Application Report
SPRACB5–September 2017

USB Composite Gadget Using CONFIG-FS on DRA7xx Devices

Ravi B

ABSTRACT

This application note explains how to create a USB composite gadget, network control model (NCM) and abstract control model (ACM) from the user space using Linux® CONFIG-FS on the DRA7xx platform.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The scope of this document is to provide guidelines to configure the USB composite gadget using CONFIG-FS.

1.1 Software Requirements

This application note is based on the Processor SDK Linux Automotive (PSDKLA) 3.02, and uses the U-Boot version 2016.05 as a reference. This application note requires that users:

- Install Processor SDK Linux Automotive 3.02
- Can build U-Boot and kernel

Table 1 lists the kernel and U-Boot commits corresponding to the 3.02 SDK.

Table 1. Kernal and U-Boot Commits

Repository	Commit ID	Headline
Kernel	89944627d53a	Late attach: Fix for accessing second level page table
U-Boot	850ffc07orba	defconfigs: dra7xx_hs_evm: Move OPTEE load address to avoid overlaps

For more information, see the release download links and software developer's guide.

1.2 Hardware Requirements

The Dra75x/J6 EVM Rev-H evaluation board is used for reference.

1.3 Build Instructions

1. Download the PSDKLA-3.02 release
2. Follow the instructions in the Processor SDK Linux Automotive Software Developer's Guide to build the U-Boot, kernel, DTB, and modules.

2 USB Composite Gadget Using CONFIG-FS

A USB composite gadget is a USB device framework which combines more than one USB-class device function. Figure 1 shows the USB composite gadget, which consists of a single control pipe (endpoint-0) and a set of configurations. Each configuration contains a set of interfaces for a USB-specific class (such as serial or mass storage, or CDC and Ethernet). Each interface contains a set of endpoints (IN-receive or OUT-transmit) on which data transfers to and from, between the device function and USB host.

For example, the composite gadget includes a mass storage device (for any storage media), CDC and RNDIS network interfaces, and so on.

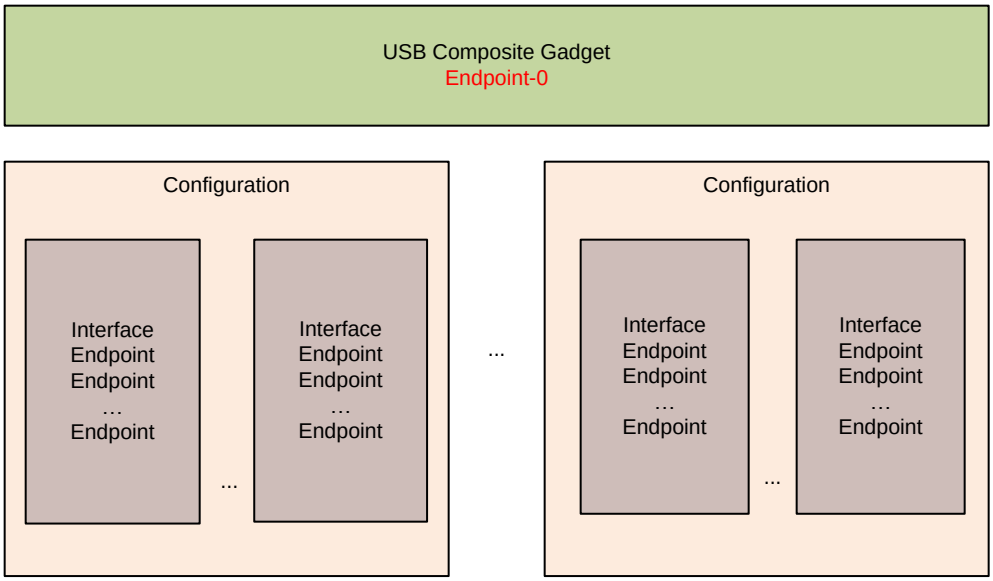

Figure 1. Block Diagram of USB Composite Gadget

The USB composite gadget framework in Linux kernel lets users combine one or more gadget functions, and exposes as a single, composite, gadget function to the USB host.

The Linux kernel supports creation of the USB composite gadget in the user space through CONFIG_FS support, where the standard kernel gadget functions are exposed through CONFIGFS as config items and groups, and both are represented as directories. Both items and groups can have attributes which are represented as directories or files. The user can create and remove directories, which can be read-only or read-write depending on what they represent.

For more information, visit gadget configfs and configfs.

3 Creating Composite Gadget From User Space

3.1 Building the Kernel

The Linux kernel supports CONFIGFS, which lets the user create the composite gadget from the user space.

