Application Note

TPS6594/LP8764 PMIC Application Guide (Part 4)

Lijia Zhu

ABSTRACT

This document was translated from a simplified Chinese source. (ZHCAFE9)

With the rapid development of new energy vehicles, automobiles are transitioning from electrification to intelligence. As high-level assisted driving quickly lands across major OEMs and Tier 1 suppliers, TDA4 is being widely used in various terminal applications such as ADAS domain controllers, body domain controllers, and LiDAR. TI's new generation PMIC family, represented by TPS6594/LP8764, is not only the optimal power solution for TDA4 SOCs, but also an excellent choice for powering other models of SOCs. TPS6594/LP8764 features high integration, high scalability, and support for high-level functional safety. Because it has many functions and is relatively complex, there will be considerable difficulty in application. This series of articles will share insights from aspects such as the main internal mechanisms of the TPS6594/LP8764 PMIC chips, system design considerations, common problem troubleshooting ideas, and custom PMIC firmware (NVM). This article is the fourth piece in the series of articles, discussing the commonly used problem locating ideas for PMIC chips.

Table of Contents

List of Figures

1 Introduction

TPS6594/LP8764 has rich functional safety mechanisms, which leads to strict external conditions required for the normal operation of the PMIC. Even for the simplest problem of being unable to start up, its root cause could be varied and diverse, and even one problem can mask another problem, and several problems coupling together to take effect is also commonplace, making location harder and harder, requiring engineers to step by step peel off the interference items to finally find the root cause of the problem.

According to the macro behavior of the PMIC, this article summarizes the encountered problems into the following three categories:

1. The PMIC cannot power up, with some power rails or none of the power rails attempting to start.
2. The PMIC powers up normally but then powers down due to unknown reasons.
3. The PMIC cannot start again after power-off.

This article introduces the ideas when locating PMIC system problems from several aspects: hardware preliminary screening, register information auxiliary locating, and changing NVM to eliminate interference factors to hit the root cause directly.

2 Preliminary Troubleshooting of Problems

TPS6594 has a VSYS control mechanism, and no matter what kind of problem it is, confirming whether the VCCA power supply of the PMIC exists first is very key. This can be verified by measuring the VINTLDO (pin 2) of TPS6594 and VINTLDO (pin 20) of LP8764 after the system powers up. If a stable 1.8V voltage signal exists on VINTLDO, it indicates that the NMOS between VSYS and VCCA opens normally and the internal digital state machine of the PMIC works normally, and location can be conducted according to the content of subsequent chapters.

If the VINT pin has no voltage, it indicates that the NMOS does not open normally, and generally speaking there are three possibilities:

1. In the entire process from power-up to power-down, VSYS or VCCA triggered OVP and did not drop below the reset level.
2. VCCA failed the Fail Short BIST due to reverse leakage.
3. The chip is damaged, there is a welding problem, or the NMOS is open-circuit. The damaged part can be determined by observing the relationship among VSYS, OVPDRV, and VCCA with an oscilloscope. If the NMOS is damaged and a replacement part cannot be found temporarily, the DS of the MOS can be shorted and the chip VSYS SENSE can be grounded to bypass the VSYS SENSE mechanism.

3 Measuring Waveforms with Oscilloscope to Locate Faults

Under the condition of determining that VCCA exists, registers can be read to judge the location where the fault occurs, but reading registers requires building a software and hardware environment. Before building the register reading environment, an oscilloscope can be used to measure all power rail outputs to conduct a preliminary judgment on the fault, and at the same time, the waveform is also a very good auxiliary for locating the root cause.

Aiming at the first type of problem, testing the outputs is not done by measuring BUCK1-5 and LDO1-4 one by one. A more time-saving practice is to measure one by one from front to back according to the sequence of the power-up sequence of the current PMIC power solution, because a fault occurring in a power rail that is earlier in the power-up sequence will prevent the subsequent power rails from opening. Figure 3-1 shows the power-up sequence of PDN-1A (TPS65941213+LP87611B4), which can be obtained by searching the keyword "PDN 1A" on TI's official website, and other PDNs can also be obtained by searching the corresponding names. In addition to the power-up and power-down sequence, the entire PFSM setting is written very clearly in the PDN file, so it must be read carefully before locating problems.

Here, PDN 1A is used as an example for illustration. When a board using this PDN has a failure of being unable to start up, the waveforms that might be measured may have the following several situations:

1. Power rails #1 and #2 have completely no output and are always at low level, indicating that the problem occurs before the power-up sequence, and a relatively large possibility is that the SPMI BIST failed, and the waveform of SPMI can be checked to see if it is normal.
2. Power rails #1 and #2 can see the complete 15 times of restarting process, which is the manifestation of the state machine entering SAFE RECOVERY 15 times and finally the recover counter reaching the upper limit and stopping at the SAFE RECOVERY state. On the waveform, 15 pulses with pulse widths greater than 1.7ms can be seen, and power rails #3 and #5 can see 15 pulses. Although the pulse width is relatively short, the tops of the pulses all reached reasonable and expected voltages. While power rail #4 did not reach the expected height, and Figure 3-2 is such an example, where the 15 pulses here only reached 200mV, indicating that this voltage is short-circuited, and the reason for the short circuit remains to be further investigated.

#	Resource	PMIC	Delay Diagram	Total Delay	Rail Name
1	GPIO9	TPS65941213-Q1		0 us	Undefined
2	GPIO10	LP876411B4-Q1		0 us	EN_3V3IO_LDSW
3	LDO1	TPS65941213-Q1		1700 us	VDD1_LPDDR_1V8
4	LDO4	TPS65941213-Q1		1700 us	VDA_LN_1V8
5	BUCK5	TPS65941213-Q1		1700 us	VDD_PHY_1V8
6	BUCK123	TPS65941213-Q1		2700 us	VDD_CPU(AVS)
7	LDO3	TPS65941213-Q1		2700 us	VDA_DLL_0V8
8	BUCK1234	LP876411B4-Q1		2700 us	VDD_CORE_0V8
9	UCK3 Monitor	TPS65941213-Q1		3700 us	VCCA_3V3
10	BUCK4	TPS65941213-Q1		3700 us	VDD_MCU_0V85
11	LDO2	TPS65941213-Q1		3700 us	VDD_IO_1V8
12	GPIO1	LP876411B4-Q1		3700 us	EN_VDDR
13	nRSTOUT	TPS65941213-Q1		12700 us	H_MCU_PORz_1V8
14	RSTOUT_SOC	TPS65941213-Q1		12700 us	H_SOC_PORz_1V8

