Usbgeneralpage

Linux USB Configuration

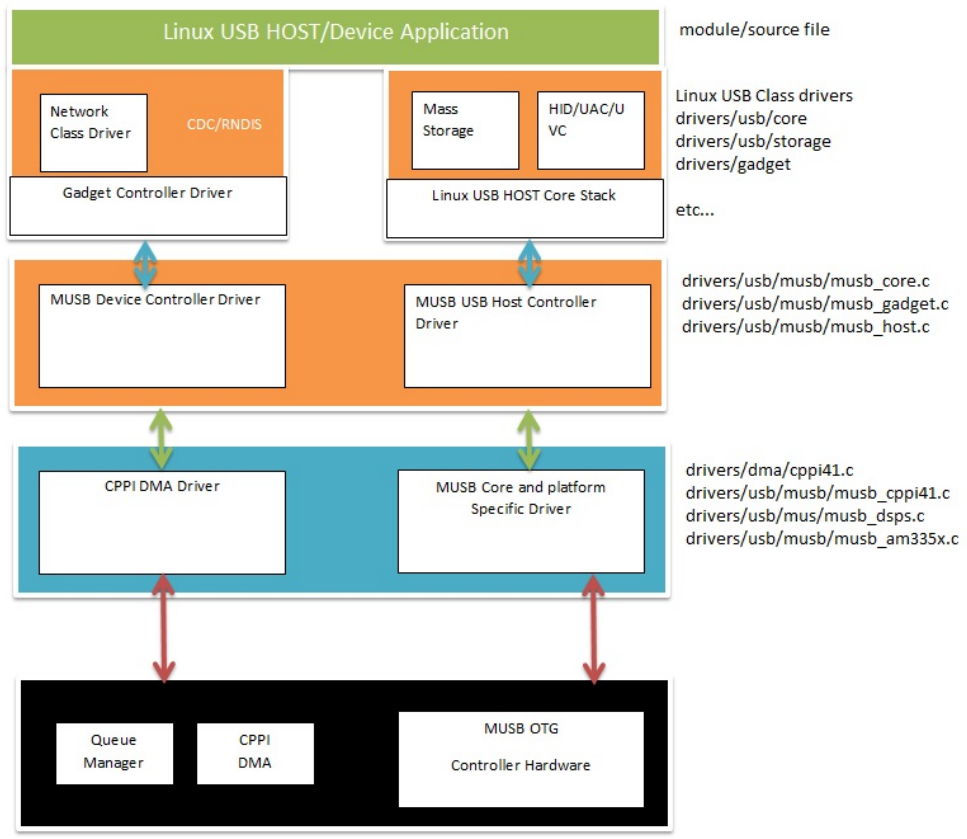
Linux PSP

USB Driver

Introduction

Linux USB Stack Architecture

Linux usb stack is an layered architecture in which musb controller hardware is at the lowest layer. The musb controller driver abstract the musb controller hardware to linux usb stack.


Linux USB HOST/Device Application
module/source file
Network Class Driver
CDC/RNDIS
Gadget Controller Driver
Mass Storage
HID/UAC/UVC
Linux USB HOST Core Stack
Linux USB Class drivers
drivers/usb/core
drivers/usb/storage
drivers/gadget
etc...
MUSB Device Controller Driver
MUSB USB Host Controller Driver
drivers/usb/musb/musb_core.c
drivers/usb/musb/musb_gadget.c
drivers/usb/musb/musb_host.c
CPPI DMA Driver
MUSB Core and platform Specific Driver
drivers/dma/cppi41.c
drivers/usb/musb/musb_cppi41.c
drivers/usb/mus/musb_dsps.c
drivers/usb/musb/musb_am335x.c
Queue Manager
CPPI DMA
MUSB OTG Controller Hardware

This page being common across all TI platforms describes the configuration of USB in linux menuconfig. Specific sections will be used for different platform to mention the differences with other platform.

Driver configuration

The MUSB OTG controller can be used in Host only, Peripheral(Gadget) only or OTG (both host and device) modes. The following section shows the configuration options for USB and its associated class drivers.

To configure the USB Driver Features through menuconfig

Use menuconfig to configure the USB driver features supported in kernel.

```
   make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- menuconfig
```

Goto Menuconfig->Device Drivers

```
Device Drivers --->
....
[ ] HID Devices  --->
[*] USB support  --->
....
```

TI81XX/AM38XX/TI811X EVM default configuration

By default the usb host mode configuration is selected in default evm defconfig is available for DM816X and DM814X. The user can chose default evm defconfig by running

for TI816X

```
# make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- ti8168_evm_defconfig
```

for TI814X

```
# make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- ti8148_evm_defconfig
```

for TI811X

```
# make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- ti811x_evm_defconfig
```

USB phy selection for MUSB OTG port

Please select NOP USB transceiver for MUSB support on all platform except Beagle or Beagle-XM which uses USB phy on TPS65950 (aka TWL4030).

```
Device Drivers --->
USB support --->
*** OTG and related infrastructure ***
[ ] GPIO based peripheral-only VBUS sensing 'transceiver'
[ ] Philips ISP1301 with OMAP OTG
[ ] TWL4030 USB Transceiver Driver
[*] NOP USB Transceiver Driver
```

USB controller in host mode

TI81XX/TI811X

In host only mode configuration, by default both ports usb0/usb1 are configured for host mode.

MUSB OTG Host Configuration

```
Device Drivers --->
USB support --->
<*> Support for Host-side USB
    *** Miscellaneous USB options ***
[*] USB device filesystem
[*] USB device class-devices (DEPRECATED)
    *** USB Host Controller Drivers ***
<*>   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
  *** Platform Glue Layer ***
   < >     TUSB6010
   < >     OMAP2430 and onwards
   < >     AM35x
   <*>     TI81XX
           TI816X usb connector's ID pin control (from software setting)  --->
           Driver Mode (USB Host)  --->
   [ ]   Disable DMA (always use PIO)
   [*]   Enable debugging messages
```

- To enable musb driver in PIO mode, select

```
[*] Disable DMA (always use PIO)
```

- To enable musb driver in DMA mode, unselect

```
[ ] Disable DMA (always use PIO)
```

Note: for DM816X you need to select usb-id pin configuration as mentioned in usb-id configuration section [1].

USB-ID pin configuration selection for TI81XX

The DM816X silicon includes the USB-ID configuration changes, where USB-ID can be controlled only through

1. software controlled

To select software controlled USB-ID configuration (applicable for TI816X and TI814x PG1.x revision only)

```
  Menuconfig->Device Drviers->USB Support
          .....
         <*>   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
                      *** Platform Glue Layer ***
         < >     TUSB6010
         < >     OMAP2430 and onwards
         < >     AM35x
         <*>     TI81XX
                 TI816X usb connector's ID pin control (from software setting)  --->
```

To select USB-ID configuration from external connector (**applicable for TI814X PG2.x/TI811X**)

```
  Menuconfig->Device Drviers->USB Support
          .....
         <*>   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
                      *** Platform Glue Layer ***
         < >     TUSB6010
         < >     OMAP2430 and onwards
         < >     AM35x
         <*>     TI81XX
                 TI816X usb connector's ID pin control (from usb connector)  --->
```

