vidtv: Virtual Digital TV driver

Author: Daniel W. S. Almeida <dwlsalmeida@gmail.com>, June 2020.

Background

Vidtv is a virtual DVB driver that aims to serve as a reference for driver writers by serving as a template. It also validates the existing media DVB APIs, thus helping userspace application writers.

Currently, it consists of:

- A fake tuner driver, which will report a bad signal quality if the chosen frequency is too far away from a table of valid frequencies for a particular delivery system.
- A fake demod driver, which will constantly poll the fake signal quality returned by the tuner, simulating a device that can lose/reacquire a lock on the signal depending on the CNR levels.
- A fake bridge driver, which is the module responsible for modprobing the fake tuner and demod modules and implementing the demux logic. This module takes parameters at initialization that will dictate how the simulation behaves.
- Code reponsible for encoding a valid MPEG Transport Stream, which is then passed to the bridge driver. This fake stream contains some hardcoded content. For now, we have a single, audio-only channel containing a single MPEG Elementary Stream, which in turn contains a SMPTE 302m encoded sine-wave. Note that this particular encoder was chosen because it is the easiest way to encode PCM audio data in a MPEG Transport Stream.

Building vidtv

vidtv is a test driver and thus is **not** enabled by default when compiling the kernel.

In order to enable compilation of vidtv:

- Enable **DVB_TEST_DRIVERS**, then
- Enable **DVB_VIDTV**

When compiled as a module, expect the following .ko files:

- dvb_vidtv_tuner.ko
- dvb_vidtv_demod.ko
- dvb_vidtv_bridge.ko

Running vidtv

When compiled as a module, run:

```
modprobe vidtv
```

That's it! The bridge driver will initialize the tuner and demod drivers as part of its own initialization.

By default, it will accept the following frequencies:

- 474 MHz for DVB-T/T2/C;
- 11,362 GHz for DVB-S/S2.

For satellite systems, the driver simulates an universal extended LNBf, with frequencies at Ku-Band, ranging from 10.7 GHz to 12.75 GHz.

You can optionally define some command-line arguments to vidtv.

Command-line arguments to vidtv

Below is a list of all arguments that can be supplied to vidtv:

drop_tslock_prob_on_low_snr
: Probability of losing the TS lock if the signal quality is bad. This probability be used by the fake demodulator driver to eventually return a status of 0 when the signal quality is not good.

recover_tslock_prob_on_good_snr:
: Probability recovering the TS lock when the signal improves. This probability be used by the fake demodulator driver to eventually return a status of 0x1f when/if the signal quality improves.

mock_power_up_delay_msec
: Simulate a power up delay. Default: 0.

mock_tune_delay_msec
: Simulate a tune delay. Default 0.

vidtv_valid_dvb_t_freqs
: Valid DVB-T frequencies to simulate, in Hz.

vidtv_valid_dvb_c_freqs
: Valid DVB-C frequencies to simulate, in Hz.

vidtv_valid_dvb_s_freqs
: Valid DVB-S/S2 frequencies to simulate at Ku-Band, in kHz.

max_frequency_shift_hz,
: Maximum shift in HZ allowed when tuning in a channel.

si_period_msec
: How often to send SI packets. Default: 40ms.

pcr_period_msec
: How often to send PCR packets. Default: 40ms.

mux_rate_kbytes_sec
: Attempt to maintain this bit rate by inserting TS null packets, if necessary. Default: 4096.

pcr_pid,
: PCR PID for all channels. Default: 0x200.

mux_buf_sz_pkts,
: Size for the mux buffer in multiples of 188 bytes.

vidtv internal structure

The kernel modules are split in the following way:

vidtv_tuner.[ch]
: Implements a fake tuner DVB driver.

vidtv_demod.[ch]
: Implements a fake demodulator DVB driver.

vidtv_bridge.[ch]
: Implements a bridge driver.