Figure 3-1. PDN 0A Power-up Sequence

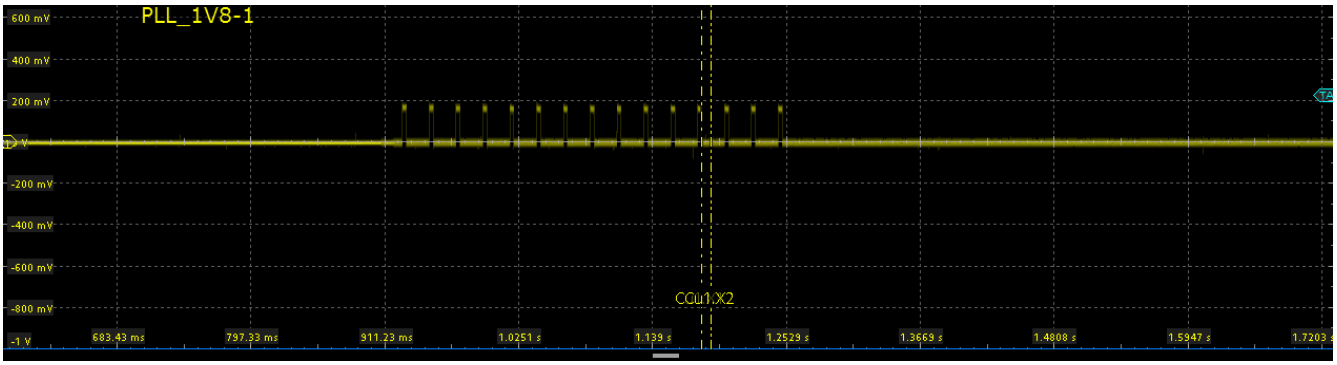

Figure 3-2. LDO Short Circuit Power-up Waveform

3. The second point inside is just an example, and situations where other power rails have problems are also possible. The idea lies in troubleshooting sequentially from front to back until the power rail that does not start normally is seen.

For the second type of problem, it is recommended to prioritize looking at registers, and determine the faulty power rail or other mechanisms based on the register results. Troubleshooting one by one with an oscilloscope will be relatively time-consuming.

For the third type of problem, it is also monitored one by one according to the power-up sequence just like the first type, to see which power rail blocked the restart. Generally, there is residual voltage on a certain power rail.

4 Using PMIC EVM to Locate Faults

4.1 Using EVM to read I2C information of the faulty board

Obtaining the register information of the PMIC is very key during the debugging process. However, because the PMIC has a large number of functional safety mechanisms, it leads to the PMIC being very easy to trigger protection and close outputs. At this time, the board-mounted SOC has no power supply and cannot obtain the register information of the PMIC through the board-mounted SOC. An independent USB-I2C protocol conversion chip (such as FT2232) can be used, cooperating with self-owned host computer software to read the register information of the PMIC, but it requires extra R&D cost to debug this set of software and hardware tools. A simpler method is to use the EVM of the TPS6594/LP8764 PMIC cooperating with the host computer software Scalable PMIC GUI to read the registers of the PMIC.

4.1.1 EVM Hardware Introduction

The photo of the TPS6594 EVM is as shown in Figure 4-1. The two most critical devices on the board are the PMIC on the front side and the MSP432 MCU on the back side. The MCU and the PMIC's I2C are interconnected together, with the MCU acting as the I2C master device and the PMIC acting as the I2C slave device. The PC communicates with the MCU through the USB interface, and the MCU forwards the commands sent from the PC to the I2C interface to complete the communication between the PC and the board's PMIC. The MCU can read the I2C of the PMIC on the EVM, and can also read the PMIC I2C information of the external board to be tested through the method of jumpers.

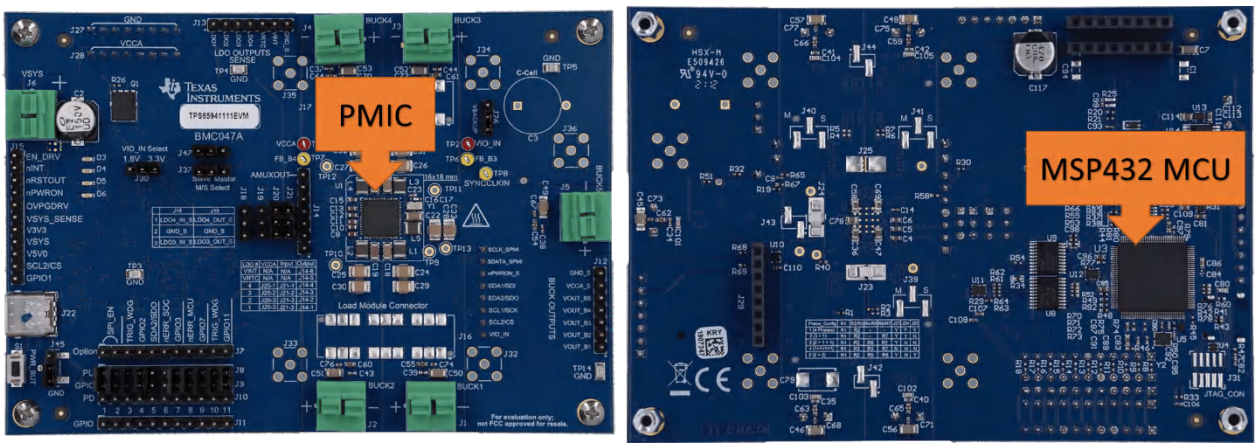

Figure 4-1. TPS6594 EVM Photo

4.1.2 Hardware Connection

The system for accessing the external board to be tested through the EVM is shown in Figure 4-2, and the building method is introduced in detail in the subsequent content of this section.

