UsbgeneralpageLinuxCore

USB Driver

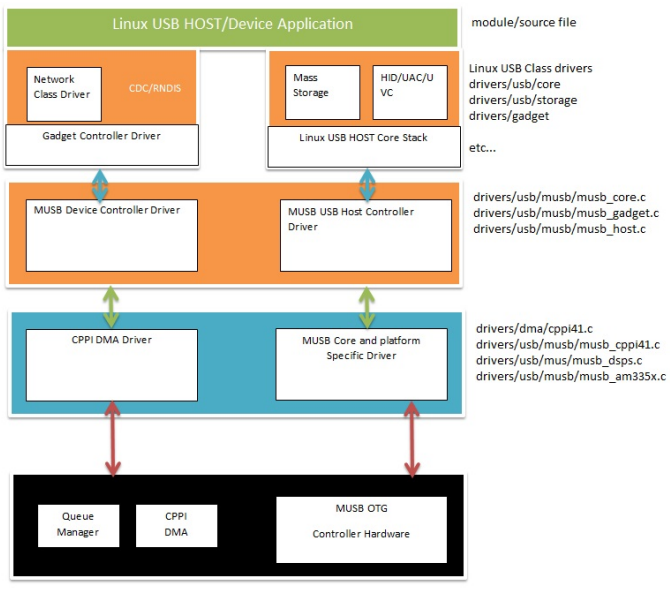
Contents

- Introduction
  - Linux USB Stack Architecture
- Driver configuration
  - To configure the USB driver features through menuconfig
  - USB phy selection for MUSB OTG port
  - Enabling DMA or PIO mode
  - Gadget configuration
    - Available options to support gadget driver
  - Host mode applications
    - Mass Storage Driver
      - USB Controller and USB MSC HOST
      - Configuration
      - Device nodes
    - USB HID Class
      - USB Controller and USB HID
      - Configuration
      - Device nodes
  - Gadget Mode Applications
    - Mass Storage Gadget
      - Configuration
      - Installation of Mass Storage Gadget Driver
    - CDC/RNDIS gadget
      - Configuration for USB controller and CDC/RNDIS Gadget
      - Installation of CDC/RNDIS Gadget Driver
      - Setting up USBNet
- Setup procedure for AM335X
- Software Interface
  - sysfs
  - musb driver debugfs
    - mount the debug file system (debugfs)
    - musb driver TEST-MODE debugfs support
      - To force musb to host mode
      - To force musb to full-speed
      - To force musb to high-speed
      - To send test packet
      - To generate test K pattern
      - To generate test J pattern
      - To generate test SE0 NAK pattern

Introduction

Linux USB Stack Architecture

Linux usb stack is a 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.



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

AM33XX has two musb controller (usb0 and usb1) and each usb controller can either act as host or gadget. OTG mode kernel build requires the gadget driver to be inserted to each musb controller, hence two gadget driver need to be inserted, one for usb0 port and other for usb1 port. This page will provide more detail on available option to support multi instance gadget driver.

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-linux-gnueabihf- menuconfig
```

Goto Menuconfig->Device Drivers

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

USB phy selection for MUSB OTG port

Please select AM335x USB PHY Driver for MUSB support on all platform .

```
-*- NOP USB Transceiver Driver
<*> AM335x USB PHY Driver
< > Samsung USB 2.0 PHY controller Driver
< > Samsung USB 3.0 PHY controller Driver
...
```

Enabling DMA or PIO mode

- To enable musb driver in PIO mode, select

```
MUSB DMA mode (Disable DMA (always use PIO))  --->
```

- To enable musb driver in DMA mode, unselect

