```
<programlisting>
/*
 * frontend.h
 *
 * Copyright (C) 2000 Marcus Metzler &lt;marcus@convergence.de&gt;
 *                  Ralph  Metzler &lt;ralph@convergence.de&gt;
 *                  Holger Waechtler &lt;holger@convergence.de&gt;
 *                  Andre Draszik &lt;ad@convergence.de&gt;
 *                  for convergence integrated media GmbH
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public License
 * as published by the Free Software Foundation; either version 2.1
 * of the License, or (at your option) any later version.
 *
 * 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 Lesser General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 *
 */

#ifndef _DVBFRONTEND_H_
#define _DVBFRONTEND_H_

#include &lt;linux/types.h&gt;

typedef enum fe_type {
        FE_QPSK,
        FE_QAM,
        FE_OFDM,
        FE_ATSC
} fe_type_t;


typedef enum fe_caps {
        FE_IS_STUPID                    = 0,
        FE_CAN_INVERSION_AUTO           = 0x1,
        FE_CAN_FEC_1_2                  = 0x2,
        FE_CAN_FEC_2_3                  = 0x4,
        FE_CAN_FEC_3_4                  = 0x8,
        FE_CAN_FEC_4_5                  = 0x10,
        FE_CAN_FEC_5_6                  = 0x20,
        FE_CAN_FEC_6_7                  = 0x40,
        FE_CAN_FEC_7_8                  = 0x80,
        FE_CAN_FEC_8_9                  = 0x100,
        FE_CAN_FEC_AUTO                 = 0x200,
        FE_CAN_QPSK                     = 0x400,
        FE_CAN_QAM_16                   = 0x800,
        FE_CAN_QAM_32                   = 0x1000,
        FE_CAN_QAM_64                   = 0x2000,
        FE_CAN_QAM_128                  = 0x4000,
        FE_CAN_QAM_256                  = 0x8000,
        FE_CAN_QAM_AUTO                 = 0x10000,
        FE_CAN_TRANSMISSION_MODE_AUTO   = 0x20000,
        FE_CAN_BANDWIDTH_AUTO           = 0x40000,
        FE_CAN_GUARD_INTERVAL_AUTO      = 0x80000,
        FE_CAN_HIERARCHY_AUTO           = 0x100000,
        FE_CAN_8VSB                     = 0x200000,
        FE_CAN_16VSB                    = 0x400000,
        FE_HAS_EXTENDED_CAPS            = 0x800000,   /* We need more bitspace
for newer APIs, indicate this. */
        FE_CAN_2G_MODULATION            = 0x10000000, /* frontend supports "2nd
generation modulation" (DVB-S2) */
        FE_NEEDS_BENDING                = 0x20000000, /* not supported anymore,
don't use (frontend requires frequency bending) */
        FE_CAN_RECOVER                  = 0x40000000, /* frontend can recover
from a cable unplug automatically */
        FE_CAN_MUTE_TS                  = 0x80000000  /* frontend can stop
spurious TS data output */
} fe_caps_t;


struct dvb_frontend_info {
        char       name[128];
        fe_type_t  type;
        __u32      frequency_min;
        __u32      frequency_max;
        __u32      frequency_stepsize;
        __u32      frequency_tolerance;
        __u32      symbol_rate_min;
        __u32      symbol_rate_max;
        __u32      symbol_rate_tolerance;       /* ppm */
        __u32      notifier_delay;              /* DEPRECATED */
        fe_caps_t  caps;
};


/**
 *  Check out the DiSEqC bus spec available on http://www.eutelsat.org/ for
 *  the meaning of this struct...
