<Product Area>

<Device>

API Specification for

Linux Host Bus Interface Driver

Draft Version 0.1

April 2015

# Table of Contents

[1 Introduction 5](#_Toc423351761)

[1.2 Purpose of the Document 5](#_Toc423351762)

[3.1 Scope 5](#_Toc423351763)

[3.1 Abbreviations 5](#_Toc423351764)

[3.1 References 5](#_Toc423351765)

[3.1 Assumptions 5](#_Toc423351766)

[2 Design Approach 5](#_Toc423351767)

[3 Public Data Structure 6](#_Toc423351768)

[3.1 IOCTL 6](#_Toc423351769)

[3.2 HBI\_LNX\_DRV\_FLASH\_ERASE\_ARGS 7](#_Toc423351770)

[3.3 HBI\_LNX\_DRV\_FLASH\_LOAD\_FWR\_ARGS 7](#_Toc423351771)

[3.4 HBI\_LNX\_DRV\_RW\_ARGS 7](#_Toc423351772)

[3.5 Compile – Time Options 7](#_Toc423351773)

[4 PROCFS Support 7](#_Toc423351774)

[4.1 open\_device 8](#_Toc423351775)

[4.2 close\_device 8](#_Toc423351776)

[4.3 write\_reg 9](#_Toc423351777)

[4.4 read\_reg 9](#_Toc423351778)

[4.5 cfgrec 9](#_Toc423351779)

[4.6 load\_fw 9](#_Toc423351780)

[4.7 start\_fw 9](#_Toc423351781)

[4.8 flash\_erase 9](#_Toc423351782)

[4.9 flash\_save\_fwrcfgrec 9](#_Toc423351783)

[4.10 flash\_load\_fwrcfgrec 10](#_Toc423351784)

[4.11 PROCFS Usage Examples 10](#_Toc423351785)

[5 HBI Boot Option 12](#_Toc423351786)

# List of Tables

[Table 1 Abbreviations used in this document 5](#_Toc416449667)

# List of Figures

**No table of figures entries found.**

|  |  |
| --- | --- |
| **Document Owner** | Shally Verma |
| **Version** | Draft |
| **Date** | 8 April 2015 |

| Document Change History | | | |
| --- | --- | --- | --- |
| Date | Version | Changed by | Change Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

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



## Purpose of the Document

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

## 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.

## Abbreviations

| Table 1 Abbreviations used in this document | |
| --- | --- |
| Abbreviation | Explanation |
| HBI | Host Bus Interface |
| VPROC | Voice Processor |
|  |  |

## References

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

[2] ZL38004/ZL38005/ZL38012 firmware Manual

[3] HBI\_Driver\_API\_Specification.doc

## Assumptions

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

# Design Approach

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

During initialization, HBI linux driver primarily will do:

1. Initialization of procfs interface to open, close, boot, configure and read or write access to VPROC device over HBI
2. Registration as “character driver” to user island and accessible as /dev/hbi<major>:minor\_num where Minor Num is fixed to 1.
3. 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)

\*We had an alternate option to register 1 driver instance per device. In that case, driver would have registered different character device node for each device to kernel. However, that approach required compile time mapping of device address to device numbers. To keep driver flexible to dynamically select and open device based on device actual address, we decided to open 1 driver instance and expose OPEN IOCTL call accepting device address and other configuration as IOCLT argument.

# Public Data Structure

Public header file for HBI linux kernel driver would primarily consists of IOCTL defines and associated arguments types.

## IOCTL

Below table summarize list of IOCTL as supported by HBI linux kernel driver along with expected input argument