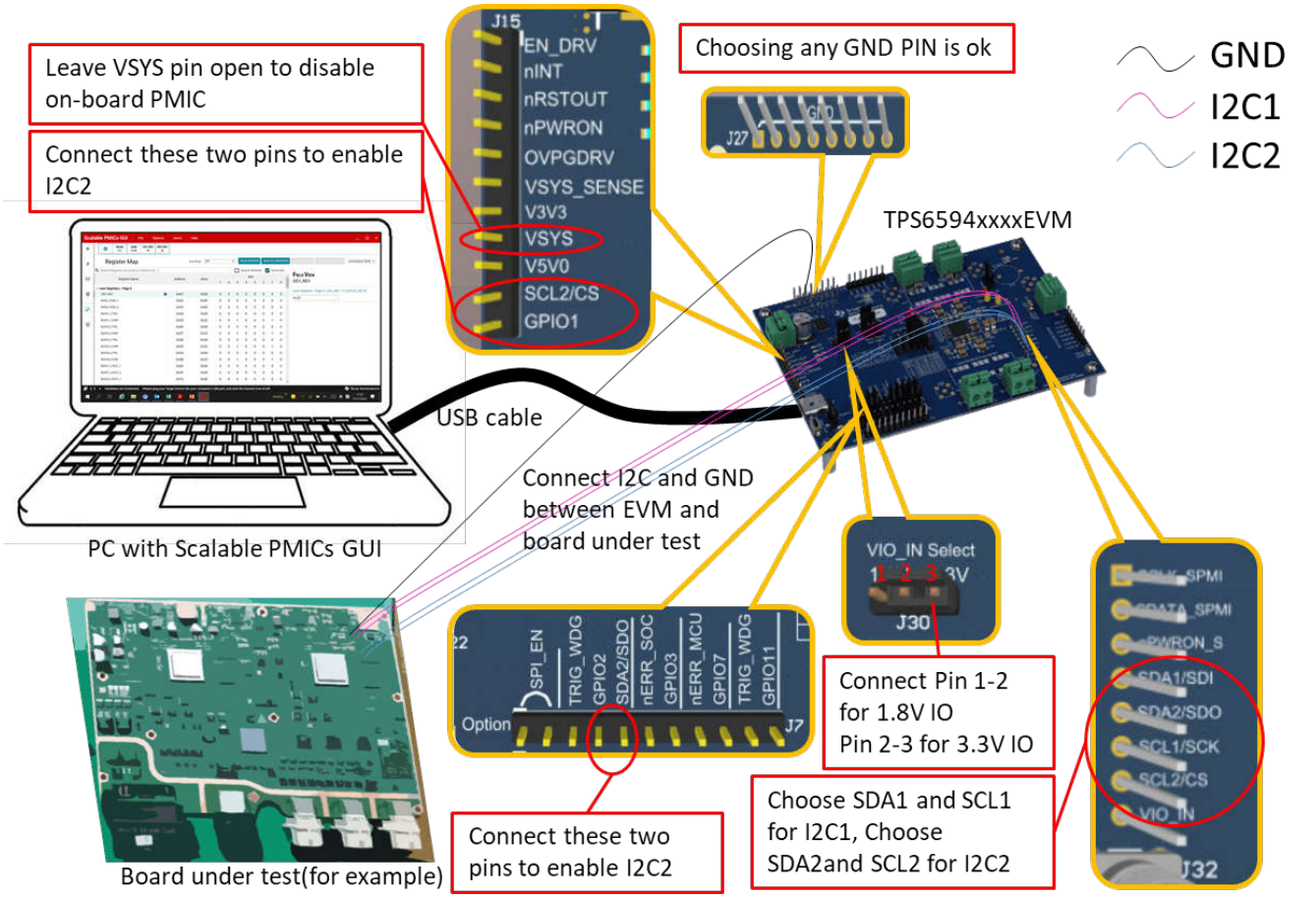

Figure 4-2. Schematic Diagram of Using EVM to Read PMIC Registers of the Board to be Tested

1. In order to avoid the I2C address of the existing PMIC chip on the EVM board and the PMIC I2C address on the external board to be tested from conflicting (using the same address), it needs to ensure that the VSYS pin of J15 on the EVM is in a floating state. If VSYS and V3V3 are shorted (for PMIC models with 5V Vin, it is VSYS and V5V0 shorted), it will enable the PMIC on the EVM, causing an I2C address conflict and reading wrong information.
2. Use jumpers to connect the SDA1/SDI and SCL2/SCK pins of the EVM to the SDA and SCL of I2C1 on the board to be tested. Use a third jumper to make the two boards share a common ground.
3. The I2C on the EVM comes with its own 1k pull-up resistor, and there are two pull-up voltage values of 1.8V and 3.3V available for choice, switched by switching the state of the jumper cap on J30. If the board to be tested can start normally, pin 2 of J30 can be floated, and at this time the I2C pull-up resistor on the board to be tested is used to provide the pull-up voltage. If the board to be tested cannot start or cannot determine the state (probabilistic startup), at this time the board to be tested cannot provide the required I2C pull-up voltage, and it needs to plug on the jumper cap on J30 to make the pull-up voltage value output by the EVM consistent with the I2C pull-up voltage designed on the board to be tested. Selecting the wrong voltage in this step may lead to damaging the hardware. Locating problems encounters the second situation more, therefore this step suggests the method of plugging on the J30 jumper cap, and the universality will be a bit better.
4. The PC and the EVM are connected through a USB A to C data cable, and the USB supplies power to the EVM. Other than this, the EVM does not need extra power supply. Generally speaking, it needs to first supply power to the board to be tested, and then supply power to the EVM, otherwise the system will fail to start up with a high probability. The reason analysis is shown in Figure 4-3, divided into two situations (a) and (b) for discussion. For situation (a), the I2C of the board to be tested is pulled up to the external switch output controlled by the PMIC. When the EVM is powered up first, the I2C signal pin obtains voltage first, and through the parasitic body diode of the external load switch, as the path shown by the red dashed arrow, flows back to the input end of the PMIC, leading to the Fail Short BIST failure, the NMOS not opening, and

the chip to be tested being in a shutdown state. At this time, the PMIC on the board to be tested cannot be accessed through I2C. For situation (b), the I2C of the board to be tested is pulled up to a certain output of the PMIC. If the EVM is powered up first at this time, the voltage will flow back to the output end of the PMIC through the path of the red dashed arrow, blocking the residual voltage detection mechanism and leading to being unable to start. At this time, the I2C on the board to be tested can work, but the PMIC has no output (the macro fault phenomenon to be located may also be like this), and at this time, because of reading wrong register information (for example, the rail currently connected to the I2C pull-up reports an SC fault), it will lead the locating direction of the problem astray, so millions of times must pay attention!

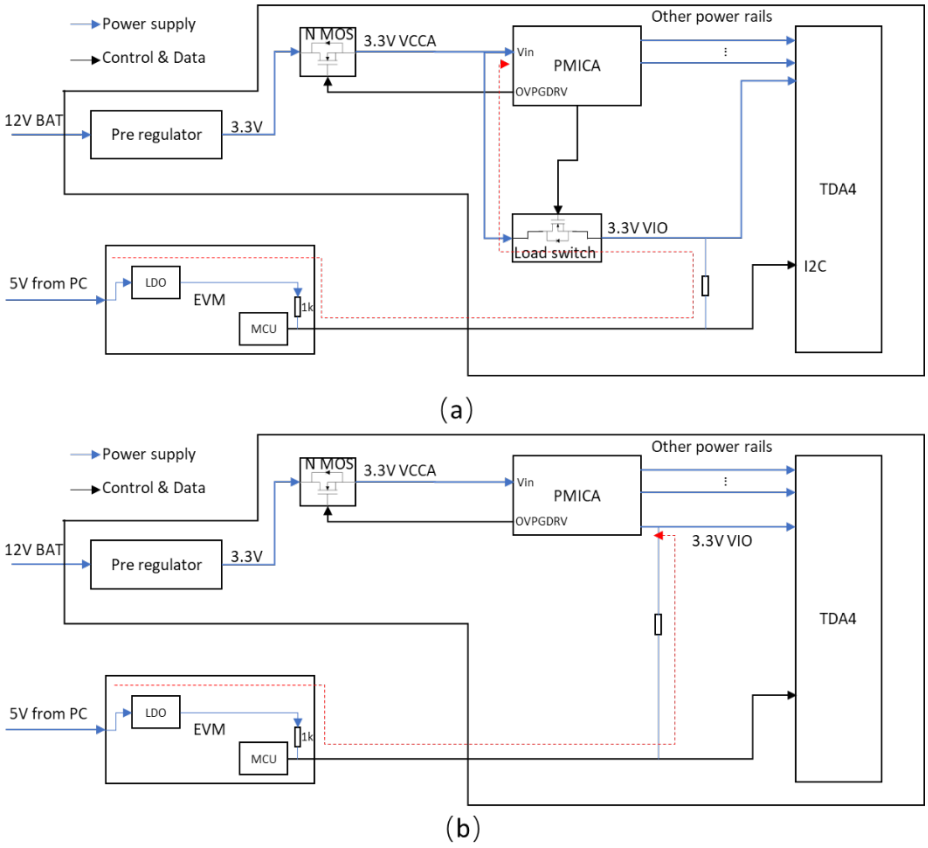

Figure 4-3. Schematic Diagram of I2C Path Current Backflow Mechanism

In the actual process of locating problems, there is also the requirement of restarting the board to be tested multiple times to reproduce probabilistic problems. In this process, plugging and unplugging the EVM board every time is a relatively troublesome thing. At this time, the method of removing the pull-up resistors of SCL and SDA on the board to be tested can be used to block the reverse leakage path.

