the other NIT extension tables identified by table ID 0x46.
<i>num_other_sdt</i>	Number of NIT tables at other_sdt array.

Those descriptors are filled by the scan routines when the tables are found. Otherwise, they're NULL.

Note

: Never alloc this struct yourself. This is meant to always be allocated via [dvb_scan_alloc_handler_table\(\)](#) or via [dvb_get_ts_tables\(\)](#).

Examples

[dvbv5-scan.c](#).

Definition at line 87 of file [dvb-scan.h](#).

8.68.2 Field Documentation

8.68.2.1 **delivery_system** `uint32_t dvb_v5_descriptors::delivery_system`

Definition at line 88 of file [dvb-scan.h](#).

8.68.2.2 **entry** `struct dvb_entry* dvb_v5_descriptors::entry`

Examples

[dvbv5-scan.c](#).

Definition at line 90 of file [dvb-scan.h](#).

8.68.2.3 nit struct [dvb_table_nit*](#) dvb_v5_descriptors::nit

Definition at line 96 of file [dvb-scan.h](#).

8.68.2.4 num_entry unsigned dvb_v5_descriptors::num_entry

Definition at line 91 of file [dvb-scan.h](#).

8.68.2.5 num_other_nits unsigned dvb_v5_descriptors::num_other_nits

Definition at line 102 of file [dvb-scan.h](#).

8.68.2.6 num_other_sdts unsigned dvb_v5_descriptors::num_other_sdts

Definition at line 105 of file [dvb-scan.h](#).

8.68.2.7 num_program unsigned dvb_v5_descriptors::num_program

Definition at line 99 of file [dvb-scan.h](#).

8.68.2.8 other_nits struct [dvb_table_nit**](#) dvb_v5_descriptors::other_nits

Definition at line 101 of file [dvb-scan.h](#).

8.68.2.9 other_sdts struct [dvb_table_sdt**](#) dvb_v5_descriptors::other_sdts

Definition at line 104 of file [dvb-scan.h](#).

8.68.2.10 pat struct [dvb_table_pat*](#) dvb_v5_descriptors::pat

Definition at line 93 of file [dvb-scan.h](#).

8.68.2.11 program `struct dvb_v5_descriptors_program* dvb_v5_descriptors::program`

Definition at line [95](#) of file [dvb-scan.h](#).

8.68.2.12 sdt `struct dvb_table_sdt* dvb_v5_descriptors::sdt`

Definition at line [97](#) of file [dvb-scan.h](#).

8.68.2.13 vct `struct atsc_table_vct* dvb_v5_descriptors::vct`

Definition at line [94](#) of file [dvb-scan.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-scan.h](#)

8.69 dvb_v5_descriptors_program Struct Reference

Associates PMT with PAT tables.

```
#include <dvb-scan.h>
```

Data Fields

- struct [dvb_table_pat_program](#) * [pat_pgm](#)
- struct [dvb_table_pmt](#) * [pmt](#)

8.69.1 Detailed Description

Associates PMT with PAT tables.

Parameters

<i>pat_pgm</i>	pointer for PAT descriptor
<i>pmt</i>	pointer for PMT descriptor

Definition at line [55](#) of file [dvb-scan.h](#).

8.69.2 Field Documentation

8.69.2.1 pat_pgm struct [dvb_table_pat_program](#)* dvb_v5_descriptors_program::pat_pgm

Definition at line 56 of file [dvb-scan.h](#).

8.69.2.2 pmt struct [dvb_table_pmt](#)* dvb_v5_descriptors_program::pmt

Definition at line 57 of file [dvb-scan.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[dvb-scan.h](#)

8.70 dvb_v5_fe_parms Struct Reference

Keeps data needed to handle the DVB frontend.

```
#include <dvb-fe.h>
```

Data Fields

- struct dvb_frontend_info [info](#)
- uint32_t [version](#)
- int [has_v5_stats](#)
- fe_delivery_system_t [current_sys](#)
- int [num_systems](#)
- fe_delivery_system_t [systems](#) [MAX_DELIVERY_SYSTEMS]
- int [legacy_fe](#)
- int [abort](#)
- int [lna](#)
- const struct [dvb_sat_lnb](#) * [lnb](#)
- int [sat_number](#)
- unsigned [freq_bpf](#)
- unsigned [diseqc_wait](#)
- unsigned [verbose](#)
- [dvb_logfunc](#) [logfunc](#)
- char * [default_charset](#)
- char * [output_charset](#)

Parameters

8.70.1 Detailed Description

Keeps data needed to handle the DVB frontend.

Parameters

<i>info</i>	Contains the DVB info properties (RO)
<i>version</i>	Version of the Linux DVB API (RO)
<i>has_v5_stats</i>	A value different than 0 indicates that the frontend supports DVBv5 stats (RO)
<i>current_sys</i>	Currently selected delivery system (RO)
<i>num_systems</i>	Number of delivery systems (RO)
<i>systems</i>	Delivery systems supported by the hardware (RO)
<i>legacy_fe</i>	A value different than 0 indicates a legacy Kernel driver using DVBv3 API only, or that DVBv3 only mode was forced by the client (RO)
<i>abort</i>	Client should set it to abort a pending operation like DTV scan (RW)
<i>lna</i>	Sets the LNA mode 0 disables; 1 enables, -1 uses auto mode (RW)
<i>lnb</i>	LNBf description (RW)
<i>sat_number</i>	Number of the satellite (used by DISEqC setup) (RW)
<i>freq_bpf</i>	SCR/Unicable band-pass filter frequency to use, in kHz
<i>verbose</i>	Verbosity level of the library (RW)
<i>dvb_logfunc</i>	Function used to write log messages (RO)
<i>default_charset</i>	Name of the charset used by the DVB standard (RW)
<i>output_charset</i>	Name of the charset to output (system specific) (RW)

The fields marked as RO should not be changed by the client, as otherwise undesired effects may happen. The ones marked as RW are ok to either read or write by the client.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 117 of file [dvb-fe.h](#).

8.70.2 Field Documentation

8.70.2.1 **abort** `int dvb_v5_fe_parms::abort`

Examples

[dvbv5-scan.c](#).

Definition at line 130 of file [dvb-fe.h](#).

8.70.2.2 current_sys `fe_delivery_system_t dvb_v5_fe_parms::current_sys`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 122 of file [dvb-fe.h](#).

8.70.2.3 default_charset `char* dvb_v5_fe_parms::default_charset`

Definition at line 146 of file [dvb-fe.h](#).

8.70.2.4 diseqc_wait `unsigned dvb_v5_fe_parms::diseqc_wait`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 139 of file [dvb-fe.h](#).

8.70.2.5 freq_bpf `unsigned dvb_v5_fe_parms::freq_bpf`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 138 of file [dvb-fe.h](#).

8.70.2.6 has_v5_stats `int dvb_v5_fe_parms::has_v5_stats`

Definition at line 121 of file [dvb-fe.h](#).

8.70.2.7 info `struct dvb_frontend_info dvb_v5_fe_parms::info`

Definition at line 119 of file [dvb-fe.h](#).

8.70.2.8 legacy_fe `int dvb_v5_fe_parms::legacy_fe`

Definition at line 125 of file [dvb-fe.h](#).

8.70.2.9 lna `int dvb_v5_fe_parms::lna`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 133 of file [dvb-fe.h](#).

8.70.2.10 lnb `const struct dvb_sat_lnb* dvb_v5_fe_parms::lnb`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 136 of file [dvb-fe.h](#).

8.70.2.11 logfunc `dvb_logfunc dvb_v5_fe_parms::logfunc`

Definition at line 143 of file [dvb-fe.h](#).

8.70.2.12 num_systems `int dvb_v5_fe_parms::num_systems`

Definition at line 123 of file [dvb-fe.h](#).

8.70.2.13 output_charset `char* dvb_v5_fe_parms::output_charset`

Definition at line 147 of file [dvb-fe.h](#).

8.70.2.14 sat_number `int dvb_v5_fe_parms::sat_number`

Examples

[dvbv5-scan.c](#), and [dvbv5-zap.c](#).

Definition at line 137 of file [dvb-fe.h](#).

8.70.2.15 systems `fe_delivery_system_t dvb_v5_fe_parms::systems[MAX_DELIVERY_SYSTEMS]`

Definition at line [124](#) of file [dvb-fe.h](#).

8.70.2.16 verbose `unsigned dvb_v5_fe_parms::verbose`

Examples

[dvb-fe-tool.c](#).

Definition at line [142](#) of file [dvb-fe.h](#).

8.70.2.17 version `uint32_t dvb_v5_fe_parms::version`

Definition at line [120](#) of file [dvb-fe.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/dvb-fe.h](#)

8.71 dvb_sat_lnb::dvbsat_freqrange Struct Reference

```
#include <dvb-sat.h>
```

Data Fields

- unsigned [low](#)
- unsigned [high](#)

8.71.1 Detailed Description

Definition at line [63](#) of file [dvb-sat.h](#).

8.71.2 Field Documentation

8.71.2.1 high `unsigned dvb_sat_lnb::dvbsat_freqrange::high`

Definition at line [64](#) of file [dvb-sat.h](#).

8.71.2.2 low unsigned dvb_sat_lnb::dvbsat_fregrange::low

Definition at line 64 of file [dvb-sat.h](#).

The documentation for this struct was generated from the following file:

- lib/include/libdvbv5/[dvb-sat.h](#)

8.72 isdb_desc_partial_reception Struct Reference

Structure containing the partial reception descriptor.

```
#include <desc_partial_reception.h>
```

Data Fields

- uint8_t [type](#)
- uint8_t [length](#)
- struct [dvb_desc](#) * [next](#)
- struct [isdb_partial_reception_service_id](#) * [partial_reception](#)

8.72.1 Detailed Description

Structure containing the partial reception descriptor.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>partial_reception</i>	vector of struct isdb_partial_reception_service_id . The length of the vector is given by: length / sizeof(struct isdb_partial_reception_service_id).

Definition at line 71 of file [desc_partial_reception.h](#).

8.72.2 Field Documentation

8.72.2.1 length uint8_t isdb_desc_partial_reception::length

Definition at line 73 of file [desc_partial_reception.h](#).

8.72.2.2 next struct [dvb_desc](#)* isdb_desc_partial_reception::next

Definition at line 74 of file [desc_partial_reception.h](#).

8.72.2.3 partial_reception struct [isdb_partial_reception_service_id](#)* isdb_desc_partial_reception↔
::partial_reception

Definition at line 76 of file [desc_partial_reception.h](#).

8.72.2.4 type uint8_t isdb_desc_partial_reception::type

Definition at line 72 of file [desc_partial_reception.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_partial_reception.h](#)

8.73 isdb_partial_reception_service_id Struct Reference

Service ID that uses partial reception.

```
#include <desc_partial_reception.h>
```

Data Fields

- uint16_t [service_id](#)

8.73.1 Detailed Description

Service ID that uses partial reception.

Parameters

<i>service↔ _id</i>	service id
-------------------------	------------

Definition at line 55 of file [desc_partial_reception.h](#).

8.73.2 Field Documentation

8.73.2.1 service_id `uint16_t isdb_partial_reception_service_id::service_id`

Definition at line 56 of file [desc_partial_reception.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_partial_reception.h](#)

8.74 isdbt_desc_terrestrial_delivery_system Struct Reference

Struct containing the ISDB-T terrestrial delivery system.

```
#include <desc_isdbt_delivery.h>
```

Data Fields

- `uint8_t type`
- `uint8_t length`
- `struct dvb_desc * next`
- `uint32_t * frequency`
- `unsigned num_freqs`
- `union {
 uint16_t bitfield
 struct {
 uint16_t transmission_mode:2
 uint16_t guard_interval:2
 uint16_t area_code:12
 }
};`

8.74.1 Detailed Description

Struct containing the ISDB-T terrestrial delivery system.

Parameters

<i>type</i>	descriptor tag
<i>length</i>	descriptor length
<i>next</i>	pointer to struct dvb_desc
<i>area_code</i>	area code. The area code definition varies from Country to Country.
<i>guard_interval</i>	guard interval
<i>transmission_mode</i>	transmission mode
<i>frequency</i>	vector with center frequencies
<i>num_freqs</i>	number of frequencies at the isdbt_desc_terrestrial_delivery_system::frequency vector

Definition at line 57 of file [desc_isdbt_delivery.h](#).

8.74.2 Field Documentation

8.74.2.1 `union { ... } isdbt_desc_terrestrial_delivery_system::@155`

8.74.2.2 area_code `uint16_t isdbt_desc_terrestrial_delivery_system::area_code`

Definition at line 70 of file [desc_isdbt_delivery.h](#).

8.74.2.3 bitfield `uint16_t isdbt_desc_terrestrial_delivery_system::bitfield`

Definition at line 66 of file [desc_isdbt_delivery.h](#).

8.74.2.4 frequency `uint32_t* isdbt_desc_terrestrial_delivery_system::frequency`

Definition at line 62 of file [desc_isdbt_delivery.h](#).

8.74.2.5 guard_interval `uint16_t isdbt_desc_terrestrial_delivery_system::guard_interval`

Definition at line 69 of file [desc_isdbt_delivery.h](#).

8.74.2.6 length `uint8_t isdbt_desc_terrestrial_delivery_system::length`

Definition at line 59 of file [desc_isdbt_delivery.h](#).

8.74.2.7 next `struct dvb_desc* isdbt_desc_terrestrial_delivery_system::next`

Definition at line 60 of file [desc_isdbt_delivery.h](#).

8.74.2.8 num_freqs `unsigned isdbt_desc_terrestrial_delivery_system::num_freqs`

Definition at line 63 of file [desc_isdbt_delivery.h](#).

8.74.2.9 transmission_mode `uint16_t isdbt_desc_terrestrial_delivery_system::transmission_mode`

Definition at line 68 of file [desc_isdbt_delivery.h](#).

8.74.2.10 type `uint8_t isdbt_desc_terrestrial_delivery_system::type`

Definition at line 58 of file [desc_isdbt_delivery.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/desc_isdbt_delivery.h](#)

8.75 ts_t Struct Reference

MPEG PES timestamp structure, used for dts and pts.

```
#include <mpeg_pes.h>
```

Data Fields

- `uint8_t one`:1
 - `uint8_t bits30`:3
 - `uint8_t tag`:4
 - union {
 - `uint16_t bitfield`
 - struct {
 - `uint16_t one1`:1
 - `uint16_t bits15`:15
- ```
};
```
- union {
    - `uint16_t bitfield2`
    - struct {
      - `uint16_t one2`:1
      - `uint16_t bits00`:15
- ```
};
```

8.75.1 Detailed Description

MPEG PES timestamp structure, used for dts and pts.

Parameters

<i>tag</i>	4 bits Should be 0010 for PTS and 0011 for DTS
<i>bits30</i>	3 bits Timestamp bits 30-32
<i>one</i>	1 bit Sould be 1
<i>bits15</i>	15 bits Timestamp bits 15-29
<i>one1</i>	1 bit Should be 1
<i>bits00</i>	15 Bits Timestamp bits 0-14
<i>one2</i>	1 bit Should be 1

Definition at line 108 of file [mpeg_pes.h](#).

8.75.2 Field Documentation

8.75.2.1 `union { ... } ts_t::@103`

8.75.2.2 `union { ... } ts_t::@105`

8.75.2.3 bitfield `uint16_t ts_t::bitfield`

Definition at line 114 of file [mpeg_pes.h](#).

8.75.2.4 bitfield2 `uint16_t ts_t::bitfield2`

Definition at line 122 of file [mpeg_pes.h](#).

8.75.2.5 bits00 `uint16_t ts_t::bits00`

Definition at line 125 of file [mpeg_pes.h](#).

8.75.2.6 bits15 `uint16_t ts_t::bits15`

Definition at line 117 of file [mpeg_pes.h](#).

8.75.2.7 bits30 `uint8_t ts_t::bits30`

Definition at line 110 of file [mpeg_pes.h](#).

8.75.2.8 one `uint8_t ts_t::one`

Definition at line 109 of file [mpeg_pes.h](#).

8.75.2.9 one1 `uint16_t ts_t::one1`

Definition at line 116 of file [mpeg_pes.h](#).

8.75.2.10 one2 `uint16_t ts_t::one2`

Definition at line 124 of file [mpeg_pes.h](#).

8.75.2.11 tag `uint8_t ts_t::tag`

Definition at line 111 of file [mpeg_pes.h](#).

The documentation for this struct was generated from the following file:

- [lib/include/libdvbv5/mpeg_pes.h](#)

9 File Documentation

9.1 doc/libdvbv5-index.doc File Reference

9.2 lib/include/libdvbv5/atsc_eit.h File Reference

Provides the table parser for the ATSC EIT (Event Information Table)

```
#include <stdint.h>
#include <unistd.h>
#include <time.h>
#include <libdvbv5/atsc_header.h>
```

Data Structures

- struct [atsc_table_eit_event](#)
ATSC EIT event table.
- union [atsc_table_eit_desc_length](#)
ATSC EIT descriptor length.
- struct [atsc_table_eit](#)
ATSC EIT table.

Macros

- #define [ATSC_TABLE_EIT](#)
ATSC EIT table ID.
- #define [atsc_eit_event_foreach](#)(_event, _eit)
Macro used to find event on an ATSC EIT table.

Functions

- `ssize_t atsc_table_eit_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, struct `atsc_table_eit` **table)
Initializes and parses ATSC EIT table.
- void `atsc_table_eit_free` (struct `atsc_table_eit` *table)
Frees all data allocated by the ATSC EIT table parser.
- void `atsc_table_eit_print` (struct `dvb_v5_fe_parms` *parms, struct `atsc_table_eit` *table)
Prints the content of the ATSC EIT table.
- void `atsc_time` (const uint32_t start_time, struct tm *tm)
Converts an ATSC EIT formatted timestamp into struct tm.

9.2.1 Detailed Description

Provides the table parser for the ATSC EIT (Event Information Table)

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The table described herein is defined at:

- ATSC A/65:2009

See also

<http://www.etherguidesystems.com/help/sdos/atsc/syntax/tablesections/eitks.aspx>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [atsc_eit.h](#).

9.3 atsc_eit.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00039 #ifndef _ATSC_EIT_H
00040 #define _ATSC_EIT_H
00041
00042 #include <stdint.h>
00043 #include <unistd.h> /* ssize_t */
00044 #include <time.h>
00045
00046 #include <libdvbv5/atsc_header.h>
00047
00053 #define ATSC_TABLE_EIT 0xCB
00054
00081 struct atsc_table_eit_event {
00082     union {
00083         uint16_t bitfield;
00084         struct {
00085             uint16_t event_id:14;
00086             uint16_t one:2;
00087         } __attribute__((packed));
00088     } __attribute__((packed));
00089     uint32_t start_time;
00090     union {
00091         uint32_t bitfield2;
00092         struct {
00093             uint32_t title_length:8;
00094             uint32_t duration:20;
00095             uint32_t etm:2;
00096             uint32_t one2:2;
00097             uint32_t :2;
00098         } __attribute__((packed));
00099     } __attribute__((packed));
00100     struct dvb_desc *descriptor;
00101     struct atsc_table_eit_event *next;
00102     struct tm start;
00103     uint16_t source_id;
00104 } __attribute__((packed));
00105
00121 union atsc_table_eit_desc_length {
00122     uint16_t bitfield;
00123     struct {
00124         uint16_t desc_length:12;
00125         uint16_t reserved:4;
00126     } __attribute__((packed));
00127 } __attribute__((packed));
00128
00146 struct atsc_table_eit {
00147     struct dvb_table_header header;
00148     uint8_t protocol_version;
00149     uint8_t events;
00150     struct atsc_table_eit_event *event;
00151 } __attribute__((packed));
00152
00160 #define atsc_eit_event_foreach(_event, _eit) \
00161     if (_eit && _eit->event) \
00162         for( struct atsc_table_eit_event *_event = _eit->event; _event; _event = _event->next
00163 ) \
00164 struct dvb_v5_fe_parms;
00165
00166 #ifdef __cplusplus
00167 extern "C" {
00168 #endif
00169
00186 ssize_t atsc_table_eit_init(struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00187                             ssize_t buflen, struct atsc_table_eit **table);

```

```
00188
00195 void atsc_table_eit_free(struct atsc_table_eit *table);
00196
00204 void atsc_table_eit_print(struct dvb_v5_fe_parms *parms,
00205                          struct atsc_table_eit *table);
00206
00215 void atsc_time(const uint32_t start_time, struct tm *tm);
00216
00217 #ifdef __cplusplus
00218 }
00219 #endif
00220
00221 #endif
```

9.4 lib/include/libdvbv5/atsc_header.h File Reference

Provides some common ATSC stuff.

```
#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/header.h>
```

Macros

- `#define ATSC_BASE_PID`
ATSC PID for the Program and System Information Protocol.

9.4.1 Detailed Description

Provides some common ATSC stuff.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [atsc_header.h](#).

9.5 atsc_header.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00020 #ifndef _ATSC_HEADER_H
00021 #define _ATSC_HEADER_H
00022
00034 #include <stdint.h>
00035 #include <unistd.h> /* ssize_t */
00036
00037 #include <libdvbv5/header.h>
00038
00044 #define ATSC_BASE_PID 0x1FFB
00045
00046 #ifndef _DOXYGEN
00047
00048 /* Deprecated, as it causes troubles with doxygen */
00049 #define ATSC_HEADER() \
00050     struct dvb_table_header header; \
00051     uint8_t protocol_version; \
00052
00053 #define ATSC_TABLE_HEADER_PRINT(_parms, _table) \
00054     dvb_table_header_print(_parms, &_table->header); \
00055     dvb_loginfo("| protocol_version %d", _table->protocol_version); \
00056
00057 #endif /* _DOXYGEN */
00058
00059 #endif /* _ATSC_HEADER_H */

```

9.6 lib/include/libdvbv5/cat.h File Reference

Provides the table parser for the CAT (Conditional Access Table)

```

#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/header.h>

```

Data Structures

- struct [dvb_table_cat](#)
ATSC CAT table.

Macros

- #define [DVB_TABLE_CAT](#)
ATSC CAT table ID.
- #define [DVB_TABLE_CAT_PID](#)
ATSC PID table ID.

Functions

- `ssize_t dvb_table_cat_init` (struct `dvb_v5_fe_parms` *parms, const `uint8_t` *buf, `ssize_t` buflen, struct `dvb_table_cat` **table)
Initializes and parses CAT table.
- `void dvb_table_cat_free` (struct `dvb_table_cat` *table)
Frees all data allocated by the CAT table parser.
- `void dvb_table_cat_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_table_cat` *table)
Prints the content of the CAT table.

9.6.1 Detailed Description

Provides the table parser for the CAT (Conditional Access Table)

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [cat.h](#).

9.6.2 Function Documentation

9.6.2.1 `dvb_table_cat_free()` `void dvb_table_cat_free (`
 struct `dvb_table_cat` * *table*)

Frees all data allocated by the CAT table parser.

Parameters

<i>table</i>	pointer to struct <code>dvb_table_cat</code> to be freed
--------------	--

9.6.2.2 `dvb_table_cat_init()` `ssize_t dvb_table_cat_init (`
 struct `dvb_v5_fe_parms` * *parms*,
 const `uint8_t` * *buf*,

```

    ssize_t buflen,
    struct dvb_table_cat ** table )

```

Initializes and parses CAT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the CAT raw data
<i>buflen</i>	length of the buffer
<i>table</i>	pointer to struct dvb_table_cat to be allocated and filled

This function allocates an CAT table and fills the fields inside the struct. It also makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

```

9.6.2.3 dvb_table_cat_print() void dvb_table_cat_print (
    struct dvb_v5_fe_parms * parms,
    struct dvb_table_cat * table )

```

Prints the content of the CAT table.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>table</i>	pointer to struct dvb_table_cat

9.7 cat.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00031 #ifndef _CAT_H
00032 #define _CAT_H
00033
00034 #include <stdint.h>

```

```

00035 #include <unistd.h> /* ssize_t */
00036
00037 #include <libdvbv5/header.h>
00038
00047 #define DVB_TABLE_CAT 0x01
00048 #define DVB_TABLE_CAT_PID 0x0001
00049
00057 struct dvb_table_cat {
00058     struct dvb_table_header header;
00059     struct dvb_desc *descriptor;
00060 } __attribute__((packed));
00061
00062 struct dvb_v5_fe_parms;
00063
00064 #ifdef __cplusplus
00065 extern "C" {
00066 #endif
00067
00083 ssize_t dvb_table_cat_init(struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00084                         ssize_t buflen, struct dvb_table_cat **table);
00085
00091 void dvb_table_cat_free(struct dvb_table_cat *table);
00092
00099 void dvb_table_cat_print(struct dvb_v5_fe_parms *parms,
00100                         struct dvb_table_cat *table);
00101
00102 #ifdef __cplusplus
00103 }
00104 #endif
00105
00106 #endif

```

9.8 lib/include/libdvbv5/countries.h File Reference

Provides ancillary code to convert ISO 3166-1 country codes.

Enumerations

- enum `dvb_country_t` {
COUNTRY_UNKNOWN , AD , AE , AF ,
AG , AI , AL , AM ,
AO , AQ , AR , AS ,
AT , AU , AW , AX ,
AZ , BA , BB , BD ,
BE , BF , BG , BH ,
BI , BJ , BL , BM ,
BN , BO , BQ , BR ,
BS , BT , BV , BW ,
BY , BZ , CA , CC ,
CD , CF , CG , CH ,
CI , CK , CL , CM ,
CN , CO , CR , CU ,
CV , CW , CX , CY ,
CZ , DE , DJ , DK ,
DM , DO , DZ , EC ,
EE , EG , EH , ER ,
ES , ET , FI , FJ ,
FK , FM , FO , FR ,
GA , GB , GD , GE ,
GF , GG , GH , GI ,
GL , GM , GN , GP ,
GQ , GR , GS , GT ,
GU , GW , GY , HK ,
HM , HN , HR , HT ,
HU , ID , IE , IL ,

```

IM, IN, IO, IQ,
IR, IS, IT, JE,
JM, JO, JP, KE,
KG, KH, KI, KM,
KN, KP, KR, KW,
KY, KZ, LA, LB,
LC, LI, LK, LR,
LS, LT, LU, LV,
LY, MA, MC, MD,
ME, MF, MG, MH,
MK, ML, MM, MN,
MO, MP, MQ, MR,
MS, MT, MU, MV,
MW, MX, MY, MZ,
NA, NC, NE, NF,
NG, NI, NL, NO,
NP, NR, NU, NZ,
OM, PA, PE, PF,
PG, PH, PK, PL,
PM, PN, PR, PS,
PT, PW, PY, QA,
RE, RO, RS, RU,
RW, SA, SB, SC,
SD, SE, SG, SH,
SI, SJ, SK, SL,
SM, SN, SO, SR,
SS, ST, SV, SX,
SY, SZ, TC, TD,
TF, TG, TH, TJ,
TK, TL, TM, TN,
TO, TR, TT, TV,
TW, TZ, UA, UG,
UM, US, UY, UZ,
VA, VC, VE, VG,
VI, VN, VU, WF,
WS, YE, YT, ZA,
ZM, ZW }

```

ISO-3166-1 alpha-2 country code.

Functions

- enum [dvb_country_t](#) [dvb_country_a2_to_id](#) (const char *name)
Converts an Unix-like 2-letter Country code into enum dvb_country_t.
- enum [dvb_country_t](#) [dvb_country_a3_to_id](#) (const char *name)
Converts a 3-letter Country code as used by MPEG-TS tables into enum dvb_country_t.
- const char * [dvb_country_to_2letters](#) (int id)
Converts an enum dvb_country_t into Unix-like 2-letter Country code.
- const char * [dvb_country_to_3letters](#) (int id)
Converts an enum dvb_country_t into a 3-letter Country code as used by MPEG-TS tables.
- const char * [dvb_country_to_name](#) (int id)
Converts an enum dvb_country_t into a Country name as used by MPEG-TS tables.
- enum [dvb_country_t](#) [dvb_guess_user_country](#) (void)
Guess the country code from the Unix environment variables.

9.8.1 Detailed Description

Provides ancillary code to convert ISO 3166-1 country codes.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Winfried Koehler

Akihiro Tsukada

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [countries.h](#).

9.9 countries.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (C) 2006, 2007, 2008, 2009 Winfried Koehler
00003  * Copyright (C) 2014 Akihiro Tsukada
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program. if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  */
00018
00031 #ifndef _COUNTRIES_H_
00032 #define _COUNTRIES_H_
00033
00034 #ifdef __cplusplus
00035 extern "C" {
00036 #endif
00037
00044 enum dvb_country_t {
00045     COUNTRY_UNKNOWN,
00046
00047     AD,
00048     AE,
00049     AF,
00050     AG,
00051     AI,
00052     AL,
00053     AM,
00054     AO,
00055     AQ,
00056     AR,
00057     AS,
00058     AT,
00059     AU,
00060     AW,
00061     AX,
00062     AZ,
00063     BA,
00064     BB,
00065     BD,
```


00566 BE,
00567 BF,
00568 BG,
00569 BH,
00570 BI,
00571 BJ,
00572 BL,
00573 BM,
00574 BN,
00575 BO,
00576 BQ,
00577 BR,
00578 BS,
00579 BT,
00580 BV,
00581 BW,
00582 BY,
00583 BZ,
00584 CA,
00585 CC,
00586 CD,
00587 CF,
00588 CG,
00589 CH,
00590 CI,
00591 CK,
00592 CL,
00593 CM,
00594 CN,
00595 CO,
00596 CR,
00597 CU,
00598 CV,
00599 CW,
00600 CX,
00601 CY,
00602 CZ,
00603 DE,
00604 DJ,
00605 DK,
00606 DM,
00607 DO,
00608 DZ,
00609 EC,
00610 EE,
00611 EG,
00612 EH,
00613 ER,
00614 ES,
00615 ET,
00616 FI,
00617 FJ,
00618 FK,
00619 FM,
00620 FO,
00621 FR,
00622 GA,
00623 GB,
00624 GD,
00625 GE,
00626 GF,
00627 GG,
00628 GH,
00629 GI,
00630 GL,
00631 GM,
00632 GN,
00633 GP,
00634 GQ,
00635 GR,
00636 GS,
00637 GT,
00638 GU,
00639 GW,
00640 GY,
00641 HK,
00642 HM,
00643 HN,
00644 HR,
00645 HT,
00646 HU,
00647 ID,
00648 IE,
00649 IL,
00650 IM,
00651 IN,
00652 IO,

```
00653    IQ,
00654    IR,
00655    IS,
00656    IT,
00657    JE,
00658    JM,
00659    JO,
00660    JP,
00661    KE,
00662    KG,
00663    KH,
00664    KI,
00665    KM,
00666    KN,
00667    KP,
00668    KR,
00669    KW,
00670    KY,
00671    KZ,
00672    LA,
00673    LB,
00674    LC,
00675    LI,
00676    LK,
00677    LR,
00678    LS,
00679    LT,
00680    LU,
00681    LV,
00682    LY,
00683    MA,
00684    MC,
00685    MD,
00686    ME,
00687    MF,
00688    MG,
00689    MH,
00690    MK,
00691    ML,
00692    MM,
00693    MN,
00694    MO,
00695    MP,
00696    MQ,
00697    MR,
00698    MS,
00699    MT,
00700    MU,
00701    MV,
00702    MW,
00703    MX,
00704    MY,
00705    MZ,
00706    NA,
00707    NC,
00708    NE,
00709    NF,
00710    NG,
00711    NI,
00712    NL,
00713    NO,
00714    NP,
00715    NR,
00716    NU,
00717    NZ,
00718    OM,
00719    PA,
00720    PE,
00721    PF,
00722    PG,
00723    PH,
00724    PK,
00725    PL,
00726    PM,
00727    PN,
00728    PR,
00729    PS,
00730    PT,
00731    PW,
00732    PY,
00733    QA,
00734    RE,
00735    RO,
00736    RS,
00737    RU,
00738    RW,
00739    SA,
```

```

00740     SB,
00741     SC,
00742     SD,
00743     SE,
00744     SG,
00745     SH,
00746     SI,
00747     SJ,
00748     SK,
00749     SL,
00750     SM,
00751     SN,
00752     SO,
00753     SR,
00754     SS,
00755     ST,
00756     SV,
00757     SX,
00758     SY,
00759     SZ,
00760     TC,
00761     TD,
00762     TF,
00763     TG,
00764     TH,
00765     TJ,
00766     TK,
00767     TL,
00768     TM,
00769     TN,
00770     TO,
00771     TR,
00772     TT,
00773     TV,
00774     TW,
00775     TZ,
00776     UA,
00777     UG,
00778     UM,
00779     US,
00780     UY,
00781     UZ,
00782     VA,
00783     VC,
00784     VE,
00785     VG,
00786     VI,
00787     VN,
00788     VU,
00789     WF,
00790     WS,
00791     YE,
00792     YT,
00793     ZA,
00794     ZM,
00795     ZW,
00796 };
00797
00807 enum dvb_country_t dvb_country_a2_to_id(const char *name);
00808
00819 enum dvb_country_t dvb_country_a3_to_id(const char *name);
00820
00830 const char *dvb_country_to_2letters(int id);
00831
00842 const char *dvb_country_to_3letters(int id);
00843
00854 const char *dvb_country_to_name(int id);
00855
00863 enum dvb_country_t dvb_guess_user_country(void);
00864
00865 #ifdef __cplusplus
00866 }
00867 #endif
00868
00869 #endif

```

9.10 lib/include/libdvbv5/crc32.h File Reference

Provides ancillary code to calculate DVB crc32 checksum.

```

#include <stdint.h>
#include <unistd.h>

```

Functions

- `uint32_t dvb_crc32 (uint8_t *data, size_t datalen, uint32_t crc)`
Calculates the crc-32 as defined at the MPEG-TS specs.

9.10.1 Detailed Description

Provides ancillary code to calculate DVB crc32 checksum.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
 Andre Roth

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [crc32.h](#).

9.11 crc32.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012-2014 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00033 #ifndef _CRC32_H
00034 #define _CRC32_H
00035
00036 #include <stdint.h>
00037 #include <unistd.h> /* size_t */
00038
00039 #ifdef __cplusplus
00040 extern "C" {
00041 #endif
00042
00051 uint32_t dvb_crc32(uint8_t *data, size_t datalen, uint32_t crc);
00052
00053 #ifdef __cplusplus
00054 }
00055 #endif
00056
00057 #endif
00058
```

9.12 lib/include/libdvbv5/desc_atsc_service_location.h File Reference

Provides the descriptors for ATSC service location.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [atsc_desc_service_location_elementary](#)
service location elementary descriptors
- struct [atsc_desc_service_location](#)
Describes the elementary streams inside a PAT table for ATSC.

Functions

- int [atsc_desc_service_location_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the service location descriptor.
- void [atsc_desc_service_location_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the service location descriptor.
- void [atsc_desc_service_location_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the service location descriptor.

9.12.1 Detailed Description

Provides the descriptors for ATSC service location.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Relevant specs

The descriptor described herein is defined at:

- ATSC A/53

See also

<http://www.etherguidesystems.com/help/sdos/atsc/semantics/descriptors/ServiceLocation.aspx>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_atsc_service_location.h](#).

9.13 desc_atsc_service_location.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013-2014 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00020 #ifndef _ATSC_SERVICE_LOCATION_H
00021 #define _ATSC_SERVICE_LOCATION_H
00022
00023 #include <libdvbv5/descriptors.h>
00024
00051 struct atsc_desc_service_location_elementary {
00052     uint8_t stream_type;
00053     union {
00054         uint16_t bitfield;
00055         struct {
00056             uint16_t elementary_pid:13;
00057             uint16_t reserved:3;
00058         } __attribute__((packed));
00059     } __attribute__((packed));
00060     unsigned char ISO_639_language_code[3];
00061 } __attribute__((packed));
00062
00075 struct atsc_desc_service_location {
00076     uint8_t type;
00077     uint8_t length;
00078     struct dvb_desc *next;
00079
00080     struct atsc_desc_service_location_elementary *elementary;
00081
00082     union {
00083         uint16_t bitfield;
00084         struct {
00085             uint16_t pcr_pid:13;
00086             uint16_t reserved:3;
00087         } __attribute__((packed));
00088     } __attribute__((packed));
00089
00090     uint8_t number_elements;
00091 } __attribute__((packed));
00092
00093 struct dvb_v5_fe_parms;
00094
00095 #ifdef __cplusplus
00096 extern "C" {
00097 #endif
00098
00114 int atsc_desc_service_location_init(struct dvb_v5_fe_parms *parms,
00115                                   const uint8_t *buf,
00116                                   struct dvb_desc *desc);
00117
00125 void atsc_desc_service_location_print(struct dvb_v5_fe_parms *parms,
00126                                      const struct dvb_desc *desc);
00127
00134 void atsc_desc_service_location_free(struct dvb_desc *desc);
00135
00136 #ifdef __cplusplus
00137 }
00138 #endif
00139
00140 #endif

```

9.14 lib/include/libdvbv5/desc_ca.h File Reference

Provides the descriptors for Conditional Access.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_ca](#)
Contains the private data for Conditional Access.

Macros

- #define [dvb_desc_ca_field_first](#)
initial descriptor field at [dvb_desc_ca](#) struct
- #define [dvb_desc_ca_field_last](#)
last descriptor field at [dvb_desc_ca](#) struct

Functions

- int [dvb_desc_ca_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the CA descriptor.
- void [dvb_desc_ca_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the CA descriptor.
- void [dvb_desc_ca_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the CA descriptor.

9.14.1 Detailed Description

Provides the descriptors for Conditional Access.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1 (2010-04)

See also

http://www.etherguidesystems.com/help/sdos/mpeg/semantics/mpeg-2/descriptors/CA_descriptor.aspx

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_ca.h](#).

9.14.2 Macro Definition Documentation

9.14.2.1 dvb_desc_ca_field_first `#define dvb_desc_ca_field_first`

initial descriptor field at `dvb_desc_ca` struct

Definition at line 77 of file `desc_ca.h`.

9.14.2.2 dvb_desc_ca_field_last `#define dvb_desc_ca_field_last`

last descriptor field at `dvb_desc_ca` struct

Definition at line 79 of file `desc_ca.h`.

9.15 desc_ca.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Described at ETSI EN 300 468 V1.11.1 (2010-04)
00019  */
00020
00038 #ifndef _CA_H
00039 #define _CA_H
00040
00041 #include <libdvbv5/descriptors.h>
00042
00056 struct dvb_desc_ca {
00057     uint8_t type;
00058     uint8_t length;
00059     struct dvb_desc *next;
00060
00061     uint16_t ca_id;
00062     union {
00063         uint16_t bitfield1;
00064         struct {
00065             uint16_t ca_pid:13;
00066             uint16_t reserved:3;
00067         } __attribute__((packed));
00068     } __attribute__((packed));
00069
00070     uint8_t *privdata;
00071     uint8_t privdata_len;
00072 } __attribute__((packed));
00073
00074 struct dvb_v5_fe_parms;
00075
00077 #define dvb_desc_ca_field_first ca_id
00079 #define dvb_desc_ca_field_last privdata
00080
00081 #ifdef __cplusplus

```



```
00082 extern "C" {
00083 #endif
00084
00100 int dvb_desc_ca_init(struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00101                     struct dvb_desc *desc);
00102
00110 void dvb_desc_ca_print(struct dvb_v5_fe_parms *parms,
00111                        const struct dvb_desc *desc);
00112
00119 void dvb_desc_ca_free(struct dvb_desc *desc);
00120
00121 #ifdef __cplusplus
00122 }
00123 #endif
00124
00125 #endif
```

9.16 lib/include/libdvbv5/desc_ca_identifier.h File Reference

Provides the descriptors for the Conditional Access identifier.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_ca_identifier](#)
Indicates if a particular bouquet, service or event is associated with a CA system.

Macros

- #define [dvb_desc_ca_identifier_field_first](#)
initial descriptor field at [dvb_desc_ca_identifier](#) struct
- #define [dvb_desc_ca_identifier_field_last](#)
last descriptor field at [dvb_desc_ca_identifier](#) struct

Functions

- int [dvb_desc_ca_identifier_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the CA identifier descriptor.
- void [dvb_desc_ca_identifier_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the CA identifier descriptor.
- void [dvb_desc_ca_identifier_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the CA identifier descriptor.

9.16.1 Detailed Description

Provides the descriptors for the Conditional Access identifier.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1 (2010-04)

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_ca_identifier.h](#).

9.16.2 Macro Definition Documentation

9.16.2.1 `dvb_desc_ca_identifier_field_first` `#define dvb_desc_ca_identifier_field_first`

initial descriptor field at [dvb_desc_ca_identifier](#) struct

Definition at line 66 of file [desc_ca_identifier.h](#).

9.16.2.2 `dvb_desc_ca_identifier_field_last` `#define dvb_desc_ca_identifier_field_last`

last descriptor field at [dvb_desc_ca_identifier](#) struct

Definition at line 68 of file [desc_ca_identifier.h](#).

9.17 desc_ca_identifier.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Described at ETSI EN 300 468 V1.11.1 (2010-04)
00019  */
00020
00036 #ifndef _CA_IDENTIFIER_H
00037 #define _CA_IDENTIFIER_H
00038
00039 #include <libdvbv5/descriptors.h>
00040
00053 struct dvb_desc_ca_identifier {
00054     uint8_t type;
00055     uint8_t length;
00056     struct dvb_desc *next;
00057
00058     uint8_t caid_count;
00059     uint16_t *caids;
00060
00061 } __attribute__((packed));
00062
00063 struct dvb_v5_fe_parms;
00064
00066 #define dvb_desc_ca_identifier_field_first ca_id
00068 #define dvb_desc_ca_identifier_field_last privdata
00069
00070 #ifdef __cplusplus
00071 extern "C" {
00072 #endif
00073
00089 int dvb_desc_ca_identifier_init(struct dvb_v5_fe_parms *parms,
00090                               const uint8_t *buf, struct dvb_desc *desc);
00091
00099 void dvb_desc_ca_identifier_print(struct dvb_v5_fe_parms *parms,
00100                                  const struct dvb_desc *desc);
00101
00108 void dvb_desc_ca_identifier_free(struct dvb_desc *desc);
00109
00110 #ifdef __cplusplus
00111 }
00112 #endif
00113
00114 #endif

```

9.18 lib/include/libdvbv5/desc_cable_delivery.h File Reference

Provides the descriptors for the cable delivery system descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- [struct dvb_desc_cable_delivery](#)

Structure containing the cable delivery system descriptor.

Functions

- int [dvb_desc_cable_delivery_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the service location descriptor.
- void [dvb_desc_cable_delivery_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the service location descriptor.

Variables

- const unsigned [dvbc_modulation_table](#) []
converts from the descriptor's modulation into enum fe_modulation, as defined by DVBv5 API.
- const unsigned [dvbc_fec_table](#) []
converts from the descriptor's FEC into enum fe_code_rate, as defined by DVBv5 API.

9.18.1 Detailed Description

Provides the descriptors for the cable delivery system descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1 (2010-04)

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_cable_delivery.h](#).

9.18.2 Variable Documentation

9.18.2.1 [dvbc_fec_table](#) const unsigned dvbc_fec_table[] [extern]

converts from the descriptor's FEC into enum fe_code_rate, as defined by DVBv5 API.

9.18.2.2 dvbc_modulation_table `const unsigned dvbc_modulation_table[] [extern]`

converts from the descriptor's modulation into enum `fe_modulation`, as defined by DVBv5 API.

9.19 desc_cable_delivery.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  * Described at ETSI EN 300 468 V1.11.1 (2010-04)
00020  */
00021
00038 #ifndef _CABLE_DELIVERY_H
00039 #define _CABLE_DELIVERY_H
00040
00041 #include <libdvbv5/descriptors.h>
00042
00057 struct dvb_desc_cable_delivery {
00058     uint8_t type;
00059     uint8_t length;
00060     struct dvb_desc *next;
00061
00062     uint32_t frequency;
00063     union {
00064         uint16_t bitfield1;
00065         struct {
00066             uint16_t fec_outer:4;
00067             uint16_t reserved_future_use:12;
00068         } __attribute__((packed));
00069     } __attribute__((packed));
00070     uint8_t modulation;
00071     union {
00072         uint32_t bitfield2;
00073         struct {
00074             uint32_t fec_inner:4;
00075             uint32_t symbol_rate:28;
00076         } __attribute__((packed));
00077     } __attribute__((packed));
00078 } __attribute__((packed));
00079
00080 struct dvb_v5_fe_parms;
00081
00082 #ifdef __cplusplus
00083 extern "C" {
00084 #endif
00085
00102 int dvb_desc_cable_delivery_init(struct dvb_v5_fe_parms *parms,
00103                                const uint8_t *buf, struct dvb_desc *desc);
00104
00112 void dvb_desc_cable_delivery_print(struct dvb_v5_fe_parms *parms,
00113                                   const struct dvb_desc *desc);
00114
00119 extern const unsigned dvbc_modulation_table[];
00120
00125 extern const unsigned dvbc_fec_table[];
00126
00127
00128 #ifdef __cplusplus
00129 }
00130 #endif
00131
00132 #endif

```

9.20 lib/include/libdvbv5/desc_event_extended.h File Reference

Provides the descriptors for the extended event descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_event_extended_item](#)
- struct [dvb_desc_event_extended](#)
Structure containing the extended event descriptor.

Functions

- int [dvb_desc_event_extended_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the extended event descriptor.
- void [dvb_desc_event_extended_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the extended event descriptor.
- void [dvb_desc_event_extended_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the extended event descriptor.

9.20.1 Detailed Description

Provides the descriptors for the extended event descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_event_extended.h](#).

9.21 desc_event_extended.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00037 #ifndef _DESC_EVENT_EXTENDED_H
00038 #define _DESC_EVENT_EXTENDED_H
00039
00040 #include <libdvbv5/descriptors.h>
00041
00042 struct dvb_desc_event_extended_item {
00043     char *description;
00044     char *description_emph;
00045     char *item;
00046     char *item_emph;
00047 };
00048
00068 struct dvb_desc_event_extended {
00069     uint8_t type;
00070     uint8_t length;
00071     struct dvb_desc *next;
00072
00073     union {
00074         struct {
00075             uint8_t last_id:4;
00076             uint8_t id:4;
00077         } __attribute__((packed));
00078         uint8_t ids;
00079     } __attribute__((packed));
00080
00081     unsigned char language[4];
00082     char *text;
00083     char *text_emph;
00084     struct dvb_desc_event_extended_item *items;
00085     int num_items;
00086 } __attribute__((packed));
00087
00088 struct dvb_v5_fe_parms;
00089
00090 #ifdef __cplusplus
00091 extern "C" {
00092 #endif
00093
00109 int dvb_desc_event_extended_init(struct dvb_v5_fe_parms *parms,
00110                                const uint8_t *buf, struct dvb_desc *desc);
00111
00119 void dvb_desc_event_extended_print(struct dvb_v5_fe_parms *parms,
00120                                   const struct dvb_desc *desc);
00121
00128 void dvb_desc_event_extended_free(struct dvb_desc *desc);
00129
00130 #ifdef __cplusplus
00131 }
00132 #endif
00133
00134 #endif

```

9.22 lib/include/libdvbv5/desc_event_short.h File Reference

Provides the descriptors for the short event descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_event_short](#)
Structure containing the short event descriptor.

Functions

- int [dvb_desc_event_short_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the short event descriptor.
- void [dvb_desc_event_short_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the short event descriptor.
- void [dvb_desc_event_short_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the short event descriptor.

9.22.1 Detailed Description

Provides the descriptors for the short event descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_event_short.h](#).

9.23 desc_event_short.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00037 #ifndef _DESC_EVENT_SHORT_H
00038 #define _DESC_EVENT_SHORT_H
00039
00040 #include <libdvbv5/descriptors.h>
00041
00060 struct dvb_desc_event_short {
00061     uint8_t type;
00062     uint8_t length;
00063     struct dvb_desc *next;
00064
00065     unsigned char language[4];
00066     char *name;
00067     char *name_emph;
00068     char *text;
00069     char *text_emph;
00070 } __attribute__((packed));
00071
00072 struct dvb_v5_fe_parms;
00073
00074 #ifdef __cplusplus
00075 extern "C" {
00076 #endif
00077
00093 int dvb_desc_event_short_init(struct dvb_v5_fe_parms *parms,
00094                             const uint8_t *buf, struct dvb_desc *desc);
00095
00103 void dvb_desc_event_short_print(struct dvb_v5_fe_parms *parms,
00104                                const struct dvb_desc *desc);
00105
00112 void dvb_desc_event_short_free(struct dvb_desc *desc);
00113
00114 #ifdef __cplusplus
00115 }
00116 #endif
00117
00118 #endif

```

9.24 lib/include/libdvbv5/desc_extension.h File Reference

Provides the descriptors for the extension descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_extension_descriptor](#)
Structure containing the extended descriptors.
- struct [dvb_ext_descriptor](#)
Structure that describes the parser functions for the extended descriptors.

Typedefs

- typedef int(* [dvb_desc_ext_init_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_extension_descriptor](#) *ext, void *desc)
Function prototype for the extended descriptors parsing init code.
- typedef void(* [dvb_desc_ext_print_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_extension_descriptor](#) *ext, const void *desc)
Function prototype for the extended descriptors parsing print code.
- typedef void(* [dvb_desc_ext_free_func](#)) (const void *desc)
Function prototype for the extended descriptors parsing free code.

Enumerations

- enum [extension_descriptors](#) {
[image_icon_descriptor](#) , [cpcm_delivery_signalling_descriptor](#) , [CP_descriptor](#) , [CP_identifier_descriptor](#) ,
[T2_delivery_system_descriptor](#) , [SH_delivery_system_descriptor](#) , [supplementary_audio_descriptor](#) ,
[network_change_notify_descriptor](#) ,
[message_descriptor](#) , [target_region_descriptor](#) , [target_region_name_descriptor](#) , [service_relocated_descriptor](#)
}
- List containing all extended descriptors used by Digital TV MPEG-TS as defined at ETSI EN 300 468 V1.11.1.*

Functions

- int [dvb_extension_descriptor_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the extended descriptor.
- void [dvb_extension_descriptor_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the extended descriptor.
- void [dvb_extension_descriptor_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the extended descriptor.

9.24.1 Detailed Description

Provides the descriptors for the extension descriptor.

The extension descriptor is used to extend the 8-bit namespace of the descriptor_tag field.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Relevant specs

The descriptor described herein is defined at:

See also

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_extension.h](#).

9.25 desc_extension.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013-2014 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00039 #ifndef _EXTENSION_DESC_H
00040 #define _EXTENSION_DESC_H
00041
00042 #include <libdvbv5/descriptors.h>
00043
00044 struct dvb_v5_fe_parms;
00045
00089 enum extension_descriptors {
00090     image_icon_descriptor           = 0x00,
00091     cpcm_delivery_signalling_descriptor = 0x01,
00092     CP_descriptor                   = 0x02,
00093     CP_identifier_descriptor         = 0x03,
00094     T2_delivery_system_descriptor   = 0x04,
00095     SH_delivery_system_descriptor   = 0x05,
00096     supplementary_audio_descriptor   = 0x06,
00097     network_change_notify_descriptor = 0x07,
00098     message_descriptor               = 0x08,
00099     target_region_descriptor         = 0x09,
00100     target_region_name_descriptor    = 0x0a,
00101     service_relocated_descriptor     = 0x0b,
00102 };
00103
00115 struct dvb_extension_descriptor {
00116     uint8_t type;
00117     uint8_t length;
00118     struct dvb_desc *next;
00119
00120     uint8_t extension_code;
00121
00122     struct dvb_desc *descriptor;
00123 } __attribute__((packed));
00124
00125
00135 typedef int (*dvb_desc_ext_init_func) (struct dvb_v5_fe_parms *parms,
00136     const uint8_t *buf,
00137     struct dvb_extension_descriptor *ext,
00138     void *desc);
00148 typedef void (*dvb_desc_ext_print_func) (struct dvb_v5_fe_parms *parms,
00149     const struct dvb_extension_descriptor *ext,
00150     const void *desc);
00157 typedef void (*dvb_desc_ext_free_func) (const void *desc);
00158
00171 struct dvb_ext_descriptor {
00172     const char *name;
00173     dvb_desc_ext_init_func init;
00174     dvb_desc_ext_print_func print;
00175     dvb_desc_ext_free_func free;
00176     ssize_t size;
00177 };
00178
00179
00180 #ifdef __cplusplus
00181 extern "C" {
00182 #endif
00183
00199 int dvb_extension_descriptor_init(struct dvb_v5_fe_parms *parms,
00200     const uint8_t *buf, struct dvb_desc *desc);
00201
00209 void dvb_extension_descriptor_print(struct dvb_v5_fe_parms *parms,
00210     const struct dvb_desc *desc);
00211
00218 void dvb_extension_descriptor_free(struct dvb_desc *desc);
00219
00220 #ifdef __cplusplus

```

```
00221 }  
00222 #endif  
00223  
00224 #endif
```

9.26 lib/include/libdvbv5/desc_frequency_list.h File Reference

Provides the descriptors for the frequency list descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_frequency_list](#)
Struct containing the frequency list descriptor.

Functions

- int [dvb_desc_frequency_list_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the frequency list descriptor.
- void [dvb_desc_frequency_list_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the frequency list descriptor.

9.26.1 Detailed Description

Provides the descriptors for the frequency list descriptor.

This descriptor lists the additional frequencies used in transmission of a multiplex on other frequencies.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_frequency_list.h](#).

9.27 desc_frequency_list.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00021 #ifndef _DESC_FREQUENCY_LIST_H
00022 #define _DESC_FREQUENCY_LIST_H
00023
00024 #include <libdvbv5/descriptors.h>
00056 struct dvb_desc_frequency_list {
00057     uint8_t type;
00058     uint8_t length;
00059     struct dvb_desc *next;
00060
00061     uint8_t frequencies;
00062     uint32_t *frequency;
00063
00064     union {
00065         uint8_t bitfield;
00066         struct {
00067             uint8_t freq_type:2;
00068             uint8_t reserved:6;
00069         } __attribute__((packed));
00070     } __attribute__((packed));
00071 } __attribute__((packed));
00072
00073 struct dvb_v5_fe_parms;
00074
00075 #ifdef __cplusplus
00076 extern "C" {
00077 #endif
00078
00095 int dvb_desc_frequency_list_init(struct dvb_v5_fe_parms *parms,
00096                                const uint8_t *buf, struct dvb_desc *desc);
00097
00105 void dvb_desc_frequency_list_print(struct dvb_v5_fe_parms *parms,
00106                                   const struct dvb_desc *desc);
00107
00108 #ifdef __cplusplus
00109 }
00110 #endif
00111
00112 #endif

```

9.28 lib/include/libdvbv5/desc_hierarchy.h File Reference

Provides the descriptors for the hierarchy descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- [struct dvb_desc_hierarchy](#)

Structure containing the hierarchy descriptor.

Functions

- int `dvb_desc_hierarchy_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, struct `dvb_desc` *desc)
Initializes and parses the hierarchy descriptor.
- void `dvb_desc_hierarchy_print` (struct `dvb_v5_fe_parms` *parms, const struct `dvb_desc` *desc)
Prints the content of the hierarchy descriptor.

9.28.1 Detailed Description

Provides the descriptors for the hierarchy descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ISO/IEC 13818-1

See also

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_hierarchy.h](#).

9.29 desc_hierarchy.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
```

```

00018  *
00019  */
00020
00039 #ifndef _HIERARCHY_H
00040 #define _HIERARCHY_H
00041
00042 #include <libdvbv5/descriptors.h>
00043
00057 struct dvb_desc_hierarchy {
00058     uint8_t type;
00059     uint8_t length;
00060     struct dvb_desc *next;
00061
00062     uint8_t hierarchy_type:4;
00063     uint8_t reserved:4;
00064
00065     uint8_t layer:6;
00066     uint8_t reserved2:2;
00067
00068     uint8_t embedded_layer:6;
00069     uint8_t reserved3:2;
00070
00071     uint8_t channel:6;
00072     uint8_t reserved4:2;
00073 } __attribute__((packed));
00074
00075 struct dvb_v5_fe_parms;
00076
00077 #ifdef __cplusplus
00078 extern "C" {
00079 #endif
00080
00097 int dvb_desc_hierarchy_init (struct dvb_v5_fe_parms *parms,
00098                             const uint8_t *buf, struct dvb_desc *desc);
00099
00107 void dvb_desc_hierarchy_print (struct dvb_v5_fe_parms *parms,
00108                               const struct dvb_desc *desc);
00109
00110 #ifdef __cplusplus
00111 }
00112 #endif
00113
00114 #endif

```

9.30 lib/include/libdvbv5/desc_isdbt_delivery.h File Reference

Provides the descriptors for the ISDB-T terrestrial delivery system.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [isdbt_desc_terrestrial_delivery_system](#)
Struct containing the ISDB-T terrestrial delivery system.

