API Specification for

Linux Host Bus Interface Driver

V 1.0

Table of Contents

1 Introduction 5
1.2 Purpose of the Document 5
3.1 Scope 5
3.1 Abbreviations 5
3.1 References 5
3.1 Assumptions 5

2 Design Approach 5
2.1 Compile – Time Options 5

3 PROCFS Support 6
4.1 init_driver 6
4.2 term_driver 7
4.3 open_device 7
4.4 close_device 7
4.5 write_reg 7
4.6 read_reg 7
4.7 cfgrec 7
4.8 load_fw 7
4.9 start_fw 8
4.10 flash_erase 8
4.11 flash_save_fwrcfgrec 8
4.12 flash_load_fwrcfgrec 8
4.13 PROCFS Usage Examples 8

5 HBI Boot Option 10

List of Tables

Table 1 Abbreviations used in this document 5

List of Figures

No table of figures entries found.

Document Owner	Shally Verma
Version	1.0
Date	29 April 2016

Document Change History			
Date	**Version**	**Changed by**	**Change Description**

1 Introduction

This document describes the linux kernel driver for Host Bus Interface driver for Microsemi VPROC device family.

1.2 Purpose of the Document

This document outlines the features exported and supported by HBI linux kernel driver

3.1 Scope

This document is intended to be used by Applications using Microsemi VPROC device family on linux platform. Currently it support only kernel level driver. User space is not covered.

3.1 Abbreviations

Table 1 Abbreviations used in this document

Abbreviation	Explanation
HBI	Host Bus Interface
VPROC	Voice Processor

3.1 References

[1] ZL38040/50/60/80/51 firmware Manual

[2] ZL38004/ZL38005/ZL38012 firmware Manual

[3] SDK API Specification Manual

3.1 Assumptions

This document assumes that user is aware of HBI Driver for Microsemi VPROC device family.

2 Design Approach

HBI linux driver is an abstraction layer on top of HBI for users on linux platform. This driver supports various options to access HBI driver features from both user and kernel space.

During initialization, HBI linux driver primarily will do:

a. Initialization of procfs interface to open, close, boot, configure and read or write access to VPROC device over HBI
b. Registration as “character driver” to user island and accessible as /dev/hbi<major>:minor_num where Minor Num is fixed to 1.
c. Support IOCTLs to open, close, boot, configure and read or write access to VPROC device

One single HBI driver instance would support multiple devices. Once driver is initialized, user can open any device in a system using HBI_OPEN ioctl call with HBI_DEV_CFG (containing device address information) as associated IOCTL argument OR proc interface (for details, see PROCFS Support Section)

2.1 Compile – Time Options

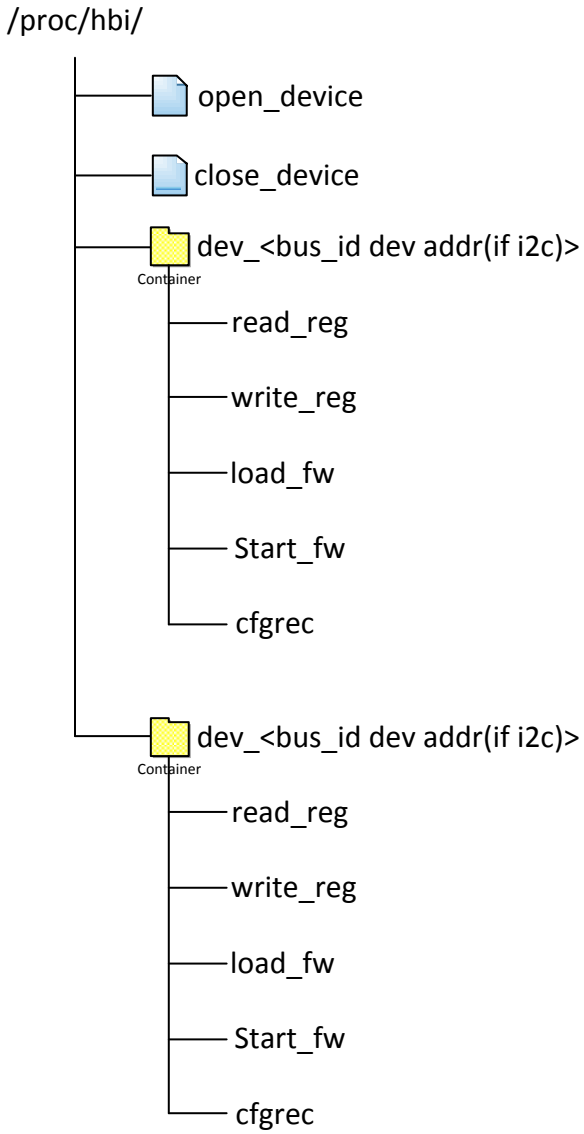
Below table summarizes the different compile time options influencing driver behavior and feature set.