5. Some PMIC models enabled I2C2, and at this time the register access permission of page 4 switches from I2C1 to I2C2. If only connecting I2C1, the software reading all registers will have the error message as shown in Figure 4-4. The registers of Page 4 are settings related to the watchdog. If the problem to be located has nothing to do with the watchdog, this alarm can be ignored. If wishing to conduct access to these registers, it needs to short the SCL2 and GPIO1 pins on J15, and the GPIO2 and SDA2 pins on J7.

Figure 4-4. Software Error Message When I2C2 is Not Connected

4.1.3 Software Installation and Simple Use

Scalable PMIC GUI software download address: https://dev.ti.com/gallery/view/PMIC/Scalable-PMICs-GUI/ver/3.0.0/. There are a total of two files, the software body and runtime, that need to be downloaded. Note to download the correct version according to one's own computer operating system. When installing, install the runtime first and then install the software body.

1. After the hardware completes the connection, after opening the software, the interface is as shown in Figure 4-5, and one can directly click the register map button on the top right corner of the corresponding PMIC model item to enter the register details page.

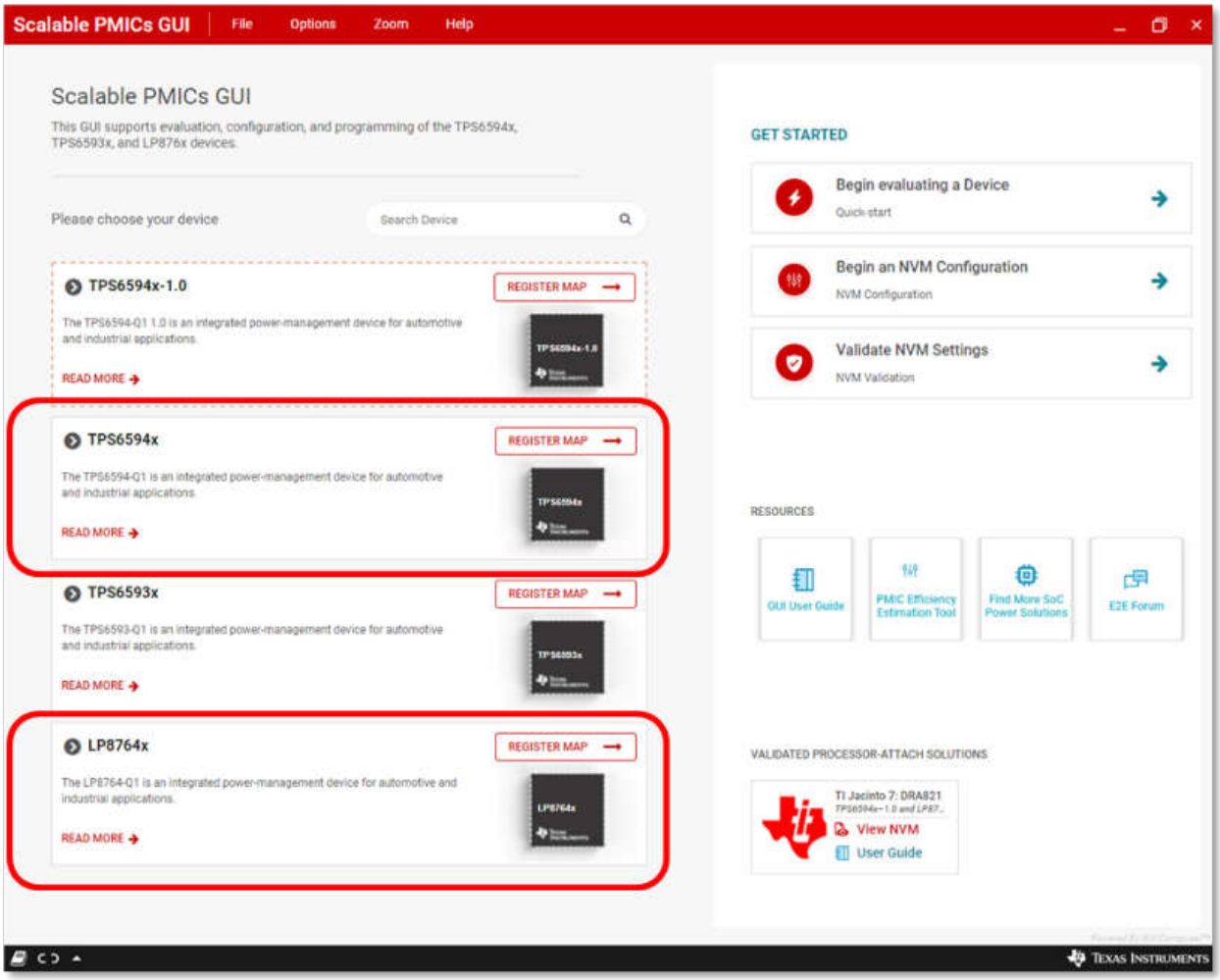

Figure 4-5. Software Error Message When I2C2 is Not Connected

2. After entering the register reading page, the software will attempt to connect the PMIC at the 0x48 address. Successful connection will display in the status bar below: "Hardware connected", as shown in Figure 4-6. After I2C communication is established, one can click the READ ALL REGISTERS button to manually refresh all register data. If it displays "Connect to AEVM Controller but failed to connect to device xxxxx (device model) on DUT with I2C @ 0x48", then it indicates that the communication between the MCU and the PC is normal, but the MCU cannot establish I2C communication with the PMIC of this address 0x48. It is possible that the PMIC with the address of 0x48 indeed does not exist in the current system or the hardware connection is unstable and did not connect the PMIC of the 0x48 address. If needing to switch the I2C address, one can click the gear mark on the top left corner of the page to open the setting pop-up window to conduct change, and the pop-up window content is as shown in Figure 4-7. It needs to be noted that multiple PMICs on the same board have different I2C addresses and their respective different device models, and it needs to select options in the two drop-down boxes of select device and I2C1 Address to the correct settings. If it displays "Hardware not connected, please plug your target device into your computer's USB port and click the connect icon at left", it indicates that the connection between the PC and the EVM has a problem, and it needs to check whether the EVM is damaged or the software driver is installed normally, and one can restart the computer to try.