Functions

- int [isdbt_desc_delivery_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the ISDB-T terrestrial delivery system descriptor.
- void [isdbt_desc_delivery_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the ISDB-T terrestrial delivery system descriptor.
- void [isdbt_desc_delivery_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the ISDB-T terrestrial delivery system descriptor.

Variables

- `const uint32_t isdbt_interval []`
Converts an ISDB-T Interval code into a string.
- `const uint32_t isdbt_mode []`
Converts an ISDB-T mode into a string.

9.30.1 Detailed Description

Provides the descriptors for the ISDB-T terrestrial delivery system.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Relevant specs

The descriptor described herein is defined at:

- ARIB STD-B10

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_isdbt_delivery.h](#).

9.30.2 Variable Documentation

9.30.2.1 `isdbt_interval` `const uint32_t isdbt_interval[]` [extern]

Converts an ISDB-T Interval code into a string.

9.30.2.2 `isdbt_mode` `const uint32_t isdbt_mode[]` [extern]

Converts an ISDB-T mode into a string.

9.31 desc_isdbt_delivery.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013-2014 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Described on ARIB STD-B10 as Terrestrial delivery system descriptor
00019  */
00020
00036 #ifndef __ISDBT_DELIVERY_H
00037 #define __ISDBT_DELIVERY_H
00038
00039 #include <libdvbv5/descriptors.h>
00040
00057 struct isdbt_desc_terrestrial_delivery_system {
00058     uint8_t type;
00059     uint8_t length;
00060     struct dvb_desc *next;
00061
00062     uint32_t *frequency;
00063     unsigned num_freqs;
00064
00065     union {
00066         uint16_t bitfield;
00067         struct {
00068             uint16_t transmission_mode:2;
00069             uint16_t guard_interval:2;
00070             uint16_t area_code:12;
00071         } __attribute__((packed));
00072     } __attribute__((packed));
00073 } __attribute__((packed));
00074
00075 struct dvb_v5_fe_parms;
00076
00077 #ifdef __cplusplus
00078 extern "C" {
00079 #endif
00080
00097 int isdbt_desc_delivery_init(struct dvb_v5_fe_parms *parms,
00098                             const uint8_t *buf, struct dvb_desc *desc);
00099
00108 void isdbt_desc_delivery_print(struct dvb_v5_fe_parms *parms,
00109                               const struct dvb_desc *desc);
00110
00118 void isdbt_desc_delivery_free(struct dvb_desc *desc);
00119
00123 extern const uint32_t isdbt_interval[];
00124
00128 extern const uint32_t isdbt_mode[];
00129
00130 #ifdef __cplusplus
00131 }
00132 #endif
00133
00134 #endif

```

9.32 lib/include/libdvbv5/desc_language.h File Reference

Provides the descriptors for the ISO639 language descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_language](#)
Structure containing the ISO639 language descriptor.

Functions

- int [dvb_desc_language_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the language descriptor.
- void [dvb_desc_language_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the language descriptor.

9.32.1 Detailed Description

Provides the descriptors for the ISO639 language descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ISO/IEC 13818-1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_language.h](#).

9.33 desc_language.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00037 #ifndef _DESC_LANGUAGE_H
00038 #define _DESC_LANGUAGE_H
00039
00040 #include <libdvbv5/descriptors.h>
00041
00055 struct dvb_desc_language {
00056     uint8_t type;
00057     uint8_t length;
00058     struct dvb_desc *next;
00059
00060     unsigned char language[4];
00061     uint8_t audio_type;
00062 } __attribute__((packed));
00063
00064 struct dvb_v5_fe_parms;
00065
00066 #ifdef __cplusplus
00067 extern "C" {
00068 #endif
00069
00086 int dvb_desc_language_init(struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00087                          struct dvb_desc *desc);
00088
00096 void dvb_desc_language_print(struct dvb_v5_fe_parms *parms,
00097                             const struct dvb_desc *desc);
00098
00099 #ifdef __cplusplus
00100 }
00101 #endif
00102
00103 #endif

```

9.34 lib/include/libdvbv5/desc_logical_channel.h File Reference

Provides the descriptors for the LCN - Logical Channel Number.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_logical_channel_number](#)
Structure containing the logical channel number entires.
- struct [dvb_desc_logical_channel](#)
Structure containing the logical channel number descriptor.

Functions

- int [dvb_desc_logical_channel_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the logical channel number descriptor.
- void [dvb_desc_logical_channel_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the logical channel number descriptor.
- void [dvb_desc_logical_channel_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the logical channel number descriptor.

9.34.1 Detailed Description

Provides the descriptors for the LCN - Logican Channel Number.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Relevant specs

The descriptor described herein is defined at:

- IEC/CENELEC DS/EN 62216-1:2011

See also

<http://tdt.telecom.pt/recursos/apresentacoes/Signalling> Specifications for DTT deployment in Portugal.pdf

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_logical_channel.h](#).

9.35 desc_logical_channel.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Described on IEC/CENELEC DS/EN 62216-1:2011
00019  *
00020  * I couldn't find the original version, so I used what's there at:
00021  * http://tdt.telecom.pt/recursos/apresentacoes/Signalling Specifications for DTT deployment in
00022  * Portugal.pdf
00023  */
00041 #ifndef __LCN_DESC_H
00042 #define __LCN_DESC_H
00043
00044 #include <libdvbv5/descriptors.h>
00045
00056 struct dvb_desc_logical_channel_number {
00057     uint16_t service_id;
00058     union {
00059         uint16_t bitfield;
00060         struct {
00061             uint16_t logical_channel_number:10;
00062             uint16_t reserved:5;
00063             uint16_t visible_service_flag:1;
00064         } __attribute__((packed));
00065     } __attribute__((packed));
00066 } __attribute__((packed));
00067
00078 struct dvb_desc_logical_channel {
00079     uint8_t type;
00080     uint8_t length;
00081     struct dvb_desc *next;
00082
00083     struct dvb_desc_logical_channel_number *lcn;
00084 } __attribute__((packed));
00085
00086 struct dvb_v5_fe_parms;
00087
00088 #ifdef __cplusplus
00089 extern "C" {
00090 #endif
00091
00107 int dvb_desc_logical_channel_init(struct dvb_v5_fe_parms *parms,
00108                                 const uint8_t *buf, struct dvb_desc *desc);
00109
00117 void dvb_desc_logical_channel_print(struct dvb_v5_fe_parms *parms,
00118                                    const struct dvb_desc *desc);
00119
00126 void dvb_desc_logical_channel_free(struct dvb_desc *desc);
00127
00128 #ifdef __cplusplus
00129 }
00130 #endif
00131
00132 #endif

```

9.36 lib/include/libdvbv5/desc_network_name.h File Reference

Provides the descriptors for the network name descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_network_name](#)
Struct containing the network name descriptor.

Functions

- int [dvb_desc_network_name_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the network name descriptor.
- void [dvb_desc_network_name_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the network name descriptor.
- void [dvb_desc_network_name_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the network name descriptor.

9.36.1 Detailed Description

Provides the descriptors for the network name descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1 (2010-04)

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_network_name.h](#).

9.37 desc_network_name.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00037 #ifndef _DESC_NETWORK_NAME_H
00038 #define _DESC_NETWORK_NAME_H
00039
00040 #include <libdvbv5/descriptors.h>
00041
00057 struct dvb_desc_network_name {
00058     uint8_t type;
00059     uint8_t length;
00060     struct dvb_desc *next;
00061
00062     char *network_name;
00063     char *network_name_emph;
00064 } __attribute__((packed));
00065
00066 struct dvb_v5_fe_parms;
00067
00068 #ifdef __cplusplus
00069 extern "C" {
00070 #endif
00071
00087 int dvb_desc_network_name_init(struct dvb_v5_fe_parms *parms,
00088                               const uint8_t *buf, struct dvb_desc *desc);
00089
00097 void dvb_desc_network_name_print(struct dvb_v5_fe_parms *parms,
00098                                  const struct dvb_desc *desc);
00099
00106 void dvb_desc_network_name_free(struct dvb_desc *desc);
00107
00108 #ifdef __cplusplus
00109 }
00110 #endif
00111
00112 #endif

```

9.38 lib/include/libdvbv5/desc_partial_reception.h File Reference

Provides the descriptors for the ISDB partial reception descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [isdb_partial_reception_service_id](#)
Service ID that uses partial reception.
- struct [isdb_desc_partial_reception](#)
Structure containing the partial reception descriptor.

Functions

- int [isdb_desc_partial_reception_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the ISDB-T partial reception descriptor.
- void [isdb_desc_partial_reception_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the ISDB-T partial reception descriptor.
- void [isdb_desc_partial_reception_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the ISDB-T partial reception descriptor.

9.38.1 Detailed Description

Provides the descriptors for the ISDB partial reception descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- IEC/CENELEC DS/EN 62216-1:2011

See also

<http://tdt.telecom.pt/recursos/apresentacoes/Signalling> Specifications for DTT deployment in Portugal.pdf

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_partial_reception.h](#).

9.39 desc_partial_reception.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Described on IEC/CENELEC DS/EN 62216-1:2011
00019  *
00020  * I couldn't find the original version, so I used what's there at:
00021  * http://tdt.telecom.pt/recursos/apresentacoes/Signalling Specifications for DTT deployment in
    Portugal.pdf
00022  */
00023
00043 #ifndef _PARTIAL_RECEPTION_H
00044 #define _PARTIAL_RECEPTION_H
00045
00046 #include <libdvbv5/descriptors.h>
00047
00055 struct isdb_partial_reception_service_id {
00056     uint16_t service_id;
00057 } __attribute__((packed));
00058
00071 struct isdb_desc_partial_reception {
00072     uint8_t type;
00073     uint8_t length;
00074     struct dvb_desc *next;
00075
00076     struct isdb_partial_reception_service_id *partial_reception;
00077 } __attribute__((packed));
00078
00079 struct dvb_v5_fe_parms;
00080
00081 #ifdef __cplusplus
00082 extern "C" {
00083 #endif
00084
00100 int isdb_desc_partial_reception_init(struct dvb_v5_fe_parms *parms,
00101                                     const uint8_t *buf, struct dvb_desc *desc);
00102
00110 void isdb_desc_partial_reception_print(struct dvb_v5_fe_parms *parms,
00111                                       const struct dvb_desc *desc);
00112
00119 void isdb_desc_partial_reception_free(struct dvb_desc *desc);
00120
00121 #ifdef __cplusplus
00122 }
00123 #endif
00124
00125 #endif

```

9.40 lib/include/libdvbv5/desc_registration_id.h File Reference

Provides the descriptors for the registration descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- [struct dvb_desc_registration](#)

Struct containing the frequency list descriptor.

Functions

- int [dvb_desc_registration_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the registration descriptor.
- void [dvb_desc_registration_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the registration descriptor.
- void [dvb_desc_registration_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the registration descriptor.

9.40.1 Detailed Description

Provides the descriptors for the registration descriptor.

This descriptor provides the format information for an Elementary Stream.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Relevant specs

The descriptor described herein is defined at:

- ISO/IEC 13818-1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_registration_id.h](#).

9.41 desc_registration_id.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2020 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00020 #ifndef _DESC_REGISTRATION_ID_H
```

```

00021 #define _DESC_REGISTRATION_ID_H
00022
00023 #include <libdvbv5/descriptors.h>
00053 struct dvb_desc_registration {
00054     uint8_t type;
00055     uint8_t length;
00056     struct dvb_desc *next;
00057
00058     uint8_t format_identifier[4];
00059     uint8_t *additional_identification_info;
00060 } __attribute__((packed));
00061
00062 struct dvb_v5_fe_parms;
00063
00064 #ifdef __cplusplus
00065 extern "C" {
00066 #endif
00067
00084 int dvb_desc_registration_init(struct dvb_v5_fe_parms *parms,
00085                               const uint8_t *buf, struct dvb_desc *desc);
00086
00094 void dvb_desc_registration_print(struct dvb_v5_fe_parms *parms,
00095                                 const struct dvb_desc *desc);
00096
00103 void dvb_desc_registration_free(struct dvb_desc *desc);
00104
00105 #ifdef __cplusplus
00106 }
00107 #endif
00108
00109 #endif

```

9.42 lib/include/libdvbv5/desc_sat.h File Reference

Provides the descriptors for the satellite delivery system descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_sat](#)
Structure containing the satellite delivery system descriptor.

Functions

- int [dvb_desc_sat_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the satellite delivery system descriptor.
- void [dvb_desc_sat_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the satellite delivery system descriptor.

Variables

- const unsigned [dvbs_dvbc_dvbs_freq_inner](#) []
converts from the descriptor's FEC into enum [fe_code_rate](#), as defined by DVBv5 API.
- const unsigned [dvbs_polarization](#) []
converts from the descriptor's polarization into enum [dvb_sat_polarization](#), as defined at [dvb-v5-std.h](#).
- const unsigned [dvbs_rolloff](#) []
converts from the descriptor's rolloff into enum [fe_rolloff](#), as defined by DVBv5 API.
- const unsigned [dvbs_modulation](#) []
converts from the descriptor's modulation into enum [fe_modulation](#), as defined by DVBv5 API.

9.42.1 Detailed Description

Provides the descriptors for the satellite delivery system descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_sat.h](#).

9.42.2 Variable Documentation

9.42.2.1 dvbs_dvbc_dvbs_freq_inner `const unsigned dvbs_dvbc_dvbs_freq_inner[]` [extern]

converts from the descriptor's FEC into enum `fe_code_rate`, as defined by DVBv5 API.

9.42.2.2 dvbs_modulation `const unsigned dvbs_modulation[]` [extern]

converts from the descriptor's modulation into enum `fe_modulation`, as defined by DVBv5 API.

9.42.2.3 dvbs_polarization `const unsigned dvbs_polarization[]` [extern]

converts from the descriptor's polarization into enum `dvb_sat_polarization`, as defined at [dvb-v5-std.h](#).

9.42.2.4 dvbs_rolloff `const unsigned dvbs_rolloff[] [extern]`

converts from the descriptor's rolloff into enum `fe_rolloff`, as defined by DVBv5 API.

9.43 desc_sat.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00037 #ifndef _SAT_H
00038 #define _SAT_H
00039
00040 #include <libdvbv5/descriptors.h>
00041
00063 struct dvb_desc_sat {
00064     uint8_t type;
00065     uint8_t length;
00066     struct dvb_desc *next;
00067
00068     uint32_t frequency;
00069     uint16_t orbit;
00070     uint8_t modulation_type:2;
00071     uint8_t modulation_system:1;
00072     uint8_t roll_off:2;
00073     uint8_t polarization:2;
00074     uint8_t west_east:1;
00075     union {
00076         uint32_t bitfield;
00077         struct {
00078             uint32_t fec:4;
00079             uint32_t symbol_rate:28;
00080         } __attribute__((packed));
00081     } __attribute__((packed));
00082 } __attribute__((packed));
00083
00084 struct dvb_v5_fe_parms;
00085
00086 #ifdef __cplusplus
00087 extern "C" {
00088 #endif
00089
00106 int dvb_desc_sat_init(struct dvb_v5_fe_parms *parms,
00107                     const uint8_t *buf, struct dvb_desc *desc);
00108
00116 void dvb_desc_sat_print(struct dvb_v5_fe_parms *parms,
00117                       const struct dvb_desc *desc);
00118
00123 extern const unsigned dvbs_dvbc_dvbs_freq_inner[];
00124
00129 extern const unsigned dvbs_polarization[];
00130
00135 extern const unsigned dvbs_rolloff[];
00136
00141 extern const unsigned dvbs_modulation[];
00142
00143 #ifdef __cplusplus
00144 }
00145 #endif
00146
00147 #endif
```

9.44 lib/include/libdvbv5/desc_service.h File Reference

Provides the descriptors for the service descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_service](#)
Structure containing the service descriptor.

Functions

- int [dvb_desc_service_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the service descriptor.
- void [dvb_desc_service_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the service descriptor.
- void [dvb_desc_service_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the service descriptor.

9.44.1 Detailed Description

Provides the descriptors for the service descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_service.h](#).

9.45 desc_service.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00037 #ifndef _DESC_SERVICE_H
00038 #define _DESC_SERVICE_H
00039
00040 #include <libdvbv5/descriptors.h>
00041
00060 struct dvb_desc_service {
00061     uint8_t type;
00062     uint8_t length;
00063     struct dvb_desc *next;
00064
00065     uint8_t service_type;
00066     char *name;
00067     char *name_emph;
00068     char *provider;
00069     char *provider_emph;
00070 } __attribute__((packed));
00071
00072 struct dvb_v5_fe_parms;
00073
00074 #ifdef __cplusplus
00075 extern "C" {
00076 #endif
00077
00093 int dvb_desc_service_init(struct dvb_v5_fe_parms *parms,
00094                          const uint8_t *buf, struct dvb_desc *desc);
00095
00103 void dvb_desc_service_print(struct dvb_v5_fe_parms *parms,
00104                             const struct dvb_desc *desc);
00105
00112 void dvb_desc_service_free(struct dvb_desc *desc);
00113
00114 #ifdef __cplusplus
00115 }
00116 #endif
00117
00118 #endif

```

9.46 lib/include/libdvbv5/desc_t2_delivery.h File Reference

Provides the descriptors for the DVB-T2 delivery system descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- [struct dvb_desc_t2_delivery_subcell_old](#)
Structure to describe transponder subcell extension and frequencies.
- [struct dvb_desc_t2_delivery_subcell](#)
Structure to describe transponder subcell extension and frequencies.

- struct [dvb_desc_t2_delivery_cell](#)
Structure to describe transponder cells.
- struct [dvb_desc_t2_delivery](#)
Structure containing the T2 delivery system descriptor.

Functions

- int [dvb_desc_t2_delivery_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_extension_descriptor](#) *ext, void *desc)
Initializes and parses the T2 delivery system descriptor.
- void [dvb_desc_t2_delivery_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_extension_descriptor](#) *ext, const void *desc)
Prints the content of the T2 delivery system descriptor.
- void [dvb_desc_t2_delivery_free](#) (const void *desc)
Frees all data allocated by the T2 delivery system descriptor.

Variables

- const unsigned [dvbt2_bw](#) []
converts from internal representation into bandwidth in Hz
- const uint32_t [dvbt2_interval](#) []
converts from internal representation into enum fe_guard_interval, as defined at DVBv5 API.
- const unsigned [dvbt2_transmission_mode](#) []
converts from the descriptor's transmission mode into enum fe_transmit_mode, as defined by DVBv5 API.
- const char * [siso_miso](#) [4]
converts from internal representation to string the SISO_MISO field of [dvb_desc_t2_delivery](#):SISO_MISO field.

9.46.1 Detailed Description

Provides the descriptors for the DVB-T2 delivery system descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_t2_delivery.h](#).

9.46.2 Variable Documentation

9.46.2.1 dvbt2_bw `const unsigned dvbt2_bw[]` [extern]

converts from internal representation into bandwidth in Hz

9.46.2.2 dvbt2_interval `const uint32_t dvbt2_interval[]` [extern]

converts from internal representation into enum `fe_guard_interval`, as defined at DVBv5 API.

9.46.2.3 dvbt2_transmission_mode `const unsigned dvbt2_transmission_mode[]` [extern]

converts from the descriptor's transmission mode into enum `fe_transmit_mode`, as defined by DVBv5 API.

9.46.2.4 siso_miso `const char* siso_miso[4]` [extern]

converts from internal representation to string the SISO_MISO field of `dvb_desc_t2_delivery`:SISO_MISO field.

9.47 desc_t2_delivery.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2013-2014 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Based on ETSI EN 300 468 V1.11.1 (2010-04)
00019  */
00020
00036 #ifndef _T2_DELIVERY_H
00037 #define _T2_DELIVERY_H
00038
00039 #include <libdvbv5/descriptors.h>
00040
00055 struct dvb_desc_t2_delivery_subcell_old {
00056     uint8_t cell_id_extension;
00057     uint16_t transposer_frequency;           // Should be 32 bits, instead
00058 } __attribute__((packed));
00059
00060
00069 struct dvb_desc_t2_delivery_subcell {
00070     uint8_t cell_id_extension;
00071     uint32_t transposer_frequency;
```

```

00072 } __attribute__((packed));
00073
00074
00086 struct dvb_desc_t2_delivery_cell {
00087     uint16_t cell_id;
00088     int num_freqs;
00089     uint32_t *centre_frequency;
00090     uint8_t subcel_length;
00091     struct dvb_desc_t2_delivery_subcell *subcel;
00092 } __attribute__((packed));
00093
00119 struct dvb_desc_t2_delivery {
00120     /* extended descriptor */
00121
00122     uint8_t plp_id;
00123     uint16_t system_id;
00124     union {
00125         uint16_t bitfield;
00126         struct {
00127             uint16_t tfs_flag:1;
00128             uint16_t other_frequency_flag:1;
00129             uint16_t transmission_mode:3;
00130             uint16_t guard_interval:3;
00131             uint16_t reserved:2;
00132             uint16_t bandwidth:4;
00133             uint16_t SISO_MISO:2;
00134         } __attribute__((packed));
00135     } __attribute__((packed));
00136
00137     uint32_t *centre_frequency;
00138     uint8_t frequency_loop_length;
00139
00140     /* Unused, as the definitions here are incomplete. */
00141     uint8_t subcel_info_loop_length;
00142     struct dvb_desc_t2_delivery_subcell_old *subcell;
00143
00144     /* Since version 1.13 */
00145     unsigned int num_cell;
00146     struct dvb_desc_t2_delivery_cell *cell;
00147 } __attribute__((packed));
00148
00149
00150 struct dvb_v5_fe_parms;
00151
00152 #ifdef __cplusplus
00153 extern "C" {
00154 #endif
00155
00172 int dvb_desc_t2_delivery_init(struct dvb_v5_fe_parms *parms,
00173                             const uint8_t *buf,
00174                             struct dvb_extension_descriptor *ext,
00175                             void *desc);
00176
00185 void dvb_desc_t2_delivery_print(struct dvb_v5_fe_parms *parms,
00186                               const struct dvb_extension_descriptor *ext,
00187                               const void *desc);
00188
00195 void dvb_desc_t2_delivery_free(const void *desc);
00196
00200 extern const unsigned dvbt2_bw[];
00201
00206 extern const uint32_t dvbt2_interval[];
00207
00212 extern const unsigned dvbt2_transmission_mode[];
00213
00218 extern const char *siso_miso[4];
00219
00220 #ifdef __cplusplus
00221 }
00222 #endif
00223
00224 #endif

```

9.48 lib/include/libdvbv5/desc_terrestrial_delivery.h File Reference

Provides the descriptors for the DVB-T terrestrial delivery system descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_terrestrial_delivery](#)
Structure containing the DVB-T terrestrial delivery system descriptor.

Functions

- int [dvb_desc_terrestrial_delivery_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the DVB-T terrestrial delivery system descriptor.
- void [dvb_desc_terrestrial_delivery_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the DVB-T terrestrial delivery system descriptor.

Variables

- const unsigned [dvbt_bw](#) []
converts from internal representation into bandwidth in Hz
- const unsigned [dvbt_modulation](#) []
converts from the descriptor's modulation into enum fe_modulation, as defined by DVBv5 API.
- const unsigned [dvbt_hierarchy](#) []
converts from the descriptor's hierarchy into enum fe_hierarchy, as defined by DVBv5 API.
- const unsigned [dvbt_code_rate](#) []
converts from the descriptor's FEC into enum fe_code_rate, as defined by DVBv5 API.
- const uint32_t [dvbt_interval](#) []
converts from internal representation into enum fe_guard_interval, as defined at DVBv5 API.
- const unsigned [dvbt_transmission_mode](#) []
converts from the descriptor's transmission mode into enum fe_transmit_mode, as defined by DVBv5 API.

9.48.1 Detailed Description

Provides the descriptors for the DVB-T terrestrial delivery system descriptor.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptor described herein is defined at:

- ETSI EN 300 468 V1.11.1

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_terrestrial_delivery.h](#).

9.48.2 Function Documentation

9.48.2.1 dvb_desc_terrestrial_delivery_init() `int dvb_desc_terrestrial_delivery_init (struct dvb_v5_fe_parms * parms, const uint8_t * buf, struct dvb_desc * desc)`

Initializes and parses the DVB-T terrestrial delivery system descriptor.

Parameters

<i>parms</i>	struct dvb_v5_fe_parms pointer to the opened device
<i>buf</i>	buffer containing the descriptor's raw data
<i>desc</i>	pointer to struct dvb_desc to be allocated and filled

This function initializes and makes sure that all fields will follow the CPU endianness. Due to that, the content of the buffer may change.

Currently, no memory is allocated internally.

Returns

On success, it returns the size of the allocated struct. A negative value indicates an error.

9.48.3 Variable Documentation

9.48.3.1 dvbt_bw `const unsigned dvbt_bw[] [extern]`

converts from internal representation into bandwidth in Hz

9.48.3.2 dvbt_code_rate `const unsigned dvbt_code_rate[] [extern]`

converts from the descriptor's FEC into enum `fe_code_rate`, as defined by DVBv5 API.

9.48.3.3 dvbt_hierarchy `const unsigned dvbt_hierarchy[] [extern]`

converts from the descriptor's hierarchy into enum `fe_hierarchy`, as defined by DVBv5 API.

9.48.3.4 dvbt_interval `const uint32_t dvbt_interval[] [extern]`

converts from internal representation into enum `fe_guard_interval`, as defined at DVBv5 API.

9.48.3.5 dvbt_modulation `const unsigned dvbt_modulation[] [extern]`

converts from the descriptor's modulation into enum `fe_modulation`, as defined by DVBv5 API.

9.48.3.6 dvbt_transmission_mode `const unsigned dvbt_transmission_mode[] [extern]`

converts from the descriptor's transmission mode into enum `fe_transmit_mode`, as defined by DVBv5 API.

9.49 desc_terrestrial_delivery.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  * Based on ETSI EN 300 468 V1.11.1 (2010-04)
00020  *
00021  */
00022
00039 #ifndef _TERRESTRIAL_DELIVERY_H
00040 #define _TERRESTRIAL_DELIVERY_H
00041
00042 #include <libdvbv5/descriptors.h>
00043
00065 struct dvb_desc_terrestrial_delivery {
00066     uint8_t type;
00067     uint8_t length;
00068     struct dvb_desc *next;
00069
00070     uint32_t centre_frequency;
00071     uint8_t reserved_future_use1:2;
00072     uint8_t mpe_fec_indicator:1;
00073     uint8_t time_slice_indicator:1;
00074     uint8_t priority:1;
00075     uint8_t bandwidth:3;
00076     uint8_t code_rate_hp_stream:3;
00077     uint8_t hierarchy_information:3;
00078     uint8_t constellation:2;
00079     uint8_t other_frequency_flag:1;
00080     uint8_t transmission_mode:2;
00081     uint8_t guard_interval:2;
00082     uint8_t code_rate_lp_stream:3;
00083     uint32_t reserved_future_use2;
00084 } __attribute__((packed));
00085
00086 struct dvb_v5_fe_parms;
00087
00088 #ifdef __cplusplus
00089 extern "C" {
00090 #endif

```

```

00091
00107 int dvb_desc_terrestrial_delivery_init(struct dvb_v5_fe_parms *parms,
00108                                     const uint8_t *buf,
00109                                     struct dvb_desc *desc);
00110
00118 void dvb_desc_terrestrial_delivery_print(struct dvb_v5_fe_parms *parms,
00119                                       const struct dvb_desc *desc);
00120
00124 extern const unsigned dvbt_bw[];
00125
00130 extern const unsigned dvbt_modulation[];
00131
00136 extern const unsigned dvbt_hierarchy[];
00137
00142 extern const unsigned dvbt_code_rate[];
00143
00148 extern const uint32_t dvbt_interval[];
00149
00154 extern const unsigned dvbt_transmission_mode[];
00155
00156 #ifdef __cplusplus
00157 }
00158 #endif
00159
00160 #endif

```

9.50 lib/include/libdvbv5/desc_ts_info.h File Reference

Provides the descriptors for the ISDB TS information descriptor.

```
#include <libdvbv5/descriptors.h>
```

Data Structures

- struct [dvb_desc_ts_info_transmission_type](#)
ISDB TS information transmission type.
- struct [dvb_desc_ts_info](#)
Structure describing the ISDB TS information descriptor.

Functions

- int [dvb_desc_ts_info_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Initializes and parses the ISDB TS information descriptor.
- void [dvb_desc_ts_info_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Prints the content of the ISDB TS information descriptor.
- void [dvb_desc_ts_info_free](#) (struct [dvb_desc](#) *desc)
Frees all data allocated by the ISDB TS information descriptor.

9.50.1 Detailed Description

Provides the descriptors for the ISDB TS information descriptor.

The TS information descriptor specifies the remote control key identifier assigned to the applicable TS and indicates the relationship between the service identifier and the transmission layer during hierarchical transmission.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Relevant specs

The descriptor described herein is defined at:

- ARIB STD-B10

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [desc_ts_info.h](#).

9.51 desc_ts_info.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2013-2014 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Described on ARIB STD-B10 as TS information descriptor
00019  */
00020
00041 #ifndef _TS_INFO_H
00042 #define _TS_INFO_H
00043
00044 #include <libdvbv5/descriptors.h>
00045
00054 struct dvb_desc_ts_info_transmission_type {
00055     uint8_t transmission_type_info;
00056     uint8_t num_of_service;
00057 } __attribute__((packed));
00058
00076 struct dvb_desc_ts_info {
00077     uint8_t type;
00078     uint8_t length;
00079     struct dvb_desc *next;
00080
00081     char *ts_name, *ts_name_emph;
00082     struct dvb_desc_ts_info_transmission_type transmission_type;
00083     uint16_t *service_id;
00084
00085     union {
00086         uint16_t bitfield;
00087         struct {
00088             uint8_t transmission_type_count:2;
00089             uint8_t length_of_ts_name:6;
00090             uint8_t remote_control_key_id:8;
00091         } __attribute__((packed));
00092     };
00093 } __attribute__((packed));
00094
00095 struct dvb_v5_fe_parms;
00096
00097 #ifdef __cplusplus
00098 extern "C" {
00099 #endif
00100
```

```

00117 int dvb_desc_ts_info_init(struct dvb_v5_fe_parms *parms,
00118                           const uint8_t *buf, struct dvb_desc *desc);
00119
00128 void dvb_desc_ts_info_print(struct dvb_v5_fe_parms *parms,
00129                             const struct dvb_desc *desc);
00130
00138 void dvb_desc_ts_info_free(struct dvb_desc *desc);
00139
00140 #ifdef __cplusplus
00141 }
00142 #endif
00143
00144 #endif

```

9.52 lib/include/libdvbv5/descriptors.h File Reference

Provides a way to handle MPEG-TS descriptors found on Digital TV streams.

```

#include <unistd.h>
#include <stdint.h>
#include <arpa/inet.h>

```

Data Structures

- struct [dvb_desc](#)
Linked list containing the several descriptors found on a MPEG-TS table.
- struct [dvb_descriptor](#)
Contains the parser information for the MPEG-TS parser code.

Macros

- #define [DVB_MAX_PAYLOAD_PACKET_SIZE](#)
Maximum size of a table session to be parsed.
- #define [DVB_CRC_SIZE](#)
number of bytes for the descriptor's CRC check
- #define [TS_Information_descriptor](#)

Typedefs

- typedef void(* [dvb_table_init_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, void **table)
Function prototype for a function that initializes the descriptors parsing on a table.
- typedef int(* [dvb_desc_init_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, struct [dvb_desc](#) *desc)
Function prototype for the descriptors parsing init code.
- typedef void(* [dvb_desc_print_func](#)) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_desc](#) *desc)
Function prototype for the descriptors parsing print code.
- typedef void(* [dvb_desc_free_func](#)) (struct [dvb_desc](#) *desc)
Function prototype for the descriptors memory free code.

Enumerations

```

• enum descriptors {
    video_stream_descriptor , audio_stream_descriptor , hierarchy_descriptor , registration_descriptor ,
    ds_alignment_descriptor , target_background_grid_descriptor , video_window_descriptor , conditional_access_descriptor
    ,
    iso639_language_descriptor , system_clock_descriptor , multiplex_buffer_utilization_descriptor , copyright_descriptor
    ,
    maximum_bitrate_descriptor , private_data_indicator_descriptor , smoothing_buffer_descriptor , std_descriptor
    ,
    ibp_descriptor , mpeg4_video_descriptor , mpeg4_audio_descriptor , iod_descriptor ,
    sl_descriptor , fmc_descriptor , external_es_id_descriptor , muxcode_descriptor ,
    fmxbuffersize_descriptor , multiplexbuffer_descriptor , content_labeling_descriptor , metadata_pointer_descriptor
    ,
    metadata_descriptor , metadata_std_descriptor , AVC_video_descriptor , ipmp_descriptor ,
    AVC_timing_and_HRD_descriptor , mpeg2_aac_audio_descriptor , flexmux_timing_descriptor , network_name_descriptor
    ,
    service_list_descriptor , stuffing_descriptor , satellite_delivery_system_descriptor , cable_delivery_system_descriptor
    ,
    VBI_data_descriptor , VBI_teletext_descriptor , bouquet_name_descriptor , service_descriptor ,
    country_availability_descriptor , linkage_descriptor , NVOB_reference_descriptor , time_shifted_service_descriptor
    ,
    short_event_descriptor , extended_event_descriptor , time_shifted_event_descriptor , component_descriptor
    ,
    mosaic_descriptor , stream_identifier_descriptor , CA_identifier_descriptor , content_descriptor ,
    parental_rating_descriptor , teletext_descriptor , telephone_descriptor , local_time_offset_descriptor ,
    subtitling_descriptor , terrestrial_delivery_system_descriptor , multilingual_network_name_descriptor ,
    multilingual_bouquet_name_descriptor ,
    multilingual_service_name_descriptor , multilingual_component_descriptor , private_data_specifier_descriptor
    , service_move_descriptor ,
    short_smoothing_buffer_descriptor , frequency_list_descriptor , partial_transport_stream_descriptor ,
    data_broadcast_descriptor ,
    scrambling_descriptor , data_broadcast_id_descriptor , transport_stream_descriptor , DSNG_descriptor ,
    PDC_descriptor , AC_3_descriptor , ancillary_data_descriptor , cell_list_descriptor ,
    cell_frequency_link_descriptor , announcement_support_descriptor , application_signalling_descriptor ,
    adaptation_field_data_descriptor ,
    service_identifier_descriptor , service_availability_descriptor , default_authority_descriptor , related_content_descriptor
    ,
    TVA_id_descriptor , content_identifier_descriptor , time_slice_fec_identifier_descriptor , ECM_repetition_rate_descriptor
    ,
    S2_satellite_delivery_system_descriptor , enhanced_AC_3_descriptor , DTS_descriptor , AAC_descriptor ,
    XAIT_location_descriptor , FTA_content_management_descriptor , extension_descriptor , CUE_identifier_descriptor
    ,
    extended_channel_name , service_location , component_name_descriptor , logical_channel_number_descriptor
    ,
    carousel_id_descriptor , association_tag_descriptor , deferred_association_tags_descriptor , hierarchical_transmission_descriptor
    ,
    digital_copy_control_descriptor , network_identifier_descriptor , partial_transport_stream_time_descriptor ,
    audio_component_descriptor ,
    hyperlink_descriptor , target_area_descriptor , data_contents_descriptor , video_decode_control_descriptor ,
    download_content_descriptor , CA_EMM_TS_descriptor , CA_contract_information_descriptor , CA_service_descriptor
    ,
    TS_Information_descriptor , extended_broadcaster_descriptor , logo_transmission_descriptor , basic_local_event_descriptor
    ,
    reference_descriptor , node_relation_descriptor , short_node_information_descriptor , STC_reference_descriptor
    ,
    series_descriptor , event_group_descriptor , SI_parameter_descriptor , broadcaster_Name_Descriptor ,
    component_group_descriptor , SI_prime_TS_descriptor , board_information_descriptor , LDT_linkage_descriptor
    ,

```

```
connected_transmission_descriptor , content_availability_descriptor , service_group_descriptor , carousel_compatible_compos
,
conditional_playback_descriptor , ISDBT_delivery_system_descriptor , partial_reception_descriptor ,
emergency_information_descriptor ,
data_component_descriptor , system_management_descriptor , atsc_stuffing_descriptor , atsc_ac3_audio_descriptor
,
atsc_caption_service_descriptor , atsc_content_advisory_descriptor , atsc_extended_channel_descriptor ,
atsc_service_location_descriptor ,
atsc_time_shifted_service_descriptor , atsc_component_name_descriptor , atsc_DCC_departing_request_descriptor
, atsc_DCC_arriving_request_descriptor ,
atsc_redistribution_control_descriptor , atsc_ATSC_private_information_descriptor , atsc_genre_descriptor }
```

List containing all descriptors used by Digital TV MPEG-TS.

Functions

- uint32_t [dvb_bcd](#) (uint32_t bcd)
Converts from BCD to CPU integer internal representation.
- void [dvb_hexdump](#) (struct [dvb_v5_fe_parms](#) *parms, const char *prefix, const unsigned char *buf, int len)
dumps data into the logs in hexadecimal format
- int [dvb_desc_parse](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, uint16_t buflen, struct [dvb_desc](#) **head_desc)
parse MPEG-TS descriptors
- void [dvb_desc_free](#) (struct [dvb_desc](#) **list)
frees a [dvb_desc](#) linked list
- void [dvb_desc_print](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_desc](#) *desc)
prints the contents of a struct [dvb_desc](#) linked list

Variables

- const [dvb_table_init_func](#) [dvb_table_initializers](#) [256]
Table with all possible descriptors.
- const struct [dvb_descriptor](#) [dvb_descriptors](#) []
Contains the parsers for the several descriptors.

9.52.1 Detailed Description

Provides a way to handle MPEG-TS descriptors found on Digital TV streams.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The descriptors herein are defined on the following specs:

- ISO/IEC 13818-1
- ETSI EN 300 468 V1.11.1 (2010-04)
- SCTE 35 2004
- <http://www.etherguidesystems.com/Help/SDOs/ATSC/Semantics/Descriptors/Default.aspx>
- <http://www.coolstf.com/tsreader/descriptors.html>
- ABNT NBR 15603-1 2007
- ATSC A/65:2009 spec

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [descriptors.h](#).

9.52.2 Macro Definition Documentation

9.52.2.1 TS_Information_descriptor #define TS_Information_descriptor

Definition at line 776 of file [descriptors.h](#).

9.53 descriptors.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012-2014 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  */
00019
00044 #ifndef _DESCRIPTORS_H
00045 #define _DESCRIPTORS_H
00046
00047 #include <unistd.h>
00048 #include <stdint.h>
00049 #include <arpa/inet.h>
00050
00055 #define DVB_MAX_PAYLOAD_PACKET_SIZE 4096
00056
00061 #define DVB_CRC_SIZE 4
00062
00063
00064 #ifndef _DOXYGEN
```

```

00065 struct dvb_v5_fe_parms;
00066 #endif
00067
00079 typedef void (*dvb_table_init_func) (struct dvb_v5_fe_parms *parms,
00080                                     const uint8_t *buf, ssize_t buflen,
00081                                     void **table);
00082
00087 extern const dvb_table_init_func dvb_table_initializers[256];
00088
00089 #ifndef _DOXYGEN
00090 #define bswap16(b) do {\
00091     b = ntohs(b); \
00092 } while (0)
00093
00094 #define bswap32(b) do {\
00095     b = ntohl(b); \
00096 } while (0)
00097
00098 /* Deprecated */
00099 #define DVB_DESC_HEADER() \
00100     uint8_t type; \
00101     uint8_t length; \
00102     struct dvb_desc *next
00103
00104 #endif /* _DOXYGEN */
00105
00117 struct dvb_desc {
00118     uint8_t type;
00119     uint8_t length;
00120     struct dvb_desc *next;
00121
00122     uint8_t data[];
00123 } __attribute__((packed));
00124
00125 #ifndef _DOXYGEN
00126
00127 #define dvb_desc_foreach( _desc, _tbl ) \
00128     if ( _tbl && _tbl->descriptor ) \
00129         for( struct dvb_desc *_desc = _tbl->descriptor; _desc; _desc = _desc->next ) \
00130
00131 #define dvb_desc_find( _struct, _desc, _tbl, _type ) \
00132     if ( _tbl && _tbl->descriptor ) \
00133         for( _struct *_desc = ( _struct *) _tbl->descriptor; _desc; _desc = ( _struct *) \
00134             _desc->next ) \
00135             if( _desc->type == _type ) \
00136
00137 #endif /* _DOXYGEN */
00138
00138 #ifdef __cplusplus
00139 extern "C" {
00140 #endif
00141
00148 uint32_t dvb_bcd(uint32_t bcd);
00149
00159 void dvb_hexdump(struct dvb_v5_fe_parms *parms, const char *prefix,
00160                 const unsigned char *buf, int len);
00161
00183 int dvb_desc_parse(struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00184                  uint16_t buflen, struct dvb_desc **head_desc);
00185
00192 void dvb_desc_free (struct dvb_desc **list);
00193
00201 void dvb_desc_print(struct dvb_v5_fe_parms *parms, struct dvb_desc *desc);
00202
00203 #ifdef __cplusplus
00204 }
00205 #endif
00206
00215 typedef int (*dvb_desc_init_func) (struct dvb_v5_fe_parms *parms,
00216                                   const uint8_t *buf, struct dvb_desc *desc);
00217
00225 typedef void (*dvb_desc_print_func) (struct dvb_v5_fe_parms *parms,
00226                                      const struct dvb_desc *desc);
00227
00234 typedef void (*dvb_desc_free_func) (struct dvb_desc *desc);
00235
00249 struct dvb_descriptor {
00250     const char *name;
00251     dvb_desc_init_func init;
00252     dvb_desc_print_func print;
00253     dvb_desc_free_func free;
00254     ssize_t size;
00255 };
00256
00261 extern const struct dvb_descriptor dvb_descriptors[];
00262
00592 enum descriptors {

```

```

00593      /* ISO/IEC 13818-1 */
00594      video_stream_descriptor                = 0x02,
00595      audio_stream_descriptor                = 0x03,
00596      hierarchy_descriptor                  = 0x04,
00597      registration_descriptor                = 0x05,
00598      ds_alignment_descriptor                = 0x06,
00599      target_background_grid_descriptor      = 0x07,
00600      video_window_descriptor                = 0x08,
00601      conditional_access_descriptor          = 0x09,
00602      iso639_language_descriptor            = 0x0a,
00603      system_clock_descriptor                = 0x0b,
00604      multiplex_buffer_utilization_descriptor = 0x0c,
00605      copyright_descriptor                  = 0x0d,
00606      maximum_bitrate_descriptor            = 0x0e,
00607      private_data_indicator_descriptor      = 0x0f,
00608      smoothing_buffer_descriptor           = 0x10,
00609      std_descriptor                        = 0x11,
00610      ibp_descriptor                        = 0x12,
00611
00612      mpeg4_video_descriptor                 = 0x1b,
00613      mpeg4_audio_descriptor                 = 0x1c,
00614      iod_descriptor                        = 0x1d,
00615      sl_descriptor                         = 0x1e,
00616      fmc_descriptor                        = 0x1f,
00617      external_es_id_descriptor              = 0x20,
00618      muxcode_descriptor                    = 0x21,
00619      fmxbuffer_size_descriptor              = 0x22,
00620      multiplexbuffer_descriptor             = 0x23,
00621      content_labeling_descriptor           = 0x24,
00622      metadata_pointer_descriptor            = 0x25,
00623      metadata_descriptor                   = 0x26,
00624      metadata_std_descriptor                = 0x27,
00625      AVC_video_descriptor                   = 0x28,
00626      ipmp_descriptor                       = 0x29,
00627      AVC_timing_and_HRD_descriptor          = 0x2a,
00628      mpeg2_aac_audio_descriptor            = 0x2b,
00629      flexmux_timing_descriptor              = 0x2c,
00630
00631      /* ETSI EN 300 468 V1.11.1 (2010-04) */
00632
00633      network_name_descriptor                = 0x40,
00634      service_list_descriptor                = 0x41,
00635      stuffing_descriptor                    = 0x42,
00636      satellite_delivery_system_descriptor    = 0x43,
00637      cable_delivery_system_descriptor        = 0x44,
00638      VBI_data_descriptor                    = 0x45,
00639      VBI_teletext_descriptor                = 0x46,
00640      bouquet_name_descriptor                = 0x47,
00641      service_descriptor                     = 0x48,
00642      country_availability_descriptor          = 0x49,
00643      linkage_descriptor                     = 0x4a,
00644      NVOB_reference_descriptor               = 0x4b,
00645      time_shifted_service_descriptor         = 0x4c,
00646      short_event_descriptor                 = 0x4d,
00647      extended_event_descriptor              = 0x4e,
00648      time_shifted_event_descriptor           = 0x4f,
00649      component_descriptor                   = 0x50,
00650      mosaic_descriptor                      = 0x51,
00651      stream_identifier_descriptor            = 0x52,
00652      CA_identifier_descriptor                = 0x53,
00653      content_descriptor                     = 0x54,
00654      parental_rating_descriptor             = 0x55,
00655      teletext_descriptor                    = 0x56,
00656      telephone_descriptor                   = 0x57,
00657      local_time_offset_descriptor            = 0x58,
00658      subtitling_descriptor                  = 0x59,
00659      terrestrial_delivery_system_descriptor  = 0x5a,
00660      multilingual_network_name_descriptor    = 0x5b,
00661      multilingual_bouquet_name_descriptor    = 0x5c,
00662      multilingual_service_name_descriptor    = 0x5d,
00663      multilingual_component_descriptor       = 0x5e,
00664      private_data_specifier_descriptor       = 0x5f,
00665      service_move_descriptor                = 0x60,
00666      short_smoothing_buffer_descriptor       = 0x61,
00667      frequency_list_descriptor              = 0x62,
00668      partial_transport_stream_descriptor     = 0x63,
00669      data_broadcast_descriptor              = 0x64,
00670      scrambling_descriptor                  = 0x65,
00671      data_broadcast_id_descriptor            = 0x66,
00672      transport_stream_descriptor             = 0x67,
00673      DSNG_descriptor                        = 0x68,
00674      PDC_descriptor                         = 0x69,
00675      AC_3_descriptor                        = 0x6a,
00676      ancillary_data_descriptor              = 0x6b,
00677      cell_list_descriptor                   = 0x6c,
00678      cell_frequency_link_descriptor          = 0x6d,
00679      announcement_support_descriptor         = 0x6e,

```

```

00680     application_signalling_descriptor           = 0x6f,
00681     adaptation_field_data_descriptor           = 0x70,
00682     service_identifier_descriptor             = 0x71,
00683     service_availability_descriptor           = 0x72,
00684     default_authority_descriptor             = 0x73,
00685     related_content_descriptor               = 0x74,
00686     TVA_id_descriptor                        = 0x75,
00687     content_identifier_descriptor             = 0x76,
00688     time_slice_fec_identifier_descriptor      = 0x77,
00689     ECM_repetition_rate_descriptor           = 0x78,
00690     S2_satellite_delivery_system_descriptor = 0x79,
00691     enhanced_AC_3_descriptor                 = 0x7a,
00692     DTS_descriptor                          = 0x7b,
00693     AAC_descriptor                          = 0x7c,
00694     XAIT_location_descriptor                 = 0x7d,
00695     FTA_content_management_descriptor        = 0x7e,
00696     extension_descriptor                     = 0x7f,
00697
00698     /* SCTE 35 2004 */
00699     CUE_identifier_descriptor                 = 0x8a,
00700
00701     extended_channel_name                     = 0xa0,
00702     service_location                         = 0xa1,
00703     /* From http://www.etherguidesystems.com/Help/SDOs/ATSC/Semantics/Descriptors/Default.aspx */
00704     component_name_descriptor                 = 0xa3,
00705
00706     /* From http://www.coolstf.com/tsreader/descriptors.html */
00707     logical_channel_number_descriptor         = 0x83,
00708
00709     /* ISDB Descriptors, as defined on ABNT NBR 15603-1 2007 */
00710
00711     carousel_id_descriptor                   = 0x13,
00712     association_tag_descriptor               = 0x14,
00713     deferred_association_tags_descriptor     = 0x15,
00714
00715     hierarchical_transmission_descriptor     = 0xc0,
00716     digital_copy_control_descriptor          = 0xc1,
00717     network_identifier_descriptor            = 0xc2,
00718     partial_transport_stream_time_descriptor = 0xc3,
00719     audio_component_descriptor               = 0xc4,
00720     hyperlink_descriptor                    = 0xc5,
00721     target_area_descriptor                   = 0xc6,
00722     data_contents_descriptor                 = 0xc7,
00723     video_decode_control_descriptor          = 0xc8,
00724     download_content_descriptor              = 0xc9,
00725     CA_EMM_TS_descriptor                    = 0xca,
00726     CA_contract_information_descriptor        = 0xcb,
00727     CA_service_descriptor                    = 0xcc,
00728     TS_Information_descriptor                = 0xcd,
00729     extended_broadcaster_descriptor          = 0xce,
00730     logo_transmission_descriptor             = 0xcf,
00731     basic_local_event_descriptor             = 0xd0,
00732     reference_descriptor                     = 0xd1,
00733     node_relation_descriptor                 = 0xd2,
00734     short_node_information_descriptor         = 0xd3,
00735     STC_reference_descriptor                 = 0xd4,
00736     series_descriptor                        = 0xd5,
00737     event_group_descriptor                   = 0xd6,
00738     SI_parameter_descriptor                  = 0xd7,
00739     broadcaster_Name_Descriptor              = 0xd8,
00740     component_group_descriptor               = 0xd9,
00741     SI_prime_TS_descriptor                   = 0xda,
00742     board_information_descriptor              = 0xdb,
00743     LDT_linkage_descriptor                   = 0xdc,
00744     connected_transmission_descriptor         = 0xdd,
00745     content_availability_descriptor           = 0xde,
00746     service_group_descriptor                 = 0xe0,
00747     carousel_compatible_composite_descriptor = 0xf7,
00748     conditional_playback_descriptor          = 0xf8,
00749     ISDBT_delivery_system_descriptor         = 0xfa,
00750     partial_reception_descriptor             = 0xfb,
00751     emergency_information_descriptor          = 0xfc,
00752     data_component_descriptor                = 0xfd,
00753     system_management_descriptor             = 0xfe,
00754
00755     /* ATSC descriptors - ATSC A/65:2009 spec */
00756     atsc_stuffing_descriptor                 = 0x80,
00757     atsc_ac3_audio_descriptor                = 0x81,
00758     atsc_caption_service_descriptor          = 0x86,
00759     atsc_content_advisory_descriptor         = 0x87,
00760     atsc_extended_channel_descriptor         = 0xa0,
00761     atsc_service_location_descriptor         = 0xa1,
00762     atsc_time_shifted_service_descriptor     = 0xa2,
00763     atsc_component_name_descriptor           = 0xa3,
00764     atsc_DCC_departing_request_descriptor    = 0xa8,
00765     atsc_DCC_arriving_request_descriptor     = 0xa9,
00766     atsc_redistribution_control_descriptor    = 0xaa,

```

```
00767         atsc_ATSC_private_information_descriptor      = 0xad,
00768         atsc_genre_descriptor                        = 0xab,
00769     };
00770
00771
00772     /*
00773     * NOTE: this is here just to avoid API break. There was a typo
00774     * on the name of this descriptor
00775     */
00776     #define TS_Information_descriptor TS_Information_descriptor
00777
00778     /* Please see desc_extension.h for extension_descriptor types */
00779
00780     #endif
```

9.54 lib/include/libdvbv5/dvb-demux.h File Reference

Provides interfaces to deal with DVB demux.

```
#include <linux/dvb/dmx.h>
```

Functions

- int [dvb_dmx_open](#) (int adapter, int demux)
Opens a DVB demux in read/write mode.
- void [dvb_dmx_close](#) (int dmx_fd)
Stops the DMX filter for the file descriptor and closes.
- void [dvb_dmx_stop](#) (int dmx_fd)
Stops the DMX filter for a given file descriptor.
- int [dvb_set_pesfilter](#) (int dmxfd, int pid, dmx_pes_type_t type, dmx_output_t output, int buffersize)
Start a filter for a MPEG-TS Packetized Elementary Stream (PES)
- int [dvb_set_section_filter](#) (int dmxfd, int pid, unsigned filtsize, unsigned char *filter, unsigned char *mask, unsigned char *mode, unsigned int flags)
Sets a MPEG-TS section filter.
- int [dvb_get_pmt_pid](#) (int dmxfd, int sid)
read the contents of the MPEG-TS PAT table, seeking for an specific service ID

9.54.1 Detailed Description

Provides interfaces to deal with DVB demux.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [dvb-demux.h](#).

9.55 dvb-demux.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * These routines were originally written as part of the dvb-apps, as:
00014  *     util functions for various ?zap implementations
00015  *
00016  *     Copyright (C) 2001 Johannes Stezenbach (js@convergence.de)
00017  *     for convergence integrated media
00018  *
00019  *     Originally licensed as GPLv2 or upper
00020  */
00021
00022 #ifndef _DVB_DEMUX_H
00023 #define _DVB_DEMUX_H
00024
00025 #include <linux/dvb/dmx.h>
00026
00027 #ifdef __cplusplus
00028 extern "C" {
00029 #endif
00030
00031 int dvb_dmx_open(int adapter, int demux);
00032
00033 void dvb_dmx_close(int dmx_fd);
00034
00035 void dvb_dmx_stop(int dmx_fd);
00036
00037 int dvb_set_pesfilter(int dmxfd, int pid, dmx_pes_type_t type,
00038                      dmx_output_t output, int buffersize);
00039
00040 int dvb_set_section_filter(int dmxfd, int pid, unsigned filtsize,
00041                            unsigned char *filter,
00042                            unsigned char *mask,
00043                            unsigned char *mode,
00044                            unsigned int flags);
00045
00046 int dvb_get_pmt_pid(int dmxfd, int sid);
00047
00048 #ifdef __cplusplus
00049 }
00050 #endif
00051 #endif

```

9.56 lib/include/libdvbv5/dvb-dev.h File Reference

Provides interfaces to handle Digital TV devices.

