Gadget Testing

This file summarizes information on basic testing of USB functions provided by gadgets.

1. ACM function

The function is provided by usb_f_acm.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "acm". The ACM function provides just one attribute in its function directory:

port_num

The attribute is read-only.

There can be at most 4 ACM/generic serial/OBEX ports in the system.

Testing the ACM function

On the host:

```
cat > /dev/ttyACM<X>
```

On the device:

```
cat /dev/ttyGS<Y>
```

then the other way round

On the device:

```
cat > /dev/ttyGS<Y>
```

On the host:

```
cat /dev/ttyACM<X>
```

2. ECM function

The function is provided by usb_f_ecm.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "ecm". The ECM function provides these attributes in its function directory:

ifname	network device interface name associated with this function instance
qmult	queue length multiplier for high and super speed
host_addr	MAC address of host's end of this Ethernet over USB link
dev_addr	MAC address of device's end of this Ethernet over USB link

and after creating the functions/ecm.<instance name> they contain default values: qmult is 5, dev_addr and host_addr are randomly selected. The ifname can be written to if the function is not bound. A write must be an interface pattern such as "usb%d", which will cause the net core to choose the next free usbX interface. By default, it is set to "usb%d".

Testing the ECM function

Configure IP addresses of the device and the host. Then:

On the device:

```
ping <host's IP>
```

On the host:

```
ping <device's IP>
```

3. ECM subset function

The function is provided by usb_f_ecm_subset.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "geth". The ECM subset function provides these attributes in its function directory:

ifname	network device interface name associated with this function instance
qmult	queue length multiplier for high and super speed
host_addr	MAC address of host's end of this Ethernet over USB link
dev_addr	MAC address of device's end of this Ethernet over USB link

and after creating the functions/ecm.<instance name> they contain default values: qmult is 5, dev_addr and host_addr are randomly selected. The ifname can be written to if the function is not bound. A write must be an interface pattern such as "usb%d", which will cause the net core to choose the next free usbX interface. By default, it is set to "usb%d".

Testing the ECM subset function

Configure IP addresses of the device and the host. Then:

On the device:

```
ping <host's IP>
```

On the host:

```
ping <device's IP>
```

4. EEM function

The function is provided by usb_f_eem.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "eem". The EEM function provides these attributes in its function directory:

ifname	network device interface name associated with this function instance
qmult	queue length multiplier for high and super speed
host_addr	MAC address of host's end of this Ethernet over USB link
dev_addr	MAC address of device's end of this Ethernet over USB link

and after creating the functions/eem.<instance name> they contain default values: qmult is 5, dev_addr and host_addr are randomly selected. The ifname can be written to if the function is not bound. A write must be an interface pattern such as "usb%d", which will cause the net core to choose the next free usbX interface. By default, it is set to "usb%d".

Testing the EEM function

Configure IP addresses of the device and the host. Then:

On the device:

```
ping <host's IP>
```

On the host:

```
ping <device's IP>
```

5. FFS function

The function is provided by usb_f_fs.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "ffs". The function directory is intentionally empty and not modifiable.

After creating the directory there is a new instance (a "device") of FunctionFS available in the system. Once a "device" is available, the user should follow the standard procedure for using FunctionFS (mount it, run the userspace process which implements the function proper). The gadget should be enabled by writing a suitable string to usb_gadget/<gadget>/UDC.

Testing the FFS function

Testing the FFS function

On the device: start the function's userspace daemon, enable the gadget

On the host: use the USB function provided by the device

6. HID function

The function is provided by usb_f_hid.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "hid". The HID function provides these attributes in its function directory:

protocol	HID protocol to use
report_desc	data to be used in HID reports, except data passed with /dev/hidg<X>
report_length	HID report length
subclass	HID subclass to use

For a keyboard the protocol and the subclass are 1, the report_length is 8, while the report_desc is:

```
$ hd my_report_desc
00000000  05 01 09 06 a1 01 05 07  19 e0 29 e7 15 00 25 01  |..........)...%.|
00000010  75 01 95 08 81 02 95 01  75 08 81 03 95 05 75 01  |u.......u.....u.|
00000020  05 08 19 01 29 05 91 02  95 01 75 03 91 03 95 06  |....).....u.....|
00000030  75 08 15 00 25 65 05 07  19 00 29 65 81 00 c0     |u...%e....)e...|
0000003f
```