1. From menuconfig of the Linux kernel, select CONFIG_CONFIGFS_FS.

   CONFIG_CONFIGFS_FS is selected through Menuconfig → FileSystem → Psuedo File System → {M} Userspace-driven configuration filesystem (see Figure 2).

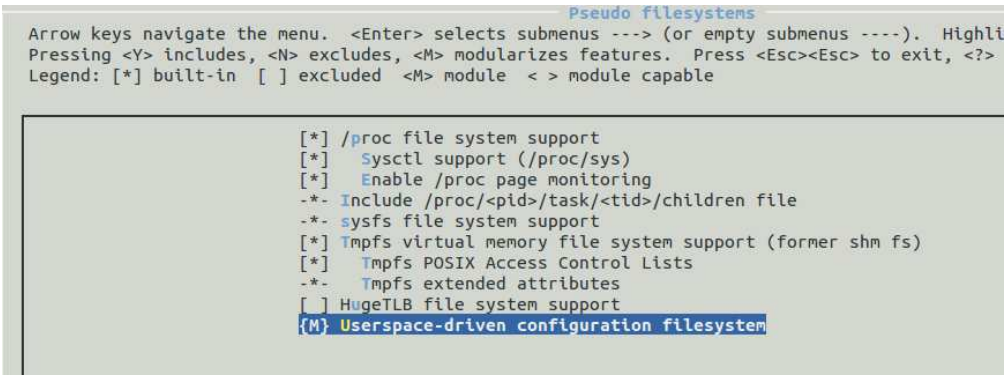

Figure 2. Selection of CONFIGFS Through menuconfig

2. Select the required gadget module, in this example the NCM and serial ACM gadget modules are selected, as shown through menuconfig (see Figure 3).

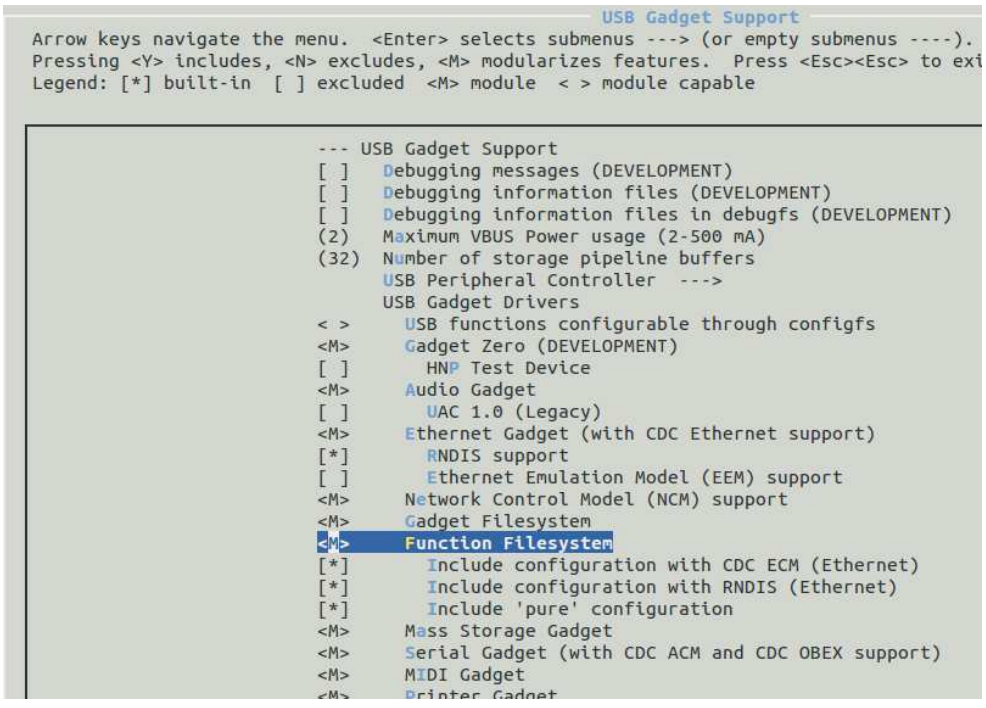

Figure 3. Select USB Configuration Through menuconfig

3. Build the kernel, DTB, and all gadget modules. You can also use the default PSDKLA-3.02 prebuilt binaries.

For more information, see the PSDKLA user's guide for building the kernel, DTB, and gadget modules.

3.2 USB Composite Gadget Through CONFIGFS

The following example shows how to create the USB composite gadget from the user space, which includes two USB device functions:

- USB NCM gadget
- USB ACM gadget

3.2.1 Creating NCM and ACM Composite Gadgets

To set up NCM and ACM composite gadgets:

1. Connect the USB0 (super-speed) port of the EVM to the Ubuntu® PC through the USB device cable.
2. Build the kernel, as explained in Section 3.1.
3. Copy the kernel, DTB, and install modules to the SD card.