```

#include "dvb-fe.h"
#include "dvb-scan.h"
#include <linux/dvb/dmx.h>

```

Data Structures

- struct [dvb_dev_list](#)
Digital TV device node properties.
- struct [dvb_device](#)
Digital TV list of devices.

Typedefs

- typedef int(* [dvb_dev_change_t](#)) (char *sysname, enum [dvb_dev_change_type](#) type, void *priv)
Describes a callback for [dvb_dev_find\(\)](#)

Enumerations

- enum [dvb_dev_type](#) {
[DVB_DEVICE_FRONTEND](#) , [DVB_DEVICE_DEMUX](#) , [DVB_DEVICE_DVR](#) , [DVB_DEVICE_NET](#) ,
[DVB_DEVICE_CA](#) , [DVB_DEVICE_CA_SEC](#) , [DVB_DEVICE_VIDEO](#) , [DVB_DEVICE_AUDIO](#) }
Type of a device entry to search.
- enum [dvb_dev_change_type](#) { [DVB_DEV_ADD](#) , [DVB_DEV_CHANGE](#) , [DVB_DEV_REMOVE](#) }
Describes the type of change to be notifier_delay.

Functions

- struct [dvb_device](#) * [dvb_dev_alloc](#) (void)
Allocate a struct [dvb_device](#).
- void [dvb_dev_free](#) (struct [dvb_device](#) *dvb)
free a struct [dvb_device](#)
- int [dvb_dev_find](#) (struct [dvb_device](#) *dvb, [dvb_dev_change_t](#) handler, void *user_priv)
finds all DVB devices on the local machine
- struct [dvb_dev_list](#) * [dvb_dev_seek_by_adapter](#) (struct [dvb_device](#) *dvb, unsigned int adapter, unsigned int num, enum [dvb_dev_type](#) type)
Find a device that matches the search criteria given by this functions's parameters.
- struct [dvb_dev_list](#) * [dvb_get_dev_info](#) (struct [dvb_device](#) *dvb, const char *sysname)
Return data about a device from its sysname.
- void [dvb_dev_stop_monitor](#) (struct [dvb_device](#) *dvb)
Stop the [dvb_dev_find](#) loop.
- void [dvb_dev_set_logpriv](#) (struct [dvb_device](#) *dvb, unsigned verbose, [dvb_logfunc_priv](#) logfunc, void *logpriv)
Sets the DVB verbosity and log function with context private data.
- void [dvb_dev_set_log](#) (struct [dvb_device](#) *dvb, unsigned verbose, [dvb_logfunc](#) logfunc)
Sets the DVB verbosity and log function.
- struct [dvb_open_descriptor](#) * [dvb_dev_open](#) (struct [dvb_device](#) *dvb, const char *sysname, int flags)
Opens a dvb device.
- void [dvb_dev_close](#) (struct [dvb_open_descriptor](#) *open_dev)
Closes a dvb device.
- int [dvb_dev_get_fd](#) (struct [dvb_open_descriptor](#) *open_dev)
returns fd from a local device This will not work for remote devices.
- ssize_t [dvb_dev_read](#) (struct [dvb_open_descriptor](#) *open_dev, void *buf, size_t count)
read from a dvb demux or dvr file
- void [dvb_dev_dmx_stop](#) (struct [dvb_open_descriptor](#) *open_dev)
Stops the demux filter for a given file descriptor.
- int [dvb_dev_set_bufsize](#) (struct [dvb_open_descriptor](#) *open_dev, int buffersize)
Start a demux or dvr buffer size.
- int [dvb_dev_dmx_set_pesfilter](#) (struct [dvb_open_descriptor](#) *open_dev, int pid, [dmx_pes_type_t](#) type, [dmx_output_t](#) output, int buffersize)
Start a filter for a MPEG-TS Packetized Elementary Stream (PES)
- int [dvb_dev_dmx_set_section_filter](#) (struct [dvb_open_descriptor](#) *open_dev, int pid, unsigned filtsize, unsigned char *filter, unsigned char *mask, unsigned char *mode, unsigned int flags)

Sets a MPEG-TS section filter.

- int [dvb_dev_dmx_get_pmt_pid](#) (struct [dvb_open_descriptor](#) *open_dev, int sid)
read the contents of the MPEG-TS PAT table, seeking for an specific service ID
- struct [dvb_v5_descriptors](#) * [dvb_dev_scan](#) (struct [dvb_open_descriptor](#) *open_dev, struct [dvb_entry](#) *entry, [check_frontend_t](#) *check_frontend, void *args, unsigned other_nit, unsigned timeout_multiply)
Scans a DVB dvb_add_scanned_transponder.
- static int [dvb_dev_remote_init](#) (struct [dvb_device](#) *d, char *server, int port)

9.56.1 Detailed Description

Provides interfaces to handle Digital TV devices.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Digital TV device node file names depend on udev configuration. For example, while frontends are typically found at `/dev/dvb/adapter?/frontend?`, the actual file name can vary from system to system, depending on the udev ruleset.

The libdvbv5 provides a set of functions to allow detecting and getting the device paths in a sane way, via libudev.

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [dvb-dev.h](#).

9.56.2 Typedef Documentation

9.56.2.1 [dvb_dev_change_t](#) typedef int(* [dvb_dev_change_t](#)) (char *sysname, enum [dvb_dev_change_type](#) type, void *priv)

Describes a callback for [dvb_dev_find\(\)](#)

sysname: Kernel's system name for the device (`dvb?.frontend?`, for example) @type: type of change, as defined by enum [dvb_dev_change_type](#)

Note

: the returned string should be freed with `free()`.

Definition at line 121 of file [dvb-dev.h](#).

9.56.3 Enumeration Type Documentation

9.56.3.1 [dvb_dev_change_type](#) enum [dvb_dev_change_type](#)

Describes the type of change to be notified.

Parameters

<i>DVB_DEV_ADD</i>	New device detected
<i>DVB_DEV_CHANGE</i>	Device has changed something
<i>DVB_DEV_REMOVE</i>	A hot-pluggable device was removed

Enumerator

<i>DVB_DEV_ADD</i>	
<i>DVB_DEV_CHANGE</i>	
<i>DVB_DEV_REMOVE</i>	

Definition at line 106 of file [dvb-dev.h](#).

9.56.4 Function Documentation

9.56.4.1 `dvb_dev_remote_init()` `static int dvb_dev_remote_init (`
 `struct dvb_device * d,`
 `char * server,`
 `int port) [inline], [static]`

Examples

[dvb-fe-tool.c](#), and [dvbv5-zap.c](#).

Definition at line 501 of file [dvb-dev.h](#).

9.56.4.2 `dvb_dev_seek_by_adapter()` `struct dvb_dev_list * dvb_dev_seek_by_adapter (`
 `struct dvb_device * dvb,`
 `unsigned int adapter,`
 `unsigned int num,`
 `enum dvb_dev_type type)`

Find a device that matches the search criteria given by this functions's parameters.

Parameters

<i>dvb</i>	pointer to struct dvb_device to be used
<i>adapter</i>	Adapter number, as defined internally at the Kernel. Always start with 0;
<i>num</i>	Digital TV device number (e. g. frontend0, net0, etc);
<i>type</i>	Type of the device, as given by enum dvb_dev_type ;

Returns

returns a pointer to a struct [dvb_dev_list](#) object or NULL if the desired device was not found.

Examples

[dvb-fe-tool.c](#), [dvbv5-scan.c](#), and [dvbv5-zap.c](#).

9.56.4.3 dvb_get_dev_info() struct [dvb_dev_list](#) * dvb_get_dev_info (
 struct [dvb_device](#) * *dvb*,
 const char * *sysname*)

Return data about a device from its sysname.

Parameters

<i>dvb</i>	pointer to struct dvb_device to be used
<i>sysname</i>	Kernel's name of the device to be opened, as obtained via dvb_dev_seek_by_adapter() or via dvb_dev_find() .

Returns

returns a pointer to a struct [dvb_dev_list](#) object or NULL if the desired device was not found.

9.57 dvb-dev.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2016 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  */
00018
00019 #ifndef _DVB_DEV_H
00020 #define _DVB_DEV_H
00021
00022 #include "dvb-fe.h"
00023 #include "dvb-scan.h"
00024
00025 #include <linux/dvb/dmx.h>
00026
00058 enum dvb_dev_type {
00059     DVB_DEVICE_FRONTEND,
00060     DVB_DEVICE_DEMUX,
00061     DVB_DEVICE_DVR,
00062     DVB_DEVICE_NET,
00063     DVB_DEVICE_CA,
00064     DVB_DEVICE_CA_SEC,
00065     DVB_DEVICE_VIDEO,
00066     DVB_DEVICE_AUDIO,
00067 };

```

```

00068
00086 struct dvb_dev_list {
00087     char *syspath;
00088     char *path;
00089     char *sysname;
00090     enum dvb_dev_type dvb_type;
00091     char *bus_addr;
00092     char *bus_id;
00093     char *manufacturer;
00094     char *product;
00095     char *serial;
00096 };
00097
00106 enum dvb_dev_change_type {
00107     DVB_DEV_ADD,
00108     DVB_DEV_CHANGE,
00109     DVB_DEV_REMOVE,
00110 };
00111
00121 typedef int (*dvb_dev_change_t) (char *sysname,
00122                                   enum dvb_dev_change_type type, void *priv);
00123
00129 struct dvb_open_descriptor;
00130
00140 struct dvb_device {
00141     /* Digital TV device lists */
00142     struct dvb_dev_list *devices;
00143     int num_devices;
00144
00145     /* Digital TV frontend access */
00146     struct dvb_v5_fe_parms *fe_parms;
00147 };
00148
00159 struct dvb_device *dvb_dev_alloc(void);
00160
00167 void dvb_dev_free(struct dvb_device *dvb);
00168
00193 int dvb_dev_find(struct dvb_device *dvb, dvb_dev_change_t handler,
00194                 void *user_priv);
00195
00209 struct dvb_dev_list *dvb_dev_seek_by_adapter(struct dvb_device *dvb,
00210                                              unsigned int adapter,
00211                                              unsigned int num,
00212                                              enum dvb_dev_type type);
00213
00224 struct dvb_dev_list *dvb_get_dev_info(struct dvb_device *dvb,
00225                                       const char *sysname);
00226
00237 void dvb_dev_stop_monitor(struct dvb_device *dvb);
00238
00256 void dvb_dev_set_logpriv(struct dvb_device *dvb,
00257                          unsigned verbose,
00258                          dvb_logfunc_priv logfunc, void *logpriv);
00259
00276 void dvb_dev_set_log(struct dvb_device *dvb,
00277                      unsigned verbose,
00278                      dvb_logfunc logfunc);
00279
00303 struct dvb_open_descriptor *dvb_dev_open(struct dvb_device *dvb,
00304                                          const char *sysname, int flags);
00305
00313 void dvb_dev_close(struct dvb_open_descriptor *open_dev);
00314
00325 int dvb_dev_get_fd(struct dvb_open_descriptor *open_dev);
00326
00339 ssize_t dvb_dev_read(struct dvb_open_descriptor *open_dev,
00340                     void *buf, size_t count);
00341
00355 void dvb_dev_dmx_stop(struct dvb_open_descriptor *open_dev);
00356
00373 int dvb_dev_set_bufsize(struct dvb_open_descriptor *open_dev,
00374                         int buffersize);
00375
00399 int dvb_dev_dmx_set_pesfilter(struct dvb_open_descriptor *open_dev,
00400                              int pid, dmxfes_type_t type,
00401                              dmxfes_output_t output, int buffersize);
00402
00425 int dvb_dev_dmx_set_section_filter(struct dvb_open_descriptor *open_dev,
00426                                    int pid, unsigned filter_size,
00427                                    unsigned char *filter,
00428                                    unsigned char *mask,
00429                                    unsigned char *mode,
00430                                    unsigned int flags);
00431
00447 int dvb_dev_dmx_get_pmt_pid(struct dvb_open_descriptor *open_dev, int sid);
00448
00470 struct dvb_v5_descriptors *dvb_dev_scan(struct dvb_open_descriptor *open_dev,

```

```

00471                                     struct dvb_entry *entry,
00472                                     check_frontend_t *check_frontend,
00473                                     void *args,
00474                                     unsigned other_nit,
00475                                     unsigned timeout_multiply);
00476
00477 /* From dvb-dev-remote.c */
00478
00479 #ifdef HAVE_DVBV5_REMOTE
00480
00481 #define REMOTE_BUF_SIZE (87 * 188)      /* 16356 bytes */
00482
00483
00497 int dvb_dev_remote_init(struct dvb_device *d, char *server, int port);
00498
00499 #else
00500
00501 static inline int dvb_dev_remote_init(struct dvb_device *d, char *server,
00502                                     int port)
00503 {
00504     return -1;
00505 };
00506
00507 #endif
00508
00509
00510 #endif

```

9.58 lib/include/libdvbv5/dvb-fe.h File Reference

Provides interfaces to deal with DVB frontend.

```

#include <stdio.h>
#include <errno.h>
#include <stdint.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <string.h>
#include "dvb-frontend.h"
#include "dvb-sat.h"
#include "dvb-log.h"

```

Data Structures

- struct [dvb_v5_fe_parms](#)
Keeps data needed to handle the DVB frontend.

Macros

- #define [ARRAY_SIZE\(x\)](#)
Calculates the number of elements of an array.
- #define [MAX_DELIVERY_SYSTEMS](#)
Max number of delivery systems for a given frontend.

Functions

- struct `dvb_v5_fe_parms` * `dvb_fe_dummy` (void)
Allocates a dummy frontend structure.
- struct `dvb_v5_fe_parms` * `dvb_fe_open_flags` (int adapter, int frontend, unsigned verbose, unsigned use_↵ legacy_call, `dvb_logfunc` logfunc, int flags)
Opens a frontend and allocates a structure to work with.
- struct `dvb_v5_fe_parms` * `dvb_fe_open` (int adapter, int frontend, unsigned verbose, unsigned use_legacy_↵ _call)
Opens a frontend and allocates a structure to work with.
- struct `dvb_v5_fe_parms` * `dvb_fe_open2` (int adapter, int frontend, unsigned verbose, unsigned use_legacy_↵ _call, `dvb_logfunc` logfunc)
Opens a frontend and allocates a structure to work with.
- void `dvb_fe_close` (struct `dvb_v5_fe_parms` *parms)
Closes the frontend and frees allocated resources.
- const char * `dvb_cmd_name` (int cmd)
Returns the string name associated with a DVBv5 command.
- const char *const * `dvb_attr_names` (int cmd)
Returns an string array with the valid string values associated with a DVBv5 command.
- int `dvb_fe_retrieve_parm` (const struct `dvb_v5_fe_parms` *parms, unsigned cmd, uint32_t *value)
Retrieves the value of a DVBv5/libdvbv5 property.
- int `dvb_fe_store_parm` (struct `dvb_v5_fe_parms` *parms, unsigned cmd, uint32_t value)
Stores the value of a DVBv5/libdvbv5 property.
- int `dvb_set_sys` (struct `dvb_v5_fe_parms` *parms, fe_delivery_system_t sys)
Sets the delivery system.
- int `dvb_add_parms_for_sys` (struct `dvb_v5_fe_parms` *parms, fe_delivery_system_t sys)
Make dvb properties reflect the current standard.
- int `dvb_set_compat_delivery_system` (struct `dvb_v5_fe_parms` *parms, uint32_t desired_system)
Sets the delivery system.
- void `dvb_fe_prt_parms` (const struct `dvb_v5_fe_parms` *parms)
Prints all the properties at the cache.
- int `dvb_fe_set_parms` (struct `dvb_v5_fe_parms` *parms)
Prints all the properties at the cache.
- int `dvb_fe_get_parms` (struct `dvb_v5_fe_parms` *parms)
Prints all the properties at the cache.
- struct dtv_stats * `dvb_fe_retrieve_stats_layer` (struct `dvb_v5_fe_parms` *parms, unsigned cmd, unsigned layer)
Retrieve the stats for a DTV layer from cache.
- int `dvb_fe_retrieve_stats` (struct `dvb_v5_fe_parms` *parms, unsigned cmd, uint32_t *value)
Retrieve the stats for a DTV layer from cache.
- int `dvb_fe_get_stats` (struct `dvb_v5_fe_parms` *parms)
Retrieve the stats from the Kernel.
- float `dvb_fe_retrieve_ber` (struct `dvb_v5_fe_parms` *parms, unsigned layer, enum fecap_scale_params *scale)
Retrieve the BER stats from cache.
- float `dvb_fe_retrieve_per` (struct `dvb_v5_fe_parms` *parms, unsigned layer)
Retrieve the PER stats from cache.
- enum `dvb_quality` `dvb_fe_retrieve_quality` (struct `dvb_v5_fe_parms` *parms, unsigned layer)
Retrieve the quality stats from cache.
- int `dvb_fe_snprintf_eng` (char *buf, int len, float val)
Ancillary function to sprintf on ENG format.

- `int dvb_fe_sprintf_stat` (struct `dvb_v5_fe_parms` *parms, uint32_t cmd, char *display_name, int layer, char **buf, int *len, int *show_layer_name)
Ancillary function to sprintf on ENG format.
- `int dvb_fe_get_event` (struct `dvb_v5_fe_parms` *parms)
Get both status statistics and dvb parameters.
- `int dvb_fe_sec_voltage` (struct `dvb_v5_fe_parms` *parms, int on, int v18)
DVB ioctl wrapper for setting SEC voltage.
- `int dvb_fe_sec_tone` (struct `dvb_v5_fe_parms` *parms, fe_sec_tone_mode_t tone)
DVB ioctl wrapper for setting SEC tone.
- `int dvb_fe_lnb_high_voltage` (struct `dvb_v5_fe_parms` *parms, int on)
DVB ioctl wrapper for setting LNBf high voltage.
- `int dvb_fe_diseqc_burst` (struct `dvb_v5_fe_parms` *parms, int mini_b)
DVB ioctl wrapper for setting SEC DiSeqC tone burst to select between satellite A or B.
- `int dvb_fe_diseqc_cmd` (struct `dvb_v5_fe_parms` *parms, const unsigned len, const unsigned char *buf)
DVB ioctl wrapper for setting SEC DiSeqC command.
- `int dvb_fe_diseqc_reply` (struct `dvb_v5_fe_parms` *parms, unsigned *len, char *buf, int timeout)
DVB ioctl wrapper for getting SEC DiSeqC reply.
- `int dvb_fe_is_satellite` (uint32_t delivery_system)
DVB Ancillary routine to check if a given Delivery system is satellite.
- `int dvb_fe_set_default_country` (struct `dvb_v5_fe_parms` *parms, const char *country)
Set default country variant of delivery systems like ISDB-T.

9.58.1 Detailed Description

Provides interfaces to deal with DVB frontend.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

The libdvbv5 API works with a set of key/value properties. There are two types of properties:

- The ones defined at the Kernel's frontend API, that are found at `/usr/include/linux/dvb/frontend.h` (actually, it uses a local copy of that file, stored at `./include/linux/dvb/frontend.h`)
- Some extra properties used by libdvbv5. Those can be found at `lib/include/libdvbv5/dvb-v5-std.h` and start at `DTV_USER_COMMAND_START`.

Just like the DTV properties, the stats are cached. That warrants that all stats are got at the same time, when `dvb_fe_get_stats()` is called.

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file `dvb-fe.h`.

9.59 dvb-fe.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019 #ifndef _DVB_FE_H
00020 #define _DVB_FE_H
00021
00022 #include <stdio.h>
00023 #include <errno.h>
00024 #include <stdint.h>
00025 #include <stdlib.h>
00026 #include <sys/types.h>
00027 #include <sys/stat.h>
00028 #include <fcntl.h>
00029 #include <sys/ioctl.h>
00030 #include <string.h>
00031 #include "dvb-frontend.h"
00032 #include "dvb-sat.h"
00033 #include "dvb-log.h"
00034
00064 #define ARRAY_SIZE(x) (sizeof(x)/sizeof((x)[0]))
00065
00071 #define MAX_DELIVERY_SYSTEMS 20
00072
00073 #ifndef _DOXYGEN
00074 /*
00075  * There are a few aliases for other properties. Those are needed just
00076  * to avoid breaking apps that depend on the library but shoudn't be used
00077  * anymore on newer apps.
00078  */
00079 #define DTV_MAX_STATS DTV_NUM_STATS_PROPS
00080 #define DTV_SIGNAL_STRENGTH DTV_STAT_SIGNAL_STRENGTH
00081 #define DTV_SNR DTV_STAT_CNR
00082 #define DTV_UNCORRECTED_BLOCKS DTV_STAT_ERROR_BLOCK_COUNT
00083
00084 #endif
00085
00117 struct dvb_v5_fe_parms {
00118     /* Information visible to the client - don't override those values */
00119     struct dvb_frontend_info info;
00120     uint32_t version;
00121     int has_v5_stats;
00122     fe_delivery_system_t current_sys;
00123     int num_systems;
00124     fe_delivery_system_t systems[MAX_DELIVERY_SYSTEMS];
00125     int legacy_fe;
00126
00127     /* The values below are specified by the library client */
00128
00129     /* Flags from the client to the library */
00130     int abort;
00131
00132     /* Linear Amplifier settings */
00133     int lna;
00134
00135     /* Satellite settings */
00136     const struct dvb_sat_lnb *lnb;
00137     int sat_number;
00138     unsigned freq_bpf;
00139     unsigned diseqc_wait;
00140
00141     /* Function to write DVB logs */
00142     unsigned verbose;
00143     dvb_logfunc logfunc;
00144
00145     /* Charsets to be used by the conversion utilities */
00146     char *default_charset;
00147     char *output_charset;
00148 };

```

```
00149
00150 #ifdef __cplusplus
00151 extern "C" {
00152 #endif
00153
00163 struct dvb_v5_fe_parms *dvb_fe_dummy(void);
00164
00194 struct dvb_v5_fe_parms *dvb_fe_open_flags(int adapter, int frontend,
00195                                           unsigned verbose,
00196                                           unsigned use_legacy_call,
00197                                           dvb_logfunc logfunc,
00198                                           int flags);
00199
00217 struct dvb_v5_fe_parms *dvb_fe_open(int adapter, int frontend,
00218                                     unsigned verbose,
00219                                     unsigned use_legacy_call);
00220
00241 struct dvb_v5_fe_parms *dvb_fe_open2(int adapter, int frontend,
00242                                       unsigned verbose, unsigned use_legacy_call,
00243                                       dvb_logfunc logfunc);
00244
00251 void dvb_fe_close(struct dvb_v5_fe_parms *parms);
00252
00267 const char *dvb_cmd_name(int cmd);
00268
00283 const char *const *dvb_attr_names(int cmd);
00284
00285 /* Get/set delivery system parameters */
00286
00300 int dvb_fe_retrieve_parm(const struct dvb_v5_fe_parms *parms,
00301                         unsigned cmd, uint32_t *value);
00302
00316 int dvb_fe_store_parm(struct dvb_v5_fe_parms *parms,
00317                      unsigned cmd, uint32_t value);
00318
00334 int dvb_set_sys(struct dvb_v5_fe_parms *parms,
00335                fe_delivery_system_t sys);
00336
00351 int dvb_add_parms_for_sys(struct dvb_v5_fe_parms *parms,
00352                          fe_delivery_system_t sys);
00353
00373 int dvb_set_compat_delivery_system(struct dvb_v5_fe_parms *parms,
00374                                   uint32_t desired_system);
00375
00384 void dvb_fe_prt_parms(const struct dvb_v5_fe_parms *parms);
00385
00398 int dvb_fe_set_parms(struct dvb_v5_fe_parms *parms);
00399
00411 int dvb_fe_get_parms(struct dvb_v5_fe_parms *parms);
00412
00413 /*
00414  * statistics functions
00415  */
00416
00440 struct dtv_stats *dvb_fe_retrieve_stats_layer(struct dvb_v5_fe_parms *parms,
00441                                              unsigned cmd, unsigned layer);
00442
00457 int dvb_fe_retrieve_stats(struct dvb_v5_fe_parms *parms,
00458                          unsigned cmd, uint32_t *value);
00459
00470 int dvb_fe_get_stats(struct dvb_v5_fe_parms *parms);
00471
00497 float dvb_fe_retrieve_ber(struct dvb_v5_fe_parms *parms, unsigned layer,
00498                          enum fecap_scale_params *scale);
00499
00522 float dvb_fe_retrieve_per(struct dvb_v5_fe_parms *parms, unsigned layer);
00523
00524
00548 enum dvb_quality dvb_fe_retrieve_quality(struct dvb_v5_fe_parms *parms,
00549                                         unsigned layer);
00550
00565 int dvb_fe_snprintf_eng(char *buf, int len, float val);
00566
00567
00589 int dvb_fe_snprintf_stat(struct dvb_v5_fe_parms *parms, uint32_t cmd,
00590                        char *display_name, int layer,
00591                        char **buf, int *len, int *show_layer_name);
00592
00603 int dvb_fe_get_event(struct dvb_v5_fe_parms *parms);
00604
00605 /*
00606  * Other functions, associated to SEC/LNB/DISEqC
00607  *
00608  * The functions below are just wrappers for the Kernel calls, in order to
00609  * manually control satellite systems.
00610  *
00611  * Instead of using most them, the best is to set the LNBf parameters, and let
```

```

00612 * the libdvbv5 to automatically handle the calls.
00613 *
00614 * NOTE: It currently lacks support for two ioctl's:
00615 * FE_DISEQC_RESET_OVERLOAD      used only on av7110.
00616 * Spec says:
00617 *   If the bus has been automatically powered off due to power overload,
00618 *   this ioctl call restores the power to the bus. The call requires read/write
00619 *   access to the device. This call has no effect if the device is manually
00620 *   powered off. Not all DVB adapters support this ioctl.
00621 *
00622 * FE_DISHNETWORK_SEND_LEGACY_CMD is used on av7110, budget, gp8psk and stv0299
00623 * Spec says:
00624 *   WARNING: This is a very obscure legacy command, used only at stv0299
00625 *   driver. Should not be used on newer drivers.
00626 *   It provides a non-standard method for selecting Diseqc voltage on the
00627 *   frontend, for Dish Network legacy switches.
00628 *   As support for this ioctl were added in 2004, this means that such dishes
00629 *   were already legacy in 2004.
00630 *
00631 * So, it doesn't make much sense on implementing support for them.
00632 */
00633
00646 int dvb_fe_sec_voltage(struct dvb_v5_fe_parms *parms, int on, int v18);
00647
00657 int dvb_fe_sec_tone(struct dvb_v5_fe_parms *parms, fe_sec_tone_mode_t tone);
00658
00668 int dvb_fe_lnb_high_voltage(struct dvb_v5_fe_parms *parms, int on);
00669
00684 int dvb_fe_diseqc_burst(struct dvb_v5_fe_parms *parms, int mini_b);
00685
00696 int dvb_fe_diseqc_cmd(struct dvb_v5_fe_parms *parms, const unsigned len,
00697                     const unsigned char *buf);
00698
00710 int dvb_fe_diseqc_reply(struct dvb_v5_fe_parms *parms, unsigned *len, char *buf,
00711                       int timeout);
00712
00719 int dvb_fe_is_satellite(uint32_t delivery_system);
00720
00734 int dvb_fe_set_default_country(struct dvb_v5_fe_parms *parms,
00735                               const char *country);
00736
00737 #ifdef __cplusplus
00738 }
00739 #endif
00740
00741 /*
00742 * Arrays from dvb-v5.h
00743 *
00744 * Those arrays can be used to translate from a DVB property into a name.
00745 *
00746 * No need to directly access them from userspace, as dvb_attr_names()
00747 * already handles them into a more standard way.
00748 */
00749
00750 #ifndef _DOXYGEN
00751
00752 extern const unsigned fe_bandwidth_name[8];
00753 extern const char *dvb_v5_name[72];
00754 extern const void *dvb_v5_attr_names[];
00755 extern const char *delivery_system_name[21];
00756 extern const char *fe_code_rate_name[30];
00757 extern const char *fe_modulation_name[22];
00758 extern const char *fe_transmission_mode_name[10];
00759 extern const unsigned fe_bandwidth_name[8];
00760 extern const char *fe_guard_interval_name[13];
00761 extern const char *fe_hierarchy_name[6];
00762 extern const char *fe_voltage_name[4];
00763 extern const char *fe_tone_name[3];
00764 extern const char *fe_inversion_name[4];
00765 extern const char *fe_pilot_name[4];
00766 extern const char *fe_rolloff_name[8];
00767
00768 #endif
00769
00770 #endif

```

9.60 lib/include/libdvbv5/dvb-file.h File Reference

Provides interfaces to deal with DVB channel and program files.

```
#include "dvb-fe.h"
```

Data Structures

- struct [dvb_elementary_pid](#)
associates an elementary stream type with its PID
- struct [dvb_entry](#)
Represents one entry on a DTV file.
- struct [dvb_file](#)
Describes an entire DVB file opened.
- struct [dvb_parse_table](#)
Describes the fields to parse on a file.
- struct [dvb_parse_struct](#)
Describes the format to parse an specific delivery system.
- struct [dvb_parse_file](#)
Describes an entire file format.

Enumerations

- enum [dvb_file_formats](#) {
 [FILE_UNKNOWN](#) , [FILE_ZAP](#) , [FILE_CHANNEL](#) , [FILE_DVBV5](#) ,
 [FILE_VDR](#) }
Known file formats.

Functions

- static void [dvb_file_free](#) (struct [dvb_file](#) *[dvb_file](#))
Deallocates memory associated with a struct [dvb_file](#).
- struct [dvb_file](#) * [dvb_read_file](#) (const char *[fname](#))
Read a file at libdvbv5 format.
- int [dvb_write_file](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#))
Write a file at libdvbv5 format.
- struct [dvb_file](#) * [dvb_read_file_format](#) (const char *[fname](#), uint32_t [delsys](#), enum [dvb_file_formats](#) [format](#))
Read a file on any format natively supported by the library.
- int [dvb_write_file_format](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#), uint32_t [delsys](#), enum [dvb_file_formats](#) [format](#))
Write a file on any format natively supported by the library.
- int [dvb_store_entry_prop](#) (struct [dvb_entry](#) *[entry](#), uint32_t [cmd](#), uint32_t [value](#))
Stores a key/value pair on a DVB file entry.
- int [dvb_retrieve_entry_prop](#) (struct [dvb_entry](#) *[entry](#), uint32_t [cmd](#), uint32_t *[value](#))
Retrieves the value associated with a key on a DVB file entry.
- int [dvb_store_channel](#) (struct [dvb_file](#) **[dvb_file](#), struct [dvb_v5_fe_parms](#) *[parms](#), struct [dvb_v5_descriptors](#) *[dvb_desc](#), int [get_detected](#), int [get_nit](#))
stored a new scanned channel into a [dvb_file](#) struct
- int [dvb_parse_delsys](#) (const char *[name](#))
Ancillary function that seeks for a delivery system.
- enum [dvb_file_formats](#) [dvb_parse_format](#) (const char *[name](#))
Ancillary function that parses the name of a file format.
- struct [dvb_file](#) * [dvb_parse_format_online](#) (const char *[fname](#), uint32_t [delsys](#), const struct [dvb_parse_file](#) *[parse_file](#))
Read and parses a one line file format.
- int [dvb_write_format_online](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#), uint32_t [delsys](#), const struct [dvb_parse_file](#) *[parse_file](#))
Writes a file into an one line file format.
- int [dvb_write_format_vdr](#) (const char *[fname](#), struct [dvb_file](#) *[dvb_file](#))
Writes a file into vdr format (compatible up to version 2.1)

Variables

- const struct [dvb_parse_file_channel_file_format](#)
File format definitions for dvb-apps channel format.
- const struct [dvb_parse_file_channel_file_zap_format](#)
File format definitions for dvb-apps zap format.

9.60.1 Detailed Description

Provides interfaces to deal with DVB channel and program files.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

There are basically two types of files used for DVB:

- files that describe the physical channels (also called as transponders);
- files that describe the several programs found on a MPEG-TS (also called as zap files).

The libdvbv5 library defines an unified type for both types. Other applications generally use different formats.

The purpose of the functions and structures defined herein is to provide support to read and write to those different formats.

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [dvb-file.h](#).

9.61 dvb-file.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  */
00016 #ifndef _DVB_FILE_H
00017 #define _DVB_FILE_H
00018
```

```

00019 #include "dvb-fe.h"
00020
00043 /*
00044  * DVB structures used to represent all files opened by the libdvbv5 library.
00045  *
00046  * Those structs represents each individual entry on a file, and the file
00047  * as a whole.
00048  */
00049
00058 struct dvb_elementary_pid {
00059     uint8_t type;
00060     uint16_t pid;
00061 };
00062
00104 struct dvb_entry {
00105     struct dtv_property props[DTV_MAX_COMMAND];
00106     unsigned int n_props;
00107     struct dvb_entry *next;
00108     uint16_t service_id;
00109     uint16_t *video_pid, *audio_pid;
00110     struct dvb_elementary_pid *other_el_pid;
00111     unsigned video_pid_len, audio_pid_len, other_el_pid_len;
00112     char *channel;
00113     char *vchannel;
00114
00115     char *location;
00116
00117     int sat_number;
00118     unsigned freq_bpf;
00119     unsigned diseqc_wait;
00120     char *lnb;
00121
00122     uint16_t network_id;
00123     uint16_t transport_id;
00124 };
00125
00126 struct dvb_file {
00136     char *fname;
00137     int n_entries;
00138     struct dvb_entry *first_entry;
00139 };
00140
00141 /*
00142  * DVB file format tables
00143  *
00144  * The structs below are used to represent oneline formats like the ones
00145  * commonly found on DVB legacy applications.
00146  */
00147
00165 struct dvb_parse_table {
00166     unsigned int prop;
00167     const char **table;
00168     unsigned int size;
00169     int mult_factor;
00170     int has_default_value;
00171     int default_value;
00172 };
00185 struct dvb_parse_struct {
00186     char *id;
00187     uint32_t delsys;
00188     const struct dvb_parse_table *table;
00189     unsigned int size;
00190 };
00191
00203 struct dvb_parse_file {
00204     int has_delsys_id;
00205     char *delimiter;
00206     struct dvb_parse_struct formats[];
00207 };
00208
00232 enum dvb_file_formats {
00233     FILE_UNKNOWN,
00234     FILE_ZAP,
00235     FILE_CHANNEL,
00236     FILE_DVBV5,
00237     FILE_VDR,
00238 };
00239
00240 struct dvb_v5_descriptors;
00241
00242 #ifdef __cplusplus
00243 extern "C" {
00244 #endif
00245
00255 static inline void dvb_file_free(struct dvb_file *dvb_file)
00256 {

```

```

00257     struct dvb_entry *entry = dvb_file->first_entry, *next;
00258     while (entry) {
00259         next = entry->next;
00260         if (entry->channel)
00261             free(entry->channel);
00262         if (entry->vchannel)
00263             free(entry->vchannel);
00264         if (entry->location)
00265             free(entry->location);
00266         if (entry->video_pid)
00267             free(entry->video_pid);
00268         if (entry->audio_pid)
00269             free(entry->audio_pid);
00270         if (entry->other_el_pid)
00271             free(entry->other_el_pid);
00272         if (entry->lmb)
00273             free(entry->lmb);
00274         free(entry);
00275         entry = next;
00276     }
00277     free(dvb_file);
00278 }
00279
00280 /*
00281  * File format description structures defined for the several formats that
00282  * the library can read natively.
00283  */
00284
00285 extern const struct dvb_parse_file channel_file_format;
00286
00287 extern const struct dvb_parse_file channel_file_zap_format;
00288
00289 /*
00290  * Prototypes for the several functions defined at dvb-file.c
00291  */
00292
00293 struct dvb_file *dvb_read_file(const char *fname);
00294
00295 int dvb_write_file(const char *fname, struct dvb_file *dvb_file);
00296
00297 struct dvb_file *dvb_read_file_format(const char *fname,
00298                                       uint32_t delsys,
00299                                       enum dvb_file_formats format);
00300
00301 int dvb_write_file_format(const char *fname,
00302                          struct dvb_file *dvb_file,
00303                          uint32_t delsys,
00304                          enum dvb_file_formats format);
00305
00306 int dvb_store_entry_prop(struct dvb_entry *entry,
00307                         uint32_t cmd, uint32_t value);
00308
00309 int dvb_retrieve_entry_prop(struct dvb_entry *entry,
00310                            uint32_t cmd, uint32_t *value);
00311
00312 int dvb_store_channel(struct dvb_file **dvb_file,
00313                      struct dvb_v5_fe_parms *parms,
00314                      struct dvb_v5_descriptors *dvb_desc,
00315                      int get_detected, int get_nit);
00316
00317 int dvb_parse_delsys(const char *name);
00318
00319 enum dvb_file_formats dvb_parse_format(const char *name);
00320
00321 /*
00322  * Routines to read a non-libdvbv5 format. They're called by
00323  * dvb_read_file_format() or dvb_write_file_format()
00324  */
00325
00326 struct dvb_file *dvb_parse_format_online(const char *fname,
00327                                          uint32_t delsys,
00328                                          const struct dvb_parse_file *parse_file);
00329
00330 int dvb_write_format_online(const char *fname,
00331                           struct dvb_file *dvb_file,
00332                           uint32_t delsys,
00333                           const struct dvb_parse_file *parse_file);
00334
00335 int dvb_write_format_vdr(const char *fname,
00336                        struct dvb_file *dvb_file);
00337
00338 #ifdef __cplusplus
00339 }
00340 #endif
00341
00342 #endif // _DVB_FILE_H

```

9.62 lib/include/libdvbv5/dvb-log.h File Reference

Provides interfaces to handle libdvbv5 log messages.

```
#include <syslog.h>
```

Typedefs

- typedef void(* [dvb_logfunc](#)) (int level, const char *fmt,...)
typedef used by dvb_fe_open2 for the log function
- typedef void(* [dvb_logfunc_priv](#)) (void *logpriv, int level, const char *fmt,...)

Functions

- void [dvb_default_log](#) (int level, const char *fmt,...)
This is the prototype of the internal log function that it is used, if the library client doesn't desire to override with something else.

9.62.1 Detailed Description

Provides interfaces to handle libdvbv5 log messages.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Andre Roth

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [dvb-log.h](#).

9.62.2 Typedef Documentation

9.62.2.1 dvb_logfunc_priv typedef void(* dvb_logfunc_priv) (void *logpriv, int level, const char *fmt,...)

Definition at line 52 of file [dvb-log.h](#).

9.63 dvb-log.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00021 #ifndef _LOG_H
00022 #define _LOG_H
00023
00024 #include <syslog.h>
00025
00044 typedef void (*dvb_logfunc)(int level, const char *fmt, ...) __attribute__((format(printf, 2, 3)));
00045
00052 typedef void (*dvb_logfunc_priv)(void *logpriv, int level, const char *fmt, ...);
00053
00054 /*
00055  * Macros used internally inside libdvbv5 frontend part, to output logs
00056  */
00057
00058 #ifndef _DOXYGEN
00059
00060 struct dvb_v5_fe_parms;
00064 dvb_logfunc_priv dvb_get_log_priv(struct dvb_v5_fe_parms *, void **);
00065
00066 #ifndef __DVB_FE_PRIV_H
00067
00068 #define dvb_loglevel(level, fmt, arg...) do {\
00069     void *priv;\
00070     dvb_logfunc_priv f = dvb_get_log_priv(parms, &priv);\
00071     if (f) {\
00072         f(priv, level, fmt, ##arg);\
00073     } else {\
00074         parms->logfunc(level, fmt, ##arg); \
00075     }\
00076 } while (0)
00077
00078 #else
00079
00080 #define dvb_loglevel(level, fmt, arg...) do {\
00081     if (parms->logfunc_priv) {\
00082         parms->logfunc_priv(parms->logpriv, level, fmt, ##arg);\
00083     } else {\
00084         parms->p.logfunc(level, fmt, ##arg); \
00085     }\
00086 } while (0)
00087
00088 #endif
00089
00090 #define dvb_log(fmt, arg...) dvb_loglevel(LOG_INFO, fmt, ##arg)
00091 #define dvb_logerr(fmt, arg...) dvb_loglevel(LOG_ERR, fmt, ##arg)
00092 #define dvb_logdbg(fmt, arg...) dvb_loglevel(LOG_DEBUG, fmt, ##arg)
00093 #define dvb_logwarn(fmt, arg...) dvb_loglevel(LOG_WARNING, fmt, ##arg)
00094 #define dvb_loginfo(fmt, arg...) dvb_loglevel(LOG_NOTICE, fmt, ##arg)
00095
00096 #define dvb_perror(msg) dvb_logerr("%s: %s", msg, strerror(errno))
00097
00098 #endif /* _DOXYGEN */
00099
00108 void dvb_default_log(int level, const char *fmt, ...) __attribute__((format(printf, 2, 3)));
00109
00110 #endif

```

9.64 lib/include/libdvbv5/dvb-sat.h File Reference

Provides interfaces to deal with DVB Satellite systems.

```
#include "dvb-v5-std.h"
```

Data Structures

- struct [dvb_sat_lnb](#)
Stores the information of a LNBf.
- struct [dvb_sat_lnb::dvbsat_freqrange](#)

Functions

- int [dvb_sat_search_lnb](#) (const char *name)
search for a LNBf entry
- int [dvb_print_lnb](#) (int index)
prints the contents of a LNBf entry at STDOUT.
- void [dvb_print_all_lnb](#) (void)
Prints all LNBf entries at STDOUT.
- const struct [dvb_sat_lnb](#) * [dvb_sat_get_lnb](#) (int index)
gets a LNBf entry at its internal database
- const char * [dvb_sat_get_lnb_name](#) (int index)
gets a LNBf entry at its internal database
- int [dvb_sat_set_parms](#) (struct [dvb_v5_fe_parms](#) *parms)
sets the satellite parameters
- int [dvb_sat_real_freq](#) (struct [dvb_v5_fe_parms](#) *p, int freq)
return the real satellite frequency

9.64.1 Detailed Description

Provides interfaces to deal with DVB Satellite systems.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [dvb-sat.h](#).

9.65 dvb-sat.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  */
00018 #ifndef _LIBSAT_H
00019 #define _LIBSAT_H
00020
00021 #include "dvb-v5-std.h"
00022
00023 /*
00024  * Satellite handling functions
00025  */
00026
00027 struct dvb_sat_lnb {
00028     const char *name;
00029     const char *alias;
00030
00031     /*
00032      * Legacy fields, kept just to avoid ABI breakages
00033      * Should not be used by new applications
00034      */
00035     unsigned lowfreq, highfreq;
00036     unsigned rangeswitch;
00037     struct dvbsat_freqrange {
00038         unsigned low, high;
00039     } freqrange[2];
00040 };
00041
00042 struct dvb_v5_fe_parms;
00043
00044 #ifdef __cplusplus
00045 extern "C" {
00046 #endif
00047
00048 /* From libsat.c */
00049
00050 int dvb_sat_search_lnb(const char *name);
00051
00052 int dvb_print_lnb(int index);
00053
00054 void dvb_print_all_lnb(void);
00055
00056 const struct dvb_sat_lnb *dvb_sat_get_lnb(int index);
00057
00058 const char *dvb_sat_get_lnb_name(int index);
00059
00060 int dvb_sat_set_parms(struct dvb_v5_fe_parms *parms);
00061
00062 int dvb_sat_real_freq(struct dvb_v5_fe_parms *p, int freq);
00063
00064 #ifdef __cplusplus
00065 }
00066 #endif
00067 #endif // _LIBSAT_H

```

9.66 lib/include/libdvbv5/dvb-scan.h File Reference

Provides interfaces to scan programs inside MPEG-TS digital TV streams.

```

#include <stdint.h>
#include <linux/dvb/dmx.h>

```

```
#include <libdvbv5/descriptors.h>
#include <libdvbv5/dvb-sat.h>
```

Data Structures

- struct [dvb_v5_descriptors_program](#)
Associates PMT with PAT tables.
- struct [dvb_v5_descriptors](#)
Contains the descriptors needed to scan the Service ID and other relevant info at a MPEG-TS Digital TV stream.
- struct [dvb_table_filter](#)
Describes the PES filters used by DVB scan.

Macros

- #define [MAX_TABLE_SIZE](#)

Typedefs

- typedef int() [check_frontend_t](#)(void *args, struct [dvb_v5_fe_parms](#) *parms)
Callback for the application to show the frontend status.

Functions

- void [dvb_table_filter_free](#) (struct [dvb_table_filter](#) *sect)
deallocates all data associated with a table filter
- int [dvb_read_section](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, unsigned char tid, uint16_t pid, void **table, unsigned timeout)
read MPEG-TS tables that comes from a DTV card
- int [dvb_read_section_with_id](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, unsigned char tid, uint16_t pid, int ts_id, void **table, unsigned timeout)
read MPEG-TS tables that comes from a DTV card with an specific table section ID
- int [dvb_read_sections](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, struct [dvb_table_filter](#) *sect, unsigned timeout)
read MPEG-TS tables that comes from a DTV card
- struct [dvb_v5_descriptors](#) * [dvb_scan_alloc_handler_table](#) (uint32_t delivery_system)
allocates a struct [dvb_v5_descriptors](#)
- void [dvb_scan_free_handler_table](#) (struct [dvb_v5_descriptors](#) *dvb_scan_handler)
frees a struct [dvb_v5_descriptors](#)
- struct [dvb_v5_descriptors](#) * [dvb_get_ts_tables](#) (struct [dvb_v5_fe_parms](#) *parms, int dmxfid, uint32_t delivery_system, unsigned other_nit, unsigned timeout_multiply)
Scans a DVB stream, looking for the tables needed to identify the programs inside a MPEG-TS.
- void [dvb_free_ts_tables](#) (struct [dvb_v5_descriptors](#) *dvb_desc)
frees a struct [dvb_v5_descriptors](#)
- struct [dvb_v5_descriptors](#) * [dvb_scan_transponder](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_entry](#) *entry, int dmxfid, [check_frontend_t](#) *check_frontend, void *args, unsigned other_nit, unsigned timeout_multiply)
Scans a DVB dvb_add_scanned_transponder.
- void [dvb_add_scanned_transponders](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_v5_descriptors](#) *dvb_scan_handler, struct [dvb_entry](#) *first_entry, struct [dvb_entry](#) *entry)
Add new transponders to a [dvb_file](#).

9.66.1 Detailed Description

Provides interfaces to scan programs inside MPEG-TS digital TV streams.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [dvb-scan.h](#).

9.66.2 Macro Definition Documentation

9.66.2.1 MAX_TABLE_SIZE `#define MAX_TABLE_SIZE`

Definition at line 39 of file [dvb-scan.h](#).

9.67 dvb-scan.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  */
00018 #ifndef __LIBSCAN_H
00019 #define __LIBSCAN_H
00020
00021 #include <stdint.h>
00022 #include <linux/dvb/dmx.h>
00023 #include <libdvbv5/descriptors.h>
00024 #include <libdvbv5/dvb-sat.h>
00025
00037 /* According with ISO/IEC 13818-1:2007 */
00038
00039 #define MAX_TABLE_SIZE 1024 * 1024
00040
00041 #ifdef __cplusplus
00042 extern "C" {
00043 #endif
```

```

00044
00045 struct dvb_entry;
00046
00055 struct dvb_v5_descriptors_program {
00056     struct dvb_table_pat_program *pat_pgm;
00057     struct dvb_table_pmt *pmt;
00058 };
00059
00087 struct dvb_v5_descriptors {
00088     uint32_t delivery_system;
00089
00090     struct dvb_entry *entry;
00091     unsigned num_entry;
00092
00093     struct dvb_table_pat *pat;
00094     struct atsc_table_vct *vct;
00095     struct dvb_v5_descriptors_program *program;
00096     struct dvb_table_nit *nit;
00097     struct dvb_table_sdt *sdt;
00098
00099     unsigned num_program;
00100
00101     struct dvb_table_nit **other_nits;
00102     unsigned num_other_nits;
00103
00104     struct dvb_table_sdt **other_sdt;
00105     unsigned num_other_sdt;
00106 };
00107
00122 struct dvb_table_filter {
00123     /* Input data */
00124     unsigned char tid;
00125     uint16_t pid;
00126     int ts_id;
00127     void **table;
00128
00129     int allow_section_gaps;
00130
00131     /*
00132      * Private temp data used by dvb_read_sections().
00133      * Should not be filled outside dvb-scan.c, as they'll be
00134      * overridden
00135      */
00136     void *priv;
00137 };
00144 void dvb_table_filter_free(struct dvb_table_filter *sect);
00145
00186 int dvb_read_section(struct dvb_v5_fe_parms *parms, int dmxf_fd,
00187     unsigned char tid, uint16_t pid, void **table,
00188     unsigned timeout);
00189
00208 int dvb_read_section_with_id(struct dvb_v5_fe_parms *parms, int dmxf_fd,
00209     unsigned char tid, uint16_t pid, int ts_id,
00210     void **table, unsigned timeout);
00211
00225 int dvb_read_sections(struct dvb_v5_fe_parms *parms, int dmxf_fd,
00226     struct dvb_table_filter *sect,
00227     unsigned timeout);
00228
00237 struct dvb_v5_descriptors *dvb_scan_alloc_handler_table(uint32_t delivery_system);
00238
00245 void dvb_scan_free_handler_table(struct dvb_v5_descriptors *dvb_scan_handler);
00246
00268 struct dvb_v5_descriptors *dvb_get_ts_tables(struct dvb_v5_fe_parms *parms, int dmxf_fd,
00269     uint32_t delivery_system,
00270     unsigned other_nit,
00271     unsigned timeout_multiply);
00272
00282 void dvb_free_ts_tables(struct dvb_v5_descriptors *dvb_desc);
00283
00293 typedef int (check_frontend_t)(void *args, struct dvb_v5_fe_parms *parms);
00294
00340 struct dvb_v5_descriptors *dvb_scan_transponder(struct dvb_v5_fe_parms *parms,
00341     struct dvb_entry *entry,
00342     int dmxf_fd,
00343     check_frontend_t *check_frontend,
00344     void *args,
00345     unsigned other_nit,
00346     unsigned timeout_multiply);
00347
00348
00391 void dvb_add_scanned_transponders(struct dvb_v5_fe_parms *parms,
00392     struct dvb_v5_descriptors *dvb_scan_handler,
00393     struct dvb_entry *first_entry,
00394     struct dvb_entry *entry);
00395
00396 #ifndef _DOXYGEN

```

```

00397 /*
00398  * Some ancillary functions used internally inside the library, used to
00399  * identify duplicated transport streams and add new found transponder entries
00400  */
00401 int dvb_estimate_freq_shift(struct dvb_v5_fe_parms *parms);
00402
00403 int dvb_new_freq_is_needed(struct dvb_entry *entry, struct dvb_entry *last_entry,
00404                          uint32_t freq, enum dvb_sat_polarization pol, int shift);
00405
00406 struct dvb_entry *dvb_scan_add_entry(struct dvb_v5_fe_parms *parms,
00407                                     struct dvb_entry *first_entry,
00408                                     struct dvb_entry *entry,
00409                                     uint32_t freq, uint32_t shift,
00410                                     enum dvb_sat_polarization pol);
00411
00412 int dvb_new_entry_is_needed(struct dvb_entry *entry,
00413                           struct dvb_entry *last_entry,
00414                           uint32_t freq, int shift,
00415                           enum dvb_sat_polarization pol, uint32_t stream_id);
00416
00417 struct dvb_entry *dvb_scan_add_entry_ex(struct dvb_v5_fe_parms *parms,
00418                                         struct dvb_entry *first_entry,
00419                                         struct dvb_entry *entry,
00420                                         uint32_t freq, uint32_t shift,
00421                                         enum dvb_sat_polarization pol,
00422                                         uint32_t stream_id);
00423
00424 void dvb_update_transponders(struct dvb_v5_fe_parms *parms,
00425                             struct dvb_v5_descriptors *dvb_scan_handler,
00426                             struct dvb_entry *first_entry,
00427                             struct dvb_entry *entry);
00428 #endif
00429
00430 #ifdef __cplusplus
00431 }
00432 #endif
00433
00434 #endif

```

9.68 lib/include/libdvbv5/dvb-v5-std.h File Reference

Provides libdvbv5 defined properties for the frontend.