OMAP35x/AM35x/AM37x/AM45x

MUSB OTG Host Configuration

```
Device Drivers --->
USB support --->
<*> Support for Host-side USB
    *** Miscellaneous USB options ***
[*] USB device filesystem
[*] USB device class-devices (DEPRECATED)
    *** USB Host Controller Drivers ***
<*> Inventra Highspeed Dual Role Controller (TI, ADI, ...)
          *** Platform Glue Layer ***
<>     TUSB6010
<*>     OMAP2430 and onwards
<>     AM35x
          Driver Mode (USB Host) --->
[ ] Disable DMA (always use PIO)
[*] Use System DMA for Mentor DMA workaround
[*] Enable debugging messages
```

- Please enable "Use System DMA for Mentor DMA workaround" on OMAP35x/AM37x as a workaround to Mentor DMA issues.
- Select AM35x for AM35x based platforms

```
[ ] TUSB6010
[ ] OMAP2430 and onwards
[*] AM35x
```

- To enable musb driver in PIO mode, select

```
[*] Disable DMA (always use PIO)
```

- To enable musb driver in DMA mode, unselect

```
[ ] Disable DMA (always use PIO)
```

EHCI Configuration (Not supported for TI81XX/TI811X)

```
Device Drivers --->
USB support --->
<*> Support for Host-side USB
    *** Miscellaneous USB options ***
[*] USB device filesystem
[*] USB device class-devices (DEPRECATED)
<*> EHCI HCD (USB2.0) Support
[ ] Root hub transaction translators
[*] Improved Transaction Translator scheduling
    *** USB Host Controller Drivers ***
```

MUSB controller in gadget mode

TI81XX/TI811X

In USB Gadget only configuration, only the usb0 port of TI816x/TI814x/TI813x/TI811x is configured for USB Gadget/Device Mode, usb1 port is unused. To enable each port as device/host mode, then you need to chose OTG configuration please refer here [2].

Note: For TI816X you need to select usb-id pin configuration as mentioned in usb-id configuration section [1].

Configuration

```
Device Drivers --->
USB support --->
      < > Support for Host-side USB
      <*> Inventra Highspeed Dual Role Controller (TI, ADI, ...)
          *** Platform Glue Layer ***
      < > TUSB6010
      < > OMAP2430 and onwards
      < > AM35x
      <*> TI81XX
             Driver Mode (USB Peripheral (gadget stack))  --->
      [ ] Disable DMA (always use PIO)
      [*] Enable debugging messages
              *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
      <*> USB Gadget Support  --->
            *** OTG and related infrastructure ***
      < > GPIO based peripheral-only VBUS sensing 'transceiver'
      [ ] Generic ULPI Transceiver Driver
      -*- NOP USB Transceiver Driver
```

Note: When Inventra HDRC is selected as USB peripheral controller, then must select the Inventra HDRC in "USB Gadget support page" as show below.

```
--- USB Gadget Support
      [ ]   Debugging messages (DEVELOPMENT) (NEW)
      [ ]   Debugging information files (DEVELOPMENT) (NEW)
      [ ]   Debugging information files in debugfs (DEVELOPMENT) (NEW)
      (2)    Maximum VBUS Power usage (2-500 mA) (NEW)
             USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
```

For Gadget RNDIS Mode configuration

```
       --- USB Gadget Support
            USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
       <*>   USB Gadget Drivers (Ethernet Gadget (with CDC Ethernet support))  --->
             Ethernet Gadget (with CDC Ethernet support) (NEW)
       [*]       RNDIS support (NEW)
```

For Gadget FileStorage configuration

```
       --- USB Gadget Support
            USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
       <M> USB Gadget Drivers
       <M> File-backed Storage Gadget
       [*] File-backed Storage Gadget testing version
```

Note: when the gadget driver built as modules, the usb gadget driver module are available at /drivers/usb/gadget directory and you need to insert the gadget driver after kernel is booted.

Please make sure that Inventra HDRC is selected as USB peripheral controller which will appear only when "USB Peripheral (gadget stack)" is selected in driver mode as shown below so after selecting Gadget Support go back to driver mode option to select "USB Peripheral (gadget stack)" and then come back again to select Inventra HDRC as USB peripheral controller.

```
--- USB Gadget Support
      [ ] Debugging messages (DEVELOPMENT)
      [ ] Debugging information files (DEVELOPMENT)
      [ ] Debugging information files in debugfs (DEVELOPMENT)
           (2)   Maximum VBUS Power usage (2-500 mA)
           USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
      <M> USB Gadget Drivers
      <M> Gadget Zero (DEVELOPMENT)
      [*] HNP Test Device
      < > Audio Gadget (EXPERIMENTAL)
      <M> Ethernet Gadget (with CDC Ethernet support)
      [*] RNDIS support
```

OMAP35x/AM35x/AM37x/AM45x

Please do not disable support for host side usb as this will disable EHCI host interface also. Gadget option in driver mode will appear only when gadget support is also selected. Please enable gadget support as given below.

```
Device Drivers --->
USB support --->
  <*> USB Gadget Support --->
      [ ] Debugging messages (DEVELOPMENT)
      [ ] Debugging information files (DEVELOPMENT)
      [ ]   Debugging information files in debugfs (DEVELOPMENT)
      (2) Maximum VBUS power usage (2-500mA) NEW
```

```
   USB Peripheral Controller (Inventra HDRC Peripheral(TI,ADI, ...)) --->
   <M> USB Gadget Drivers
   <M> File-backed Storage Gadget
```

Please make sure that Inventra HDRC is selected as USB peripheral controller which will appear only when "USB Peripheral (gadget stack)" is selected in driver mode as shown below so after selecting Gadget Support go back to driver mode option to select "USB Peripheral (gadget stack)" and then come back again to select Inventra HDRC as USB peripheral controller.

```
 Device Drivers --->
 USB support --->
 <*> Support for Host-side USB
     *** Miscellaneous USB options ***
 [*] USB device filesystem
 [*] USB device class-devices (DEPRECATED)
     *** USB Host Controller Drivers ***
 <*> Inventra Highspeed Dual Role Controller (TI, ADI, ...)
           **** Platform Glue Layer ***
 <>     TUSB6010
 <*>     OMAP2430 and onwards
 <>     AM35x
           Driver Mode (USB Peripheral (gadget stack)) --->
 [ ] Disable DMA (always use PIO)
 [*] Use System DMA for Mentor DMA workaround
 [*] Enable debugging messages
```

MUSB OTG controller in OTG mode

TI81XX/TI811X

OTG not supported.

OMAP35x/AM35x/AM37x/AM45x

OTG Configuration

Both Host and Gadget driver should be selected for OTG support.If gadget driver is built as module then the host side module will be initialized only after gadget module is inserted after bootup.

If "Rely on targeted peripheral list" is also selected then make sure to update `drivers/usb/core/otg_whitelist.h` with the desired supported device class identification ids.