Such a sequence of bytes can be stored to the attribute with echo:

```
$ echo -ne \\x05\\x01\\x09\\x06\\xa1.....
```

Testing the HID function

Device:

- create the gadget
- connect the gadget to a host, preferably not the one used to control the gadget
- run a program which writes to /dev/hidg<N>, e.g. a userspace program found in Documentation/usb/gadget_hid.rst:

```
$ ./hid_gadget_test /dev/hidg0 keyboard
```

Host:

- observe the keystrokes from the gadget

7. LOOPBACK function

The function is provided by usb_f_ss_lb.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "Loopback". The LOOPBACK function provides these attributes in its function directory:

qlen	depth of loopback queue
bulk_buflen	buffer length

Testing the LOOPBACK function

device: run the gadget

host: test-usb (tools/usb/testusb.c)

8. MASS STORAGE function

The function is provided by usb_f_mass_storage.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "mass_storage". The MASS STORAGE function provides these attributes in its directory: files:

stall	Set to permit function to halt bulk endpoints. Disabled on some USB devices known not to work correctly. You should set it to true.
num_buffers	Number of pipeline buffers. Valid numbers are 2..4. Available only if CONFIG_USB_GADGET_DEBUG_FILES is set.

and a default lun.0 directory corresponding to SCSI LUN #0.

A new lun can be added with mkdir:

```
$ mkdir functions/mass_storage.0/partition.5
```

Lun numbering does not have to be continuous, except for lun #0 which is created by default. A maximum of 8 luns can be specified and they all must be named following the <name>.<number> scheme. The numbers can be 0..8. Probably a good convention is to name the luns "lun.<number>", although it is not mandatory.

In each lun directory there are the following attribute files:

file	The path to the backing file for the LUN. Required if LUN is not marked as removable.
ro	Flag specifying access to the LUN shall be read-only. This is implied if CD-ROM emulation is enabled as well as when it was impossible to open "filename" in R/W mode.
removable	Flag specifying that LUN shall be indicated as being removable.
cdrom	Flag specifying that LUN shall be reported as being a CD-ROM.
nofua	Flag specifying that FUA flag in SCSI WRITE(10,12)

Testing the MASS STORAGE function

device: connect the gadget, enable it host: dmesg, see the USB drives appear (if system configured to automatically mount)

9. MIDI function

The function is provided by usb_f_midi.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "midi". The MIDI function provides these attributes in its function directory:

buflen	MIDI buffer length
id	ID string for the USB MIDI adapter
in_ports	number of MIDI input ports
index	index value for the USB MIDI adapter
out_ports	number of MIDI output ports
qlen	USB read request queue length

Testing the MIDI function

There are two cases: playing a mid from the gadget to the host and playing a mid from the host to the gadget.

1. Playing a mid from the gadget to the host:

host:

```
$ arecordmidi -l
 Port    Client name                      Port name
 14:0    Midi Through                     Midi Through Port-0
 24:0    MIDI Gadget                      MIDI Gadget MIDI 1
$ arecordmidi -p 24:0 from_gadget.mid
```

gadget:

```
$ aplaymidi -l
 Port    Client name                      Port name
 20:0    f_midi                           f_midi

$ aplaymidi -p 20:0 to_host.mid
```

2. Playing a mid from the host to the gadget

gadget:

```
$ arecordmidi -l
 Port    Client name                      Port name
 20:0    f_midi                           f_midi

$ arecordmidi -p 20:0 from_host.mid
```

host:

```
$ aplaymidi -l
 Port    Client name                      Port name
 14:0    Midi Through                     Midi Through Port-0
 24:0    MIDI Gadget                      MIDI Gadget MIDI 1

$ aplaymidi -p24:0 to_gadget.mid
```

The from_gadget.mid should sound identical to the to_host.mid.

The from_host.id should sound identical to the to_gadget.mid.