```

#include <stddef.h>
#include "dvb-frontend.h"

```

Macros

- `#define DTV_USER_COMMAND_START`
Start number for libdvbv5 user commands.
- `#define DTV_POLARIZATION`
Satellite polarization (for Satellite delivery systems)
- `#define DTV_VIDEO_PID`
Video PID.
- `#define DTV_AUDIO_PID`
Audio PID.
- `#define DTV_SERVICE_ID`
MPEG TS service ID.
- `#define DTV_CH_NAME`
Digital TV service name.
- `#define DTV_VCHANNEL`
Digital TV channel number.
- `#define DTV_SAT_NUMBER`
Number of the satellite (used on multi-dish Satellite systems)

- `#define DTV_DISEQC_WAIT`
Extra time needed to wait for DiSeqC to complete, in ms.
- `#define DTV_DISEQC_LNB`
LNBf name.
- `#define DTV_FREQ_BPF`
SCR/Unicable band-pass filter frequency in kHz.
- `#define DTV_PLS_CODE`
DVB-T2 PLS code.
- `#define DTV_PLS_MODE`
DVB-T2 PLS mode.
- `#define DTV_COUNTRY_CODE`
Country variant of international delivery system standard.
- `#define DTV_MAX_USER_COMMAND`
Last user command.
- `#define DTV_USER_NAME_SIZE`
Number of user commands.
- `#define DTV_STAT_COMMAND_START`
Start number for libdvbv5 statistics commands.
- `#define DTV_STATUS`
Lock status of a DTV frontend.
- `#define DTV_BER`
Bit Error Rate.
- `#define DTV_PER`
Packet Error Rate.
- `#define DTV_QUALITY`
A quality indicator that represents if a locked channel provides a good, OK or poor signal.
- `#define DTV_PRE_BER`
Bit Error Rate before Viterbi.
- `#define DTV_MAX_STAT_COMMAND`
Last statistics command.
- `#define DTV_STAT_NAME_SIZE`
Number of statistics commands.
- `#define DTV_NUM_KERNEL_STATS`
Number of statistics commands provided by the Kernel.
- `#define DTV_NUM_STATS_PROPS`
Total number of statistics commands.

Enumerations

- enum `dvb_sat_polarization` {
 `POLARIZATION_OFF` , `POLARIZATION_H` , `POLARIZATION_V` , `POLARIZATION_L` ,
 `POLARIZATION_R` }
Polarization types for Satellite systems.
- enum `dvb_quality` { `DVB_QUAL_UNKNOWN` , `DVB_QUAL_POOR` , `DVB_QUAL_OK` , `DVB_QUAL_GOOD` }
Provides an estimation about the user's experience while watching to a given MPEG stream.

9.68.1 Detailed Description

Provides libdvbv5 defined properties for the frontend.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [dvb-v5-std.h](#).

9.69 dvb-v5-std.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2014 - Mauro Carvalho Chehab
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  * Per-delivery system properties defined at libdvbv5 scope, following
00019  * the same model as defined at the Linux DVB media specs:
00020  * http://linuxtv.org/downloads/v4l-dvb-apis/FE\_GET\_SET\_PROPERTY.html
00021  */
00022 #ifndef _DVB_V5_STD_H
00023 #define _DVB_V5_STD_H
00024
00025 #include <stddef.h>
00026 #include "dvb-frontend.h"
00027
00039 /*
00040  * User DTV codes, for internal usage. There are two sets of
00041  * properties. One for DTV properties and another one for statistics
00042  */
00043
00044 /*
00045  * First set: DTV properties that don't belong to Kernel space
00046  *
00047  * Those properties contain data that comes from the MPEG-TS
00048  * tables, like audio/video/other PIDs, and satellite config
00049  */
00050
00107 #define DTV_USER_COMMAND_START 256
00108
00109 #define DTV_POLARIZATION (DTV_USER_COMMAND_START + 0)
00110 #define DTV_VIDEO_PID (DTV_USER_COMMAND_START + 1)
00111 #define DTV_AUDIO_PID (DTV_USER_COMMAND_START + 2)
00112 #define DTV_SERVICE_ID (DTV_USER_COMMAND_START + 3)
00113 #define DTV_CH_NAME (DTV_USER_COMMAND_START + 4)
00114 #define DTV_VCHANNEL (DTV_USER_COMMAND_START + 5)
00115 #define DTV_SAT_NUMBER (DTV_USER_COMMAND_START + 6)
00116 #define DTV_DISEQC_WAIT (DTV_USER_COMMAND_START + 7)

```

```

00117 #define DTV_DISEQC_LNB          (DTV_USER_COMMAND_START + 8)
00118 #define DTV_FREQ_BPF            (DTV_USER_COMMAND_START + 9)
00119 #define DTV_PLS_CODE            (DTV_USER_COMMAND_START + 10)
00120 #define DTV_PLS_MODE            (DTV_USER_COMMAND_START + 11)
00121 #define DTV_COUNTRY_CODE        (DTV_USER_COMMAND_START + 12)
00122
00123 #define DTV_MAX_USER_COMMAND    DTV_COUNTRY_CODE
00124
00125 #define DTV_USER_NAME_SIZE      (1 + DTV_MAX_USER_COMMAND - DTV_USER_COMMAND_START)
00126
00138 enum dvb_sat_polarization {
00139     POLARIZATION_OFF          = 0,
00140     POLARIZATION_H            = 1,
00141     POLARIZATION_V            = 2,
00142     POLARIZATION_L            = 3,
00143     POLARIZATION_R            = 4,
00144 };
00145
00146 /*
00147  * Second set: DTV statistics
00148  *
00149  * Those properties contain statistics measurements that aren't
00150  * either provided by the Kernel via property cmd/value pair,
00151  * like status (with has its own ioctl), or that are derivated
00152  * measures from two or more Kernel reported stats.
00153  */
00154
00197 #define DTV_STAT_COMMAND_START 512
00198
00199 #define DTV_STATUS              (DTV_STAT_COMMAND_START + 0)
00200 #define DTV_BER                (DTV_STAT_COMMAND_START + 1)
00201 #define DTV_PER                (DTV_STAT_COMMAND_START + 2)
00202 #define DTV_QUALITY            (DTV_STAT_COMMAND_START + 3)
00203 #define DTV_PRE_BER            (DTV_STAT_COMMAND_START + 4)
00204
00205 #define DTV_MAX_STAT_COMMAND    DTV_PRE_BER
00206
00207 #define DTV_STAT_NAME_SIZE      (1 + DTV_MAX_STAT_COMMAND - DTV_STAT_COMMAND_START)
00208
00209 /* There are currently 8 stats provided on Kernelspace */
00210 #define DTV_NUM_KERNEL_STATS    8
00211
00212 #define DTV_NUM_STATS_PROPS      (DTV_NUM_KERNEL_STATS + DTV_STAT_NAME_SIZE)
00213
00231 enum dvb_quality {
00232     DVB_QUAL_UNKNOWN = 0,
00233     DVB_QUAL_POOR,
00234     DVB_QUAL_OK,
00235     DVB_QUAL_GOOD,
00236 };
00237
00238 #ifndef _DOXYGEN
00239
00240 /*
00241  * Some tables to translate from value to string
00242  *
00243  * These tables are raw ways to translate from some DTV values into strings.
00244  * Please use the API-provided function dvb_cmd_name() and dvb_dvb_attr_names(),
00245  * instead of using the tables directly.
00246  */
00247
00248 extern const unsigned int sys_dvbt_props[];
00249 extern const unsigned int sys_dvbt2_props[];
00250 extern const unsigned int sys_isdbt_props[];
00251 extern const unsigned int sys_atsc_props[];
00252 extern const unsigned int sys_atscmh_props[];
00253 extern const unsigned int sys_dvbc_annex_ac_props[];
00254 extern const unsigned int sys_dvbc_annex_b_props[];
00255 extern const unsigned int sys_dvbs_props[];
00256 extern const unsigned int sys_dvbs2_props[];
00257 extern const unsigned int sys_turbo_props[];
00258 extern const unsigned int sys_isdbs_props[];
00259 extern const unsigned int *dvb_v5_delivery_system[];
00260 extern const char *dvb_sat_pol_name[6];
00261 extern const char *dvb_user_name[DTV_USER_NAME_SIZE + 1];
00262 extern const char *dvb_stat_name[DTV_STAT_NAME_SIZE + 1];
00263 extern const void *dvb_user_attr_names[];
00264
00265 #endif /* DOXYGEN_SHOULD_SKIP_THIS */
00266
00267 #endif

```

9.70 lib/include/libdvbv5/eit.h File Reference

Provides the table parser for the DVB EIT (Event Information Table)

```
#include <stdint.h>
#include <unistd.h>
#include <time.h>
#include <libdvbv5/header.h>
```

Data Structures

- struct [dvb_table_eit_event](#)
DVB EIT event table.
- struct [dvb_table_eit](#)
DVB EIT table.

Macros

- #define [DVB_TABLE_EIT](#)
DVB EIT table ID for the actual TS.
- #define [DVB_TABLE_EIT_OTHER](#)
DVB EIT table ID for other TS.
- #define [DVB_TABLE_EIT_PID](#)
DVB EIT Program ID.
- #define [DVB_TABLE_EIT_SCHEDULE](#)
Start table ID for the DVB EIT schedule data on the actual TS The range has 0x0f elements (0x50 to 0x5F).
- #define [DVB_TABLE_EIT_SCHEDULE_OTHER](#)
Start table ID for the DVB EIT schedule data on other TS The range has 0x0f elements (0x60 to 0x6F).
- #define [dvb_eit_event_foreach](#)(_event, _eit)
Macro used to find event on a DVB EIT table.

Functions

- ssize_t [dvb_table_eit_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, struct [dvb_table_eit](#) **table)
Initializes and parses EIT table.
- void [dvb_table_eit_free](#) (struct [dvb_table_eit](#) *table)
Frees all data allocated by the DVB EIT table parser.
- void [dvb_table_eit_print](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_table_eit](#) *table)
Prints the content of the DVB EIT table.
- void [dvb_time](#) (const uint8_t data[5], struct tm *tm)
Converts a DVB EIT formatted timestamp into struct tm.

Variables

- const char * [dvb_eit_running_status_name](#) [8]
Converts a running_status field into string.

9.70.1 Detailed Description

Provides the table parser for the DVB EIT (Event Information Table)

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The table described herein is defined at:

- ETSI EN 300 468

See also

<http://www.etherguidesystems.com/Help/SDOs/dvb/syntax/tablesections/EIT.aspx>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [eit.h](#).

9.70.2 Variable Documentation

9.70.2.1 dvb_eit_running_status_name `const char* dvb_eit_running_status_name[8] [extern]`

Converts a running_status field into string.

9.71 eit.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  */
00019 */
00020
00040 #ifndef _EIT_H
00041 #define _EIT_H
00042
00043 #include <stdint.h>
00044 #include <unistd.h> /* ssize_t */
00045 #include <time.h>
00046
00047 #include <libdvbv5/header.h>
00048
00068 #define DVB_TABLE_EIT 0x4E
00069 #define DVB_TABLE_EIT_OTHER 0x4F
00070 #define DVB_TABLE_EIT_PID 0x12
00071
00072 #define DVB_TABLE_EIT_SCHEDULE 0x50
00073 #define DVB_TABLE_EIT_SCHEDULE_OTHER 0x60
00074
00105 struct dvb_table_eit_event {
00106     uint16_t event_id;
00107     union {
00108         uint16_t bitfield1; /* first 2 bytes are MJD, they need to be bswapped */
00109         uint8_t dvbstart[5];
00110     } __attribute__((packed));
00111     uint8_t dvbduration[3];
00112     union {
00113         uint16_t bitfield2;
00114         struct {
00115             uint16_t desc_length:12;
00116             uint16_t free_CA_mode:1;
00117             uint16_t running_status:3;
00118         } __attribute__((packed));
00119     } __attribute__((packed));
00120     struct dvb_desc *descriptor;
00121     struct dvb_table_eit_event *next;
00122     struct tm start;
00123     uint32_t duration;
00124     uint16_t service_id;
00125 } __attribute__((packed));
00126
00146 struct dvb_table_eit {
00147     struct dvb_table_header header;
00148     uint16_t transport_id;
00149     uint16_t network_id;
00150     uint8_t last_segment;
00151     uint8_t last_table_id;
00152     struct dvb_table_eit_event *event;
00153 } __attribute__((packed));
00154
00162 #define dvb_eit_event_foreach(_event, _eit) \
00163     if (_eit && _eit->event) \
00164         for( struct dvb_table_eit_event *_event = _eit->event; _event; _event = _event->next )
00165
00166 struct dvb_v5_fe_parms;
00167
00169 extern const char *dvb_eit_running_status_name[8];
00170
00171 #ifdef __cplusplus
00172 extern "C" {
00173 #endif
00174
00191 ssize_t dvb_table_eit_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00192                             ssize_t buflen, struct dvb_table_eit **table);
00193

```

```
00200 void dvb_table_eit_free(struct dvb_table_eit *table);
00201
00209 void dvb_table_eit_print(struct dvb_v5_fe_parms *parms,
00210                          struct dvb_table_eit *table);
00211
00220 void dvb_time(const uint8_t data[5], struct tm *tm);
00221
00222 #ifdef __cplusplus
00223 }
00224 #endif
00225
00226 #endif
```

9.72 lib/include/libdvbv5/header.h File Reference

Provides the MPEG TS table headers.

```
#include <stdint.h>
#include <unistd.h>
```

Data Structures

- struct [dvb_ts_packet_header](#)
Header of a MPEG-TS transport packet.
- struct [dvb_table_header](#)
Header of a MPEG-TS table.

Functions

- void [dvb_table_header_init](#) (struct [dvb_table_header](#) *header)
Initializes and parses MPEG-TS table header.
- void [dvb_table_header_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_table_header](#) *header)
Prints the content of the MPEG-TS table header.

9.72.1 Detailed Description

Provides the MPEG TS table headers.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [header.h](#).

9.73 header.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  * Described at ISO/IEC 13818-1
00020  */
00021
00022 #ifndef _HEADER_H
00023 #define _HEADER_H
00024
00025 #include <stdint.h>
00026 #include <unistd.h> /* ssize_t */
00027
00057 struct dvb_ts_packet_header {
00058     uint8_t sync_byte;
00059     union {
00060         uint16_t bitfield;
00061         struct {
00062             uint16_t pid:13;
00063             uint16_t transport_priority:1;
00064             uint16_t payload_unit_start_indicator:1;
00065             uint16_t transport_error_indicator:1;
00066         } __attribute__((packed));
00067     } __attribute__((packed));
00068     uint8_t continuity_counter:4;
00069     uint8_t adaptation_field_control:2;
00070     uint8_t transport_scrambling_control:2;
00071
00072     /* Only if adaptation_field_control > 1 */
00073     uint8_t adaptation_field_length;
00074     /* Only if adaptation_field_length >= 1 */
00075     struct {
00076         uint8_t extension:1;
00077         uint8_t private_data:1;
00078         uint8_t splicing_point:1;
00079         uint8_t OPCR:1;
00080         uint8_t PCR:1;
00081         uint8_t priority:1;
00082         uint8_t random_access:1;
00083         uint8_t discontinued:1;
00084     } __attribute__((packed));
00085 } __attribute__((packed));
00086
00103 struct dvb_table_header {
00104     uint8_t table_id;
00105     union {
00106         uint16_t bitfield;
00107         struct {
00108             uint16_t section_length:12;
00109             uint8_t one:2;
00110             uint8_t zero:1;
00111             uint8_t syntax:1;
00112         } __attribute__((packed));
00113     } __attribute__((packed));
00114     uint16_t id; /* TS ID */
00115     uint8_t current_next:1;
00116     uint8_t version:5;
00117     uint8_t one2:2;
00118
00119     uint8_t section_id; /* section_number */
00120     uint8_t last_section; /* last_section_number */
00121 } __attribute__((packed));
00122
00123 struct dvb_v5_fe_parms;
00124
00125 #ifdef __cplusplus
00126 extern "C" {
00127 #endif
00128

```

```

00135 void dvb_table_header_init (struct dvb_table_header *header);
00143 void dvb_table_header_print (struct dvb_v5_fe_parms *parms,
00144                             const struct dvb_table_header *header);
00145
00146 #ifdef __cplusplus
00147 }
00148 #endif
00149
00150 #endif

```

9.74 lib/include/libdvbv5/mgt.h File Reference

Provides the table parser for the ATSC MGT (Master Guide Table)

```

#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/atsc_header.h>

```

Data Structures

- struct [atsc_table_mgt_table](#)
ATSC tables description at MGT table.
- struct [atsc_table_mgt](#)
ATSC MGT table.

Macros

- #define [ATSC_TABLE_MGT](#)
ATSC MGT table ID.
- #define [atsc_mgt_table_foreach](#)(_table, _mgt)
Macro used to find a table inside a MGT table.

Functions

- ssize_t [atsc_table_mgt_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, struct [atsc_table_mgt](#) **table)
Initializes and parses MGT table.
- void [atsc_table_mgt_free](#) (struct [atsc_table_mgt](#) *table)
Frees all data allocated by the MGT table parser.
- void [atsc_table_mgt_print](#) (struct [dvb_v5_fe_parms](#) *parms, struct [atsc_table_mgt](#) *table)
Prints the content of the MGT table.

9.74.1 Detailed Description

Provides the table parser for the ATSC MGT (Master Guide Table)

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Relevant specs

The table described herein is defined at:

- ATSC A/65:2009

See also

<http://www.etherguidesystems.com/help/sdos/atsc/syntax/tablesections/MGT.aspx>↵

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [mgt.h](#).

9.74.2 Macro Definition Documentation

9.74.2.1 atsc_mgt_table_foreach `#define atsc_mgt_table_foreach(
 _table,
 _mgt)`

Macro used to find a table inside a MGT table.

Parameters

<code>_table</code>	channel to seek
<code>_mgt</code>	pointer to struct atsc_table_mgt_table

Definition at line [136](#) of file [mgt.h](#).

9.75 mgt.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00038 #ifndef _MGT_H
00039 #define _MGT_H
00040
00041 #include <stdint.h>
00042 #include <unistd.h> /* ssize_t */
00043
00044 #include <libdvbv5/atsc_header.h>
00045
00051 #define ATSC_TABLE_MGT 0xC7
00052
00077 struct atsc_table_mgt_table {
00078     uint16_t type;
00079     union {
00080         uint16_t bitfield;
00081         struct {
00082             uint16_t pid:13;
00083             uint16_t one:3;
00084         } __attribute__((packed));
00085     } __attribute__((packed));
00086     uint8_t type_version:5;
00087     uint8_t one2:3;
00088     uint32_t size;
00089     union {
00090         uint16_t bitfield2;
00091         struct {
00092             uint16_t desc_length:12;
00093             uint16_t one3:4;
00094         } __attribute__((packed));
00095     } __attribute__((packed));
00096     struct dvb_desc *descriptor;
00097     struct atsc_table_mgt_table *next;
00098 } __attribute__((packed));
00099
00122 struct atsc_table_mgt {
00123     struct dvb_table_header header;
00124     uint8_t protocol_version;
00125     uint16_t tables;
00126     struct atsc_table_mgt_table *table;
00127     struct dvb_desc *descriptor;
00128 } __attribute__((packed));
00129
00136 #define atsc_mgt_table_foreach( _table, _mgt ) \
00137     if ( _mgt && _mgt->_table ) \
00138         for( struct atsc_table_mgt_table *_table = _mgt->table; _table; _table = _table->next
00139             ) \
00139
00140 struct dvb_v5_fe_parms;
00141
00142 #ifdef __cplusplus
00143 extern "C" {
00144 #endif
00145
00162 ssize_t atsc_table_mgt_init(struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00163                             ssize_t buflen, struct atsc_table_mgt **table);
00164
00171 void atsc_table_mgt_free(struct atsc_table_mgt *table);
00172
00180 void atsc_table_mgt_print(struct dvb_v5_fe_parms *parms,
00181                           struct atsc_table_mgt *table);
00182
00183 #ifdef __cplusplus
00184 }
00185 #endif
00186

```

```
00187 #endif
```

9.76 lib/include/libdvbv5/mpeg_es.h File Reference

Provides the table parser for the MPEG-TS Elementary Stream.

```
#include <stdint.h>
#include <unistd.h>
```

Data Structures

- struct [dvb_mpeg_es_seq_start](#)
MPEG ES Sequence header.
- struct [dvb_mpeg_es_pic_start](#)
MPEG ES Picture start header.

Macros

- #define [DVB_MPEG_ES_PIC_START](#)
Picture Start.
- #define [DVB_MPEG_ES_USER_DATA](#)
User Data.
- #define [DVB_MPEG_ES_SEQ_START](#)
Sequence Start.
- #define [DVB_MPEG_ES_SEQ_EXT](#)
Extension.
- #define [DVB_MPEG_ES_GOP](#)
Group Of Pictures.
- #define [DVB_MPEG_ES_SLICES](#)
Slices.

Enumerations

- enum [dvb_mpeg_es_frame_t](#) {
 [DVB_MPEG_ES_FRAME_UNKNOWN](#) , [DVB_MPEG_ES_FRAME_I](#) , [DVB_MPEG_ES_FRAME_P](#) ,
 [DVB_MPEG_ES_FRAME_B](#) ,
 [DVB_MPEG_ES_FRAME_D](#) }
MPEG frame types.

Functions

- int [dvb_mpeg_es_seq_start_init](#) (const uint8_t *buf, ssize_t buflen, struct [dvb_mpeg_es_seq_start](#) *seq_start)
Initialize a struct [dvb_mpeg_es_seq_start](#) from buffer.
- void [dvb_mpeg_es_seq_start_print](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_mpeg_es_seq_start](#) *seq_start)
Print details of struct [dvb_mpeg_es_seq_start](#).
- int [dvb_mpeg_es_pic_start_init](#) (const uint8_t *buf, ssize_t buflen, struct [dvb_mpeg_es_pic_start](#) *pic_start)
Initialize a struct [dvb_mpeg_es_pic_start](#) from buffer.
- void [dvb_mpeg_es_pic_start_print](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_mpeg_es_pic_start](#) *pic_start)
Print details of struct [dvb_mpeg_es_pic_start](#).

Variables

- const char * [dvb_mpeg_es_frame_names](#) [5]
Vector that translates from enum `dvb_mpeg_es_frame_t` to string.

9.76.1 Detailed Description

Provides the table parser for the MPEG-TS Elementary Stream.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Relevant specs

The table described herein is defined in ISO 13818-2

See also

<http://dvd.sourceforge.net/dvdinfo/mpeghdrs.html>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [mpeg_es.h](#).

9.77 mpeg_es.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013-2014 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
00018  */
00019
00020 #ifndef _MPEG_ES_H
00021 #define _MPEG_ES_H
00022
00040 #include <stdint.h>
00041 #include <unistd.h> /* ssize_t */
00042

```

```

00063 #define DVB_MPEG_ES_PIC_START 0x00
00064 #define DVB_MPEG_ES_USER_DATA 0xb2
00065 #define DVB_MPEG_ES_SEQ_START 0xb3
00066 #define DVB_MPEG_ES_SEQ_EXT 0xb5
00067 #define DVB_MPEG_ES_GOP 0xb8
00068 #define DVB_MPEG_ES_SLICES 0x01 ... 0xaf
00069
00088 struct dvb_mpeg_es_seq_start {
00089     union {
00090         uint32_t bitfield;
00091         struct {
00092             uint32_t type:8;
00093             uint32_t sync:24;
00094         } __attribute__((packed));
00095     } __attribute__((packed));
00096     union {
00097         uint32_t bitfield2;
00098         struct {
00099             uint32_t framerate:4;
00100             uint32_t aspect:4;
00101             uint32_t height:12;
00102             uint32_t width:12;
00103         } __attribute__((packed));
00104     } __attribute__((packed));
00105     union {
00106         uint32_t bitfield3;
00107         struct {
00108             uint32_t qm_nonintra:1;
00109             uint32_t qm_intra:1;
00110             uint32_t constrained:1;
00111             uint32_t vbv:10; // Size of video buffer verifier = 16*1024*vbv buf size
00112             uint32_t one:1;
00113             uint32_t bitrate:18;
00114         } __attribute__((packed));
00115     } __attribute__((packed));
00116 } __attribute__((packed));
00117
00130 struct dvb_mpeg_es_pic_start {
00131     union {
00132         uint32_t bitfield;
00133         struct {
00134             uint32_t type:8;
00135             uint32_t sync:24;
00136         } __attribute__((packed));
00137     } __attribute__((packed));
00138     union {
00139         uint32_t bitfield2;
00140         struct {
00141             uint32_t dummy:3;
00142             uint32_t vbv_delay:16;
00143             uint32_t coding_type:3;
00144             uint32_t temporal_ref:10;
00145         } __attribute__((packed));
00146     } __attribute__((packed));
00147 } __attribute__((packed));
00148
00165 enum dvb_mpeg_es_frame_t
00166 {
00167     DVB_MPEG_ES_FRAME_UNKNOWN,
00168     DVB_MPEG_ES_FRAME_I,
00169     DVB_MPEG_ES_FRAME_P,
00170     DVB_MPEG_ES_FRAME_B,
00171     DVB_MPEG_ES_FRAME_D
00172 };
00173
00178 extern const char *dvb_mpeg_es_frame_names[5];
00179
00180 struct dvb_v5_fe_parms;
00181
00182 #ifdef __cplusplus
00183 extern "C" {
00184 #endif
00185
00200 int dvb_mpeg_es_seq_start_init (const uint8_t *buf, ssize_t buflen,
00201                                struct dvb_mpeg_es_seq_start *seq_start);
00202
00212 void dvb_mpeg_es_seq_start_print(struct dvb_v5_fe_parms *parms,
00213                                 struct dvb_mpeg_es_seq_start *seq_start);
00214
00229 int dvb_mpeg_es_pic_start_init (const uint8_t *buf, ssize_t buflen,
00230                                 struct dvb_mpeg_es_pic_start *pic_start);
00231
00241 void dvb_mpeg_es_pic_start_print(struct dvb_v5_fe_parms *parms,
00242                                 struct dvb_mpeg_es_pic_start *pic_start);
00243
00244 #ifdef __cplusplus
00245 }

```

```
00246 #endif
00247
00248 #endif
```

9.78 lib/include/libdvbv5/mpeg_pes.h File Reference

Provides the table parser for the MPEG-PES Elementary Stream.

```
#include <stdint.h>
#include <unistd.h>
```

Data Structures

- struct [ts_t](#)
MPEG PES timestamp structure, used for dts and pts.
- struct [dvb_mpeg_pes_optional](#)
MPEG PES optional header.
- struct [dvb_mpeg_pes](#)
MPEG PES data structure.

Macros

- #define [DVB_MPEG_PES](#)
MPEG Packetized Elementary Stream magic.
- #define [DVB_MPEG_PES_AUDIO](#)
PES Audio.
- #define [DVB_MPEG_PES_VIDEO](#)
PES Video.
- #define [DVB_MPEG_STREAM_MAP](#)
PES Stream map.
- #define [DVB_MPEG_STREAM_PADDING](#)
PES padding.
- #define [DVB_MPEG_STREAM_PRIVATE_2](#)
PES private.
- #define [DVB_MPEG_STREAM_ECM](#)
PES ECM Stream.
- #define [DVB_MPEG_STREAM_EMM](#)
PES EMM Stream.
- #define [DVB_MPEG_STREAM_DIRECTORY](#)
PES Stream directory.
- #define [DVB_MPEG_STREAM_DSMCC](#)
PES DSMCC.
- #define [DVB_MPEG_STREAM_H222E](#)
PES H.222.1 type E.

Functions

- `ssize_t dvb_mpeg_pes_init` (struct `dvb_v5_fe_parms` *parms, const uint8_t *buf, ssize_t buflen, uint8_t *table)
Initialize a struct `dvb_mpeg_pes` from buffer.
- void `dvb_mpeg_pes_free` (struct `dvb_mpeg_pes` *pes)
Deallocate memory associated with a struct `dvb_mpeg_pes`.
- void `dvb_mpeg_pes_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_mpeg_pes` *pes)
Print details of struct `dvb_mpeg_pes`.

9.78.1 Detailed Description

Provides the table parser for the MPEG-PES Elementary Stream.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Relevant specs

The table described herein is defined in ISO 13818-1

See also

<http://dvd.sourceforge.net/dvdinfo/pes-hdr.html>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [mpeg_pes.h](#).

9.79 mpeg_pes.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2013-2014 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  *
```

```

00018  */
00019
00020 #ifndef _MPEG_PES_H
00021 #define _MPEG_PES_H
00022
00040 #include <stdint.h>
00041 #include <unistd.h> /* ssize_t */
00042
00043
00080 #define DVB_MPEG_PES 0x00001
00081
00082 #define DVB_MPEG_PES_AUDIO 0xc0 ... 0xcf
00083 #define DVB_MPEG_PES_VIDEO 0xe0 ... 0xef
00084
00085 #define DVB_MPEG_STREAM_MAP 0xBC
00086 #define DVB_MPEG_STREAM_PADDING 0xBE
00087 #define DVB_MPEG_STREAM_PRIVATE_2 0x5F
00088 #define DVB_MPEG_STREAM_ECM 0x70
00089 #define DVB_MPEG_STREAM_EMM 0x71
00090 #define DVB_MPEG_STREAM_DIRECTORY 0xFF
00091 #define DVB_MPEG_STREAM_DSMCC 0x7A
00092 #define DVB_MPEG_STREAM_H222E 0xF8
00093
00108 struct ts_t {
00109     uint8_t one:1;
00110     uint8_t bits30:3;
00111     uint8_t tag:4;
00112
00113     union {
00114         uint16_t bitfield;
00115         struct {
00116             uint16_t one1:1;
00117             uint16_t bits15:15;
00118         } __attribute__((packed));
00119     } __attribute__((packed));
00120
00121     union {
00122         uint16_t bitfield2;
00123         struct {
00124             uint16_t one2:1;
00125             uint16_t bits00:15;
00126         } __attribute__((packed));
00127     } __attribute__((packed));
00128 } __attribute__((packed));
00129
00152 struct dvb_mpeg_pes_optional {
00153     union {
00154         uint16_t bitfield;
00155         struct {
00156             uint16_t PES_extension:1;
00157             uint16_t PES_CRC:1;
00158             uint16_t additional_copy_info:1;
00159             uint16_t DSM_trick_mode:1;
00160             uint16_t ES_rate:1;
00161             uint16_t ESCR:1;
00162             uint16_t PTS_DTS:2;
00163             uint16_t original_or_copy:1;
00164             uint16_t copyright:1;
00165             uint16_t data_alignment_indicator:1;
00166             uint16_t PES_priority:1;
00167             uint16_t PES_scrambling_control:2;
00168             uint16_t two:2;
00169         } __attribute__((packed));
00170     } __attribute__((packed));
00171     uint8_t length;
00172     uint64_t pts;
00173     uint64_t dts;
00174 } __attribute__((packed));
00175
00186 struct dvb_mpeg_pes {
00187     union {
00188         uint32_t bitfield;
00189         struct {
00190             uint32_t stream_id:8;
00191             uint32_t sync:24;
00192         } __attribute__((packed));
00193     } __attribute__((packed));
00194     uint16_t length;
00195     struct dvb_mpeg_pes_optional optional[];
00196 } __attribute__((packed));
00197
00198 struct dvb_v5_fe_parms;
00199
00200 #ifdef __cplusplus
00201 extern "C" {
00202 #endif
00203

```



```
00219 ssize_t dvb_mpeg_pes_init(struct dvb_v5_fe_parms *parms, const uint8_t *buf, ssize_t buflen,
00220                             uint8_t *table);
00221
00231 void dvb_mpeg_pes_free(struct dvb_mpeg_pes *pes);
00232
00242 void dvb_mpeg_pes_print (struct dvb_v5_fe_parms *parms, struct dvb_mpeg_pes *pes);
00243
00244 #ifdef __cplusplus
00245 }
00246 #endif
00247
00248 #endif
```

9.80 lib/include/libdvbv5/mpeg_ts.h File Reference

Provides the table parser for the MPEG-PES Elementary Stream.

```
#include <stdint.h>
#include <unistd.h>
```

Data Structures

- struct [dvb_mpeg_ts_adaption](#)
MPEG TS header adaption field.
- struct [dvb_mpeg_ts](#)
MPEG TS header.

Macros

- #define [DVB_MPEG_TS](#)
MPEG Transport Stream magic.
- #define [DVB_MPEG_TS_PACKET_SIZE](#)
Size of an MPEG packet.

Functions

- [ssize_t dvb_mpeg_ts_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, uint8_t *table, ssize_t *table_length)
Initialize a struct [dvb_mpeg_ts](#) from buffer.
- void [dvb_mpeg_ts_free](#) (struct [dvb_mpeg_ts](#) *ts)
Deallocate memory associated with a struct [dvb_mpeg_ts](#).
- void [dvb_mpeg_ts_print](#) (struct [dvb_v5_fe_parms](#) *parms, struct [dvb_mpeg_ts](#) *ts)
Print details of struct [dvb_mpeg_ts](#).

9.80.1 Detailed Description

Provides the table parser for the MPEG-PES Elementary Stream.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Andre Roth

Relevant specs

The table described herein is defined in ISO 13818-1

See also

http://en.wikipedia.org/wiki/MPEG_transport_stream

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [mpeg_ts.h](#).

9.81 mpeg_ts.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c) 2013-2014 - Andre Roth <neolynx@gmail.com>
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU Lesser General Public License as published by
00006  * the Free Software Foundation version 2.1 of the License.
00007  *
00008  * This program is distributed in the hope that it will be useful,
00009  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00010  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011  * GNU Lesser General Public License for more details.
00012  *
00013  * You should have received a copy of the GNU Lesser General Public License
00014  * along with this program; if not, write to the Free Software
00015  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00016  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00017  */
00018 */
00019
00020 #ifndef _MPEG_TS_H
00021 #define _MPEG_TS_H
00022
00039 #include <stdint.h>
00040 #include <unistd.h> /* ssize_t */
00041
00050 #define DVB_MPEG_TS 0x47
00051 #define DVB_MPEG_TS_PACKET_SIZE 188
00052
00070 struct dvb_mpeg_ts_adaption {
00071     uint8_t length;
00072     struct {
00073         uint8_t extension:1;
00074         uint8_t private_data:1;
00075         uint8_t splicing_point:1;
00076         uint8_t OPCR:1;
```

```

00077         uint8_t PCR:1;
00078         uint8_t priority:1;
00079         uint8_t random_access:1;
00080         uint8_t discontinued:1;
00081     } __attribute__((packed));
00082     uint8_t data[];
00083 } __attribute__((packed));
00084
00101 struct dvb_mpeg_ts {
00102     uint8_t sync_byte;
00103     union {
00104         uint16_t bitfield;
00105         struct {
00106             uint16_t pid:13;
00107             uint16_t priority:1;
00108             uint16_t payload_start:1;
00109             uint16_t tei:1;
00110         } __attribute__((packed));
00111     } __attribute__((packed));
00112     struct {
00113         uint8_t continuity_counter:4;
00114         uint8_t payload:1;
00115         uint8_t adaptation_field:1;
00116         uint8_t scrambling:2;
00117     } __attribute__((packed));
00118     struct dvb_mpeg_ts_adaption adaption[];
00119 } __attribute__((packed));
00120
00121 struct dvb_v5_fe_parms;
00122
00123 #ifdef __cplusplus
00124 extern "C" {
00125 #endif
00126
00143 ssize_t dvb_mpeg_ts_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf, ssize_t buflen,
00144                          uint8_t *table, ssize_t *table_length);
00145
00155 void dvb_mpeg_ts_free(struct dvb_mpeg_ts *ts);
00156
00166 void dvb_mpeg_ts_print(struct dvb_v5_fe_parms *parms, struct dvb_mpeg_ts *ts);
00167
00168 #ifdef __cplusplus
00169 }
00170 #endif
00171
00172 #endif

```

9.82 lib/include/libdvbv5/nit.h File Reference

Provides the descriptors for NIT MPEG-TS table.

```

#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/header.h>
#include <libdvbv5/descriptors.h>

```

Data Structures

- union [dvb_table_nit_transport_header](#)
MPEG-TS NIT transport header.
- struct [dvb_table_nit_transport](#)
MPEG-TS NIT transport table.
- struct [dvb_table_nit](#)
MPEG-TS NIT table.

Macros

- `#define DVB_TABLE_NIT`
NIT table ID.
- `#define DVB_TABLE_NIT2`
NIT table ID (alternative table ID)
- `#define DVB_TABLE_NIT_PID`
NIT Program ID.
- `#define dvb_nit_transport_foreach(_tran, _nit)`
Macro used to find a transport inside a NIT table.

Typedefs

- `typedef void nit_handler_callback_t(struct dvb_table_nit *nit, struct dvb_desc *desc, void *priv)`
typedef for a callback used when a NIT table entry is found
- `typedef void nit_tran_handler_callback_t(struct dvb_table_nit *nit, struct dvb_table_nit_transport *tran, struct dvb_desc *desc, void *priv)`
typedef for a callback used when a NIT transport table entry is found

Functions

- `ssize_t dvb_table_nit_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf, ssize_t buflen, struct dvb_table_nit **table)`
Initializes and parses NIT table.
- `void dvb_table_nit_free (struct dvb_table_nit *table)`
Frees all data allocated by the NIT table parser.
- `void dvb_table_nit_print (struct dvb_v5_fe_parms *parms, struct dvb_table_nit *table)`
Prints the content of the NIT table.
- `void dvb_table_nit_descriptor_handler (struct dvb_v5_fe_parms *parms, struct dvb_table_nit *table, enum descriptors descriptor, nit_handler_callback_t *call_nit, nit_tran_handler_callback_t *call_tran, void *priv)`
For each entry at NIT and NIT transport tables, call a callback.

9.82.1 Detailed Description

Provides the descriptors for NIT MPEG-TS table.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Bug Report

Please submit bug report and patches to linux-media@vger.kernel.org

Relevant specs

The table described herein is defined at:

- ISO/IEC 13818-1
- ETSI EN 300 468

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [nit.h](#).

9.83 nit.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00021 #ifndef _NIT_H
00022 #define _NIT_H
00023
00024 #include <stdint.h>
00025 #include <unistd.h> /* ssize_t */
00026
00027 #include <libdvbv5/header.h>
00028 #include <libdvbv5/descriptors.h>
00029
00061 #define DVB_TABLE_NIT 0x40
00062 #define DVB_TABLE_NIT2 0x41
00063 #define DVB_TABLE_NIT_PID 0x10
00064
00079 union dvb_table_nit_transport_header {
00080     uint16_t bitfield;
00081     struct {
00082         uint16_t transport_length:12;
00083         uint16_t reserved:4;
00084     } __attribute__((packed));
00085 } __attribute__((packed));
00086
00109 struct dvb_table_nit_transport {
00110     uint16_t transport_id;
00111     uint16_t network_id;
00112     union {
00113         uint16_t bitfield;
00114         struct {
00115             uint16_t desc_length:12;
00116             uint16_t reserved:4;
00117         } __attribute__((packed));
00118     } __attribute__((packed));
00119     struct dvb_desc *descriptor;
00120     struct dvb_table_nit_transport *next;
00121 } __attribute__((packed));
00122
00143 struct dvb_table_nit {
00144     struct dvb_table_header header;
00145     union {
00146         uint16_t bitfield;
00147         struct {

```

```

00148             uint16_t desc_length:12;
00149             uint16_t reserved:4;
00150         } __attribute__((packed));
00151     } __attribute__((packed));
00152     struct dvb_desc *descriptor;
00153     struct dvb_table_nit_transport *transport;
00154 } __attribute__((packed));
00155
00164 typedef void nit_handler_callback_t(struct dvb_table_nit *nit,
00165                                     struct dvb_desc *desc,
00166                                     void *priv);
00167
00177 typedef void nit_tran_handler_callback_t(struct dvb_table_nit *nit,
00178                                           struct dvb_table_nit_transport *tran,
00179                                           struct dvb_desc *desc,
00180                                           void *priv);
00181
00189 #define dvb_nit_transport_foreach( _tran, _nit ) \
00190     if (_nit && _nit->transport) \
00191         for (struct dvb_table_nit_transport *_tran = _nit->transport; _tran; _tran =
00192             _tran->next) \
00193 struct dvb_v5_fe_parms;
00194
00195 #ifdef __cplusplus
00196 extern "C" {
00197 #endif
00198
00215 ssize_t dvb_table_nit_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00216                          ssize_t buflen, struct dvb_table_nit **table);
00217
00224 void dvb_table_nit_free(struct dvb_table_nit *table);
00225
00233 void dvb_table_nit_print(struct dvb_v5_fe_parms *parms, struct dvb_table_nit *table);
00234
00264 void dvb_table_nit_descriptor_handler(
00265     struct dvb_v5_fe_parms *parms,
00266     struct dvb_table_nit *table,
00267     enum descriptors descriptor,
00268     nit_handler_callback_t *call_nit,
00269     nit_tran_handler_callback_t *call_tran,
00270     void *priv);
00271
00272 #ifdef __cplusplus
00273 }
00274 #endif
00275
00276 #endif

```

9.84 lib/include/libdvbv5/pat.h File Reference

Provides the descriptors for PAT MPEG-TS table.

```

#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/header.h>

```

Data Structures

- struct [dvb_table_pat_program](#)
MPEG-TS PAT program table.
- struct [dvb_table_pat](#)
MPEG-TS PAT table.

Macros

- #define [DVB_TABLE_PAT](#)
PAT table ID.
- #define [DVB_TABLE_PAT_PID](#)
PAT Program ID.
- #define [dvb_pat_program_foreach](#)(_pgm, _pat)
Macro used to find programs on a PAT table.

Functions

- `ssize_t dvb_table_pat_init` (struct `dvb_v5_fe_parms` *parms, const `uint8_t` *buf, `ssize_t` buflen, struct `dvb_table_pat` **table)
Initializes and parses PAT table.
- `void dvb_table_pat_free` (struct `dvb_table_pat` *table)
Frees all data allocated by the PAT table parser.
- `void dvb_table_pat_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_table_pat` *table)
Prints the content of the PAT table.

9.84.1 Detailed Description

Provides the descriptors for PAT MPEG-TS table.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The table described herein is defined at:

- ISO/IEC 13818-1

See also

<http://www.etherguidesystems.com/help/sdos/mpeg/syntax/tablesections/pat.aspx>↵

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [pat.h](#).

9.85 pat.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00039 #ifndef _PAT_H
00040 #define _PAT_H
00041
00042 #include <stdint.h>
00043 #include <unistd.h> /* ssize_t */
00044
00045 #include <libdvbv5/header.h>
00046
00055 #define DVB_TABLE_PAT 0x00
00056 #define DVB_TABLE_PAT_PID 0x0000
00057
00077 struct dvb_table_pat_program {
00078     uint16_t service_id;
00079     union {
00080         uint16_t bitfield;
00081         struct {
00082             uint16_t pid:13;
00083             uint8_t reserved:3;
00084         } __attribute__((packed));
00085     } __attribute__((packed));
00086     struct dvb_table_pat_program *next;
00087 } __attribute__((packed));
00088
00108 struct dvb_table_pat {
00109     struct dvb_table_header header;
00110     uint16_t programs;
00111     struct dvb_table_pat_program *program;
00112 } __attribute__((packed));
00113
00121 #define dvb_pat_program_foreach(_pgm, _pat) \
00122     if (_pat && _pat->program) \
00123         for (struct dvb_table_pat_program *_pgm = _pat->program; _pgm; _pgm = _pgm->next) \
00124
00125 struct dvb_v5_fe_parms;
00126
00127 #ifdef __cplusplus
00128 extern "C" {
00129 #endif
00130
00147 ssize_t dvb_table_pat_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00148                             ssize_t buflen, struct dvb_table_pat **table);
00149
00156 void dvb_table_pat_free(struct dvb_table_pat *table);
00157
00165 void dvb_table_pat_print(struct dvb_v5_fe_parms *parms,
00166                          struct dvb_table_pat *table);
00167
00168 #ifdef __cplusplus
00169 }
00170 #endif
00171
00172 #endif

```

9.86 lib/include/libdvbv5/pmt.h File Reference

Provides the descriptors for PMT MPEG-TS table.


```
#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/header.h>
```

Data Structures

- struct [dvb_table_pmt_stream](#)
MPEG-TS PMT stream table.
- struct [dvb_table_pmt](#)
MPEG-TS PMT table.

Macros

- #define [DVB_TABLE_PMT](#)
PMT table ID.
- #define [dvb_pmt_field_first](#)
First field at the struct.
- #define [dvb_pmt_field_last](#)
First field that are not part of the received data.
- #define [dvb_pmt_stream_foreach](#)(_stream, _pmt)
Macro used to find streams on a PMT table.

Enumerations

- enum [dvb_streams](#) {
[stream_video](#) , [stream_video_h262](#) , [stream_audio](#) , [stream_audio_13818_3](#) ,
[stream_private_sections](#) , [stream_private_data](#) , [stream_mheg](#) , [stream_h222](#) ,
[stream_h222_1](#) , [stream_13818_6_A](#) , [stream_13818_6_B](#) , [stream_13818_6_C](#) ,
[stream_13818_6_D](#) , [stream_h222_aux](#) , [stream_audio_adts](#) , [stream_video_14496_2](#) ,
[stream_audio_latm](#) , [stream_14496_1_pes](#) , [stream_14496_1_iso](#) , [stream_download](#) ,
[stream_video_h264](#) , [stream_audio_14496_3](#) , [stream_video_hevc](#) , [stream_video_cavs](#) ,
[stream_video_moto](#) , [stream_audio_a52](#) , [stream_scte_27](#) , [stream_audio_sdds](#) ,
[stream_audio_dts_hdmv](#) , [stream_audio_e_ac3](#) , [stream_audio_dts](#) , [stream_audio_a52_vls](#) ,
[stream_spu_vls](#) , [stream_audio_sdds2](#) }
Add support for MPEG-TS Stream types.

Functions

- ssize_t [dvb_table_pmt_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, struct [dvb_table_pmt](#) **table)
Initializes and parses PMT table.
- void [dvb_table_pmt_free](#) (struct [dvb_table_pmt](#) *table)
Frees all data allocated by the PMT table parser.
- void [dvb_table_pmt_print](#) (struct [dvb_v5_fe_parms](#) *parms, const struct [dvb_table_pmt](#) *table)
Prints the content of the PAT table.

Variables

- const char * [pmt_stream_name](#) []
Converts from enum [dvb_streams](#) into a string.

9.86.1 Detailed Description

Provides the descriptors for PMT MPEG-TS table.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Andre Roth

Relevant specs

The table described herein is defined at:

- ISO/IEC 13818-1

See also

<http://www.etherguidesystems.com/help/sdos/mpeg/syntax/tablesections/pmts.↵.aspx>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [pmt.h](#).

9.86.2 Macro Definition Documentation

9.86.2.1 dvb_pmt_field_first `#define dvb_pmt_field_first`

First field at the struct.

Definition at line 237 of file [pmt.h](#).

9.86.2.2 dvb_pmt_field_last `#define dvb_pmt_field_last`

First field that are not part of the received data.

Definition at line 240 of file [pmt.h](#).

9.87 pmt.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  */
00019 */
00020
00039 #ifndef _PMT_H
00040 #define _PMT_H
00041
00042 #include <stdint.h>
00043 #include <unistd.h> /* ssize_t */
00044
00045 #include <libdvbv5/header.h>
00046
00052 #define DVB_TABLE_PMT 0x02
00053
00106 enum dvb_streams {
00107     stream_video = 0x01,
00108     stream_video_h262 = 0x02,
00109     stream_audio = 0x03,
00110     stream_audio_13818_3 = 0x04,
00111     stream_private_sections = 0x05,
00112     stream_private_data = 0x06,
00113     stream_mheg = 0x07,
00114     stream_h222 = 0x08,
00115     stream_h222_1 = 0x09,
00116     stream_13818_6_A = 0x0A,
00117     stream_13818_6_B = 0x0B,
00118     stream_13818_6_C = 0x0C,
00119     stream_13818_6_D = 0x0D,
00120     stream_h222_aux = 0x0E,
00121     stream_audio_adts = 0x0F,
00122     stream_video_14496_2 = 0x10,
00123     stream_audio_latm = 0x11,
00124     stream_14496_1_pes = 0x12,
00125     stream_14496_1_iso = 0x13,
00126     stream_download = 0x14,
00127     stream_video_h264 = 0x1b,
00128     stream_audio_14496_3 = 0x1c,
00129     stream_video_hevc = 0x24,
00130     stream_video_cavs = 0x42,
00131     stream_video_moto = 0x80,
00132     stream_audio_a52 = 0x81,
00133     stream_scte_27 = 0x82,
00134     stream_audio_sdds = 0x84,
00135     stream_audio_dts_hdmv = 0x85,
00136     stream_audio_e_ac3 = 0x87,
00137     stream_audio_dts = 0x8a,
00138     stream_audio_a52_vls = 0x91,
00139     stream_spu_vls = 0x92,
00140     stream_audio_sdds2 = 0x94,
00141 };
00142
00147 extern const char *pmt_stream_name[];
00148
00172 struct dvb_table_pmt_stream {
00173     uint8_t type;
00174     union {
00175         uint16_t bitfield;
00176         struct {
00177             uint16_t elementary_pid:13;
00178             uint16_t reserved:3;
00179         } __attribute__((packed));
00180     } __attribute__((packed));
00181     union {
00182         uint16_t bitfield2;
00183         struct {
00184             uint16_t desc_length:10;
00185             uint16_t zero:2;

```

```

00186             uint16_t reserved2:4;
00187         } __attribute__((packed));
00188     } __attribute__((packed));
00189     struct dvb_desc *descriptor;
00190     struct dvb_table_pmt_stream *next;
00191 } __attribute__((packed));
00192
00214 struct dvb_table_pmt {
00215     struct dvb_table_header header;
00216     union {
00217         uint16_t bitfield;
00218         struct {
00219             uint16_t pcr_pid:13;
00220             uint16_t reserved2:3;
00221         } __attribute__((packed));
00222     } __attribute__((packed));
00223
00224     union {
00225         uint16_t bitfield2;
00226         struct {
00227             uint16_t desc_length:10;
00228             uint16_t zero3:2;
00229             uint16_t reserved3:4;
00230         } __attribute__((packed));
00231     } __attribute__((packed));
00232     struct dvb_desc *descriptor;
00233     struct dvb_table_pmt_stream *stream;
00234 } __attribute__((packed));
00235
00237 #define dvb_pmt_field_first header
00238
00240 #define dvb_pmt_field_last descriptor
00241
00249 #define dvb_pmt_stream_foreach(_stream, _pmt) \
00250     if (_pmt && _pmt->stream) \
00251         for (struct dvb_table_pmt_stream *_stream = _pmt->stream; _stream; _stream = \
00252             _stream->next) \
00253 struct dvb_v5_fe_parms;
00254
00255 #ifdef __cplusplus
00256 extern "C" {
00257 #endif
00258
00275 ssize_t dvb_table_pmt_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00276                             ssize_t buflen, struct dvb_table_pmt **table);
00277
00284 void dvb_table_pmt_free(struct dvb_table_pmt *table);
00285
00293 void dvb_table_pmt_print(struct dvb_v5_fe_parms *parms,
00294                           const struct dvb_table_pmt *table);
00295
00296 #ifdef __cplusplus
00297 }
00298 #endif
00299
00300 #endif

```

9.88 lib/include/libdvbv5/sdt.h File Reference

Provides the descriptors for SDT MPEG-TS table.

```

#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/header.h>

```

Data Structures

- struct [dvb_table_sdt_service](#)
MPEG-TS SDT service table.
- struct [dvb_table_sdt](#)
MPEG-TS SDT table.

Macros

- `#define DVB_TABLE_SDT`
SDT table ID.
- `#define DVB_TABLE_SDT2`
SDT table ID (alternative table ID)
- `#define DVB_TABLE_SDT_PID`
SDT Program ID.
- `#define dvb_sdt_service_foreach(_service, _sdt)`
Macro used to find services on a SDT table.

Functions

- `ssize_t dvb_table_sdt_init` (struct `dvb_v5_fe_parms` *parms, const `uint8_t` *buf, `ssize_t` buflen, struct `dvb_table_sdt` **table)
Initializes and parses SDT table.
- `void dvb_table_sdt_free` (struct `dvb_table_sdt` *table)
Frees all data allocated by the SDT table parser.
- `void dvb_table_sdt_print` (struct `dvb_v5_fe_parms` *parms, struct `dvb_table_sdt` *table)
Prints the content of the SDT table.

9.88.1 Detailed Description

Provides the descriptors for SDT MPEG-TS table.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab
Andre Roth

Relevant specs

The table described herein is defined at:

- ISO/IEC 13818-1

See also

<http://www.etherguidesystems.com/Help/SDOs/dvb/syntax/tablesections/SDT.aspx>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [sdt.h](#).

9.89 sdt.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
00003  * Copyright (c) 2012 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00021 #ifndef _SDT_H
00022 #define _SDT_H
00023
00042 #include <stdint.h>
00043 #include <unistd.h> /* ssize_t */
00044
00045 #include <libdvbv5/header.h>
00046
00058 #define DVB_TABLE_SDT 0x42
00059 #define DVB_TABLE_SDT2 0x46
00060 #define DVB_TABLE_SDT_PID 0x0011
00061
00087 struct dvb_table_sdt_service {
00088     uint16_t service_id;
00089     uint8_t EIT_present_following:1;
00090     uint8_t EIT_schedule:1;
00091     uint8_t reserved:6;
00092     union {
00093         uint16_t bitfield;
00094         struct {
00095             uint16_t desc_length:12;
00096             uint16_t free_CA_mode:1;
00097             uint16_t running_status:3;
00098         } __attribute__((packed));
00099     } __attribute__((packed));
00100     struct dvb_desc *descriptor;
00101     struct dvb_table_sdt_service *next;
00102 } __attribute__((packed));
00103
00123 struct dvb_table_sdt {
00124     struct dvb_table_header header;
00125     uint16_t network_id;
00126     uint8_t reserved;
00127     struct dvb_table_sdt_service *service;
00128 } __attribute__((packed));
00129
00137 #define dvb_sdt_service_foreach(_service, _sdt) \
00138     if (_sdt && _sdt->service) \
00139         for (struct dvb_table_sdt_service *_service = _sdt->service; _service; _service = \
00140             _service->next) \
00141
00141 struct dvb_v5_fe_parms;
00142
00143 #ifdef __cplusplus
00144 extern "C" {
00145 #endif
00146
00163 ssize_t dvb_table_sdt_init (struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00164                             ssize_t buflen, struct dvb_table_sdt **table);
00165
00172 void dvb_table_sdt_free(struct dvb_table_sdt *table);
00173
00181 void dvb_table_sdt_print(struct dvb_v5_fe_parms *parms, struct dvb_table_sdt *table);
00182
00183 #ifdef __cplusplus
00184 }
00185 #endif
00186
00187 #endif

```

9.90 lib/include/libdvbv5/vct.h File Reference

Provides the descriptors for TVCT and CVCT tables.

```
#include <stdint.h>
#include <unistd.h>
#include <libdvbv5/atsc_header.h>
```

Data Structures

- struct [atsc_table_vct_channel](#)
ATSC VCT channel table (covers both CVCT and TVCT)
- struct [atsc_table_vct](#)
ATSC VCT table (covers both CVCT and TVCT)
- union [atsc_table_vct_descriptor_length](#)
ATSC VCT descriptor length.

Macros

- #define [ATSC_TABLE_TVCT](#)
TVCT table ID.
- #define [ATSC_TABLE_CVCT](#)
CVCT table ID.
- #define [ATSC_TABLE_VCT_PID](#)
Program ID with the VCT tables on it.
- #define [atsc_vct_channel_foreach](#)(_channel, _vct)
Macro used to find channels on a VCT table.

Functions

- ssize_t [atsc_table_vct_init](#) (struct [dvb_v5_fe_parms](#) *parms, const uint8_t *buf, ssize_t buflen, struct [atsc_table_vct](#) **table)
Initializes and parses VCT table.
- void [atsc_table_vct_free](#) (struct [atsc_table_vct](#) *table)
Frees all data allocated by the VCT table parser.
- void [atsc_table_vct_print](#) (struct [dvb_v5_fe_parms](#) *parms, struct [atsc_table_vct](#) *table)
Prints the content of the VCT table.

9.90.1 Detailed Description

Provides the descriptors for TVCT and CVCT tables.

Copyright

GNU Lesser General Public License version 2.1 (LGPLv2.1)

Author

Mauro Carvalho Chehab

Andre Roth

Relevant specs

The table described herein is defined at:

- ATSC A/65:2009

See also

<http://www.etherguidesystems.com/help/sdos/atsc/syntax/tablesections/TVCT.aspx>

<http://www.etherguidesystems.com/help/sdos/atsc/syntax/tablesections/CVCT.aspx>

Bug Report

Please submit bug reports and patches to linux-media@vger.kernel.org

Definition in file [vct.h](#).

