Application Report

TAS2563 End System Integration Guide

Jin Gao

Abstract

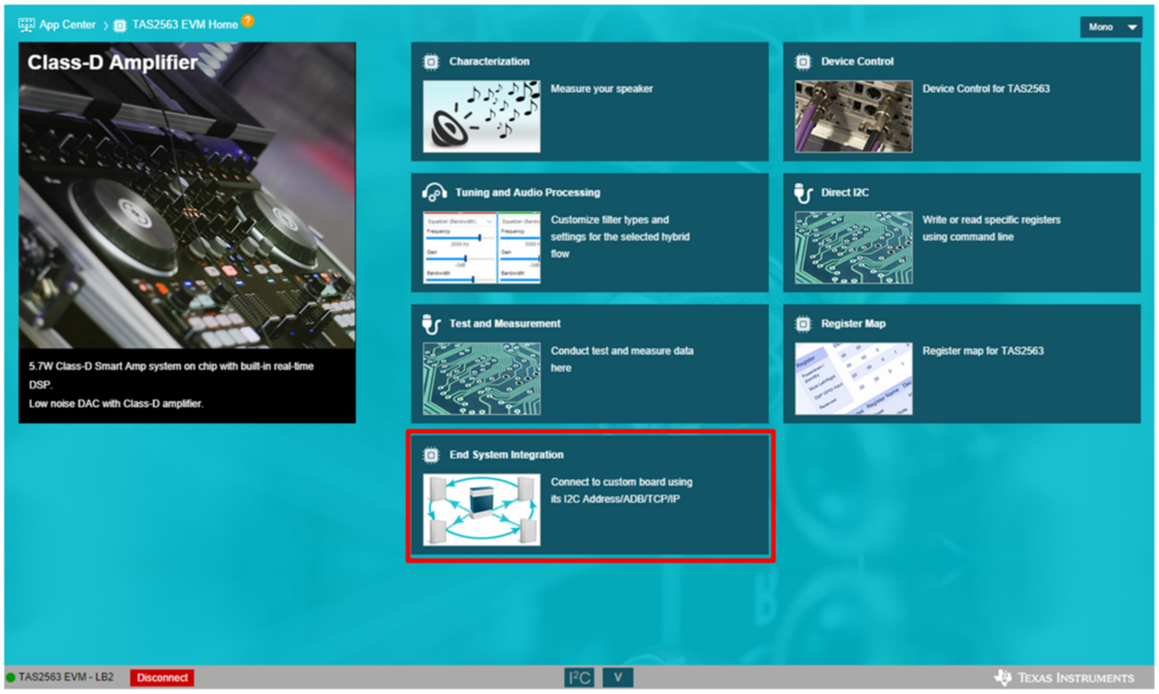
This document provides an overview of End System Integration and Factory Test Calibration Process for TAS2563 (6.1-W Boosted Class-D Audio Amplifier With Integrated DSP And IV Sense). It will cover binary and configuration files export in PPC3 and some of the common programming sequences for device driver in the end system.

Table of Contents

Trademarks

1 End System Integration

Binary and FTCFG files are the necessary files required by the driver for end system integration. These files contain all of the tuning, speaker characteristic data, and register data. To follow the implementation flowchart, binary and CFG files dumping should only be done after characterization, device configuration, and tuning. End system integration process can be accessed by selecting the "End System Integration" in the Device Home page.



5.7W Class-D Smart Amp system on chip with built-in real-time DSP.
Low noise DAC with Class-D amplifier.