MIDI files can be played to speakers/headphones with e.g. timidity installed:

```
$ aplaymidi -l
 Port    Client name                      Port name
 14:0    Midi Through                     Midi Through Port-0
 24:0    MIDI Gadget                      MIDI Gadget MIDI 1
128:0    TiMidity                         TiMidity port 0
128:1    TiMidity                         TiMidity port 1
128:2    TiMidity                         TiMidity port 2
128:3    TiMidity                         TiMidity port 3

$ aplaymidi -p 128:0 file.mid
```

MIDI ports can be logically connected using the aconnect utility, e.g.:

```
$ aconnect 24:0 128:0 # try it on the host
```

After the gadget's MIDI port is connected to timidity's MIDI port, whatever is played at the gadget side with aplaymidi -l is audible in host's speakers/headphones.

10. NCM function

The function is provided by usb_f_ncm.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "ncm". The NCM function provides these attributes in its function directory:

ifname	network device interface name associated with this function instance
qmult	queue length multiplier for high and super speed
host_addr	MAC address of host's end of this Ethernet over USB link
dev_addr	MAC address of device's end of this Ethernet over USB link

and after creating the functions/ncm.<instance name> they contain default values: qmult is 5, dev_addr and host_addr are randomly selected. The ifname can be written to if the function is not bound. A write must be an interface pattern such as "usb%d", which will cause the net core to choose the next free usbX interface. By default, it is set to "usb%d".

Testing the NCM function

Configure IP addresses of the device and the host. Then:

On the device:

```
ping <host's IP>
```

On the host:

```
ping <device's IP>
```

11. OBEX function

The function is provided by usb_f_obex.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "obex". The OBEX function provides just one attribute in its function directory:

port_num

The attribute is read-only.

There can be at most 4 ACM/generic serial/OBEX ports in the system.

Testing the OBEX function

On device:

```
seriald -f /dev/ttyGS<Y> -s 1024
```

On host:

```
serialc -v <vendorID> -p <productID> -i<interface#> -a1 -s1024 \
        -t<out endpoint addr> -r<in endpoint addr>
```

where seriald and serialc are Felipe's utilities found here:

https://github.com/felipebalbi/usb-tools.git master

12. PHONET function

The function is provided by usb_f_phonet.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "phonet". The PHONET function provides just one attribute in its function directory:

ifname	network device interface name associated with this function instance

Testing the PHONET function

It is not possible to test the SOCK_STREAM protocol without a specific piece of hardware, so only SOCK_DGRAM has been tested. For the latter to work, in the past I had to apply the patch mentioned here:

http://www.spinics.net/lists/linux-usb/msg85689.html

These tools are required:

git://git.gitorious.org/meego-cellular/phonet-utils.git

On the host:

```
$ ./phonet -a 0x10 -i usbpn0
$ ./pnroute add 0x6c usbpn0
$./pnroute add 0x10 usbpn0
$ ifconfig usbpn0 up
```

On the device:

```
$ ./phonet -a 0x6c -i upnlink0
$ ./pnroute add 0x10 upnlink0
$ ifconfig upnlink0 up
```

Then a test program can be used:

```
http://www.spinics.net/lists/linux-usb/msg85690.html
```

On the device:

```
$ ./pnxmit -a 0x6c -r
```

On the host:

```
$ ./pnxmit -a 0x10 -s 0x6c
```

As a result some data should be sent from host to device. Then the other way round:

On the host:

```
$ ./pnxmit -a 0x10 -r
```

On the device:

```
$ ./pnxmit -a 0x6c -s 0x10
```

13. RNDIS function

The function is provided by usb_f_rndis.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "rndis". The RNDIS function provides these attributes in its function directory:

ifname	network device interface name associated with this function instance
qmult	queue length multiplier for high and super speed
host_addr	MAC address of host's end of this Ethernet over USB link
dev_addr	MAC address of device's end of this Ethernet over USB link

and after creating the functions/rndis.<instance name> they contain default values: qmult is 5, dev_addr and host_addr are randomly selected. The ifname can be written to if the function is not bound. A write must be an interface pattern such as "usb%d", which will cause the net core to choose the next free usbX interface. By default, it is set to "usb%d".

Testing the RNDIS function

Configure IP addresses of the device and the host. Then:

On the device:

```
ping <host's IP>
```

On the host:

```
ping <device's IP>
```

14. SERIAL function

The function is provided by usb_f_gser.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "gser". The SERIAL function provides just one attribute in its function directory:

port_num

The attribute is read-only.