9.91 vct.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2013 - Mauro Carvalho Chehab <mchehab@kernel.org>
00003  * Copyright (c) 2013 - Andre Roth <neolynx@gmail.com>
00004  *
00005  * This program is free software; you can redistribute it and/or modify
00006  * it under the terms of the GNU Lesser General Public License as published by
00007  * the Free Software Foundation version 2.1 of the License.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU Lesser General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU Lesser General Public License
00015  * along with this program; if not, write to the Free Software
00016  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
00017  * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
00018  *
00019  */
00020
00040 #ifndef _VCT_H
00041 #define _VCT_H
00042
00043 #include <stdint.h>
00044 #include <unistd.h> /* ssize_t */
00045
00046 #include <libdvbv5/atsc_header.h>
00047
00059 #define ATSC_TABLE_TVCT      0xc8
00060 #define ATSC_TABLE_CVCT      0xc9
00061 #define ATSC_TABLE_VCT_PID   0x1ffb
00062
00101 struct atsc_table_vct_channel {
00102     uint16_t        __short_name[7];
00103
00104     union {
00105         uint32_t    bitfield1;
00106         struct {
00107             uint32_t    modulation_mode:8;
00108             uint32_t    minor_channel_number:10;
00109             uint32_t    major_channel_number:10;

```



```

00110         uint32_t         reserved1:4;
00111     } __attribute__((packed));
00112 } __attribute__((packed));
00113
00114     uint32_t         carrier_frequency;
00115     uint16_t         channel_tsid;
00116     uint16_t         program_number;
00117     union {
00118         uint16_t bitfield2;
00119         struct {
00120             uint16_t         service_type:6;
00121             uint16_t         reserved2:3;
00122             uint16_t         hide_guide:1;
00123             uint16_t         out_of_band:1;          /* CVCT only */
00124             uint16_t         path_select:1;          /* CVCT only */
00125             uint16_t         hidden:1;
00126             uint16_t         access_controlled:1;
00127             uint16_t         ETM_location:2;
00128         }
00129     } __attribute__((packed));
00130 } __attribute__((packed));
00131
00132     uint16_t source_id;
00133     union {
00134         uint16_t bitfield3;
00135         struct {
00136             uint16_t descriptors_length:10;
00137             uint16_t reserved3:6;
00138         } __attribute__((packed));
00139     } __attribute__((packed));
00140
00141     /*
00142     * Everything after atsc_table_vct_channel::descriptor (including it)
00143     * won't be bit-mapped to the data parsed from the MPEG TS. So,
00144     * metadata are added there
00145     */
00146     struct dvb_desc *descriptor;
00147     struct atsc_table_vct_channel *next;
00148
00149     /* The channel_short_name is converted to locale charset by vct.c */
00150     char short_name[32];
00151 } __attribute__((packed));
00152
00153 struct atsc_table_vct {
00154     struct dvb_table_header header;
00155     uint8_t protocol_version;
00156
00157     uint8_t num_channels_in_section;
00158
00159     struct atsc_table_vct_channel *channel;
00160     struct dvb_desc *descriptor;
00161 } __attribute__((packed));
00162
00163 union atsc_table_vct_descriptor_length {
00164     uint16_t bitfield;
00165     struct {
00166         uint16_t descriptor_length:10;
00167         uint16_t reserved:6;
00168     } __attribute__((packed));
00169 } __attribute__((packed));
00170
00171 #define atsc_vct_channel_foreach(_channel, _vct) \
00172     if (_vct && _vct->channel) \
00173         for (struct atsc_table_vct_channel *_channel = _vct->channel; _channel; _channel = \
00174             _channel->next) \
00175
00176 struct dvb_v5_fe_parms;
00177
00178 #ifdef __cplusplus
00179 extern "C" {
00180 #endif
00181
00182 ssize_t atsc_table_vct_init(struct dvb_v5_fe_parms *parms, const uint8_t *buf,
00183     ssize_t buflen, struct atsc_table_vct **table);
00184 void atsc_table_vct_free(struct atsc_table_vct *table);
00185 void atsc_table_vct_print(struct dvb_v5_fe_parms *parms,
00186     struct atsc_table_vct *table);
00187
00188 #ifdef __cplusplus
00189 }
00190 #endif
00191 #endif

```

10 Example Documentation

10.1 dvbv5-scan.c

```

/*
 * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation version 2
 * of the License.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
 *
 * Based on dvbv5-tzap utility.
 */
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/time.h>
#include <argp.h>
#include <config.h>
#ifdef ENABLE_NLS
# define _(string) gettext(string)
# include "gettext.h"
# include <locale.h>
# include <langinfo.h>
# include <iconv.h>
#else
# define _(string) string
#endif
# define N_(string) string
#include <linux/dvb/dmx.h>
#include "libdvbv5/dvb-file.h"
#include "libdvbv5/dvb-demux.h"
#include "libdvbv5/dvb-dev.h"
#include "libdvbv5/dvb-v5-std.h"
#include "libdvbv5/dvb-scan.h"
#include "libdvbv5/countries.h"
#define PROGRAM_NAME "dvbv5-scan"
#define DEFAULT_OUTPUT "dvb_channel.conf"
const char *argp_program_version = PROGRAM_NAME " version " V4L_UTILS_VERSION;
const char *argp_program_bug_address = "Mauro Carvalho Chehab <mchehab@kernel.org>";
struct arguments {
    char *confname, *lnb_name, *output, *demux_dev;
    unsigned adapter, n_adapter, adapter_fe, adapter_dmx, frontend, demux, get_detected, get_nit;
    int lna, lnb, sat_number, freq_bpf;
    unsigned diseqc_wait, dont_add_new_freqs, timeout_multiply;
    unsigned other_nit;
    enum dvb_file_formats input_format, output_format;
    const char *cc;
    /* Used by status print */
    unsigned n_status_lines;
};

static const struct argp_option options[] = {
    {"adapter", 'a', N_("adapter#"), 0, N_("use given adapter (default 0)"), 0},
    {"frontend", 'f', N_("frontend#"), 0, N_("use given frontend (default 0)"), 0},
    {"demux", 'd', N_("demux#"), 0, N_("use given demux (default 0)"), 0},
    {"lnbf", 'l', N_("LNBf_type"), 0, N_("type of LNBf to use. 'help' lists the
available ones"), 0},
    {"lna", 'w', N_("LNA (0, 1, -1)"), 0, N_("enable/disable/auto LNA power"), 0},
    {"sat_number", 'S', N_("satellite_number"), 0, N_("satellite number. If not specified, disable
DISEqC"), 0},
    {"freq_bpf", 'U', N_("frequency"), 0, N_("SCR/Unicable band-pass filter frequency to
use, in kHz"), 0},
    {"wait", 'W', N_("time"), 0, N_("adds additional wait time for DISEqC command
completion"), 0},
    {"nit", 'N', NULL, 0, N_("use data from NIT table on the output file"),
0},
    {"get_frontend", 'G', NULL, 0, N_("use data from get_frontend on the output
file"), 0},
    {"verbose", 'v', NULL, 0, N_("be (very) verbose"), 0},

```

```

        {"output",      'o',      N_("file"),      0, N_("output filename (default: ") DEFAULT_OUTPUT
        ")", 0},
        {"file-freqs-only", 'F', NULL,      0, N_("don't use the other frequencies discovered
during scan"), 0},
        {"timeout-multiply", 'T', N_("factor"), 0, N_("Multiply scan timeouts by this factor"), 0},
        {"parse-other-nit", 'p', NULL,      0, N_("Parse the other NIT/SDT tables"), 0},
        {"input-format", 'I', N_("format"), 0, N_("Input format: CHANNEL, DVBV5 (default:
DVBV5)"), 0},
        {"output-format", 'O', N_("format"), 0, N_("Output format: VDR, CHANNEL, ZAP, DVBV5
(default: DVBV5)"), 0},
        {"cc",          'C',          N_("country_code"), 0, N_("Set the default country to be used (in ISO
3166-1 two letter code)"), 0},
        {"help",        '?',          0,          0,          N_("Give this help list"), -1},
        {"usage",        -3,          0,          0,          N_("Give a short usage message")},
        {"version",      'V',          0,          0,          N_("Print program version"), -1},
        { 0, 0, 0, 0, 0, 0 }
};
static int verbose = 0;
#define CHANNEL_FILE "channels.conf"
#define ERROR(x...)
do {
    fprintf(stderr, _("ERROR: "));
    fprintf(stderr, x);
    fprintf(stderr, "\n");
} while (0)
#define PERROR(x...)
do {
    fprintf(stderr, _("ERROR: "));
    fprintf(stderr, x);
    fprintf(stderr, " (%s)\n", strerror(errno));
} while (0)
static int print_frontend_stats(struct arguments *args,
                              struct dvb_v5_fe_parms *parms)
{
    char buf[512], *p;
    int rc, i, len, show;
    uint32_t status = 0;
    /* Move cursor up and cleans down */
    if (isatty(STDERR_FILENO) && args->n_status_lines)
        fprintf(stderr, "\r\x1b[%dA\x1b[J", args->n_status_lines);
    args->n_status_lines = 0;
    if (isatty(STDERR_FILENO)) {
        rc = dvb_fe_retrieve_stats(parms, DTV_STATUS, &status);
        if (rc)
            status = 0;
        if (status & FE_HAS_LOCK)
            fprintf(stderr, "\x1b[1;32m");
        else
            fprintf(stderr, "\x1b[33m");
    }
    p = buf;
    len = sizeof(buf);
    dvb_fe_snprintf_stat(parms, DTV_STATUS, NULL, 0, &p, &len, &show);
    for (i = 0; i < MAX_DTV_STATS; i++) {
        show = 1;
        dvb_fe_snprintf_stat(parms, DTV_QUALITY, _("Quality"),
                            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_SIGNAL_STRENGTH, _("Signal"),
                            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_CNR, _("C/N"),
                            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_ERROR_BLOCK_COUNT, _("UCB"),
                            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_BER, _("postBER"),
                            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_PRE_BER, _("preBER"),
                            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_PER, _("PER"),
                            i, &p, &len, &show);
        if (p != buf) {
            if (args->n_status_lines)
                fprintf(stderr, "\t%s\n", buf);
            else
                fprintf(stderr, "%s\n", buf);
            args->n_status_lines++;
            p = buf;
            len = sizeof(buf);
        }
    }
    fflush(stderr);
    return 0;
}
static int check_frontend(void *__args,
                          struct dvb_v5_fe_parms *parms)
{
    struct arguments *args = __args;
    int rc, i;

```

```

fe_status_t status;
args->n_status_lines = 0;
for (i = 0; i < args->timeout_multiply * 40; i++) {
    if (parms->abort)
        return 0;
    rc = dvb_fe_get_stats(parms);
    if (rc)
        PERROR(_("dvb_fe_get_stats failed"));
    rc = dvb_fe_retrieve_stats(parms, DTV_STATUS, &status);
    if (rc)
        status = 0;
    print_frontend_stats(args, parms);
    if (status & FE_HAS_LOCK)
        break;
    usleep(100000);
};
if (isatty(STDERR_FILENO)) {
    fprintf(stderr, "\x1b[37m");
}
return (status & FE_HAS_LOCK) ? 0 : -1;
}
static int run_scan(struct arguments *args, struct dvb_device *dvb)
{
    struct dvb_v5_fe_parms *parms = dvb->fe_parms;
    struct dvb_file *dvb_file = NULL, *dvb_file_new = NULL;
    struct dvb_entry *entry;
    struct dvb_open_descriptor *dmx_fd;
    int count = 0, shift;
    uint32_t freq, sys;
    enum dvb_sat_polarization pol;
    /* This is used only when reading old formats */
    switch (parms->current_sys) {
    case SYS_DVBT:
    case SYS_DVBS:
    case SYS_DVBC_ANNEX_A:
    case SYS_ATSC:
        sys = parms->current_sys;
        break;
    case SYS_DVBC_ANNEX_C:
        sys = SYS_DVBC_ANNEX_A;
        break;
    case SYS_DVBC_ANNEX_B:
        sys = SYS_ATSC;
        break;
    case SYS_ISDBT:
    case SYS_DTMB:
        sys = SYS_DVBT;
        break;
    default:
        sys = SYS_UNDEFINED;
        break;
    }
    dvb_file = dvb_read_file_format(args->confname, sys,
                                   args->input_format);
    if (!dvb_file)
        return -2;
    /* FIXME: should be replaced by dvb_dev_open() */
    dmx_fd = dvb_dev_open(dvb, args->demux_dev, O_RDWR);
    if (!dmx_fd) {
        perror(_("opening demux failed"));
        return -3;
    }
    for (entry = dvb_file->first_entry; entry != NULL; entry = entry->next) {
        struct dvb_v5_descriptors *dvb_scan_handler = NULL;
        uint32_t stream_id;
        /*
         * If the channel file has duplicated frequencies, or some
         * entries without any frequency at all, discard.
         */
        if (dvb_retrieve_entry_prop(entry, DTV_FREQUENCY, &freq))
            continue;
        shift = dvb_estimate_freq_shift(parms);
        if (dvb_retrieve_entry_prop(entry, DTV_POLARIZATION, &pol))
            pol = POLARIZATION_OFF;
        if (dvb_retrieve_entry_prop(entry, DTV_STREAM_ID, &stream_id))
            stream_id = NO_STREAM_ID_FILTER;
        if (!dvb_new_entry_is_needed(dvb_file->first_entry, entry,
                                     freq, shift, pol, stream_id))
            continue;
        count++;
        dvb_log(_("Scanning frequency #%d %d"), count, freq);
        /*
         * update params->lbn only if it differs from entry->lbn
         * (and "--lbnf" option was not provided),
         * to avoid linear search of LNB types for every entries.
         */
        if (!args->lbn_name && entry->lbn &&

```

```

        (!parms->lbn || strcmp(entry->lbn, parms->lbn->alias)))
        parms->lbn = dvb_sat_get_lbn(dvb_sat_search_lbn(entry->lbn));
    /*
     * Run the scanning logic
     */
    dvb_scan_handler = dvb_dev_scan(dmx_fd, entry,
                                    &check_frontend, args,
                                    args->other_nit,
                                    args->timeout_multiply);

    if (parms->abort) {
        dvb_scan_free_handler_table(dvb_scan_handler);
        break;
    }
    if (!dvb_scan_handler)
        continue;
    /*
     * Store the service entry
     */
    dvb_store_channel(&dvb_file_new, parms, dvb_scan_handler,
                    args->get_detected, args->get_nit);
    /*
     * Add new transponders based on NIT table information
     */
    if (!args->dont_add_new_freqs)
        dvb_add_scanned_transponders(parms, dvb_scan_handler,
                                     dvb_file->first_entry, entry);
    /*
     * Free the scan handler associated with the transponder
     */
    dvb_scan_free_handler_table(dvb_scan_handler);
}
if (dvb_file_new)
    dvb_write_file_format(args->output, dvb_file_new,
                        parms->current_sys, args->output_format);
dvb_file_free(dvb_file);
if (dvb_file_new)
    dvb_file_free(dvb_file_new);
dvb_dev_close(dmx_fd);
return 0;
}

static error_t parse_opt(int k, char *optarg, struct argp_state *state)
{
    struct arguments *args = state->input;
    switch (k) {
        case 'a':
            args->adapter = strtoul(optarg, NULL, 0);
            args->n_adapter++;
            break;
        case 'f':
            args->frontend = strtoul(optarg, NULL, 0);
            args->adapter_fe = args->adapter;
            break;
        case 'd':
            args->demux = strtoul(optarg, NULL, 0);
            args->adapter_dmx = args->adapter;
            break;
        case 'w':
            if (!strcmp(optarg, "on")) {
                args->lna = 1;
            } else if (!strcmp(optarg, "off")) {
                args->lna = 0;
            } else if (!strcmp(optarg, "auto")) {
                args->lna = LNA_AUTO;
            } else {
                int val = strtoul(optarg, NULL, 0);
                if (!val)
                    args->lna = 0;
                else if (val > 0)
                    args->lna = 1;
                else
                    args->lna = LNA_AUTO;
            }
            break;
        case 'l':
            args->lbn_name = optarg;
            break;
        case 'S':
            args->sat_number = strtoul(optarg, NULL, 0);
            break;
        case 'U':
            args->freq_bpf = strtoul(optarg, NULL, 0);
            break;
        case 'W':
            args->diseqc_wait = strtoul(optarg, NULL, 0);
            break;
        case 'N':
            args->get_nit++;
    }
}

```

```

        break;
    case 'G':
        args->get_detected++;
        break;
    case 'F':
        args->dont_add_new_freqs++;
        break;
    case 'p':
        args->other_nit++;
        break;
    case 'v':
        verbose++;
        break;
    case 'T':
        args->timeout_multiply = strtoul(optarg, NULL, 0);
        break;
    case 'I':
        args->input_format = dvb_parse_format(optarg);
        break;
    case 'O':
        args->output_format = dvb_parse_format(optarg);
        break;
    case 'o':
        args->output = optarg;
        break;
    case 'C':
        args->cc = strdup(optarg, 2);
        break;
    case '?':
        argp_state_help(state, state->out_stream,
                        ARGV_HELP_SHORT_USAGE | ARGV_HELP_LONG
                        | ARGV_HELP_DOC);
        fprintf(state->out_stream, _("\nReport bugs to %s.\n"), argp_program_bug_address);
        exit(0);
    case 'V':
        fprintf (state->out_stream, "%s\n", argp_program_version);
        exit(0);
    case -3:
        argp_state_help(state, state->out_stream, ARGV_HELP_USAGE);
        exit(0);
    default:
        return ARGV_ERR_UNKNOWN;
};
return 0;
}
static int *timeout_flag;
static void do_timeout(int x)
{
    (void)x;
    if (*timeout_flag == 0) {
        *timeout_flag = 1;
        alarm(5);
        signal(SIGALRM, do_timeout);
    } else {
        /* something has gone wrong ... exit */
        exit(1);
    }
}
int main(int argc, char **argv)
{
    struct arguments args = {};
    int err, lnb = -1, idx = -1;
    struct dvb_device *dvb;
    struct dvb_dev_list *dvb_dev;
    struct dvb_v5_fe_parms *parms;
    const struct argp argp = {
        .options = options,
        .parser = parse_opt,
        .doc = N_("scan DVB services using the channel file"),
        .args_doc = N_("<initial file>"),
    };
#ifdef ENABLE_NLS
    setlocale (LC_ALL, "");
    bindtextdomain (PACKAGE, LOCALEDIR);
    textdomain (PACKAGE);
#endif
    memset(&args, 0, sizeof(args));
    args.sat_number = -1;
    args.output = DEFAULT_OUTPUT;
    args.input_format = FILE_DVBV5;
    args.output_format = FILE_DVBV5;
    args.timeout_multiply = 1;
    args.adapter = (unsigned)-1;
    args.lna = LNA_AUTO;
    if (argp_parse(&argp, argc, argv, ARGV_NO_HELP | ARGV_NO_EXIT, &idx, &args)) {
        argp_help(&argp, stderr, ARGV_HELP_SHORT_USAGE, PROGRAM_NAME);
        return -1;
    }

```

```

    }
    if (args.timeout_multiply == 0)
        args.timeout_multiply = 1;
    if (args.n_adapter == 1) {
        args.adapter_fe = args.adapter;
        args.adapter_dmx = args.adapter;
    }
    if (args.lnb_name) {
        lnb = dvb_sat_search_lnb(args.lnb_name);
        if (lnb < 0) {
            printf(_("Please select one of the LNBf's below:\n"));
            dvb_print_all_lnb();
            exit(1);
        } else {
            printf(_("Using LNBf "));
            dvb_print_lnb(lnb);
        }
    }
    if (idx < argc)
        args.confname = argv[idx];
    if (!args.confname || idx < 0) {
        argp_help(&argp, stderr, ARGP_HELP_STD_HELP, PROGRAM_NAME);
        return -1;
    }
    if ((args.input_format == FILE_ZAP) ||
        (args.input_format == FILE_UNKNOWN) ||
        (args.output_format == FILE_UNKNOWN)) {
        fprintf(stderr, _("ERROR: Please specify a valid format\n"));
        argp_help(&argp, stderr, ARGP_HELP_STD_HELP, PROGRAM_NAME);
        return -1;
    }
    dvb = dvb_dev_alloc();
    if (!dvb)
        return -1;
    dvb_dev_set_log(dvb, verbose, NULL);
    dvb_dev_find(dvb, NULL, NULL);
    parms = dvb->fe_parms;
    dvb_dev = dvb_dev_seek_by_adapter(dvb, args.adapter_dmx, args.demux, DVB_DEVICE_DEMUX);
    if (!dvb_dev) {
        fprintf(stderr, _("Couldn't find demux device node\n"));
        dvb_dev_free(dvb);
        return -1;
    }
    args.demux_dev = dvb_dev->sysname;
    if (verbose)
        fprintf(stderr, _("using demux '%s'\n"), args.demux_dev);
    dvb_dev = dvb_dev_seek_by_adapter(dvb, args.adapter_fe, args.frontend,
                                     DVB_DEVICE_FRONTEND);
    if (!dvb_dev)
        return -1;
    if (!dvb_dev_open(dvb, dvb_dev->sysname, O_RDWR)) {
        free(args.demux_dev);
        return -1;
    }
    if (lnb >= 0)
        parms->lnb = dvb_sat_get_lnb(lnb);
    if (args.sat_number >= 0)
        parms->sat_number = args.sat_number;
    parms->diseqc_wait = args.diseqc_wait;
    parms->freq_bpf = args.freq_bpf;
    parms->lna = args.lna;
    err = dvb_fe_set_default_country(parms, args.cc);
    if (err < 0)
        fprintf(stderr, _("Failed to set the country code:%s\n"), args.cc);
    timeout_flag = &parms->abort;
    signal(SIGTERM, do_timeout);
    signal(SIGINT, do_timeout);
    err = run_scan(&args, dvb);
    dvb_dev_free(dvb);
    return err;
}

```

10.2 dvbv5-zap.c

```

/*
 * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation version 2
 * of the License.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of

```

```

* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
* Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
*
* Based on dvb-apps tzap utility, made by:
* Bernard Hatt 24/2/04
*/
#define _FILE_OFFSET_BITS 64
#define _LARGEFILE_SOURCE 1
#define _LARGEFILE64_SOURCE 1
/*
* Use a buffer big enough at least 1 second of data. It is interesting
* To have it multiple of a page. So, define it as a multiply of
* 4096.
*/
#define DVB_BUF_SIZE (4096 * 8 * 188)
/*
* Size of the buffer on read operations. The better is if it is
* smaller than DVB_BUF_SIZE, as we want to give more time for
* write() syscalls to be able to flush data.
*/
#define BUFLLEN (188 * 512)
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <errno.h>
#include <signal.h>
#include <argp.h>
#include <sys/time.h>
#include <time.h>
#include <config.h>
#ifdef ENABLE_NLS
# define _(string) gettext(string)
# include "gettext.h"
# include <locale.h>
# include <langinfo.h>
# include <iconv.h>
#else
# define _(string) string
#endif
# define N_(string) string
#include <linux/dvb/dmx.h>
#include "libdvbv5/dvb-file.h"
#include "libdvbv5/dvb-demux.h"
#include "libdvbv5/dvb-dev.h"
#include "libdvbv5/dvb-scan.h"
#include "libdvbv5/header.h"
#include "libdvbv5/countries.h"
#define CHANNEL_FILE "channels.conf"
#define PROGRAM_NAME "dvbv5-zap"
#ifdef O_LARGEFILE
# define O_LARGEFILE 0
#endif
const int NANO_SECONDS_IN_SEC = 1000000000;
const char *argp_program_version = PROGRAM_NAME " version " V4L_UTILS_VERSION;
const char *argp_program_bug_address = "Mauro Carvalho Chehab <mchehab@kernel.org>";
struct arguments {
    char *confname, *lbn_name, *output, *demux_dev, *dvr_dev, *dvr_fname;
    char *filename, *dvr_pipe;
    unsigned adapter, frontend, demux, get_detected, get_nit;
    int lna, lnb, sat_number;
    unsigned diseqc_wait, silent, verbose, frontend_only, freq_bpf;
    unsigned timeout, dvr, rec_psi, exit_after_tuning;
    unsigned n_apid, n_vpid, extra_pids, all_pids;
    enum dvb_file_formats input_format, output_format;
    unsigned traffic_monitor, low_traffic, non_human, port;
    char *search, *server;
    const char *cc;
    /* Used by status print */
    unsigned n_status_lines;
};
static const struct argp_option options[] = {
    {"adapter", 'a', N_("adapter#"), 0, N_("use given adapter (default 0)"), 0},
    {"audio-pid", 'A', N_("audio-pid#"), 0, N_("audio pid program to use (default 0)"), 0},
    {"channels", 'c', N_("file"), 0, N_("read channels list from 'file'"), 0},
    {"extra-pids", 'E', NULL, 0, N_("output all channel pids"), 0},
    {"demux", 'd', N_("demux#"), 0, N_("use given demux (default 0)"), 0},
    {"frontend", 'f', N_("frontend#"), 0, N_("use given frontend (default 0)"), 0},
    {"input-format", 'I', N_("format"), 0, N_("Input format: ZAP, CHANNEL, DVBV5 (default: DVBV5)"), 0},
    {"lna", 'w', N_("LNA (0, 1, -1)"), 0, N_("enable/disable/auto LNA power"), 0},

```



```

case SYS_DVBC_ANNEX_C:
    sys = SYS_DVBC_ANNEX_A;
    break;
case SYS_DVBC_ANNEX_B:
    sys = SYS_ATSC;
    break;
case SYS_ISDBT:
case SYS_DTIMB:
    sys = SYS_DVBT;
    break;
default:
    sys = SYS_UNDEFINED;
    break;
}
dvb_file = dvb_read_file_format(args->confname, sys,
                                args->input_format);
if (!dvb_file)
    return -2;
for (entry = dvb_file->first_entry; entry != NULL; entry = entry->next) {
    if (entry->channel && !strcmp(entry->channel, channel))
        break;
    if (entry->vchannel && !strcmp(entry->vchannel, channel))
        break;
}
/*
 * Give a second shot, using a case insensitive seek
 */
if (!entry) {
    for (entry = dvb_file->first_entry; entry != NULL;
         entry = entry->next) {
        if (entry->channel && !strcasecmp(entry->channel, channel))
            break;
    }
}
/*
 * When this tool is used to just tune to a channel, to monitor it or
 * to capture all PIDs, all it needs is a frequency.
 * So, let the tool to accept a frequency as the tuning channel on those
 * cases.
 * This way, a file in "channel" format can be used instead of a zap file.
 * It is also easier to use it for testing purposes.
 */
if (!entry && (!args->dvr && !args->rec_psi)) {
    uint32_t f, freq = atoi(channel);
    if (freq) {
        for (entry = dvb_file->first_entry; entry != NULL;
             entry = entry->next) {
            dvb_retrieve_entry_prop(entry, DTV_FREQUENCY, &f);
            if (f == freq)
                break;
        }
    }
}
if (!entry) {
    ERROR("Can't find channel");
    dvb_file_free(dvb_file);
    return -3;
}
/*
 * Both the DVBv5 format and the command line parameters may
 * specify the LNBf. If both have the definition, use the one
 * provided by the command line parameter, overriding the one
 * stored in the channel file.
 */
if (entry->lnb && !parms->lnb) {
    int lnb = dvb_sat_search_lnb(entry->lnb);
    if (lnb == -1) {
        ERROR("unknown LNB %s\n", entry->lnb);
        dvb_file_free(dvb_file);
        return -1;
    }
    parms->lnb = dvb_sat_get_lnb(lnb);
}
if (parms->sat_number < 0 && entry->sat_number >= 0)
    parms->sat_number = entry->sat_number;
if (entry->other_el_pid) {
    int i, type = -1;
    for (i = 0; i < entry->other_el_pid_len; i++) {
        if (type != entry->other_el_pid[i].type) {
            type = entry->other_el_pid[i].type;
            if (i)
                fprintf(stderr, "\n");
            fprintf(stderr, _("service has pid type %02x: "), type);
        }
        fprintf(stderr, " %d", entry->other_el_pid[i].pid);
    }
    fprintf(stderr, "\n");
}

```

```

    }
    /* First of all, set the delivery system */
    dvb_retrieve_entry_prop(entry, DTV_DELIVERY_SYSTEM, &sys);
    dvb_set_compat_delivery_system(parms, sys);
    /* Copy data into parms */
    for (i = 0; i < entry->n_props; i++) {
        uint32_t data = entry->props[i].u.data;
        /* Don't change the delivery system */
        if (entry->props[i].cmd == DTV_DELIVERY_SYSTEM)
            continue;
        dvb_fe_store_parm(parms, entry->props[i].cmd, data);
        if (parms->current_sys == SYS_ISDBT) {
            dvb_fe_store_parm(parms, DTV_ISDBT_PARTIAL_RECEPTION, 0);
            dvb_fe_store_parm(parms, DTV_ISDBT_SOUND_BROADCASTING, 0);
            dvb_fe_store_parm(parms, DTV_ISDBT_LAYER_ENABLED, 0x07);
            if (entry->props[i].cmd == DTV_CODE_RATE_HP) {
                dvb_fe_store_parm(parms, DTV_ISDBT_LAYERA_FEC,
                                   data);
                dvb_fe_store_parm(parms, DTV_ISDBT_LAYERB_FEC,
                                   data);
                dvb_fe_store_parm(parms, DTV_ISDBT_LAYERC_FEC,
                                   data);
            } else if (entry->props[i].cmd == DTV_MODULATION) {
                dvb_fe_store_parm(parms,
                                   DTV_ISDBT_LAYERA_MODULATION,
                                   data);
                dvb_fe_store_parm(parms,
                                   DTV_ISDBT_LAYERB_MODULATION,
                                   data);
                dvb_fe_store_parm(parms,
                                   DTV_ISDBT_LAYERC_MODULATION,
                                   data);
            }
        }
        if (parms->current_sys == SYS_ATSC &&
            entry->props[i].cmd == DTV_MODULATION) {
            if (data != VSB_8 && data != VSB_16)
                dvb_fe_store_parm(parms,
                                   DTV_DELIVERY_SYSTEM,
                                   SYS_DVBC_ANNEX_B);
        }
    }
    *out_file = dvb_file;
    *out_entry = entry;
    return 0;
}

static int setup_frontend(struct arguments *args,
                         struct dvb_v5_fe_parms *parms)
{
    int rc;
    uint32_t freq;
    if (args->silent < 2) {
        rc = dvb_fe_retrieve_parm(parms, DTV_FREQUENCY, &freq);
        if (rc < 0) {
            ERROR("can't get the frequency");
            return -1;
        }
        fprintf(stderr, _("tuning to %i Hz\n"), freq);
    }
    rc = dvb_fe_set_parms(parms);
    if (rc < 0) {
        ERROR("dvb_fe_set_parms failed");
        return -1;
    }
    return 0;
}

static void do_timeout(int x)
{
    (void)x;
    if (timeout_flag == 0) {
        timeout_flag = 1;
        alarm(2);
        signal(SIGALRM, do_timeout);
    } else {
        /* something has gone wrong ... exit */
        fprintf(stderr, "Forcing program stop due to timeout or terminate signal\n");
        exit(1);
    }
}

static int print_non_human_stats(FILE *fd, struct dvb_v5_fe_parms *parms)
{
    int rc;
    fe_status_t status;
    uint32_t snr = 0, _signal = 0, quality = 0;
    uint32_t ber = 0, per = 0, pre_ber = 0, uncorrected_blocks = 0;
    rc = dvb_fe_get_stats(parms);
    if (rc < 0) {

```

```

        ERROR("dvb_fe_get_stats failed");
        return -1;
    }
    dvb_fe_retrieve_stats(parms, DTV_STATUS, &status);
    dvb_fe_retrieve_stats(parms, DTV_QUALITY, &quality);
    dvb_fe_retrieve_stats(parms, DTV_STAT_SIGNAL_STRENGTH, &_signal);
    dvb_fe_retrieve_stats(parms, DTV_STAT_CNR, &snr);
    dvb_fe_retrieve_stats(parms, DTV_BER, &ber);
    dvb_fe_retrieve_stats(parms, DTV_STAT_ERROR_BLOCK_COUNT, &uncorrected_blocks);
    dvb_fe_retrieve_stats(parms, DTV_PRE_BER, &pre_ber);
    dvb_fe_retrieve_stats(parms, DTV_PER, &per);
    fprintf(fd, "status %02x | quality %02x | signal %04x | snr %04x | ber %08x | unc %08x | pre_ber %08x | per %08x | ",
        status, quality, _signal, snr, ber, uncorrected_blocks, pre_ber, per);
    if (status & FE_HAS_LOCK)
        fprintf(fd, "FE_HAS_LOCK");
    fprintf(fd, "\n");
    fflush(fd);
    return 0;
}

static int print_frontend_stats(FILE *fd,
                               struct arguments *args,
                               struct dvb_v5_fe_parms *parms)
{
    char buf[512], *p;
    int rc, i, len, show;
    uint32_t status = 0;
    if (args->non_human)
        return print_non_human_stats(fd, parms);
    /* Move cursor up and cleans down */
    if (isatty(fileno(fd)) && args->n_status_lines)
        fprintf(fd, "\r\x1b[%dA\x1b[J", args->n_status_lines);
    args->n_status_lines = 0;
    rc = dvb_fe_get_stats(parms);
    if (rc) {
        ERROR("dvb_fe_get_stats failed");
        return -1;
    }
    p = buf;
    len = sizeof(buf);
    dvb_fe_snprintf_stat(parms, DTV_STATUS, NULL, 0, &p, &len, &show);
    for (i = 0; i < MAX_DTV_STATS; i++) {
        show = 1;
        dvb_fe_snprintf_stat(parms, DTV_QUALITY, _("Quality"),
            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_SIGNAL_STRENGTH, _("Signal"),
            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_CNR, _("C/N"),
            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_ERROR_BLOCK_COUNT, _("UCB"),
            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_BER, _("postBER"),
            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_PRE_BER, _("preBER"),
            i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_PER, _("PER"),
            i, &p, &len, &show);
        if (p != buf) {
            if (args->n_status_lines)
                fprintf(fd, "\t%s\n", buf);
            else
                fprintf(fd, "%s\n", buf);
            args->n_status_lines++;
            p = buf;
            len = sizeof(buf);
        }
    }
    fflush(fd);
    /* While not lock, display status on a new line */
    dvb_fe_retrieve_stats(parms, DTV_STATUS, &status);
    if (!isatty(fileno(fd)) || !(status & FE_HAS_LOCK))
        fprintf(fd, "\n");
    return 0;
}

static int check_frontend(struct arguments *args,
                         struct dvb_v5_fe_parms *parms)
{
    int rc;
    fe_status_t status = 0;
    do {
        rc = dvb_fe_get_stats(parms);
        if (rc) {
            ERROR("dvb_fe_get_stats failed");
            usleep(1000000);
            continue;
        }
        status = 0;
    }
}

```

```

        rc = dvb_fe_retrieve_stats(parms, DTV_STATUS, &status);
        if (rc) {
            ERROR("dvb_fe_retrieve_stats failed");
            usleep(1000000);
            continue;
        }
        if (!args->silent)
            print_frontend_stats(stderr, args, parms);
        if (status & FE_HAS_LOCK)
            break;
        usleep(1000000);
    } while (!timeout_flag);
    if (args->silent < 2)
        print_frontend_stats(stderr, args, parms);
    return status & FE_HAS_LOCK;
}

static void get_show_stats(FILE *fp, struct arguments *args,
                           struct dvb_v5_fe_parms *parms,
                           int loop)
{
    int rc;
    args->n_status_lines = 0;
    do {
        rc = dvb_fe_get_stats(parms);
        if (!rc)
            print_frontend_stats(fp, args, parms);
        if (!timeout_flag && loop)
            usleep(1000000);
    } while (!timeout_flag && loop);
}

static struct timespec *elapsed_time(struct timespec *start)
{
    static struct timespec elapsed;
    struct timespec end;
    if (!start->tv_sec && !start->tv_nsec)
        return NULL;
    if (clock_gettime(CLOCK_MONOTONIC, &end))
        return NULL;
    elapsed.tv_sec = end.tv_sec - start->tv_sec;
    elapsed.tv_nsec = end.tv_nsec - start->tv_nsec;
    if (elapsed.tv_nsec < 0) {
        elapsed.tv_sec--;
        elapsed.tv_nsec += NANO_SECONDS_IN_SEC;
    }
    return &elapsed;
}

static void copy_to_file(struct dvb_open_descriptor *in_fd, int out_fd,
                        int timeout, int silent)
{
    char buf[BUFLLEN];
    int r, first = 1;
    long long int rc = 0LL;
    struct timespec start, *elapsed;
    /* Initialize start time, due to -EOVERFLOW with first == 1 */
    clock_gettime(CLOCK_MONOTONIC, &start);
    while (timeout_flag == 0) {
        r = dvb_dev_read(in_fd, buf, sizeof(buf));
        if (r < 0) {
            if (r == -EOVERFLOW) {
                elapsed = elapsed_time(&start);
                if (!elapsed)
                    fprintf(stderr, _("buffer overrun at %lld\n"), rc);
                else
                    fprintf(stderr, _("buffer overrun after %lld.%02ld seconds\n"),
                            (long long)elapsed->tv_sec,
                            elapsed->tv_nsec / 1000000);
                continue;
            }
            ERROR("Read failed");
            break;
        }
        /*
         * It takes a while for a DVB device to start streaming, as the
         * hardware may be waiting for some locks. The safest way to
         * ensure that a program record will have the start amount of
         * time specified by the user is to restart the timeout alarm
         * here, after the first succeeded read.
         *
         * So, let's reset the start time here.
         */
        if (first) {
            if (timeout > 0)
                alarm(timeout);
            clock_gettime(CLOCK_MONOTONIC, &start);
            first = 0;
        }
        if (write(out_fd, buf, r) < 0) {

```

```

        PERROR(_("Write failed"));
        break;
    }
    rc += r;
}
if (silent < 2) {
    if (timeout)
        fprintf(stderr, _("received %lld bytes (%lld Kbytes/sec)\n"), rc,
                rc / (1024 * timeout));
    else
        fprintf(stderr, _("received %lld bytes\n"), rc);
}
}
static error_t parse_opt(int k, char *optarg, struct argp_state *state)
{
    struct arguments *args = state->input;
    switch (k) {
    case 'a':
        args->adapter = strtoul(optarg, NULL, 0);
        break;
    case 'f':
        args->frontend = strtoul(optarg, NULL, 0);
        break;
    case 'd':
        args->demux = strtoul(optarg, NULL, 0);
        break;
    case 't':
        args->timeout = strtoul(optarg, NULL, 0);
        break;
    case 'I':
        args->input_format = dvb_parse_format(optarg);
        break;
    case 'o':
        args->filename = strdup(optarg);
        /* fall through */
    case 'r':
        args->dvr = 1;
        break;
    case 'p':
        args->rec_psi = 1;
        break;
    case 'x':
        args->exit_after_tuning = 1;
        break;
    case 'c':
        args->confname = strdup(optarg);
        break;
    case 'w':
        if (!strcasecmp(optarg, "on")) {
            args->lna = 1;
        } else if (!strcasecmp(optarg, "off")) {
            args->lna = 0;
        } else if (!strcasecmp(optarg, "auto")) {
            args->lna = LNA_AUTO;
        } else {
            int val = strtoul(optarg, NULL, 0);
            if (!val)
                args->lna = 0;
            else if (val > 0)
                args->lna = 1;
            else
                args->lna = LNA_AUTO;
        }
        break;
    case 'l':
        args->lnb_name = strdup(optarg);
        break;
    case 'S':
        args->sat_number = strtoul(optarg, NULL, 0);
        break;
    case 'U':
        args->freq_bpf = strtoul(optarg, NULL, 0);
        break;
    case 'W':
        args->diseqc_wait = strtoul(optarg, NULL, 0);
        break;
    case 's':
        args->silent++;
        break;
    case 'v':
        args->verbose++;
        break;
    case 'A':
        args->n_apid = strtoul(optarg, NULL, 0);
        break;
    case 'V':
        args->n_vpid = strtoul(optarg, NULL, 0);

```

```

        break;
    case 'E':
        args->extra_pids = 1;
        break;
    case 'P':
        args->all_pids = 1;
        break;
    case 'm':
        args->traffic_monitor = 1;
        break;
    case 'N':
        args->non_human = 1;
        break;
    case 'X':
        args->low_traffic = atoi(optarg);
        break;
    case 'L':
        args->search = strdup(optarg);
        break;
    case 'C':
        args->cc = strndup(optarg, 2);
        break;
    case 'H':
        args->server = strdup(optarg);
        break;
    case 'T':
        args->port = atoi(optarg);
        break;
    case 'D':
        args->dvr_pipe = strdup(optarg);
        break;
    case '?':
        argp_state_help(state, state->out_stream,
                        ARGP_HELP_SHORT_USAGE | ARGP_HELP_LONG
                        | ARGP_HELP_DOC);
        fprintf(state->out_stream, _("\nReport bugs to %s.\n"), argp_program_bug_address);
        exit(0);
    case -4:
        fprintf (state->out_stream, "%s\n", argp_program_version);
        exit(0);
    case -3:
        argp_state_help(state, state->out_stream, ARGP_HELP_USAGE);
        exit(0);
    default:
        return ARGP_ERR_UNKNOWN;
    };
    return 0;
}

static char *print_bytes(float val)
{
    static char buf[20];
    char *prefix = "";
    if (val >= 500 * 1024 * 1024) {
        prefix = "G";
        val /= 1024 * 1024 * 1024.;
    } else if (val >= 500 * 1024) {
        prefix = "M";
        val /= 1024 * 1024.;
    } else if (val >= 500) {
        prefix = "K";
        val /= 1024.;
    }
    if (*prefix) {
        if (snprintf(buf, sizeof(buf), "%8.3f %s", val, prefix) <= 0)
            return "NaN ";
    } else {
        if (snprintf(buf, sizeof(buf), "%9.3f ", val) <= 0)
            return "NaN ";
    }
    return buf;
}

int do_traffic_monitor(struct arguments *args, struct dvb_device *dvb,
                      int out_fd, int timeout)
{
    struct dvb_open_descriptor *fd, *dvr_fd;
    struct timespec startt;
    struct dvb_v5_fe_parms *parms = dvb->fe_parms;
    unsigned long long pidt[0x2001], wait, cont_err = 0;
    unsigned long long err_cnt[0x2000];
    signed char pid_cont[0x2000];
    int i, first = 1;
    memset(pidt, 0, sizeof(pidt));
    memset(err_cnt, 0, sizeof(err_cnt));
    memset(pid_cont, 0, sizeof(pid_cont));
    args->exit_after_tuning = 1;
    check_frontend(args, parms);
    dvr_fd = dvb_dev_open(dvb, args->dvr_dev, O_RDONLY);

```

```

if (!dvr_fd)
    return -1;
fprintf(stderr, _("dvb_dev_set_bufsize: buffer set to %d\n"), DVB_BUF_SIZE);
dvb_dev_set_bufsize(dvr_fd, DVB_BUF_SIZE);
fd = dvb_dev_open(dvb, args->demux_dev, O_RDWR);
if (!fd) {
    dvb_dev_close(dvr_fd);
    return -1;
}
if (args->silent < 2)
    fprintf(stderr, _(" dvb_set_pesfilter to 0x2000\n"));
if (dvb_dev_dmx_set_pesfilter(fd, 0x2000, DMX_PES_OTHER,
    DMX_OUT_TS_TAP, 0) < 0) {
    dvb_dev_close(dvr_fd);
    dvb_dev_close(fd);
    return -1;
}
if (clock_gettime(CLOCK_MONOTONIC, &startt)) {
    fprintf(stderr, _("Can't get timespec\n"));
    return -1;
}
wait = 1000;
monitor_log(_("%.2fs: Starting capture\n"));
while (1) {
    struct timespec *elapsed;
    unsigned char buffer[BUFLen];
    int pid, ok, diff;
    ssize_t r;
    if (timeout_flag)
        break;
    if ((r = dvb_dev_read(dvr_fd, buffer, BUFLen)) <= 0) {
        if (r == -EOverflow) {
            monitor_log(_("%.2fs: buffer overrun\n"));
            continue;
        }
        monitor_log(_("%.2fs: read() returned error %zd\n"), r);
        break;
    }
    /*
     * It takes a while for a DVB device to start streaming, as the
     * hardware may be waiting for some locks. The safest way to
     * ensure that a program record will have the start amount of
     * time specified by the user is to restart the timeout alarm
     * here, after the first succeeded read.
     */
    if (first) {
        if (timeout > 0)
            alarm(timeout);
        first = 0;
    }
    if (out_fd >= 0) {
        if (write(out_fd, buffer, r) < 0) {
            PERROR(_("Write failed"));
            break;
        }
    }
    if (r != BUFLen) {
        monitor_log(_("%.2fs: only read %zd bytes\n"), r);
        break;
    }
    for (i = 0; i < BUFLen; i += 188) {
        struct dvb_ts_packet_header *h = (void *)&buffer[i];
        if (h->sync_byte != 0x47) {
            monitor_log(_("%.2fs: invalid sync byte. Discarding %zd bytes\n"), r);
            continue;
        }
        bswap16(h->bitfield);
        /*
         * ITU-T Rec. H.222.0 decoders shall discard Transport
         * Stream packets with the adaptation_field_control
         * field set to a value of '00' (invalid). Packets with
         * a value of '01' are NULL packets. Yet, as those are
         * actually part of the stream, we won't be discarding,
         * as we want to take them into account for traffic
         * estimation purposes.
         */
        if (h->adaptation_field_control == 0)
            continue;
        ok = 1;
        pid = h->pid;
        if (pid > 0x1fff) {
            monitor_log(_("%.2fs: invalid pid: 0x%04x\n"),
                pid);
            pid = 0x1fff;
        }
    }
}

```



```

/*
 * After 1 second of processing, check if are there
 * any issues with regards to frame continuity for
 * non-NULL packets.
 *
 * According to ITU-T H.222.0 | ISO/IEC 13818-1, the
 * continuity counter isn't incremented if the packet
 * is 00 or 10. It is only incremented on odd values.
 *
 * Also, don't check continuity errors on the first
 * second, as the frontend is still starting streaming
 */
if (pid < 0x1fff && h->adaptation_field_control & 1) {
    int discontinued = 0;
    if (h->adaptation_field_control & 2) {
        if (h->adaptation_field_length >= 1) {
            discontinued = h->discontinuity_counter;
        } else {
            monitor_log(_("%2fs: pid %d has adaption layer, but size is
too small!\n"),
                                pid);
        }
    }
    if (wait < 2000)
        discontinued = 1;
    if (!discontinued && pid_cont[pid] >= 0) {
        unsigned int next = (pid_cont[pid] + 1) % 16;
        if (next != h->continuity_counter) {
            monitor_log(_("%2fs: pid %d, expecting %d received %d\n"),
                                pid, next,
                                h->continuity_counter);
            discontinued = 1;
            cont_err++;
            err_cnt[pid]++;
        }
    }
    if (discontinued)
        pid_cont[pid] = -1;
    else
        pid_cont[pid] = h->continuity_counter;
}
if (args->search) {
    int i, sl = strlen(args->search);
    ok = 0;
    if (pid != 0x1fff) {
        for (i = 0; i < (188 - sl); ++i) {
            if (!memcmp((char *)h + i, args->search, sl))
                ok = 1;
        }
    }
    if (ok) {
        pidt[pid]++;
        pidt[0x2000]++;
    }
}
elapsed = elapsed_time(&startt);
if (!elapsed)
    diff = wait;
else
    diff = (unsigned long long)elapsed->tv_sec * 1000
        + elapsed->tv_nsec * 1000 / NANO_SECONDS_IN_SEC;
if (diff > wait) {
    unsigned long long other_pidt = 0, other_err_cnt = 0;
    if (isatty(STDOUT_FILENO))
        printf("\x1b[1H\x1b[2J");
    args->n_status_lines = 0;
    printf(_("% PID          FREQ          SPEED          TOTAL\n"));
    int _pid = 0;
    for (_pid = 0; _pid < 0x2000; _pid++) {
        if (pidt[_pid]) {
            if (args->low_traffic && (pidt[_pid] * 1000. / diff) <
args->low_traffic) {
                other_pidt += pidt[_pid];
                other_err_cnt += err_cnt[_pid];
                continue;
            }
            printf("%5d %9.2f p/s %sbps ",
                    _pid,
                    pidt[_pid] * 1000. / diff,
                    print_bytes(pidt[_pid] * 1000. * 8 * 188 / diff));
            if (pidt[_pid] * 188 / 1024)
                printf("%8llu KB", (pidt[_pid] * 188 + 512) / 1024);
            else
                printf(" %8llu B", pidt[_pid] * 188);
            if (err_cnt[_pid] > 0)
                printf(" %8llu continuity errors",

```

```

        err_cnt[_pid]);
        printf("\n");
    }
}
if (other_pid) {
    printf(_("OTHER"));
    printf(" %9.2f p/s %sbps ",
           other_pid * 1000. / diff,
           print_bytes(other_pid * 1000. * 8 * 188 / diff));
    if (other_pid * 188 / 1024)
        printf("%8llu KB", (other_pid * 188 + 512) / 1024);
    else
        printf(" %8llu B", other_pid * 188);
    if (other_err_cnt > 0)
        printf(" %8llu continuity errors",
               other_err_cnt);
    printf("\n");
}
/* 0x2000 is the total traffic */
printf("TOT %11.2f p/s %sbps %8llu KB\n",
       pidt[_pid] * 1000. / diff,
       print_bytes(pidt[_pid] * 1000. * 8 * 188 / diff),
       (pidt[_pid] * 188 + 512) / 1024);
printf("\n");
get_show_stats(stdout, args, parms, 0);
wait += 1000;
if (cont_err)
    printf("CONTINUITY errors: %llu\n", cont_err);
}
}
monitor_log(_("%.2fs: Stopping capture\n"));
dvb_dev_close(dvr_fd);
dvb_dev_close(fd);
return 0;
}
static void set_signals(struct arguments *args)
{
    signal(SIGTERM, do_timeout);
    signal(SIGINT, do_timeout);
    if (args->timeout > 0) {
        signal(SIGALRM, do_timeout);
        alarm(args->timeout);
    }
}
static char *default_dvr_pipe = "/tmp/dvr-pipe";
int main(int argc, char **argv)
{
    struct arguments args = {};
    char *homedir = getenv("HOME");
    char *channel = NULL;
    int lnb = -1, idx = -1;
    int pmtpid = 0;
    struct dvb_file *dvb_file = NULL;
    const struct dvb_entry *dvb_entry = NULL;
    struct dvb_open_descriptor *pat_fd = NULL, *pmt_fd = NULL;
    struct dvb_open_descriptor *sdt_fd = NULL;
    struct dvb_open_descriptor *sid_fd = NULL, *dvr_fd = NULL;
    int file_fd = -1;
    int err = -1;
    int r, ret;
    struct dvb_v5_fe_parms *parms = NULL;
    struct dvb_device *dvb;
    struct dvb_dev_list *dvb_dev;
    const struct argp argp = {
        .options = options,
        .parser = parse_opt,
        .doc = N_("DVB zap utility"),
        .args_doc = N_("<channel name> [or <frequency> if in monitor mode]"),
    };
#ifdef ENABLE_NLS
    setlocale(LC_ALL, "");
    bindtextdomain(PACKAGE, LOCALEDIR);
    textdomain(PACKAGE);
#endif
    args.sat_number = -1;
    args.lna = LNA_AUTO;
    args.input_format = FILE_DVBV5;
    args.dvr_pipe = default_dvr_pipe;
    args.low_traffic = 1;
    if (argp_parse(&argp, argc, argv, ARGP_NO_HELP | ARGP_NO_EXIT, &idx, &args)) {
        argp_help(&argp, stderr, ARGP_HELP_SHORT_USAGE, PROGRAM_NAME);
        return -1;
    }
    if (idx < argc)
        channel = argv[idx];
    if (!channel) {
        argp_help(&argp, stderr, ARGP_HELP_STD_HELP, PROGRAM_NAME);

```

```

        return -1;
    }
    if (args.input_format == FILE_UNKNOWN) {
        ERROR("Please specify a valid format\n");
        argp_help(&argp, stderr, ARGP_HELP_STD_HELP, PROGRAM_NAME);
        return -1;
    }
    if (!args.traffic_monitor && args.search) {
        ERROR("search string can be used only on monitor mode\n");
        argp_help(&argp, stderr, ARGP_HELP_STD_HELP, PROGRAM_NAME);
        return -1;
    }
    if (args.lnb_name) {
        lnb = dvb_sat_search_lnb(args.lnb_name);
        if (lnb < 0) {
            printf(_("Please select one of the LNBf's below:\n"));
            dvb_print_all_lnb();
            exit(1);
        } else {
            printf(_("Using LNBf "));
            dvb_print_lnb(lnb);
        }
    }
    dvb = dvb_dev_alloc();
    if (!dvb)
        return -1;
    if (args.server && args.port) {
        printf(_("Connecting to %s:%d\n"), args.server, args.port);
        ret = dvb_dev_remote_init(dvb, args.server, args.port);
        if (ret < 0)
            return -1;
    }
    dvb_dev_set_log(dvb, args.verbose, NULL);
    dvb_dev_find(dvb, NULL, NULL);
    parms = dvb->fe_parms;
    dvb_dev = dvb_dev_seek_by_adapter(dvb, args.adapter, args.demux, DVB_DEVICE_DEMUX);
    if (!dvb_dev) {
        fprintf(stderr, _("Couldn't find demux device node\n"));
        dvb_dev_free(dvb);
        return -1;
    }
    args.demux_dev = dvb_dev->sysname;
    dvb_dev = dvb_dev_seek_by_adapter(dvb, args.adapter, args.demux, DVB_DEVICE_DVR);
    if (!dvb_dev) {
        fprintf(stderr, _("Couldn't find dvr device node\n"));
        dvb_dev_free(dvb);
        return -1;
    }
    args.dvr_dev = dvb_dev->sysname;
    args.dvr_fname = dvb_dev->path;
    if (args.silent < 2)
        fprintf(stderr, _("using demux '%s'\n"), args.demux_dev);
    if (!args.confname) {
        if (!homedir)
            ERROR("$HOME not set");
        r = asprintf(&args.confname, "%s/.tzap/%i/%s",
                    homedir, args.adapter, CHANNEL_FILE);
        if (access(args.confname, R_OK) {
            free(args.confname);
            r = asprintf(&args.confname, "%s/.tzap/%s",
                        homedir, CHANNEL_FILE);
        }
    }
    fprintf(stderr, _("reading channels from file '%s'\n"), args.confname);
    dvb_dev = dvb_dev_seek_by_adapter(dvb, args.adapter, args.frontend,
                                      DVB_DEVICE_FRONTEND);
    if (!dvb_dev)
        return -1;
    if (!dvb_dev_open(dvb, dvb_dev->sysname, O_RDWR))
        goto err;
    if (lnb >= 0)
        parms->lnb = dvb_sat_get_lnb(lnb);
    if (args.sat_number >= 0)
        parms->sat_number = args.sat_number;
    parms->diseqc_wait = args.diseqc_wait;
    parms->freq_bpf = args.freq_bpf;
    parms->lna = args.lna;
    r = dvb_fe_set_default_country(parms, args.cc);
    if (r < 0)
        fprintf(stderr, _("Failed to set the country code:%s\n"), args.cc);
    if (parse(&args, parms, channel, &dvb_file, &dvb_entry))
        goto err;
    if (setup_frontend(&args, parms) < 0)
        goto err;
    if (args.exit_after_tuning) {
        set_signals(&args);
        err = 0;
    }

```