The MPEG related code is split in the following way:

vidtv_ts.[ch]

: Code to work with MPEG TS packets, such as TS headers, adaptation fields, PCR packets and NULL packets.

vidtv_psi.[ch]

: This is the PSI generator. PSI packets contain general information about a MPEG Transport Stream. A PSI generator is needed so userspace apps can retrieve information about the Transport Stream and eventually tune into a (dummy) channel.

  Because the generator is implemented in a separate file, it can be reused elsewhere in the media subsystem.

  Currently vidtv supports working with 5 PSI tables: PAT, PMT, SDT, NIT and EIT.

  The specification for PAT and PMT can be found in *ISO 13818-1: Systems*, while the specification for the SDT, NIT, EIT can be found in *ETSI EN 300 468: Specification for Service Information (SI) in DVB systems*.

  It isn't strictly necessary, but using a real TS file helps when debugging PSI tables. Vidtv currently tries to replicate the PSI structure found in this file: TS1Globo.ts.

  A good way to visualize the structure of streams is by using DVBInspector.

vidtv_pes.[ch]

: Implements the PES logic to convert encoder data into MPEG TS packets. These can then be fed into a TS multiplexer and eventually into userspace.

vidtv_encoder.h

: An interface for vidtv encoders. New encoders can be added to this driver by implementing the calls in this file.

vidtv_s302m.[ch]

: Implements a S302M encoder to make it possible to insert PCM audio data in the generated MPEG Transport Stream. The relevant specification is available online as *SMPTE 302M-2007: Television - Mapping of AES3 Data into MPEG-2 Transport Stream*.

  The resulting MPEG Elementary Stream is conveyed in a private stream with a S302M registration descriptor attached.

  This shall enable passing an audio signal into userspace so it can be decoded and played by media software. The corresponding decoder in ffmpeg is located in 'libavcodec/s302m.c' and is experimental.

vidtv_channel.[ch]

: Implements a 'channel' abstraction.

  When vidtv boots, it will create some hardcoded channels:

1. Their services will be concatenated to populate the SDT.
2. Their programs will be concatenated to populate the PAT
3. Their events will be concatenated to populate the EIT
4. For each program in the PAT, a PMT section will be created
5. The PMT section for a channel will be assigned its streams.
6. Every stream will have its corresponding encoder polled in a loop to produce TS packets. These packets may be interleaved by the muxer and then delivered to the bridge.

vidtv_mux.[ch]

Implements a MPEG TS mux, loosely based on the ffmpeg implementation in "libavcodec/mpegtsenc.c"

The muxer runs a loop which is responsible for:

1. Keeping track of the amount of time elapsed since the last iteration.
2. Polling encoders in order to fetch 'elapsed_time' worth of data.
3. Inserting PSI and/or PCR packets, if needed.
4. Padding the resulting stream with NULL packets if necessary in order to maintain the chosen bit rate.
5. Delivering the resulting TS packets to the bridge driver so it can pass them to the demux.

Testing vidtv with v4l-utils

Using the tools in v4l-utils is a great way to test and inspect the output of vidtv. It is hosted here: v4l-utils Documentation.

From its webpage:

```
The v4l-utils are a series of packages for handling media devices.

It is hosted at http://git.linuxtv.org/v4l-utils.git, and packaged
on most distributions.

It provides a series of libraries and utilities to be used to
control several aspect of the media boards.
```

Start by installing v4l-utils and then modprobing vidtv:

```
modprobe dvb_vidtv_bridge
```

If the driver is OK, it should load and its probing code will run. This will pull in the tuner and demod drivers.