Option	Description
HBI_ENABLE_PROCFS	If defined, then HBI driver will expose PROCFS interfaces
FLASH_PRESENT	Enable / Disable proc fs and IOCTL for flash operations support.
HBI_BUF_SIZE	Maximum buffer size used by HBI driver

3 PROCFS Support

HBI linux kernel driver initialize and create proc entries if compiled with HBI_ENABLE_PROCFS defined.

HBI linux kernel driver procfs interface looks something like this:



4.1 init_driver

This interface initializes a driver to be used by procfs. This should be 1st call made before invoking any other call. Usage:

cat /proc/hbi/init_driver

4.2 term_driver

This terminates the driver. This should be the invoked at last after all work is done.

Usage: cat /proc/hbi/term_driver

User should call term as many times init is called.

4.3 open_device

This interface opens a VPROC device. It should be called before invoking any other interface.

For device at i2c bus, it takes bus number and device address as an input.

echo <bus_num>:<dev_addr(in hex)> > /proc/hbi/open_device

Example, to open device at i2c bus 1 with slave address 0x45 give

echo 1:45 > /proc/hbi/open_device

If device is opened successfully, then procfs entry will be created with name “dev_145” (where 1 is bus id and 45 device address in hex) for further operations on device.

4.4 close_device

Write to this file closes the requested device. It takes device address (in hex) as input.

Example, to close device 0x45 enter

echo 45 > /proc/hbi/close_device

If device is closed successfully, then user should not see dev_<busnum,addr> entry

4.5 write_reg

Write to this file writes data from user buffer to device register. It takes register address and data bytes in hex as input. Example, to write 0x12345678 at register 0x000C from device at address 0x45 on i2c bus 1

echo 000C 12345678 > /proc/hbi/dev_145/write_reg

4.6 read_reg

Write to this file reads data from device memory to a user buffer. It takes register address and size as an input. Address in hex and size as number of bytes in decimal (should be multiple of 2).

Example to read 4 bytes from register 0x000c from device at address 0x45 on i2c bus 1

echo 000C 4 > /proc/hbi/dev_145/read_reg

if output of the command isn’t visible to console, run “dmesg” after command is complete

4.7 cfgrec

Read of this file will display current configuration record in device.

Write to this file will update device at address 0x45 on bus 1 on i2c with user passed configuration record file. User can update configuration record with command

cat <cfgrec_file_name.cr2> > /proc/hbi/dev_145/cfgrec

4.8 load_fw

Write to this file will load boot image into device. This interface now **ONLY** supports a binary image format unlike previous version which had .s3 boot image support. Binary format can be generated by running .s3 image through a tool given in sdk.

Example, to load boot image for device at address 0x45 at i2c bus 1

cat <firmware image file name.bin> > /proc/hbi/dev_145/load_fw

4.9 start_fw

Read to this file will start execution of mage loaded by load_fw interface. Example, to start firmware image loaded on device with address 0x45 at i2c bus 1

cat /proc/hbi/dev_145/start_fw

This command should be executed immediate after load_fw or flash_load_fwrcfgrec command else it may return an error.

4.10 flash_erase

Applicable if flash is present. Write to this file will erase inputted firmware image and associated configuration record from flash. Example to erase image 1 from device at address 0x45 on i2c bus 1

echo 1 > /proc/hbi/dev_145/flash_erase

Read to this file will simply erase whole flash

cat /proc/hbi/dev_145/flash_erase

4.11 flash_save_fwrcfgrec

Read to this file will save current loaded firmware and configuration record in RAM on flash. This is supposed to be used immediate after start_fw command. Example to save firmware on flash at device 0x45 i2c bus 1

cat /proc/hbi/dev_145/flash_save_fwrcfgrec

4.12 flash_load_fwrcfgrec

Write to this file will read given firmware image and associated configuration record from flash. Example, to read firmware image number 1 from flash into VPROC device ram at address 0x45 i2c bus 1 give

echo 1 > /proc/hbi/dev_145/flash_load_fwrcfgrec

4.13 PROCFS Usage Examples

Prerequisite

Load hbi linux module:

insmod hbi.ko (if user have root permission) or

sudo insmod hbi.ko

User should see “hbi” directory created under /proc i.e.

