GbtSmbus Utility User Guide

April 20, 2018

Revision 1.0

**Table of Contents**

Abstract

Revision History

Utility Feature

Utility Requirement

Function Overview

Console Command

GUI usage without External File

GUI usage with External File

Externel file

GbtSmbusCommonDesign.json

SmbDatasheet.json

Smbus Conguration BIN file

Abstract

Since Philips corporation made a standard specification for device telecommunication (I2C) in 1994, it becomes the most popular bus for engineers who are willing to exploit I2C compatible chip on computer development in recent year. The benefit of I2C is its universal and easy-handle. After that, Intel corporation based on the concept of I2C-designed, and then develop a similar type which named System Management Bus(SMB). Both of them make a chip possible to communicate with other compatible chips by simple protocol in two serial bus-lines.

Revision History

| Table 1-0 | | | |
| --- | --- | --- | --- |
| Revision Number | Description | Revision Date | Owner |
| 1.0 | Initial release | Aoril 20, 2018 | Chant |
| **End 1-0** | | | |

Utility Feature

Smbus-hierarchy-tree utility support two method to monitor SMB interface. One is legacy method, the other is GUI control method. In legacy method, User may configure smbus device by typing raw command just like every default solution.

For GUI control method, utility will auto scan smbus devices that the host can find as many as possible. Credit to a smart AI module in utility, so we can achieve the auto scan on smbus.

After AI finish the auto scan, user can feel free to control selected device and modify its configuration.

The spotlight of this utility is external file( GbtSmbusCommonDesign.json & SmbDatasheet.json ) supported. You may call them as “Smbus Navigation”. With this exteranl file, the utility performs ALL smbus device in the same time and take a note on specific configuration offset register which is defined in datasheet. Those functions work in one purpose: Let user feel convenient to design logic circuit! Besides, It is free for user to prepare this external file or not. Make your own decision.

Figure 1. Smbus Utility encodes and compresses the smbus device’s register individually, and then FW may easily allocate all the smbus device by decoding the package.



Figure 2. Smbus Utility will build the “Smbus Hierarchy Tree” with external file



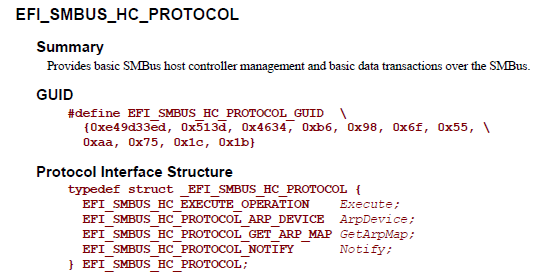
Utility Requirement

Supported Operating Systems

1. EFI Shell version 2.6

BIOS Requirements

1. Have to register Smbus protocol on Runtime Service



HW Requirements

1. The smbus switch must belongs to PCA954X family

Utility Overview -- Console Command

1. ***Write smbus register with given offset and value***

Command Line Interface

**GbtSmbus.efi -w <Operation> <Smbus address> <Offset> <Register>**

Where,

|  |  |
| --- | --- |
| **Operator** | **Description** |
| -w | Indicate Smbus host to execute write command protocol |
| Operation | Write “Byte” or “Word” register |
| **Smbus Address** | Target the slave address in 8 bits (including r/**w** bit) |
| Offset | The location where user decide to write |
| Register | The data that user decide to write |

**Example: Write a byte value 0x18 into slave smbus 0xB0 with offset 0x06**



1. ***Read smbus register with given offset***

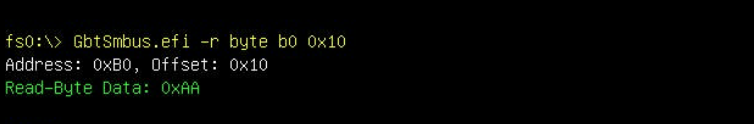
Command Line Interface

**GbtSmbus.efi -r <Operation> <Smbus address> <Offset>**

Where,

|  |  |
| --- | --- |
| **Operator** | **Description** |
| -r | Indicate Smbus host to execute read command protocol |
| Operation | Read “Byte” or “Word” register |
| **Smbus Address** | Target the slave address in 8 bits (including r/**w** bit) |
| Offset | The location where user decide to read |

