USB General Guide Linux v3.8

Linux USB Configuration

GLSDK

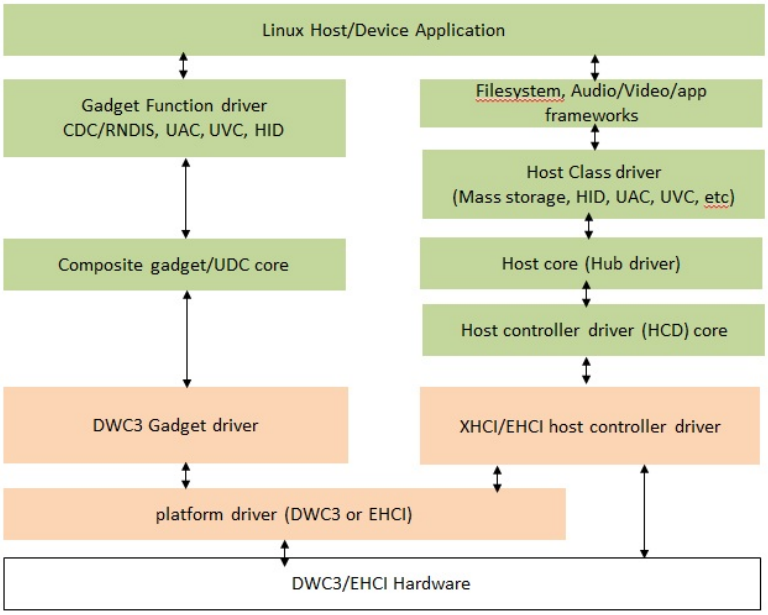
USB Driver

Contents

Introduction

Linux USB Stack Architecture

Linux usb stack is an layered architecture in which host/device controller hardware is at the lowest layer (eg. DWC3 or EHCI or musb). The respective host controller and platform driver (dwc3/ehci/musb controller driver) glues the hardware and rest of usb host/device stack.



This page being common across all TI platforms describes the configuration of USB in linux 3.8 kernel menuconfig. Specific sections will be used for different platform to mention the differences with other platform.

Driver configuration

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-arm-linux-gnueabihf- menuconfig
```

For all USB configuration goto Menuconfig->Device Drivers->"USB support"

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

The default defconfig "omap2plus_defconfig" used for dra7xx/omap5.

```
# make ARCH=arm CROSS_COMPILE=arm-arm-linux-gnueabihf- omap2plus_defconfig
```

Select DWC3 & XHCI host controller

```
Device Drivers --->
USB support --->
  <*>   Support for Host-side USB
  [ ]     USB verbose debug messages
  [*]     USB announce new devices
  *** Miscellaneous USB options ***
  [*]     USB runtime power management (autosuspend) and wakeup
  <*>   DesignWare USB3 DRD Core Support
        DWC3 Mode Selection (Dual Role mode)  --->
  ...
   <*>   xHCI HCD (USB 3.0) support
  [ ]     Debugging for the xHCI host controller
```

Select EHCI host controller

```
Device Drivers --->
USB support --->
  <*>   Support for Host-side USB
  [ ]     USB verbose debug messages
  [*]     USB announce new devices
  *** Miscellaneous USB options ***
  ....
   <*> EHCI HCD (USB 2.0) support
  ....
  <*>  EHCI support for OMAP3 and later chips
```

Select USB2/USB3 phy driver

```
Device Drivers --->
USB support --->
  <*>   Support for Host-side USB
  [ ]     USB verbose debug messages
  [*]     USB announce new devices
  *** Miscellaneous USB options ***
  ...
     *** USB Physical Layer drivers ***
  <*>   OMAP USB2 PHY Driver
  <*>   OMAP USB3 PHY Driver
  -*-   OMAP CONTROL USB Driver
  ...
  -*-   NOP USB Transceiver Driver
```

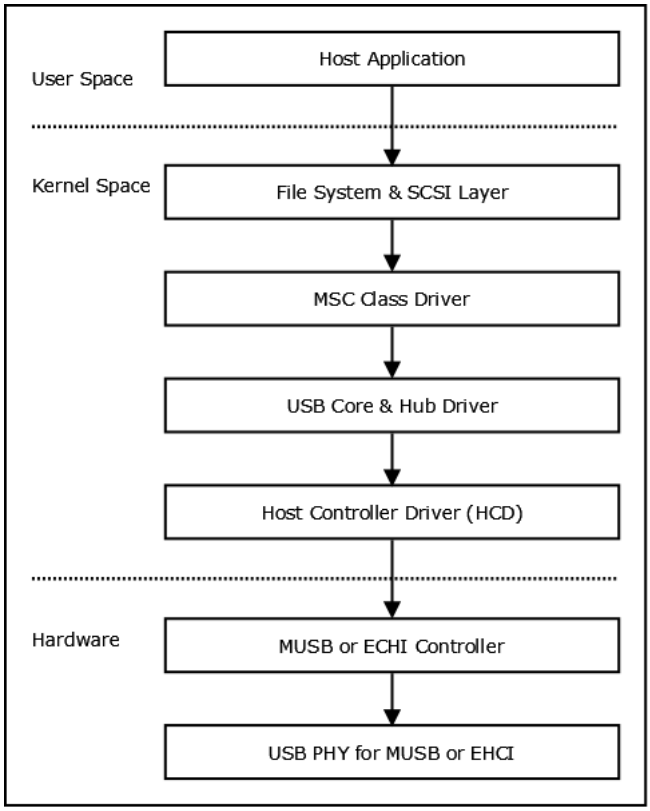
Select Palmas phy driver