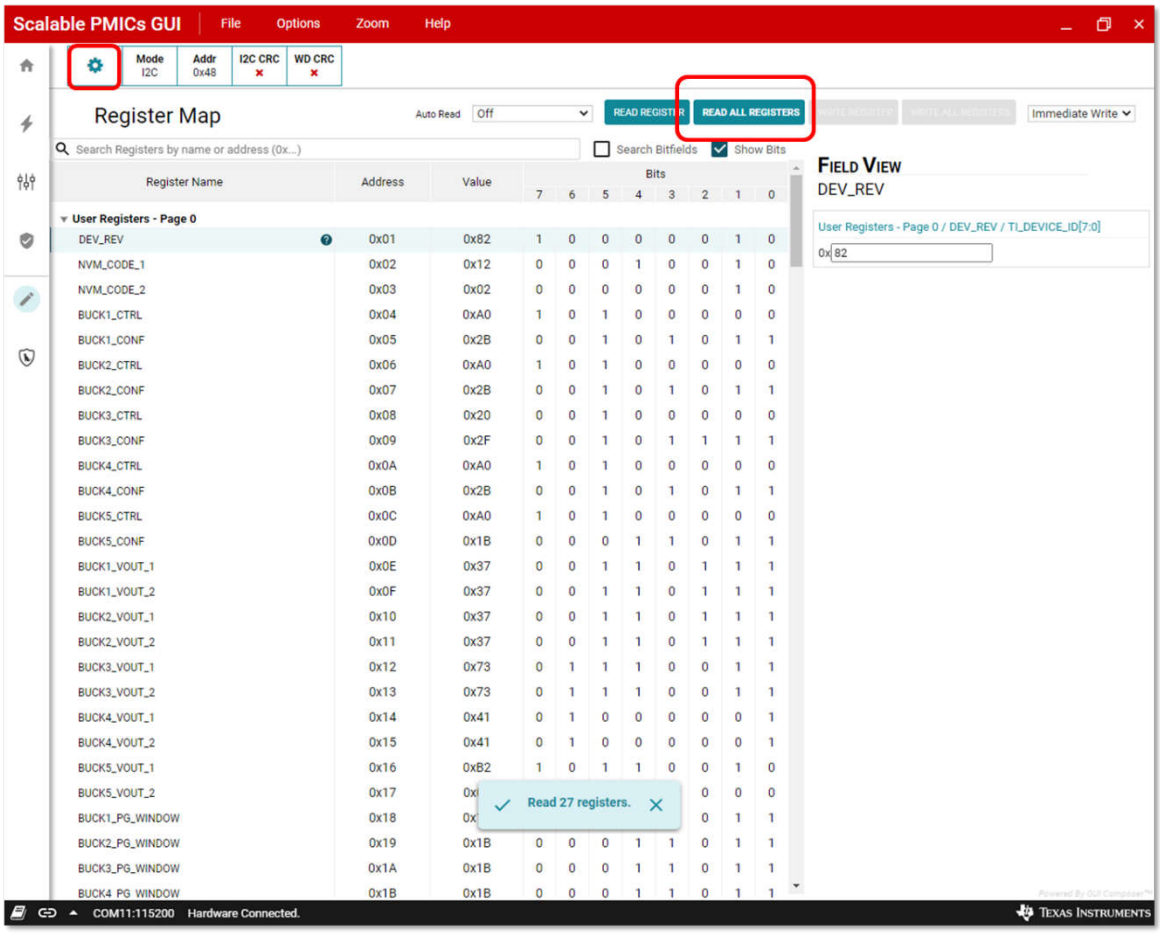

Figure 4-6. Register Reading Interface

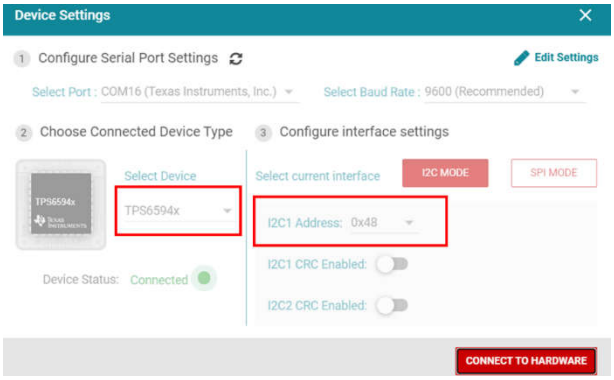

Figure 4-7. PMIC Switch Address Pop-up Window

More usage methods of the GUI software can refer to the Scalable PMIC's GUI User's Guide document. This article only discusses part of the functions needed to be used for locating problems.

4.1.4 Reading and Saving of Interrupt Information

The interrupt information is stored within the register address range from 0x5A to 0x6C. After clicking each corresponding bit inside the software, the detailed explanation of this bit can be seen in the right sidebar, and the analysis of alarm information refers to the second piece of the series of articles. In order to facilitate subsequent further analysis and discussion, storing the alarm information down is very necessary. In addition to saving the software of the corresponding part of registers through the method of screenshots, it can also use the method of scripts to save into a pure text form.

As shown in Figure 4-8, clicking option on the menu bar on any interface, choosing the Scripting option, a script pop-up window appears, as shown in Figure 4-9. The content of this pop-up window is merely a routine and is in write-protect mode unable to be edited, and it needs to save the current routine as a script file before opening again then it can be edited. Clicking the book-shaped icon on the top right corner of the window, a log window will appear below the current window, and the read register information will all be displayed in the log window.

Clicking the taskbar icon to return to the Scalable PMIC GUI main window (no need to close the scripting window), according to software operation step 2 connect up the I2C of the PMIC to be tested. Copy the script in the appendix to the scripting window, click the green triangle icon to run the script, and the log window will display the read register information, as shown in Figure 4-10. Clicking the save button, the data will be saved as a CSV file. Using scripts can also read some hidden registers for use in locating problems. In the appendix, a part of internal registers related to SPMI, PFSM, and FSM is additionally added, and they have certain reference value in locating. For the parsing of this part of register content, please consult the FAE team.

It needs to be noted that even if I2C is not connected, the script will also run normally, only that the read register values are all 0, and special attention needs to be paid to avoid saving useless data.