You can also use PSDKLA-3.02 prebuilt binaries.

Next, boot the kernel to root prompt, and follow these steps:

1. Insert gadget modules.
```
# modprobe libcomposite
```
2. Mount the configfs file system, if not mounted already.
```
# mount -t configfs none /sys/kernel/config
```
3. Switch to device mode. By default, in the PSDKLA-3.02 release the USB0 port is configured in DRD mode, therefore configure the USB0 port in device mode.
```
# mount -t debugfs debugfs /mnt
# echo "device" > /mnt/48890000.usb/mode
```
4. Create the composite gadget.
```
# cd /sys/kernel/config/
# cd usb_gadget/
# mkdir j6g
# cd j6g
```
Figure 4 shows the files and directories that are automatically created.

```
root@dra7xx-evm:/sys/kernel/config/usb_gadget/j6g# ls -l
-rw-r--r--    1 root     root          4096 Feb 28 11:08 UDC
-rw-r--r--    1 root     root          4096 Feb 28 11:08 bDeviceClass
-rw-r--r--    1 root     root          4096 Feb 28 11:08 bDeviceProtocol
-rw-r--r--    1 root     root          4096 Feb 28 11:08 bDeviceSubClass
-rw-r--r--    1 root     root          4096 Feb 28 11:08 bMaxPacketSize0
-rw-r--r--    1 root     root          4096 Feb 28 11:08 bcdDevice
-rw-r--r--    1 root     root          4096 Feb 28 11:08 bcdUSB
drwxr-xr-x    2 root     root             0 Feb 28 11:08 configs
drwxr-xr-x    2 root     root             0 Feb 28 11:08 functions
-rw-r--r--    1 root     root          4096 Feb 28 11:08 idProduct
-rw-r--r--    1 root     root          4096 Feb 28 11:08 idVendor
drwxr-xr-x    2 root     root             0 Feb 28 11:08 os_desc
drwxr-xr-x    2 root     root             0 Feb 28 11:08 strings
```

Figure 4. Composite Gadget Configuration Items as Files and Directories

5. Update the VID and PID and strings (see Figure 5).
```
# echo "0xA55A" > idVendor
# echo "0x0111" > idProduct
# mkdir strings/0x409
# cd strings/
# cd 0x409/
```

```
root@dra7xx-evm:/sys/kernel/config/usb_gadget/j6g/strings/0x409# ls -l
-rw-r--r--    1 root     root          4096 Feb 28 11:13 manufacturer
-rw-r--r--    1 root     root          4096 Feb 28 11:13 product
-rw-r--r--    1 root     root          4096 Feb 28 11:13 serialnumber
```

Figure 5. VID, PID, and Manufacturer String Configuration

```
# echo "0123456789" > serialnumber
# echo "Xyz Inc." > manufacturer
# echo "NCM+ACM gadget" > product
# cd ../..
```

6. Create the USB device functions (NCM and ACM).

```
cd functions/
mkdir acm.gs0
mkdir ncm.usb0
cd ..
```

7. Create the configuration and update the strings.

```
cd configs/
mkdir c.1
cd c.1
mkdir strings/0x409
cd strings/0x409/
echo "ACM+NCM" > configuration
cd ../../../..
```

8. Create symbolic links.

```
ln -s functions/acm.gs0 configs/c.1
ln -s functions/ncm.usb0 configs/c.1
```

9. Attach the USB0 port to UDC.

```
# echo "48890000.usb" > UDC
```

Issue the previously mentioned command, and the DRA7xx EVM (USB composite gadget NCM and ACM) is connected to the Ubuntu host. The host enumerates the USB composite gadget (ACM and NCM), and the USB0 interface and serial device /dev/ttyGS0 are created, as shown in Figure 6. The /dev/ttyACM0 interface is created on the Ubuntu host.

```
root@dra7xx-evm:/sys/kernel/config/usb_gadget/j6g# echo "48890000.usb" > UDC
[  858.846732] usb0: HOST MAC 6e:65:4f:e8:af:c6
[  858.851127] usb0: MAC 8e:ea:ea:13:a1:09
[  858.855008] dwc3 48890000.usb: otg: gadget gadget registered
[  858.868486] IPv6: ADDRCONF(NETDEV_UP): usb0: link is not ready
root@dra7xx-evm:/sys/kernel/config/usb_gadget/j6g# [  859.289071] configfs-gadget gadget: high-speed config #1: c
[  859.296926] IPv6: ADDRCONF(NETDEV_CHANGE): usb0: link becomes ready

root@dra7xx-evm:/sys/kernel/config/usb_gadget/j6g# ifconfig usb0
usb0      Link encap:Ethernet  HWaddr 8E:EA:EA:13:A1:09
          inet6 addr: fe80::8cea:eaff:fe13:a109%3068527384/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27 errors:0 dropped:0 overruns:0 frame:0
          TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6307 (6.1 KiB)  TX bytes:6392 (6.2 KiB)

root@dra7xx-evm:/sys/kernel/config/usb_gadget/j6g# ls -l /dev/ttyGS0
crw-rw----    1 root     dialout   242,   0 Feb 28 11:15 /dev/ttyGS0
```