**Example: Read a byte value from slave smbus 0xB2 with offset 0x10**



1. ***Dump smbus all configuration into BIN file***

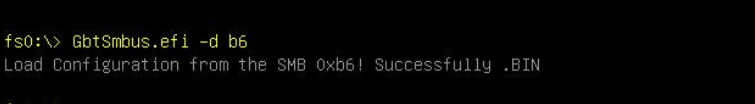
Command Line Interface

**GbtSmbus.efi -d <Smbus address>**

Where,

|  |  |
| --- | --- |
| **Operator** | **Description** |
| -d | Indicate Smbus host to execute read command protocol from offset 0~255 and dump value into BIN file |
| **Smbus Address** | Target the slave address in 8 bits (including r/**w** bit) |

**Example: Dump 0xB6 configuration into SmbConfig\_B6.bin**



1. ***Pour BIN file configuration into indicated smbus address register***

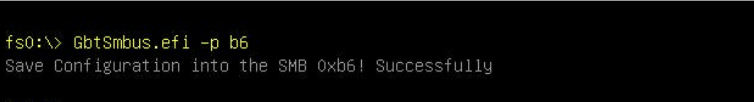
Command Line Interface

**GbtSmbus.efi -p <Smbus address>**

Where,

|  |  |
| --- | --- |
| **Operator** | **Description** |
| -p | Draw the value from BIN file and write all configuration back to indicated smbus address |
| **Smbus Address** | Target the slave address in 8 bits (including r/**w** bit) |

**Example: Dump 0xB6 configuration into SmbConfig\_B6**



Utility Overview -- GUI usage with External File

1. ***Auto Scan smbus device that Host can find out***

Command Line Interface

**GbtSmbus.efi –g**

Where,

|  |  |
| --- | --- |
| **Operator** | **Description** |
| -g | Indicate GUI menu |

***Utility can work with External file. That will make GUI function perform similar to RU-like screen.***

Utility with external file may build the “smbus tree” and list the smbus device on screen from smbus tree as many as possible. The precise to smbus tree depends on how accurate the external file would be.

In general, the external file- **GbtSmbusCommonDesign.json** will follow **Gigabyte smbus common design block diagram**. Refer to the figure paste below. it shows the relationship beween smbus tree menu and common design.

Please refer to “External File” topic, to realize how to create one by Common Design and device datasheet.



External File – GbtSmbusCommonDesign.json

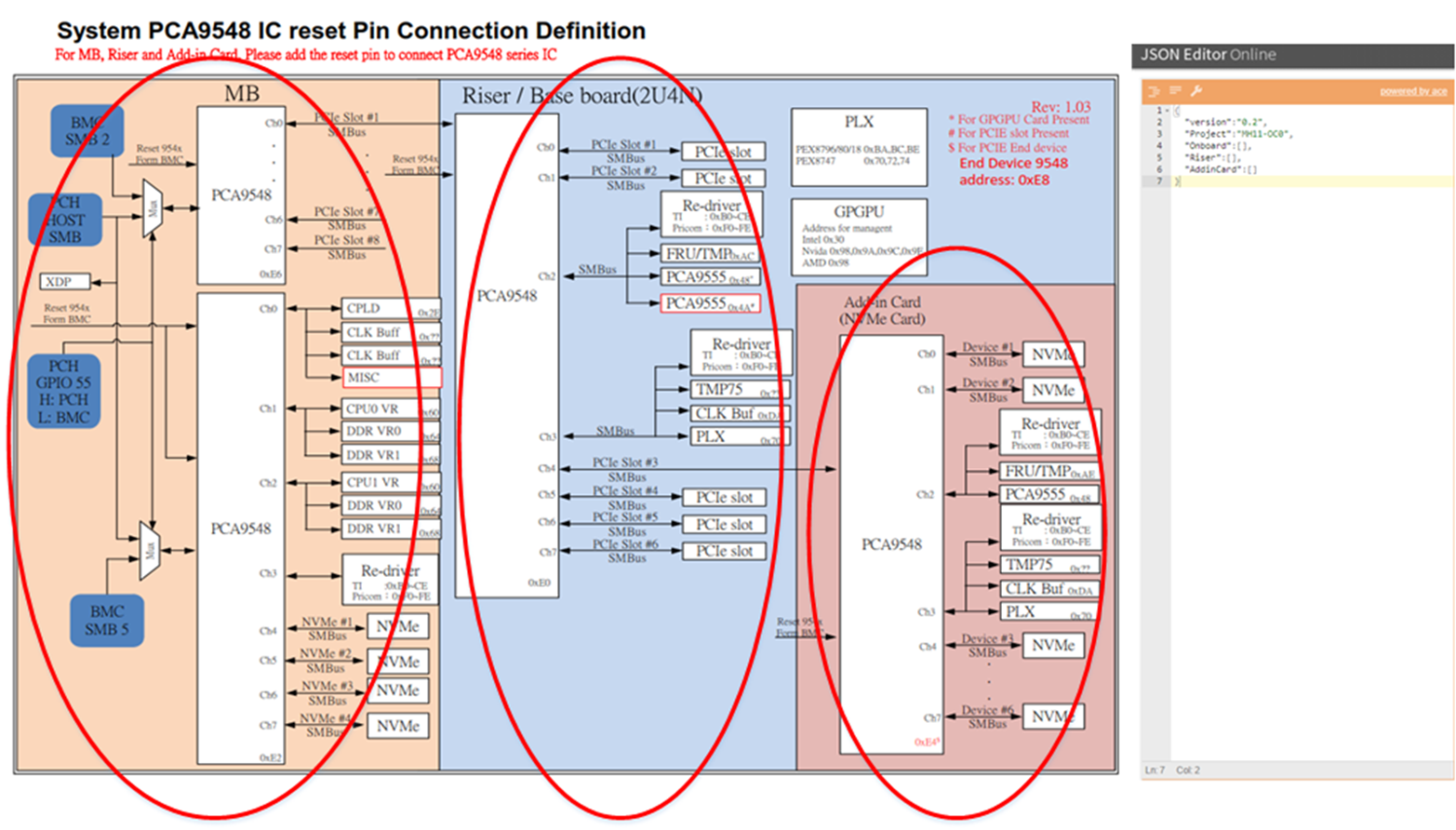
This chapter will teach user how to transfer a **Common Design Block Diagram** into **External File(JSON format).** Remember to name the file as “GbtSmbusCommonDesign.json” or utility can’t recognize the external file and function fail.