 */
struct dvb_diseqc_master_cmd {
        __u8 msg [6];   /*  { framing, address, command, data [3] } */
        __u8 msg_len;   /*  valid values are 3...6  */
};


struct dvb_diseqc_slave_reply {
        __u8 msg [4];   /*  { framing, data [3] } */
        __u8 msg_len;   /*  valid values are 0...4, 0 means no msg  */
        int  timeout;   /*  return from ioctl after timeout ms with */
};                      /*  errorcode when no message was received  */


typedef enum fe_sec_voltage {
        SEC_VOLTAGE_13,
        SEC_VOLTAGE_18,
        SEC_VOLTAGE_OFF
} fe_sec_voltage_t;


typedef enum fe_sec_tone_mode {
        SEC_TONE_ON,
        SEC_TONE_OFF
} fe_sec_tone_mode_t;


typedef enum fe_sec_mini_cmd {
        SEC_MINI_A,
        SEC_MINI_B
} fe_sec_mini_cmd_t;


typedef enum fe_status {
        FE_HAS_SIGNAL   = 0x01,   /* found something above the noise level */
        FE_HAS_CARRIER  = 0x02,   /* found a DVB signal  */
        FE_HAS_VITERBI  = 0x04,   /* FEC is stable  */
        FE_HAS_SYNC     = 0x08,   /* found sync bytes  */
        FE_HAS_LOCK     = 0x10,   /* everything's working... */
        FE_TIMEDOUT     = 0x20,   /* no lock within the last ~2 seconds */
        FE_REINIT       = 0x40    /* frontend was reinitialized,  */
} fe_status_t;                    /* application is recommended to reset */
                                  /* DiSEqC, tone and parameters */

typedef enum fe_spectral_inversion {
        INVERSION_OFF,
        INVERSION_ON,
        INVERSION_AUTO
} fe_spectral_inversion_t;


typedef enum fe_code_rate {
        FEC_NONE = 0,
        FEC_1_2,
        FEC_2_3,
        FEC_3_4,
        FEC_4_5,
        FEC_5_6,
        FEC_6_7,
        FEC_7_8,
        FEC_8_9,
        FEC_AUTO,
        FEC_3_5,
        FEC_9_10,
} fe_code_rate_t;


typedef enum fe_modulation {
        QPSK,
        QAM_16,
        QAM_32,
        QAM_64,
        QAM_128,
        QAM_256,
        QAM_AUTO,
        VSB_8,
        VSB_16,
        PSK_8,
        APSK_16,
        APSK_32,
        DQPSK,
} fe_modulation_t;

typedef enum fe_transmit_mode {
        TRANSMISSION_MODE_2K,
        TRANSMISSION_MODE_8K,
        TRANSMISSION_MODE_AUTO,
        TRANSMISSION_MODE_4K
} fe_transmit_mode_t;

typedef enum fe_bandwidth {
        BANDWIDTH_8_MHZ,
        BANDWIDTH_7_MHZ,
        BANDWIDTH_6_MHZ,
        BANDWIDTH_AUTO
} fe_bandwidth_t;


typedef enum fe_guard_interval {
        GUARD_INTERVAL_1_32,
        GUARD_INTERVAL_1_16,
        GUARD_INTERVAL_1_8,
        GUARD_INTERVAL_1_4,
        GUARD_INTERVAL_AUTO
} fe_guard_interval_t;


typedef enum fe_hierarchy {
        HIERARCHY_NONE,
        HIERARCHY_1,
        HIERARCHY_2,
        HIERARCHY_4,
        HIERARCHY_AUTO
} fe_hierarchy_t;


struct dvb_qpsk_parameters {
        __u32           symbol_rate;  /* symbol rate in Symbols per second */
        fe_code_rate_t  fec_inner;    /* forward error correction (see above) */
};

struct dvb_qam_parameters {
        __u32           symbol_rate; /* symbol rate in Symbols per second */
        fe_code_rate_t  fec_inner;   /* forward error correction (see above) */
        fe_modulation_t modulation;  /* modulation type (see above) */
};

struct dvb_vsb_parameters {
        fe_modulation_t modulation;  /* modulation type (see above) */
```

```
};

struct dvb_ofdm_parameters {
        fe_bandwidth_t      bandwidth;
        fe_code_rate_t      code_rate_HP;  /* high priority stream code rate */
        fe_code_rate_t      code_rate_LP;  /* low priority stream code rate */
        fe_modulation_t     constellation; /* modulation type (see above) */
        fe_transmit_mode_t  transmission_mode;
        fe_guard_interval_t guard_interval;
        fe_hierarchy_t      hierarchy_information;
};


struct dvb_frontend_parameters {
        __u32 frequency;     /* (absolute) frequency in Hz for QAM/OFDM/ATSC */
                             /* intermediate frequency in kHz for QPSK */
        fe_spectral_inversion_t inversion;
        union {
                struct dvb_qpsk_parameters qpsk;
                struct dvb_qam_parameters  qam;
                struct dvb_ofdm_parameters ofdm;
                struct dvb_vsb_parameters vsb;
