<Product Area>

<Device>

API Specification for

Host Bus Interface Driver

Draft Version 0.1

April 2015

# Table of Contents

[1 Introduction 5](#_Toc416449601)

[1.2 Purpose of the Document 5](#_Toc416449602)

[1.3 Scope 5](#_Toc416449603)

[1.4 Abbreviations 5](#_Toc416449604)

[1.5 References 5](#_Toc416449605)

[1.6 Assumptions 5](#_Toc416449606)

[2 Design Approach 5](#_Toc416449607)

[3 Public Data Structure 6](#_Toc416449608)

[2.1 HBI\_COMMAND 6](#_Toc416449609)

[2.2 Compile – Time Options 7](#_Toc416449610)

[2.2.1 HBI Driver Options 8](#_Toc416449611)

[2.2.2 Platform Specific Option 8](#_Toc416449612)

[2.3 HBI\_DEV\_CFG 8](#_Toc416449613)

[2.4 HBI\_STATUS 8](#_Toc416449614)

[2.5 HBI\_HANDLE 9](#_Toc416449615)

[2.6 HBI\_RESET\_MODE 9](#_Toc416449616)

[3 API 10](#_Toc416449617)

[3.1 HBI\_STATUS HBI\_init(void) 10](#_Toc416449618)

[3.2 HBI\_STATUS HBI\_open(HBI\_HANDLE \*handle, HBI\_DEV\_CFG \*cfg) 10](#_Toc416449619)

[3.3 HBI\_STATUS HBI\_close(HBI\_HANDLE handle) 10](#_Toc416449620)

[3.4 HBI\_STATUS HBI\_reset(HBI\_HANDLE handle, HBI\_RESET\_MODE reset\_mode) 11](#_Toc416449621)

[3.5 HBI\_STATUS HBI\_write(HBI\_HANDLE handle, void \*to , void \*from, size\_t size) 11](#_Toc416449622)

[3.6 HBI\_STATUS HBI\_read(HBI\_HANDLE handle, void \*to, void \*from, size\_t size) 11](#_Toc416449623)

[3.7 HBI\_STATUS HBI\_set\_command(HBI\_HANDLE handle, HBI\_COMMAND cmd, void \*val) 12](#_Toc416449624)

[HBI\_STATUS\_SUCCESS 12](#_Toc416449625)

# 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 software layer that implements Microsemi voice processor device Host Bus Interface.



## Purpose of the Document

This document outlines the public interfaces of “Host Bus Interface” Driver for use by application willing to setup control path to voice processor devices.

## Scope

Communication to Voice Processor is divided in two layers: Transport and Physical.

While Physical layer is defined and implemented by “System Service Layer”, this document defines host side transport layer driver referred as HBI driver.

While layer need to generic to work for both Galileo and Timberwolf device.However current version is documented for Timberwolf device class.

## Abbreviations

| Table Abbreviations used in this document | |
| --- | --- |
| Abbreviation | Explanation |
| HBI | Host Bus Interface |
|  |  |
|  |  |

## References

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

[2] ZL38004/ZL38005/ZL38012 firmware Manual

## Assumptions

This document assumes that user is aware of voice processor device transport layer of protocol.

# Design Approach

The HBI driver should be light weight and portable in nature and use static allocations.Driver should be Small, Simple Yet Powerful, Complete and Scalable to meet whole feature set of both Galileo and Timberwolf and accommodate future extension of feature set in device.

Driver needs one time initialization call. Once initialized, users can open devices upto a limit defined by HBI\_MAX\_NUM\_DEVS. Maximum number of users on a single device will be restricted by HBI\_MAX\_INST\_PER\_DEV macro.

User need to mention these parameters during compile time so that enough resources could be allocated.

# Public Data Structure

## HBI\_COMMAND

HBI Driver supports command-response interface. Out of all exposed, certain commands are issued to voice processor device firmware and some processed by HBI driver itself to perform particular task.