P.S. Recommend a website to user for creating and modifying a JSON file.

<https://jsoneditoronline.org/>

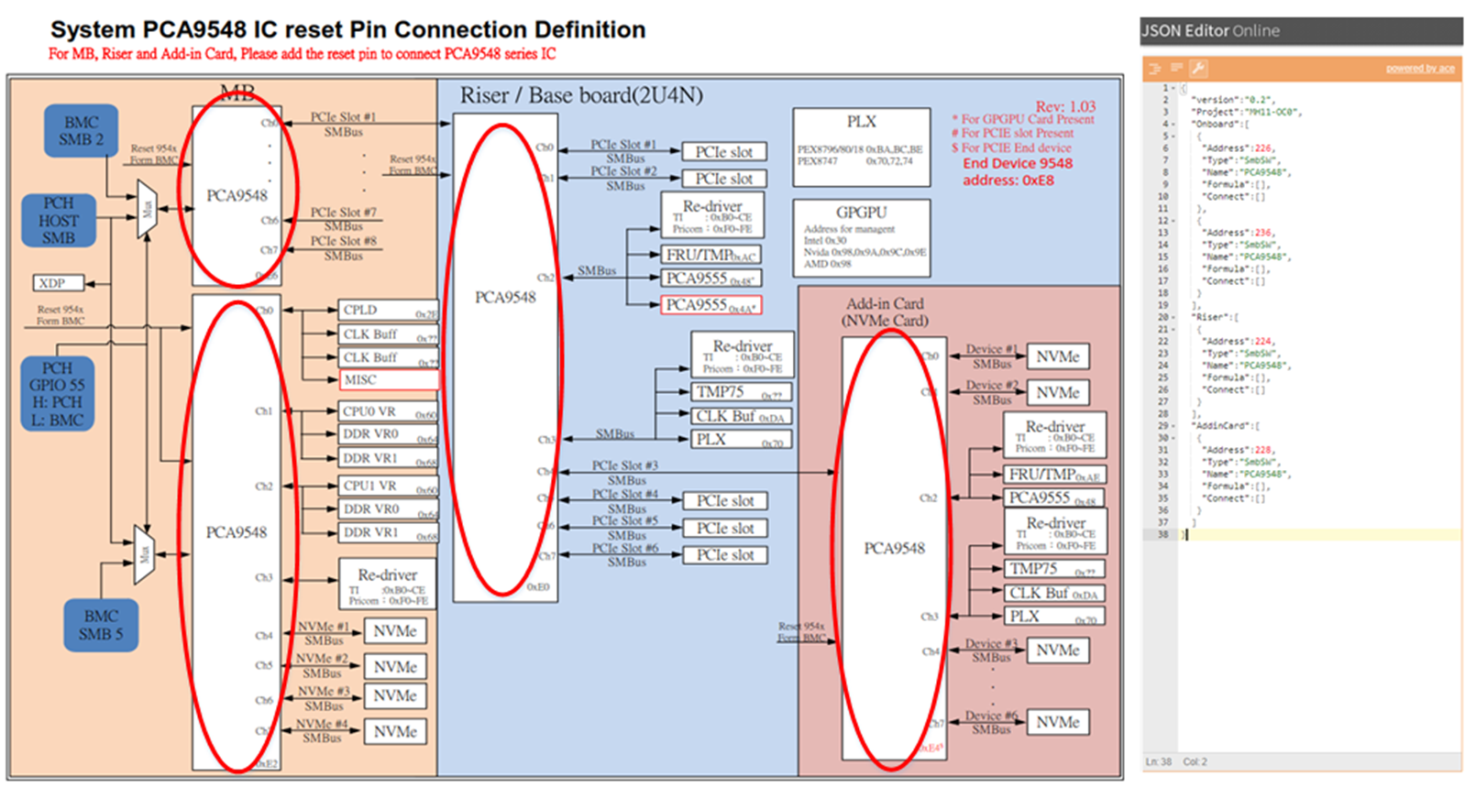
**Step 1.**

**Create the modules that might be used in this project. Add the external file version and project name. every module’s data structure have to be an array.**



**Step 2.**

**Found out all the smbus switch in module and add into to JSON file individually. Follow definition of the data structure.**

