USB Driver for MX31

Freescale Semiconductor

Linux BSP

1 Hardware Operation

The hardware operation of USB is detailed in the hardware documentation.

2 Software Operation

The Host and Gadget drivers fit into the existing Linux USB driver framework.

3 Source Code Structure Configuration

The Host driver is implemented in drivers/usb/host/ehci-arc.c. It is not compiled by itself, rather it is #included by ehci-hcd.c, as a « bus glue » component.

The Gadget driver lives in drivers/usb/gadget/arcotg_udc.c, and is implemented as a standard USB udc (USB Device Controller) driver.

For USB-OTG operation, several other files are used. drivers/usb/otg/fsl_otg.c and drivers/usb/otg/otg_fsm.c provide OTG state machine support, and combine to form the isp1504_arc.ko kernel module. This code fits between the USB transceiver and the Host and Gadget drivers. It notices transitions on the ID pin, and switches between host and peripheral mode by enabling and disabling the appropriate drivers.

4 Linux Menu Configuration Options

USB Host and Gadget support are configured in two areas of the kernel configuration tree. I recommend configuring USB drivers as modules since if they're built into the kernel, they'll hang the system at boot time if used with a baseboard prior to Rev.C.

4.1 Host configuration

Enable these options:

Device Drivers -> "USB Support -> USB device filesystem -- allows the lsusb command to work.

Device Drivers -> "USB Support -> EHCI HCD (USB 2.0) support

Device Drivers -> "USB Support -> EHCI HCD (USB 2.0) support -> Support for ARC controller

Device Drivers -> "USB Support -> EHCI HCD (USB 2.0) support -> Support for ARC controller -> Support for Host1 port on ARC controller

Device Drivers -> "USB Support -> EHCI HCD (USB 2.0) support -> Support for ARC controller -> Support for Host2 port on ARC controller for Host2 support (conflicts with NAND and ATA).

Device Drivers -> "USB Support -> EHCI HCD (USB 2.0) support -> Support for ARC controller -> Support for OTG HS Host port on ARC controller

Device Drivers -> "USB Support -> EHCI HCD (USB 2.0) support -> Support for ARC controller -> Root Hub Transaction Translators

You can now choose support for the USB devices you plan to connect to the host port.

4.1.1 For hard disk, flash drive, and other mass storage devices

USB Mass Storage support -- This will also enable SCSI support.

Device Drivers -> SCSI device support

Device Drivers -> SCSI device support -> Probe all LUNs on each SCSI device

Enable support for whatever file systems you may encounter.

Device Drivers -> File systems -> Partition Types -> Advanced partition selection

Device Drivers -> File systems -> Partition Types -> PC BIOS (MSDOS partition tables) support

4.1.2 For HID device support (mice, keyboards)

USB configuration -> USB Human Interface Device (full HID) support

USB configuration -> HID input layer support

USB configuration -> /dev/hiddev raw HID device support

Device Drivers -> Input device support -> Event Interface

For Mouse:
Device Drivers -> Input device support -> Event Interface -> Mouse interface

Device Drivers -> Input device support -> Event Interface -> Provide legacy /dev/psaux device

Device Drivers -> Input device support -> Event Interface -> Mice

Device Drivers -> Input device support -> Event Interface -> PS/2 mouse

For Keyboard
Device Drivers -> Input device support -> Event Interface Keyboards

Device Drivers -> Input device support -> Event Interface AT keyboard support

4.2 Gadget configuration

Device Drivers -> USB Support -> Gadget Support -> USB Gadgets

Device Drivers -> USB Support -> Gadget Support ->USB Peripheral Controller -> ARC USB Device Controller

Device Drivers -> USB Support -> Gadget Support ->Select OTG transceiver: High Speed (Only high speed transceiver is selectable at this time)

To allow for dynamic gadget switching, these need to be configured as modules.

USB networking support.