```
 MUSB DMA mode (TI CPPI 4.1 (AM335x))  --->
```

Gadget configuration

Available options to support gadget driver

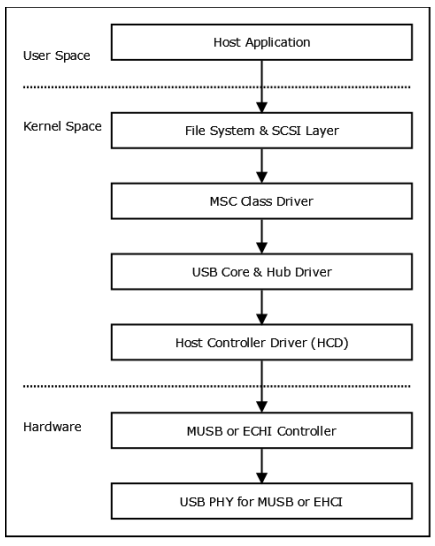
This option can be seen at Drivers->USB Support -> USB Gadget Support as shown below.

```
--- USB Gadget Support
[*]   Debugging messages (DEVELOPMENT)
[ ]     Verbose debugging Messages (DEVELOPMENT)
[*]   Debugging information files (DEVELOPMENT)
[*]   Debugging information files in debugfs (DEVELOPMENT)
(2)   Maximum VBUS Power usage (2-500 mA)
(2)   Number of storage pipeline buffers
      USB Peripheral Controller  --->
<M>   USB Gadget Drivers
< >     USB functions configurable through configfs
<M>     Gadget Zero (DEVELOPMENT)
<M>     Audio Gadget
[ ]       UAC 1.0 (Legacy)
<M>     Ethernet Gadget (with CDC Ethernet support)
[*]       RNDIS support
[ ]       Ethernet Emulation Model (EEM) support
<M>     Network Control Model (NCM) support
<M>     Gadget Filesystem
....
```

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

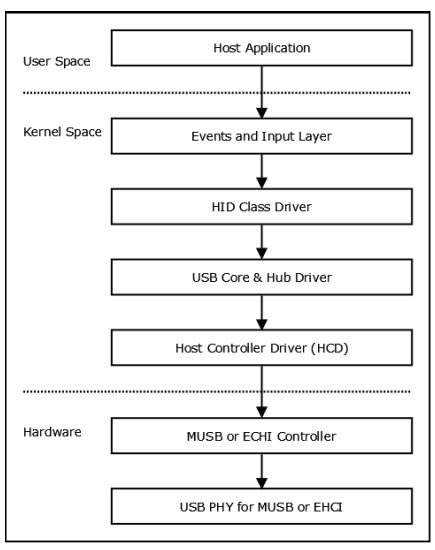
```
  Device Drivers --->
 SCSI device support --->
  <*> SCSI device support
  [*] legacy /proc/scsi/support
  --- SCSI support type (disk, tape, CD-ROM)
  <*> SCSI disk support
 USB support --->
  <*> Support for Host-side USB
  [...]
   --- USB Device Class drivers
   <*> USB Mass Storage support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

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

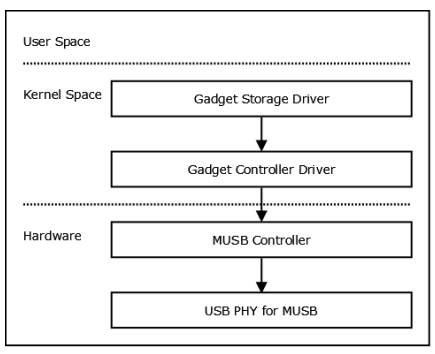
```
  Device Drivers --->
 HID Devices --->
  <*> Generic HID Support
            *** USB Input Devices ***
  <*> USB Human Interface Device(full HID) support
```

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 File:Evtest.zip application. Usage: ./evtest /dev/input/event*

Gadget Mode Applications

Mass Storage Gadget



Configuration

```
 Device Drivers --->
USB support --->
 <*> Support for USB Gadgets
 USB Peripheral Controller (Inventra HDRC Peripheral(TI, ...)) --->
 ...
 <M> USB Gadget Drivers
 <M> Mass Storage Gadget
```