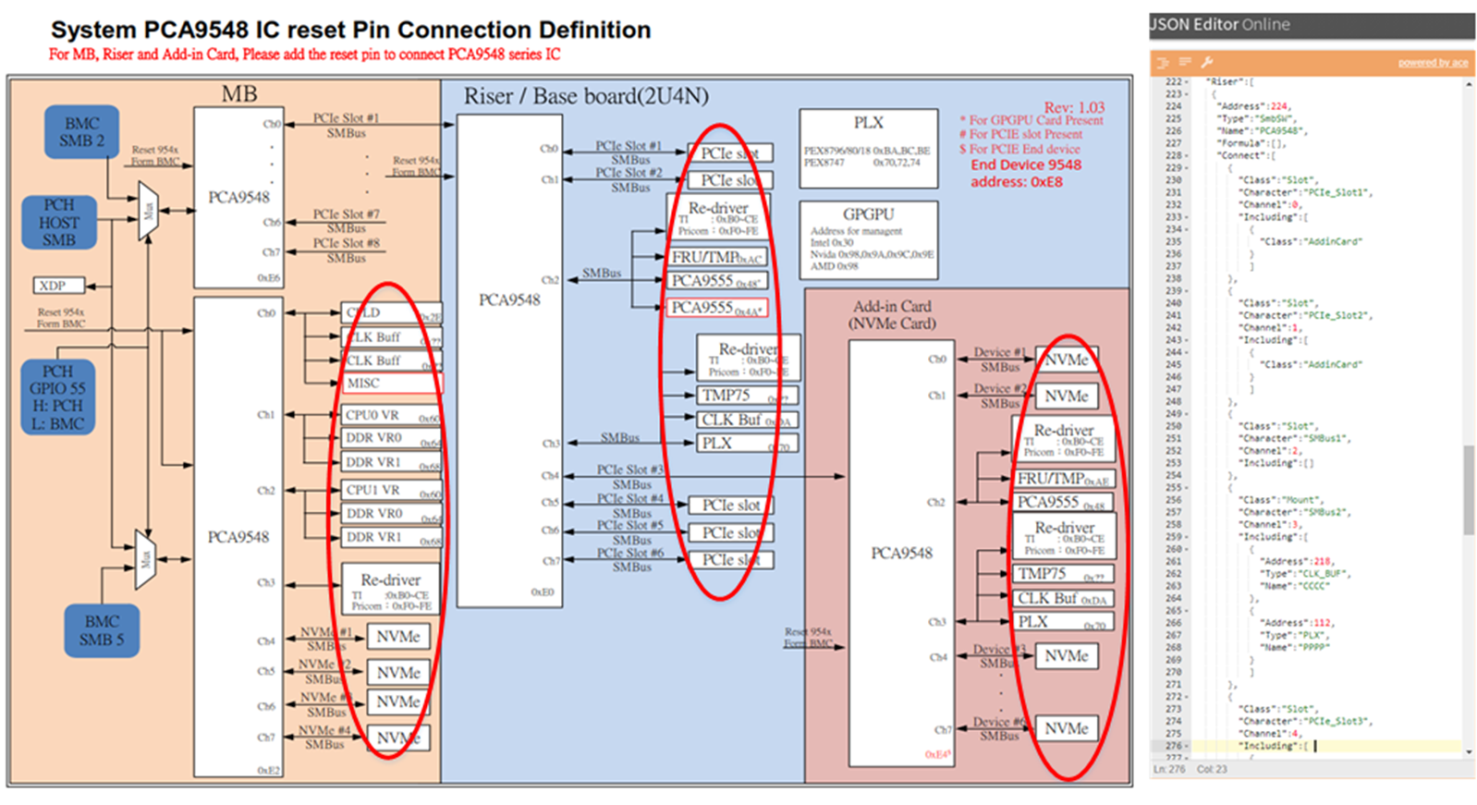


Where,

|  |  |
| --- | --- |
| **Title** | **Description** |
| Address | Smbus switch address in decimal ( 0xE2 -> 226) |
| Type | Always write “SmbSW”, or algorithm can’t recognize it. This item has to be a string. |
| Name | This value will show on screen after utility finish the scan. This item has to be a string. |
| Formula | Reserve. |
| Connect | Indicate the devices will hang behind this smbus switch. This item has to be an array data structure. |

**Step 3.**

**Add other smbus device which is hung behind the its own smbus switch. Follow definition of the data structure.**



Where,

|  |  |
| --- | --- |
| **Title** | **Description** |
| Class | The devices stay behind the smbus switch is “Slot” or “Mount” attribute.  “Slot” means this “Channel” is prepared for other module(ex: Riser or AddinCard)  “Mount” means these devices are exclusive from pluggable module.  This item has to be a string. |
| Character | The name that user defined. This item has to be a string. |
| Channel | Smbus switch channel. (Ex: PCA9548 has 8 channels, from 0~7). This item has to be a number. |
| Including@ | The detail information of smbus device that stay behind the smbus switch. This item has to be an array data structure. |
| @Address | Smbus device address. This item should declare in decimal. |
| @Type | Smbus device type. This value will show on screen after utility finish the scan. This item has to be a string. |
| @Name | Smbus device name. This value will show on screen after utility finish the scan. This item has to be a string. |

External File – SmbusDataSheet.json

This chapter will teach user how to transfer a **IC specification** into **External File(JSON format)**.

Remember to name the file as “SmbDatasheet.json” or utility can’t recognize the external file and function fail..

P.S. Recommend a website to user for creating and modifying a JSON file.

<https://jsoneditoronline.org/>

**Step 1.**

Classify the datasheet information into 4 groups.