Customize filter types and settings for the selected hybrid flow

Write or read specific registers using command line

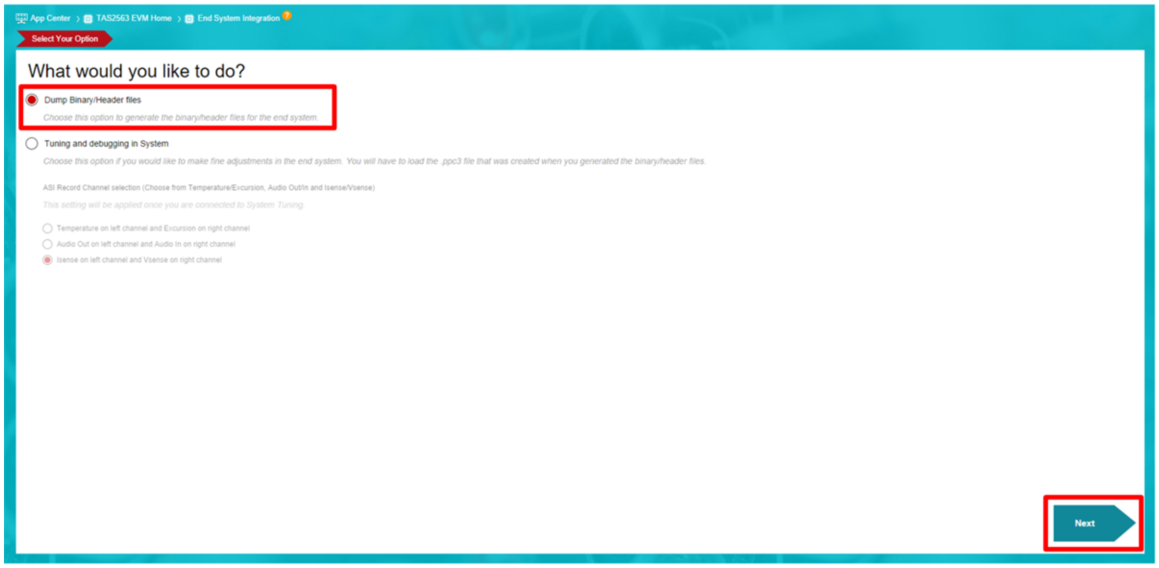
Conduct test and measure data here

Connect to custom board using its I2C Address/ADB/TCP/IP

Figure 1-1. Device Home

1.1 Step 1: Exporting Binary and Configuration Files

Select “Dump Binary/Header files” to generate binary file and select “Next” to continue. Note that “Tuning and debugging in System” option is available for users to tune and debug after TAS2563 is integrated in users' system.



What would you like to do?

[✓ Dump Binary/Header files]

Choose this option to generate the binary/header files for the end system.

Tuning and debugging in System

Choose this option if you would like to make fine adjustments in the end system. You will have to load the .ppc3 file that was created when you generated the binary/header files.

ASI Record Channel selection (Choose from Temperature/Excursion, Audio Out/In and Isense/Vsense)

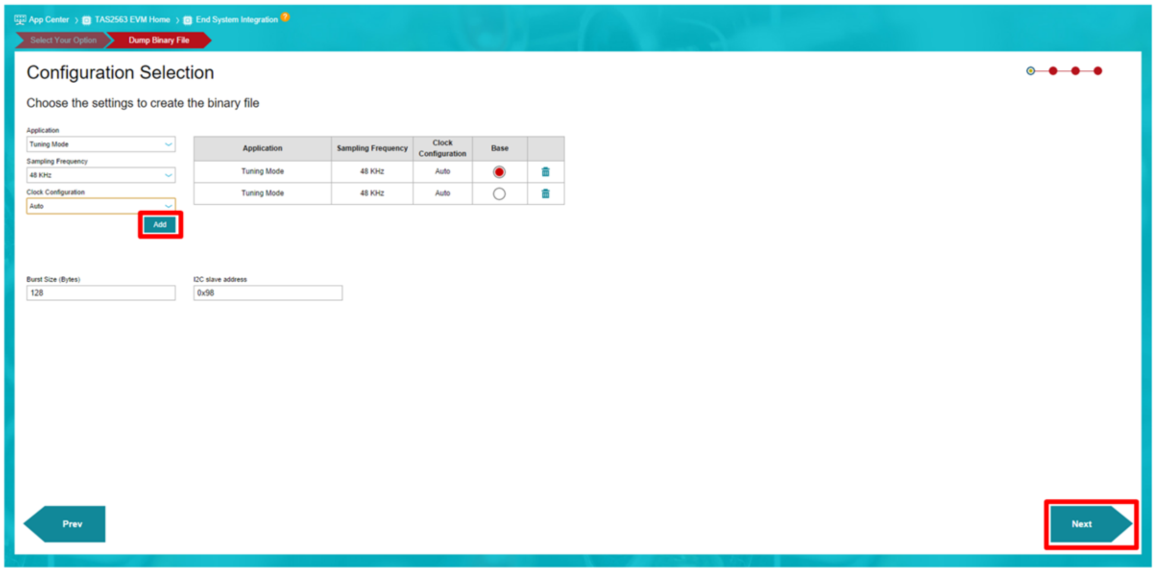
This setting will be applied once you are connected to System Tuning.

- Temperature on left channel and Excursion on right channel
- Audio Out on left channel and Audio In on right channel
- [✓ Isense on left channel and Vsense on right channel]

Figure 1-2. Dump Binary/Header Files

1.2 Step 2: Configuration Selection

Select the desired configurations for the binary file. Multiple configurations of the tuning data can be generated. For example, a tuning data for music playing and another for notification chime. Once it is complete, please select “Next” to continue.