```
Device Drivers --->
USB support --->
  <*>   Support for Host-side USB
  [ ]     USB verbose debug messages
  [*]     USB announce new devices
  *** Miscellaneous USB options ***
  ...
  -*-   NOP USB Transceiver Driver
  <*>   Palmas USB Transceiver Driver
```

Host mode applications

Mass Storage Driver

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



USB Controller and USB MSC HOST

Configuration

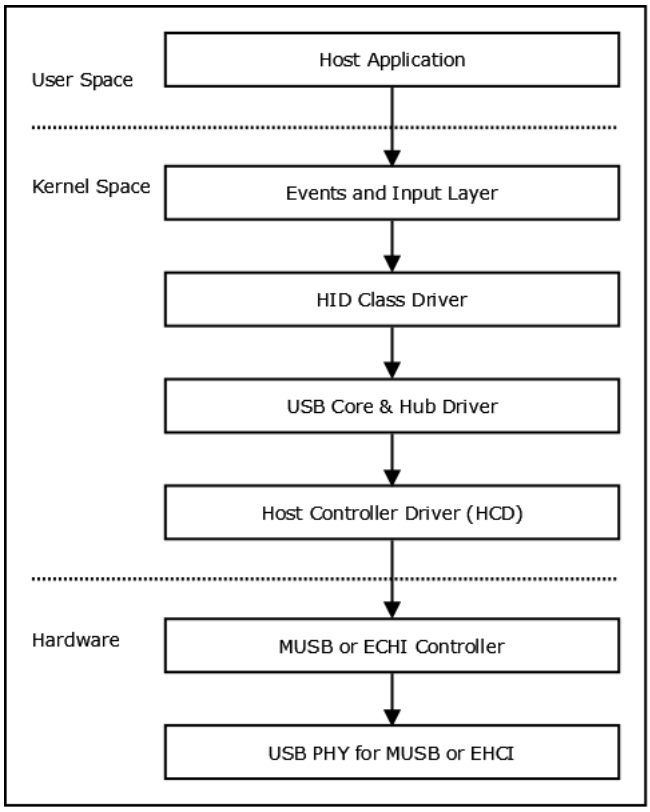
```
   Device Drivers --->
   SCSI device support --->
    <*> SCSI device support
    [*] legacy /proc/scsi/support
    --- SCSI support type (disk, tape, CD-ROM)
    <*> SCSI disk support
   USB support --->
    <*> Support for Host-side USB
    [...]
     --- USB Device Class drivers
     <*> USB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