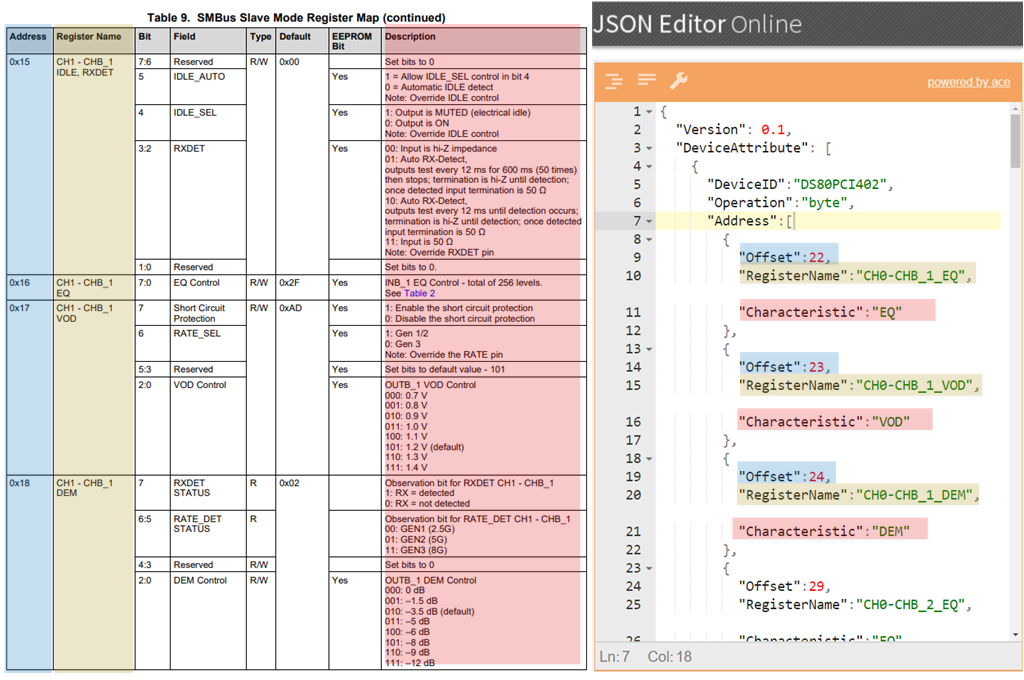


Where,

|  |  |
| --- | --- |
| **Title** | **Description** |
| DeviceAttribute | Contain the objects with datasheet information. This item has to be an array data structure. |
| **DeviceID** | The smbus device name. Which must be matched to the string of “Name” in GbtSmbusCommonDesign.json |
| Operation | Determine the smbus write data length which is byte, word or block format. |
| **Address** | The specification shows user where to control the modulation. User may setup an array later… |