OTG option in driver mode will appear only when gadget support is also selected.Please enable gadget support as given below.

```
  Device Drivers --->
  USB support --->
   <*> USB Gadget Support --->
       [ ] Debugging messages (DEVELOPMENT)
       [ ] Debugging information files (DEVELOPMENT)
       [ ]   Debugging information files in debugfs (DEVELOPMENT)
       (2) Maximum VBUS power usage (2-500mA) NEW
   USB Peripheral Controller (Inventra HDRC Peripheral(TI,ADI, ...)) --->
   <M> USB Gadget Drivers
   <M> File-backed Storage Gadget
```

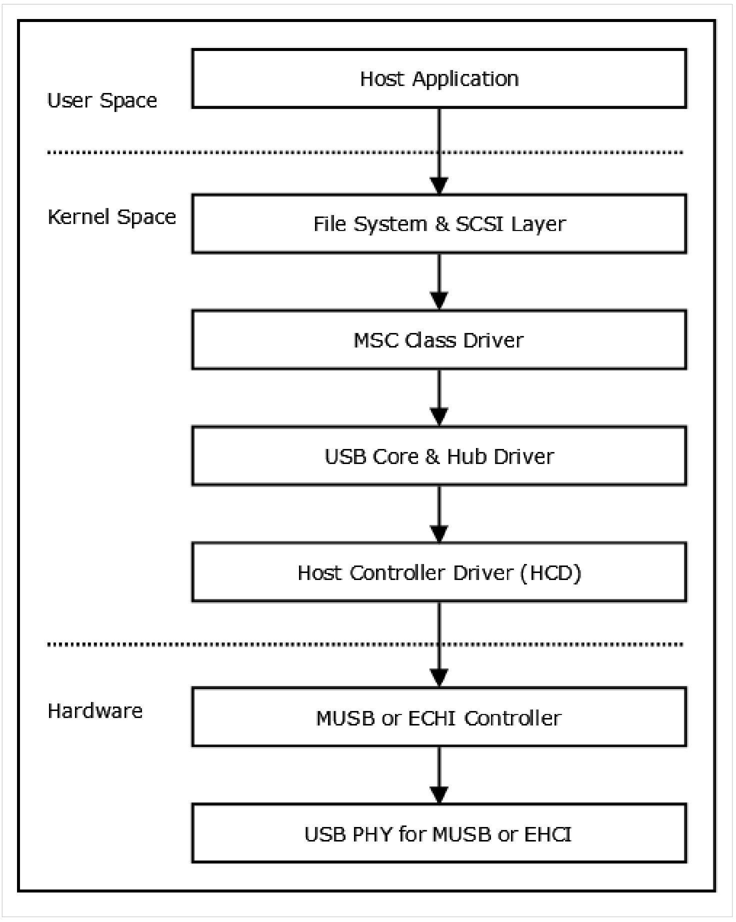
Please make sure that Inventra HDRC is selected as USB peripheral controller which will appear only when OTG is selected as below.

```
 Device Drivers --->
 USB support --->
 <*> Support for Host-side USB
     *** Miscellaneous USB options ***
 [*] USB device filesystem
 [*] USB device class-devices (DEPRECATED)
     *** USB Host Controller Drivers ***
 <*> Inventra Highspeed Dual Role Controller (TI, ADI, ...)
           **** Platform Glue Layer ***
 <>     TUSB6010
 <*>     OMAP2430 and onwards
 <>     AM35x
           Driver Mode (Both Host and peripheral : USB OTG (On The Go) Device) --->
 [ ] Disable DMA (always use PIO)
 [*] Use System DMA for Mentor DMA workaround
 [*] Enable debugging messages
```

Host mode applications

Mass Storage Driver

This figure illustrates the stack diagram of the system with USB Mass Storage class.



Configuration

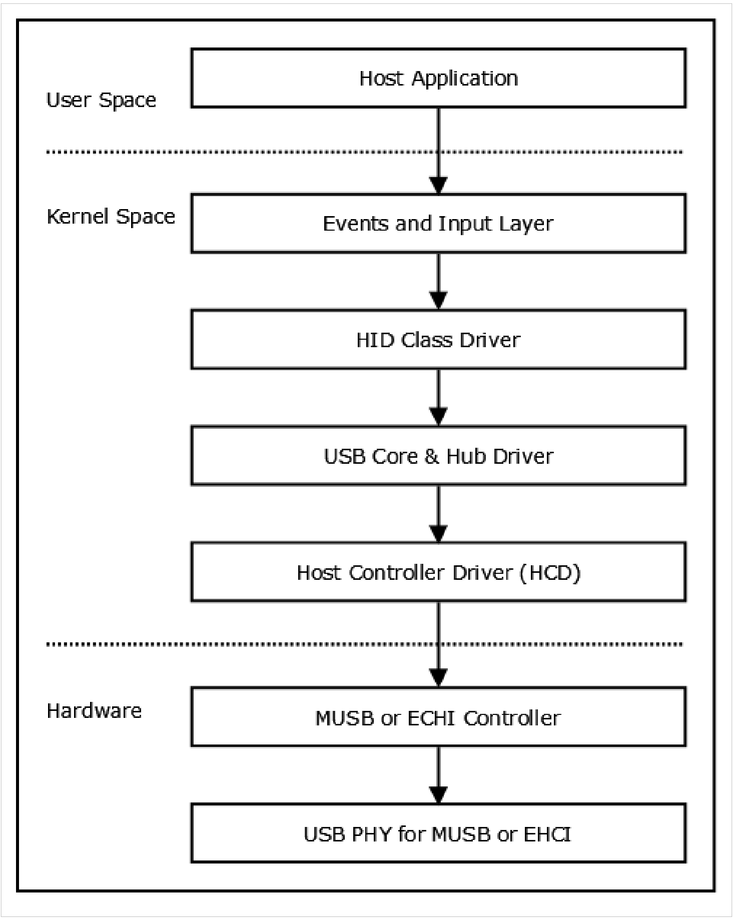
```
    Device Drivers --->
    SCSI device support --->
     <*> SCSI device support
     [*] legacy /proc/scsi/support
     --- SCSI support type (disk, tape, CD-ROM)
     <*> SCSI disk support
    USB support --->
     <*> Support for Host-side USB
     [...]
      --- USB Device Class drivers
      <*> USB Mass Storage support
```

Device nodes

The SCSI sub system creates /dev/sd* devices with help of mdev. For example when USB stick or HDD is inserted /dev/sda1 will be created. Use fdisk utility to create a partition, mkfs.<vfat/ext2> to format the device with vfat/ext2 file system, use mount command for mounting the usb mass storage device.

USB HID Class

USB Mouse and Keyboards that conform to the USB HID specifications are supported.



Configuration

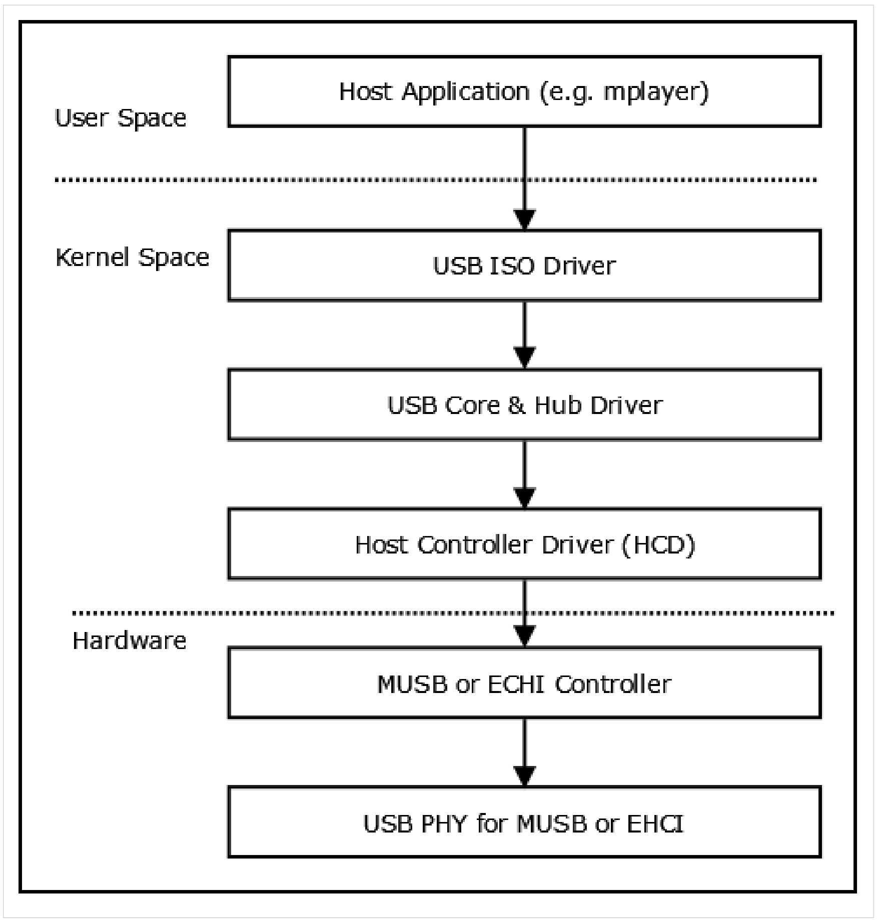
```
    Device Drivers --->
    HID Devices --->
     <*> Generic HID Support
               *** USB Input Devices ***
     <*> USB Human Interface Device(full HID) support
```