Configuration Selection

Choose the settings to create the binary file

Application: Tuning Mode

Sampling Frequency: 48 KHz

Clock Configuration: Auto

Application	Sampling Frequency	Clock Configuration	Base	
Tuning Mode	48 KHz	Auto	●	🗑
Tuning Mode	48 KHz	Auto	○	🗑

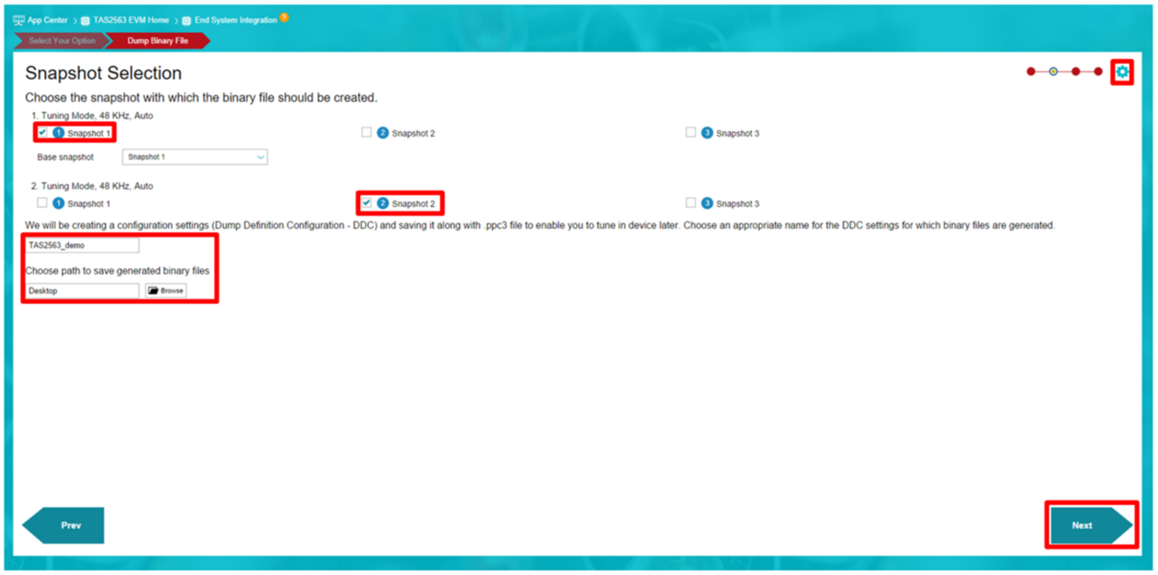
Burst Size (Bytes): 128

I2C slave address: 0x98

Figure 1-3. Binary File Configuration

1.3 Step 3: Snapshot Selection

Assign the tuning snapshot previously saved during “Tuning and Audio Processing” page to each configurations. Please name the file folder and assign the path to save the files. In addition, Various data can be read through the ASI Record Channel (shown in Figure 1-5) to suit the system application. The ASI Record Channel can be accessed by the gear icon in the top right corner. An example application of "Audio In & Out" in ASI Record Channel can be used for echo reference. Select “Next” to begin configuring the .ftcfg file for factory test calibration.



Snapshot Selection

Choose the snapshot with which the binary file should be created.

1. Tuning Mode, 48 KHz, Auto

[✓ Snapshot 1] Snapshot 2 Snapshot 3

Base snapshot: Snapshot 1

2. Tuning Mode, 48 KHz, Auto

Snapshot 1 [✓ Snapshot 2] Snapshot 3

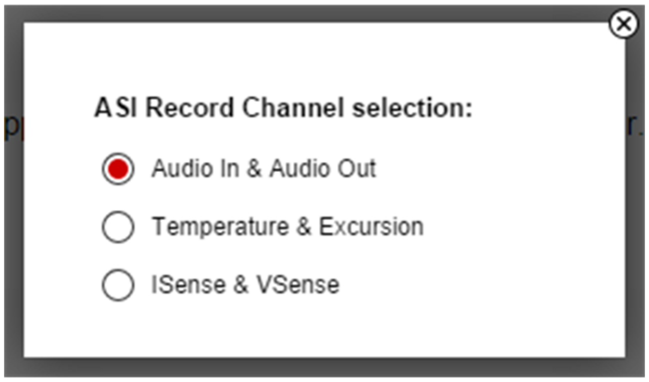
We will be creating a configuration settings (Dump Definition Configuration - DDC) and saving it along with .ppc3 file to enable you to tune in device later. Choose an appropriate name for the DDC settings for which binary files are generated.

TAS2563_demo

Choose path to save generated binary files

Desktop Browse

Figure 1-4. Snapshot Selection



ASI Record Channel selection:

[✓ Audio In & Audio Out]

Temperature & Excursion

ISense & VSense

Figure 1-5. ASI Record Channel Selection

Note

Steps 1 to 4 cover the settings for the binary file. Before users are able to export the binary file, please continue to the next section to complete the FTC configuration.