Enable Device Drivers -> USB Support -> Gadget Support -> Ethernet Gadget (with RNDIS support)

Mass Storage

Device Drivers -> USB Support -> Gadget Support -> File-backed Storage Gadget

4.3 USB OTG

The role of the OTG_HS port (Host or Gadget) depends on kernel configuration. If the kernel is not configured to support USB On-The-Go (USB-OTG), the MX31's role is determined by which driver is loaded. This is the same way that earlier releases behaved. If the kernel is configured to support USB-OTG, and the proper drivers are loaded, then the role of the MX31 is determined by the type of cable plugged into the OTG port. You should start with the kernel configured to support the USB Host and Gadget options you want. (See the previous section for details.) The imx31ads_defconfig default configuration is an excellent place to start. To add USB-OTG support:

- Enable CONFIG_USB_OTG
  Device Drivers -> USB Support -> Gadget Support -> OTG Support

- Disable CONFIG_USB_OTG_WHITELIST
  Device Drivers -> USB Support -> Rely on OTG Targeted Peripherals List

- Enable CONFIG_USB_EHCI_ARC_OTGHS
  Device Drivers -> USB Support -> Support for OTG HS Host port on ARC controller

5 Board Configuration Options

You'll need at least a Rev.C MX31ADS baseboard for USB to work. USB also requires NVCC5 to be 2.7v. If you're not using the MC13783 board, set CPU board jumper JP28 to position 1-2. If you are using the MC13783 board, make sure CPU board JP28 is not populated and set base board jumper JP21 (under the MC13783 board) to position 2-3.

This version of the code supports the Host1, Host2 and the OTG_HS port. The OTG_FS port is not electrically connected on Rev.C boards.

USB Connector	Controller	Speed	Comment
SER_HOST_FS(J5),	Host1	Low Speed and Full Speed	
ULPI_HOST_HS(J4)	Host2	Low, Full and High Speed	
OTG_FS(J2)	OTG	Low Speed and Full Speed	The role of the OTG_HS port (Host or Gadget) is determined
OTG_HS(J1	OTG	Low, Full and High Speed	

There are devices on the MX31ADS which conflict with USB pins which cannot be used at the same time as parts of the USB controller.

- USBH1 conflicts with ATA and SPI1.
- USBH2 conflicts with ATA and NAND flash.

6 Programming Interface

Both libusb and gadgetfs interfaces are supported for interfacing to USB devices from user space.

7. Usage Example

7.1. Host testing

1. Load the host driver: The host driver now accepts an optional port parameter to restrict which ports are used. It's a comma separated list; valid values are "h1", "h2" and "otg". Using the port parameter is useful when the driver is configured to support more ports than are currently usable. For example, if the driver is built with support for the Host1, Host2 and OTG ports, but Host1 and Host2 are unavailable because the ATA driver is being used, then you can specify port=otg to eliminate the error messages you'd ordinarily get about Host1 and Host2 conflicts if you tried to use all the ports.

   ```
   mx31# modprobe ehci-hcd  [port=h1,h2,otg]
   ```

2. Plug in a USB device, e.g. a flash drive, into the SER_HOST_FS (J5) port. You'll see console messages like this:

   ```
   usb 1-1: new full speed USB device using ehci_hcd_h1 and address 2
   usb 1-1: not running at top speed; connect to a high speed hub
   scsi0 : SCSI emulation for USB Mass Storage devices
   Vendor:           Model: USB DISK 25X      Rev: PMAP
   Type:   Direct-Access                      ANSI SCSI revision: 00
   SCSI device sda: 251904 512-byte hdwr sectors (129 MB)
   sda: assuming Write Enabled
   sda: assuming drive cache: write through
   SCSI device sda: 251904 512-byte hdwr sectors (129 MB)
   sda: assuming Write Enabled
   sda: assuming drive cache: write through
   /dev/scsi/host0/bus0/target0/lun0: p1
   Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
   ```

3. You can now use the device. The proper device nodes will be created by udev..

4. Mount the drive, and look at it:

   ```
   mx31# mkdir /mnt/usb
   mx31# mount /dev/sda1 /mnt/usb
   mx31# ls -l /mnt/usb/
   -rwxr--r--    1 root     root         7199 Jul 18  2005 dd-out.txt
   -rwxr--r--    1 root     root        15600 Jul 18  2005 dsmes.out
   -rwxr--r--    1 root     root     57024000 Jan  1  1980 katu50m.yuv
   -rwxr--r--    1 root     root         6382 Nov  3  2005 linux.inf
   -rwxr--r--    1 root     root           36 Jun 30  2005 metrower
   -rwxr--r--    1 root     root        46692 Jan  1  1980 top
   ```