Figure 6. Kernel Logs Show Enumeration of USB Composite Gadget by Host

3.2.2 Verifying NCM Composite Gadget

Verify the NCM gadget interface through ping between the EVM and the Ubuntu host.

1. From the EVM, bring up the USB0 interface.
   ```
   # ifconfig usb0 192.168.100.10 up
   ```
2. From the Ubuntu host, bring up the USB0 interface.
   ```
   # ifconfig usb0 192.168.100.5 up
   ```
3. Ping from both the EVM and the Ubuntu host. Figure 7 shows the ping from the EVM.

```
root@dra7xx-evm:/sys/kernel/config/usb_gadget/j6g# ping 192.168.100.5 -s 65000
PING 192.168.100.5 (192.168.100.5): 65000 data bytes
65008 bytes from 192.168.100.5: seq=0 ttl=64 time=4.226 ms
65008 bytes from 192.168.100.5: seq=1 ttl=64 time=4.316 ms
65008 bytes from 192.168.100.5: seq=2 ttl=64 time=4.406 ms
65008 bytes from 192.168.100.5: seq=3 ttl=64 time=4.551 ms
65008 bytes from 192.168.100.5: seq=4 ttl=64 time=4.371 ms
```

Figure 7. Ping From EVM and Host PC

3.2.3 Running iperf (IPV6) on NCM Interface

Reboot the EVM and follow the steps in Section 3.2.1 to create the USB composite gadget. Do the following to run iperf:

1. From the EVM, bring up the interface (see Figure 8).
   ```
   # ifconfig usb0 up
   ```

```
root@dra7xx-evm:~# ifconfig usb0
usb0      Link encap:Ethernet  HWaddr 06:FE:5A:37:5E:BB
          inet6 addr: fe80::4fe:5aff:fe37:5ebb%3068707608/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5581 (5.4 KiB)  TX bytes:5556 (5.4 KiB)
```

Figure 8. Enable USB0 Interface

2. From the EVM, run the iperf udp server.
   ```
   # iperf -s -u -V -b
   ```

3. From the Ubuntu host, run the iperf udp client.
   1. Disconnect all interfaces in the Ubuntu host PC before running the iperf client.
   2. Copy the inet6 address from the EVM (red box in Figure 8) and issue the iperf client from the host PC.

```
$ iperf -V -c fe80::4fe:5aff:fe37:5ebb%usb0 -u -b 300m
------------------------------------------------------------
Client connecting to fe80::4fe:5aff:fe37:5ebb%usb0, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local fe80::90c8:43ff:fed8:8d20 port 59866 connected with fe80::4fe:5aff:fe37:5ebb port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   326 MBytes   274 Mbits/sec
[  3] Sent 232681 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec   325 MBytes   272 Mbits/sec   0.037 ms 1111/232680 (0.48%)
[  3]  0.0-10.0 sec  22 datagrams received out-of-order
```

Figure 9. Running iperf From Host PC

The iperf result shows there is 272 Mbps with 0.48% data loss.

3.2.4 Verifying the ACM Serial Gadget Interface

After the steps for creating the USB composite gadget are complete, the NCM and ACM interface is created, as shown in Section 3.2.1.

1. From the EVM, do the terminal settings.
   ```
   # stty -F /dev/ttyGS0 -icanon
   ```
2. Run the linux-serial-test application from the EVM.
   ```
   # ./linux-serial-test -w 10 -a 100 -s -e -p /dev/ttyGS0 -b 9600
   ```
3. Run the linux-serial-test application from the Ubuntu host.
   ```
   # chmod +x /dev/ttyACM0
   # sudo ./linux-serial-test -w 10 -a 100 -s -e -p /dev/ttyGS0 -b 9600
   ```

NOTE: Git clone the linux-serial-test application and compile it for the DRA7xx platform and the Ubuntu host.

4 References

- Texas Instruments, DRA7xx Technical Reference Manual
- Texas Instruments, PSDKLA 3.02 release
- Texas Instruments, Processor SDK LINUX Automotive Software Developer's Guide
- GitHub, Git Source Reference for Linux Serial Test Application
- Linux, configfs - Userspace-driven kernel object configuration
- Linux, Linux USB gadget configured through configfs

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265