2 Factory Test and Calibration (FTC)

Speaker parameters can vary from the vendor datasheet with a tolerance, and each speaker may vary from each other as well. This can lead to inaccurate speaker protection when one speaker model is used to fit all speakers. For this reason TI's Smart Amp algorithms can be calibrated accordingly to each speaker variance to ensure proper and accurate function of speaker protection. The FTC is performed to calibrate the Smart Amp algorithm and protect the speaker from parameter variations. The FTC obtains the speaker Re, f_0, Q, and voice coil temperature as seen below in step three of FTC Process shown below.

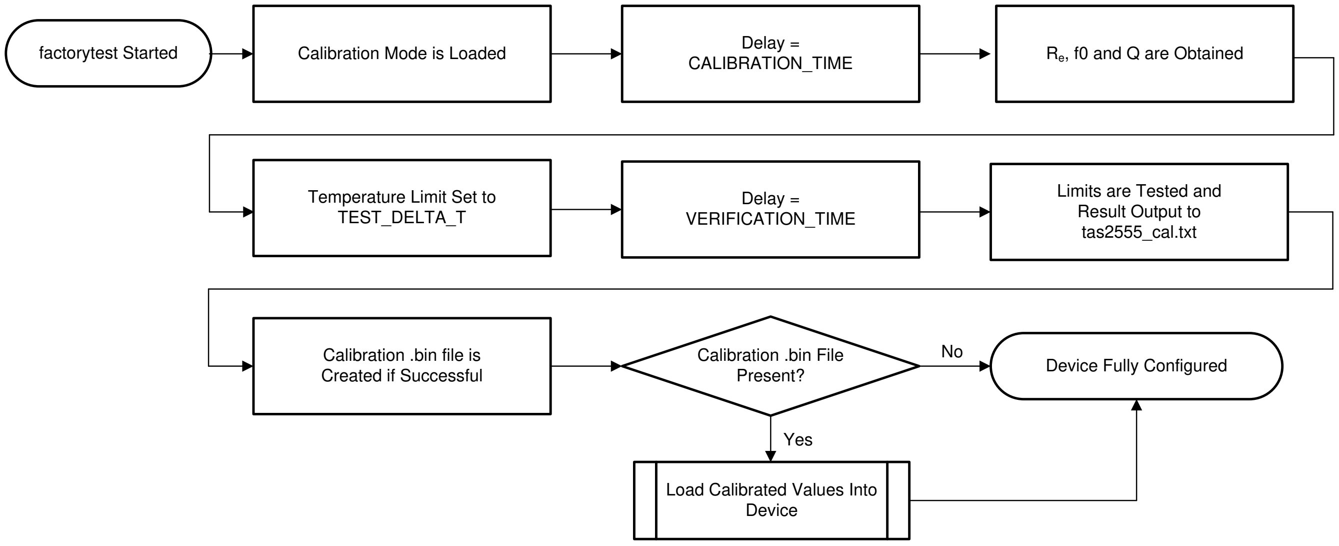
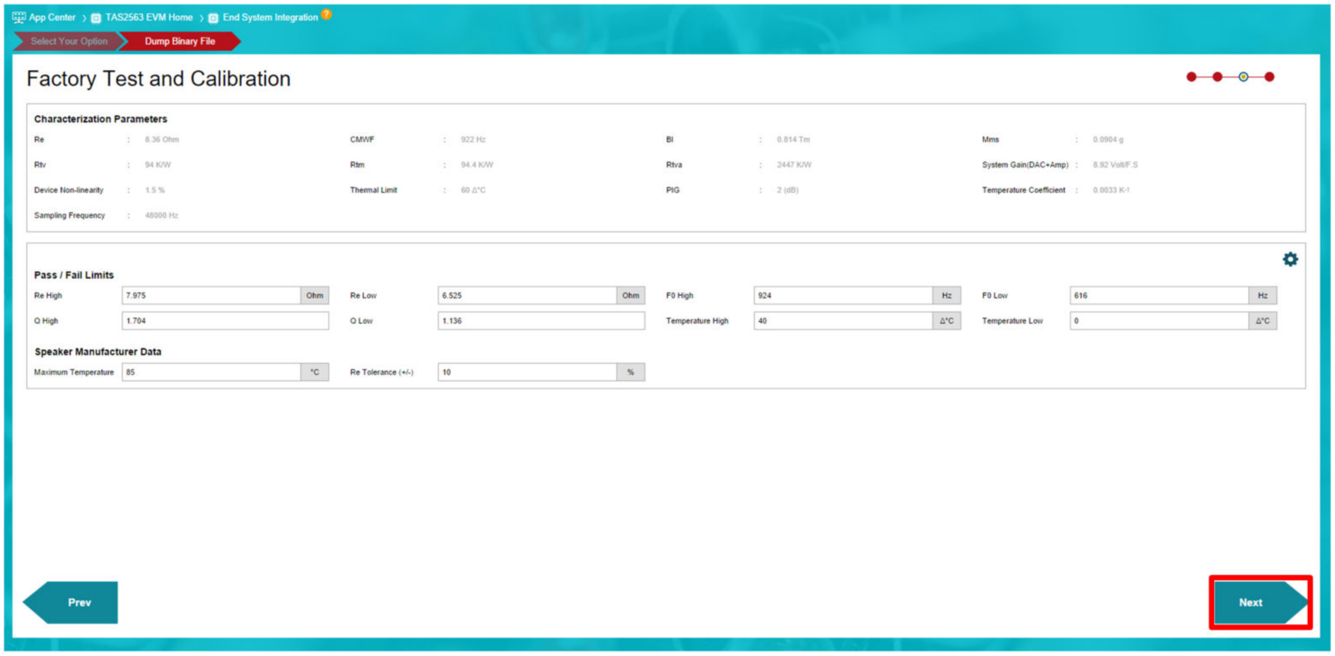

