SL811HS Linux USB Host Control Driver User's Guide

Cypress Semiconductor Corporation • 3901 North First Street • San Jose • CA 95134 • 408-943-2600
March 18, 2002

SL811HS Linux USB HCD

1. Introduction 3
1.1 Scope 3
1.2 Overview 3
1.3 Linux USB stack 3
1.4 SL811HS Linux Host Controller Driver 4
2. Definitions 6
3. System Development Environment 6
3.1 SL811HS 6
3.2 Development target 6
3.2.1 Accelent IDP board 6
3.2.2 Linux kernel version 2.4.4 6
3.3 Host computer with cross-development tools 7
4. Setting up the SL811HS Development Environment 7
5. Example Hardware Connection 7
6. Installing the Linux Kernel Source Code 8
7. Installing the GCC-ARM Cross-Development Toolchain 9
8. Physical-to-Virtual Memory Mapping 9
9. Interrupts 10
10. Building and Flashing the Linux Image 10
10.1 Building the Image 10
10.2 Flashing the image 11
11. Working with USB Devices 11
11.1 USB Device File System 11
11.2 USB Mass Storage Class 12
11.3 ADMtek Pegasus-based USB to Ethernet device 12
11.4 USB Keyboard 13
11.5 USB Printer 13
12. Tested Devices 13
13. References 13
13.1 SL811HS Application Notes 13
13.2 SL811HS/T USB Host/Slave Controllers Hardware Specification 13
13.3 Programming Guide for Linux USB Device Drivers. (http://usb.cs.tum.edu/usbdoc) 13
13.4 Acclent StrongARM IDP: Linux Software User's Guide 13
13.5 Linux USB Guide (http://www.linux-usb.org) 14
14. Document Revision History 15

1. INTRODUCTION

1.1 Scope

The primary objective of this document is to help developers to understand the SL811HS Linux USB host controller driver. This document covers the following topics:

- A general overview of the SL811HS board developed by Cypress Semiconductor, the Linux USB stack, and the SL811HS Linux HCD
- Description of the Linux kernel installation process
- Description of the software environment setup process
- Description of the process to set up USB test devices

1.2 Overview

The SL811HS USB Host embedded host controller is a single chip USB embedded host solution that can communicate with either full-speed or low-speed USB peripherals. The SL811HS can interface to devices such as microprocessors, micro-controllers, DSPs or directly to a variety of buses such as ISA, PCMCIA and others. The SL811HS USB controller conforms to the low and full speed requirements of USB Specification Revision 2.0.

Cypress Semiconductor has developed the SL811HS Linux host driver. The host driver can be integrated into the Linux USB host stack to provide a complete USB solution for the Linux environment.

1.3 Linux USB stack

The Linux USB stack can be broken down into three main layers: USB device driver, USB core, and the USB host controller driver (HCD). The USB device driver establishes virtual connections, configures, and communicates with the device(s). It assembles the data into a USB request block (URB) and passes the request to the USB core by a set of application programming interfaces (API).

The USB core is an intermediate layer between the device driver and the HCD. The USB core handles the following:

- USB device enumeration and configuration
- Loading and unloading of the device driver as required
- Interfacing with the device driver via a set of APIs

- Interfacing with the HCD via a set of APIs

The Linux USB HCD is a hardware abstraction layer that hides the hardware control implementation from the rest of the USB stack. The HCD accepts USB requests from the Linux USB core, parses the USB request and creates a USB transaction. The HCD schedules the USB transaction and transmits it when the bandwidth is available.

The Linux USB stack is shown in figure 1.