There can be at most 4 ACM/generic serial/OBEX ports in the system.

Testing the SERIAL function

On host:

```
insmod usbserial
echo VID PID >/sys/bus/usb-serial/drivers/generic/new_id
```

On host:

```
cat > /dev/ttyUSB<X>
```

On target:

```
cat /dev/ttyGS<Y>
```

then the other way round

On target:

```
cat > /dev/ttyGS<Y>
```

On host:

```
cat /dev/ttyUSB<X>
```

15. SOURCESINK function

The function is provided by usb_f_ss_lb.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "SourceSink". The SOURCESINK function provides these attributes in its function directory:

pattern	0 (all zeros), 1 (mod63), 2 (none)
isoc_interval	1..16
isoc_maxpacket	0 - 1023 (fs), 0 - 1024 (hs/ss)
isoc_mult	0..2 (hs/ss only)
isoc_maxburst	0..15 (ss only)
bulk_buflen	buffer length
bulk_qlen	depth of queue for bulk
iso_qlen	depth of queue for iso

Testing the SOURCESINK function

device: run the gadget

host: test-usb (tools/usb/testusb.c)

16. UAC1 function (legacy implementation)

The function is provided by usb_f_uac1_legacy.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "uac1_legacy". The uac1 function provides these attributes in its function directory:

audio_buf_size	audio buffer size
fn_cap	capture pcm device file name
fn_cntl	control device file name
fn_play	playback pcm device file name
req_buf_size	ISO OUT endpoint request buffer size
req_count	ISO OUT endpoint request count

The attributes have sane default values.

Testing the UAC1 function

device: run the gadget

host:

```
aplay -l # should list our USB Audio Gadget
```

17. UAC2 function

The function is provided by usb_f_uac2.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "uac2". The uac2 function provides these attributes in its function directory:

c_chmask	capture channel mask
c_srate	list of capture sampling rates (comma-separated)
c_ssize	capture sample size (bytes)
c_sync	capture synchronization type (async/adaptive)
c_mute_present	capture mute control enable
c_volume_present	capture volume control enable
c_volume_min	capture volume control min value (in 1/256 dB)
c_volume_max	capture volume control max value (in 1/256 dB)
c_volume_res	capture volume control resolution (in 1/256 dB)
c_hs_bint	capture bInterval for HS/SS (1-4: fixed, 0: auto)

fb_max	maximum extra bandwidth in async mode
p_chmask	playback channel mask
p_srate	list of playback sampling rates (comma-separated)
p_ssize	playback sample size (bytes)
p_mute_present	playback mute control enable
p_volume_present	playback volume control enable
p_volume_min	playback volume control min value (in 1/256 dB)
p_volume_max	playback volume control max value (in 1/256 dB)
p_volume_res	playback volume control resolution (in 1/256 dB)
p_hs_bint	playback bInterval for HS/SS (1-4: fixed, 0: auto)
req_number	the number of pre-allocated request for both capture and playback
function_name	name of the interface

The attributes have sane default values.

Testing the UAC2 function

device: run the gadget host: aplay -l# should list our USB Audio Gadget

This function does not require real hardware support, it just sends a stream of audio data to/from the host. In order to actually hear something at the device side, a command similar to this must be used at the device side:

```
$ arecord -f dat -t wav -D hw:2,0 | aplay -D hw:0,0 &
```

e.g.:

```
$ arecord -f dat -t wav -D hw:CARD=UAC2Gadget,DEV=0 | \
  aplay -D default:CARD=OdroidU3
```

18. UVC function

The function is provided by usb_f_uvc.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "uvc". The uvc function provides these attributes in its function directory:

streaming_interval	interval for polling endpoint for data transfers
streaming_maxburst	bMaxBurst for super speed companion descriptor
streaming_maxpacket	maximum packet size this endpoint is capable of sending or receiving when this configuration is selected

There are also "control" and "streaming" subdirectories, each of which contain a number of their subdirectories. There are some sane defaults provided, but the user must provide the following:

control header	create in control/header, link from control/class/fs and/or control/class/ss
streaming header	create in streaming/header, link from streaming/class/fs and/or streaming/class/hs and/or streaming/class/ss
format description	create in streaming/mjpeg and/or streaming/uncompressed
frame description	create in streaming/mjpeg/<format> and/or in streaming/uncompressed/<format>