Installation of Mass Storage Gadget Driver

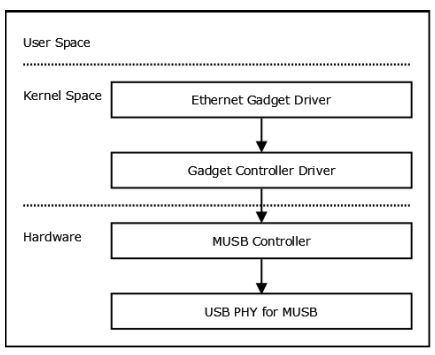
Using Mass storage gadget, you can expose the storage media like MMC (/dev/mmcblk0), usb HDD (/dev/sda, etc) as removable media to standard windows/Linux host. To load the mass storage gadget driver, use the below command

```
 example
#insmod <g_mass_storage.ko> file=/dev/sda1
#insmod <g_mass_storage.ko> file=/dev/mmcblk0
```

CDC/RNDIS gadget

The CDC/RNDIS gadget driver that is used to send standard Ethernet frames using USB.

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



Configuration for USB controller and CDC/RNDIS Gadget

```
Device Drivers --->
USB support --->
<*> Support for USB Gadgets
USB Peripheral Controller (Inventra HDRC Peripheral (TI, ...)) --->
<M> USB Gadget Drivers
<M> Ethernet Gadget
(*)   RNDIS support (EXPERIMENTAL) (NEW)
```

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

```
USB Peripheral Controller (Inventra HDRC Peripheral (TI, ...)) --->
<M> USB Gadget Drivers
<M> Ethernet Gadget
[ ]   RNDIS support (EXPERIMENTAL) (NEW)
```

Installation of CDC/RNDIS Gadget Driver

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

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

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
```

Setup procedure for AM335X

1) Build uImage and usb gadget modules

Use the default omap2plus_defconfig and build the kernel uImage and gadget drivers as modules (like g_ether.ko, g_mass_storage.ko ..etc). omap2plus_defconfig has all gadget as module .

2) chosing right usb connector/cables

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

- To use usb0/usb1 in host mode, connect usb device through a mini/micro-A plug to standard-A receptacle cable.
- To use usb0/usb1 in device mode, connect the board to external host using mini/micro-B plug to standard-A plug cable.

If the board has standard-A receptacle

- To use usb0/usb1 in host mode , connect devices directly or through HUB.
- To use usb0/usb1 in device mode , connect the board to external host using standard-A plug to standard-A plug cable.

3) Insert the gadget modules

Load the kernel image and make sure above setup is done before insert the modules. Insert the gadget modules .

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

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 then urb submission is disabled.

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, in which '***X***' is '0' or '1' for USB0 or USB1 port respectively. The entry "musb-hdrc.***X***" in the commands below might be "musb-hdrc.***X***.auto" in some older versions of Linux kernel.

CAUTION: The following command should only be run **once** until USB reset. Otherwise, the result is undefined. To stop the test or switch to a different test, a system reboot (which is the easy way to generate USB reset) is required.

To force musb to host mode

```
    #echo "force host" > /sys/kernel/debug/musb-hdrc.X/testmode
```

To force musb to full-speed

```
    #echo "force full-speed" > /sys/kernel/debug/musb-hdrc.X/testmode
```

To force musb to high-speed

```
    #echo "force high-speed" > /sys/kernel/debug/musb-hdrc.X/testmode
```

To send test packet

```
    #echo "test packet" > /sys/kernel/debug/musb-hdrc.X/testmode
```

To generate test K pattern

```
    #echo "test K" > /sys/kernel/debug/musb-hdrc.X/testmode
```

To generate test J pattern

```
    #echo "test K" > /sys/kernel/debug/musb-hdrc.X/testmode
```

To generate test SE0 NAK pattern

```
  #echo "test SE0 NAK" > /sys/kernel/debug/musb-hdrc.X/testmode
```


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