Device nodes

The event sub system creates /dev/input/event* devices with the help of mdev. When mouse or keyboard is connected the device nodes with /dev/input/event[0/1/2..] will be created. The HID events can be captured with application. Usage: ./evtest /dev/input/event*

USB Audio

The image below shows the USB stack architecture with USB Audio/Video class.



Configuration

```
Device Drivers --->
Sound card support--->
 <*> Sound card support
   Advanced Linux Sound Architecture --->
   <*> Advanced Linux Sound Architecture
      [*] USB sound devices --->
      <*> USB Audio/MIDI driver
```

Resources

For testing USB Audio support we need any ALSA compliant audio player/capture application. Kindly read the Audio driver section to get more inputs on this.

arecord and aplay tools can be used for record and playing audio stream to/from usb audio devices.

To list the audio output (speaker) devices available use

```
#aplay -l
```

For playing audio on usb audio speaker.

```
#aplay -D "hw:0,0" -f S16_LE -r 11025 <samplefile.wav>
```

To list the audio input (mike) devices available use

```
#arecord -l
```

For recording audio from usb audio input/mike device.

```
#arecord -D "hw:0,0" -f S16_LE -r 16000 <samplefile.wav>
```

```
Note: use the appropriate device number "hw:0,0" or "hw:1,0" or "hw:2,0", use "aplay -l" or "arecord -l" to get list of available audio devices/
```

USB Video

Configuration

```
Device Drivers --->
  Multimedia support --->
        *** Multimedia core support ***
    <*> Video for Linux
    [*] Enable Video For Linux API 1 (DEPRECATED)
    [*] V4L2 sub-device userspace API (EXPERIMENTAL)
        *** Multimedia Drivers ***
    [*] Video capture adapters --->
        [*] V4L USB devices --->
            <*> USB Video Class (UVC)
             [*]     UVC input events device support
```

Resources

For testing USB Video support we need a user level application like "mplayer" to stream video from an USB camera. If you are using mplayer as the capture application, then you must export the DISPLAY to a X server. Then, execute the following command:

```
$ mplayer tv:// -tv driver=v4l2:width=320:height=240
```

You can use the yavta [3] tool to capture frames from USB webcam. Please check if the tool is already available in the filesystem provided with your SDK before proceeding to building it yourself. For example:

```
$ yavta -n30 -c300 -s 640x480 -f mjpeg -F /dev/video2
```

To view the frames capture by yavta on the host, you can use gstreamer.

```
# gst-launch-1.0 multifilesrc location=frame-%06d.bin index=0 stop-index=300 ! jpegdec ! videoconvert ! autovideosink
```

USB CDC-HOST

The CDC-Host configuration will enable the CDC-ACM host class driver, which support network communication devices like USB modems.

Configuration

Select usb host configuration through menuconfig as explained in host configuration section [4] and select CDC ACM support as mentioned below.

```
 Device Drivers --->
 USB support --->
      <*>   Support for Host-side USB
      ....
      *** USB Device Class drivers ***
        <*> USB Modem (CDC ACM) support
        < > USB Printer support
        < > USB Wireless Device Management support
        < > USB Test and Measurement Class support
```

```
Device Drivers --->
  Network device supports  --->
    USB Network Adapters  --->
      <*> Multi-purpose USB Networking Framework
      < >   ASIX AX88xxx Based USB 2.0 Ethernet Adapters
      <*>   CDC Ethernet support (smart devices such as cable modems)
      < >   CDC EEM support
```

Resources

The test setup includes connnecting the two EVMs, EVM-1 is configured in host mode (enabling CDC-host configuration as explained above) and EVM-2 is configured in CDC-device mode configuration (refer cdc-configuration [5]). Connect EVM-2 (CDC-device act as CDC-modem) to EVM-1 (CDC-Host), the EVM-1 CDC-Host will rececognize the EVM-2 as usb-modem and creates the new **usbx** network interface.

on EVM-2 (CDC-Device)

```
#ifconfig usb0 192.168.0.3 up
```

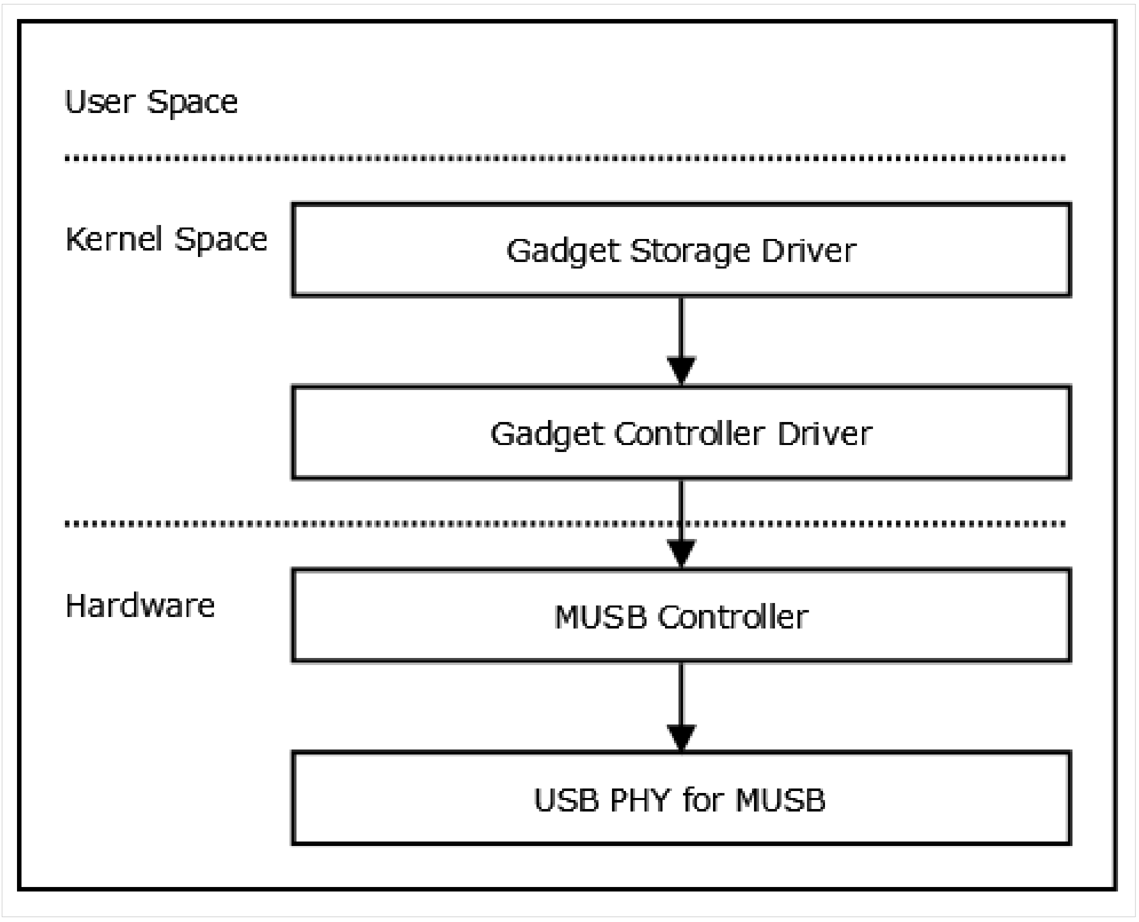
on EVM-1 (CDC-Host): A new interface say usb0 or usb1 will be created.

```
#ifconfig usbx 192.168.0.5 up
```