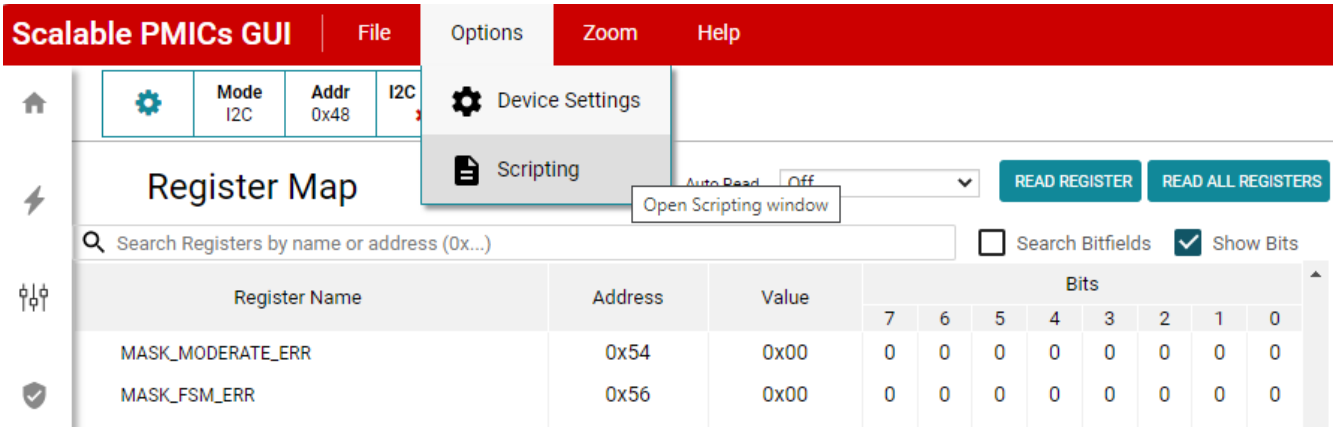

Figure 4-8. Scripting Menu

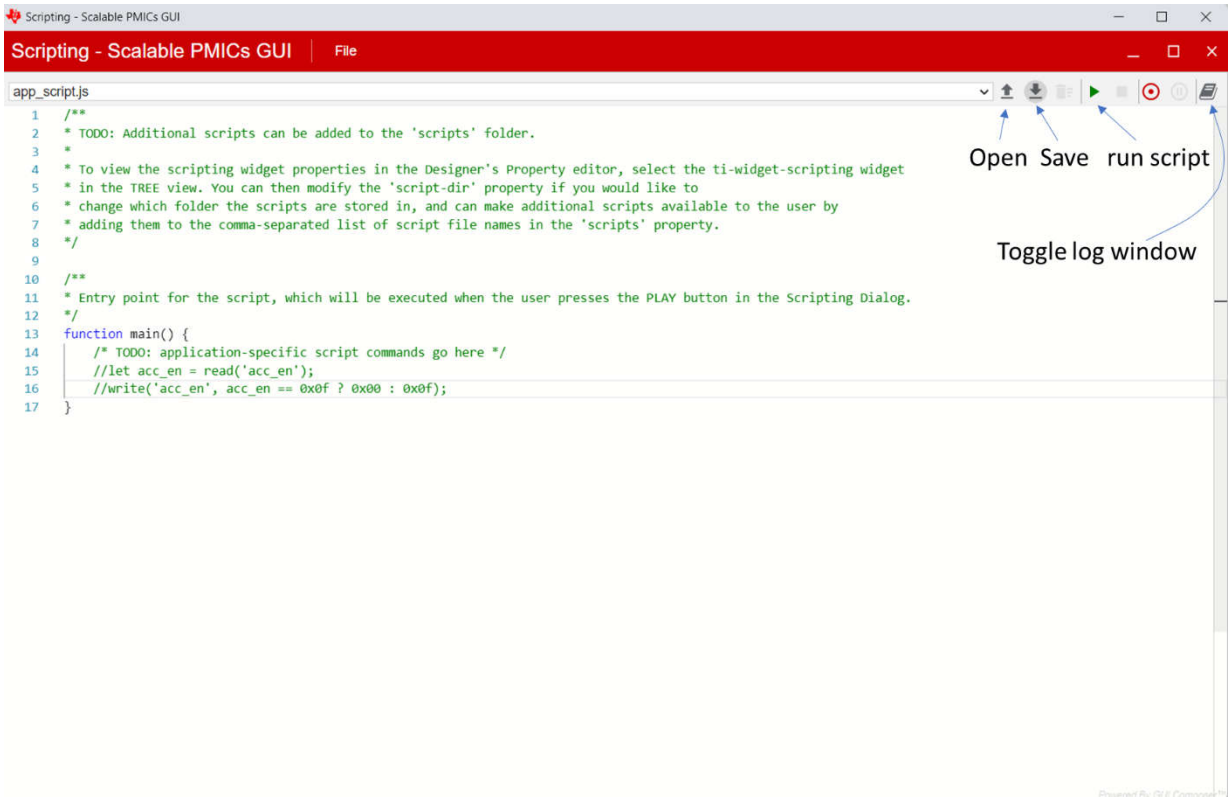

Figure 4-9. Scripting Window 1

Figure 4-10. Scripting Window 2

5 Changing NVM to Locate Problems

The previous several methods all observe the phenomena caused by problems through the method of "consultation" thereby locating faults. Sometimes through "consultation" multiple combined fault effects' final results might be seen. In order to locate the root cause, through the method of temporarily changing the NVM, a part of causes of disease can be shielded to determine the final root cause. This is just like "doing surgery" for the board, and a well-designed "surgery" plan can accelerate the location of problems.

5.1 Using Scalable PMIC GUI Software to Update NVM

1. The Scalable PMIC GUI software provides a tool to update NVM, as shown in Figure 5-1. Clicking the Begin an NVM Configuration button on the top right corner of the software main interface enters the NVM editing interface as shown in Figure 5-2. In this interface, select the corresponding device model on the left, and at this time skip to programming on the bottom right corner lights up. Click the skip to programming button.

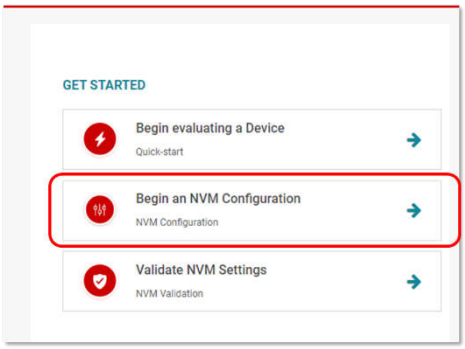

Figure 5-1. Begin an NVM Configuration Button

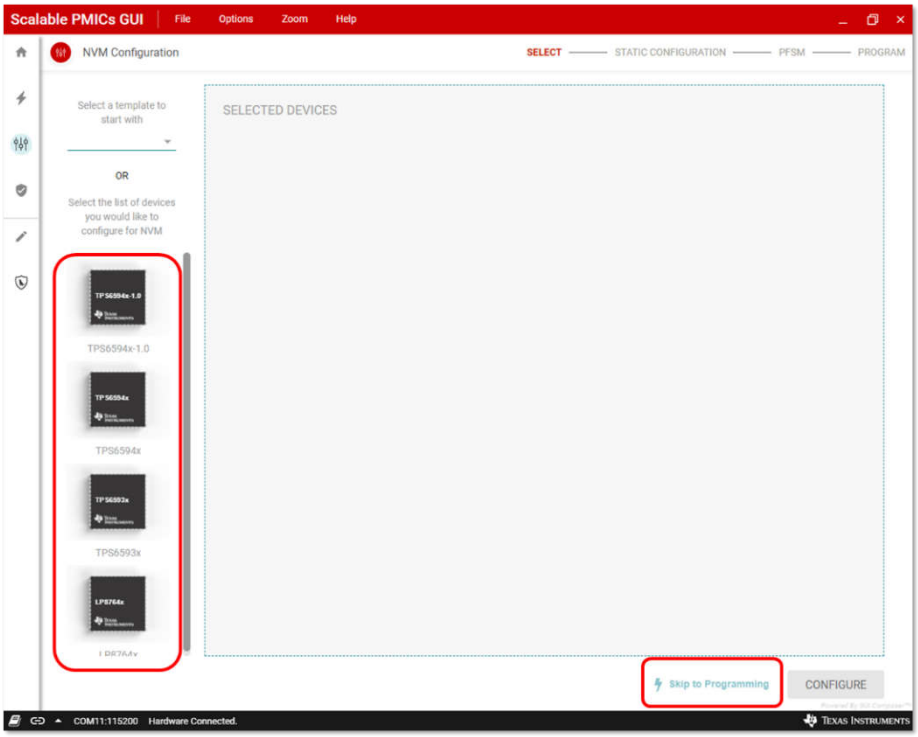

Figure 5-2. NVM Configuration Interface

2. Afterwards, enter the program NVM interface as shown in Figure 5-3, click the Select from your computer button on the left side, and load the NVM firmware that needs to be burned. This firmware can be the original NVM of the device (to restore the product to its original state), or an NVM modified based on the original NVM (to locate problems), or an NVM generated by oneself (to satisfy special purposes). The topic of generating NVM by oneself will be discussed in the fifth piece of this series. Pay attention to the information

on the top right corner to confirm whether the chip is in a connected state. The interface when not connected is as shown inside the green box, and it needs to choose the right I2C address and then click connect to hardware to connect the PMIC.