| **Modulation** | The specification shows user what function(modulation) can this device support. User may create an array later... |

**Step 2.**

Transfer the Datasheet offset and register name into **Address** array.

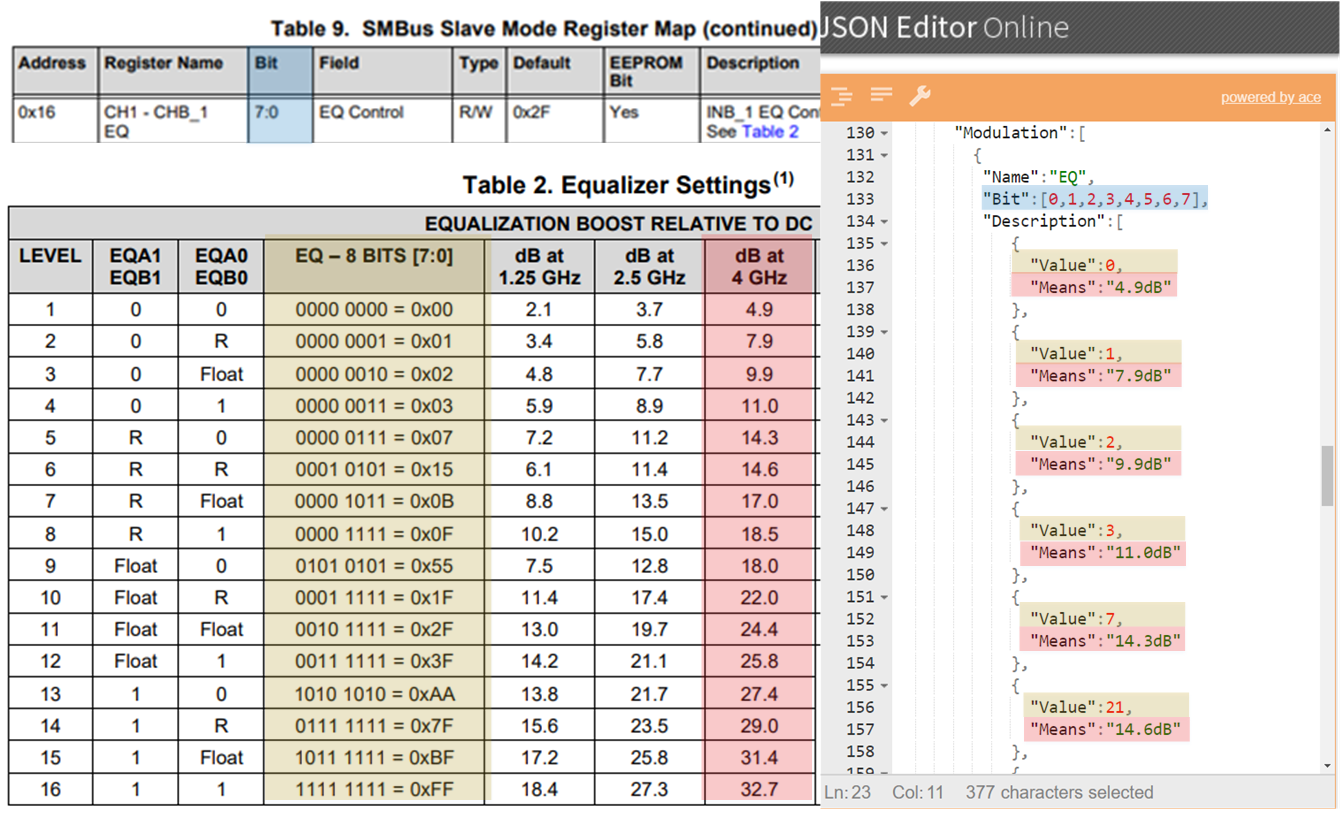


Where,

|  |  |
| --- | --- |
| **Title** | **Description** |
| Offset | Smbus device offset in decimal ( 0x16 -> 22) |
| RegisterName | Smbus device offset representation. This item has to be a string. |
| **Characteristic** | Smbus device offset meaning. This item has to be a string. This string should match to the string of “Name” defined in modulation array. |