Using dvb-fe-tool

The first step to check whether the demod loaded successfully is to run:

```
$ dvb-fe-tool
Device Dummy demod for DVB-T/T2/C/S/S2 (/dev/dvb/adapter0/frontend0) capabilities:
    CAN_FEC_1_2
    CAN_FEC_2_3
    CAN_FEC_3_4
    CAN_FEC_4_5
    CAN_FEC_5_6
    CAN_FEC_6_7
    CAN_FEC_7_8
    CAN_FEC_8_9
    CAN_FEC_AUTO
    CAN_GUARD_INTERVAL_AUTO
    CAN_HIERARCHY_AUTO
    CAN_INVERSION_AUTO
    CAN_QAM_16
    CAN_QAM_32
    CAN_QAM_64
    CAN_QAM_128
    CAN_QAM_256
    CAN_QAM_AUTO
    CAN_QPSK
    CAN_TRANSMISSION_MODE_AUTO
DVB API Version 5.11, Current v5 delivery system: DVBC/ANNEX_A
Supported delivery systems:
    DVBT
    DVBT2
    [DVBC/ANNEX_A]
    DVBS
    DVBS2
Frequency range for the current standard:
From:            51.0 MHz
To:              2.15 GHz
Step:            62.5 kHz
Tolerance:       29.5 MHz
Symbol rate ranges for the current standard:
```

```
From:            1.00 MBauds
To:              45.0 MBauds
```

This should return what is currently set up at the demod struct, i.e.:

```
static const struct dvb_frontend_ops vidtv_demod_ops = {
        .delsys = {
                SYS_DVBT,
                SYS_DVBT2,
                SYS_DVBC_ANNEX_A,
                SYS_DVBS,
                SYS_DVBS2,
        },

        .info = {
                .name                   = "Dummy demod for DVB-T/T2/C/S/S2",
                .frequency_min_hz       = 51 * MHz,
                .frequency_max_hz       = 2150 * MHz,
                .frequency_stepsize_hz  = 62500,
                .frequency_tolerance_hz = 29500 * kHz,
                .symbol_rate_min        = 1000000,
                .symbol_rate_max        = 45000000,

                .caps = FE_CAN_FEC_1_2 |
                        FE_CAN_FEC_2_3 |
                        FE_CAN_FEC_3_4 |
                        FE_CAN_FEC_4_5 |
                        FE_CAN_FEC_5_6 |
                        FE_CAN_FEC_6_7 |
                        FE_CAN_FEC_7_8 |
                        FE_CAN_FEC_8_9 |
                        FE_CAN_QAM_16 |
                        FE_CAN_QAM_64 |
                        FE_CAN_QAM_32 |
                        FE_CAN_QAM_128 |
                        FE_CAN_QAM_256 |
                        FE_CAN_QAM_AUTO |
                        FE_CAN_QPSK |
                        FE_CAN_FEC_AUTO |
                        FE_CAN_INVERSION_AUTO |
                        FE_CAN_TRANSMISSION_MODE_AUTO |
                        FE_CAN_GUARD_INTERVAL_AUTO |
                        FE_CAN_HIERARCHY_AUTO,
        }

        ....
```

For more information on dvb-fe-tools check its online documentation here: dvb-fe-tool Documentation.

Using dvb-scan

In order to tune into a channel and read the PSI tables, we can use dvb-scan.

For this, one should provide a configuration file known as a 'scan file', here's an example:

```
[Channel]
FREQUENCY = 474000000
MODULATION = QAM/AUTO
SYMBOL_RATE = 6940000
INNER_FEC = AUTO
DELIVERY_SYSTEM = DVBC/ANNEX_A
```

> **Note**
>
> The parameters depend on the video standard you're testing.

> **Note**
>
> Vidtv is a fake driver and does not validate much of the information in the scan file. Just specifying 'FREQUENCY' and 'DELIVERY_SYSTEM' should be enough for DVB-T/DVB-T2. For DVB-S/DVB-C however, you should also provide 'SYMBOL_RATE'.

You can browse scan tables online here: dvb-scan-tables.

Assuming this channel is named 'channel.conf', you can then run:

```
$ dvbv5-scan channel.conf
dvbv5-scan ~/vidtv.conf
ERROR    command BANDWIDTH_HZ (5) not found during retrieve
```

```
Cannot calc frequency shift. Either bandwidth/symbol-rate is unavailable (yet).
Scanning frequency #1 330000000
    (0x00) Signal= -68.00dBm
Scanning frequency #2 474000000
Lock   (0x1f) Signal= -34.45dBm C/N= 33.74dB UCB= 0
Service Beethoven, provider LinuxTV.org: digital television
```

For more information on dvb-scan, check its documentation online here: dvb-scan Documentation.