factorytest Started
Calibration Mode is Loaded
Delay = CALIBRATION_TIME
R_e, f0 and Q are Obtained
Temperature Limit Set to TEST_DELTA_T
Delay = VERIFICATION_TIME
Limits are Tested and Result Output to tas2555_cal.txt
Calibration .bin file is Created if Successful
Calibration .bin File Present?
No
Device Fully Configured
Yes
Load Calibrated Values Into Device

Figure 2-1. Factory Test Calibration Process

These parameters are then compared against pass/fail limits defined by the customer in Step 4. Speakers that pass the parameter limits test will be configured with new calibrated model as seen in the last step of Figure 2-1. TI recommends to perform FTC on all units in order to ensure proper protection of each speaker unit. For information on how to implement the FTC in your system please refer to the TAS2555, TAS2557, and TAS2559 Factory Test and Calibration Guide. Figure 2-2 contains all of the parameters the calibration test can measure during each test. By setting the range of tolerable pass/fail limits, users can determine if the speakers under testing are suitable for the system.

2.1 Step 4: Pass/Fail Limits

Define the Pass/Fail Limits of each parameter based on the speaker datasheet tolerance or on the application requirement. Select “Next” to generate the binary, ftcfg, cfg, and header files



Factory Test and Calibration

Characterization Parameters

Re	8.36 Ohm	CMWF	922 Hz	Bl	0.814 Tm	Mms	0.0904 g
Rtv	94 K/W	Rtm	94.4 K/W	Rtva	2447 K/W	System Gain(DAC+Amp)	8.92 Volt/F.S
Device Non-linearity	1.5 %	Thermal Limit	60 Δ°C	PIG	2 (dB)	Temperature Coefficient	0.0033 K-1
Sampling Frequency	48000 Hz						

Pass / Fail Limits

Re High	7.975 Ohm	Re Low	6.525 Ohm	F0 High	924 Hz	F0 Low	616 Hz
Q High	1.704	Q Low	1.136	Temperature High	40 Δ°C	Temperature Low	0 Δ°C

