User's Guide

Scalable PMIC's GUI User's Guide

ABSTRACT

This document covers the usage and capabilities of the Scalable PMIC's graphical user interface (GUI) tool from Texas Instruments. This GUI is intended to be used with the analog EVM Control (AEVM) to evaluate and configure the TPS6594-Q1, TPS6593-Q1, LP8764-Q1, LP8762-Q1, and LP876242-Q1 family of devices.

Table of Contents

List of Figures

List of Tables

Trademarks

SimpleLink™ and Launchpad™ are trademarks of Texas Instruments.

1 Introduction

This tool is based upon GUI Composer and requires the MSP432E401Y SimpleLink™ Ethernet Microcontroller as the analog EVM controller (AEVM), which is integrated in the PMIC EVM and also available with the Launchpad™, see 1 in the Section 11. The AEVM provides a USB interface to the host personal computer (PC) for receiving commands and then communicates with the PMIC using either an I^2C or SPI protocol.

The GUI supports multiple devices with a single executable (and single AEVM), which eliminates the need to install multiple GUIs when working with more than one device. Multiple devices are configured in a primary/secondary configuration and the AEVM communicates to each device over the selected shared medium: I^2C or SPI.

2 Supported Features

The GUI supports the following features:

- Interface Multiple PMIC devices with a single GUI
- Quick-start and Register views to read and write to PMIC registers after power up
- Status Indicators for Interrupts and GPIO states
- Programming and Validation of non-volatile memory (NVM)
- Watchdog configuration and evaluation
- Multiple platforms [1]: Microsoft Windows,® Linux® 32 and 64-bit, and Mac OSX
- Web based and standalone versions available
- Links to additional collateral, support forums, and FAQ

[1] Please refer to dev.ti.com for version compatibility with GUI Composer.

3 Revisions

This section details the features added with each release of the Scalable PMIC's GUI. Section 14 also lists known issues for each version.

Release 1.0.0 is the initial pre-production release and named the Programmable Processor PMIC's GUI. This version of the tool allows for evaluation and programming but does not provide a mechanism to create NVM configurations.

Release 2.0.0 built upon the features of release 1.0.0, specifically adding the NVM Configuration, NVM Validation, and WatchDog evaluation pages.

Release 3.0.0 is the latest release and named the Scalable PMIC's GUI. This document reflects this latest version.

Table 3-1. GUI Revisions

Revision	Release Date	Devices Supported	Feature Updates and Additions
1.0.0	December 2019	• TPS6594-Q1 Revision 1.0 Silicon	Initial Pre-Production Release • Quick-start Page • Register Map Page • NVM Programming
2.0.0	November 2020	• TPS6594-Q1 • LP8764-Q1 Revision 1.0 Silicon • TPS6593-Q1 • TPS6594-Q1 Revision 1.0 Silicon	• NVM Creation and validation • NVM Templates • Scripting • LP8764-Q1 Device Support • Improved programming speed • Watchdog Evaluation Page • SPI and CRC enabled serial communication • Automated Timing adjustements to meet desired delay times.
3.0.0	June 2022	• TPS6594-Q1 • LP8764-Q1 Revision 2.0 Silicon • TPS6593-Q1 • LP8762-Q1 • LP876242-Q1	• NVM Creation updated to allow for more flexibility in NVM design. • More feedback mechanisms to identify potential NVM issues. • Multi-SPI Communication controls Chip Select for all PMICs on the SPI bus. • Improved transitions between pages. • Removed automated timing adjustments to meet desired delay times. • Added Validation errors and instructions to manually control delay timing.