perform the ping from EVM-1 to EVM-2 (vice-versa).

Gadget Mode Applications

Mass Storage Gadget



Configuration

1. Select USB Gadget support

```
  Device Drivers --->
  USB support --->
  < >   Support for Host-side USB
    *** Enable Host or Gadget support to see Inventra options ***
    *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
  <*>   USB Gadget Support  --->
   *** OTG and related infrastructure ***
```

2. Select device/gadget controller support

```
< >   Support for Host-side USB
   <*>   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
        *** Platform Glue Layer ***
   < >     TUSB6010
   < >     OMAP2430 and onwards
   < >     AM35x
   <*>     TI81XX
   TI81XX usb connector's ID pin control (from software setting)  --->
```

```
      Driver Mode (USB Peripheral (gadget stack))  --->
[ ]   Disable DMA (always use PIO)
[*]   Enable debugging messages
 *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
<*>   USB Gadget Support  --->
      *** OTG and related infrastructure ***
```

3. Select gadget driver, also select USB Peripheral controller

```
Device Drivers --->
USB support --->
USB Gadget Support --->
[ ]   Debugging messages (DEVELOPMENT)
[ ]   Debugging information files (DEVELOPMENT)
[ ]   Debugging information files in debugfs (DEVELOPMENT)
     (2)   Maximum VBUS Power usage (2-500 mA
USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
< >   GPIO based peripheral-only VBUS sensing 'transceiver'
[ ]   Generic ULPI Transceiver Driver
   -*-   NOP USB Transceiver Driver
<M>   USB Gadget Drivers
< > Gadget Zero (DEVELOPMENT)
< >     Audio Gadget (EXPERIMENTAL)
< >     Ethernet Gadget (with CDC Ethernet support)
< >     Gadget Filesystem (EXPERIMENTAL)
< >     Function Filesystem (EXPERIMENTAL)
<M>     File-backed Storage Gadget
[*]       File-backed Storage Gadget testing version
<M>     Mass Storage Gadget
```

Installation of File Storage Gadget Driver

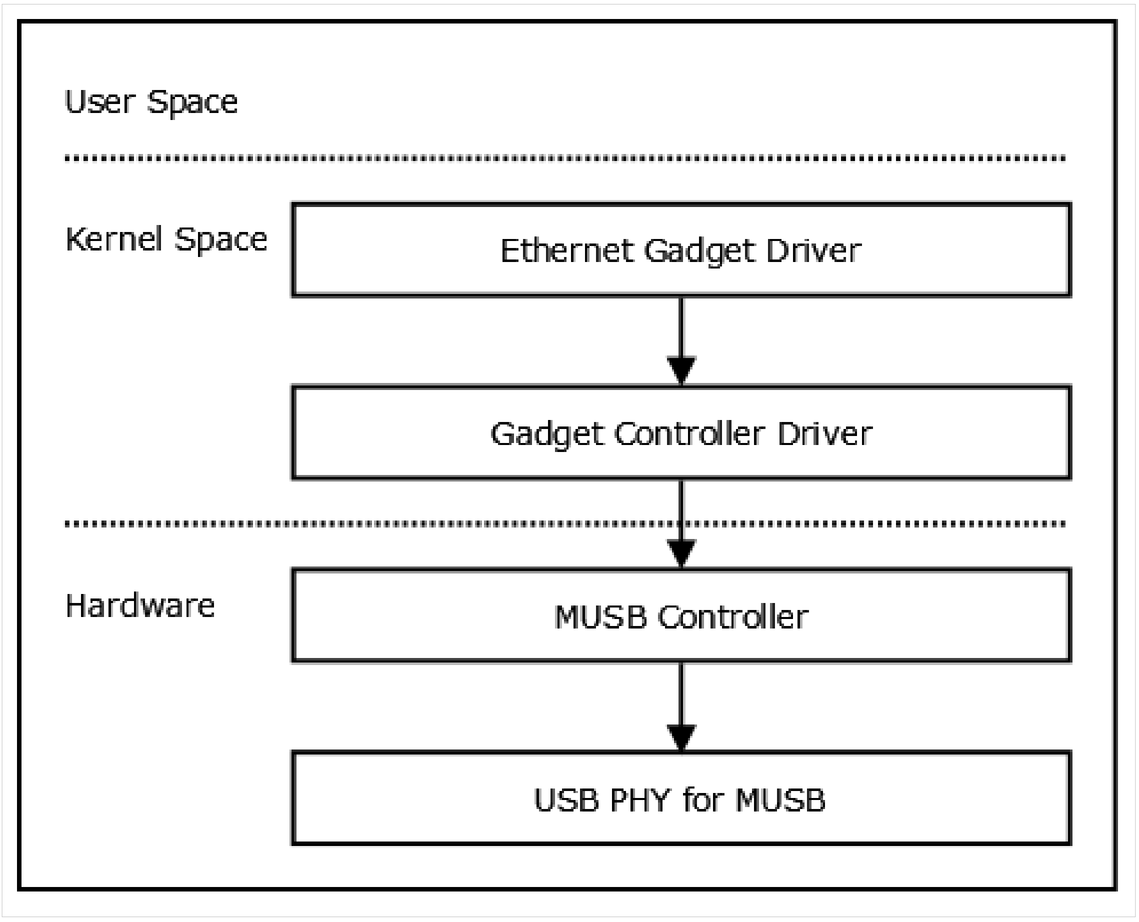
Let us assume that we are interested in exposing /dev/mmcblk0 block device to the file storage gadget device to host (windows/linux). To that effect we need to issue the following command to load the file storage gadget driver. The module parameter **buflen** can be used set buffersize of the gadget driver for better performance, please refer to /driver/usb/gadget/file_storage.c for more information on module parameters.

```
example
#insmod <g_file_storage.ko> file=/dev/sda1 buflen=65536 stall=0 removable=1
#insmod <g_file_storage.ko> file=/dev/mmcblk0 stall=0 removable=1
```

CDC/RNDIS gadget

The CDC/RNDIS gadget driver that is used to send standard Ethernet frames using USB.
For OMAP35x : Please enable "Use System DMA for Rx endpoints" to fix the flood ping hang issue with packet size of more than 16KB on OMAP35x due to Mentor DMA lockup issue.

The image below shows the USB stack architecture with CDC/RNDIS gadget.