**Step 3.**

Transfer the Datasheet register meaning into **modulation** array.



Where,

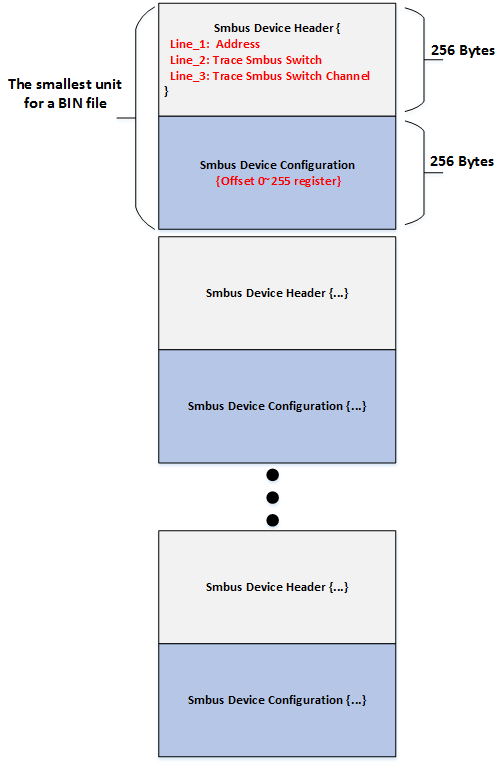
|  |  |
| --- | --- |
| **Title** | **Description** |
| Name | Smbus device modulation item. This item has to be a string. This string should match to the string of “Characteristic” defined in Address array. |
| Bit | Enumerate the number of bits that effect the this function |
| Description@ | Contain the objects with datasheet’s function information. This item has to be an array data structure. |
| Value | Based on “Bit” translate the value to “Means”. This item should be decimal value. |
| Means | The string stands for the “Value” by specification defined. This item should be string. |