        } u;
};


struct dvb_frontend_event {
        fe_status_t status;
        struct dvb_frontend_parameters parameters;
};

/* S2API Commands */
#define DTV_UNDEFINED           0
#define DTV_TUNE                1
#define DTV_CLEAR               2
#define DTV_FREQUENCY           3
#define DTV_MODULATION          4
#define DTV_BANDWIDTH_HZ        5
#define DTV_INVERSION           6
#define DTV_DISEQC_MASTER       7
#define DTV_SYMBOL_RATE         8
#define DTV_INNER_FEC           9
#define DTV_VOLTAGE             10
#define DTV_TONE                11
#define DTV_PILOT               12
#define DTV_ROLLOFF             13
#define DTV_DISEQC_SLAVE_REPLY  14

/* Basic enumeration set for querying unlimited capabilities */
#define DTV_FE_CAPABILITY_COUNT 15
#define DTV_FE_CAPABILITY       16
#define DTV_DELIVERY_SYSTEM     17

/* ISDB-T and ISDB-Tsb */
#define DTV_ISDBT_PARTIAL_RECEPTION     18
#define DTV_ISDBT_SOUND_BROADCASTING    19
```

```
#define DTV_ISDBT_SB_SUBCHANNEL_ID      20
#define DTV_ISDBT_SB_SEGMENT_IDX        21
#define DTV_ISDBT_SB_SEGMENT_COUNT      22

#define DTV_ISDBT_LAYERA_FEC                    23
#define DTV_ISDBT_LAYERA_MODULATION             24
#define DTV_ISDBT_LAYERA_SEGMENT_COUNT          25
#define DTV_ISDBT_LAYERA_TIME_INTERLEAVING      26

#define DTV_ISDBT_LAYERB_FEC                    27
#define DTV_ISDBT_LAYERB_MODULATION             28
#define DTV_ISDBT_LAYERB_SEGMENT_COUNT          29
#define DTV_ISDBT_LAYERB_TIME_INTERLEAVING      30

#define DTV_ISDBT_LAYERC_FEC                    31
#define DTV_ISDBT_LAYERC_MODULATION             32
#define DTV_ISDBT_LAYERC_SEGMENT_COUNT          33
#define DTV_ISDBT_LAYERC_TIME_INTERLEAVING      34

#define DTV_API_VERSION         35

#define DTV_CODE_RATE_HP        36
#define DTV_CODE_RATE_LP        37
#define DTV_GUARD_INTERVAL      38
#define DTV_TRANSMISSION_MODE   39
#define DTV_HIERARCHY           40

#define DTV_ISDBT_LAYER_ENABLED 41

#define DTV_ISDBS_TS_ID         42

#define DTV_MAX_COMMAND                         DTV_ISDBS_TS_ID

typedef enum fe_pilot {
        PILOT_ON,
        PILOT_OFF,
        PILOT_AUTO,
} fe_pilot_t;

typedef enum fe_rolloff {
        ROLLOFF_35, /* Implied value in DVB-S, default for DVB-S2 */
        ROLLOFF_20,
        ROLLOFF_25,
        ROLLOFF_AUTO,
} fe_rolloff_t;

typedef enum fe_delivery_system {
        SYS_UNDEFINED,
        SYS_DVBC_ANNEX_AC,
        SYS_DVBC_ANNEX_B,
        SYS_DVBT,
        SYS_DSS,
        SYS_DVBS,
        SYS_DVBS2,
        SYS_DVBH,
        SYS_ISDBT,
        SYS_ISDBS,
        SYS_ISDBC,
        SYS_ATSC,
        SYS_ATSCMH,
        SYS_DMBTH,
        SYS_CMMB,
        SYS_DAB,
} fe_delivery_system_t;

struct dtv_cmds_h {
        char    *name;          /* A display name for debugging purposes */

        __u32   cmd;            /* A unique ID */

        /* Flags */
        __u32   set:1;          /* Either a set or get property */
        __u32   buffer:1;       /* Does this property use the buffer? */
        __u32   reserved:30;    /* Align */
};

struct dtv_property {
        __u32 cmd;
        __u32 reserved[3];
        union {
                __u32 data;
                struct {
                        __u8 data[32];
                        __u32 len;
                        __u32 reserved1[3];
                        void *reserved2;
                } buffer;
        } u;
        int result;
} __attribute__ ((packed));

/* num of properties cannot exceed DTV_IOCTL_MAX_MSGS per ioctl */
#define DTV_IOCTL_MAX_MSGS 64

struct dtv_properties {
        __u32 num;
        struct dtv_property *props;
};

#define <link linkend="FE_GET_PROPERTY">FE_SET_PROPERTY</link>
_IOW('o', 82, struct dtv_properties)
#define <link linkend="FE_GET_PROPERTY">FE_GET_PROPERTY</link>
_IOR('o', 83, struct dtv_properties)


/**
 * When set, this flag will disable any zigzagging or other "normal" tuning
 * behaviour. Additionally, there will be no automatic monitoring of the lock
 * status, and hence no frontend events will be generated. If a frontend device
 * is closed, this flag will be automatically turned off when the device is
 * reopened read-write.