Configuration for USB controller and CDC/RNDIS Gadget

```
1. Select USB Gadget support

  Device Drivers --->
  USB support --->
  < >   Support for Host-side USB
    *** Enable Host or Gadget support to see Inventra options ***
    *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
  <*>   USB Gadget Support  --->
   *** OTG and related infrastructure ***
```

2. Select device/gadget controller support

```
< >   Support for Host-side USB
   <*>   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
        *** Platform Glue Layer ***
   < >     TUSB6010
   < >     OMAP2430 and onwards
   < >     AM35x
   <*>     TI81XX
   TI81XX usb connector's ID pin control (from software setting)  --->
       Driver Mode (USB Peripheral (gadget stack))  --->
  [ ]   Disable DMA (always use PIO)
  [*]   Enable debugging messages
   *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
  <*>   USB Gadget Support  --->
       *** OTG and related infrastructure ***
```

3. Select cdc/rndis gadget driver,select 'M' to build gadget as module and '*' for builtin. Also select USB Peripheral controller.

```
  Device Drivers --->
  USB support --->
  USB Gadget Support --->
  [ ]   Debugging messages (DEVELOPMENT)
  [ ]   Debugging information files (DEVELOPMENT)
  [ ]   Debugging information files in debugfs (DEVELOPMENT)
      (2)   Maximum VBUS Power usage (2-500 mA
  USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
 <M>   USB Gadget Drivers (Ethernet Gadget (with CDC Ethernet support))  --->
        Ethernet Gadget (with CDC Ethernet support)
  [*]       RNDIS support
  [ ]       Ethernet Emulation Model (EEM) support
```

Please do not select RNDIS support for testing ethernet gadget with Linux 2.4, IXIA and MACOS host machine.

Installation of CDC/RNDIS Gadget Driver

Inserting the CDC/RNDIS gadget driver as module is as follows:

```
 $ insmod  <path to g_ether.ko>
```

Installing driver on Windows to recognize Linux RNDIS gadget

- Download and save linux.inf file to windows host. The file is available in Linux Kernel source under `Documentation/usb/linux.inf`
- Plug in the device to windows host.
- Open Device Manager and look for the newly recognized USB serial COM port. Right click and choose Update Driver Software
- Follow the steps in below screen shots
- Browse to the `linux.inf` file and click on OK.
- You should see Linux RNDIS gadget recognized by Windows.

NOTE
On windows 10, installing unsigned driver is not permitted by default. Therefore, driver signature verification needs to be turned off. See here. [6]

NOTE
On windows 10, a windows update might remove the untrusted drivers so you will have to re-install the driver in that case.

Setting up USBNet

The CDC/RNDIS Gadget driver will create a Ethernet device by the name usb0. You need to assign an IP address to the device and bring up the device. The typical command for that would be:

```
 $ ifconfig usb0  <IP_ADDR> netmask 255.255.255.0 up
```

UVC Webcam Gadget

Kernel Configuration

1. Enable Multimedia and Camera/video grabbers support

```
  Device Drivers --->
  <M> Multimedia support  --->
     *** Multimedia core support ***
     [*]   Cameras/video grabbers support
```

2. Enable USB Gadget and Webcam Gadget support

```
  Device Drivers --->
  USB support --->
  <M>   USB Gadget Support  --->
     <M>     USB Webcam Gadget
```

uvc-gadget application

The g_webcam gadget provides a V4L2 /dev/videoX interface so that user space can pump camera captured data that needs to be sent over USB to the Host. For testing purposes we can use the uvc-gadget application. uvc-gadget should already be part of the filesystem distribution you received as part of the TI SDK release for your device. If not, to build this application from source, do the following on the target.

1. git clone git://git.ideasonboard.org/uvc-gadget.git
2. Update uvc-gadget.c to point to the correct uvc.h location in your Kernel sources. E.g. #include "/usr/src/ti-linux-kernel/drivers/usb/gadget/function/uvc.h"
3. make

Running UVC Gadget

On the Target

```
  > modprobe g_webcam
  >./uvc-gadget -d /dev/videoX
```

videoX might be video0 or video1 on your platform. It must match the video node that was created when g_webcam was loaded.

On the Host

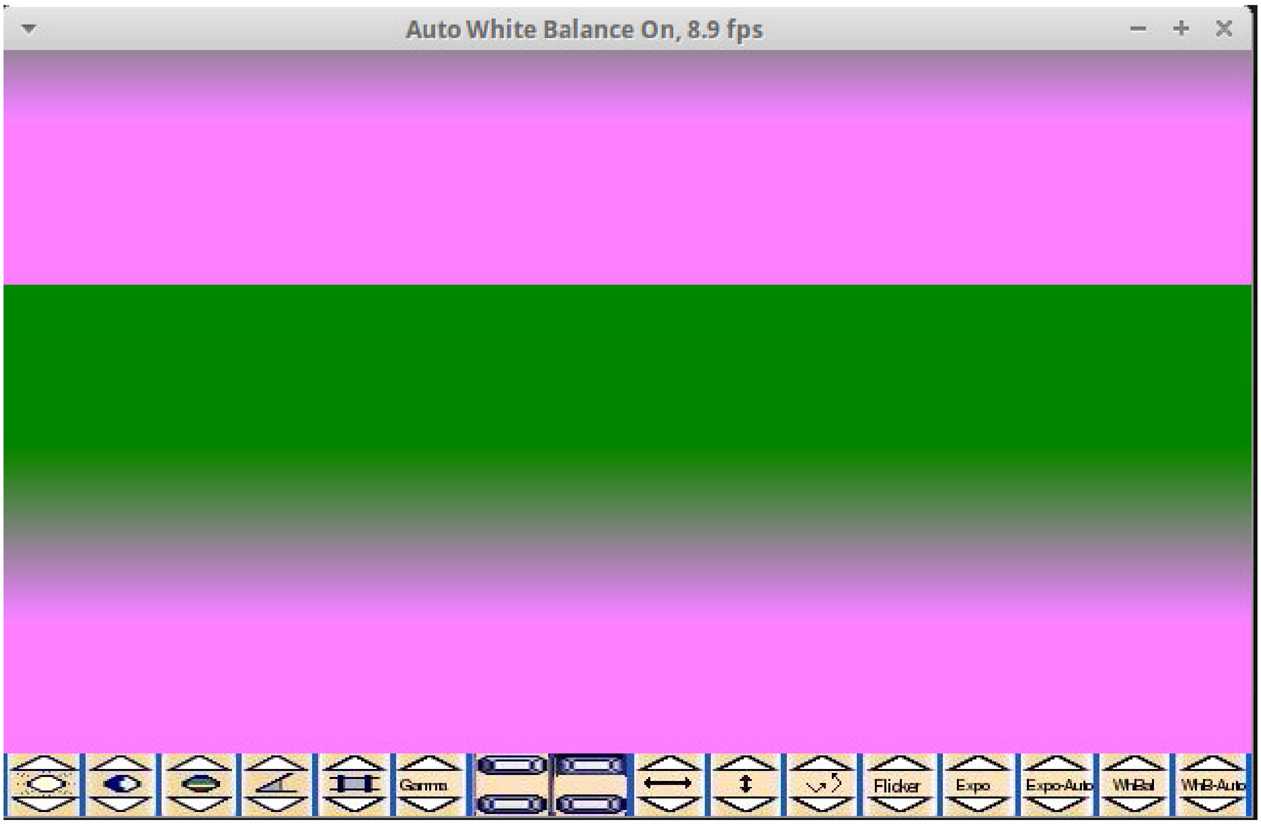
```
luvcview -d /dev/videoY -f yuv
```

OR

```
  guvcview -d /dev/videoY -f yuv
```

videoY might be video0 or video1 on your host. It must match the video node that was created when the target's was connected to Host's USB port.