   If you want to use the lsusb command

5. Mount the usbfs filesystem. You can do this manually:

   ```
     mx31# mount -t usbfs none /proc/bus/usb
   ```

   Or you can place this entry in /etc/fstab (default):

```
none    /proc/bus/usb   usbfs   defaults        0       0
```

6. Use lsusb command:

   ```
   mx31# lsusb

   Bus 001 Device 002: ID 0d7d:1900 Phison Electronics Corp.
   Bus 001 Device 001: ID 0000:0000
   ```

7. Be sure to unmount the drive before disconnecting the cable.

   ```
   mx31# umount /dev/sda1
   ```

7.2. Gadget testing

USB Gadget drivers consist of two parts: the low-level hardware driver, which Linux calls the "udc" driver, and a higher-level hardware independent driver that provides the actual gadget functionality.

NOTE: The mx31 can only assume the personality of one kind of gadget at a time, so you should only load one gadget driver at a time.

7.2.1. Mass Storage gadget

The upper-level file-storage gadget driver uses a file or block device to store its data. The name of the backing-store medium is passed to the driver as an argument at module load time. This example uses a ramdisk but you can substitute any block device, like an ATA drive, if you wish.

1. Load the Mass Storage gadget driver:

   ```
   mx31# modprobe g_file_storage file=/dev/ram0
   ```

   Assuming your modules.dep file is current, this will also auto-load the arcotg_udc module, which is the low-level udc driver for mx31. You should see these console messages:

   ```
   ARC USBOTG Device Controller driver version 1 August 2005 init
   ARC USBOTG h/w ID=0x5  revision=0x40
   g_file_storage gadget: File-backed Storage Gadget, version: 20
   October 2004
   g_file_storage gadget: Number of LUNs=1
   g_file_storage gadget-lun0: ro=0, file: /dev/ram0
   arcotg_udc: arc_udc bind to driver g_file_storage
   ```

2. Connect a Mini-B-to-A cable between the OTG_HS(J1) connector and a USB Host machine. The mx31 will appear on the host machine as a new hard drive. You'll need to partition the drive and format a partition before you can write data for the first

time. In order to not lose data, be sure to unmount or otherwise stop access to the mx31 from the host before disconnecting the cable.

3. When you're done being a MS device, you can unload the driver:

   ```
   mx31# rmmod g_file_storage
   ```

   NOTE: If you wish to have a more persistent backing medium, you can use a real disk file. For example, you can use something like this script:

   ```
   #!/bin/sh

   if [ ! -f /tmp/file ] ; then
           dd if=/dev/zero of=/tmp/file bs=1M count=64
   fi

   modprobe g_file_storage.ko   file=/tmp/file
   ```

7.2.2. Networking gadget

1. To test the USB networking gadget, first load the gadget driver:

   ```
   mx31# modprobe g_ether
   usb0: Ethernet Gadget, version: Equinox 2004
   usb0: using arc_udc, OUT ep1out IN ep1in STATUS ep2in
   usb0: MAC 02:ce:30:e4:37:c5
   usb0: HOST MAC 82:d6:25:ce:e5:23
   usb0: RNDIS ready
   arcotg_udc: arc_udc bind to driver ether
   ```

2. You should see a new network interface:

   ```
   mx31# ifconfig usb0
   usb0    Link encap:Ethernet  HWaddr 02:CE:30:E4:37:C5
           BROADCAST MULTICAST  MTU:1500  Metric:1
           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
   ```

3. Configure the usb net interface on the mx31 with a not-otherwise-routable IP address. The IP address you choose will vary by location. It should not be in the range of your LAN subnet.

   ```
   mx31# ifconfig usb0 10.0.9.2 up
   ```

4. Now, plug the Mini-B-to-A cable into the OTG_HS(J1) connector and a USB host machine. I'm going to use another linux machine for this example. When you connect the cable, you should see something similar to these messages in the linux host machine's log file. (Use the dmesg command.)