Speaker Manufacturer Data

Maximum Temperature	85 °C	Re Tolerance (+/-)	10 %

Figure 2-2. FTC Pass/Fail Limits

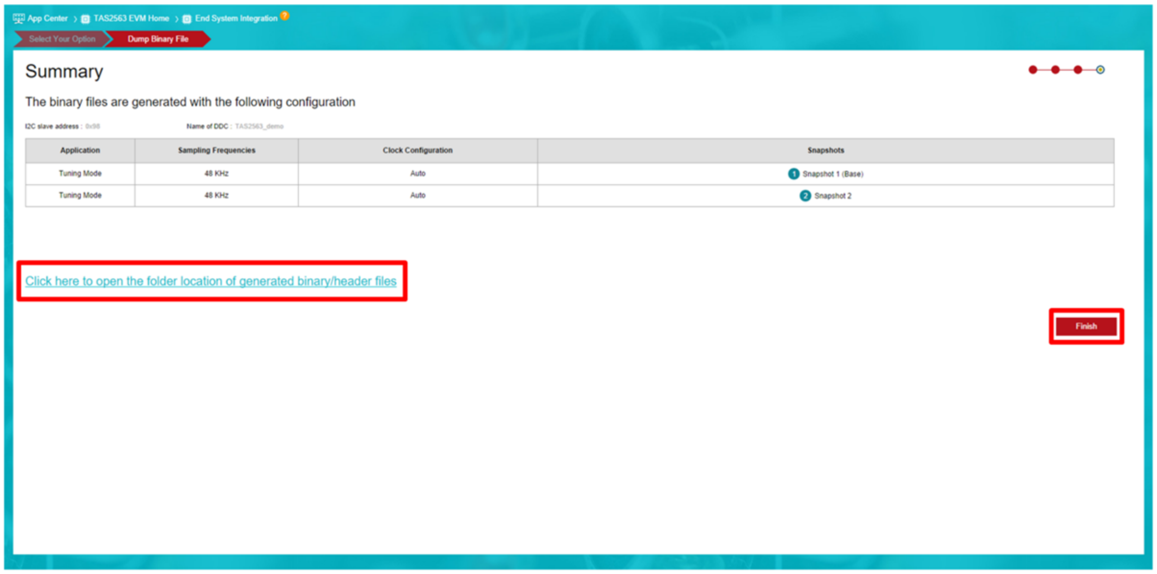
Note

In some cases, if the speaker parameter tolerances are not defined in the speaker datasheet, it is recommended to use Re variation as the limits for Re, and 6-Sigma deviation of Re variation as the variation limits for the rest of the speaker parameters

---

2.2 Step 5: Configuration Summary

This the summary page of the configuration. The binary, ftcfg, cfg, and header files should be available in the saved path. Select the highlighted text to open the file directory or “Finish” to exit the wizard.



Summary

The binary files are generated with the following configuration

I2C slave address : 0x98 Name of DDC : TAS2563_demo

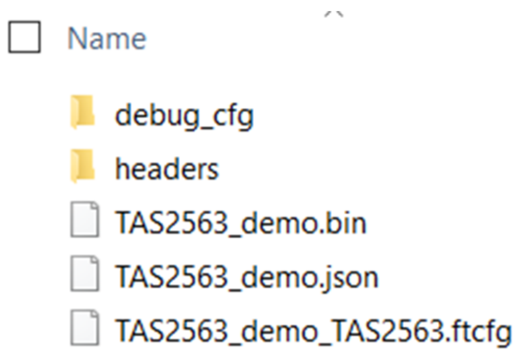
Application	Sampling Frequencies	Clock Configuration	Snapshots
Tuning Mode	48 KHz	Auto	1 Snapshot 1 (Base)
Tuning Mode	48 KHz	Auto	2 Snapshot 2

Click here to open the folder location of generated binary/header files

Figure 2-3. Files Export Completion Page

The folllowing figure shows the generated files from PPC3 end system integration dumping.

- All of the tuning, configuration, and characteristic data are stored in the binary file with extension .bin (e.g. TAS2563_demo.bin). The binary file will be used by the driver to initialize the speaker.
- The debug_cfg, json file, and headers folder contain the segmented tuning data in .cfg, .json, .h format, respectively, for necessary debugging purposes.
- The factory test calibration file with extension .ftcfg contains the pass/fail limit for model calibration of each speaker during the factory test phase.


Name

debug_cfg
headers
TAS2563_demo.bin
TAS2563_demo.json
TAS2563_demo_TAS2563.ftcfg

Figure 2-4. Generated Binary/Headers/CFG Files

For more information on the End System Integration process, users can also reference to TAS2555 End-System Integration Guide or TAS2555, TAS2557, and TAS2559 Factory Test and Calibration Guide . This mark the final step of the implmentation flowchart described in the TAS2563 Quick Start Guide

Once this step is complete, users are ready to integrate the Smart Amp with the driver. Proceed to Section 3 for driver integration.

3 Device Driver Integration

This section describes the sequence of programming by the device driver in the end system. The sequence of programming depends on the state in which the end system is currently in. Each state serves a specific purpose as listed below.

- Case 1: Initialization - Programming TAS2563 for the first time
- Case 2: Power up - Play audio content
- Case 3: New Configuration - Reconfigure TAS2563 with same program and same PLL
- Case 4: New Configuration and PLL - Reconfigure TAS2563 with same program and different PLL
- Case 5: New Configuration and Program - Reconfigure TAS2563 with different program

Example device driver code is available with TI. Users can use the example driver to port to their platform.