```

        check_frontend(&args, parms);
        goto err;
    }
    if (args.traffic_monitor) {
        if (args.filename) {
            file_fd = open(args.filename,
                           O_LARGEFILE |
                           O_WRONLY | O_CREAT | O_TRUNC,
                           0644);
            if (file_fd < 0) {
                PERROR(_("open of '%s' failed"), args.filename);
                return -1;
            }
        }
        set_signals(&args);
        err = do_traffic_monitor(&args, dvb, file_fd, args.timeout);
        goto err;
    }
    if (args.rec_psi) {
        sid_fd = dvb_dev_open(dvb, args.demux_dev, O_RDWR);
        if (!sid_fd) {
            ERROR("opening sid demux failed");
            return -1;
        }
        pmtpid = dvb_dev_dmx_get_pmt_pid(sid_fd, dvb_entry->service_id);
        dvb_dev_close(sid_fd);
        if (pmtpid <= 0) {
            fprintf(stderr, _("couldn't find pmt-pid for sid %04x\n"),
                    dvb_entry->service_id);
            goto err;
        }
        pat_fd = dvb_dev_open(dvb, args.demux_dev, O_RDWR);
        if (!pat_fd) {
            ERROR("opening pat demux failed");
            goto err;
        }
        if (dvb_dev_dmx_set_pesfilter(pat_fd, 0, DMX_PES_OTHER,
                                     args.dvr ? DMX_OUT_TS_TAP : DMX_OUT_DECODER,
                                     args.dvr ? 64 * 1024 : 0) < 0)
            goto err;
        pmt_fd = dvb_dev_open(dvb, args.demux_dev, O_RDWR);
        if (!pmt_fd) {
            ERROR("opening pmt demux failed");
            goto err;
        }
        if (dvb_dev_dmx_set_pesfilter(pmt_fd, pmtpid, DMX_PES_OTHER,
                                     args.dvr ? DMX_OUT_TS_TAP : DMX_OUT_DECODER,
                                     args.dvr ? 64 * 1024 : 0) < 0)
            goto err;

        /*
         * SDT may also be needed in order to play some streams
         */
        sdt_fd = dvb_dev_open(dvb, args.demux_dev, O_RDWR);
        if (!sdt_fd) {
            ERROR("opening sdt demux failed");
            goto err;
        }
        if (dvb_dev_dmx_set_pesfilter(sdt_fd, 0x0011, DMX_PES_OTHER,
                                     args.dvr ? DMX_OUT_TS_TAP : DMX_OUT_DECODER,
                                     args.dvr ? 64 * 1024 : 0) < 0)
            goto err;
    }
    if (args.all_pids) {
        struct dvb_open_descriptor *video_fd = dvb_dev_open(dvb, args.demux_dev, O_RDWR);
        if (!video_fd) {
            ERROR("failed opening '%s'", args.demux_dev);
            goto err;
        }
        fprintf(stderr, _("dvb_dev_set_bufsize: buffer set to %d\n"), DVB_BUF_SIZE);
        dvb_dev_set_bufsize(video_fd, DVB_BUF_SIZE);
        if (args.silent < 2) {
            fprintf(stderr, _("pass all PIDs to TS\n"));
            fprintf(stderr, _(" dvb_set_pesfilter %d\n"), 0x2000);
        }
        if (dvb_dev_dmx_set_pesfilter(video_fd, 0x2000, DMX_PES_OTHER,
                                     DMX_OUT_TS_TAP, 0) < 0) {
            goto err;
        }
    }
    } else {
        if (dvb_entry->video_pid_len) {
            struct dvb_open_descriptor *video_fd = dvb_dev_open(dvb, args.demux_dev, O_RDWR);
            if (!video_fd) {
                ERROR("failed opening '%s'", args.demux_dev);
                goto err;
            }
            fprintf(stderr, _("dvb_dev_set_bufsize: buffer set to %d\n"), DVB_BUF_SIZE);
            dvb_dev_set_bufsize(video_fd, DVB_BUF_SIZE);

```

```

        if (args.n_vpid >= dvb_entry->video_pid_len) {
            args.n_vpid = 0;
        }
        for (int i = 0; i < dvb_entry->video_pid_len; i++) {
            if (!args.extra_pids && i != args.n_vpid) {
                continue;
            }
            if (args.silent < 2) {
                fprintf(stderr, _("video%2$s pid %1$d\n"),
                    dvb_entry->video_pid[i], i == args.n_vpid ? "" : "+");
            }
            if (dvb_dev_dmx_set_pesfilter(video_fd, dvb_entry->video_pid[i],
                i == args.n_vpid ? DMX_PES_VIDEO : DMX_PES_OTHER,
                args.dvr ? DMX_OUT_TS_TAP : DMX_OUT_DECODER,
                args.dvr ? 1024 * 1024 : 0) < 0) {
                goto err;
            }
        }
    }
    if (dvb_entry->audio_pid_len) {
        if (args.n_apid >= dvb_entry->audio_pid_len) {
            args.n_apid = 0;
        }
        for (int i = 0; i < dvb_entry->audio_pid_len; i++) {
            if (!args.extra_pids && i != args.n_apid) {
                continue;
            }
            struct dvb_open_descriptor *audio_fd = dvb_dev_open(dvb, args.demux_dev,
O_RDWR);

            if (!audio_fd) {
                ERROR("failed opening '%s'", args.demux_dev);
                goto err;
            }
            if (args.silent < 2) {
                fprintf(stderr, _("audio%2$s pid %1$d\n"),
                    dvb_entry->audio_pid[i], i == args.n_apid ? "" : "+");
            }
            if (dvb_dev_dmx_set_pesfilter(audio_fd, dvb_entry->audio_pid[i],
                i == args.n_apid ? DMX_PES_AUDIO : DMX_PES_OTHER,
                args.dvr ? DMX_OUT_TS_TAP : DMX_OUT_DECODER,
                args.dvr ? 64 * 1024 : 0) < 0) {
                goto err;
            }
        }
    }
    if (args.extra_pids && dvb_entry->other_el_pid_len) {
        for (int i = dvb_entry->other_el_pid_len - 1; i >= 0; i--) {
            struct dvb_open_descriptor * other_fd = dvb_dev_open(dvb, args.demux_dev,
O_RDWR);

            if (!other_fd) {
                ERROR("failed opening '%s'", args.demux_dev);
                goto err;
            }
            if (args.silent < 2) {
                fprintf(stderr, _("other pid %d (%d)\n"),
                    dvb_entry->other_el_pid[i].pid,
dvb_entry->other_el_pid[i].type);
            }
            if (dvb_dev_dmx_set_pesfilter(other_fd, dvb_entry->other_el_pid[i].pid,
DMX_PES_OTHER,

                args.dvr ? DMX_OUT_TS_TAP : DMX_OUT_DECODER,
                args.dvr ? 64 * 1024 : 0) < 0) {
                goto err;
            }
        }
    }
}
if ((dvb_entry->video_pid_len || dvb_entry->audio_pid_len) && !args.all_pids && !args.rec_psi) {
    printf(_("PMT record is disabled.\n"
        "Please notice that some streams can only be decoded with PMT data.\n"
        "Use '-p' option to also record PMT.\n"));
}
set_signals(&args);
if (!check_frontend(&args, parms)) {
    err = 1;
    fprintf(stderr, _("frontend doesn't lock\n"));
    goto err;
}
if (args.dvr) {
    if (args.filename) {
        file_fd = STDOUT_FILENO;
        if (strcmp(args.filename, "-") != 0) {
            file_fd = open(args.filename,
                O_LARGEFILE |
                O_WRONLY | O_CREAT | O_TRUNC,
                0644);
            if (file_fd < 0) {

```

```

        PERROR(_("open of '%s' failed"),
               args.filename);
        return -1;
    }
}

}
if (args.silent < 2)
    get_show_stats(stderr, &args, parms, 0);
if (file_fd >= 0) {
    dvr_fd = dvb_dev_open(dvb, args.dvr_dev, O_RDONLY);
    if (!dvr_fd) {
        ERROR("failed opening '%s'", args.dvr_dev);
        goto err;
    }
    if (!timeout_flag)
        fprintf(stderr, _("Record to file '%s' started\n"), args.filename);
    copy_to_file(dvr_fd, file_fd, args.timeout, args.silent);
} else if (args.server && args.port) {
    struct stat st;
    if (stat(args.dvr_pipe, &st) == -1) {
        if (mknod(args.dvr_pipe,
                  S_IRUSR | S_IWUSR | S_IFIFO, 0) < 0) {
            PERROR("Can't create pipe %s",
                   args.dvr_pipe);
            return -1;
        }
    } else {
        if (!S_ISFIFO(st.st_mode)) {
            ERROR("%s exists but is not a pipe",
                  args.dvr_pipe);
            return -1;
        }
    }
    fprintf(stderr, _("DVR pipe interface '%s' will be opened\n"), args.dvr_pipe);
    dvr_fd = dvb_dev_open(dvb, args.dvr_dev, O_RDONLY);
    if (!dvr_fd) {
        ERROR("failed opening '%s'", args.dvr_dev);
        err = -1;
        goto err;
    }
    file_fd = open(args.dvr_pipe,
                   O_LARGEFILE |
#ifdef O_WRONLY
                   O_WRONLY,
                   0644);
    if (file_fd < 0) {
        PERROR(_("open of '%s' failed"),
               args.filename);
        err = -1;
        goto err;
    }
    copy_to_file(dvr_fd, file_fd, args.timeout, args.silent);
} else {
    if (!timeout_flag)
        fprintf(stderr, _("DVR interface '%s' can now be opened\n"),
               args.dvr_fname);
    get_show_stats(stderr, &args, parms, 1);
    if (args.silent < 2)
        get_show_stats(stderr, &args, parms, 0);
} else {
    /* Wait until timeout or being killed */
    while (!timeout_flag) {
        get_show_stats(stderr, &args, parms, 1);
        usleep(1000000);
    }
}
err = 0;
err:
    dvb_dev_free(dvb);
    if (dvb_file) {
        dvb_file_free(dvb_file);
    }
    /*
     * Just to make Valgrind happier. It should be noticed
     * That, if an error happens or if the program exits via
     * timeout code at forced mode, it may not free those.
     */
    if (args.confname)
        free(args.confname);
    if (args.filename)
        free(args.filename);
    if (args.lnb_name)
        free(args.lnb_name);
    if (args.search)
        free(args.search);

```

```

    if (args.server)
        free(args.server);
    if (args.dvr_pipe != default_dvr_pipe)
        free(args.dvr_pipe);
    return err;
}

```

10.3 dvb-fe-tool.c

```

/*
 * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation version 2
 * of the License.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
 */
#include "libdvbv5/dvb-file.h"
#include "libdvbv5/dvb-dev.h"
#include <config.h>
#include <argp.h>
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#ifdef ENABLE_NLS
# define _(string) gettext(string)
# include "gettext.h"
# include <locale.h>
# include <langinfo.h>
# include <iconv.h>
#else
# define _(string) string
#endif
# define N_(string) string
#define PROGRAM_NAME "dvb-fe-tool"
const char *argp_program_version = PROGRAM_NAME " version " V4L_UTILS_VERSION;
const char *argp_program_bug_address = "Mauro Carvalho Chehab <mchehab@kernel.org>";
static const char doc[] = N_(
    "\nA DVB frontend tool using API version 5\n"
    "\nOn the options below, the arguments are:\n"
    " ADAPTER      - the dvb adapter to control\n"
    " FRONTEND      - the dvb frontend to control\n"
    " SERVER        - server address which is running the dvb5-daemon\n"
    " PORT          - server port used by the dvb5-daemon\n");
static const struct argp_option options[] = {
    {"verbose", 'v', 0, 0, N_("enables debug messages"), 0},
    {"adapter", 'a', N_("ADAPTER"), 0, N_("dvb adapter"), 0},
    {"frontend", 'f', N_("FRONTEND"), 0, N_("dvb frontend"), 0},
    {"set-delsys", 'd', N_("PARAMS"), 0, N_("set delivery system"), 0},
    {"femon", 'm', 0, 0, N_("monitors frontend stats on an streaming frontend"), 0},
    {"acoustical", 'A', 0, 0, N_("beeps if signal quality is good. Also enables femon mode. Please notice that console beep should be enabled on your wm."), 0},
    {"set", 's', N_("PARAMS"), 0, N_("set frontend"), 0},
    {"get", 'g', 0, 0, N_("get frontend"), 0},
    {"server", 'H', N_("SERVER"), 0, N_("dvbv5-daemon host IP address"), 0},
    {"tcp-port", 'T', N_("PORT"), 0, N_("dvbv5-daemon host tcp port"), 0},
    {"device-mon", 'D', 0, 0, N_("monitors device insert/removal"), 0},
    {"count", 'c', N_("COUNT"), 0, N_("samples to take (default 0 = infinite)"), 0},
    {"help", '?', 0, 0, N_("Give this help list"), -1},
    {"usage", -3, 0, 0, N_("Give a short usage message"), 0},
    {"version", 'V', 0, 0, N_("Print program version"), -1},
    {0, 0, 0, 0, 0, 0}
};
static int adapter = 0;
static int frontend = 0;
static unsigned get = 0;
static char *set_params = NULL;
static char *server = NULL;
static unsigned port = 0;

```

```

static int verbose = 0;
static int delsys = 0;
static int femon = 0;
static int acoustical = 0;
static int timeout_flag = 0;
static int device_mon = 0;
static int count = 0;
static void do_timeout(int x)
{
    (void)x;
    if (timeout_flag == 0) {
        timeout_flag = 1;
        alarm(2);
        signal(SIGALRM, do_timeout);
    } else {
        /* something has gone wrong ... exit */
        exit(1);
    }
}
#define ERROR(x...)
do {
    fprintf(stderr, _("ERROR: "));
    fprintf(stderr, x);
    fprintf(stderr, "\n");
} while (0)
static error_t parse_opt(int k, char *arg, struct argp_state *state)
{
    switch (k) {
    case 'a':
        adapter = atoi(arg);
        break;
    case 'f':
        frontend = atoi(arg);
        break;
    case 'd':
        delsys = dvb_parse_delsys(arg);
        if (delsys < 0)
            return ARGP_ERR_UNKNOWN;
        break;
    case 'm':
        femon++;
        break;
    case 'A':
        femon++;
        acoustical++;
        break;
#ifdef 0
    case 's':
        set_params = arg;
        break;
#endif
    case 'g':
        get++;
        break;
    case 'D':
        device_mon++;
        break;
    case 'H':
        server = arg;
        break;
    case 'T':
        port = atoi(arg);
        break;
    case 'v':
        verbose ++;
        break;
    case 'c':
        count = atoi(arg);
        break;
    case '?':
        argp_state_help(state, state->out_stream,
            ARGP_HELP_SHORT_USAGE | ARGP_HELP_LONG
            | ARGP_HELP_DOC);
        fprintf(state->out_stream, _("\nReport bugs to %s.\n"), argp_program_bug_address);
        exit(0);
    case 'V':
        fprintf (state->out_stream, "%s\n", argp_program_version);
        exit(0);
    case -3:
        argp_state_help(state, state->out_stream, ARGP_HELP_USAGE);
        exit(0);
    default:
        return ARGP_ERR_UNKNOWN;
    }
    return 0;
}
static struct argp argp = {

```



```

        .options = options,
        .parser = parse_opt,
        .doc = doc,
};

static int print_frontend_stats(FILE *fd,
                               struct dvb_v5_fe_parms *parms)
{
    char buf[512], *p;
    int rc, i, len, show, n_status_lines = 0;
    rc = dvb_fe_get_stats(parms);
    if (rc) {
        ERROR(_("dvb_fe_get_stats failed"));
        return -1;
    }
    p = buf;
    len = sizeof(buf);
    dvb_fe_snprintf_stat(parms, DTV_STATUS, NULL, 0, &p, &len, &show);
    for (i = 0; i < MAX_DTV_STATS; i++) {
        show = 1;
        dvb_fe_snprintf_stat(parms, DTV_QUALITY, _("Quality"),
                             i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_SIGNAL_STRENGTH, _("Signal"),
                             i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_CNR, _("C/N"),
                             i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_STAT_ERROR_BLOCK_COUNT, _("UCB"),
                             i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_BER, _("postBER"),
                             i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_PRE_BER, _("preBER"),
                             i, &p, &len, &show);
        dvb_fe_snprintf_stat(parms, DTV_PER, _("PER"),
                             i, &p, &len, &show);
        if (p != buf) {
            if (isatty(fileno(fd))) {
                enum dvb_quality qual;
                int color;
                qual = dvb_fe_retrieve_quality(parms, 0);
                switch (qual) {
                    case DVB_QUAL_POOR:
                        color = 31;
                        break;
                    case DVB_QUAL_OK:
                        color = 36;
                        break;
                    case DVB_QUAL_GOOD:
                        color = 32;
                        break;
                    case DVB_QUAL_UNKNOWN:
                    default:
                        color = 0;
                        break;
                }
                fprintf(fd, "\033[%dm", color);
                /*
                 * It would be great to change the BELL
                 * tone depending on the quality. The legacy
                 * femon used to do that, but this doesn't
                 * work anymore with modern Linux distros.
                 *
                 * So, just print a bell if quality is good.
                 *
                 * The console audio should be enabled
                 * at the window manager for this to
                 * work.
                 */
                if (acoustical) {
                    if (qual == DVB_QUAL_GOOD)
                        fprintf(fd, "\a");
                }
            }
            if (n_status_lines)
                fprintf(fd, "\t%s\n", buf);
            else
                fprintf(fd, "%s\n", buf);
            n_status_lines++;
            p = buf;
            len = sizeof(buf);
        }
    }
    fflush(fd);
    return 0;
}

static void get_show_stats(struct dvb_v5_fe_parms *parms)
{
    int rc;
    signal(SIGTERM, do_timeout);

```

```

    signal(SIGINT, do_timeout);
    do {
        rc = dvb_fe_get_stats(parms);
        if (!rc)
            print_frontend_stats(stderr, parms);
        if (count > 0 && !--count)
            break;
        if (!timeout_flag)
            usleep(1000000);
    } while (!timeout_flag);
}
static const char * const event_type[] = {
    [DVB_DEV_ADD] = "added",
    [DVB_DEV_CHANGE] = "changed",
    [DVB_DEV_REMOVE] = "removed",
};
static int dev_change_monitor(char *sysname,
                             enum dvb_dev_change_type type, void *user_priv)
{
    if (type >= ARRAY_SIZE(event_type))
        printf("unknown event on device %s\n", sysname);
    else
        printf("device %s was %s\n", sysname, event_type[type]);
    free(sysname);
    return 0;
}
int main(int argc, char *argv[])
{
    struct dvb_device *dvb;
    struct dvb_dev_list *dvb_dev;
    struct dvb_v5_fe_parms *parms;
    int ret, fe_flags = O_RDWR;
#ifdef ENABLE_NLS
    setlocale (LC_ALL, "");
    bindtextdomain (PACKAGE, LOCALEDIR);
    textdomain (PACKAGE);
#endif
    if (argp_parse(&argp, argc, argv, ARGV_NO_HELP | ARGV_NO_EXIT, 0, 0)) {
        argp_help(&argp, stderr, ARGV_HELP_SHORT_USAGE, PROGRAM_NAME);
        return -1;
    }
    /*
     * If called without any option, be verbose, to print the
     * DVB frontend information.
     */
    if (!get && !delsys && !set_params && !femon)
        verbose++;
    if (!delsys && !set_params)
        fe_flags = O_RDONLY;
    dvb = dvb_dev_alloc();
    if (!dvb)
        return -1;
    if (server && port) {
        printf(_("Connecting to %s:%d\n"), server, port);
        ret = dvb_dev_remote_init(dvb, server, port);
        if (ret < 0)
            return -1;
    }
    dvb_dev_set_log(dvb, verbose, NULL);
    if (device_mon) {
        dvb_dev_find(dvb, &dev_change_monitor, NULL);
        while (1) {
            usleep(1000000);
        }
    }
    dvb_dev_find(dvb, NULL, NULL);
    parms = dvb->fe_parms;
    dvb_dev = dvb_dev_seek_by_adapter(dvb, adapter, frontend,
                                      DVB_DEVICE_FRONTEND);
    if (!dvb_dev)
        return -1;
    if (!dvb_dev_open(dvb, dvb_dev->sysname, fe_flags))
        return -1;
    if (delsys) {
        printf(_("Changing delivery system to: %s\n"),
              delivery_system_name[delsys]);
        dvb_set_sys(parms, delsys);
        goto ret;
    }
    if 0
        if (set_params)
            do_something();
#ifdef ENABLE_NLS
    if (get) {
        dvb_fe_get_parms(parms);
        dvb_fe_prt_parms(parms);
    }

```

```

        if (femon)
            get_show_stats(parms);
ret:
    dvb_dev_free(dvb);
    return 0;
}

```

10.4 dvb-format-convert.c

```

/*
 * Copyright (c) 2011-2012 - Mauro Carvalho Chehab
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation version 2
 * of the License.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 * Or, point your browser to http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
 */
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/time.h>
#include <argp.h>
#include <config.h>
#ifdef ENABLE_NLS
# define _(string) gettext(string)
# include "gettext.h"
# include <locale.h>
# include <langinfo.h>
# include <iconv.h>
#else
# define _(string) string
#endif
# define N_(string) string
#include "libdvbv5/dvb-file.h"
#include "libdvbv5/dvb-demux.h"
#include "libdvbv5/dvb-scan.h"
#define PROGRAM_NAME "dvb-format-convert"
struct arguments {
    char *input_file, *output_file;
    enum dvb_file_formats input_format, output_format;
    int delsys;
};

static const struct argp_option options[] = {
    {"input-format", 'I', N_("format"), 0, N_("Valid input formats: ZAP, CHANNEL, DVBV5"),
    0},
    {"output-format", 'O', N_("format"), 0, N_("Valid output formats: VDR, ZAP, CHANNEL, DVBV5"), 0},
    {"delsys", 's', N_("system"), 0, N_("Delivery system type. Needed if input or output format is ZAP"), 0},
    {"help", '?', 0, 0, N_("Give this help list"), -1},
    {"usage", -3, 0, 0, N_("Give a short usage message")},
    {"version", 'V', 0, 0, N_("Print program version"), -1},
    {0, 0, 0, 0, 0, 0}
};

const char *argp_program_version = PROGRAM_NAME " version " V4L_UTILS_VERSION;
const char *argp_program_bug_address = "Mauro Carvalho Chehab <mchehab@kernel.org>";
static error_t parse_opt(int k, char *optarg, struct argp_state *state)
{
    struct arguments *args = state->input;
    switch (k) {
        case 'I':
            args->input_format = dvb_parse_format(optarg);
            break;
        case 'O':
            args->output_format = dvb_parse_format(optarg);
            break;
        case 's':
            args->delsys = dvb_parse_delsys(optarg);
    }
}

```

```

        break;
    case '?':
        argp_state_help(state, state->out_stream,
                        ARGP_HELP_SHORT_USAGE | ARGP_HELP_LONG
                        | ARGP_HELP_DOC);
        fprintf(state->out_stream, _("\nReport bugs to %s.\n"), argp_program_bug_address);
        exit(0);
    case 'v':
        fprintf (state->out_stream, "%s\n", argp_program_version);
        exit(0);
    case -3:
        argp_state_help(state, state->out_stream, ARGP_HELP_USAGE);
        exit(0);
    default:
        return ARGP_ERR_UNKNOWN;
    };
    return 0;
}

static int convert_file(struct arguments *args)
{
    struct dvb_file *dvb_file = NULL;
    int ret;
    printf(_("Reading file %s\n"), args->input_file);
    dvb_file = dvb_read_file_format(args->input_file, args->delsys,
                                    args->input_format);

    if (!dvb_file) {
        fprintf(stderr, _("Error reading file %s\n"), args->input_file);
        return -1;
    }
    printf(_("Writing file %s\n"), args->output_file);
    ret = dvb_write_file_format(args->output_file, dvb_file,
                                args->delsys, args->output_format);
    dvb_file_free(dvb_file);
    return ret;
}