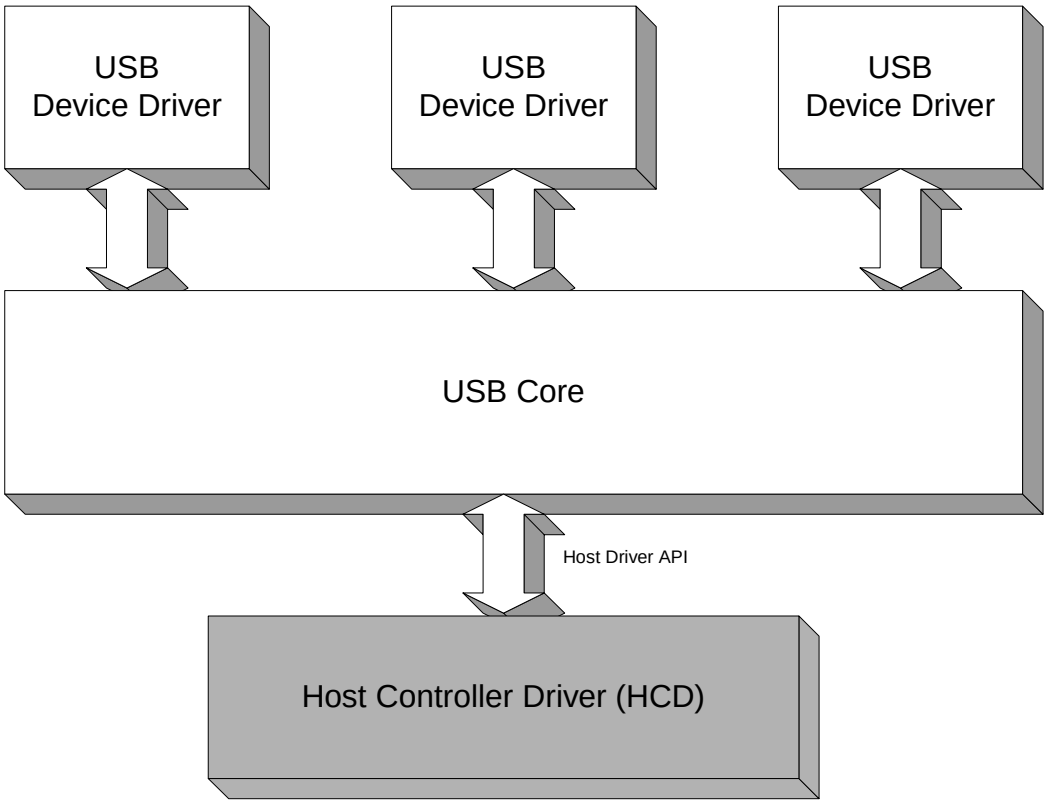

Figure 1. Linux USB stack

1.4 SL811HS Linux Host Controller Driver

The SL811HS HCD complements the existing Linux USB stack by providing supports for the SL811HS USB host controller. The SL811HS Linux HCD can be broken down into the following major functionalities:

- Support for a virtual root hub

- USB request queuing
- USB request scheduling
- Interrupt handling
- USB requests translating into USB transactions and packets
- USB packet transmitting and receiving

The HCD supports the interrupt, control, and bulk endpoints, but the current version does not support isochrononus endpoints.

The HCD interfaces with the Linux USB core by a set of application interfaces (API). The following table describes a subset of the API functions.

Function Name	Description
Usb_alloc_bus()	This function creates the host controller bus structure with specific driver operations and initialize all internal objects..
Usb_free_bus()	This function deallocates an existing host controller bus structure.
Usb_register_bus()	This function uses the previously allocated usb_bus structure and registers the host controller driver to the usb core.
Usb_deregister_bus()	This function deregisters the usb_bus structure.
Usb_new_device()	This function connects a new USB device to the USB core. It initializes the USB device information and sets up the topology.
usb_roothub_string()	Obtain the root hub identifier.
usb_submit_urb()	This is a function pointer that points to the HCD submit function. It allows the USB core to transmit data

Table 1. USB Core HCD API

2. DEFINITIONS

HCD	Host Controller Driver
SL811HS	The SL811HS is a Cypress USB 1.1 host controller
USB	Universal Serial Bus
USB CORE	A Linux USB stack
IDP	The Integrated Development Platform built by Accelent Corporation (http://www.accelent.com)
Linux Host	A host computer that compiles the source code
Target	The Integrated Development Platform.
URB	USB Request Block, a data structure containing the USB request.

3. SYSTEM DEVELOPMENT ENVIRONMENT

The system development environment is composed of

- SL811HS
- A target consisting of an Accelent IDP board and Linux version 2.4.4
- Host computer and cross-compile tools

3.1 SL811HS

3.2 Development target

3.2.1 Accelent IDP board

The SL811HS HCD has been tested with the Intel SA1100/1110 processor on the Accelent IDP board. It can be ported to other architectures supported by Linux.

3.2.2 Linux kernel version 2.4.4

The SL811HS HCD has been tested with the Linux kernel version 2.4.4 with kernel patches from Accelent. The host driver has not been tested with other Linux kernels, but it can be modified to accommodate other kernel releases, if necessary.

3.3 Host computer with cross-development tools

A host computer has cross-development tools to create a bootable image for the target. This host machine can be any Linux machine with GNU tools installed.