   ```
   Dec 14 13:15:54 d600 kernel: usb 4-2: new high speed USB device
   using ehci_hcd and address 9
   ```

```
Dec 14 13:15:54 d600 kernel: usb 4-2: configuration #3 chosen from
2 choices
Dec 14 13:15:59 d600 kernel: eth1: register usbnet at usb-
0000:07:00.2-2, CDC Ethernet Device
Dec 14 13:15:59 d600 kernel: usbcore: registered new driver usbnet
Dec 14 13:16:07 d600 ifup: No configuration found for eth1
```

5. Find the name of the new network interface on the host machine in the above messages. In this case, it's eth1.

6. Configure the usb net interface on the host machine with another address on the same network you chose for the mx31, above:

```
d600:~ # ifconfig eth1 10.0.9.1 up
```

7. You should now be able to do any standard network thing:

```
d600:~ # ping -c 3   10.0.9.2
PING 10.0.9.2 (10.0.9.2) 56(84) bytes of data.
64 bytes from 10.0.9.2: icmp_seq=1 ttl=64 time=2.19 ms
64 bytes from 10.0.9.2: icmp_seq=2 ttl=64 time=0.189 ms
64 bytes from 10.0.9.2: icmp_seq=3 ttl=64 time=0.185 ms

--- 10.0.9.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.185/0.855/2.191/0.944 ms
```

7.3. Testing Host and Gadget together

You can connect the host and device ports on the mx31 together and test host and gadget at the same time. It's basically the same as the above scenarios, where both ends of the usb cable are connected to the mx31.

1. Setup Mass Storage Gadget

```
mx31#  modprobe g_file_storage file=/dev/ram0
ARC USBOTG Device Controller driver version 1 August 2005 init
ARC USBOTG h/w ID=0x5  revision=0x40
g_file_storage gadget: File-backed Storage Gadget, version: 20 October 2004
g_file_storage gadget: Number of LUNs=1
g_file_storage gadget-lun0: ro=0, file: /dev/ram0
arcotg_udc: arc_udc bind to driver g_file_storage
mx31#
```

2. Load the host driver:

```
mx31#  modprobe ehci-hcd
...
ehci_hcd_h1 ehci_hcd_h1: USB 0.0 initialized, EHCI 1.00, driver 26 Oct 2004
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
mx31#
```

3. Connect the MiniB-to-A cable between the OTG_HS (J1) connector and the SER_HOST_FS (J5) connector. You should see these messages on the console:

```
usb 1-1: new full speed USB device using ehci_hcd_h1 and address 2
usb 1-1: not running at top speed; connect to a high speed hub
g_file_storage gadget: full speed config #1
scsi0 : SCSI emulation for USB Mass Storage devices
  Vendor: Linux     Model: File-Stor Gadget  Rev: 0314
  Type:   Direct-Access                      ANSI SCSI revision: 02
SCSI device sda: 32768 512-byte hdwr sectors (17 MB)
sda: assuming drive cache: write through
SCSI device sda: 32768 512-byte hdwr sectors (17 MB)
sda: assuming drive cache: write through
 /dev/scsi/host0/bus0/target0/lun0: unknown partition table
Attached scsi disk sda at scsi0, channel 0, id 0, lun 0
```

4. Partition the disk, and make a file system:

```
mx31# fdisk /dev/sda

Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel.
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by
w(rite)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (2-1024, default 2):
Using default value 2
Last cylinder or +size or +sizeM or +sizeK (2-1024, default 1024):
Using default value 1024

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
SCSI device sda: 32768 512-byte hdwr sectors (17 MB)
sda: assuming drive cache: write through
 /dev/scsi/host0/bus0/target0/lun0: p1
SCSI device sda: 32768 512-byte hdwr sectors (17 MB)
sda: assuming drive cache: write through
 /dev/scsi/host0/bus0/target0/lun0: p1
Syncing disks.
mx31#


mx31# mkfs.ext2 /dev/sda1
mke2fs 1.34 (25-Jul-2003)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
```

```
Fragment size=1024 (log=0)
4096 inodes, 16368 blocks
818 blocks (5.00%) reserved for the super user
First data block=1
2 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
      8193

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
mx31#
mx31#
```

5. Now you can mount and access the "drive":

```
mx31# mount /dev/sda1 /mnt/usb/
mx31# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                73.3G     61.5G      8.1G  88% /
rwfs                    512.0k     28.0k    484.0k   5% /mnt/rwfs
rwfs                    512.0k     28.0k    484.0k   5% /tmp
rwfs                    512.0k     28.0k    484.0k   5% /var
/dev/sda1                15.5M     13.0k     14.7M   0% /mnt/usb
mx31#
```

7.4. Testing USB-OTG

There are three components to a Linux USB OTG system: the transceiver driver, the host driver and the gadget driver. The host and gadget drivers are essentially the same as on systems with no OTG support, with additional logic to handle dynamic activation and deactivation. The transceiver driver is responsible for noticing USB cable insertion events, and telling the host and gadget drivers to become active or inactive.