Each frame description contains frame interval specification, and each such specification consists of a number of lines with an inverval value in each line. The rules stated above are best illustrated with an example:

```
# mkdir functions/uvc.usb0/control/header/h
# cd functions/uvc.usb0/control/
# ln -s header/h class/fs
# ln -s header/h class/ss
# mkdir -p functions/uvc.usb0/streaming/uncompressed/u/360p
# cat <<EOF > functions/uvc.usb0/streaming/uncompressed/u/360p/dwFrameInterval
666666
1000000
5000000
EOF
# cd $GADGET_CONFIGFS_ROOT
# mkdir functions/uvc.usb0/streaming/header/h
# cd functions/uvc.usb0/streaming/header/h
# ln -s ../../uncompressed/u
# cd ../../class/fs
```

```
# ln -s ../../header/h
# cd ../../class/hs
# ln -s ../../header/h
# cd ../../class/ss
# ln -s ../../header/h
```

Testing the UVC function

device: run the gadget, modprobe vivid:

```
# uvc-gadget -u /dev/video<uvc video node #> -v /dev/video<vivid video node #>
```

where uvc-gadget is this program:
http://git.ideasonboard.org/uvc-gadget.git

with these patches:

http://www.spinics.net/lists/linux-usb/msg99220.html

host:

```
luvcview -f yuv
```

19. PRINTER function

The function is provided by usb_f_printer.ko module.

Function-specific configfs interface

The function name to use when creating the function directory is "printer". The printer function provides these attributes in its function directory:

pnp_string	Data to be passed to the host in pnp string
q_len	Number of requests per endpoint

Testing the PRINTER function

The most basic testing:

device: run the gadget:

```
# ls -l /devices/virtual/usb_printer_gadget/
```

should show g_printer<number>.

If udev is active, then /dev/g_printer<number> should appear automatically.

host:

If udev is active, then e.g. /dev/usb/lp0 should appear.

host->device transmission:

device:

```
# cat /dev/g_printer<number>
```

host:

```
# cat > /dev/usb/lp0
```

device->host transmission:

```
# cat > /dev/g_printer<number>
```

host:

```
# cat /dev/usb/lp0
```

More advanced testing can be done with the prn_example described in Documentation/usb/gadget_printer.rst.

20. UAC1 function (virtual ALSA card, using u_audio API)

The function is provided by usb_f_uac1.ko module. It will create a virtual ALSA card and the audio streams are simply sinked to and sourced from it.

Function-specific configfs interface

The function name to use when creating the function directory is "uac1". The uac1 function provides these attributes in its function directory:

c_chmask	capture channel mask
c_srate	list of capture sampling rates (comma-separated)
c_ssize	capture sample size (bytes)
c_mute_present	capture mute control enable
c_volume_present	capture volume control enable
c_volume_min	capture volume control min value (in 1/256 dB)
c_volume_max	capture volume control max value (in 1/256 dB)
c_volume_res	capture volume control resolution (in 1/256 dB)
p_chmask	playback channel mask
p_srate	list of playback sampling rates (comma-separated)
p_ssize	playback sample size (bytes)
p_mute_present	playback mute control enable
p_volume_present	playback volume control enable
p_volume_min	playback volume control min value (in 1/256 dB)
p_volume_max	playback volume control max value (in 1/256 dB)
p_volume_res	playback volume control resolution (in 1/256 dB)
req_number	the number of pre-allocated requests for both capture and playback
function_name	name of the interface

The attributes have sane default values.

Testing the UAC1 function

device: run the gadget host: aplay -l# should list our USB Audio Gadget

This function does not require real hardware support, it just sends a stream of audio data to/from the host. In order to actually hear something at the device side, a command similar to this must be used at the device side:

```
$ arecord -f dat -t wav -D hw:2,0 | aplay -D hw:0,0 &
```

e.g.:

```
$ arecord -f dat -t wav -D hw:CARD=UAC1Gadget,DEV=0 | \
  aplay -D default:CARD=OdroidU3
```