 */
#define FE_TUNE_MODE_ONESHOT 0x01


#define <link linkend="FE_GET_INFO">FE_GET_INFO</link>                _IOR('o',
61, struct dvb_frontend_info)

#define <link linkend="FE_DISEQC_RESET_OVERLOAD">FE_DISEQC_RESET_OVERLOAD</link>
  _IO('o', 62)
#define <link
linkend="FE_DISEQC_SEND_MASTER_CMD">FE_DISEQC_SEND_MASTER_CMD</link>  _IOW('o',
63, struct dvb_diseqc_master_cmd)
#define <link
linkend="FE_DISEQC_RECV_SLAVE_REPLY">FE_DISEQC_RECV_SLAVE_REPLY</link> _IOR('o',
64, struct dvb_diseqc_slave_reply)
#define <link linkend="FE_DISEQC_SEND_BURST">FE_DISEQC_SEND_BURST</link>
_IO('o', 65)  /* fe_sec_mini_cmd_t */

#define <link linkend="FE_SET_TONE">FE_SET_TONE</link>                _IO('o',
66)  /* fe_sec_tone_mode_t */
#define <link linkend="FE_SET_VOLTAGE">FE_SET_VOLTAGE</link>
_IO('o', 67)  /* fe_sec_voltage_t */
#define <link
linkend="FE_ENABLE_HIGH_LNB_VOLTAGE">FE_ENABLE_HIGH_LNB_VOLTAGE</link> _IO('o',
68)  /* int */

#define <link linkend="FE_READ_STATUS">FE_READ_STATUS</link>
_IOR('o', 69, fe_status_t)
#define <link linkend="FE_READ_BER">FE_READ_BER</link>                _IOR('o',
70, __u32)
#define <link linkend="FE_READ_SIGNAL_STRENGTH">FE_READ_SIGNAL_STRENGTH</link>
 _IOR('o', 71, __u16)
#define <link linkend="FE_READ_SNR">FE_READ_SNR</link>                _IOR('o',
72, __u16)
#define <link
linkend="FE_READ_UNCORRECTED_BLOCKS">FE_READ_UNCORRECTED_BLOCKS</link> _IOR('o',
73, __u32)

#define <link linkend="FE_SET_FRONTEND">FE_SET_FRONTEND</link>
_IOW('o', 76, struct dvb_frontend_parameters)
#define <link linkend="FE_GET_FRONTEND">FE_GET_FRONTEND</link>
_IOR('o', 77, struct dvb_frontend_parameters)
#define <link
linkend="FE_SET_FRONTEND_TUNE_MODE">FE_SET_FRONTEND_TUNE_MODE</link>  _IO('o',
81) /* unsigned int */
#define <link linkend="FE_GET_EVENT">FE_GET_EVENT</link>                _IOR('o',
78, struct dvb_frontend_event)

#define <link
linkend="FE_DISHNETWORK_SEND_LEGACY_CMD">FE_DISHNETWORK_SEND_LEGACY_CMD</link>
_IO('o', 80) /* unsigned int */

#endif /*_DVBFRONTEND_H_*/
</programlisting>
```