4. SETTING UP THE SL811HS DEVELOPMENT ENVIRONMENT

The setup process involves the following:

1) Connect the SL811HS to the development platform
2) Install the Linux Kernel on the host computer
3) Install cross-development tools on the host computer
4) Define the memory map
5) Define the SL811HS hardware interrupt
6) Compile and download the kernel
7) Test the SL811HS HCD with USB devices

The following sections will describe each of the steps in detail.

5. EXAMPLE HARDWARE CONNECTION

The SL811HS connects to the Accelent IDP via the expansion bus. The Accelent IDP utilizes the Intel StrongARM SA1110 processor. The following is the wiring list for the IDP and SL811HS:

Expansion Bus (P16)	IDP SIGNAL	SL811HS SIGNAL
37	XB_A16	A0
57	/XB_OE	NRD
59	/XB_WE	NWR
71	/XB_CS7	NCS
10	GND	GND
85	/RESET_OUT	NRST
77	IRQ_XB	INTRQ
12	GND	GND
16	XB_D0	D0
18	XB_D1	D1
20	XB_D2	D2
22	XB_D3	D3
24	XB_D4	D4
26	XB_D5	D5
28	XB_D6	D6

30	XB_D7	D7
79	GND	GND
6	+5V	+5V

Table 2. IDP extension bus connection

In this development platform, the physical address register is mapped to 0x4380000 and the data register is mapped to 0x43810000. Section 8 will describe the physical-to-virtual memory address mapping in detail.

6. INSTALLING THE LINUX KERNEL SOURCE CODE

Accelent Corporation supplies the Linux kernel version 2.4.4 along with three patches. These patches are modifications to the existing 2.4.4 kernel to support the SA1110 processor and the IDP board. Cypress Semiconductor has supplied another patch file to support the SL811HS host driver. This patch will install the SL811HS HCD source code and add the SL811HS HCD into the configuration file. The following files are included in the patch file:

Hc_simple.c	A simple HCD front end that interacts with the USB core and handles USB request blocks.
Hc_sl811.c	A HCD that handles interrupts as well as the transmission and reception of packets.
Hc_sl811_rh.c	This file contains virtual root hub routines
Hc_simple.h	header file
Hc_sl811.h	header file

The following summarizes the steps to install the Linux kernel source code.

1) Install the Linux kernel version 2.4.4
2) Apply patch 2.4.4-rmk3
3) Apply patch 2.4.4-rmk3-np1
4) Apply patch 2.4.4-rmk3-np1-asi1
5) Apply patch SL811HS
6) Copy rootfs.tar.gz to the kernel directory
7) Make sure bin2bin and mkfs.jffs2 are in the current directory

7. INSTALLING THE GCC-ARM CROSS-DEVELOPMENT TOOLCHAIN

The Accelent SDK CD-ROM includes an ARM cross-development toolchain to create a downloadable image for the target. Refer to chapter 5 of the Accelent IDP User's Guide for toolchain installation.

8. PHYSICAL-TO-VIRTUAL MEMORY MAPPING

The Linux kernel accesses the SL811HS addresses via virtual memory addressing. Therefore, all SL811HS physical memory addresses must be mapped to the corresponding virtual memory addresses. Table 3 contains the memory mapping information.

	Physical Address	**Kernel Virtual Address**
Access to Address Register	0x43800000	0xD3800000
Access to Data Register	0x43810000	0xD3810000

Table 3. SL811HS Memory Mapping

The Linux kernel uses the accelent_io_desc[] data structure to store the memory mapping information. This data structure is located in the file **$Linux/arch/mach_sa1100/accelent.c**. This information is board specific, and must be modified for different boards. The following code excerpt maps the SL811HS physical addresses to the Linux virtual address.

```
static struct map_desc accelent_io_desc[] __initdata = {
…….
{0xd3800000,  /* Virtual Base Address */
0x43800000,   /* Physical Base Address */
0x00020000,   /* Memory Region Size */
DOMAIN_IO,  /* Domain */
1,  /* Readable: 1 = readable, 0 = not readable */
1,  /* Writeable: 1 = writeable, 0 = not writeable */
0,  /* Cacheable: 1 = Cacheable, 0 = not cacheable */
0   /* Blockable: 1 = Blockable, 0 = not blockable */
},
…….
}
```

Note: The physical and virtual address has been customized for the SL811HS connected to the Accelent IDP. The physical and virtual address may be different for other development platforms.