3. The interface after loading the NVM file is as shown in Figure 5-4. If needing to replace the firmware, one can click the icon shown in the red box to reselect the NVM file. Finally click the buttons in the green box from top to bottom, check the connection state and start burning. If seeing the prompt displaying cannot connect to WD address in red words just like that shown in the figure, it is also normal, and the reason refers to the explanation of step 5 of the hardware part.
4. Thereafter three pop-up windows will pop out in sequence, as shown in Figure 5-5. For the first two, please be sure to click the red buttons, and for the third pop-up window, choose according to requirements. Up to this point, the NVM in the PMIC has been updated into the required firmware. It needs to be noted that during two consecutive NVM update operations in the program NVM interface (two updates for the same chip or changing a chip to conduct NVM update), a prompt of burning failure will appear the second time. At this time, it only needs to reselect the file, click the icon shown in the green box in Figure 16, and then program one more time, which is a known bug of the scalable PMIC GUI software.

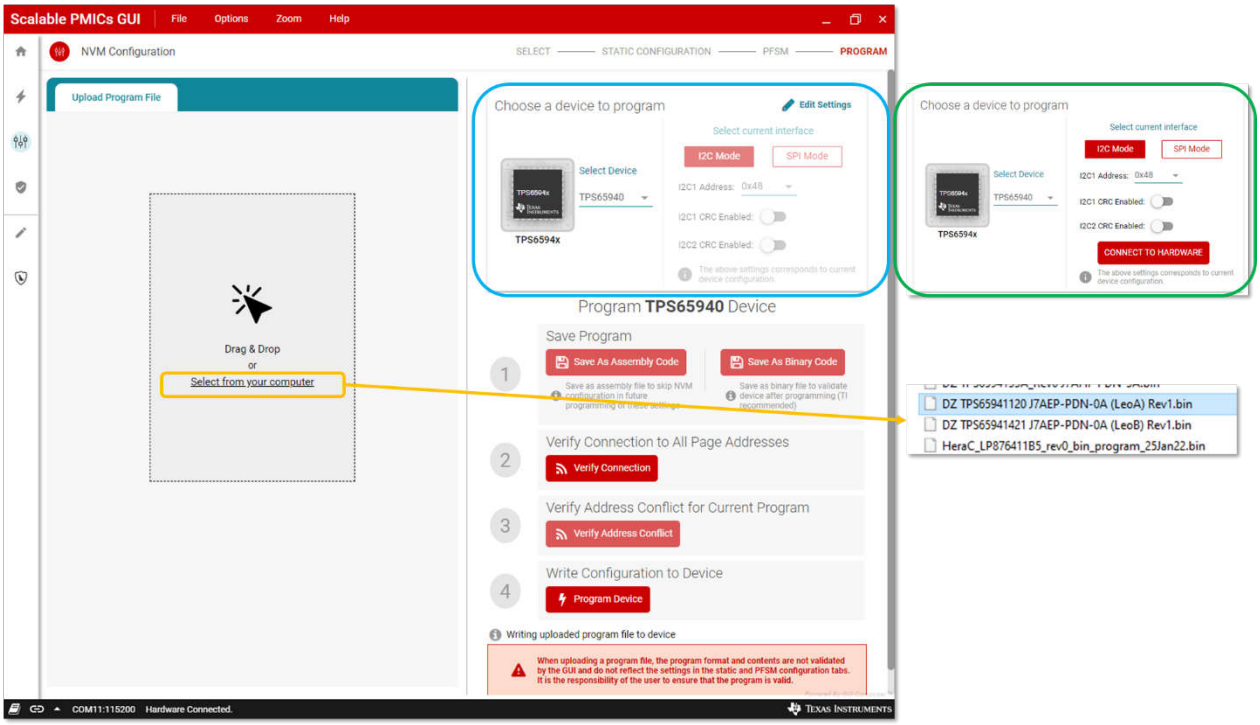

Figure 5-3. Program NVM Interface 1

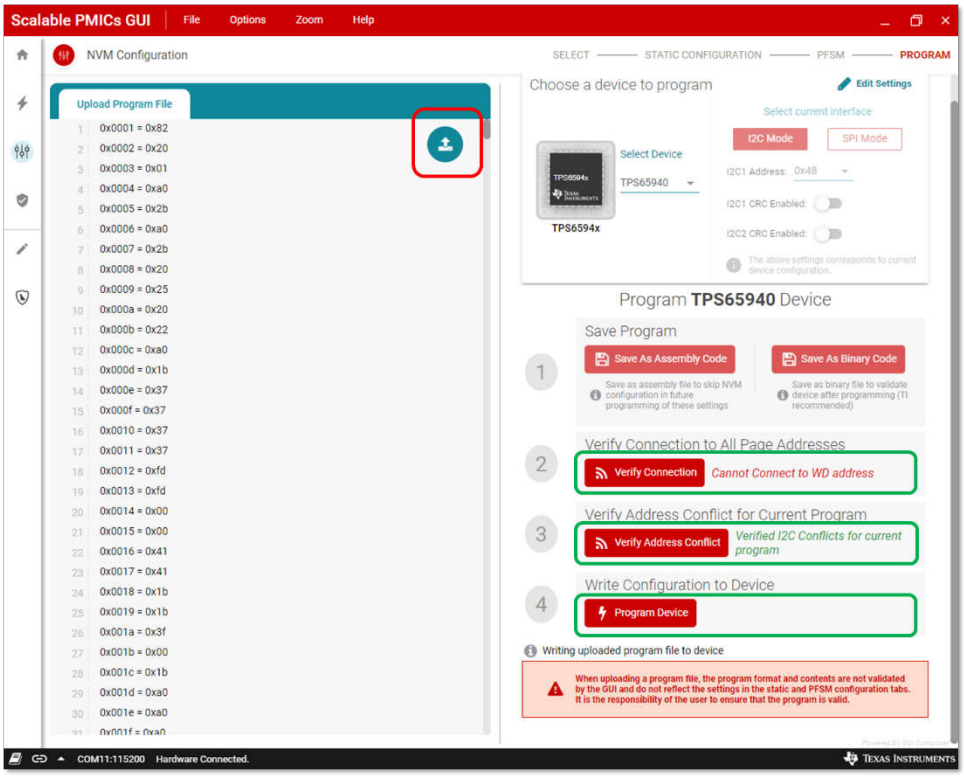

Figure 5-4. Program NVM Interface 2

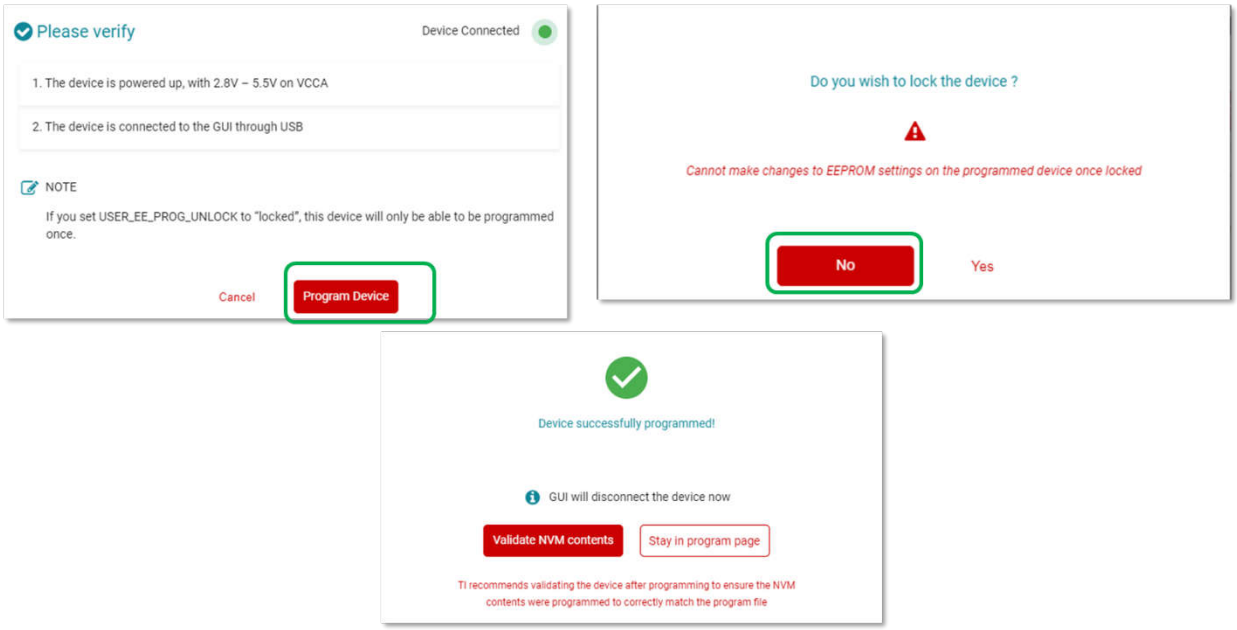

Figure 5-5. Program NVM Pop-up Window

5.2 Actual Application Case

In order to locate the fault of individual single boards powering down a short time after starting up under high temperature, it needs to find a way to obtain register information. However, during the shutdown process of the faulty single board, because VCCA rushed high, it triggered VCCA OVP (the case introduced in the third piece of this series; the workaround solution for this case is that the SOC configures the BUCK's slew rate to a lower rate

after power-up, but the power-down occurred in a very short time after power-up, and the SOC did not have time to configure registers), and register information was lost because of the closing of the NMOS.