These commands are conditionally compiled options. For example, if host system doesn’t have flash then they can disable all of the flash related commands by de-selecting respective directive at compile time.

Below table summarize list of commands as supported by Timberwolf voice processor device class and respective descriptions. Some of the commands may require additional input for successful execution. Please see Timberwolf device firmware manual for required input w.r.t commands listed below.

|  |  |
| --- | --- |
| Command | Description |
| HBI\_CMD\_LOAD\_FWRCFG\_FROM\_FLASH | This command instructs device to load respective firmware and associated configuration record from flash connected to voice processor device, if any. |
| HBI\_CMD\_SAVE\_FWRCFG\_TO\_FLASH | This command instructs device to save current firmware and configuration record in memory to flash connected to voice processor device, if any. |
| HBI\_CMD\_ERASE\_WHOLE\_FLASH | This command instructs the firmware to erase whole flash connected to voice processor device, if any. |
| HBI\_CMD\_ERASE\_FWRCFG\_FROM\_FLASH | This command instructs firmware to erase specific firmware and associated configuration record from flash connected to voice processor device.if any |
| HBI\_CMD\_LOAD\_FWR\_FROM\_HOST | This command instructs **HBI driver** to write firmware image in to voice processor device and inform the firmware after write operation is complete |
| HBI\_CMD\_START\_FWR | This command instructs voice processor device Boot Code to start executing current firmware in RAM after it is being loaded either from flash or host |

## Compile – Time Options

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

List includes both Platform Specific and HBI specific defines.

### HBI Driver Options

|  |  |
| --- | --- |
| **Option** | **Description** |
| HBI\_ENABLE\_BOOT\_FROM\_HOST | If defined, then HBI driver will support booting voice processor device with specific firmware image from host |
| HBI\_MAX\_INST\_PER\_DEV | This defines maximum number of simultaneous user on a single voice processor device |
| HBI\_ENABLE\_FLASH\_OP | If defined, indicates the Microsemi Voice Processor device is connected to slave flash. In this case HBI driver will support all flash related operations |
| HBI\_DEV\_ENDIAN | This defines endian-ness of device either big or little at compile time. Please note since this is compile time option, thus once selected will configure voice processor device to a selected endian mode. |

### Platform Specific Option

These options can be defined in a ‘C’ style header file named, for example, platform.h that can be linked to SSL and HBI. Platform specific header file would carry following details:

|  |  |
| --- | --- |
| **Option** | **Description** |
| VPROC\_ MAX\_NUM\_DEVS | This defines maximum number of voice processor device in a system |
| VPROC\_DEV\_TYPE | This defines voice processor device i.e. value will be set to ZL38040, ZL38050, ZL38051,ZL38060,ZL38080 for Timberwolf and ZL38004, ZL38005 , ZL38012(?) for Galileo |
| VPROC\_BUS\_TYPE | This defines physical bus in use. This is system level/board specific define and ideally supposed to be used by SSL layer |
| BUILD\_TYPE | Release or Debug. If release, all Print functions would be faked or none. If debug, then print functions would be enabled. User can further select a debug level. This build option can be used by HBI and SSL both. |

## HBI\_DEV\_CFG

HBI\_DEV\_CFG is a data type which user use to setup run-time device configuration at the time of device open. Currently, it only input device number to be opened from user.