On the host application you should see a scrolling test pattern like below



Testing different bandwidth settings

g_webcam gadget uses an ISO endpoint to transfer the video stream to the host. This endpoint can have different settings to alter the bandwidth used by it. The below module parameter options can be tweaked wile loading g_webcam.

```
parm:           streaming_interval:1 - 16 (uint)
parm:           streaming_maxpacket:1 - 1023 (FS), 1 - 3072 (hs/ss) (uint)
parm:           streaming_maxburst:0 - 15 (ss only) (uint)
```

NOTE: Not all combinations will work so be careful with these settings. Some known configurations are shown below.

- High-Speed, high-bandwidth transfer test

```
modprobe g_webcam streaming_maxpacket=3062 streaming_interval=1
```

- Super-Speed, burst transfer test

```
modprobe g_webcam streaming_maxpacket=1024 streaming_maxburst=10
```

Modular testing on MUSB

Mentor USB (MUSB) linux driver has been reorganized in v2.6.37 to support multi platform config. Modular structure has also changed due to this and thus now onward there will be below modules on MUSB

```
1. musb_hdrc.ko: The core controller module.
2. cppi30dma.ko, cppi41dma.ko or musbhsdma.ko: The dma controller module.
3. ti81xx.ko, am35x.ko or omap2430.ko: The platform glue module.
4. g_file_storage.ko or g_ether.ko: The gadget controller module.
```

All the above four modules need to be inserted sequentially for an OTG (both host and device) configured or gadget only configured kernel. Fourth module (gadget controller) would not be needed in a host only config.

One port as host and other port as Gadget (for TI81XX/TI813X)

The TI816X/TI814X/TI813X/TI811X has two mentor usb controller and hence each port usb0/usb1 can be configured in host or gadget mode. This feature will provide the user to configure the each port in host/gadget mode.

To configure the usb0/usb1 port to host/gadget mode, do the following steps

1) Select host and gadget support

```
 Menuconfig->Device Drviers->USB Support
        <*>   Support for Host-side USB
        [ ]     USB verbose debug messages
        [*]     USB announce new devices
        *** Miscellaneous USB options ***
        ......
        <*>   USB Gadget Support  --->
```

2) Select USB OTG support (for TI814X/TI813X/TI811X)

```
 Menuconfig->Device Drviers->USB Support
        <*>   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
                    *** Platform Glue Layer ***
        < >     TUSB6010
        < >     OMAP2430 and onwards
        < >     AM35x
        <*>     TI81XX
                TI816X usb connector's ID pin control (from usb connector)  --->
                Driver Mode (Both host and peripheral:  USB OTG (On The Go) Device)  --->
        [ ]   Disable DMA (always use PIO)
        [*]   Enable debugging messages
```

3) Select USB OTG support (for TI816X)

For DM816X, you need to chose the USB-ID configuration through 1) software settings or 2)external connector. In case of software setting for each port usb0 or usb1 you need to configure the mode (Host or Device). You can chose 1) both port as host or 2)one port as host or other as device.

```
 Menuconfig->Device Drviers->USB Support
        <*>   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
                    *** Platform Glue Layer ***
        < >     TUSB6010
        < >     OMAP2430 and onwards
        < >     AM35x
        <*>     TI81XX
                TI816X usb connector's ID pin control (from software setting)  --->
                          Force TI816X USB0 to  (Host mode)  --->
                          Force TI816X USB1 to  (Host mode)  --->
                Driver Mode (Both host and peripheral:  USB OTG (On The Go) Device)  --->
        [ ]   Disable DMA (always use PIO)
```

```
         [*]   Enable debugging messages
```

4) Select Gadget device controller and gadget driver as modules

```
  Menuconfig->Device Drviers->USB Support
         <*>   USB Gadget Support  --->
               --- USB Gadget Support
               [ ]   Debugging messages (DEVELOPMENT) (NEW)
               [ ]   Debugging information files (DEVELOPMENT) (NEW)
               [ ]   Debugging information files in debugfs (DEVELOPMENT) (NEW)
               (2)   Maximum VBUS Power usage (2-500 mA) (NEW)
               USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...))  --->
              <M>   USB Gadget Drivers
              <M>     Gadget Zero (DEVELOPMENT)
              [ ]       HNP Test Device (NEW)
              < >     Audio Gadget (EXPERIMENTAL) (NEW)
              <M>     Ethernet Gadget (with CDC Ethernet support)
              [*]       RNDIS support (NEW)
              [ ]       Ethernet Emulation Model (EEM) support (NEW)
              < >     Gadget Filesystem (EXPERIMENTAL) (NEW)
              < >     Function Filesystem (EXPERIMENTAL) (NEW)
              <M>     File-backed Storage Gadget
              [*]       File-backed Storage Gadget testing version
```

5) Unselect the OTG Targeted Peripherals list

```
  Menuconfig->Device Drviers->USB Support
      <*>   Support for Host-side USB
            ....
           [*] USB runtime power management (autosuspend) and wakeup
           -*- OTG support
           [ ]   Rely on OTG Targeted Peripherals List
           [ ]   Disable external hubs
```

6) Build uImage and usb gadget modules

Build the kernel image and the two usb gadget as modules (like g_ether.ko, g_file_storage.ko, g_mass_storage.ko or g_zero.ko ..etc).

7) Setup for TI81XX/TI814X

If the board has mini-AB or micro-AB receptacle for usb0/usb1 port then

- To configure usb0/usb1 as host mode, connect mini/micro-A to A receptacle to usb0/usb1 port.
- To configure usb0/usb1 as device mode, connect mini/micro-B to standard A cable.

If the board has standard-A receptacle

- To configured usb0/usb1 as host mode , connect device directly or through HUB.
- To configured usb0/usb1 as device mode , connect **Standard A to A USB cable**.

8) Insert the two gadget modules

Load the kernel image and Make sure above setup is done before insert the modules. Insert the gadget modules for usb0 port.

```
 # insmod <module>.ko    (eg: #insert g_ether.ko)
```

Insert the gadget module for usb1 port.

```
  # insmod <module>.ko    (eg: #insert g_file_storage.ko file=<file path> stall=0 buflen=65536)
```

Note: Please note that the usb port will be enabled only after inserting usb gadget module for port0/port1, depending on the cable type connected to each port, the first inserted gadget module will enable usb0 port for to host/device mode, the second inserted gadget module will enable the usb1 port to host/device mode.

In scenario where usb0 as mass-storage gadget mode and usb1 as host mode, and the use-case is to expose the usb flash drive or USB-HDD connected to usb1 port as mass-storage media through usb0 gadget port to windows/linux host. The workaround for this use-case could be

1. insmod g_ether.ko (usb0 port will be enabled as cdc/rndis gadget)
2. insmod g_zero.ko (usb1 port will enabled as host mode and all devices (eg.USB-HDD/flash drive etc) connected to this port will be enumerated
3. rmmod g_ether.ko (remove the gadget driver from usb0 port)
4. insmod g_file_storage.ko file=/dev/sda stall=0 buflen=65536 (or use g_mass_storage.ko)

Now the usb0 port can be connected to window/linux host as mass storage gadget with USB-HDD (/dev/sda) as storage media.