USB Controller and USB HID

Configuration

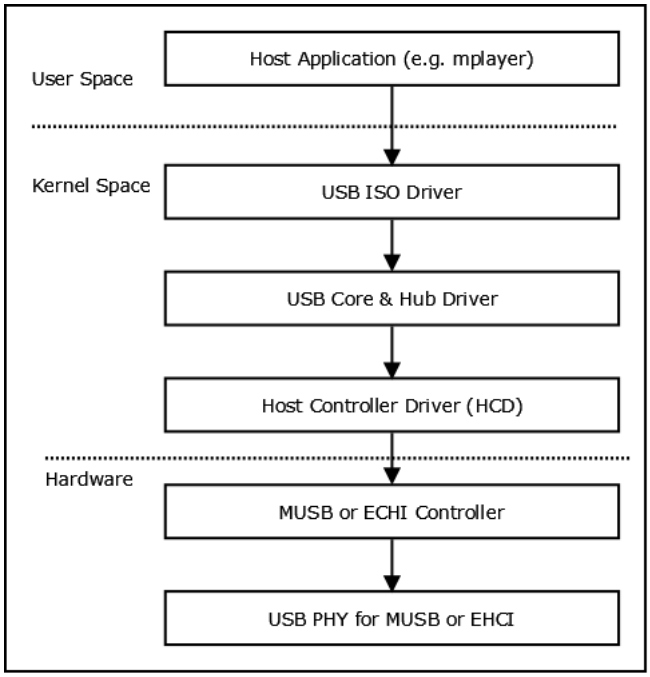
```
Device Drivers --->
   HID drivers  --->
      -*- HID bus support
      [ ] Battery level reporting for HID devices
      [ ] /dev/hidraw raw HID device support
      < > User-space I/O driver support for HID subsystem
      <*> USB HID support  --->
          Special HID drivers  --->
          I2C 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 File:Evtest.zip 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
   [*] Preclaim OSS device numbers
   <*> Advanced Linux Sound Architecture  --->
       ...
       [*] USB sound devices  --->
          <M>   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 support ***
      [ ] Enable old-style fixed minor ranges on drivers/video devices
          *** Media drivers ***
      [*] Media USB Adapters  --->
          <M> USB Video Class (UVC)
          [*] UVC input events device support
```

```
          <M>   GSPCA based webcams  --->
      [*]   V4L platform devices  --->
```

Resources

For testing USB Video support we need a user level application like "mplayer" or "capture" to stream video from an USB camera.

You can use the capture tool to stream video from USB camera.

```
$ ./capture -d /dev/video0 -i z1.yuv -w 320 -l 240
```

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 for specific controller 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
```

```
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
```

Dual Role Device (DRD) support

DRD is Dual Role Device, where usb host controller (dwc3) can be configured to "host-mode" or "device-mode" at instance of time, by any one of the following methods

1. **The role switching to "device" or "host" can done by writing to dwc3 debugfs nodes**
2. **The dynamic role switching to "host" or "device" mode based on usb-id pin when the**

usb cable (type-A for host, type-B for device) is connected to usb1 or usb2 ports.

DRD Setup

In order to DRD to work both host stack and device stack along with gadget modules must be initialized. After kernel boot, by default the DRD port is configured to host mode. To initialize the gadget stack do the following procedure.

```
Note: only the DRD capable port can be switched to either "host" or "device" modes.
This operation is not required for host-only or peripheral-only ports.
```

1. After kernel is booted, mount debugfs and switch DRD port to device mode.

```
mount -t debugfs debugfs /mnt
```

The two nodes /mnt/4889000.dwc3 and /mnt/488d000.dwc3 for usb1 & usb2 port are created respectively.

2.switch the DRD ports (usb1 or usb2) to "device" mode

```
echo "device" > /mnt/48890000.dwc3/mode.
echo "device" > /mnt/488d0000.dwc3/mode.
```

3. Insert all the gadget module.

```
insmod <libcomposite.ko>
insmod <gadget.ko> for DRD port1 or port2
```

```
Note: If both ports are configured for DRD mode, the two gadgets must be inserted
for each port1 and port2 respectively.
```

4.Now ready to switch b/w host/device mode based on cable connect.

5.If host cable (mini/micro A-Receptacle, ID pin Ground) connected to DRD capable portX,the port should switch to host mode. Based on ID pin (in this case ID is Ground), port switches to host mode, as part of this if previous mode is device, the device stack is removed and host stack get initialized.

6.If device cable (mini/micro B-cable, ID pin Float) connected to DRD capable portX, the port should switch to device mode. Based on ID pin (In this case ID is Open), port switches to device mode, as part of this if previous mode is host, the host stack is removed and device stack get initialized.

DRD Testing on TI-dra7xx EVM (Standalone alone testing with onboard usb ports usb1 and usb2)

Build kernel and usb gadget modules

1. Build the kernel default omap2plus_defconfig

2. make sure the dr_mode is set to "otg" for usb1/2 in platform specific .dtsi specific files.

```
dr_mode = "otg";
```

```
For dra7xx, set dr_mode to "otg" in arch/arm/boot/dts/dra7.dtsi and arch/arm/boot/dts/dra7-evm.dts files.
For omap5, set dr_mode to "otg" in arch/arm/boot/dts/omap5.dtsi file.
```

3. Build the kernel and usb gadgets as modules (for example g_zero.ko , g_mass_storage.ko etc).

5. create the sd card with file system and boot partition, refer to GLSDK software developers guide for more info.

6. copy arch/arm/boot/uImage and arch/arm/boot/dts/dra7-evm.dtb to /media/boot partition of sd card.

7. install the modules to sdcard rootfs mount directory.