Using dvb-zap

dvbv5-zap is a command line tool that can be used to record MPEG-TS to disk. The typical use is to tune into a channel and put it into record mode. The example below - which is taken from the documentation - illustrates that[1]:

```
$ dvbv5-zap -c dvb_channel.conf "beethoven" -o music.ts -P -t 10
using demux 'dvb0.demux0'
reading channels from file 'dvb_channel.conf'
tuning to 474000000 Hz
pass all PID's to TS
dvb_set_pesfilter 8192
dvb_dev_set_bufsize: buffer set to 6160384
Lock   (0x1f) Quality= Good Signal= -34.66dBm C/N= 33.41dB UCB= 0 postBER= 0 preBER= 1.05x10^-3 PER= 0
Lock   (0x1f) Quality= Good Signal= -34.57dBm C/N= 33.46dB UCB= 0 postBER= 0 preBER= 1.05x10^-3 PER= 0
Record to file 'music.ts' started
received 24587768 bytes (2401 Kbytes/sec)
Lock   (0x1f) Quality= Good Signal= -34.42dBm C/N= 33.89dB UCB= 0 postBER= 0 preBER= 2.44x10^-3 PER= 0
```

[1] In this example, it records 10 seconds with all program ID's stored at the music.ts file.

The channel can be watched by playing the contents of the stream with some player that recognizes the MPEG-TS format, such as `mplayer` or `vlc`.

By playing the contents of the stream one can visually inspect the workings of vidtv, e.g., to play a recorded TS file with:

```
$ mplayer music.ts
```

or, alternatively, running this command on one terminal:

```
$ dvbv5-zap -c dvb_channel.conf "beethoven" -P -r &
```

And, on a second terminal, playing the contents from DVR interface with:

```
$ mplayer /dev/dvb/adapter0/dvr0
```

For more information on dvb-zap check its online documentation here: dvb-zap Documentation. See also: zap.

What can still be improved in vidtv

Add *debugfs* integration

Although frontend drivers provide DVBv5 statistics via the .read_status call, a nice addition would be to make additional statistics available to userspace via debugfs, which is a simple-to-use, RAM-based filesystem specifically designed for debug purposes.

The logic for this would be implemented on a separate file so as not to pollute the frontend driver. These statistics are driver-specific and can be useful during tests.

The Siano driver is one example of a driver using debugfs to convey driver-specific statistics to userspace and it can be used as a reference.

This should be further enabled and disabled via a Kconfig option for convenience.

Add a way to test video

Currently, vidtv can only encode PCM audio. It would be great to implement a barebones version of MPEG-2 video encoding so we can also test video. The first place to look into is *ISO 13818-2: Information technology â€" Generic coding of moving pictures and associated audio information â€" Part 2: Video*, which covers the encoding of compressed video in MPEG Transport Streams.

This might optionally use the Video4Linux2 Test Pattern Generator, v4l2-tpg, which resides at:

```
drivers/media/common/v4l2-tpg/
```

Add white noise simulation

The vidtv tuner already has code to identify whether the chosen frequency is too far away from a table of valid frequencies. For now, this means that the demodulator can eventually lose the lock on the signal, since the tuner will report a bad signal quality.

A nice addition is to simulate some noise when the signal quality is bad by:

- Randomly dropping some TS packets. This will trigger a continuity error if the continuity counter is updated but the packet is not passed on to the demux.
- Updating the error statistics accordingly (e.g. BER, etc).
- Simulating some noise in the encoded data.

Functions and structs used within vidtv

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 491)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_bridge.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 493)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_channel.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 495)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_demod.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 497)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_encoder.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 499)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_mux.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 501)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_pes.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 503)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_psi.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 505)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_s302m.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 507)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_ts.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 509)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_tuner.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 511)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_common.c
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master][Documentation][driver-api][media][drivers]vidtv.rst, line 513)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/media/test-drivers/vidtv/vidtv_tuner.c
```