int main(int argc, char **argv)
{
    struct arguments args = {};
    int idx = -1, missing = 0;
    const struct argp argp = {
        .options = options,
        .parser = parse_opt,
        .doc = N_("scan DVB services using the channel file"),
        .args_doc = N_("<input file> <output file>"),
    };
#ifdef ENABLE-NLS
    setlocale (LC_ALL, "");
    bindtextdomain (PACKAGE, LOCALEDIR);
    textdomain (PACKAGE);
#endif
    memset(&args, 0, sizeof(args));
    argp_parse(&argp, argc, argv, ARGP_NO_HELP | ARGP_NO_EXIT, &idx, &args);
    if (idx + 1 < argc) {
        args.input_file = argv[idx];
        args.output_file = argv[idx + 1];
    }
    if (args.input_format == FILE_UNKNOWN) {
        fprintf(stderr, _("ERROR: Please specify a valid input format\n"));
        missing = 1;
    }
    else if (!args.input_file) {
        fprintf(stderr, _("ERROR: Please specify a valid input file\n"));
        missing = 1;
    }
    else if (args.output_format == FILE_UNKNOWN) {
        fprintf(stderr, _("ERROR: Please specify a valid output format\n"));
        missing = 1;
    }
    else if (!args.output_file) {
        fprintf(stderr, _("ERROR: Please specify a valid output file\n"));
        missing = 1;
    }
    else if ((args.input_format == FILE_ZAP) ||
              (args.output_format == FILE_ZAP)) &&
              (args.delsys <= 0 || args.delsys == SYS_ISDBS)) {
        fprintf(stderr, _("ERROR: Please specify a valid delivery system for ZAP format\n"));
        missing = 1;
    }
    if (missing) {
        argp_help(&argp, stderr, ARGP_HELP_STD_HELP, PROGRAM_NAME);
        return -1;
    }
    return convert_file(&args);
}

```


Index

- __pad0__
 - atsc_table_eit_event, [150](#)
- __short_name
 - atsc_table_vct_channel, [159](#)
- AAC_descriptor
 - Digital TV table parsing, [89](#)
- abort
 - dvb_v5_fe_parms, [276](#)
- AC_3_descriptor
 - Digital TV table parsing, [88](#)
- access_controlled
 - atsc_table_vct_channel, [160](#)
- AD
 - Ancillary functions and macros, [56](#)
- adaptation_field
 - dvb_mpeg_ts, [232](#)
- adaptation_field_control
 - dvb_ts_packet_header, [268](#)
- adaptation_field_data_descriptor
 - Digital TV table parsing, [88](#)
- adaptation_field_length
 - dvb_ts_packet_header, [268](#)
- adaption
 - dvb_mpeg_ts, [232](#)
- additional_copy_info
 - dvb_mpeg_pes_optional, [228](#)
- additional_identification_info
 - dvb_desc_registration, [187](#)
- AE
 - Ancillary functions and macros, [56](#)
- AF
 - Ancillary functions and macros, [56](#)
- AG
 - Ancillary functions and macros, [56](#)
- AI
 - Ancillary functions and macros, [56](#)
- AL
 - Ancillary functions and macros, [56](#)
- alias
 - dvb_sat_lnb, [240](#)
- allow_section_gaps
 - dvb_table_filter, [247](#)
- AM
 - Ancillary functions and macros, [56](#)
- Ancillary functions and macros, [53](#)
 - AD, [56](#)
 - AE, [56](#)
 - AF, [56](#)
 - AG, [56](#)
 - AI, [56](#)
 - AL, [56](#)
 - AM, [56](#)
 - AO, [56](#)
 - AQ, [56](#)
 - AR, [56](#)
 - ARRAY_SIZE, [55](#)
 - AS, [56](#)
 - AT, [56](#)
 - atsc_time, [61](#)
 - AU, [56](#)
 - AW, [56](#)
 - AX, [56](#)
 - AZ, [57](#)
 - BA, [57](#)
 - BB, [57](#)
 - BD, [57](#)
 - BE, [57](#)
 - BF, [57](#)
 - BG, [57](#)
 - BH, [57](#)
 - BI, [57](#)
 - BJ, [57](#)
 - BL, [57](#)
 - BM, [57](#)
 - BN, [57](#)
 - BO, [57](#)
 - BQ, [57](#)
 - BR, [57](#)
 - BS, [57](#)
 - BT, [57](#)
 - BV, [57](#)
 - BW, [57](#)
 - BY, [57](#)
 - BZ, [57](#)
 - CA, [57](#)
 - CC, [57](#)
 - CD, [57](#)
 - CF, [57](#)
 - CG, [57](#)
 - CH, [57](#)
 - CI, [57](#)
 - CK, [57](#)
 - CL, [57](#)
 - CM, [57](#)
 - CN, [57](#)
 - CO, [57](#)
 - COUNTRY_UNKNOWN, [56](#)
 - CR, [57](#)
 - CU, [57](#)
 - CV, [57](#)
 - CW, [57](#)
 - CX, [57](#)
 - CY, [57](#)
 - CZ, [57](#)
 - DE, [57](#)
 - DJ, [57](#)
 - DK, [57](#)
 - DM, [57](#)
 - DO, [57](#)

dvb_country_a2_to_id, [63](#)
dvb_country_a3_to_id, [63](#)
dvb_country_t, [56](#)
dvb_country_to_2letters, [63](#)
dvb_country_to_3letters, [64](#)
dvb_country_to_name, [64](#)
dvb_crc32, [64](#)
dvb_default_log, [65](#)
dvb_guess_user_country, [65](#)
dvb_logfunc, [56](#)
DZ, [57](#)
EC, [57](#)
EE, [57](#)
EG, [58](#)
EH, [58](#)
ER, [58](#)
ES, [58](#)
ET, [58](#)
FI, [58](#)
FJ, [58](#)
FK, [58](#)
FM, [58](#)
FO, [58](#)
FR, [58](#)
GA, [58](#)
GB, [58](#)
GD, [58](#)
GE, [58](#)
GF, [58](#)
GG, [58](#)
GH, [58](#)
GI, [58](#)
GL, [58](#)
GM, [58](#)
GN, [58](#)
GP, [58](#)
GQ, [58](#)
GR, [58](#)
GS, [58](#)
GT, [58](#)
GU, [58](#)
GW, [58](#)
GY, [58](#)
HK, [58](#)
HM, [58](#)
HN, [58](#)
HR, [58](#)
HT, [58](#)
HU, [58](#)
ID, [58](#)
IE, [58](#)
IL, [58](#)
IM, [58](#)
IN, [58](#)
IO, [58](#)
IQ, [58](#)
IR, [58](#)
IS, [58](#)
IT, [58](#)
JE, [58](#)
JM, [58](#)
JO, [58](#)
JP, [59](#)
KE, [59](#)
KG, [59](#)
KH, [59](#)
KI, [59](#)
KM, [59](#)
KN, [59](#)
KP, [59](#)
KR, [59](#)
KW, [59](#)
KY, [59](#)
KZ, [59](#)
LA, [59](#)
LB, [59](#)
LC, [59](#)
LI, [59](#)
LK, [59](#)
LR, [59](#)
LS, [59](#)
LT, [59](#)
LU, [59](#)
LV, [59](#)
LY, [59](#)
MA, [59](#)
MC, [59](#)
MD, [59](#)
ME, [59](#)
MF, [59](#)
MG, [59](#)
MH, [59](#)
MK, [59](#)
ML, [59](#)
MM, [59](#)
MN, [59](#)
MO, [59](#)
MP, [59](#)
MQ, [59](#)
MR, [59](#)
MS, [59](#)
MT, [59](#)
MU, [59](#)
MV, [59](#)
MW, [59](#)
MX, [59](#)
MY, [59](#)
MZ, [59](#)
NA, [59](#)
NC, [59](#)
NE, [59](#)
NF, [60](#)
NG, [60](#)
NI, [60](#)
NL, [60](#)
NO, [60](#)

- NP, [60](#)
- NR, [60](#)
- NU, [60](#)
- NZ, [60](#)
- OM, [60](#)
- PA, [60](#)
- PE, [60](#)
- PF, [60](#)
- PG, [60](#)
- PH, [60](#)
- PK, [60](#)
- PL, [60](#)
- PM, [60](#)
- PN, [60](#)
- PR, [60](#)
- PS, [60](#)
- PT, [60](#)
- PW, [60](#)
- PY, [60](#)
- QA, [60](#)
- RE, [60](#)
- RO, [60](#)
- RS, [60](#)
- RU, [60](#)
- RW, [60](#)
- SA, [60](#)
- SB, [60](#)
- SC, [60](#)
- SD, [60](#)
- SE, [60](#)
- SG, [60](#)
- SH, [60](#)
- SI, [60](#)
- SJ, [60](#)
- SK, [60](#)
- SL, [60](#)
- SM, [60](#)
- SN, [60](#)
- SO, [60](#)
- SR, [60](#)
- SS, [60](#)
- ST, [60](#)
- SV, [60](#)
- SX, [60](#)
- SY, [61](#)
- SZ, [61](#)
- TC, [61](#)
- TD, [61](#)
- TF, [61](#)
- TG, [61](#)
- TH, [61](#)
- TJ, [61](#)
- TK, [61](#)
- TL, [61](#)
- TM, [61](#)
- TN, [61](#)
- TO, [61](#)
- TR, [61](#)
- TT, [61](#)
- TV, [61](#)
- TW, [61](#)
- TZ, [61](#)
- UA, [61](#)
- UG, [61](#)
- UM, [61](#)
- US, [61](#)
- UY, [61](#)
- UZ, [61](#)
- VA, [61](#)
- VC, [61](#)
- VE, [61](#)
- VG, [61](#)
- VI, [61](#)
- VN, [61](#)
- VU, [61](#)
- WF, [61](#)
- WS, [61](#)
- YE, [61](#)
- YT, [61](#)
- ZA, [61](#)
- ZM, [61](#)
- ZW, [61](#)
- ancillary_data_descriptor
 - Digital TV table parsing, [88](#)
- announcement_support_descriptor
 - Digital TV table parsing, [88](#)
- AO
 - Ancillary functions and macros, [56](#)
- application_signalling_descriptor
 - Digital TV table parsing, [88](#)
- AQ
 - Ancillary functions and macros, [56](#)
- AR
 - Ancillary functions and macros, [56](#)
- area_code
 - isdbt_desc_terrestrial_delivery_system, [283](#)
- ARRAY_SIZE
 - Ancillary functions and macros, [55](#)
- AS
 - Ancillary functions and macros, [56](#)
- aspect
 - dvb_mpeg_es_seq_start, [223](#)
- association_tag_descriptor
 - Digital TV table parsing, [89](#)
- AT
 - Ancillary functions and macros, [56](#)
- atsc_ac3_audio_descriptor
 - Digital TV table parsing, [90](#)
- atsc_ATSC_private_information_descriptor
 - Digital TV table parsing, [90](#)
- ATSC_BASE_PID
 - Digital TV table parsing, [73](#)
- atsc_caption_service_descriptor
 - Digital TV table parsing, [90](#)
- atsc_component_name_descriptor
 - Digital TV table parsing, [90](#)

- atsc_content_advisory_descriptor
 - Digital TV table parsing, [90](#)
- atsc_DCC_arriving_request_descriptor
 - Digital TV table parsing, [90](#)
- atsc_DCC_departing_request_descriptor
 - Digital TV table parsing, [90](#)
- atsc_desc_service_location, [143](#)
 - bitfield, [144](#)
 - elementary, [144](#)
 - length, [144](#)
 - next, [144](#)
 - number_elements, [144](#)
 - pcr_pid, [145](#)
 - reserved, [145](#)
 - type, [145](#)
- atsc_desc_service_location_elementary, [145](#)
 - bitfield, [146](#)
 - elementary_pid, [146](#)
 - ISO_639_language_code, [146](#)
 - reserved, [146](#)
 - stream_type, [146](#)
- atsc_desc_service_location_free
 - Parsers for several MPEG-TS descriptors, [112](#)
- atsc_desc_service_location_init
 - Parsers for several MPEG-TS descriptors, [113](#)
- atsc_desc_service_location_print
 - Parsers for several MPEG-TS descriptors, [113](#)
- atsc_eit_event_foreach
 - Digital TV table parsing, [73](#)
- atsc_extended_channel_descriptor
 - Digital TV table parsing, [90](#)
- atsc_genre_descriptor
 - Digital TV table parsing, [90](#)
- atsc_mgt_table_foreach
 - mgt.h, [384](#)
- atsc_redistribution_control_descriptor
 - Digital TV table parsing, [90](#)
- atsc_service_location_descriptor
 - Digital TV table parsing, [90](#)
- atsc_stuffing_descriptor
 - Digital TV table parsing, [90](#)
- ATSC_TABLE_CVCT
 - Digital TV table parsing, [74](#)
- ATSC_TABLE_EIT
 - Digital TV table parsing, [74](#)
- atsc_table_eit, [147](#)
 - event, [147](#)
 - events, [147](#)
 - header, [147](#)
 - protocol_version, [148](#)
- atsc_table_eit_desc_length, [148](#)
 - bitfield, [149](#)
 - desc_length, [149](#)
 - reserved, [149](#)
- atsc_table_eit_event, [149](#)
 - __pad0__, [150](#)
 - bitfield, [150](#)
 - bitfield2, [150](#)
 - descriptor, [151](#)
 - duration, [151](#)
 - etm, [151](#)
 - event_id, [151](#)
 - next, [151](#)
 - one, [151](#)
 - one2, [151](#)
 - source_id, [151](#)
 - start, [152](#)
 - start_time, [152](#)
 - title_length, [152](#)
- atsc_table_eit_free
 - Digital TV table parsing, [92](#)
- atsc_table_eit_init
 - Digital TV table parsing, [92](#)
- atsc_table_eit_print
 - Digital TV table parsing, [92](#)
- ATSC_TABLE_MGT
 - Digital TV table parsing, [74](#)
- atsc_table_mgt, [152](#)
 - descriptor, [153](#)
 - header, [153](#)
 - protocol_version, [153](#)
 - table, [153](#)
 - tables, [153](#)
- atsc_table_mgt_free
 - Digital TV table parsing, [93](#)
- atsc_table_mgt_init
 - Digital TV table parsing, [93](#)
- atsc_table_mgt_print
 - Digital TV table parsing, [93](#)
- atsc_table_mgt_table, [154](#)
 - bitfield, [155](#)
 - bitfield2, [155](#)
 - desc_length, [155](#)
 - descriptor, [155](#)
 - next, [155](#)
 - one, [155](#)
 - one2, [155](#)
 - one3, [156](#)
 - pid, [156](#)
 - size, [156](#)
 - type, [156](#)
 - type_version, [156](#)
- ATSC_TABLE_TVCT
 - Digital TV table parsing, [74](#)
- atsc_table_vct, [156](#)
 - channel, [157](#)
 - descriptor, [157](#)
 - header, [157](#)
 - num_channels_in_section, [157](#)
 - protocol_version, [157](#)
- atsc_table_vct_channel, [158](#)
 - __short_name, [159](#)
 - access_controlled, [160](#)
 - bitfield1, [160](#)
 - bitfield2, [160](#)
 - bitfield3, [160](#)

- carrier_frequency, [160](#)
- channel_tsid, [160](#)
- descriptor, [160](#)
- descriptors_length, [160](#)
- ETM_location, [161](#)
- hidden, [161](#)
- hide_guide, [161](#)
- major_channel_number, [161](#)
- minor_channel_number, [161](#)
- modulation_mode, [161](#)
- next, [161](#)
- out_of_band, [161](#)
- path_select, [162](#)
- program_number, [162](#)
- reserved1, [162](#)
- reserved2, [162](#)
- reserved3, [162](#)
- service_type, [162](#)
- short_name, [162](#)
- source_id, [162](#)
- atsc_table_vct_descriptor_length, [163](#)
 - bitfield, [163](#)
 - descriptor_length, [163](#)
 - reserved, [164](#)
- atsc_table_vct_free
 - Digital TV table parsing, [95](#)
- atsc_table_vct_init
 - Digital TV table parsing, [95](#)
- ATSC_TABLE_VCT_PID
 - Digital TV table parsing, [74](#)
- atsc_table_vct_print
 - Digital TV table parsing, [95](#)
- atsc_time
 - Ancillary functions and macros, [61](#)
- atsc_time_shifted_service_descriptor
 - Digital TV table parsing, [90](#)
- atsc_vct_channel_foreach
 - Digital TV table parsing, [74](#)
- AU
 - Ancillary functions and macros, [56](#)
- audio_component_descriptor
 - Digital TV table parsing, [89](#)
- audio_pid
 - dvb_entry, [212](#)
- audio_pid_len
 - dvb_entry, [212](#)
- audio_stream_descriptor
 - Digital TV table parsing, [86](#)
- audio_type
 - dvb_desc_language, [182](#)
- AVC_timing_and_HRD_descriptor
 - Digital TV table parsing, [87](#)
- AVC_video_descriptor
 - Digital TV table parsing, [87](#)
- AW
 - Ancillary functions and macros, [56](#)
- AX
 - Ancillary functions and macros, [56](#)
- AZ
 - Ancillary functions and macros, [57](#)
- BA
 - Ancillary functions and macros, [57](#)
- bandwidth
 - dvb_desc_t2_delivery, [193](#)
 - dvb_desc_terrestrial_delivery, [200](#)
- basic_local_event_descriptor
 - Digital TV table parsing, [89](#)
- BB
 - Ancillary functions and macros, [57](#)
- BD
 - Ancillary functions and macros, [57](#)
- BE
 - Ancillary functions and macros, [57](#)
- BF
 - Ancillary functions and macros, [57](#)
- BG
 - Ancillary functions and macros, [57](#)
- BH
 - Ancillary functions and macros, [57](#)
- BI
 - Ancillary functions and macros, [57](#)
- bitfield
 - atsc_desc_service_location, [144](#)
 - atsc_desc_service_location_elementary, [146](#)
 - atsc_table_eit_desc_length, [149](#)
 - atsc_table_eit_event, [150](#)
 - atsc_table_mgt_table, [155](#)
 - atsc_table_vct_descriptor_length, [163](#)
 - dvb_desc_frequency_list, [178](#)
 - dvb_desc_logical_channel_number, [184](#)
 - dvb_desc_sat, [189](#)
 - dvb_desc_t2_delivery, [193](#)
 - dvb_desc_ts_info, [203](#)
 - dvb_mpeg_es_pic_start, [221](#)
 - dvb_mpeg_es_seq_start, [224](#)
 - dvb_mpeg_pes, [226](#)
 - dvb_mpeg_pes_optional, [228](#)
 - dvb_mpeg_ts, [232](#)
 - dvb_table_header, [249](#)
 - dvb_table_nit, [251](#)
 - dvb_table_nit_transport, [253](#)
 - dvb_table_nit_transport_header, [255](#)
 - dvb_table_pat_program, [257](#)
 - dvb_table_pmt, [259](#)
 - dvb_table_pmt_stream, [262](#)
 - dvb_table_sdt_service, [265](#)
 - dvb_ts_packet_header, [269](#)
 - isdbt_desc_terrestrial_delivery_system, [283](#)
 - ts_t, [285](#)
- bitfield1
 - atsc_table_vct_channel, [160](#)
 - dvb_desc_ca, [166](#)
 - dvb_desc_cable_delivery, [170](#)
 - dvb_table_eit_event, [245](#)
- bitfield2
 - atsc_table_eit_event, [150](#)

- atsc_table_mgt_table, [155](#)
- atsc_table_vct_channel, [160](#)
- dvb_desc_cable_delivery, [170](#)
- dvb_mpeg_es_pic_start, [221](#)
- dvb_mpeg_es_seq_start, [224](#)
- dvb_table_eit_event, [245](#)
- dvb_table_pmt, [259](#)
- dvb_table_pmt_stream, [262](#)
- ts_t, [285](#)
- bitfield3
 - atsc_table_vct_channel, [160](#)
 - dvb_mpeg_es_seq_start, [224](#)
- bitrate
 - dvb_mpeg_es_seq_start, [224](#)
- bits00
 - ts_t, [285](#)
- bits15
 - ts_t, [285](#)
- bits30
 - ts_t, [285](#)
- BJ
 - Ancillary functions and macros, [57](#)
- BL
 - Ancillary functions and macros, [57](#)
- BM
 - Ancillary functions and macros, [57](#)
- BN
 - Ancillary functions and macros, [57](#)
- BO
 - Ancillary functions and macros, [57](#)
- board_information_descriptor
 - Digital TV table parsing, [90](#)
- bouquet_name_descriptor
 - Digital TV table parsing, [87](#)
- BQ
 - Ancillary functions and macros, [57](#)
- BR
 - Ancillary functions and macros, [57](#)
- broadcaster_Name_Descriptor
 - Digital TV table parsing, [90](#)
- BS
 - Ancillary functions and macros, [57](#)
- BT
 - Ancillary functions and macros, [57](#)
- bus_addr
 - dvb_dev_list, [207](#)
- bus_id
 - dvb_dev_list, [207](#)
- BV
 - Ancillary functions and macros, [57](#)
- BW
 - Ancillary functions and macros, [57](#)
- BY
 - Ancillary functions and macros, [57](#)
- BZ
 - Ancillary functions and macros, [57](#)
- CA
 - Ancillary functions and macros, [57](#)
- CA_contract_information_descriptor
 - Digital TV table parsing, [89](#)
- CA_EMM_TS_descriptor
 - Digital TV table parsing, [89](#)
- ca_id
 - dvb_desc_ca, [166](#)
- CA_identifier_descriptor
 - Digital TV table parsing, [87](#)
- ca_pid
 - dvb_desc_ca, [166](#)
- CA_service_descriptor
 - Digital TV table parsing, [89](#)
- cable_delivery_system_descriptor
 - Digital TV table parsing, [87](#)
- caid_count
 - dvb_desc_ca_identifier, [168](#)
- caids
 - dvb_desc_ca_identifier, [168](#)
- carousel_compatible_composite_descriptor
 - Digital TV table parsing, [90](#)
- carousel_id_descriptor
 - Digital TV table parsing, [89](#)
- carrier_frequency
 - atsc_table_vct_channel, [160](#)
- cat.h
 - dvb_table_cat_free, [291](#)
 - dvb_table_cat_init, [291](#)
 - dvb_table_cat_print, [292](#)
- CC
 - Ancillary functions and macros, [57](#)
- CD
 - Ancillary functions and macros, [57](#)
- cell
 - dvb_desc_t2_delivery, [193](#)
- cell_frequency_link_descriptor
 - Digital TV table parsing, [88](#)
- cell_id
 - dvb_desc_t2_delivery_cell, [196](#)
- cell_id_extension
 - dvb_desc_t2_delivery_subcell, [198](#)
 - dvb_desc_t2_delivery_subcell_old, [199](#)
- cell_list_descriptor
 - Digital TV table parsing, [88](#)
- centre_frequency
 - dvb_desc_t2_delivery, [193](#)
 - dvb_desc_t2_delivery_cell, [196](#)
 - dvb_desc_terrestrial_delivery, [200](#)
- CF
 - Ancillary functions and macros, [57](#)
- CG
 - Ancillary functions and macros, [57](#)
- CH
 - Ancillary functions and macros, [57](#)
- channel
 - atsc_table_vct, [157](#)
 - dvb_desc_hierarchy, [180](#)
 - dvb_entry, [213](#)
- Channel and transponder file read/write, [135](#)

- channel_file_format, 143
- channel_file_zap_format, 143
- dvb_file_formats, 136
- dvb_file_free, 137
- dvb_parse_delsys, 137
- dvb_parse_format, 138
- dvb_parse_format_online, 138
- dvb_read_file, 139
- dvb_read_file_format, 139
- dvb_retrieve_entry_prop, 139
- dvb_store_channel, 140
- dvb_store_entry_prop, 141
- dvb_write_file, 141
- dvb_write_file_format, 142
- dvb_write_format_online, 142
- dvb_write_format_vdr, 142
- FILE_CHANNEL, 137
- FILE_DVBV5, 137
- FILE_UNKNOWN, 137
- FILE_VDR, 137
- FILE_ZAP, 137
- channel_file_format
 - Channel and transponder file read/write, 143
- channel_file_zap_format
 - Channel and transponder file read/write, 143
- channel_tsid
 - atsc_table_vct_channel, 160
- check_frontend_t
 - Digital TV frontend scan, 44
- CI
 - Ancillary functions and macros, 57
- CK
 - Ancillary functions and macros, 57
- CL
 - Ancillary functions and macros, 57
- CM
 - Ancillary functions and macros, 57
- CN
 - Ancillary functions and macros, 57
- CO
 - Ancillary functions and macros, 57
- code_rate_hp_stream
 - dvb_desc_terrestrial_delivery, 200
- code_rate_lp_stream
 - dvb_desc_terrestrial_delivery, 200
- coding_type
 - dvb_mpeg_es_pic_start, 221
- component_descriptor
 - Digital TV table parsing, 87
- component_group_descriptor
 - Digital TV table parsing, 90
- component_name_descriptor
 - Digital TV table parsing, 89
- conditional_access_descriptor
 - Digital TV table parsing, 86
- conditional_playback_descriptor
 - Digital TV table parsing, 90
- connected_transmission_descriptor
 - Digital TV table parsing, 90
- constellation
 - dvb_desc_terrestrial_delivery, 200
- constrained
 - dvb_mpeg_es_seq_start, 224
- content_availability_descriptor
 - Digital TV table parsing, 90
- content_descriptor
 - Digital TV table parsing, 87
- content_identifier_descriptor
 - Digital TV table parsing, 88
- content_labeling_descriptor
 - Digital TV table parsing, 87
- continuity_counter
 - dvb_mpeg_ts, 232
 - dvb_ts_packet_header, 269
- copyright
 - dvb_mpeg_pes_optional, 228
- copyright_descriptor
 - Digital TV table parsing, 86
- country_availability_descriptor
 - Digital TV table parsing, 87
- COUNTRY_UNKNOWN
 - Ancillary functions and macros, 56
- CP_descriptor
 - Parsers for several MPEG-TS descriptors, 112
- CP_identifier_descriptor
 - Parsers for several MPEG-TS descriptors, 112
- cpcm_delivery_signalling_descriptor
 - Parsers for several MPEG-TS descriptors, 112
- CR
 - Ancillary functions and macros, 57
- CU
 - Ancillary functions and macros, 57
- CUE_identifier_descriptor
 - Digital TV table parsing, 89
- current_next
 - dvb_table_header, 249
- current_sys
 - dvb_v5_fe_parms, 276
- CV
 - Ancillary functions and macros, 57
- CW
 - Ancillary functions and macros, 57
- CX
 - Ancillary functions and macros, 57
- CY
 - Ancillary functions and macros, 57
- CZ
 - Ancillary functions and macros, 57
- data
 - dvb_desc, 164
 - dvb_mpeg_ts_adaption, 234
- data_alignment_indicator
 - dvb_mpeg_pes_optional, 228
- data_broadcast_descriptor
 - Digital TV table parsing, 88
- data_broadcast_id_descriptor

- Digital TV table parsing, [88](#)
- data_component_descriptor
 - Digital TV table parsing, [90](#)
- data_contents_descriptor
 - Digital TV table parsing, [89](#)
- DE
 - Ancillary functions and macros, [57](#)
- default_authority_descriptor
 - Digital TV table parsing, [88](#)
- default_charset
 - dvb_v5_fe_parms, [277](#)
- default_value
 - dvb_parse_table, [239](#)
- deferred_association_tags_descriptor
 - Digital TV table parsing, [89](#)
- delimiter
 - dvb_parse_file, [236](#)
- delivery_system
 - dvb_v5_descriptors, [272](#)
- delsys
 - dvb_parse_struct, [237](#)
- desc_ca.h
 - dvb_desc_ca_field_first, [303](#)
 - dvb_desc_ca_field_last, [303](#)
- desc_ca_identifier.h
 - dvb_desc_ca_identifier_field_first, [305](#)
 - dvb_desc_ca_identifier_field_last, [305](#)
- desc_cable_delivery.h
 - dvbc_fec_table, [307](#)
 - dvbc_modulation_table, [307](#)
- desc_isdbt_delivery.h
 - isdbt_interval, [319](#)
 - isdbt_mode, [319](#)
- desc_length
 - atsc_table_eit_desc_length, [149](#)
 - atsc_table_mgt_table, [155](#)
 - dvb_table_eit_event, [245](#)
 - dvb_table_nit, [252](#)
 - dvb_table_nit_transport, [253](#)
 - dvb_table_pmt, [259](#)
 - dvb_table_pmt_stream, [262](#)
 - dvb_table_sdt_service, [266](#)
- desc_sat.h
 - dvbs_dvbc_dvbs_freq_inner, [331](#)
 - dvbs_modulation, [331](#)
 - dvbs_polarization, [331](#)
 - dvbs_rolloff, [331](#)
- desc_t2_delivery.h
 - dvbt2_bw, [336](#)
 - dvbt2_interval, [336](#)
 - dvbt2_transmission_mode, [336](#)
 - siso_miso, [336](#)
- desc_terrestrial_delivery.h
 - dvb_desc_terrestrial_delivery_init, [339](#)
 - dvbt_bw, [339](#)
 - dvbt_code_rate, [339](#)
 - dvbt_hierarchy, [339](#)
 - dvbt_interval, [339](#)
 - dvbt_modulation, [340](#)
 - dvbt_transmission_mode, [340](#)
- description
 - dvb_desc_event_extended_item, [174](#)
- description_emph
 - dvb_desc_event_extended_item, [174](#)
- descriptor
 - atsc_table_eit_event, [151](#)
 - atsc_table_mgt, [153](#)
 - atsc_table_mgt_table, [155](#)
 - atsc_table_vct, [157](#)
 - atsc_table_vct_channel, [160](#)
 - dvb_extension_descriptor, [218](#)
 - dvb_table_cat, [242](#)
 - dvb_table_eit_event, [245](#)
 - dvb_table_nit, [252](#)
 - dvb_table_nit_transport, [253](#)
 - dvb_table_pmt, [259](#)
 - dvb_table_pmt_stream, [262](#)
 - dvb_table_sdt_service, [266](#)
- descriptor_length
 - atsc_table_vct_descriptor_length, [163](#)
- descriptors
 - Digital TV table parsing, [86](#)
- descriptors.h
 - TS_Information_descriptor, [346](#)
- descriptors_length
 - atsc_table_vct_channel, [160](#)
- devices
 - dvb_device, [209](#)
- Digital TV demux, [132](#)
 - dvb_dmx_close, [132](#)
 - dvb_dmx_open, [132](#)
 - dvb_dmx_stop, [133](#)
 - dvb_get_pmt_pid, [133](#)
 - dvb_set_pesfilter, [134](#)
 - dvb_set_section_filter, [134](#)
- Digital TV device enumeration, [11](#)
 - dvb_dev_alloc, [13](#)
 - dvb_dev_close, [13](#)
 - dvb_dev_dmx_get_pmt_pid, [13](#)
 - dvb_dev_dmx_set_pesfilter, [14](#)
 - dvb_dev_dmx_set_section_filter, [14](#)
 - dvb_dev_dmx_stop, [15](#)
 - dvb_dev_find, [15](#)
 - dvb_dev_free, [16](#)
 - dvb_dev_get_fd, [16](#)
 - dvb_dev_open, [17](#)
 - dvb_dev_read, [17](#)
 - dvb_dev_set_bufsize, [18](#)
 - dvb_dev_set_log, [18](#)
 - dvb_dev_set_logpriv, [19](#)
 - dvb_dev_stop_monitor, [19](#)
 - dvb_dev_type, [12](#)
 - DVB_DEVICE_AUDIO, [12](#)
 - DVB_DEVICE_CA, [12](#)
 - DVB_DEVICE_CA_SEC, [12](#)
 - DVB_DEVICE_DEMUX, [12](#)

- DVB_DEVICE_DVR, [12](#)
- DVB_DEVICE_FRONTEND, [12](#)
- DVB_DEVICE_NET, [12](#)
- DVB_DEVICE_VIDEO, [12](#)
- Digital TV frontend control, [20](#)
 - DTV_AUDIO_PID, [23](#)
 - DTV_BER, [23](#)
 - DTV_CH_NAME, [23](#)
 - DTV_COUNTRY_CODE, [23](#)
 - DTV_DISEQC_LNB, [23](#)
 - DTV_DISEQC_WAIT, [23](#)
 - DTV_FREQ_BPF, [24](#)
 - DTV_MAX_STAT_COMMAND, [24](#)
 - DTV_MAX_USER_COMMAND, [24](#)
 - DTV_NUM_KERNEL_STATS, [24](#)
 - DTV_NUM_STATS_PROPS, [24](#)
 - DTV_PER, [24](#)
 - DTV_PLS_CODE, [25](#)
 - DTV_PLS_MODE, [25](#)
 - DTV_POLARIZATION, [25](#)
 - DTV_PRE_BER, [25](#)
 - DTV_QUALITY, [26](#)
 - DTV_SAT_NUMBER, [26](#)
 - DTV_SERVICE_ID, [26](#)
 - DTV_STAT_COMMAND_START, [26](#)
 - DTV_STAT_NAME_SIZE, [26](#)
 - DTV_STATUS, [27](#)
 - DTV_USER_COMMAND_START, [27](#)
 - DTV_USER_NAME_SIZE, [27](#)
 - DTV_VCHANNEL, [27](#)
 - DTV_VIDEO_PID, [27](#)
 - dvb_add_parms_for_sys, [29](#)
 - dvb_attr_names, [29](#)
 - dvb_cmd_name, [30](#)
 - dvb_fe_close, [30](#)
 - dvb_fe_diseqc_burst, [30](#)
 - dvb_fe_diseqc_cmd, [31](#)
 - dvb_fe_diseqc_reply, [31](#)
 - dvb_fe_dummy, [31](#)
 - dvb_fe_get_event, [32](#)
 - dvb_fe_get_parms, [32](#)
 - dvb_fe_get_stats, [32](#)
 - dvb_fe_is_satellite, [33](#)
 - dvb_fe_lnb_high_voltage, [33](#)
 - dvb_fe_open, [33](#)
 - dvb_fe_open2, [34](#)
 - dvb_fe_open_flags, [34](#)
 - dvb_fe_prt_parms, [35](#)
 - dvb_fe_retrieve_ber, [35](#)
 - dvb_fe_retrieve_parm, [36](#)
 - dvb_fe_retrieve_per, [36](#)
 - dvb_fe_retrieve_quality, [37](#)
 - dvb_fe_retrieve_stats, [37](#)
 - dvb_fe_retrieve_stats_layer, [38](#)
 - dvb_fe_sec_tone, [38](#)
 - dvb_fe_sec_voltage, [39](#)
 - dvb_fe_set_default_country, [39](#)
 - dvb_fe_set_parms, [39](#)
 - dvb_fe_snprintf_eng, [40](#)
 - dvb_fe_snprintf_stat, [40](#)
 - dvb_fe_store_parm, [41](#)
 - DVB_QUAL_GOOD, [29](#)
 - DVB_QUAL_OK, [29](#)
 - DVB_QUAL_POOR, [29](#)
 - DVB_QUAL_UNKNOWN, [29](#)
 - dvb_quality, [28](#)
 - dvb_set_compat_delivery_system, [41](#)
 - dvb_set_sys, [42](#)
 - MAX_DELIVERY_SYSTEMS, [28](#)
- Digital TV frontend scan, [43](#)
 - check_frontend_t, [44](#)
 - dvb_add_scanned_transponders, [44](#)
 - dvb_dev_scan, [45](#)
 - dvb_free_ts_tables, [45](#)
 - dvb_get_ts_tables, [46](#)
 - dvb_read_section, [46](#)
 - dvb_read_section_with_id, [47](#)
 - dvb_read_sections, [47](#)
 - dvb_scan_alloc_handler_table, [48](#)
 - dvb_scan_free_handler_table, [48](#)
 - dvb_scan_transponder, [48](#)
 - dvb_table_filter_free, [49](#)
- Digital TV table parsing, [65](#)
 - AAC_descriptor, [89](#)
 - AC_3_descriptor, [88](#)
 - adaptation_field_data_descriptor, [88](#)
 - ancillary_data_descriptor, [88](#)
 - announcement_support_descriptor, [88](#)
 - application_signalling_descriptor, [88](#)
 - association_tag_descriptor, [89](#)
 - atsc_ac3_audio_descriptor, [90](#)
 - atsc_ATSC_private_information_descriptor, [90](#)
 - ATSC_BASE_PID, [73](#)
 - atsc_caption_service_descriptor, [90](#)
 - atsc_component_name_descriptor, [90](#)
 - atsc_content_advisory_descriptor, [90](#)
 - atsc_DCC_arriving_request_descriptor, [90](#)
 - atsc_DCC_departing_request_descriptor, [90](#)
 - atsc_eit_event_foreach, [73](#)
 - atsc_extended_channel_descriptor, [90](#)
 - atsc_genre_descriptor, [90](#)
 - atsc_redistribution_control_descriptor, [90](#)
 - atsc_service_location_descriptor, [90](#)
 - atsc_stuffing_descriptor, [90](#)
 - ATSC_TABLE_CVCT, [74](#)
 - ATSC_TABLE_EIT, [74](#)
 - atsc_table_eit_free, [92](#)
 - atsc_table_eit_init, [92](#)
 - atsc_table_eit_print, [92](#)
 - ATSC_TABLE_MGT, [74](#)
 - atsc_table_mgt_free, [93](#)
 - atsc_table_mgt_init, [93](#)
 - atsc_table_mgt_print, [93](#)
 - ATSC_TABLE_TVCT, [74](#)
 - atsc_table_vct_free, [95](#)
 - atsc_table_vct_init, [95](#)

ATSC_TABLE_VCT_PID, 74
atsc_table_vct_print, 95
atsc_time_shifted_service_descriptor, 90
atsc_vct_channel_foreach, 74
audio_component_descriptor, 89
audio_stream_descriptor, 86
AVC_timing_and_HRD_descriptor, 87
AVC_video_descriptor, 87
basic_local_event_descriptor, 89
board_information_descriptor, 90
bouquet_name_descriptor, 87
broadcaster_Name_Descriptor, 90
CA_contract_information_descriptor, 89
CA_EMM_TS_descriptor, 89
CA_identifier_descriptor, 87
CA_service_descriptor, 89
cable_delivery_system_descriptor, 87
carousel_compatible_composite_descriptor, 90
carousel_id_descriptor, 89
cell_frequency_link_descriptor, 88
cell_list_descriptor, 88
component_descriptor, 87
component_group_descriptor, 90
component_name_descriptor, 89
conditional_access_descriptor, 86
conditional_playback_descriptor, 90
connected_transmission_descriptor, 90
content_availability_descriptor, 90
content_descriptor, 87
content_identifier_descriptor, 88
content_labeling_descriptor, 87
copyright_descriptor, 86
country_availability_descriptor, 87
CUE_identifier_descriptor, 89
data_broadcast_descriptor, 88
data_broadcast_id_descriptor, 88
data_component_descriptor, 90
data_contents_descriptor, 89
default_authority_descriptor, 88
deferred_association_tags_descriptor, 89
descriptors, 86
digital_copy_control_descriptor, 89
download_content_descriptor, 89
ds_alignment_descriptor, 86
DSNG_descriptor, 88
DTS_descriptor, 89
dvb_bcd, 96
DVB_CRC_SIZE, 75
dvb_desc_ext_free_func, 82
dvb_desc_ext_init_func, 82
dvb_desc_ext_print_func, 84
dvb_desc_free, 96
dvb_desc_free_func, 84
dvb_desc_init_func, 84
dvb_desc_parse, 96
dvb_desc_print, 97
dvb_desc_print_func, 85
dvb_descriptors, 107
dvb_eit_event_foreach, 75
dvb_hexdump, 97
DVB_MAX_PAYLOAD_PACKET_SIZE, 75
DVB_MPEG_ES_FRAME_B, 91
DVB_MPEG_ES_FRAME_D, 91
DVB_MPEG_ES_FRAME_I, 91
dvb_mpeg_es_frame_names, 107
DVB_MPEG_ES_FRAME_P, 91
dvb_mpeg_es_frame_t, 90
DVB_MPEG_ES_FRAME_UNKNOWN, 91
DVB_MPEG_ES_GOP, 75
DVB_MPEG_ES_PIC_START, 75
dvb_mpeg_es_pic_start_init, 97
dvb_mpeg_es_pic_start_print, 98
DVB_MPEG_ES_SEQ_EXT, 76
DVB_MPEG_ES_SEQ_START, 76
dvb_mpeg_es_seq_start_init, 98
dvb_mpeg_es_seq_start_print, 98
DVB_MPEG_ES_SLICES, 76
DVB_MPEG_ES_USER_DATA, 76
DVB_MPEG_PES, 76
DVB_MPEG_PES_AUDIO, 76
dvb_mpeg_pes_free, 99
dvb_mpeg_pes_init, 99
dvb_mpeg_pes_print, 99
DVB_MPEG_PES_VIDEO, 77
DVB_MPEG_STREAM_DIRECTORY, 77
DVB_MPEG_STREAM_DSMCC, 77
DVB_MPEG_STREAM_ECM, 77
DVB_MPEG_STREAM_EMM, 77
DVB_MPEG_STREAM_H222E, 77
DVB_MPEG_STREAM_MAP, 78
DVB_MPEG_STREAM_PADDING, 78
DVB_MPEG_STREAM_PRIVATE_2, 78
DVB_MPEG_TS, 78
dvb_mpeg_ts_free, 99
dvb_mpeg_ts_init, 100
DVB_MPEG_TS_PACKET_SIZE, 78
dvb_mpeg_ts_print, 100
dvb_nit_transport_foreach, 78
dvb_pat_program_foreach, 79
dvb_pmt_stream_foreach, 79
dvb_sdt_service_foreach, 79
dvb_streams, 91
DVB_TABLE_CAT, 80
DVB_TABLE_CAT_PID, 80
DVB_TABLE_EIT, 80
dvb_table_eit_free, 100
dvb_table_eit_init, 101
DVB_TABLE_EIT_OTHER, 80
DVB_TABLE_EIT_PID, 80
dvb_table_eit_print, 101
DVB_TABLE_EIT_SCHEDULE, 80
DVB_TABLE_EIT_SCHEDULE_OTHER, 80
dvb_table_header_init, 101
dvb_table_header_print, 102
dvb_table_init_func, 85
dvb_table_initializers, 107

DVB_TABLE_NIT, 81
DVB_TABLE_NIT2, 81
dvb_table_nit_descriptor_handler, 102
dvb_table_nit_free, 102
dvb_table_nit_init, 103
DVB_TABLE_NIT_PID, 81
dvb_table_nit_print, 103
DVB_TABLE_PAT, 81
dvb_table_pat_free, 103
dvb_table_pat_init, 104
DVB_TABLE_PAT_PID, 81
dvb_table_pat_print, 104
DVB_TABLE_PMT, 81
dvb_table_pmt_free, 104
dvb_table_pmt_init, 105
dvb_table_pmt_print, 105
DVB_TABLE_SDT, 82
DVB_TABLE_SDT2, 82
dvb_table_sdt_free, 105
dvb_table_sdt_init, 106
DVB_TABLE_SDT_PID, 82
dvb_table_sdt_print, 106
dvb_time, 106
ECM_repetition_rate_descriptor, 89
emergency_information_descriptor, 90
enhanced_AC_3_descriptor, 89
event_group_descriptor, 90
extended_broadcaster_descriptor, 89
extended_channel_name, 89
extended_event_descriptor, 87
extension_descriptor, 89
external_es_id_descriptor, 86
flexmux_timing_descriptor, 87
fmc_descriptor, 86
fmxbuffersize_descriptor, 87
frequency_list_descriptor, 88
FTA_content_management_descriptor, 89
hierarchical_transmission_descriptor, 89
hierarchy_descriptor, 86
hyperlink_descriptor, 89
ibp_descriptor, 86
iod_descriptor, 86
ipmp_descriptor, 87
ISDBT_delivery_system_descriptor, 90
iso639_language_descriptor, 86
LDT_linkage_descriptor, 90
linkage_descriptor, 87
local_time_offset_descriptor, 87
logical_channel_number_descriptor, 89
logo_transmission_descriptor, 89
maximum_bitrate_descriptor, 86
metadata_descriptor, 87
metadata_pointer_descriptor, 87
metadata_std_descriptor, 87
mosaic_descriptor, 87
mpeg2_aac_audio_descriptor, 87
mpeg4_audio_descriptor, 86
mpeg4_video_descriptor, 86
multilingual_bouquet_name_descriptor, 88
multilingual_component_descriptor, 88
multilingual_network_name_descriptor, 88
multilingual_service_name_descriptor, 88
multiplex_buffer_utilization_descriptor, 86
multiplexbuffer_descriptor, 87
muxcode_descriptor, 86
network_identifier_descriptor, 89
network_name_descriptor, 87
nit_handler_callback_t, 85
nit_tran_handler_callback_t, 85
node_relation_descriptor, 90
NVOD_reference_descriptor, 87
parental_rating_descriptor, 87
partial_reception_descriptor, 90
partial_transport_stream_descriptor, 88
partial_transport_stream_time_descriptor, 89
PDC_descriptor, 88
pmt_stream_name, 107
private_data_indicator_descriptor, 86
private_data_specifier_descriptor, 88
reference_descriptor, 90
registration_descriptor, 86
related_content_descriptor, 88
S2_satellite_delivery_system_descriptor, 89
satellite_delivery_system_descriptor, 87
scrambling_descriptor, 88
series_descriptor, 90
service_availability_descriptor, 88
service_descriptor, 87
service_group_descriptor, 90
service_identifier_descriptor, 88
service_list_descriptor, 87
service_location, 89
service_move_descriptor, 88
short_event_descriptor, 87
short_node_information_descriptor, 90
short_smoothing_buffer_descriptor, 88
SI_parameter_descriptor, 90
SI_prime_TS_descriptor, 90
sl_descriptor, 86
smoothing_buffer_descriptor, 86
STC_reference_descriptor, 90
std_descriptor, 86
stream_13818_6_A, 91
stream_13818_6_B, 91
stream_13818_6_C, 91
stream_13818_6_D, 91
stream_14496_1_iso, 91
stream_14496_1_pes, 91
stream_audio, 91
stream_audio_13818_3, 91
stream_audio_14496_3, 91
stream_audio_a52, 91
stream_audio_a52_vls, 92
stream_audio_adts, 91
stream_audio_dts, 92
stream_audio_dts_hdmv, 91

- stream_audio_e_ac3, [92](#)
- stream_audio_latm, [91](#)
- stream_audio_sdds, [91](#)
- stream_audio_sdds2, [92](#)
- stream_download, [91](#)
- stream_h222, [91](#)
- stream_h222_1, [91](#)
- stream_h222_aux, [91](#)
- stream_identifier_descriptor, [87](#)
- stream_mheg, [91](#)
- stream_private_data, [91](#)
- stream_private_sections, [91](#)
- stream_scte_27, [91](#)
- stream_spu_vls, [92](#)
- stream_video, [91](#)
- stream_video_14496_2, [91](#)
- stream_video_cavs, [91](#)
- stream_video_h262, [91](#)
- stream_video_h264, [91](#)
- stream_video_hevc, [91](#)
- stream_video_moto, [91](#)
- stuffing_descriptor, [87](#)
- subtitling_descriptor, [87](#)
- system_clock_descriptor, [86](#)
- system_management_descriptor, [90](#)
- target_area_descriptor, [89](#)
- target_background_grid_descriptor, [86](#)
- telephone_descriptor, [87](#)
- teletext_descriptor, [87](#)
- terrestrial_delivery_system_descriptor, [87](#)
- time_shifted_event_descriptor, [87](#)
- time_shifted_service_descriptor, [87](#)
- time_slice_fec_identifier_descriptor, [88](#)
- transport_stream_descriptor, [88](#)
- TS_Information_descriptor, [89](#)
- TVA_id_descriptor, [88](#)
- VBI_data_descriptor, [87](#)
- VBI_teletext_descriptor, [87](#)
- video_decode_control_descriptor, [89](#)
- video_stream_descriptor, [86](#)
- video_window_descriptor, [86](#)
- XAIT_location_descriptor, [89](#)
- digital_copy_control_descriptor
 - Digital TV table parsing, [89](#)
- discontinued
 - dvb_mpeg_ts_adaption, [234](#)
 - dvb_ts_packet_header, [269](#)
- diseqc_wait
 - dvb_entry, [213](#)
 - dvb_v5_fe_parms, [277](#)
- DJ
 - Ancillary functions and macros, [57](#)
- DK
 - Ancillary functions and macros, [57](#)
- DM
 - Ancillary functions and macros, [57](#)
- DO
 - Ancillary functions and macros, [57](#)
- doc/libdvbv5-index.doc, [286](#)
- download_content_descriptor
 - Digital TV table parsing, [89](#)
- ds_alignment_descriptor
 - Digital TV table parsing, [86](#)
- DSM_trick_mode
 - dvb_mpeg_pes_optional, [229](#)
- DSNG_descriptor
 - Digital TV table parsing, [88](#)
- dts
 - dvb_mpeg_pes_optional, [229](#)
- DTS_descriptor
 - Digital TV table parsing, [89](#)
- DTV_AUDIO_PID
 - Digital TV frontend control, [23](#)
- DTV_BER
 - Digital TV frontend control, [23](#)
- DTV_CH_NAME
 - Digital TV frontend control, [23](#)
- DTV_COUNTRY_CODE
 - Digital TV frontend control, [23](#)
- DTV_DISEQC_LNB
 - Digital TV frontend control, [23](#)
- DTV_DISEQC_WAIT
 - Digital TV frontend control, [23](#)
- DTV_FREQ_BPF
 - Digital TV frontend control, [24](#)
- DTV_MAX_STAT_COMMAND
 - Digital TV frontend control, [24](#)
- DTV_MAX_USER_COMMAND
 - Digital TV frontend control, [24](#)
- DTV_NUM_KERNEL_STATS
 - Digital TV frontend control, [24](#)
- DTV_NUM_STATS_PROPS
 - Digital TV frontend control, [24](#)
- DTV_PER
 - Digital TV frontend control, [24](#)
- DTV_PLS_CODE
 - Digital TV frontend control, [25](#)
- DTV_PLS_MODE
 - Digital TV frontend control, [25](#)
- DTV_POLARIZATION
 - Digital TV frontend control, [25](#)
- DTV_PRE_BER
 - Digital TV frontend control, [25](#)
- DTV_QUALITY
 - Digital TV frontend control, [26](#)
- DTV_SAT_NUMBER
 - Digital TV frontend control, [26](#)
- DTV_SERVICE_ID
 - Digital TV frontend control, [26](#)
- DTV_STAT_COMMAND_START
 - Digital TV frontend control, [26](#)
- DTV_STAT_NAME_SIZE
 - Digital TV frontend control, [26](#)
- DTV_STATUS
 - Digital TV frontend control, [27](#)
- DTV_USER_COMMAND_START

- Digital TV frontend control, [27](#)
- DTV_USER_NAME_SIZE
 - Digital TV frontend control, [27](#)
- DTV_VCHANNEL
 - Digital TV frontend control, [27](#)
- DTV_VIDEO_PID
 - Digital TV frontend control, [27](#)
- dummy
 - dvb_mpeg_es_pic_start, [221](#)
- duration
 - atsc_table_eit_event, [151](#)
 - dvb_table_eit_event, [245](#)
- dvb-dev.h
 - DVB_DEV_ADD, [354](#)
 - DVB_DEV_CHANGE, [354](#)
 - dvb_dev_change_t, [353](#)
 - dvb_dev_change_type, [353](#)
 - dvb_dev_remote_init, [354](#)
 - DVB_DEV_REMOVE, [354](#)
 - dvb_dev_seek_by_adapter, [354](#)
 - dvb_get_dev_info, [355](#)
- dvb-log.h
 - dvb_logfunc_priv, [367](#)
- dvb-scan.h
 - MAX_TABLE_SIZE, [372](#)
- dvb_add_parms_for_sys
 - Digital TV frontend control, [29](#)
- dvb_add_scanned_transponders
 - Digital TV frontend scan, [44](#)
- dvb_attr_names
 - Digital TV frontend control, [29](#)
- dvb_bcd
 - Digital TV table parsing, [96](#)
- dvb_cmd_name
 - Digital TV frontend control, [30](#)
- dvb_country_a2_to_id
 - Ancillary functions and macros, [63](#)
- dvb_country_a3_to_id
 - Ancillary functions and macros, [63](#)
- dvb_country_t
 - Ancillary functions and macros, [56](#)
- dvb_country_to_2letters
 - Ancillary functions and macros, [63](#)
- dvb_country_to_3letters
 - Ancillary functions and macros, [64](#)
- dvb_country_to_name
 - Ancillary functions and macros, [64](#)
- dvb_crc32
 - Ancillary functions and macros, [64](#)
- DVB_CRC_SIZE
 - Digital TV table parsing, [75](#)
- dvb_default_log
 - Ancillary functions and macros, [65](#)
- dvb_desc, [164](#)
 - data, [164](#)
 - length, [164](#)
 - next, [165](#)
 - type, [165](#)
- dvb_desc_ca, [165](#)
 - bitfield1, [166](#)
 - ca_id, [166](#)
 - ca_pid, [166](#)
 - length, [166](#)
 - next, [166](#)
 - privdata, [167](#)
 - privdata_len, [167](#)
 - reserved, [167](#)
 - type, [167](#)
- dvb_desc_ca_field_first
 - desc_ca.h, [303](#)
- dvb_desc_ca_field_last
 - desc_ca.h, [303](#)
- dvb_desc_ca_free
 - Parsers for several MPEG-TS descriptors, [113](#)
- dvb_desc_ca_identifier, [167](#)
 - caid_count, [168](#)
 - caids, [168](#)
 - length, [168](#)
 - next, [168](#)
 - type, [168](#)
- dvb_desc_ca_identifier_field_first
 - desc_ca_identifier.h, [305](#)
- dvb_desc_ca_identifier_field_last
 - desc_ca_identifier.h, [305](#)
- dvb_desc_ca_identifier_free
 - Parsers for several MPEG-TS descriptors, [114](#)
- dvb_desc_ca_identifier_init
 - Parsers for several MPEG-TS descriptors, [114](#)
- dvb_desc_ca_identifier_print
 - Parsers for several MPEG-TS descriptors, [114](#)
- dvb_desc_ca_init
 - Parsers for several MPEG-TS descriptors, [115](#)
- dvb_desc_ca_print
 - Parsers for several MPEG-TS descriptors, [115](#)
- dvb_desc_cable_delivery, [169](#)
 - bitfield1, [170](#)
 - bitfield2, [170](#)
 - fec_inner, [170](#)
 - fec_outer, [170](#)
 - frequency, [170](#)
 - length, [170](#)
 - modulation, [170](#)
 - next, [170](#)
 - reserved_future_use, [171](#)
 - symbol_rate, [171](#)
 - type, [171](#)
- dvb_desc_cable_delivery_init
 - Parsers for several MPEG-TS descriptors, [115](#)
- dvb_desc_cable_delivery_print
 - Parsers for several MPEG-TS descriptors, [116](#)
- dvb_desc_event_extended, [171](#)
 - id, [172](#)
 - ids, [172](#)
 - items, [172](#)
 - language, [172](#)
 - last_id, [172](#)

- length, 173
- next, 173
- num_items, 173
- text, 173
- text_emph, 173
- type, 173
- dvb_desc_event_extended_free
 - Parsers for several MPEG-TS descriptors, 116
- dvb_desc_event_extended_init
 - Parsers for several MPEG-TS descriptors, 116
- dvb_desc_event_extended_item, 173
 - description, 174
 - description_emph, 174
 - item, 174
 - item_emph, 174
- dvb_desc_event_extended_print
 - Parsers for several MPEG-TS descriptors, 117
- dvb_desc_event_short, 174
 - language, 176
 - length, 176
 - name, 176
 - name_emph, 176
 - next, 176
 - text, 176
 - text_emph, 177
 - type, 177
- dvb_desc_event_short_free
 - Parsers for several MPEG-TS descriptors, 117
- dvb_desc_event_short_init
 - Parsers for several MPEG-TS descriptors, 117
- dvb_desc_event_short_print
 - Parsers for several MPEG-TS descriptors, 118
- dvb_desc_ext_free_func
 - Digital TV table parsing, 82
- dvb_desc_ext_init_func
 - Digital TV table parsing, 82
- dvb_desc_ext_print_func
 - Digital TV table parsing, 84
- dvb_desc_free
 - Digital TV table parsing, 96
- dvb_desc_free_func
 - Digital TV table parsing, 84
- dvb_desc_frequency_list, 177
 - bitfield, 178
 - freq_type, 178
 - frequencies, 178
 - frequency, 178
 - length, 178
 - next, 178
 - reserved, 178
 - type, 179
- dvb_desc_frequency_list_init
 - Parsers for several MPEG-TS descriptors, 118
- dvb_desc_frequency_list_print
 - Parsers for several MPEG-TS descriptors, 118
- dvb_desc_hierarchy, 179
 - channel, 180
 - embedded_layer, 180
 - hierarchy_type, 180
 - layer, 180
 - length, 180
 - next, 180
 - reserved, 180
 - reserved2, 180
 - reserved3, 181
 - reserved4, 181
 - type, 181
- dvb_desc_hierarchy_init
 - Parsers for several MPEG-TS descriptors, 119
- dvb_desc_hierarchy_print
 - Parsers for several MPEG-TS descriptors, 119
- dvb_desc_init_func
 - Digital TV table parsing, 84
- dvb_desc_language, 181
 - audio_type, 182
 - language, 182
 - length, 182
 - next, 182
 - type, 182
- dvb_desc_language_init
 - Parsers for several MPEG-TS descriptors, 119
- dvb_desc_language_print
 - Parsers for several MPEG-TS descriptors, 121
- dvb_desc_logical_channel, 182
 - lcn, 183
 - length, 183
 - next, 183
 - type, 183
- dvb_desc_logical_channel_free
 - Parsers for several MPEG-TS descriptors, 121
- dvb_desc_logical_channel_init
 - Parsers for several MPEG-TS descriptors, 121
- dvb_desc_logical_channel_number, 183
 - bitfield, 184
 - logical_channel_number, 184
 - reserved, 184
 - service_id, 185
 - visible_service_flag, 185
- dvb_desc_logical_channel_print
 - Parsers for several MPEG-TS descriptors, 122
- dvb_desc_network_name, 185
 - length, 186
 - network_name, 186
 - network_name_emph, 186
 - next, 186
 - type, 186
- dvb_desc_network_name_free
 - Parsers for several MPEG-TS descriptors, 122
- dvb_desc_network_name_init
 - Parsers for several MPEG-TS descriptors, 122
- dvb_desc_network_name_print
 - Parsers for several MPEG-TS descriptors, 123
- dvb_desc_parse
 - Digital TV table parsing, 96
- dvb_desc_print
 - Digital TV table parsing, 97

- dvb_desc_print_func
 - Digital TV table parsing, [85](#)
- dvb_desc_registration, [186](#)
 - additional_identification_info, [187](#)
 - format_identifier, [187](#)
 - length, [187](#)
 - next, [187](#)
 - type, [187](#)
- dvb_desc_registration_free
 - Parsers for several MPEG-TS descriptors, [123](#)
- dvb_desc_registration_init
 - Parsers for several MPEG-TS descriptors, [123](#)
- dvb_desc_registration_print
 - Parsers for several MPEG-TS descriptors, [124](#)
- dvb_desc_sat, [188](#)
 - bitfield, [189](#)
 - fec, [189](#)
 - frequency, [189](#)
 - length, [189](#)
 - modulation_system, [189](#)
 - modulation_type, [189](#)
 - next, [189](#)
 - orbit, [189](#)
 - polarization, [190](#)
 - roll_off, [190](#)
 - symbol_rate, [190](#)
 - type, [190](#)
 - west_east, [190](#)
- dvb_desc_sat_init
 - Parsers for several MPEG-TS descriptors, [124](#)
- dvb_desc_sat_print
 - Parsers for several MPEG-TS descriptors, [124](#)
- dvb_desc_service, [190](#)
 - length, [191](#)
 - name, [191](#)
 - name_emph, [191](#)
 - next, [191](#)
 - provider, [191](#)
 - provider_emph, [191](#)
 - service_type, [192](#)
 - type, [192](#)
- dvb_desc_service_free
 - Parsers for several MPEG-TS descriptors, [125](#)
- dvb_desc_service_init
 - Parsers for several MPEG-TS descriptors, [125](#)
- dvb_desc_service_print
 - Parsers for several MPEG-TS descriptors, [125](#)
- dvb_desc_t2_delivery, [192](#)
 - bandwidth, [193](#)
 - bitfield, [193](#)
 - cell, [193](#)
 - centre_frequency, [193](#)
 - frequency_loop_length, [194](#)
 - guard_interval, [194](#)
 - num_cell, [194](#)
 - other_frequency_flag, [194](#)
 - plp_id, [194](#)
 - reserved, [194](#)
 - SISO_MISO, [194](#)
 - subcel_info_loop_length, [194](#)
 - subcell, [195](#)
 - system_id, [195](#)
 - tfs_flag, [195](#)
 - transmission_mode, [195](#)
- dvb_desc_t2_delivery_cell, [195](#)
 - cell_id, [196](#)
 - centre_frequency, [196](#)
 - num_freqs, [196](#)
 - subcel, [196](#)
 - subcel_length, [196](#)
- dvb_desc_t2_delivery_free
 - Parsers for several MPEG-TS descriptors, [126](#)
- dvb_desc_t2_delivery_init
 - Parsers for several MPEG-TS descriptors, [126](#)
- dvb_desc_t2_delivery_print
 - Parsers for several MPEG-TS descriptors, [126](#)
- dvb_desc_t2_delivery_subcell, [197](#)
 - cell_id_extension, [198](#)
 - transposer_frequency, [198](#)
- dvb_desc_t2_delivery_subcell_old, [198](#)
 - cell_id_extension, [199](#)
 - transposer_frequency, [199](#)
- dvb_desc_terrestrial_delivery, [199](#)
 - bandwidth, [200](#)
 - centre_frequency, [200](#)
 - code_rate_hp_stream, [200](#)
 - code_rate_lp_stream, [200](#)
 - constellation, [200](#)
 - guard_interval, [201](#)
 - hierarchy_information, [201](#)
 - length, [201](#)
 - mpe_fec_indicator, [201](#)
 - next, [201](#)
 - other_frequency_flag, [201](#)
 - priority, [201](#)
 - reserved_future_use1, [201](#)
 - reserved_future_use2, [202](#)
 - time_slice_indicator, [202](#)
 - transmission_mode, [202](#)
 - type, [202](#)
- dvb_desc_terrestrial_delivery_init
 - desc_terrestrial_delivery.h, [339](#)
- dvb_desc_terrestrial_delivery_print
 - Parsers for several MPEG-TS descriptors, [127](#)
- dvb_desc_ts_info, [202](#)
 - bitfield, [203](#)
 - length, [203](#)
 - length_of_ts_name, [203](#)
 - next, [203](#)
 - remote_control_key_id, [203](#)
 - service_id, [204](#)
 - transmission_type, [204](#)
 - transmission_type_count, [204](#)
 - ts_name, [204](#)
 - ts_name_emph, [204](#)
 - type, [204](#)

- dvb_desc_ts_info_free
 - Parsers for several MPEG-TS descriptors, [127](#)
- dvb_desc_ts_info_init
 - Parsers for several MPEG-TS descriptors, [127](#)
- dvb_desc_ts_info_print
 - Parsers for several MPEG-TS descriptors, [127](#)
- dvb_desc_ts_info_transmission_type, [204](#)
 - num_of_service, [205](#)
 - transmission_type_info, [205](#)
- dvb_descriptor, [205](#)
 - free, [206](#)
 - init, [206](#)
 - name, [206](#)
 - print, [206](#)
 - size, [206](#)
- dvb_descriptors
 - Digital TV table parsing, [107](#)
- DVB_DEV_ADD
 - dvb-dev.h, [354](#)
- dvb_dev_alloc
 - Digital TV device enumeration, [13](#)
- DVB_DEV_CHANGE
 - dvb-dev.h, [354](#)
- dvb_dev_change_t
 - dvb-dev.h, [353](#)
- dvb_dev_change_type
 - dvb-dev.h, [353](#)
- dvb_dev_close
 - Digital TV device enumeration, [13](#)
- dvb_dev_dmx_get_pmt_pid
 - Digital TV device enumeration, [13](#)
- dvb_dev_dmx_set_pesfilter
 - Digital TV device enumeration, [14](#)
- dvb_dev_dmx_set_section_filter
 - Digital TV device enumeration, [14](#)
- dvb_dev_dmx_stop
 - Digital TV device enumeration, [15](#)
- dvb_dev_find
 - Digital TV device enumeration, [15](#)
- dvb_dev_free
 - Digital TV device enumeration, [16](#)
- dvb_dev_get_fd
 - Digital TV device enumeration, [16](#)
- dvb_dev_list, [207](#)
 - bus_addr, [207](#)
 - bus_id, [207](#)
 - dvb_type, [208](#)
 - manufacturer, [208](#)
 - path, [208](#)
 - product, [208](#)
 - serial, [208](#)
 - sysname, [208](#)
 - syspath, [208](#)
- dvb_dev_open
 - Digital TV device enumeration, [17](#)
- dvb_dev_read
 - Digital TV device enumeration, [17](#)
- dvb_dev_remote_init
 - dvb-dev.h, [354](#)
- DVB_DEV_REMOVE
 - dvb-dev.h, [354](#)
- dvb_dev_scan
 - Digital TV frontend scan, [45](#)
- dvb_dev_seek_by_adapter
 - dvb-dev.h, [354](#)
- dvb_dev_set_bufsize
 - Digital TV device enumeration, [18](#)
- dvb_dev_set_log
 - Digital TV device enumeration, [18](#)
- dvb_dev_set_logpriv
 - Digital TV device enumeration, [19](#)
- dvb_dev_stop_monitor
 - Digital TV device enumeration, [19](#)
- dvb_dev_type
 - Digital TV device enumeration, [12](#)
- dvb_device, [209](#)
 - devices, [209](#)
 - fe_parms, [209](#)
 - num_devices, [210](#)
- DVB_DEVICE_AUDIO
 - Digital TV device enumeration, [12](#)
- DVB_DEVICE_CA
 - Digital TV device enumeration, [12](#)
- DVB_DEVICE_CA_SEC
 - Digital TV device enumeration, [12](#)
- DVB_DEVICE_DEMUX
 - Digital TV device enumeration, [12](#)
- DVB_DEVICE_DVR
 - Digital TV device enumeration, [12](#)
- DVB_DEVICE_FRONTEND
 - Digital TV device enumeration, [12](#)
- DVB_DEVICE_NET
 - Digital TV device enumeration, [12](#)
- DVB_DEVICE_VIDEO
 - Digital TV device enumeration, [12](#)
- dvb_dmx_close
 - Digital TV demux, [132](#)
- dvb_dmx_open
 - Digital TV demux, [132](#)
- dvb_dmx_stop
 - Digital TV demux, [133](#)
- dvb_eit_event_foreach
 - Digital TV table parsing, [75](#)
- dvb_eit_running_status_name
 - eit.h, [379](#)
- dvb_elementary_pid, [210](#)
 - pid, [210](#)
 - type, [211](#)
- dvb_entry, [211](#)
 - audio_pid, [212](#)
 - audio_pid_len, [212](#)
 - channel, [213](#)
 - diseqc_wait, [213](#)
 - freq_bpf, [213](#)
 - lnb, [213](#)
 - location, [213](#)

- n_props, [214](#)
- network_id, [214](#)
- next, [214](#)
- other_el_pid, [214](#)
- other_el_pid_len, [214](#)
- props, [215](#)
- sat_number, [215](#)
- service_id, [215](#)
- transport_id, [215](#)
- vchannel, [215](#)
- video_pid, [216](#)
- video_pid_len, [216](#)
- dvb_ext_descriptor, [216](#)
 - free, [217](#)
 - init, [217](#)
 - name, [217](#)
 - print, [217](#)
 - size, [217](#)
- dvb_extension_descriptor, [218](#)
 - descriptor, [218](#)
 - extension_code, [218](#)
 - length, [218](#)
 - next, [218](#)
 - type, [219](#)
- dvb_extension_descriptor_free
 - Parsers for several MPEG-TS descriptors, [128](#)
- dvb_extension_descriptor_init
 - Parsers for several MPEG-TS descriptors, [128](#)
- dvb_extension_descriptor_print
 - Parsers for several MPEG-TS descriptors, [128](#)
- dvb_fe_close
 - Digital TV frontend control, [30](#)
- dvb_fe_diseqc_burst
 - Digital TV frontend control, [30](#)
- dvb_fe_diseqc_cmd
 - Digital TV frontend control, [31](#)
- dvb_fe_diseqc_reply
 - Digital TV frontend control, [31](#)
- dvb_fe_dummy
 - Digital TV frontend control, [31](#)
- dvb_fe_get_event
 - Digital TV frontend control, [32](#)
- dvb_fe_get_parms
 - Digital TV frontend control, [32](#)
- dvb_fe_get_stats
 - Digital TV frontend control, [32](#)
- dvb_fe_is_satellite
 - Digital TV frontend control, [33](#)
- dvb_fe_lnb_high_voltage
 - Digital TV frontend control, [33](#)
- dvb_fe_open
 - Digital TV frontend control, [33](#)
- dvb_fe_open2
 - Digital TV frontend control, [34](#)
- dvb_fe_open_flags
 - Digital TV frontend control, [34](#)
- dvb_fe_prt_parms
 - Digital TV frontend control, [35](#)
- dvb_fe_retrieve_ber
 - Digital TV frontend control, [35](#)
- dvb_fe_retrieve_parm
 - Digital TV frontend control, [36](#)
- dvb_fe_retrieve_per
 - Digital TV frontend control, [36](#)
- dvb_fe_retrieve_quality
 - Digital TV frontend control, [37](#)
- dvb_fe_retrieve_stats
 - Digital TV frontend control, [37](#)
- dvb_fe_retrieve_stats_layer
 - Digital TV frontend control, [38](#)
- dvb_fe_sec_tone
 - Digital TV frontend control, [38](#)
- dvb_fe_sec_voltage
 - Digital TV frontend control, [39](#)
- dvb_fe_set_default_country
 - Digital TV frontend control, [39](#)
- dvb_fe_set_parms
 - Digital TV frontend control, [39](#)
- dvb_fe_snprintf_eng
 - Digital TV frontend control, [40](#)
- dvb_fe_snprintf_stat
 - Digital TV frontend control, [40](#)
- dvb_fe_store_parm
 - Digital TV frontend control, [41](#)
- dvb_file, [219](#)
 - first_entry, [219](#)
 - fname, [220](#)
 - n_entries, [220](#)
- dvb_file_formats
 - Channel and transponder file read/write, [136](#)
- dvb_file_free
 - Channel and transponder file read/write, [137](#)
- dvb_free_ts_tables
 - Digital TV frontend scan, [45](#)
- dvb_get_dev_info
 - dvb-dev.h, [355](#)
- dvb_get_pmt_pid
 - Digital TV demux, [133](#)
- dvb_get_ts_tables
 - Digital TV frontend scan, [46](#)
- dvb_guess_user_country
 - Ancillary functions and macros, [65](#)
- dvb_hexdump
 - Digital TV table parsing, [97](#)
- dvb_logfunc
 - Ancillary functions and macros, [56](#)
- dvb_logfunc_priv
 - dvb-log.h, [367](#)
- DVB_MAX_PAYLOAD_PACKET_SIZE
 - Digital TV table parsing, [75](#)
- DVB_MPEG_ES_FRAME_B
 - Digital TV table parsing, [91](#)
- DVB_MPEG_ES_FRAME_D
 - Digital TV table parsing, [91](#)
- DVB_MPEG_ES_FRAME_I
 - Digital TV table parsing, [91](#)

- dvb_mpeg_es_frame_names
 - Digital TV table parsing, [107](#)
- DVB_MPEG_ES_FRAME_P
 - Digital TV table parsing, [91](#)
- dvb_mpeg_es_frame_t
 - Digital TV table parsing, [90](#)
- DVB_MPEG_ES_FRAME_UNKNOWN
 - Digital TV table parsing, [91](#)
- DVB_MPEG_ES_GOP
 - Digital TV table parsing, [75](#)
- DVB_MPEG_ES_PIC_START
 - Digital TV table parsing, [75](#)
- dvb_mpeg_es_pic_start, [220](#)
 - bitfield, [221](#)
 - bitfield2, [221](#)
 - coding_type, [221](#)
 - dummy, [221](#)
 - sync, [221](#)
 - temporal_ref, [222](#)
 - type, [222](#)
 - vbv_delay, [222](#)
- dvb_mpeg_es_pic_start_init
 - Digital TV table parsing, [97](#)
- dvb_mpeg_es_pic_start_print
 - Digital TV table parsing, [98](#)
- DVB_MPEG_ES_SEQ_EXT
 - Digital TV table parsing, [76](#)
- DVB_MPEG_ES_SEQ_START
 - Digital TV table parsing, [76](#)