USB EHCI Electrical testing (Not applicable for TI81XX/TI811X)

USB EHCI electrical test is supported in software. Please use below command to perform various electrical tests.

```
$ echo 'Options' > sys/devices/platform/ehci-omap.0/portN
Where 'options' can be,
reset   --> Reset Device
t-j     --> Send TEST_J on suspended port
t-k     --> Send TEST_K on suspended port
t-pkt   --> Send TEST_PACKET[53] on suspended port
t-force --> Send TEST_FORCE_ENABLE on suspended port
t-se0   --> Send TEST_SE0_NAK on suspended port
```

USB OTG (HNP/SRP) testing (Not applicable for TI81XX/TI811X)

Please choose the configuration as described in driver configuration section for OTG and follow the steps below for testing.

- 1. Boot the OTG build image on two OMAP35x EVM.
- 2. If gadget driver is built as module then insert it to complete USB initialization.
- 3. Connect mini-A side of the OTG cable to one of the EVM (say EVM-1) and mini-B side on the other (say EVM-2).

In this scenario EVM-1 will become initial host or A-device and EVM-2 will become initial device or B-device. A-device will provide bus power throughout the bus communication even if it becomes peripheral using HNP.

There will not be any connect event at this point of time as Vbus power is not yet switched-on. Vbus power can be switched-on from A-device or from B-device using SRP.

- 4. Request to switch-on the Vbus power using below command on any EVM.

```
$ echo "F" > /proc/driver/musb_hdrc
```

If this command is executed on B-device then SRP protocol will be used to request A-device to switch-on the Vbus power.

- 5. Now the connect event occurs, enumeration will complete and gadget driver on B-device will be ready to use if this driver is in "Targeted Peripheral List (TPL)" of A-device.

If TPL is disabled on A-device then gadget driver will be ready to use soon after enumeration.

If TPL is enabled and gadget driver of B-device is not in TPL list of A-device then there will be an automatic trial of HNP from usb core by suspending the bus. This will cause a role switch and B-device will enumerate A-device. Now the gadget driver of A-device will be configured if it is on the TPL list of B-device.

Currently this is the only way possible for HNP testing but we have added a suspend proc entry to start HNP in other than this scenario.

- 6. Complete all the communication between A-device and B-device.
- 7. Start HNP by executing below command on host side.

```
$ echo "S" > /proc/driver/musb_hdrc
```

It will suspend the bus and role-switch will follow after that.

- 8. Repeat step 4, 5, 6 and 7 for further testing.

Software Interface

The USB driver exposes its state/control through the sysfs and the procfs interfaces. The following sections talks about these.

sysfs

sysfs attribute	**Description**
mode	The entry `/sys/devices/platform/musb_hdrc.0/mode` is a read-only entry. It will show the state of the OTG (though this feature is not supported) state machine. This will be true even if the driver has been compiled without OTG support. Only the states like A_HOST, B_PERIPHERAL, that makes sense for non-OTG will show up.
vbus	The entry `/sys/devices/platform/musb_hdrc.0/vbus` is a write-only entry. It is used to set the VBUS timeout value during OTG. If the current OTG state is a_wait_bcon then urb submission is disabled.

procfs

The procfs entry `/proc/driver/musb_hdrc` is used to control the driver behaviour as well as check the status of the driver.

```
In case of TI81XX/TI811X two nodes will be created for each controller /proc/driver/musb_hdrc.0 and /proc/driver/musb_hdrc.1 respectively.
```

- 1. The following command will show the usage of this proc entry

```
# echo "?" > /proc/driver/musb_hdrc.x
```

- 2. Specifically the most important usage of this entry would be to start an USB session(host mode) by issuing the following command:

```
# echo "F" > /proc/driver/musb_hdrc.x
```

- 3. To enable DEBUG message from musb driver, issue the following command:

```
# echo "D<num>" > /proc/driver/musb_hdrc : <num> can be 1 to 8.
Use dmesg to get dump of musb driver debug output
```

```
# dmesg
```

```
example (for two instances of musb controller)
# echo D8 > /proc/driver/musb_hdrc.0   //set debug_level 8 for musb contoller-0
# echo D8 > /proc/driver/musb_hdrc.1   //set debug_level 8 for musb contoller-1
# dmesg                                //dump the debug prints on console
```

musb driver debugfs

To use the debugfs feature of kernel and musb, you need to enable the kernel debugfs option through menuconfig, as shown below

```
     Menuconfig->kernel hacking -->
          [ ] Enable unused/obsolete exported symbols
          [*] Debug Filesystem
          [ ] Run 'make headers_check' when building vmlinux
          [*] Kernel debugging
```

mount the debug file system (debugfs)

```
     #mount -t debugfs none /sys/kernel/debug/
```

musb driver TEST-MODE debugfs support

Issue the following command

To force musb to host mode

```
     #echo "force host" > /sys/kernel/debug/testmode
```

To force musb to full-speed

```
     #echo "force full-speed" > /sys/kernel/debug/testmode
```

To force musb to high-speed

```
     #echo "force high-speed" > /sys/kernel/debug/testmode
```

To send test packet

```
     #echo "test packet" > /sys/kernel/debug/testmode
```

To generate test K pattern

```
     #echo "test K" > /sys/kernel/debug/testmode
```

To generate test J pattern

```
     #echo "test J" > /sys/kernel/debug/testmode
```

To generate test SE0 NAK pattern

```
     #echo "test SE0 NAK" > /sys/kernel/debug/testmode
```

References

[1] http://processors.wiki.ti.com/index.php/Usbgeneralpage#USB-ID_pin_configuration_selection_for_DM816X
[2] http://processors.wiki.ti.com/index.php/Usbgeneralpage#One_port_as_host_and_other_port_as_Gadget_.28for_DM81XX.29

[3] http://git.ideasonboard.org/?p=yavta.git;a=summary

[4] http://processors.wiki.ti.com/index.php/Usbgeneralpage#USB_controller_in_host_mode

[5] http://processors.wiki.ti.com/index.php/Usbgeneralpage#CDC.2FRNDIS_gadget

[6] https://answers.microsoft.com/en-us/insider/forum/insider_wintp-insider_devices/how-do-i-disable-driver-signature-enforcement-win/a53ec7ca-bdd3-4f39-a3af-3bd92336d248

Article Sources and Contributors

Usbgeneralpage *Source*: http://processors.wiki.ti.com/index.php?oldid=233692 *Contributors*: AJ, AjayGupta, Ravibabu31, Rogerq, SekharNori, Vigneshr

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source*: http://processors.wiki.ti.com/index.php?title=File:TIBanner.png *License*: unknown *Contributors*: Nsnehaprabha
File:Linuxusbarch3.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Linuxusbarch3.JPG *License*: unknown *Contributors*: Georgecherian, Ravibabu31
Image:usb-msc.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Usb-msc.png *License*: unknown *Contributors*: AjayGupta, SanjeevPremi
Image:usb-hid.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Usb-hid.png *License*: unknown *Contributors*: AjayGupta, SanjeevPremi
Image:usb-iso.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Usb-iso.png *License*: unknown *Contributors*: AjayGupta, SanjeevPremi
Image:usb-gadget.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Usb-gadget.png *License*: unknown *Contributors*: Hvaibhav
Image:usb-cdc.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Usb-cdc.png *License*: unknown *Contributors*: Hvaibhav
Image:Luvcview.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Luvcview.jpg *License*: unknown *Contributors*: Rogerq