```
#make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=/media/rootfs
```

Setup/Testing DRD feature

1. The DRA7XX-EVM has usb1 (usb3.0 compatible) and usb2(usb2.0 only) ports, DRD can be tested by connecting usb1 and usb2 ports back to back with host-A type device-B type usb cables.

2. Remove any usb cable/adapter connected to usb1 or usb2 port. Boot the kernel. By default both usb1 and usb2 ports are set to host mode. In host mode you cannot insert the gadget modules. Hence configure usb1 and usb2 ports to device mode.

2.1 mount debugfs node

```
# mount -t debugfs debugfs /mnt
```

2.2 swtich to device mode for usb1 port

```
# echo "device" > /mnt/48890000-dwc3/mode
```

2.3 swtich to device mode for usb2 port

```
# echo "device" > /mnt/488d0000-dwc3/mode
```

3. Note: To switch b/w "host" and "device" mode manually you can use

To swtich to host mode for usb1 port

```
# echo "host" > /mnt/48890000-dwc3/mode
```

To swtich to device mode for usb1 port

```
# echo "device" > /mnt/48890000-dwc3/mode
```

3B) To swtich to host mode for usb2 port

```
# echo "host" > /mnt/4888d0000-dwc3/mode
```

To swtich to device mode for usb1 port

```
# echo "device" > /mnt/488d000-dwc3/mode
```

4.Load the usb gadget modules for usb1 port and usb2 port respectively.

```
 # insmod /lib/modules/3.8.13-xxx/kernel/drivers/usb/gadget/libcomposite.ko
```

```
# insmod /lib/modules/3.8.13-xxx/kernel/drivers/usb/gadget/g_zero.ko (for usb1 port)
```

```
# insmod /lib/modules/3.8.13-xxx/kernel/drivers/usb/gadget/g_mass_storage.ko file=/dev/ram0 stall=0 (for usb2 port)
```

The g_zero gadget is loaded to usb1, and g_mass_storage to usb2.

5.To view the current mode of usb1 or usb2

```
# cat /mnt/48890000-dwc3/mode
```

```
# cat /mnt/488d0000-dwc3/mode
```

6.Now connect the type-A cable (micro-A receptacle) to usb1 port, the dyanmic usb-id pin is detected and usb1 port is configured to host mode automatically. connect usb2 port to usb1 port through type-B (mini-B type) cable. usb1 host will enumerate usb2-device(g_mass_storage.ko).

8.Reverse the cable, connect type-A receptacle to usb2 port (host-mode) and type-B cable to usb1 port (device mode). Now the usb2 becomes host dynamically and enumerate the usb1(device) g_zero gadget.

Note: The role switch can happen only if the current mode is different from previous mode. make sure build the kernel and device tree blob and load appropriate dtb file (dra7-evm.dtb or omap5-uevm.dtb)

DRD Testing for TI-OMAP5-UEVM

1. Follow the same procedure explained above for dra7xx EVM. The OMAP5 has only one DRD port, hence debug entry created for one drd port.

2. The debugfs node for usb1 port is "/<mount-point>/4a030000-dwc3/mode"

kernel 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 debugfs <mount-dir>
```

dwc3 driver TEST-MODE debugfs support

Issue the following command

To send test packet

```
    #echo "test_packet" > /<mount-dir>/<controller-node>/testmode
```

To generate test K pattern

```
    #echo "test_k" > /<mount-dir>/<controller-node>/testmode
```

To generate test J pattern

```
    #echo "test_j" > /<mount-dir>/<controller-node>/testmode
```

To generate test SE0 NAK pattern

```
    #echo "test_se0_nak" > /<mount-dir>/<controller-node>/testmode
```

- **The <controller-node> is the specific instance of dwc3 contoller.**

For example: on dra7xx, the controller-node would be 48890000.dwc3 for usb1 and 488d0000.dwc3 for usb2, etc.

{{

1. switchcategory:MultiCore=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **USB General Guide Linux v3.8** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **USB General Guide Linux v3.8** here.

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article **USB General Guide Linux v3.8** here.*

DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article **USB General Guide Linux v3.8** here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article **USB General Guide Linux v3.8** here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **USB General Guide Linux v3.8** here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **USB General Guide Linux v3.8** here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article **USB General Guide Linux v3.8** here.*

*For technical s please post yo questions at http://e2e.ti.cor Please post on comments abo article **USB Ge Guide Linux v** here.*

}}

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity