

# UsbgeneralpageLinux-v3p1

---

## USB Driver

## Contents

---

### Introduction

#### Linux USB Stack Architecture

### Driver configuration

#### To configure the USB driver features through menuconfig

##### AM33XX EVM default configuration

#### USB phy selection for MUSB OTG port

#### Enabling DMA or PIO mode

#### MUSB OTG gadget configuration

##### Available options to support multi instance gadget driver

##### Selecting individual 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

### Modular testing on MUSB

### Setup procedure for AM335X

### Device removal from and reconnect to MUSB OTG port

### 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

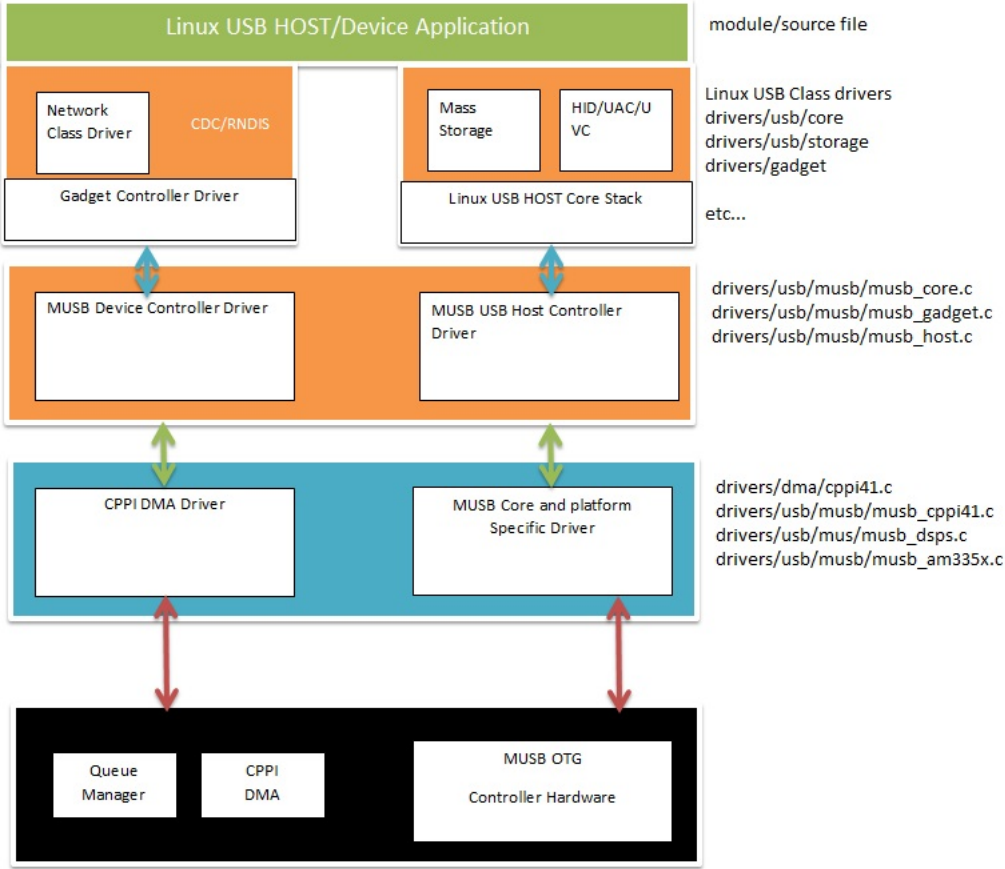
##### To generate test SE0 NAK pattern

## 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

The default musb driver mode in linux v3.1 kernel has been changed to OTG mode which is the only option available. AM33XX has two musb controller (usbo 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 usbo 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-arago-linux-gnueabi- menuconfig
```

Goto Menuconfig->Device Drivers

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

### AM33XX EVM default configuration

Use below command to use default kernel config for AM335X

```
# make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- am335x_evm_defconfig
```

Note: The arm tool chain used here is for example purpose only.

## USB phy selection for MUSB OTG port

Please select NOP USB transceiver for MUSB support on all platform .

```
Device Drivers --->
USB support --->
*** OTG and related infrastructure ***
[ ] GPIO based peripheral-only VBUS sensing 'transceiver'
[ ] Generic ULPI Transceiver Driver
```

[ ] TWL4030 USB Transceiver Driver

-\*- NOP USB Transceiver Driver

## Enabling DMA or PIO mode

- 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)

## MUSB OTG gadget configuration

### Available options to support multi instance gadget driver

In the current release, gadget driver can be configured as

- One gadget as builtin and other as module\*** - This is the default setting in am335x\_evm\_defconfig on current release and is suitable for a board where one port is hardwired to be host only. The builtin gadget would make sure that the host only port works soon after bootup. Second gadget driver module can be inserted after bootup.

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

--- USB Gadget Support

[ ] 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 (Inventra HDRC USB Peripheral (TI, ADI, ...)) --->

<\*> Select one gadget as builtin for one port

Select USB port to bind builtin gadget (USB-1) --->

<M> USB Gadget Drivers

<M>     Ethernet Gadget (with CDC Ethernet support)

[\*]     RNDIS support

<M>     Mass Storage Gadget

....

- Both gadget driver as modules** - Change the kernel config as shown below for this. Please note that the musb port will be in usable form only after a gadget module is inserted.

--- USB Gadget Support

[ ] 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 (Inventra HDRC USB Peripheral (TI, ADI, ...)) --->

<> Select one gadget as builtin for one port

<M> USB Gadget Drivers

<M>     Ethernet Gadget (with CDC Ethernet support)

[\*]     RNDIS support

<M>     Mass Storage Gadget

### Selecting individual gadget driver

Select the Inventra HDRC in "USB Gadget support page" as show below.

Device Drivers --->

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)

(2) Number of storage pipeline buffers

<>    Select one gadget as builtin for one port

USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...)) --->

<M> USB Gadget Drivers (Ethernet Gadget (with CDC Ethernet support)) --->

For Gadget RNDIS Mode configuration

Device Drivers --->

USB support --->

<\*> USB Gadget Support --->

--- USB Gadget Support

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) (NEW)

[\*] RNDIS support (NEW)

For Gadget MassStorage configuration

Device Drivers --->

USB support --->

<\*> USB Gadget Support --->

--- USB Gadget Support

USB Peripheral Controller (Inventra HDRC USB Peripheral (TI, ADI, ...)) --->

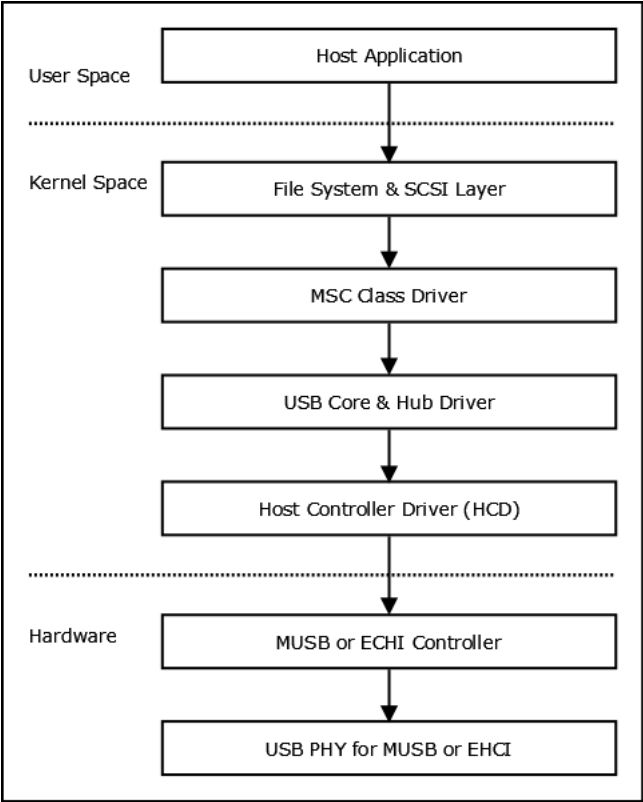
<M> USB Gadget Drivers

<M> Mass Storage Gadget

# 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

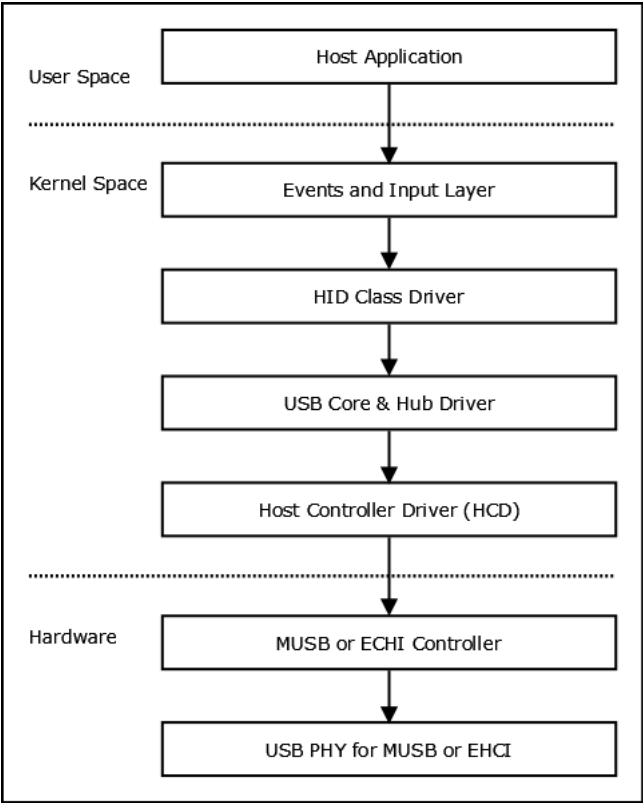
```
Device Drivers --->
SCSI device support --->
<*> SCSI device support
[*] legacy /proc/scsi/support
--- SCSI support type (disk, tape, CD-ROM)
<*> SCSI disk support
USB support --->
<*> Support for Host-side USB
[...]
```

### 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

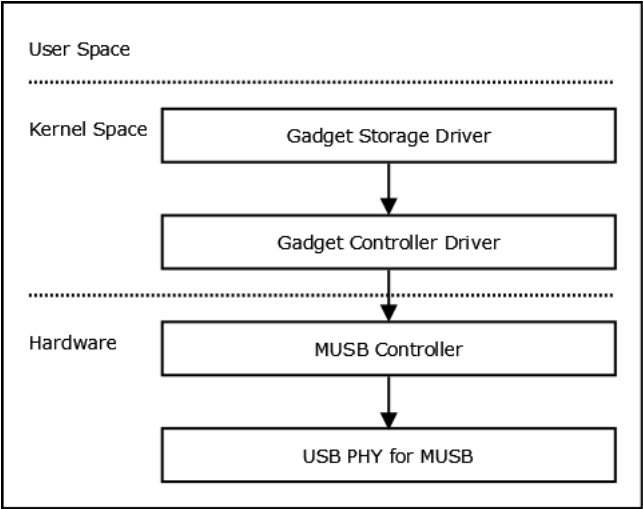
```
Device Drivers --->
HID Devices --->
<*> Generic HID Support
*** USB Input Devices ***
<*> USB Human Interface Device(full HID) support
```

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

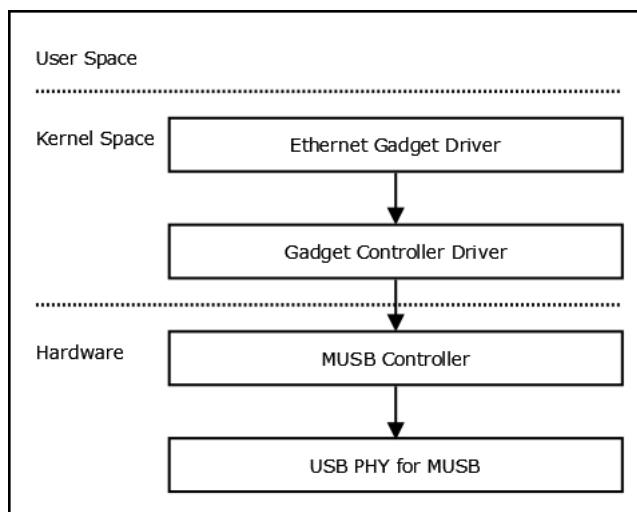
Using Mass storage gadget, you can expose the storage media like MMC (/dev/mmcblk0), usb HDD (/dev/sda, etc) as removable media to standard windows/Linux host. To load the mass storage gadget driver, use the below command

```
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
```

## Modular testing on MUSB

Mentor USB (MUSB) linux driver has been reorganized in v2.6.37 onwards 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 and 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.
```

# Setup procedure for AM335X

## 1) Build uImage and usb gadget modules

Use the default am335x\_evm\_defconfig and build the kernel uImage and gadget drivers as modules (like g\_ether.ko, g\_mass\_storage.ko ..etc). am335x\_evm\_defconfig has one gadget as builin for one usb port so insert the gadget driver module for other usb port.

## 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 for another usb port.

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

# Device removal from and reconnect to MUSB OTG port

An OTG port is expected to work both in host and gadget mode. So whenever device/HUB is disconnected from root port, the VBUS is switched OFF, so that when the same port is connected to other host, it should be able to work in device mode. Hence while removing the device/HUB from root port and after re-inserting any device/HUB to the OTG port, you should start the VBUS using the below command

For porto

```
#echo F > /proc/driver/musb_hdrc.0
```

For port1

```
#echo F > /proc/driver/musb_hdrc.1
```

# 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

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 K" > /sys/kernel/debug/testmode
```

To generate test SE0 NAK pattern

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

Keystone=

{{

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 [UsbgeneralpageLinux-v3p1](#) here.

- 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 [UsbgeneralpageLinux-v3p1](#) here.

C2000=For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article [UsbgeneralpageLinux-v3p1](#) here.

DaVinci=For technical support on DaVinciplease post your questions on The DaVinci Forum. Please post only comments about the article [UsbgeneralpageLinux-v3p1](#) here.

MSP430=For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article [UsbgeneralpageLinux-v3p1](#) here.

OMAP35x=For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article [UsbgeneralpageLinux-v3p1](#) here.



[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](#)



**This page was last edited on 1 June 2012, at 01:06.**

Content is available under [Creative Commons Attribution-ShareAlike](#) unless otherwise noted.