The dumped .bin file has various definitions (see the following code snippet).

```
#define TAS2563_BLOCK_PLL                    0x00
#define TAS2563_BLOCK_PGM_ALL                0x0d
#define TAS2563_BLOCK_PGM_DEV_A              0x01
#define TAS2563_BLOCK_PGM_DEV_B              0x08
#define TAS2563_BLOCK_CFG_COEFF_DEV_A        0x03
#define TAS2563_BLOCK_CFG_COEFF_DEV_B        0x0a
#define TAS2563_BLOCK_CFG_PRE_DEV_A          0x04
#define TAS2563_BLOCK_CFG_PRE_DEV_B          0x0b
#define TAS2563_BLOCK_CFG_POST               0x05
#define TAS2563_BLOCK_CFG_POST_POWER         0x06
```

Since TAS2563 supports different application (such as Mono and TAS2563 Dual-Mono system), it is possible that not all blocks are present in one application. If a block is not present, that step can be skipped.

For Section 3.1 through 3.5, please use the following annotation for device applications.

[A]: applies to device A

[B]: applies to device B

[AB] applies to both device A and B

3.1 Case 1 : Initialization

Programming TAS2563 for the first time

Use case: device initialization during system boot up

Status: DSP program memory and coefficient memory are empty. Assuming configuration 0 is the default tuning data.

Steps to playback:

1. Hardware reset the device
   - It is strongly recommended to hardware reset the device for reliable operation before initialization
2. Software reset the device
   - It is strongly recommended to software reset the device for reliable operation before initialization
3. Initialization
   - This code is not part of the binary file. The purpose of this step is to format ASI selection, IRQ configuration, and other relevant system configurations For both TAS2563 Mono and Stereo application:
   - [A] See TAS2563 Mono/Stereo Driver

   -
     ```
     int tas2563_load_default(struct tas2563_priv *pTAS2563)
     ```
   - [AB] See TAS2563 Mono/Stereo Driver

   -
     ```
     int tas2563_load_default(struct tas2563_priv *pTAS2563)
     ```
4. Download the Program
   a. [AB] Load *TAS2563_BLOCK_PGM_ALL* of configuration 0 in broadcasting mode
   b. [A] Load *TAS2563_BLOCK_PGM_DEV_A* of configuration 0
   c. [B] Load *TAS2563_BLOCK_PGM_DEV_B* of configuration 0
5. Download PLL
   - For TAS2563 Mono application:
     – [A] Load *TAS2563_BLOCK_PLL* of configuration 0
   - For TAS2563 Stereo application:
     – [AB] Load *TAS2563_BLOCK_PLL* of configuration 0
6. Download the pre-data of the configuration
   - [A] Load *TAS2563_BLOCK_CFG_PRE_DEV_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_PRE_DEV_B* of configuration 0
7. Download the coefficient
   - [A] Load *TAS2563_BLOCK_CFG_COEFF_DEV_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_COEFF_DEV_B* of configuration 0
8. Download calibration data if present
   - [A] Load *TAS2563_BLOCK_CFG_CAL_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_CAL_B* of configuration 0
9. Initialization Complete; Proceed to next step for Audio Playback
10. Feed the PLL clock. Audio stream may start any time after this step
11. [A/AB] Unmute; This code is not part of the binary file.

    -
      ```
      static int tas2563_mute(struct snd_soc_dai *dai, int mute)
      ```

Steps to sleep:

1. [AB] mute and shutdown TAS2563; This code is not part of the binary file.

-
  ```
  static int tas2563_mute(struct snd_soc_dai *dai, int mute)
  ```

2. Turn off PLL Clock

3.2 Case 2: Power up

Play audio content

Power up: TAS2563 has been programmed, and there is no changed needed for configuration

Use case: Power up device to play audio content

Status: DSP program memory and coefficient memory have been programmed. Assuming configuration 0 is the default tuning data.

Steps to playback:

1. Feed the PLL clock. Audio stream may start any time after this step.
2. [A/AB] Unmute; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```

Steps to sleep:

1. [AB] mute and shutdown TAS2563; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```
2. Turn off PLL Clock

3.3 Case 3: New Configuration

Reconfigure TAS2563 with same program and same PLL

Use case: Switch to another tuning (PPC3 snapshot) during music playback. Assuming the new configuration is X.

Status: DSP program memory has been programmed but coefficients need to be updated.

Steps to playback:

Note

Skip Step 3 and 4 if audio or music is playing. Stop after Step 2 if audio or music is not playing.

1. Download the coefficient
   - [A] Load *TAS2563_BLOCK_CFG_COEFF_DEV_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_COEFF_DEV_B* of configuration 0
2. Download calibration data if present
   - [A] Load *TAS2563_BLOCK_CFG_CAL_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_CAL_B* of configuration 0
3. Feed the PLL Clock. Audio stream may start any time after this step.
4. [A/AB] Unmute; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```

Steps to sleep:

1. [AB] mute and shutdown TAS2563; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```
2. Turn off PLL Clock

3.4 Case 4: New Configuration and PLL

Reconfigure tuning data and different PLL with the same program

Use case: Device is in playback with 48 kHz sample rate and needs to switch to music playback with 44.1 kHz sample rate. Assuming the new configuration index is X

Status: DSP program memory has been programmed but coefficients and PLL need to be updated.