Smbus Configuration BIN File

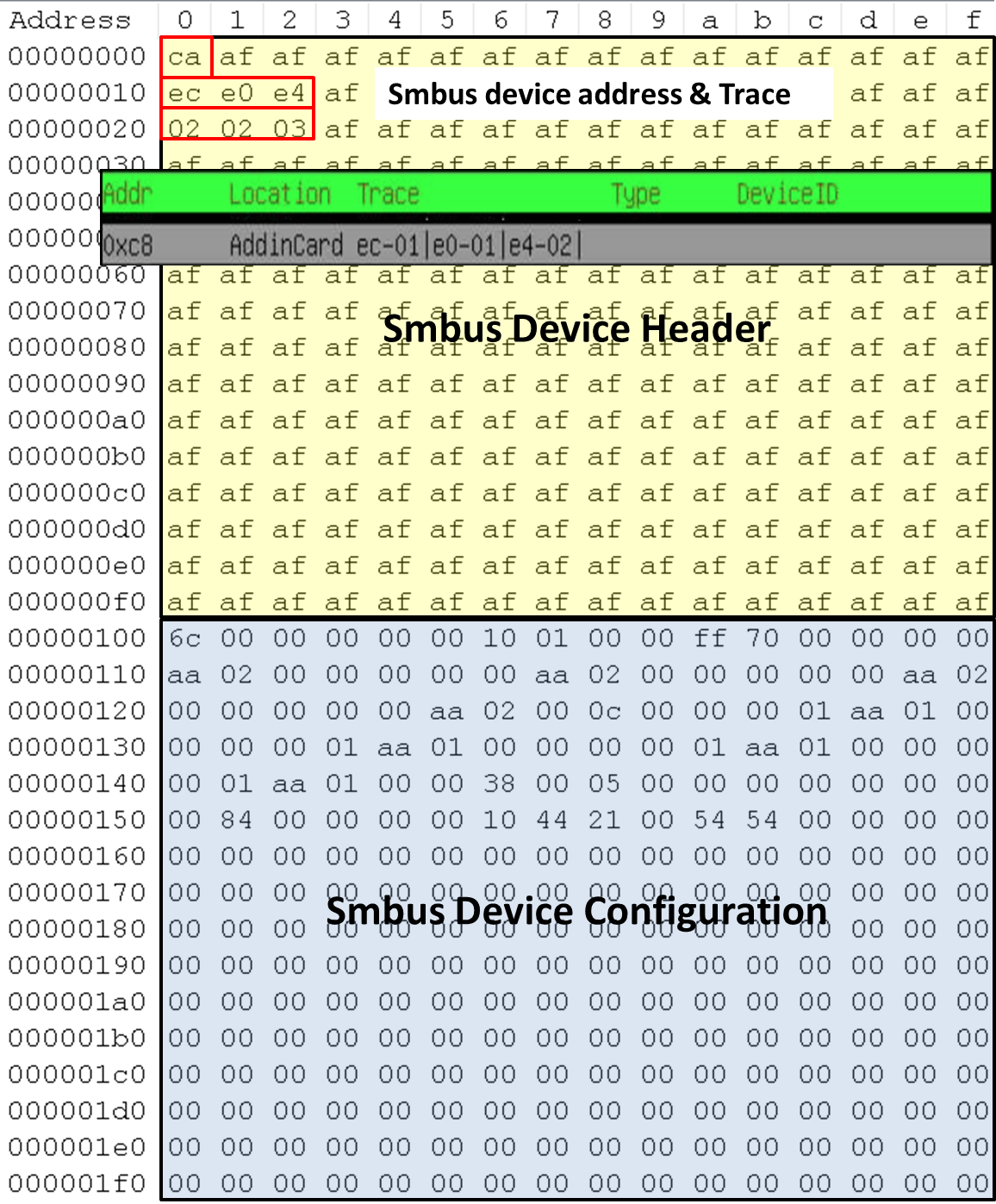
The BIN file not only read/write by utility, but also combine into ROM while User are building BIOS code. The classic cascade format make it convenient for user to handle smbus configuration and notice where this device locate.

In this chapter, User will learn the BIN file structure. Furthermore, user are going to realize how to modify BIN file without utility function, and let this file still workable.

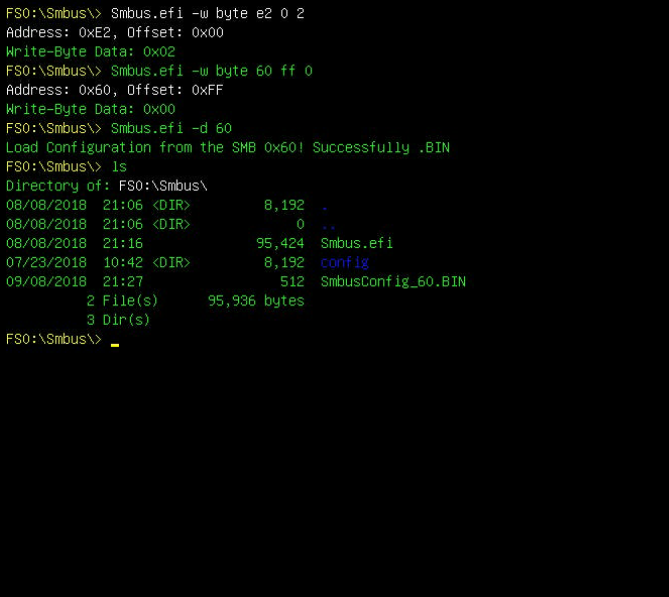
**The illustration diagram for cascade BIN file structure.**



**Take a view of raw data in 0xB0 BIN file**



[User Guide] How to Dump Register of Power Solution Chip on Purley Platform



Step 4

Step 2

Step 3

**Step 1.**

**Enable the PCH Smbus controller by GPIO tool (Please refer to GPIO user guide for detail information)**

**Step 2**

**Write Smbus switch chip 0xE2 by specific offset and data (refer to demonstration)**

**Step 3**

**Write Power solution chip 0x60 0x64 0x68 by specific offset and data (refer to demonstration), Configure the chip one by one until all tree power solution chips has been written**

**Step 4**

**Dump Power solution chip register as “configuration table” and save the result into binary file (refer to demonstration) Configure the chip one by one until all tree power solution chips’ configuration table has been dumped**

**(P.S. for Binary file meaning, please refer to chapter -- Smbus Configuration BIN File )**