|  |  |  |  |
| --- | --- | --- | --- |
| IOCTL | Description | Input | Output |
| HBI\_OPEN | This IOCTL will open a requested device | HBI\_DEV\_CFG structure with bus number and device address information | Output of this call will be handle to opened device |
| HBI\_CLOSE | This IOCLT will close a requested device | Device handle of the type unsigned int | None |
| HBI\_WRITE | Write given buffer to requested device memory | HBI\_LNX\_DRV\_RW\_ARGS (see details below) | None |
| HBI\_READ | Reads into a user buffer from device | HBI\_LNX\_DRV\_RW\_ARGS (see details below) | None |
| HBI\_LOAD\_FWR | Loads a boot image from host to device | HBI\_DATA | None |
| HBI\_START\_FWR | Run the firmware loaded into RAM | None | None |
| HBI\_FLASH\_ERASE | Erases whole or specific image(if give) on flash. If null argument is passed, then IOCTL function would erase whole flash else retrieve specific image number from argument list and erase that. | None or HBI\_LNX\_DRV\_FLASH\_ERASE\_ARGS. | None |
| HBI\_FLASH\_SAVE\_FWR | Save Firmware and Configuration record in RAM to flash | None | Image number of image just written |
| HBI\_FLASH\_LOAD\_FWR | Loads requested firmware and configuration record from RAM | HBI\_LNX\_DRV\_FLASH\_LOAD\_FWR\_ARGS |  |

## HBI\_LNX\_DRV\_FLASH\_ERASE\_ARGS

typedef struct

{

HBI\_STATUS status;

\_\_u32 image\_number; /\* firmware image number to be erased from flash \*/

}HBI\_LNX\_DRV\_FLASH\_ERASE\_ARGS;

## HBI\_LNX\_DRV\_FLASH\_LOAD\_FWR\_ARGS

Argument associated with HBI\_FLASH\_LOAD\_FWR IOCTL

typedef enum struct hbi\_lnx\_drv\_load\_fwr\_args{

HBI\_STATUS status;

\_\_u32 image\_number; /\* image number of firmware image on flash to read from \*/

} HBI\_LNX\_DRV\_LOAD\_FWR\_ARGS;

## HBI\_LNX\_DRV\_RW\_ARGS

This data type is an input with HBI\_READ and HBI\_WRITE IOCTL.

typedef struct hbi\_lnx\_drv\_rw\_arg

{

HBI\_HANDLE handle; /\* handle as returned by call to HBI\_OPEN IOCTL \*/

HBI\_STATUS status; /\* status of call \*/

void \*to; /\* destination location to read/write to \*/

void \*from; /\* source location to read/write from \*/

size\_t len; /\* len of data in bytes(should be multiple of 2)\*/

}HBI\_LNX\_DRV\_RW\_ARG;

## 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 |

# 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:



## 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.

## 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

## 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

## 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

## 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

## load\_fw

Write to this file will load boot image into driver. Example, to load boot image for device at address 0x45 at i2c bus 1

cat <firmware image file name.s3> > /proc/hbi/dev\_145/load\_fw

## start\_fw

Read to this file will write image to file loaded by load\_fw interface. Example, to start downloading of boot 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.

## 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

## 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

## 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

## PROCFS Usage Examples

Prerequisite

Load hbi linux module:

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

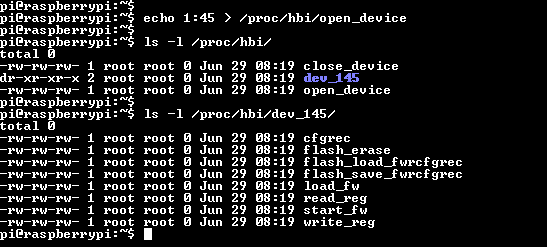
sudo insmod hbi.ko

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

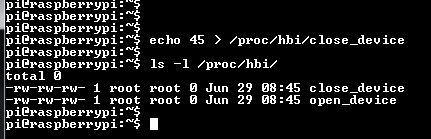
/proc/hbi as in snapshot below:



* + 1. Open device at address 0x45 i2c bus 1



* + 1. Close device address 0x45 i2c bus 1 (remove entry dev\_145)



* + 1. 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
  + 1. 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)

# 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.

* 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

* 1. IOCTL interface

HBI\_LOAD\_FW, HBI\_FLASH\_SAVE\_FW and HBI\_START\_FW are the supported ioctls for boot image loading. Every HBI\_LOAD\_FW call should be followed by HBI\_START\_FW and/or HBI\_FLASH\_SAVE\_FWR.

Please refer to IOCTL section for more details.