*typedef struct*

*{*

*int dev\_addr; /\* device addr to open (i2c bus addr or chip select) \*/*

*}HBI\_DEV\_CFG;*

## HBI\_STATUS

Below table gives status codes for HBI calls

|  |  |
| --- | --- |
| Code | Description |
| HBI\_STATUS\_NOT\_INIT | Driver not initialized. It requires to call HBI\_init() |
| HBI\_STATUS\_INVALID\_PARAM | Invalid or Bad parameter is passed to a HBI driver function call. Ex. any unallocated, out of range, unrecognized inputs can result to returning this return code. |
| HBI\_STATUS\_BAD\_HANDLE | Bad device handle i.e. given handle is not registered with HBI driver |
| HBI\_STATUS\_INTERNAL\_ERR | This error code is returned if there fails any platform specific call.Ex, any SSL call fails |
| HBI\_STATUS\_RESOURCE\_ERR | This error code is returned if HBI driver unable to acquire any resource. For example, given device is already opened or a lock is acquired. |
| HBI\_STATUS\_BAD\_IMAGE | Invalid firmware image number. This code is returned if any flash related HBI Command requiring a firmware image number is invoked and respective firmware image not found on flash. |
| HBI\_STATUS\_FLASH\_FULL | This code is returned by “Save Firmware To Flash” HBI command if flash device doesn’t have enough space to store it |
| HBI\_STATUS\_NO\_FLASH\_PRESENT | This code is returned when no flash is found in system |
| HBI\_STATUS\_COMMAND\_ERROR | This code is returned if any of the HBI Command fails |
| HBI\_STATUS\_INCOMPAT\_APP | This code is returned if firmware is incompatible to device |
| HBI\_STATUS\_SUCCESS | This code is returned on successful execution of HBI Driver calls |

## HBI\_HANDLE

HBI\_HANDLE is a Data type that represent handle to device being opened.

## HBI\_RESET\_MODE

Below table lists down different reset modes supported by **Timberwolf** voice processor device

|  |  |
| --- | --- |
| **Reset Modes** | **Description** |
| HBI\_RST\_AEC | Resets Acoustic Echo Filter Coefficients |
| HBI\_RST\_APP\_RST | Restart firmware without reloading firmware from flash, if present. In case of timberwolf device, it resets cross point switch configuration. |
| HBI\_RST\_POR | This simulates Power On Reset where it will clear everything including DSP memory and restart device with invocation of internal boot ROM code followed by loading of firmware from flash, if present. If flash not present in system then device will stop at Boot ROM prompt waiting on host to download firmware image to device RAM. This mode takes little longer to reset the device. |
| HBI\_RST\_HW | Resets voice processor device i.e. clears internal DSP memory and registers without reloading firmware from flash, if present. This mode takes less time to reset device and should be preferred if user want to reset device register values to default. |

# API

A HBI driver will be providing an API set for following feature set:

* Driver initialization/termination
* Device open/close
* Device address read/write
* Device reset
* Utility functions (conditionally compiled functions):
  + Boot firmware on device via host
  + Boot firmware device via flash
  + Configuration record read/write from/to flash
  + Erasing an image from flash
  + Erase complete flash

## HBI\_STATUS HBI\_init(void)

**Description:**

This will initialize data structure local to HBI driver and setup infrastructure for device communication. For example, Initialization of SSL layer may be invoked here.

**Parameters:**

None

**Return Codes:**

HBI\_STATUS\_RESOURCE\_ERR

HBI\_STATUS\_INTERNAL\_ERR

## HBI\_STATUS HBI\_open(HBI\_HANDLE \*handle, HBI\_DEV\_CFG \*cfg)

**Description:**

This function opens a Host Bus Interface to a device identified by address given in HBI\_DEV\_CFG. This is the function where SSL\_port\_open() & SSL\_lock\_create() will be called and device specific information is initialized and maintained locally.

**Parameters:**

Handle – Pointer passed by user and updated by function at successful execution.

HBI\_DEV\_CFG – Run time device configuration as selected by user

**Return Codes:**

HBI\_STATUS\_NOT\_INIT

HBI\_STATUS\_INVALID\_PARAM

HBI\_STATUS\_RESOURCE\_ERR

HBI\_STATUS\_INTERNAL\_ERR

HBI\_STATUS\_SUCCESS

## HBI\_STATUS HBI\_close(HBI\_HANDLE handle)

**Description:**

This function closes the host bus interface to device. Here SSL\_port\_close() will be called and any other device specific resources will be freed and information will be reset. Basically it will be reverse of HBI\_open() call.