- dvb_mpeg_es_seq_start, [222](#)
 - aspect, [223](#)
 - bitfield, [224](#)
 - bitfield2, [224](#)
 - bitfield3, [224](#)
 - bitrate, [224](#)
 - constrained, [224](#)
 - framerate, [224](#)
 - height, [224](#)
 - one, [224](#)
 - qm_intra, [225](#)
 - qm_nonintra, [225](#)
 - sync, [225](#)
 - type, [225](#)
 - vbv, [225](#)
 - width, [225](#)
- dvb_mpeg_es_seq_start_init
 - Digital TV table parsing, [98](#)
- dvb_mpeg_es_seq_start_print
 - Digital TV table parsing, [98](#)
- DVB_MPEG_ES_SLICES
 - Digital TV table parsing, [76](#)
- DVB_MPEG_ES_USER_DATA
 - Digital TV table parsing, [76](#)
- DVB_MPEG_PES
 - Digital TV table parsing, [76](#)
- dvb_mpeg_pes, [225](#)
 - bitfield, [226](#)
 - length, [226](#)
 - optional, [226](#)
 - stream_id, [227](#)
 - sync, [227](#)
- DVB_MPEG_PES_AUDIO
 - Digital TV table parsing, [76](#)
- dvb_mpeg_pes_free
 - Digital TV table parsing, [99](#)
- dvb_mpeg_pes_init
 - Digital TV table parsing, [99](#)
- dvb_mpeg_pes_optional, [227](#)
 - additional_copy_info, [228](#)
 - bitfield, [228](#)
 - copyright, [228](#)
 - data_alignment_indicator, [228](#)
 - DSM_trick_mode, [229](#)
 - dts, [229](#)
 - ES_rate, [229](#)
 - ESCR, [229](#)
 - length, [229](#)
 - original_or_copy, [229](#)
 - PES_CRC, [229](#)
 - PES_extension, [229](#)
 - PES_priority, [230](#)
 - PES_scrambling_control, [230](#)
 - pts, [230](#)
 - PTS_DTS, [230](#)
 - two, [230](#)
- dvb_mpeg_pes_print
 - Digital TV table parsing, [99](#)
- DVB_MPEG_PES_VIDEO
 - Digital TV table parsing, [77](#)
- DVB_MPEG_STREAM_DIRECTORY
 - Digital TV table parsing, [77](#)
- DVB_MPEG_STREAM_DSMCC
 - Digital TV table parsing, [77](#)
- DVB_MPEG_STREAM_ECM
 - Digital TV table parsing, [77](#)
- DVB_MPEG_STREAM_EMM
 - Digital TV table parsing, [77](#)
- DVB_MPEG_STREAM_H222E
 - Digital TV table parsing, [77](#)
- DVB_MPEG_STREAM_MAP
 - Digital TV table parsing, [78](#)
- DVB_MPEG_STREAM_PADDING
 - Digital TV table parsing, [78](#)
- DVB_MPEG_STREAM_PRIVATE_2
 - Digital TV table parsing, [78](#)
- DVB_MPEG_TS
 - Digital TV table parsing, [78](#)
- dvb_mpeg_ts, [230](#)
 - adaptation_field, [232](#)
 - adaption, [232](#)
 - bitfield, [232](#)
 - continuity_counter, [232](#)
 - payload, [232](#)
 - payload_start, [232](#)
 - pid, [232](#)
 - priority, [232](#)

- scrambling, [233](#)
- sync_byte, [233](#)
- tei, [233](#)
- dvb_mpeg_ts_adaption, [233](#)
 - data, [234](#)
 - discontinued, [234](#)
 - extension, [234](#)
 - length, [234](#)
 - OPCR, [234](#)
 - PCR, [235](#)
 - priority, [235](#)
 - private_data, [235](#)
 - random_access, [235](#)
 - splicing_point, [235](#)
- dvb_mpeg_ts_free
 - Digital TV table parsing, [99](#)
- dvb_mpeg_ts_init
 - Digital TV table parsing, [100](#)
- DVB_MPEG_TS_PACKET_SIZE
 - Digital TV table parsing, [78](#)
- dvb_mpeg_ts_print
 - Digital TV table parsing, [100](#)
- dvb_nit_transport_foreach
 - Digital TV table parsing, [78](#)
- dvb_open_descriptor, [235](#)
- dvb_parse_delsys
 - Channel and transponder file read/write, [137](#)
- dvb_parse_file, [236](#)
 - delimiter, [236](#)
 - formats, [236](#)
 - has_delsys_id, [237](#)
- dvb_parse_format
 - Channel and transponder file read/write, [138](#)
- dvb_parse_format_online
 - Channel and transponder file read/write, [138](#)
- dvb_parse_struct, [237](#)
 - delsys, [237](#)
 - id, [237](#)
 - size, [238](#)
 - table, [238](#)
- dvb_parse_table, [238](#)
 - default_value, [239](#)
 - has_default_value, [239](#)
 - mult_factor, [239](#)
 - prop, [239](#)
 - size, [239](#)
 - table, [239](#)
- dvb_pat_program_foreach
 - Digital TV table parsing, [79](#)
- dvb_pmt_field_first
 - pmt.h, [401](#)
- dvb_pmt_field_last
 - pmt.h, [401](#)
- dvb_pmt_stream_foreach
 - Digital TV table parsing, [79](#)
- dvb_print_all_lnb
 - Satellite Equipment Control, [51](#)
- dvb_print_lnb
 - Satellite Equipment Control, [51](#)
- DVB_QUAL_GOOD
 - Digital TV frontend control, [29](#)
- DVB_QUAL_OK
 - Digital TV frontend control, [29](#)
- DVB_QUAL_POOR
 - Digital TV frontend control, [29](#)
- DVB_QUAL_UNKNOWN
 - Digital TV frontend control, [29](#)
- dvb_quality
 - Digital TV frontend control, [28](#)
- dvb_read_file
 - Channel and transponder file read/write, [139](#)
- dvb_read_file_format
 - Channel and transponder file read/write, [139](#)
- dvb_read_section
 - Digital TV frontend scan, [46](#)
- dvb_read_section_with_id
 - Digital TV frontend scan, [47](#)
- dvb_read_sections
 - Digital TV frontend scan, [47](#)
- dvb_retrieve_entry_prop
 - Channel and transponder file read/write, [139](#)
- dvb_sat_get_lnb
 - Satellite Equipment Control, [51](#)
- dvb_sat_get_lnb_name
 - Satellite Equipment Control, [52](#)
- dvb_sat_lnb, [239](#)
 - alias, [240](#)
 - freqrange, [240](#)
 - highfreq, [240](#)
 - lowfreq, [241](#)
 - name, [241](#)
 - rangeswitch, [241](#)
- dvb_sat_lnb::dvbsat_freqrange, [279](#)
 - high, [279](#)
 - low, [279](#)
- dvb_sat_polarization
 - Satellite Equipment Control, [50](#)
- dvb_sat_real_freq
 - Satellite Equipment Control, [52](#)
- dvb_sat_search_lnb
 - Satellite Equipment Control, [52](#)
- dvb_sat_set_parms
 - Satellite Equipment Control, [53](#)
- dvb_scan_alloc_handler_table
 - Digital TV frontend scan, [48](#)
- dvb_scan_free_handler_table
 - Digital TV frontend scan, [48](#)
- dvb_scan_transponder
 - Digital TV frontend scan, [48](#)
- dvb_sdt_service_foreach
 - Digital TV table parsing, [79](#)
- dvb_set_compat_delivery_system
 - Digital TV frontend control, [41](#)
- dvb_set_pesfilter
 - Digital TV demux, [134](#)
- dvb_set_section_filter

- Digital TV demux, [134](#)
- dvb_set_sys
 - Digital TV frontend control, [42](#)
- dvb_store_channel
 - Channel and transponder file read/write, [140](#)
- dvb_store_entry_prop
 - Channel and transponder file read/write, [141](#)
- dvb_streams
 - Digital TV table parsing, [91](#)
- DVB_TABLE_CAT
 - Digital TV table parsing, [80](#)
- dvb_table_cat, [241](#)
 - descriptor, [242](#)
 - header, [242](#)
- dvb_table_cat_free
 - cat.h, [291](#)
- dvb_table_cat_init
 - cat.h, [291](#)
- DVB_TABLE_CAT_PID
 - Digital TV table parsing, [80](#)
- dvb_table_cat_print
 - cat.h, [292](#)
- DVB_TABLE_EIT
 - Digital TV table parsing, [80](#)
- dvb_table_eit, [242](#)
 - event, [243](#)
 - header, [243](#)
 - last_segment, [243](#)
 - last_table_id, [243](#)
 - network_id, [243](#)
 - transport_id, [243](#)
- dvb_table_eit_event, [244](#)
 - bitfield1, [245](#)
 - bitfield2, [245](#)
 - desc_length, [245](#)
 - descriptor, [245](#)
 - duration, [245](#)
 - dvbduration, [245](#)
 - dvbstart, [246](#)
 - event_id, [246](#)
 - free_CA_mode, [246](#)
 - next, [246](#)
 - running_status, [246](#)
 - service_id, [246](#)
 - start, [246](#)
- dvb_table_eit_free
 - Digital TV table parsing, [100](#)
- dvb_table_eit_init
 - Digital TV table parsing, [101](#)
- DVB_TABLE_EIT_OTHER
 - Digital TV table parsing, [80](#)
- DVB_TABLE_EIT_PID
 - Digital TV table parsing, [80](#)
- dvb_table_eit_print
 - Digital TV table parsing, [101](#)
- DVB_TABLE_EIT_SCHEDULE
 - Digital TV table parsing, [80](#)
- DVB_TABLE_EIT_SCHEDULE_OTHER
 - Digital TV table parsing, [80](#)
- dvb_table_filter, [247](#)
 - allow_section_gaps, [247](#)
 - pid, [247](#)
 - priv, [247](#)
 - table, [248](#)
 - tid, [248](#)
 - ts_id, [248](#)
- dvb_table_filter_free
 - Digital TV frontend scan, [49](#)
- dvb_table_header, [248](#)
 - bitfield, [249](#)
 - current_next, [249](#)
 - id, [249](#)
 - last_section, [249](#)
 - one, [249](#)
 - one2, [250](#)
 - section_id, [250](#)
 - section_length, [250](#)
 - syntax, [250](#)
 - table_id, [250](#)
 - version, [250](#)
 - zero, [250](#)
- dvb_table_header_init
 - Digital TV table parsing, [101](#)
- dvb_table_header_print
 - Digital TV table parsing, [102](#)
- dvb_table_init_func
 - Digital TV table parsing, [85](#)
- dvb_table_initializers
 - Digital TV table parsing, [107](#)
- DVB_TABLE_NIT
 - Digital TV table parsing, [81](#)
- dvb_table_nit, [251](#)
 - bitfield, [251](#)
 - desc_length, [252](#)
 - descriptor, [252](#)
 - header, [252](#)
 - reserved, [252](#)
 - transport, [252](#)
- DVB_TABLE_NIT2
 - Digital TV table parsing, [81](#)
- dvb_table_nit_descriptor_handler
 - Digital TV table parsing, [102](#)
- dvb_table_nit_free
 - Digital TV table parsing, [102](#)
- dvb_table_nit_init
 - Digital TV table parsing, [103](#)
- DVB_TABLE_NIT_PID
 - Digital TV table parsing, [81](#)
- dvb_table_nit_print
 - Digital TV table parsing, [103](#)
- dvb_table_nit_transport, [252](#)
 - bitfield, [253](#)
 - desc_length, [253](#)
 - descriptor, [253](#)
 - network_id, [253](#)
 - next, [253](#)

- reserved, 254
- transport_id, 254
- dvb_table_nit_transport_header, 254
 - bitfield, 255
 - reserved, 255
 - transport_length, 255
- DVB_TABLE_PAT
 - Digital TV table parsing, 81
- dvb_table_pat, 255
 - header, 256
 - program, 256
 - programs, 256
- dvb_table_pat_free
 - Digital TV table parsing, 103
- dvb_table_pat_init
 - Digital TV table parsing, 104
- DVB_TABLE_PAT_PID
 - Digital TV table parsing, 81
- dvb_table_pat_print
 - Digital TV table parsing, 104
- dvb_table_pat_program, 256
 - bitfield, 257
 - next, 257
 - pid, 258
 - reserved, 258
 - service_id, 258
- DVB_TABLE_PMT
 - Digital TV table parsing, 81
- dvb_table_pmt, 258
 - bitfield, 259
 - bitfield2, 259
 - desc_length, 259
 - descriptor, 259
 - header, 260
 - pcr_pid, 260
 - reserved2, 260
 - reserved3, 260
 - stream, 260
 - zero3, 260
- dvb_table_pmt_free
 - Digital TV table parsing, 104
- dvb_table_pmt_init
 - Digital TV table parsing, 105
- dvb_table_pmt_print
 - Digital TV table parsing, 105
- dvb_table_pmt_stream, 260
 - bitfield, 262
 - bitfield2, 262
 - desc_length, 262
 - descriptor, 262
 - elementary_pid, 262
 - next, 262
 - reserved, 262
 - reserved2, 262
 - type, 263
 - zero, 263
- DVB_TABLE_SDT
 - Digital TV table parsing, 82
- dvb_table_sdt, 263
 - header, 264
 - network_id, 264
 - reserved, 264
 - service, 264
- DVB_TABLE_SDT2
 - Digital TV table parsing, 82
- dvb_table_sdt_free
 - Digital TV table parsing, 105
- dvb_table_sdt_init
 - Digital TV table parsing, 106
- DVB_TABLE_SDT_PID
 - Digital TV table parsing, 82
- dvb_table_sdt_print
 - Digital TV table parsing, 106
- dvb_table_sdt_service, 264
 - bitfield, 265
 - desc_length, 266
 - descriptor, 266
 - EIT_present_following, 266
 - EIT_schedule, 266
 - free_CA_mode, 266
 - next, 266
 - reserved, 266
 - running_status, 266
 - service_id, 267
- dvb_time
 - Digital TV table parsing, 106
- dvb_ts_packet_header, 267
 - adaptation_field_control, 268
 - adaptation_field_length, 268
 - bitfield, 269
 - continuity_counter, 269
 - discontinued, 269
 - extension, 269
 - OPCR, 269
 - payload_unit_start_indicator, 269
 - PCR, 270
 - pid, 270
 - priority, 270
 - private_data, 270
 - random_access, 270
 - splicing_point, 270
 - sync_byte, 270
 - transport_error_indicator, 271
 - transport_priority, 271
 - transport_scrambling_control, 271
- dvb_type
 - dvb_dev_list, 208
- dvb_v5_descriptors, 271
 - delivery_system, 272
 - entry, 272
 - nit, 272
 - num_entry, 273
 - num_other_nits, 273
 - num_other_sdt, 273
 - num_program, 273
 - other_nits, 273

- other_sdt, [273](#)
 - pat, [273](#)
 - program, [273](#)
 - sdt, [274](#)
 - vct, [274](#)
- dvb_v5_descriptors_program, [274](#)
 - pat_pgm, [274](#)
 - pmt, [275](#)
- dvb_v5_fe_parms, [275](#)
 - abort, [276](#)
 - current_sys, [276](#)
 - default_charset, [277](#)
 - diseqc_wait, [277](#)
 - freq_bpf, [277](#)
 - has_v5_stats, [277](#)
 - info, [277](#)
 - legacy_fe, [277](#)
 - lna, [278](#)
 - lnb, [278](#)
 - logfunc, [278](#)
 - num_systems, [278](#)
 - output_charset, [278](#)
 - sat_number, [278](#)
 - systems, [278](#)
 - verbose, [279](#)
 - version, [279](#)
- dvb_write_file
 - Channel and transponder file read/write, [141](#)
- dvb_write_file_format
 - Channel and transponder file read/write, [142](#)
- dvb_write_format_online
 - Channel and transponder file read/write, [142](#)
- dvb_write_format_vdr
 - Channel and transponder file read/write, [142](#)
- dvbc_fec_table
 - desc_cable_delivery.h, [307](#)
- dvbc_modulation_table
 - desc_cable_delivery.h, [307](#)
- dvbduration
 - dvb_table_eit_event, [245](#)
- dvbs_dvbc_dvbs_freq_inner
 - desc_sat.h, [331](#)
- dvbs_modulation
 - desc_sat.h, [331](#)
- dvbs_polarization
 - desc_sat.h, [331](#)
- dvbs_rolloff
 - desc_sat.h, [331](#)
- dvbstart
 - dvb_table_eit_event, [246](#)
- dvbt2_bw
 - desc_t2_delivery.h, [336](#)
- dvbt2_interval
 - desc_t2_delivery.h, [336](#)
- dvbt2_transmission_mode
 - desc_t2_delivery.h, [336](#)
- dvbt_bw
 - desc_terrestrial_delivery.h, [339](#)
- dvbt_code_rate
 - desc_terrestrial_delivery.h, [339](#)
- dvbt_hierarchy
 - desc_terrestrial_delivery.h, [339](#)
- dvbt_interval
 - desc_terrestrial_delivery.h, [339](#)
- dvbt_modulation
 - desc_terrestrial_delivery.h, [340](#)
- dvbt_transmission_mode
 - desc_terrestrial_delivery.h, [340](#)
- DZ
 - Ancillary functions and macros, [57](#)
- EC
 - Ancillary functions and macros, [57](#)
- ECM_repetition_rate_descriptor
 - Digital TV table parsing, [89](#)
- EE
 - Ancillary functions and macros, [57](#)
- EG
 - Ancillary functions and macros, [58](#)
- EH
 - Ancillary functions and macros, [58](#)
- eit.h
 - dvb_eit_running_status_name, [379](#)
- EIT_present_following
 - dvb_table_sdt_service, [266](#)
- EIT_schedule
 - dvb_table_sdt_service, [266](#)
- elementary
 - atsc_desc_service_location, [144](#)
- elementary_pid
 - atsc_desc_service_location_elementary, [146](#)
 - dvb_table_pmt_stream, [262](#)
- embedded_layer
 - dvb_desc_hierarchy, [180](#)
- emergency_information_descriptor
 - Digital TV table parsing, [90](#)
- enhanced_AC_3_descriptor
 - Digital TV table parsing, [89](#)
- entry
 - dvb_v5_descriptors, [272](#)
- ER
 - Ancillary functions and macros, [58](#)
- ES
 - Ancillary functions and macros, [58](#)
- ES_rate
 - dvb_mpeg_pes_optional, [229](#)
- ESCR
 - dvb_mpeg_pes_optional, [229](#)
- ET
 - Ancillary functions and macros, [58](#)
- etm
 - atsc_table_eit_event, [151](#)
- ETM_location
 - atsc_table_vct_channel, [161](#)
- event
 - atsc_table_eit, [147](#)
 - dvb_table_eit, [243](#)

- event_group_descriptor
 - Digital TV table parsing, [90](#)
- event_id
 - atsc_table_eit_event, [151](#)
 - dvb_table_eit_event, [246](#)
- events
 - atsc_table_eit, [147](#)
- extended_broadcaster_descriptor
 - Digital TV table parsing, [89](#)
- extended_channel_name
 - Digital TV table parsing, [89](#)
- extended_event_descriptor
 - Digital TV table parsing, [87](#)
- extension
 - dvb_mpeg_ts_adaption, [234](#)
 - dvb_ts_packet_header, [269](#)
- extension_code
 - dvb_extension_descriptor, [218](#)
- extension_descriptor
 - Digital TV table parsing, [89](#)
- extension_descriptors
 - Parsers for several MPEG-TS descriptors, [112](#)
- external_es_id_descriptor
 - Digital TV table parsing, [86](#)
- fe_parms
 - dvb_device, [209](#)
- fec
 - dvb_desc_sat, [189](#)
- fec_inner
 - dvb_desc_cable_delivery, [170](#)
- fec_outer
 - dvb_desc_cable_delivery, [170](#)
- FI
 - Ancillary functions and macros, [58](#)
- FILE_CHANNEL
 - Channel and transponder file read/write, [137](#)
- FILE_DVBV5
 - Channel and transponder file read/write, [137](#)
- FILE_UNKNOWN
 - Channel and transponder file read/write, [137](#)
- FILE_VDR
 - Channel and transponder file read/write, [137](#)
- FILE_ZAP
 - Channel and transponder file read/write, [137](#)
- first_entry
 - dvb_file, [219](#)
- FJ
 - Ancillary functions and macros, [58](#)
- FK
 - Ancillary functions and macros, [58](#)
- flexmux_timing_descriptor
 - Digital TV table parsing, [87](#)
- FM
 - Ancillary functions and macros, [58](#)
- fmc_descriptor
 - Digital TV table parsing, [86](#)
- fmxbuffersize_descriptor
 - Digital TV table parsing, [87](#)
- fname
 - dvb_file, [220](#)
- FO
 - Ancillary functions and macros, [58](#)
- format_identifier
 - dvb_desc_registration, [187](#)
- formats
 - dvb_parse_file, [236](#)
- FR
 - Ancillary functions and macros, [58](#)
- framerate
 - dvb_mpeg_es_seq_start, [224](#)
- free
 - dvb_descriptor, [206](#)
 - dvb_ext_descriptor, [217](#)
- free_CA_mode
 - dvb_table_eit_event, [246](#)
 - dvb_table_sdt_service, [266](#)
- freq_bpf
 - dvb_entry, [213](#)
 - dvb_v5_fe_parms, [277](#)
- freq_type
 - dvb_desc_frequency_list, [178](#)
- freqrange
 - dvb_sat_lnb, [240](#)
- frequencies
 - dvb_desc_frequency_list, [178](#)
- frequency
 - dvb_desc_cable_delivery, [170](#)
 - dvb_desc_frequency_list, [178](#)
 - dvb_desc_sat, [189](#)
 - isdbt_desc_terrestrial_delivery_system, [283](#)
- frequency_list_descriptor
 - Digital TV table parsing, [88](#)
- frequency_loop_length
 - dvb_desc_t2_delivery, [194](#)
- FTA_content_management_descriptor
 - Digital TV table parsing, [89](#)
- GA
 - Ancillary functions and macros, [58](#)
- GB
 - Ancillary functions and macros, [58](#)
- GD
 - Ancillary functions and macros, [58](#)
- GE
 - Ancillary functions and macros, [58](#)
- GF
 - Ancillary functions and macros, [58](#)
- GG
 - Ancillary functions and macros, [58](#)
- GH
 - Ancillary functions and macros, [58](#)
- GI
 - Ancillary functions and macros, [58](#)
- GL
 - Ancillary functions and macros, [58](#)
- GM
 - Ancillary functions and macros, [58](#)

- GN
 - Ancillary functions and macros, [58](#)
- GP
 - Ancillary functions and macros, [58](#)
- GQ
 - Ancillary functions and macros, [58](#)
- GR
 - Ancillary functions and macros, [58](#)
- GS
 - Ancillary functions and macros, [58](#)
- GT
 - Ancillary functions and macros, [58](#)
- GU
 - Ancillary functions and macros, [58](#)
- guard_interval
 - dvb_desc_t2_delivery, [194](#)
 - dvb_desc_terrestrial_delivery, [201](#)
 - isdbt_desc_terrestrial_delivery_system, [283](#)
- GW
 - Ancillary functions and macros, [58](#)
- GY
 - Ancillary functions and macros, [58](#)
- has_default_value
 - dvb_parse_table, [239](#)
- has_delsys_id
 - dvb_parse_file, [237](#)
- has_v5_stats
 - dvb_v5_fe_parms, [277](#)
- header
 - atsc_table_eit, [147](#)
 - atsc_table_mgt, [153](#)
 - atsc_table_vct, [157](#)
 - dvb_table_cat, [242](#)
 - dvb_table_eit, [243](#)
 - dvb_table_nit, [252](#)
 - dvb_table_pat, [256](#)
 - dvb_table_pmt, [260](#)
 - dvb_table_sdt, [264](#)
- height
 - dvb_mpeg_es_seq_start, [224](#)
- hidden
 - atsc_table_vct_channel, [161](#)
- hide_guide
 - atsc_table_vct_channel, [161](#)
- hierarchical_transmission_descriptor
 - Digital TV table parsing, [89](#)
- hierarchy_descriptor
 - Digital TV table parsing, [86](#)
- hierarchy_information
 - dvb_desc_terrestrial_delivery, [201](#)
- hierarchy_type
 - dvb_desc_hierarchy, [180](#)
- high
 - dvb_sat_lnb::dvbsat_freqlange, [279](#)
- highfreq
 - dvb_sat_lnb, [240](#)
- HK
 - Ancillary functions and macros, [58](#)
- HM
 - Ancillary functions and macros, [58](#)
- HN
 - Ancillary functions and macros, [58](#)
- HR
 - Ancillary functions and macros, [58](#)
- HT
 - Ancillary functions and macros, [58](#)
- HU
 - Ancillary functions and macros, [58](#)
- hyperlink_descriptor
 - Digital TV table parsing, [89](#)
- ibp_descriptor
 - Digital TV table parsing, [86](#)
- ID
 - Ancillary functions and macros, [58](#)
- id
 - dvb_desc_event_extended, [172](#)
 - dvb_parse_struct, [237](#)
 - dvb_table_header, [249](#)
- ids
 - dvb_desc_event_extended, [172](#)
- IE
 - Ancillary functions and macros, [58](#)
- IL
 - Ancillary functions and macros, [58](#)
- IM
 - Ancillary functions and macros, [58](#)
- image_icon_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- IN
 - Ancillary functions and macros, [58](#)
- info
 - dvb_v5_fe_parms, [277](#)
- init
 - dvb_descriptor, [206](#)
 - dvb_ext_descriptor, [217](#)
- IO
 - Ancillary functions and macros, [58](#)
- iod_descriptor
 - Digital TV table parsing, [86](#)
- ipmp_descriptor
 - Digital TV table parsing, [87](#)
- IQ
 - Ancillary functions and macros, [58](#)
- IR
 - Ancillary functions and macros, [58](#)
- IS
 - Ancillary functions and macros, [58](#)
- isdb_desc_partial_reception, [280](#)
 - length, [280](#)
 - next, [280](#)
 - partial_reception, [281](#)
 - type, [281](#)
- isdb_desc_partial_reception_free
 - Parsers for several MPEG-TS descriptors, [130](#)
- isdb_desc_partial_reception_init
 - Parsers for several MPEG-TS descriptors, [130](#)

- isdb_desc_partial_reception_print
 - Parsers for several MPEG-TS descriptors, [130](#)
- isdb_partial_reception_service_id, [281](#)
 - service_id, [281](#)
- ISDBT_delivery_system_descriptor
 - Digital TV table parsing, [90](#)
- isdbt_desc_delivery_free
 - Parsers for several MPEG-TS descriptors, [131](#)
- isdbt_desc_delivery_init
 - Parsers for several MPEG-TS descriptors, [131](#)
- isdbt_desc_delivery_print
 - Parsers for several MPEG-TS descriptors, [131](#)
- isdbt_desc_terrestrial_delivery_system, [282](#)
 - area_code, [283](#)
 - bitfield, [283](#)
 - frequency, [283](#)
 - guard_interval, [283](#)
 - length, [283](#)
 - next, [283](#)
 - num_freqs, [283](#)
 - transmission_mode, [283](#)
 - type, [284](#)
- isdbt_interval
 - desc_isdbt_delivery.h, [319](#)
- isdbt_mode
 - desc_isdbt_delivery.h, [319](#)
- iso639_language_descriptor
 - Digital TV table parsing, [86](#)
- ISO_639_language_code
 - atsc_desc_service_location_elementary, [146](#)
- IT
 - Ancillary functions and macros, [58](#)
- item
 - dvb_desc_event_extended_item, [174](#)
- item_emph
 - dvb_desc_event_extended_item, [174](#)
- items
 - dvb_desc_event_extended, [172](#)
- JE
 - Ancillary functions and macros, [58](#)
- JM
 - Ancillary functions and macros, [58](#)
- JO
 - Ancillary functions and macros, [58](#)
- JP
 - Ancillary functions and macros, [59](#)
- KE
 - Ancillary functions and macros, [59](#)
- KG
 - Ancillary functions and macros, [59](#)
- KH
 - Ancillary functions and macros, [59](#)
- KI
 - Ancillary functions and macros, [59](#)
- KM
 - Ancillary functions and macros, [59](#)
- KN
 - Ancillary functions and macros, [59](#)
- KP
 - Ancillary functions and macros, [59](#)
- KR
 - Ancillary functions and macros, [59](#)
- KW
 - Ancillary functions and macros, [59](#)
- KY
 - Ancillary functions and macros, [59](#)
- KZ
 - Ancillary functions and macros, [59](#)
- LA
 - Ancillary functions and macros, [59](#)
- language
 - dvb_desc_event_extended, [172](#)
 - dvb_desc_event_short, [176](#)
 - dvb_desc_language, [182](#)
- last_id
 - dvb_desc_event_extended, [172](#)
- last_section
 - dvb_table_header, [249](#)
- last_segment
 - dvb_table_eit, [243](#)
- last_table_id
 - dvb_table_eit, [243](#)
- layer
 - dvb_desc_hierarchy, [180](#)
- LB
 - Ancillary functions and macros, [59](#)
- LC
 - Ancillary functions and macros, [59](#)
- lcn
 - dvb_desc_logical_channel, [183](#)
- LDT_linkage_descriptor
 - Digital TV table parsing, [90](#)
- legacy_fe
 - dvb_v5_fe_parms, [277](#)
- length
 - atsc_desc_service_location, [144](#)
 - dvb_desc, [164](#)
 - dvb_desc_ca, [166](#)
 - dvb_desc_ca_identifier, [168](#)
 - dvb_desc_cable_delivery, [170](#)
 - dvb_desc_event_extended, [173](#)
 - dvb_desc_event_short, [176](#)
 - dvb_desc_frequency_list, [178](#)
 - dvb_desc_hierarchy, [180](#)
 - dvb_desc_language, [182](#)
 - dvb_desc_logical_channel, [183](#)
 - dvb_desc_network_name, [186](#)
 - dvb_desc_registration, [187](#)
 - dvb_desc_sat, [189](#)
 - dvb_desc_service, [191](#)
 - dvb_desc_terrestrial_delivery, [201](#)
 - dvb_desc_ts_info, [203](#)
 - dvb_extension_descriptor, [218](#)
 - dvb_mpeg_pes, [226](#)
 - dvb_mpeg_pes_optional, [229](#)

- dvb_mpeg_ts_adaption, [234](#)
 - isdb_desc_partial_reception, [280](#)
 - isdbt_desc_terrestrial_delivery_system, [283](#)
- length_of_ts_name
 - dvb_desc_ts_info, [203](#)
- LI
 - Ancillary functions and macros, [59](#)
- lib/include/libdvbv5/atsc_eit.h, [286](#), [288](#)
- lib/include/libdvbv5/atsc_header.h, [289](#), [290](#)
- lib/include/libdvbv5/cat.h, [290](#), [292](#)
- lib/include/libdvbv5/countries.h, [293](#), [295](#)
- lib/include/libdvbv5/crc32.h, [298](#), [299](#)
- lib/include/libdvbv5/desc_atsc_service_location.h, [300](#), [301](#)
- lib/include/libdvbv5/desc_ca.h, [301](#), [303](#)
- lib/include/libdvbv5/desc_ca_identifier.h, [304](#), [306](#)
- lib/include/libdvbv5/desc_cable_delivery.h, [306](#), [308](#)
- lib/include/libdvbv5/desc_event_extended.h, [309](#), [310](#)
- lib/include/libdvbv5/desc_event_short.h, [310](#), [312](#)
- lib/include/libdvbv5/desc_extension.h, [312](#), [314](#)
- lib/include/libdvbv5/desc_frequency_list.h, [315](#), [316](#)
- lib/include/libdvbv5/desc_hierarchy.h, [316](#), [317](#)
- lib/include/libdvbv5/desc_isdbt_delivery.h, [318](#), [320](#)
- lib/include/libdvbv5/desc_language.h, [320](#), [322](#)
- lib/include/libdvbv5/desc_logical_channel.h, [322](#), [324](#)
- lib/include/libdvbv5/desc_network_name.h, [324](#), [326](#)
- lib/include/libdvbv5/desc_partial_reception.h, [326](#), [328](#)
- lib/include/libdvbv5/desc_registration_id.h, [328](#), [329](#)
- lib/include/libdvbv5/desc_sat.h, [330](#), [332](#)
- lib/include/libdvbv5/desc_service.h, [333](#), [334](#)
- lib/include/libdvbv5/desc_t2_delivery.h, [334](#), [336](#)
- lib/include/libdvbv5/desc_terrestrial_delivery.h, [337](#), [340](#)
- lib/include/libdvbv5/desc_ts_info.h, [341](#), [342](#)
- lib/include/libdvbv5/descriptors.h, [343](#), [346](#)
- lib/include/libdvbv5/dvb-demux.h, [350](#), [351](#)
- lib/include/libdvbv5/dvb-dev.h, [351](#), [355](#)
- lib/include/libdvbv5/dvb-fe.h, [357](#), [360](#)
- lib/include/libdvbv5/dvb-file.h, [362](#), [364](#)
- lib/include/libdvbv5/dvb-log.h, [367](#), [368](#)
- lib/include/libdvbv5/dvb-sat.h, [368](#), [370](#)
- lib/include/libdvbv5/dvb-scan.h, [370](#), [372](#)
- lib/include/libdvbv5/dvb-v5-std.h, [374](#), [376](#)
- lib/include/libdvbv5/eit.h, [378](#), [380](#)
- lib/include/libdvbv5/header.h, [381](#), [382](#)
- lib/include/libdvbv5/mgt.h, [383](#), [385](#)
- lib/include/libdvbv5/mpeg_es.h, [386](#), [387](#)
- lib/include/libdvbv5/mpeg_pes.h, [389](#), [390](#)
- lib/include/libdvbv5/mpeg_ts.h, [392](#), [393](#)
- lib/include/libdvbv5/nit.h, [394](#), [396](#)
- lib/include/libdvbv5/pat.h, [397](#), [399](#)
- lib/include/libdvbv5/pmt.h, [399](#), [402](#)
- lib/include/libdvbv5/sdt.h, [403](#), [405](#)
- lib/include/libdvbv5/vct.h, [406](#), [407](#)
- linkage_descriptor
 - Digital TV table parsing, [87](#)
- LK
 - Ancillary functions and macros, [59](#)
- Ina
 - dvb_v5_fe_parms, [278](#)
- Inb
 - dvb_entry, [213](#)
 - dvb_v5_fe_parms, [278](#)
- local_time_offset_descriptor
 - Digital TV table parsing, [87](#)
- location
 - dvb_entry, [213](#)
- logfunc
 - dvb_v5_fe_parms, [278](#)
- logical_channel_number
 - dvb_desc_logical_channel_number, [184](#)
- logical_channel_number_descriptor
 - Digital TV table parsing, [89](#)
- logo_transmission_descriptor
 - Digital TV table parsing, [89](#)
- low
 - dvb_sat_Inb::dvbsat_freqrange, [279](#)
- lowfreq
 - dvb_sat_Inb, [241](#)
- LR
 - Ancillary functions and macros, [59](#)
- LS
 - Ancillary functions and macros, [59](#)
- LT
 - Ancillary functions and macros, [59](#)
- LU
 - Ancillary functions and macros, [59](#)
- LV
 - Ancillary functions and macros, [59](#)
- LY
 - Ancillary functions and macros, [59](#)
- MA
 - Ancillary functions and macros, [59](#)
- major_channel_number
 - atsc_table_vct_channel, [161](#)
- manufacturer
 - dvb_dev_list, [208](#)
- MAX_DELIVERY_SYSTEMS
 - Digital TV frontend control, [28](#)
- MAX_TABLE_SIZE
 - dvb-scan.h, [372](#)
- maximum_bitrate_descriptor
 - Digital TV table parsing, [86](#)
- MC
 - Ancillary functions and macros, [59](#)
- MD
 - Ancillary functions and macros, [59](#)
- ME
 - Ancillary functions and macros, [59](#)
- message_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- metadata_descriptor
 - Digital TV table parsing, [87](#)
- metadata_pointer_descriptor
 - Digital TV table parsing, [87](#)
- metadata_std_descriptor
 - Digital TV table parsing, [87](#)

- MF
 - Ancillary functions and macros, [59](#)
- MG
 - Ancillary functions and macros, [59](#)
- mgt.h
 - atsc_mgt_table_foreach, [384](#)
- MH
 - Ancillary functions and macros, [59](#)
- minor_channel_number
 - atsc_table_vct_channel, [161](#)
- MK
 - Ancillary functions and macros, [59](#)
- ML
 - Ancillary functions and macros, [59](#)
- MM
 - Ancillary functions and macros, [59](#)
- MN
 - Ancillary functions and macros, [59](#)
- MO
 - Ancillary functions and macros, [59](#)
- modulation
 - dvb_desc_cable_delivery, [170](#)
- modulation_mode
 - atsc_table_vct_channel, [161](#)
- modulation_system
 - dvb_desc_sat, [189](#)
- modulation_type
 - dvb_desc_sat, [189](#)
- mosaic_descriptor
 - Digital TV table parsing, [87](#)
- MP
 - Ancillary functions and macros, [59](#)
- mpe_fec_indicator
 - dvb_desc_terrestrial_delivery, [201](#)
- mpeg2_aac_audio_descriptor
 - Digital TV table parsing, [87](#)
- mpeg4_audio_descriptor
 - Digital TV table parsing, [86](#)
- mpeg4_video_descriptor
 - Digital TV table parsing, [86](#)
- MQ
 - Ancillary functions and macros, [59](#)
- MR
 - Ancillary functions and macros, [59](#)
- MS
 - Ancillary functions and macros, [59](#)
- MT
 - Ancillary functions and macros, [59](#)
- MU
 - Ancillary functions and macros, [59](#)
- mult_factor
 - dvb_parse_table, [239](#)
- multilingual_bouquet_name_descriptor
 - Digital TV table parsing, [88](#)
- multilingual_component_descriptor
 - Digital TV table parsing, [88](#)
- multilingual_network_name_descriptor
 - Digital TV table parsing, [88](#)
- multilingual_service_name_descriptor
 - Digital TV table parsing, [88](#)
- multiplex_buffer_utilization_descriptor
 - Digital TV table parsing, [86](#)
- multiplexbuffer_descriptor
 - Digital TV table parsing, [87](#)
- muxcode_descriptor
 - Digital TV table parsing, [86](#)
- MV
 - Ancillary functions and macros, [59](#)
- MW
 - Ancillary functions and macros, [59](#)
- MX
 - Ancillary functions and macros, [59](#)
- MY
 - Ancillary functions and macros, [59](#)
- MZ
 - Ancillary functions and macros, [59](#)
- n_entries
 - dvb_file, [220](#)
- n_props
 - dvb_entry, [214](#)
- NA
 - Ancillary functions and macros, [59](#)
- name
 - dvb_desc_event_short, [176](#)
 - dvb_desc_service, [191](#)
 - dvb_descriptor, [206](#)
 - dvb_ext_descriptor, [217](#)
 - dvb_sat_lnb, [241](#)
- name_emph
 - dvb_desc_event_short, [176](#)
 - dvb_desc_service, [191](#)
- NC
 - Ancillary functions and macros, [59](#)
- NE
 - Ancillary functions and macros, [59](#)
- network_change_notify_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- network_id
 - dvb_entry, [214](#)
 - dvb_table_eit, [243](#)
 - dvb_table_nit_transport, [253](#)
 - dvb_table_sdt, [264](#)
- network_identifier_descriptor
 - Digital TV table parsing, [89](#)
- network_name
 - dvb_desc_network_name, [186](#)
- network_name_descriptor
 - Digital TV table parsing, [87](#)
- network_name_emph
 - dvb_desc_network_name, [186](#)
- next
 - atsc_desc_service_location, [144](#)
 - atsc_table_eit_event, [151](#)
 - atsc_table_mgt_table, [155](#)
 - atsc_table_vct_channel, [161](#)
 - dvb_desc, [165](#)

- dvb_desc_ca, 166
- dvb_desc_ca_identifier, 168
- dvb_desc_cable_delivery, 170
- dvb_desc_event_extended, 173
- dvb_desc_event_short, 176
- dvb_desc_frequency_list, 178
- dvb_desc_hierarchy, 180
- dvb_desc_language, 182
- dvb_desc_logical_channel, 183
- dvb_desc_network_name, 186
- dvb_desc_registration, 187
- dvb_desc_sat, 189
- dvb_desc_service, 191
- dvb_desc_terrestrial_delivery, 201
- dvb_desc_ts_info, 203
- dvb_entry, 214
- dvb_extension_descriptor, 218
- dvb_table_eit_event, 246
- dvb_table_nit_transport, 253
- dvb_table_pat_program, 257
- dvb_table_pmt_stream, 262
- dvb_table_sdt_service, 266
- isdb_desc_partial_reception, 280
- isdbt_desc_terrestrial_delivery_system, 283
- NF
 - Ancillary functions and macros, 60
- NG
 - Ancillary functions and macros, 60
- NI
 - Ancillary functions and macros, 60
- nit
 - dvb_v5_descriptors, 272
- nit_handler_callback_t
 - Digital TV table parsing, 85
- nit_tran_handler_callback_t
 - Digital TV table parsing, 85
- NL
 - Ancillary functions and macros, 60
- NO
 - Ancillary functions and macros, 60
- node_relation_descriptor
 - Digital TV table parsing, 90
- NP
 - Ancillary functions and macros, 60
- NR
 - Ancillary functions and macros, 60
- NU
 - Ancillary functions and macros, 60
- num_cell
 - dvb_desc_t2_delivery, 194
- num_channels_in_section
 - atsc_table_vct, 157
- num_devices
 - dvb_device, 210
- num_entry
 - dvb_v5_descriptors, 273
- num_freqs
 - dvb_desc_t2_delivery_cell, 196
- isdbt_desc_terrestrial_delivery_system, 283
- num_items
 - dvb_desc_event_extended, 173
- num_of_service
 - dvb_desc_ts_info_transmission_type, 205
- num_other_nits
 - dvb_v5_descriptors, 273
- num_other_sdt
 - dvb_v5_descriptors, 273
- num_program
 - dvb_v5_descriptors, 273
- num_systems
 - dvb_v5_fe_parms, 278
- number_elements
 - atsc_desc_service_location, 144
- NVOD_reference_descriptor
 - Digital TV table parsing, 87
- NZ
 - Ancillary functions and macros, 60
- OM
 - Ancillary functions and macros, 60
- one
 - atsc_table_eit_event, 151
 - atsc_table_mgt_table, 155
 - dvb_mpeg_es_seq_start, 224
 - dvb_table_header, 249
 - ts_t, 285
- one1
 - ts_t, 285
- one2
 - atsc_table_eit_event, 151
 - atsc_table_mgt_table, 155
 - dvb_table_header, 250
 - ts_t, 286
- one3
 - atsc_table_mgt_table, 156
- OPCR
 - dvb_mpeg_ts_adaption, 234
 - dvb_ts_packet_header, 269
- optional
 - dvb_mpeg_pes, 226
- orbit
 - dvb_desc_sat, 189
- original_or_copy
 - dvb_mpeg_pes_optional, 229
- other_el_pid
 - dvb_entry, 214
- other_el_pid_len
 - dvb_entry, 214
- other_frequency_flag
 - dvb_desc_t2_delivery, 194
 - dvb_desc_terrestrial_delivery, 201
- other_nits
 - dvb_v5_descriptors, 273
- other_sdt
 - dvb_v5_descriptors, 273
- out_of_band
 - atsc_table_vct_channel, 161

- output_charset
 - dvb_v5_fe_parms, 278
- PA
 - Ancillary functions and macros, 60
- parental_rating_descriptor
 - Digital TV table parsing, 87
- Parsers for several MPEG-TS descriptors, 107
 - atsc_desc_service_location_free, 112
 - atsc_desc_service_location_init, 113
 - atsc_desc_service_location_print, 113
 - CP_descriptor, 112
 - CP_identifier_descriptor, 112
 - cpcm_delivery_signalling_descriptor, 112
 - dvb_desc_ca_free, 113
 - dvb_desc_ca_identifier_free, 114
 - dvb_desc_ca_identifier_init, 114
 - dvb_desc_ca_identifier_print, 114
 - dvb_desc_ca_init, 115
 - dvb_desc_ca_print, 115
 - dvb_desc_cable_delivery_init, 115
 - dvb_desc_cable_delivery_print, 116
 - dvb_desc_event_extended_free, 116
 - dvb_desc_event_extended_init, 116
 - dvb_desc_event_extended_print, 117
 - dvb_desc_event_short_free, 117
 - dvb_desc_event_short_init, 117
 - dvb_desc_event_short_print, 118
 - dvb_desc_frequency_list_init, 118
 - dvb_desc_frequency_list_print, 118
 - dvb_desc_hierarchy_init, 119
 - dvb_desc_hierarchy_print, 119
 - dvb_desc_language_init, 119
 - dvb_desc_language_print, 121
 - dvb_desc_logical_channel_free, 121
 - dvb_desc_logical_channel_init, 121
 - dvb_desc_logical_channel_print, 122
 - dvb_desc_network_name_free, 122
 - dvb_desc_network_name_init, 122
 - dvb_desc_network_name_print, 123
 - dvb_desc_registration_free, 123
 - dvb_desc_registration_init, 123
 - dvb_desc_registration_print, 124
 - dvb_desc_sat_init, 124
 - dvb_desc_sat_print, 124
 - dvb_desc_service_free, 125
 - dvb_desc_service_init, 125
 - dvb_desc_service_print, 125
 - dvb_desc_t2_delivery_free, 126
 - dvb_desc_t2_delivery_init, 126
 - dvb_desc_t2_delivery_print, 126
 - dvb_desc_terrestrial_delivery_print, 127
 - dvb_desc_ts_info_free, 127
 - dvb_desc_ts_info_init, 127
 - dvb_desc_ts_info_print, 127
 - dvb_extension_descriptor_free, 128
 - dvb_extension_descriptor_init, 128
 - dvb_extension_descriptor_print, 128
 - extension_descriptors, 112
 - image_icon_descriptor, 112
 - isdb_desc_partial_reception_free, 130
 - isdb_desc_partial_reception_init, 130
 - isdb_desc_partial_reception_print, 130
 - isdbt_desc_delivery_free, 131
 - isdbt_desc_delivery_init, 131
 - isdbt_desc_delivery_print, 131
 - message_descriptor, 112
 - network_change_notify_descriptor, 112
 - service_relocated_descriptor, 112
 - SH_delivery_system_descriptor, 112
 - supplementary_audio_descriptor, 112
 - T2_delivery_system_descriptor, 112
 - target_region_descriptor, 112
 - target_region_name_descriptor, 112
- partial_reception
 - isdb_desc_partial_reception, 281
- partial_reception_descriptor
 - Digital TV table parsing, 90
- partial_transport_stream_descriptor
 - Digital TV table parsing, 88
- partial_transport_stream_time_descriptor
 - Digital TV table parsing, 89
- pat
 - dvb_v5_descriptors, 273
- pat_pgm
 - dvb_v5_descriptors_program, 274
- path
 - dvb_dev_list, 208
- path_select
 - atsc_table_vct_channel, 162
- payload
 - dvb_mpeg_ts, 232
- payload_start
 - dvb_mpeg_ts, 232
- payload_unit_start_indicator
 - dvb_ts_packet_header, 269
- PCR
 - dvb_mpeg_ts_adaption, 235
 - dvb_ts_packet_header, 270
- pcr_pid
 - atsc_desc_service_location, 145
 - dvb_table_pmt, 260
- PDC_descriptor
 - Digital TV table parsing, 88
- PE
 - Ancillary functions and macros, 60
- PES_CRC
 - dvb_mpeg_pes_optional, 229
- PES_extension
 - dvb_mpeg_pes_optional, 229
- PES_priority
 - dvb_mpeg_pes_optional, 230
- PES_scrambling_control
 - dvb_mpeg_pes_optional, 230
- PF
 - Ancillary functions and macros, 60
- PG

- Ancillary functions and macros, 60
- PH
 - Ancillary functions and macros, 60
- pid
 - atsc_table_mgt_table, 156
 - dvb_elementary_pid, 210
 - dvb_mpeg_ts, 232
 - dvb_table_filter, 247
 - dvb_table_pat_program, 258
 - dvb_ts_packet_header, 270
- PK
 - Ancillary functions and macros, 60
- PL
 - Ancillary functions and macros, 60
- plp_id
 - dvb_desc_t2_delivery, 194
- PM
 - Ancillary functions and macros, 60
- pmt
 - dvb_v5_descriptors_program, 275
- pmt.h
 - dvb_pmt_field_first, 401
 - dvb_pmt_field_last, 401
- pmt_stream_name
 - Digital TV table parsing, 107
- PN
 - Ancillary functions and macros, 60
- polarization
 - dvb_desc_sat, 190
- POLARIZATION_H
 - Satellite Equipment Control, 50
- POLARIZATION_L
 - Satellite Equipment Control, 50
- POLARIZATION_OFF
 - Satellite Equipment Control, 50
- POLARIZATION_R
 - Satellite Equipment Control, 51
- POLARIZATION_V
 - Satellite Equipment Control, 50
- PR
 - Ancillary functions and macros, 60
- print
 - dvb_descriptor, 206
 - dvb_ext_descriptor, 217
- priority
 - dvb_desc_terrestrial_delivery, 201
 - dvb_mpeg_ts, 232
 - dvb_mpeg_ts_adaption, 235
 - dvb_ts_packet_header, 270
- priv
 - dvb_table_filter, 247
- private_data
 - dvb_mpeg_ts_adaption, 235
 - dvb_ts_packet_header, 270
- private_data_indicator_descriptor
 - Digital TV table parsing, 86
- private_data_specifier_descriptor
 - Digital TV table parsing, 88
- privdata
 - dvb_desc_ca, 167
- privdata_len
 - dvb_desc_ca, 167
- product
 - dvb_dev_list, 208
- program
 - dvb_table_pat, 256
 - dvb_v5_descriptors, 273
- program_number
 - atsc_table_vct_channel, 162
- programs
 - dvb_table_pat, 256
- prop
 - dvb_parse_table, 239
- props
 - dvb_entry, 215
- protocol_version
 - atsc_table_eit, 148
 - atsc_table_mgt, 153
 - atsc_table_vct, 157
- provider
 - dvb_desc_service, 191
- provider_emph
 - dvb_desc_service, 191
- PS
 - Ancillary functions and macros, 60
- PT
 - Ancillary functions and macros, 60
- pts
 - dvb_mpeg_pes_optional, 230
- PTS_DTS
 - dvb_mpeg_pes_optional, 230
- PW
 - Ancillary functions and macros, 60
- PY
 - Ancillary functions and macros, 60
- QA
 - Ancillary functions and macros, 60
- qm_intra
 - dvb_mpeg_es_seq_start, 225
- qm_nonintra
 - dvb_mpeg_es_seq_start, 225
- random_access
 - dvb_mpeg_ts_adaption, 235
 - dvb_ts_packet_header, 270
- rangeswitch
 - dvb_sat_lnb, 241
- RE
 - Ancillary functions and macros, 60
- reference_descriptor
 - Digital TV table parsing, 90
- registration_descriptor
 - Digital TV table parsing, 86
- related_content_descriptor
 - Digital TV table parsing, 88
- remote_control_key_id

- dvb_desc_ts_info, 203
- reserved
 - atsc_desc_service_location, 145
 - atsc_desc_service_location_elementary, 146
 - atsc_table_eit_desc_length, 149
 - atsc_table_vct_descriptor_length, 164
 - dvb_desc_ca, 167
 - dvb_desc_frequency_list, 178
 - dvb_desc_hierarchy, 180
 - dvb_desc_logical_channel_number, 184
 - dvb_desc_t2_delivery, 194
 - dvb_table_nit, 252
 - dvb_table_nit_transport, 254
 - dvb_table_nit_transport_header, 255
 - dvb_table_pat_program, 258
 - dvb_table_pmt_stream, 262
 - dvb_table_sdt, 264
 - dvb_table_sdt_service, 266
- reserved1
 - atsc_table_vct_channel, 162
- reserved2
 - atsc_table_vct_channel, 162
 - dvb_desc_hierarchy, 180
 - dvb_table_pmt, 260
 - dvb_table_pmt_stream, 262
- reserved3
 - atsc_table_vct_channel, 162
 - dvb_desc_hierarchy, 181
 - dvb_table_pmt, 260
- reserved4
 - dvb_desc_hierarchy, 181
- reserved_future_use
 - dvb_desc_cable_delivery, 171
- reserved_future_use1
 - dvb_desc_terrestrial_delivery, 201
- reserved_future_use2
 - dvb_desc_terrestrial_delivery, 202
- RO
 - Ancillary functions and macros, 60
- roll_off
 - dvb_desc_sat, 190
- RS
 - Ancillary functions and macros, 60
- RU
 - Ancillary functions and macros, 60
- running_status
 - dvb_table_eit_event, 246
 - dvb_table_sdt_service, 266
- RW
 - Ancillary functions and macros, 60
- S2_satellite_delivery_system_descriptor
 - Digital TV table parsing, 89
- SA
 - Ancillary functions and macros, 60
- sat_number
 - dvb_entry, 215
 - dvb_v5_fe_parms, 278
- Satellite Equipment Control, 49
 - dvb_print_all_inb, 51
 - dvb_print_inb, 51
 - dvb_sat_get_inb, 51
 - dvb_sat_get_inb_name, 52
 - dvb_sat_polarization, 50
 - dvb_sat_real_freq, 52
 - dvb_sat_search_inb, 52
 - dvb_sat_set_parms, 53
 - POLARIZATION_H, 50
 - POLARIZATION_L, 50
 - POLARIZATION_OFF, 50
 - POLARIZATION_R, 51
 - POLARIZATION_V, 50
- satellite_delivery_system_descriptor
 - Digital TV table parsing, 87
- SB
 - Ancillary functions and macros, 60
- SC
 - Ancillary functions and macros, 60
- scrambling
 - dvb_mpeg_ts, 233
- scrambling_descriptor
 - Digital TV table parsing, 88
- SD
 - Ancillary functions and macros, 60
- sdt
 - dvb_v5_descriptors, 274
- SE
 - Ancillary functions and macros, 60
- section_id
 - dvb_table_header, 250
- section_length
 - dvb_table_header, 250
- serial
 - dvb_dev_list, 208
- series_descriptor
 - Digital TV table parsing, 90
- service
 - dvb_table_sdt, 264
- service_availability_descriptor
 - Digital TV table parsing, 88
- service_descriptor
 - Digital TV table parsing, 87
- service_group_descriptor
 - Digital TV table parsing, 90
- service_id
 - dvb_desc_logical_channel_number, 185
 - dvb_desc_ts_info, 204
 - dvb_entry, 215
 - dvb_table_eit_event, 246
 - dvb_table_pat_program, 258
 - dvb_table_sdt_service, 267
 - isdb_partial_reception_service_id, 281
- service_identifier_descriptor
 - Digital TV table parsing, 88
- service_list_descriptor
 - Digital TV table parsing, 87
- service_location

- Digital TV table parsing, [89](#)
- service_move_descriptor
 - Digital TV table parsing, [88](#)
- service_relocated_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- service_type
 - atsc_table_vct_channel, [162](#)
 - dvb_desc_service, [192](#)
- SG
 - Ancillary functions and macros, [60](#)
- SH
 - Ancillary functions and macros, [60](#)
- SH_delivery_system_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- short_event_descriptor
 - Digital TV table parsing, [87](#)
- short_name
 - atsc_table_vct_channel, [162](#)
- short_node_information_descriptor
 - Digital TV table parsing, [90](#)
- short_smoothing_buffer_descriptor
 - Digital TV table parsing, [88](#)
- SI
 - Ancillary functions and macros, [60](#)
- SI_parameter_descriptor
 - Digital TV table parsing, [90](#)
- SI_prime_TS_descriptor
 - Digital TV table parsing, [90](#)
- SISO_MISO
 - dvb_desc_t2_delivery, [194](#)
- siso_miso
 - desc_t2_delivery.h, [336](#)
- size
 - atsc_table_mgt_table, [156](#)
 - dvb_descriptor, [206](#)
 - dvb_ext_descriptor, [217](#)
 - dvb_parse_struct, [238](#)
 - dvb_parse_table, [239](#)
- SJ
 - Ancillary functions and macros, [60](#)
- SK
 - Ancillary functions and macros, [60](#)
- SL
 - Ancillary functions and macros, [60](#)
- sl_descriptor
 - Digital TV table parsing, [86](#)
- SM
 - Ancillary functions and macros, [60](#)
- smoothing_buffer_descriptor
 - Digital TV table parsing, [86](#)
- SN
 - Ancillary functions and macros, [60](#)
- SO
 - Ancillary functions and macros, [60](#)
- source_id
 - atsc_table_eit_event, [151](#)
 - atsc_table_vct_channel, [162](#)
- splicing_point
 - dvb_mpeg_ts_adaption, [235](#)
 - dvb_ts_packet_header, [270](#)
- SR
 - Ancillary functions and macros, [60](#)
- SS
 - Ancillary functions and macros, [60](#)
- ST
 - Ancillary functions and macros, [60](#)
- start
 - atsc_table_eit_event, [152](#)
 - dvb_table_eit_event, [246](#)
- start_time
 - atsc_table_eit_event, [152](#)
- STC_reference_descriptor
 - Digital TV table parsing, [90](#)
- std_descriptor
 - Digital TV table parsing, [86](#)
- stream
 - dvb_table_pmt, [260](#)
- stream_13818_6_A
 - Digital TV table parsing, [91](#)
- stream_13818_6_B
 - Digital TV table parsing, [91](#)
- stream_13818_6_C
 - Digital TV table parsing, [91](#)
- stream_13818_6_D
 - Digital TV table parsing, [91](#)
- stream_14496_1_iso
 - Digital TV table parsing, [91](#)
- stream_14496_1_pes
 - Digital TV table parsing, [91](#)
- stream_audio
 - Digital TV table parsing, [91](#)
- stream_audio_13818_3
 - Digital TV table parsing, [91](#)
- stream_audio_14496_3
 - Digital TV table parsing, [91](#)
- stream_audio_a52
 - Digital TV table parsing, [91](#)
- stream_audio_a52_vls
 - Digital TV table parsing, [92](#)
- stream_audio_adts
 - Digital TV table parsing, [91](#)
- stream_audio_dts
 - Digital TV table parsing, [92](#)
- stream_audio_dts_hdmv
 - Digital TV table parsing, [91](#)
- stream_audio_e_ac3
 - Digital TV table parsing, [92](#)
- stream_audio_latm
 - Digital TV table parsing, [91](#)
- stream_audio_sdds
 - Digital TV table parsing, [91](#)
- stream_audio_sdds2
 - Digital TV table parsing, [92](#)
- stream_download
 - Digital TV table parsing, [91](#)
- stream_h222

- Digital TV table parsing, [91](#)
- stream_h222_1
 - Digital TV table parsing, [91](#)
- stream_h222_aux
 - Digital TV table parsing, [91](#)
- stream_id
 - dvb_mpeg_pes, [227](#)
- stream_identifier_descriptor
 - Digital TV table parsing, [87](#)
- stream_mheg
 - Digital TV table parsing, [91](#)
- stream_private_data
 - Digital TV table parsing, [91](#)
- stream_private_sections
 - Digital TV table parsing, [91](#)
- stream_scte_27
 - Digital TV table parsing, [91](#)
- stream_spu_vls
 - Digital TV table parsing, [92](#)
- stream_type
 - atsc_desc_service_location_elementary, [146](#)
- stream_video
 - Digital TV table parsing, [91](#)
- stream_video_14496_2
 - Digital TV table parsing, [91](#)
- stream_video_cavs
 - Digital TV table parsing, [91](#)
- stream_video_h262
 - Digital TV table parsing, [91](#)
- stream_video_h264
 - Digital TV table parsing, [91](#)
- stream_video_hevc
 - Digital TV table parsing, [91](#)
- stream_video_moto
 - Digital TV table parsing, [91](#)
- stuffing_descriptor
 - Digital TV table parsing, [87](#)
- subcel
 - dvb_desc_t2_delivery_cell, [196](#)
- subcel_info_loop_length
 - dvb_desc_t2_delivery, [194](#)
- subcel_length
 - dvb_desc_t2_delivery_cell, [196](#)
- subcell
 - dvb_desc_t2_delivery, [195](#)
- subtitling_descriptor
 - Digital TV table parsing, [87](#)
- supplementary_audio_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- SV
 - Ancillary functions and macros, [60](#)
- SX
 - Ancillary functions and macros, [60](#)
- SY
 - Ancillary functions and macros, [61](#)
- symbol_rate
 - dvb_desc_cable_delivery, [171](#)
 - dvb_desc_sat, [190](#)
- sync
 - dvb_mpeg_es_pic_start, [221](#)
 - dvb_mpeg_es_seq_start, [225](#)
 - dvb_mpeg_pes, [227](#)
- sync_byte
 - dvb_mpeg_ts, [233](#)
 - dvb_ts_packet_header, [270](#)
- syntax
 - dvb_table_header, [250](#)
- sysname
 - dvb_dev_list, [208](#)
- syspath
 - dvb_dev_list, [208](#)
- system_clock_descriptor
 - Digital TV table parsing, [86](#)
- system_id
 - dvb_desc_t2_delivery, [195](#)
- system_management_descriptor
 - Digital TV table parsing, [90](#)
- systems
 - dvb_v5_fe_parms, [278](#)
- SZ
 - Ancillary functions and macros, [61](#)
- T2_delivery_system_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- table
 - atsc_table_mgt, [153](#)
 - dvb_parse_struct, [238](#)
 - dvb_parse_table, [239](#)
 - dvb_table_filter, [248](#)
- table_id
 - dvb_table_header, [250](#)
- tables
 - atsc_table_mgt, [153](#)
- tag
 - ts_t, [286](#)
- target_area_descriptor
 - Digital TV table parsing, [89](#)
- target_background_grid_descriptor
 - Digital TV table parsing, [86](#)
- target_region_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- target_region_name_descriptor
 - Parsers for several MPEG-TS descriptors, [112](#)
- TC
 - Ancillary functions and macros, [61](#)
- TD
 - Ancillary functions and macros, [61](#)
- tei
 - dvb_mpeg_ts, [233](#)
- telephone_descriptor
 - Digital TV table parsing, [87](#)
- teletext_descriptor
 - Digital TV table parsing, [87](#)
- temporal_ref
 - dvb_mpeg_es_pic_start, [222](#)
- terrestrial_delivery_system_descriptor
 - Digital TV table parsing, [87](#)

- text
 - dvb_desc_event_extended, 173
 - dvb_desc_event_short, 176
- text_emph
 - dvb_desc_event_extended, 173
 - dvb_desc_event_short, 177
- TF
 - Ancillary functions and macros, 61
- tfs_flag
 - dvb_desc_t2_delivery, 195
- TG
 - Ancillary functions and macros, 61
- TH
 - Ancillary functions and macros, 61
- tid
 - dvb_table_filter, 248
- time_shifted_event_descriptor
 - Digital TV table parsing, 87
- time_shifted_service_descriptor
 - Digital TV table parsing, 87
- time_slice_fec_identifier_descriptor
 - Digital TV table parsing, 88
- time_slice_indicator
 - dvb_desc_terrestrial_delivery, 202
- title_length
 - atsc_table_eit_event, 152
- TJ
 - Ancillary functions and macros, 61
- TK
 - Ancillary functions and macros, 61
- TL
 - Ancillary functions and macros, 61
- TM
 - Ancillary functions and macros, 61
- TN
 - Ancillary functions and macros, 61
- TO
 - Ancillary functions and macros, 61
- TR
 - Ancillary functions and macros, 61
- transmission_mode
 - dvb_desc_t2_delivery, 195
 - dvb_desc_terrestrial_delivery, 202
 - isdbt_desc_terrestrial_delivery_system, 283
- transmission_type
 - dvb_desc_ts_info, 204
- transmission_type_count
 - dvb_desc_ts_info, 204
- transmission_type_info
 - dvb_desc_ts_info_transmission_type, 205
- transport
 - dvb_table_nit, 252
- transport_error_indicator
 - dvb_ts_packet_header, 271
- transport_id
 - dvb_entry, 215
 - dvb_table_eit, 243
 - dvb_table_nit_transport, 254
- transport_length
 - dvb_table_nit_transport_header, 255
- transport_priority
 - dvb_ts_packet_header, 271
- transport_scrambling_control
 - dvb_ts_packet_header, 271
- transport_stream_descriptor
 - Digital TV table parsing, 88
- transposer_frequency
 - dvb_desc_t2_delivery_subcell, 198
 - dvb_desc_t2_delivery_subcell_old, 199
- ts_id
 - dvb_table_filter, 248
- TS_Information_descriptor
 - descriptors.h, 346
- TS_Information_descriptor
 - Digital TV table parsing, 89
- ts_name
 - dvb_desc_ts_info, 204
- ts_name_emph
 - dvb_desc_ts_info, 204
- ts_t, 284
 - bitfield, 285
 - bitfield2, 285
 - bits00, 285
 - bits15, 285
 - bits30, 285
 - one, 285
 - one1, 285
 - one2, 286
 - tag, 286
- TT
 - Ancillary functions and macros, 61
- TV
 - Ancillary functions and macros, 61
- TVA_id_descriptor
 - Digital TV table parsing, 88
- TW
 - Ancillary functions and macros, 61
- two
 - dvb_mpeg_pes_optional, 230
- type
 - atsc_desc_service_location, 145
 - atsc_table_mgt_table, 156
 - dvb_desc, 165
 - dvb_desc_ca, 167
 - dvb_desc_ca_identifier, 168
 - dvb_desc_cable_delivery, 171
 - dvb_desc_event_extended, 173
 - dvb_desc_event_short, 177
 - dvb_desc_frequency_list, 179
 - dvb_desc_hierarchy, 181
 - dvb_desc_language, 182
 - dvb_desc_logical_channel, 183
 - dvb_desc_network_name, 186
 - dvb_desc_registration, 187
 - dvb_desc_sat, 190
 - dvb_desc_service, 192

- dvb_desc_terrestrial_delivery, [202](#)
- dvb_desc_ts_info, [204](#)
- dvb_elementary_pid, [211](#)
- dvb_extension_descriptor, [219](#)
- dvb_mpeg_es_pic_start, [222](#)
- dvb_mpeg_es_seq_start, [225](#)
- dvb_table_pmt_stream, [263](#)
- isdb_desc_partial_reception, [281](#)
- isdbt_desc_terrestrial_delivery_system, [284](#)
- type_version
 - atsc_table_mgt_table, [156](#)
- TZ
 - Ancillary functions and macros, [61](#)
- UA
 - Ancillary functions and macros, [61](#)
- UG
 - Ancillary functions and macros, [61](#)
- UM
 - Ancillary functions and macros, [61](#)
- US
 - Ancillary functions and macros, [61](#)
- UY
 - Ancillary functions and macros, [61](#)
- UZ
 - Ancillary functions and macros, [61](#)
- VA
 - Ancillary functions and macros, [61](#)
- VBI_data_descriptor
 - Digital TV table parsing, [87](#)
- VBI_teletext_descriptor
 - Digital TV table parsing, [87](#)
- vbv
 - dvb_mpeg_es_seq_start, [225](#)
- vbv_delay
 - dvb_mpeg_es_pic_start, [222](#)
- VC
 - Ancillary functions and macros, [61](#)
- vchannel
 - dvb_entry, [215](#)
- vct
 - dvb_v5_descriptors, [274](#)
- VE
 - Ancillary functions and macros, [61](#)
- verbose
 - dvb_v5_fe_parms, [279](#)
- version
 - dvb_table_header, [250](#)
 - dvb_v5_fe_parms, [279](#)
- VG
 - Ancillary functions and macros, [61](#)
- VI
 - Ancillary functions and macros, [61](#)
- video_decode_control_descriptor
 - Digital TV table parsing, [89](#)
- video_pid
 - dvb_entry, [216](#)
- video_pid_len
 - dvb_entry, [216](#)
- video_stream_descriptor
 - Digital TV table parsing, [86](#)
- video_window_descriptor
 - Digital TV table parsing, [86](#)
- visible_service_flag
 - dvb_desc_logical_channel_number, [185](#)
- VN
 - Ancillary functions and macros, [61](#)
- VU
 - Ancillary functions and macros, [61](#)
- west_east
 - dvb_desc_sat, [190](#)
- WF
 - Ancillary functions and macros, [61](#)
- width
 - dvb_mpeg_es_seq_start, [225](#)
- WS
 - Ancillary functions and macros, [61](#)
- XAIT_location_descriptor
 - Digital TV table parsing, [89](#)
- YE
 - Ancillary functions and macros, [61](#)
- YT
 - Ancillary functions and macros, [61](#)
- ZA
 - Ancillary functions and macros, [61](#)
- zero
 - dvb_table_header, [250](#)
 - dvb_table_pmt_stream, [263](#)
- zero3
 - dvb_table_pmt, [260](#)
- ZM
 - Ancillary functions and macros, [61](#)
- ZW
 - Ancillary functions and macros, [61](#)