/proc/hbi as in snapshot below:

4.13.1 Open device at address 0x45 i2c bus 1

```
pi@raspberrypi:~$
pi@raspberrypi:~$ echo 1:45 > /proc/hbi/open_device
pi@raspberrypi:~$
pi@raspberrypi:~$ ls -l /proc/hbi/
total 0
-rw-rw-rw- 1 root root 0 Jun 29 08:19 close_device
dr-xr-xr-x 2 root root 0 Jun 29 08:19 dev_145
-rw-rw-rw- 1 root root 0 Jun 29 08:19 open_device
pi@raspberrypi:~$
pi@raspberrypi:~$ ls -l /proc/hbi/dev_145/
total 0
-rw-rw-rw- 1 root root 0 Jun 29 08:19 cfgrec
-rw-rw-rw- 1 root root 0 Jun 29 08:19 flash_erase
-rw-rw-rw- 1 root root 0 Jun 29 08:19 flash_load_fwrcfgrec
-rw-rw-rw- 1 root root 0 Jun 29 08:19 flash_save_fwrcfgrec
-rw-rw-rw- 1 root root 0 Jun 29 08:19 load_fw
-rw-rw-rw- 1 root root 0 Jun 29 08:19 read_reg
-rw-rw-rw- 1 root root 0 Jun 29 08:19 start_fw
-rw-rw-rw- 1 root root 0 Jun 29 08:19 write_reg
pi@raspberrypi:~$
```

4.13.2 Close device address 0x45 i2c bus 1 (remove entry dev_145)

```
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$ echo 45 > /proc/hbi/close_device
pi@raspberrypi:~$
pi@raspberrypi:~$ ls -l /proc/hbi/
total 0
-rw-rw-rw- 1 root root 0 Jun 29 08:45 close_device
-rw-rw-rw- 1 root root 0 Jun 29 08:45 open_device
pi@raspberrypi:~$
pi@raspberrypi:~$
```

4.13.3 Updating firmware on device and save it to flash
Open -> load_fw -> flash_save_fwrcfgrec

- echo 1:45 > /pro c/hbi/open_fw
- cat <filename>.s3 > /proc/hbi/dev_145/load_fw
- cat /proc/hbi/dev_145/start_fw (download firmware to device)
- cat /proc/hbi/dev_145/flash_save_fwrcfgrec (save downloaded firmware to flash)

4.11.4 Updating firmware on device and run it from RAM

- echo 1:45 > /proc/hbi/open_device
- cat <filename>.s3 > /proc/hbi/dev_145/load_fw
- cat /proc/hbi/dev_145/start_fw
- echo 1 > /proc/hbi/dev_145/start_fw

4.11.5 Boot device from specific image from flash.

- echo 1:45 > /proc/hbi/open_device (open device i2c bus 1 address 0x45)
- echo 2 > /proc/hbi/dev_145/flash_load_fwrcfgrec (read image number 2 from flash)
- echo 1 > /proc/hbi/dev_145/start_fw (set to start firmware execution)

5 HBI Boot Option

HBI linux kernel driver support two interfaces for boot from host: procfs and ioctl. In operations, ioctl interface is similar to procfs interface only exception is ioctl command takes buffer (either precompiled or runtime read from a file by an application) as an input whereas procfs take file as an input. This section only describes procfs interface for booting.

5.1 PROCFS interface

HBI linux driver expose following device specific interfaces.
/proc/hbi/dev_<addr>/load_fw
/proc/hbi/dev_<addr>/start_fw
/proc/hbi/dev_<addr>/flash_save_fw

These interfaces should be used in following sequence

- Load firmware image from host and start its execution:
  - /proc/hbi/dev_<addr>/load_fw → /proc/hbi/dev_<addr>/start_fw

- Load firmware from host and save it to flash
  - /proc/hbi/dev_<addr>/load_fw->/proc/hbi/dev_<addr>/flash_save_fw

- Load firmware from host and save it to flash and execute it
  - /proc/hbi/dev_<addr>/load_fw -> /proc/hbi/dev_<addr>/flash_save_fw → /proc/hbi/dev_<addr>/start_fw

For usage for each procfs interface, refer to section 4 PROCFS Support