Through oscilloscope waveforms, after clarifying that this fault of VCCA OVP was caused by the PMIC power-down behavior rather than it causing the power-down itself, we can, through the method of updating the NVM firmware, change the power-up and power-down slope of the BUCK to a level that will not trigger VCCA OVP, as shown in Figure 5-6, using a text editor to modify the corresponding register bits in the original NVM bin file and then saving as a new bin file. Using this new NVM, we successfully read the register interrupt information, and located the faulty power rail that triggered the power-down behavior.

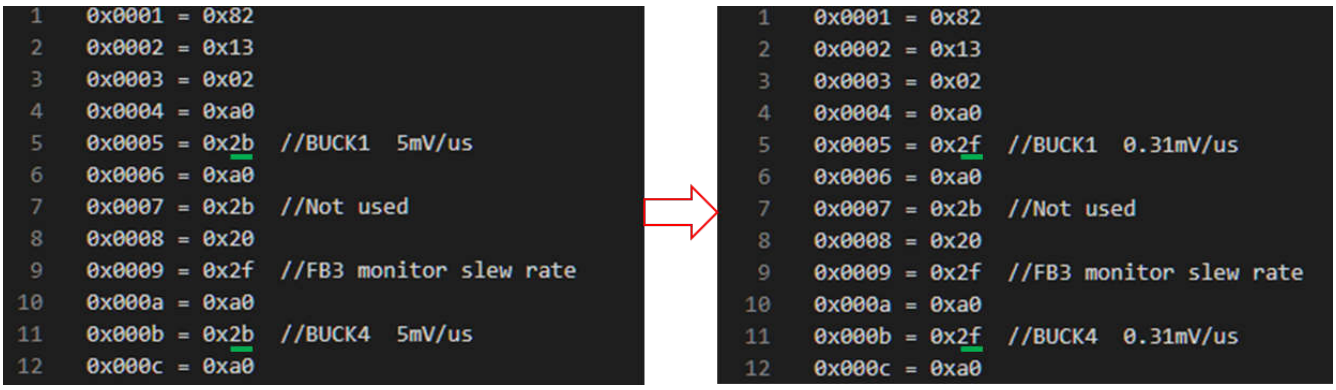

Figure 5-6. Modifying NVM bin File

6 Appendix - Register Reading Script

```
function main() {
read("DEV_REV"); //check device type TPS6594 or LP8764
read("NVM_CODE_1"); //check OPN
read("NVM_CODE_2"); //check OPN
 read('INT_TOP'); //INT
 read('INT_BUCK');
 read('INT_BUCK1_2');
 read('INT_BUCK3_4');
 read('INT_BUCK5');
 read('INT_LDO_VMON');
 read('INT_LDO1_2');
 read('INT_LDO3_4');
 read('INT_VMON');
 read('INT_GPIO');
 read('INT_GPIO1_8');
 read('INT_STARTUP');
 read('INT_MISC');
 read('INT_MODERATE_ERR');
 read('INT_SEVERE_ERR');
 read('INT_FSM_ERR');
 read('INT_COMM_ERR');
 read('INT_READBACK_ERR');
 read('INT_ESM');
 read('MISC_CTRL');
 read('ENABLE_DRV_STAT');
 read('RECOV_CNT_REG_1'); //RECOVCNT status
 read('RECOV_CNT_REG_2');
read('TEST_SPMI_1'); //SPMI related internal registers
read('TEST_SPMI_2');
read('TEST_SPMI_5');
read('TEST_SPMI_6');
read('TEST_SPMI_7');
read('TEST_PFSM_0'); //PFSM status
read('TEST_PFSM_1');
read('TEST_PFSM_2');
read('TEST_PFSM_3');
read('TEST_FFSM_1'); //FSM status
read('TEST_FFSM_2');
}
```

7 References

1. Datasheet "TPS6594-Q1 Power Management IC (PMIC) with 5 BUCKs and 4 LDOs for Safety-Relevant Automotive Applications"
2. Datasheet "LP8764-Q1 Four-Phase, 20-A Buck Converter With Integrated Switches"
3. Application Note "Scalable PMIC NVM Update Guide"
4. User's Guide "Scalable PMIC's GUI User's Guide"
5. User Guide "TPS65941213-Q1 and LP876411B4-Q1 PMIC User Guide for J721E, PDN-1A"
6. User Guide "TPS6594x-Q1 Evaluation Module user's guide"

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025