9. INTERRUPTS

The SL811HS interrupt line is physically connected to the SA1110 general-purpose I/O line 13 and it can be configured as either an interrupt line or an I/O line via the GPIO registers. During initialization, the SL811HS HCD configures the GPIO 13 line as an interrupt line that triggers on the rising edge of the signal. The following code segment shows this configuration.

```
void init_irq(void)
{
   GPDR &= ~(1<<13); // Set GPIO13 as an interrupt

/* Set GPIO13 irq to be a rise edge trigger */

   set_GPIO_IRQ_edge (1<<13  , GPIO_RISING_EDGE);
}
```

Similar to the memory mapping, the interrupt line has been customized for the SA1110 IDP. The interrupt setup method may be different for other platforms.

10. BUILDING AND FLASHING THE LINUX IMAGE

10.1 Building the Image

From the kernel directory, type in the following commands to build a kernel image:

```
make menuconfig
make dep
make zImage
make modules
make modules_install
make jffs_rootfs
make jffs_image
```

Three images are created with the build process and they are listed below:

nk_jffs_flash.bin	this image will flash the entire system (kernel and rootfs)
nk_kernel.bin	this image will be loaded to RAM and the code will run from RAM.
nk_kernel_flash.bin	this image will flash the kernel, but not the rootfs.

10.2 Flashing the image

1) Rename the image file to nk.bin and place on an ATA flash card
2) Boot the Accelent IDP with the ATA flash card inserted
3) The image will be transferred to the RAM or flash depending on the image

11. WORKING WITH USB DEVICES

11.1 USB Device File System

The USB device file system is a dynamically generated file system. This file system is typically mounted in the /proc/bus/usb directory. The files within this directory are useful as a debugging tool. To make the USB device file system available, you must select the following from the configuration menu

USB ->Preliminary USB Device File System

Once the kernel is compiled with the USB file system support, you can mount the USB file system using the mount command as follows:

mount –t usbdevfs none /proc/bus/usb

The output in the /proc/bus/usb directory should appear as follows:

```
dr-xr-xr-x   1 root     root            0 Jan  1 00:00 001
-r--r--r--   1 root     root            0 Jan  2 00:38 devices
-r--r--r--   1 root     root            0 Jan  2 00:38 drivers
```

The “devices” file contains information about the currently connected USB devices. It includes USB enumeration data such as topology, device descriptor, current configuration, endpoint descriptor, etc. This information is useful to check if the USB device has been successfully enumerated.

The “drivers” file contains information about the currently installed USB device drivers.

11.2 USB Mass Storage Class

The mass storage driver presents the USB device as a SCSI device, therefore the SCSI support must be enabled. In the configuration menu, you must select the following option:

```
SCSI Support->SCSI support
SCSI Support->SCSI generic support
USB->USB mass storage support
```

You may select the options to support for various type of mass storage devices

```
SCSI Support->SCSI disk support  (USB IDE drive)
SCSI Support->SCSI CD-ROM support
SCSI Support->SCSI Tape
```

Once the kernel is loaded, you can access the mass storage using the following command:

```
mkdir /mnt/usbhd
mknod /dev/sda b 8 1
mount –t vfat /dev/sda /mnt/usbhd
```

11.3 ADMtek Pegasus-based USB to Ethernet device

To enable Pegasus-based devices, you must select the configuration option “USB->USB ADMtek Pegasus-based Ethernet devices support”

Once it is compiled into the kernel, the setup process is similar to other Ethernet devices, i.e. you can use the “ifconfig” command to configure the device. The following command assigns a static IP address to the USB device.

```
ifconfig eth0 172.19.3.33
```

11.4 USB Keyboard

From the configuration menu, you should select the following option:

Input Core->Input Core Support
Input Core->keyboard support
USB->USB human interface device (HID) support
Character Devices->PS/2 keyboard support

Unselect the following option from the configuration menu:

Character Device->ASI: Support for keyboard

11.5 USB Printer

You need to create a device node entry for the USB printer. Use the following command:

```
mkdir /dev/usb
mknod /dev/usb/lp0 c 180 0
```

12. TESTED DEVICES

The following devices have been tested with the Accelent IDP board:

Linksys Pegasus-based USB to Ethernet Adaptor
ISD315 USB mass storage device
USB keyboard
USB printer
USB hub

13. REFERENCES

13.1 SL811HS Application Notes

13.2 SL811HS/T USB Host/Slave Controllers Hardware Specification

13.3 Programming Guide for Linux USB Device Drivers. (http://usb.cs.tum.edu/usbdoc)

13.4 Acclent StrongARM IDP: Linux Software User's Guide

13.5 Linux USB Guide (http://www.linux-usb.org)

14. DOCUMENT REVISION HISTORY

Revision #	Date	Comments
1.0	3/22/2002	Initial Cypress revision