Since USB configuration is a fairly dynamic thing on any system, these drivers should all be built as modules. The transceiver driver must be loaded first, followed by the host driver, and then the gadget driver.

1. load the transceiver driver

```
mx31# modprobe isp1504_arc
driver isp1504_arc, Revision: 1.0
mx31#
```

2. load the host driver

```
mx31# modprobe ehci-hcd
ehci_hcd_h1 ehci_hcd_h1: new USB bus registered, assigned bus number 1
ehci_hcd_h1 ehci_hcd_h1: USB 0.0 initialized, EHCI 1.00, driver 26 Oct
2004
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
ehci_hcd_h2 ehci_hcd_h2: new USB bus registered, assigned bus number 2
```

```
ehci_hcd_h2 ehci_hcd_h2: USB 0.0 initialized, EHCI 1.00, driver 26 Oct
2004
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
ehci_hcd_otg_hs ehci_hcd_otg_hs: new USB bus registered, assigned bus
number 3
ehci_hcd_otg_hs ehci_hcd_otg_hs: USB 0.0 initialized, EHCI 1.00, driver
26 Oct 2004
hub 3-0:1.0: USB hub found
hub 3-0:1.0: 1 port detected
mx31#
```

3. load the gadget drivers

```
mx31# modprobe g_file_storage file=/dev/ram/0
ARC USBOTG Device Controller driver version 1 August 2005 init
ARC USBOTG h/w ID=0x5  revision=0x40
g_file_storage gadget: File-backed Storage Gadget, version: 20 October
2004
g_file_storage gadget: Number of LUNs=1
g_file_storage gadget-lun0: ro=0, file: /dev/rd/0
arcotg_udc: gadget arc_udc bound to driver g_file_storage
mx31#
```

Depending on which type of cable is inserted into the OTG_HS (J1) connector, the MX31 will act as a USB Host or USB Gadget. To be a gadget, connect the supplied Mini-B-to-A to a host machine. To be a host, connect a USB device to the MX31 using a cable with a Mini-A connector.

If you wish to change the MX31's gadget type, you can unload the current gadget driver and load another one:

```
mx31# rmmod g_file_storage
unregistered gadget driver 'g_file_storage'

mx31# modprobe g_ether
usb0: Ethernet Gadget, version: Equinox 2004
usb0: using arc_udc, OUT ep1out IN ep1in STATUS ep2in
usb0: MAC 52:fa:4c:61:bd:71
usb0: HOST MAC d2:37:15:4d:51:ce
usb0: RNDIS ready
arcotg_udc: gadget arc_udc bound to driver ether
```

8. Known Issues

```
Pin conflicts.
There are devices on the MX31ADS which conflict with USB pins, and which
cannot be used at the same time as parts of the USB controller.

- USBH1 conflicts with ATA.
- USBH2 conflicts with ATA and NAND flash.


USB Power
```

The host ports on the mx31 are able to supply a limited amount of power to connected devices. Current is limited to approximately 150mA per port. If you connect a device that requires more bus power, you'll see messages on the console similar to this:

```
usb 2-1: new high speed USB device using ehci_h2 and address 2
usb 2-1: no configuration chosen from 1 choice
```

The device will not be usable.

Some USB devices do not correctly report their power requirements. If a device reports that it needs less than 150mA when in fact it requires more, you'll see messages on the console similar to these:

```
hub 2-0:1.0: Cannot enable port 1.  Maybe the USB cable is bad?
usb 2-1: new high speed USB device using ehci_h2 and address 5
hub 2-0:1.0: Cannot enable port 1.  Maybe the USB cable is bad?
usb 2-1: new high speed USB device using ehci_h2 and address 6
usb 2-1: device not accepting address 6, error -71
hub 2-0:1.0: Cannot enable port 1.  Maybe the USB cable is bad?
```

To use devices that require more than 150mA of power, you can plug them into a powered hub, and plug the hub into the mx31.