Steps to playback:

Note

[AB] if the music is playing and the devices are running, load the mute sequence to shutdown. If the music is not playing, skip Step 1, 6, and 7

1. [AB] mute and shutdown TAS2563; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```
2. Download PLL
   - For TAS2563 Mono application:
     – [A] Load *TAS2563_BLOCK_PLL* of configuration 0
   - For TAS2563 Stereo application:
     – [AB] Load *TAS2563_BLOCK_PLL* of configuration 0
3. Download the pre-data of the configuration
   - [A] Load *TAS2563_BLOCK_CFG_PRE_DEV_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_PRE_DEV_B* of configuration 0
4. Download the coefficient
   - [A] Load *TAS2563_BLOCK_CFG_COEFF_DEV_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_COEFF_DEV_B* of configuration 0
5. Download calibration data if present
   - [A] Load *TAS2563_BLOCK_CFG_CAL_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_CAL_B* of configuration 0
6. Feed the PLL clock. Audio stream may start any time after this step
7. [A/AB] Unmute; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```

Steps to sleep:

1. [AB] mute and shutdown TAS2563; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```
2. Turn off PLL Clock

3.5 Case 5: New Configuration and Program

Reconfigure TAS2563 and reprogram the DSP

Use case: Change from speaker protection mode (tuning mode) to ROM1 mode for factory testing. Assuming the new configuration index is X.

Status: DSP program memory and coefficients needs to be reprogrammed.

Steps to playback:

1. [AB] mute and shutdown TAS2563; This code is not part of the binary file.

   -
     ```
     static int tas2563_mute(struct snd_soc_dai *dai, int mute)
     ```
2. Hardware reset the device
   - It is strongly recommended to hardware reset the device for reliable operation before initialization
3. Software reset the device
   - It is strongly recommended to software reset the device for reliable operation before initialization
4. Initialization
   - This code is not part of the binary file. The purpose of this step is to format ASI selection, IRQ configuration, and other relevant system configurations. For both TAS2563 Mono and Stereo application; This code is not part of the binary file:
   - [A] See TAS2563 Mono/Stereo Driver

   -
     ```
     int tas2563_load_default(struct tas2563_priv *pTAS2563)
     ```
   - [AB] See TAS2563 Mono/Stereo Driver

   -
     ```
     int tas2563_load_default(struct tas2563_priv *pTAS2563)
     ```
5. Download the Program
   a. [AB] Load *TAS2563_BLOCK_PGM_ALL* of configuration 0 in broadcasting mode
   b. [A] Load *TAS2563_BLOCK_PGM_DEV_A* of configuration 0
   c. [B] Load *TAS2563_BLOCK_PGM_DEV_B* of configuration 0
6. Download PLL
   - For TAS2563 Mono application:
     – [A] Load *TAS2563_BLOCK_PLL* of configuration 0
   - For TAS2563 Stereo application:
     – [AB] Load *TAS2563_BLOCK_PLL* of configuration 0
7. Download the pre-data of the configuration
   - [A] Load *TAS2563_BLOCK_CFG_PRE_DEV_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_PRE_DEV_B* of configuration 0
8. Download the coefficient
   - [A] Load *TAS2563_BLOCK_CFG_COEFF_DEV_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_COEFF_DEV_B* of configuration 0
9. Download calibration data if present
   - [A] Load *TAS2563_BLOCK_CFG_CAL_A* of configuration 0
   - [B] Load *TAS2563_BLOCK_CFG_CAL_B* of configuration 0
10. Initialization Complete; Proceed to next step for Audio Playback

Note

If the music is playing, continue to Step 12. If the music is not playing, stop at Step 10

11. Feed the PLL clock. Audio stream may start any time after this step
12. [A/AB] Unmute; This code is not part of the binary file.

    -
      ```
      static int tas2563_mute(struct snd_soc_dai *dai, int mute)
      ```

Steps to sleep:

1. [AB] mute and shutdown TAS2563; This code is not part of the binary file.

    -
      ```
      static int tas2563_mute(struct snd_soc_dai *dai, int mute)
      ```
2. Turn off PLL Clock

4 Summary

This document serves as an overview of the End System Integration specifically for TAS2563. The End System Integration is the last step of the whole development process; however, PPC3 also offer “Tuning and debugging in System” option for direct tuning and debuggin after the device is integrated with the host system. For more information about the development process please refer to the TAS2563 Product page, TAS2563 Quick Start Guide, or visit us on the E2E Forum.