**Parameters:**

Handle – reference handle as returned by HBI\_open()

**Return Code:**

HBI\_STATUS\_NOT\_INIT

HBI\_STATUS\_BAD\_HANDLE

HBI\_STATUS\_INTERNAL\_ERR

HBI\_STATUS\_SUCCESS

## HBI\_STATUS HBI\_reset(HBI\_HANDLE handle, HBI\_RESET\_MODE reset\_mode)

**Description:**

This function resets the device as per given mode.

**Input:**

Handle – Device Handle

Reset Mode – Modes as defined in HBI\_RESET\_MODE enumeration

**Return Code:**

HBI\_STATUS\_NOT\_INIT

HBI\_STATUS\_BAD\_HANDLE

HBI\_STATUS\_INTERNAL\_ERR

HBI\_STATUS\_INVALID\_PARAM

HBI\_STATUS\_SUCCESS

## HBI\_STATUS HBI\_write(HBI\_HANDLE handle, void \*to , void \*from, size\_t size)

**Description:**

This function writes data from user buffer to device memory.

Current voice processor device support 16-bit addressing, thus current pointer will be mapped to 16-bit address and depending upon size we can do 16-bit or 32-bit data write operation.

Prototype is kept generic so as to make HBI layer porting easy (without a need to change App) should we need to do 32-bit address read/write.

**Parameters:**

handle – Device Handle as returned by HBI\_open() call

to – destination location to be written to. Basically a device address

from – source/user buffer to copy data from

size – length of data to be written

**Return Code:**

HBI\_STATUS\_NOT\_INIT

HBI\_STATUS\_BAD\_HANDLE

HBI\_STATUS\_INVALID\_PARAM  
HBI\_STATUS\_INTERNAL\_ERR

HBI\_STATUS\_SUCCESS

## HBI\_STATUS HBI\_read(HBI\_HANDLE handle, void \*to, void \*from, size\_t size)

**Description:**

This function reads data from device memory to a user buffer.

Currently this function will be implemented to support 16-bit address read/write as current voice processor device Timberwolf and Galileo support 16-bit address mechanism. However, prototyping is kept generic enough to port it to 32-bit or any other address mechanism, if needed.

**Paramaters:**

handle – device handle as returned by HBI\_open()

from – device address to be read

to – user buffer where read data to be written/stored

size – length of the data to be read from device

**Return Codes:**

HBI\_STATUS\_NOT\_INIT

HBI\_STATUS\_INVALID\_PARAM

HBI\_STATUS\_INTERNAL\_ERR

HBI\_STATUS\_BAD\_HANDLE

HBI\_STATUS\_SUCCESS

\*HBI\_read() and HBI\_write() are primary calls to access voice processor device register. It gives user flexibility to do 16-bit register read/write or bulk read write.

## HBI\_STATUS HBI\_set\_command(HBI\_HANDLE handle, HBI\_COMMAND cmd, void \*val)

**Description:**

This function will issue commands supported over ‘Host Bus Interface’ to voice processor device[see section HBI\_COMMAND]. Please see firmware manual for information on various host commands supported by voice processor device.

**Parameters:**

Handle – device handle as returned by HBI\_open() call and where commands to be issued

cmd – HBI command as described in respective voice processor device firmware manual

val – value as relevant to current command in use

Return Codes:

HBI\_STATUS\_NOT\_INIT

HBI\_STATUS\_INVALID\_PARAM

HBI\_STATUS\_BAD\_HANDLE

HBI\_STATUS\_NO\_FLASH\_PRESENT

HBI\_STATUS\_FLASH\_FULL

HBI\_STATUS\_BAD\_IMAGE

HBI\_STATUS\_COMMAND\_ERROR

HBI\_STATUS\_INCOMPAT\_APP

HBI\_STATUS\_SUCCESS