4 Overview

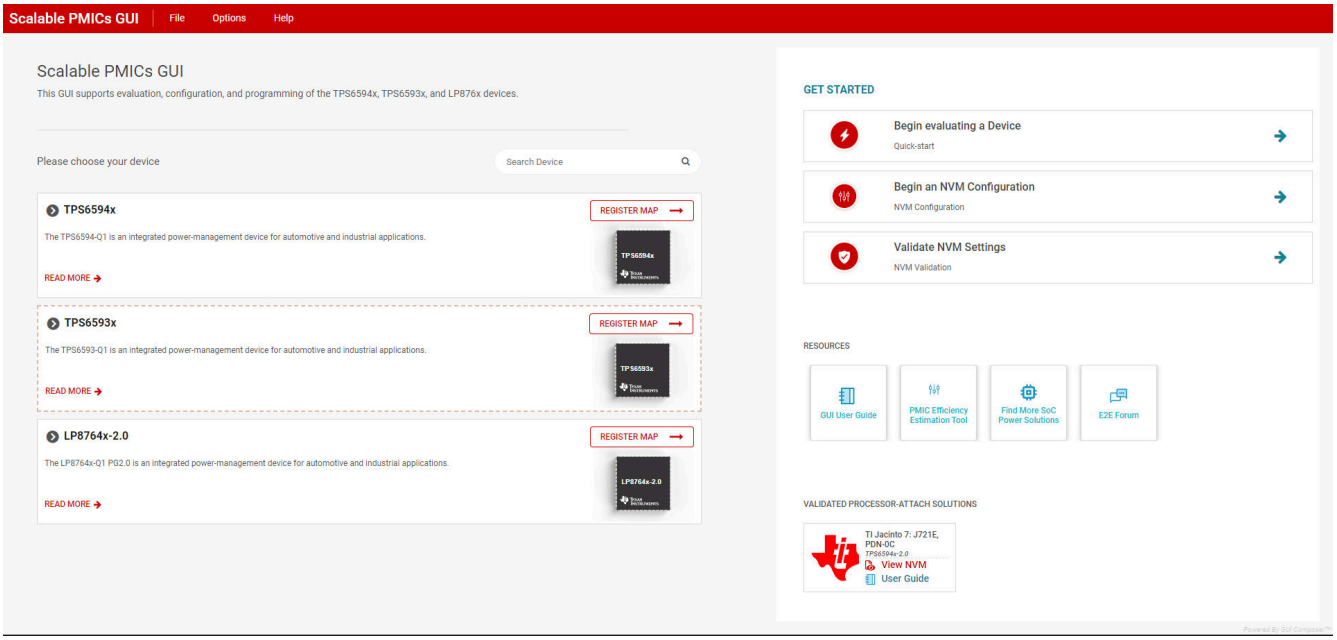

Figure 4-1. GUI Overview

The GUI provides pages to both evaluate and configure the PMIC. For evaluation, the quick start, register map, and watchdog pages provide an interface to monitor and control the PMIC via a SPI or I^2C interface. Configuration of the PMIC NVM is done through the NVM Configuration page and the NVM validation page is provided to read the NVM content of the PMIC. More information on the Quick-start, Register Map, NVM Configuration, NVM Validation, and Watchdog Evaluation pages will be provided in the subsequent sections.

The GUI also provides templates and references to the user guides and data sheets to assist in the development process. It is recommended, when applicable, to use an existing template as a starting point for development. These templates not only provide a graphical representation of the PMIC operation but also generate the required NVM files to program the PMIC.

Device Selection

The Device Selection page is the first page presented when the GUI is started. From this page a specific device can be selected by selecting the Register Map within a given device window. The other pages can also be selected without regard of a particular device. From these pages it is necessary to use the Device Interface Settings window to select which device as well as the interface.

Register Map

The Register Map page is a list view of the available user registers with the ability to read and write to those registers.

Quick-start

The Quick-start page is recommended as the starting point for evaluating the device. The Quick-start page provides a graphical view for interacting with the PMIC and configuring the device registers. In addition to this abstracted view is the actual register view provided from the Register Map Page.

NVM Configuration

The NVM Configuration page provides a means to both configure and program the non-volatile memory (NVM) settings. The configuration includes both the static settings as well as the pre-configured state machine (PFSM) settings.

NVM Validation

The NVM Validation page reads out the NVM content and can be used to verify the contents of the NVM match what was programmed from the NVM Configuration page.

Watchdog Evaluation

The WatchDog Evaluation page is a unique page providing a graphical interface for both configuring and exercising the Watchdog feature on select PMICs.

5 Getting Started

Getting started involves the following steps:

1. Find the GUI within the Gallery.
2. Download the required software.
   a. GUI Composer Runtime for running the GUI from a web browser
   b. An offline copy of the GUI
3. Launch the GUI.

5.1 Finding the GUI

The PMIC GUI is based upon GUI Composer which is compatible with either Chrome™ (version 46+) or Firefox™ (version 38+). The Chrome™ web browser is recommended and used throughout this document for demonstration. The GUI is found through the TI Development tools at TI DevTools page. Navigating to the Gallery from the Explore tab, highlighted in red in Figure 5-1, is one way to enter the Gallery.

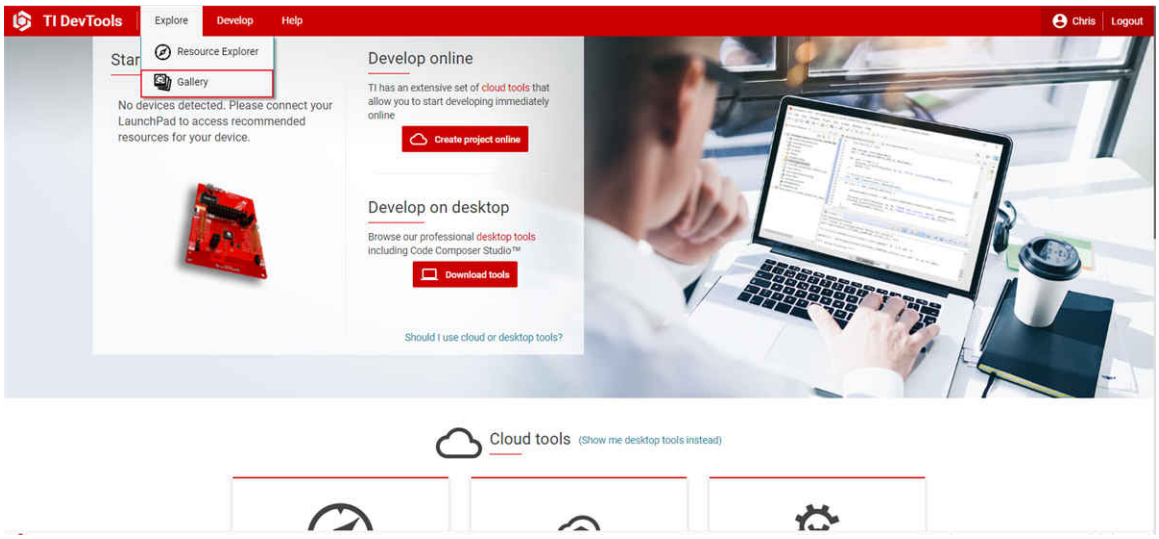

Figure 5-1. GUI Composer Gallery

When in the gallery, locate the Scalable PMIC's GUI panel shown in Figure 5-2. If the panel is not visible on the main page, then use the search bar and enter the term *PMIC*.

Note

The name of the GUI has changed so panels are available for both the Programmable-Processor-PMICs-GUI and the Scalable-PMICs-GUI. The Scalable-PMICs-GUI version 3.0.0 is the latest version.

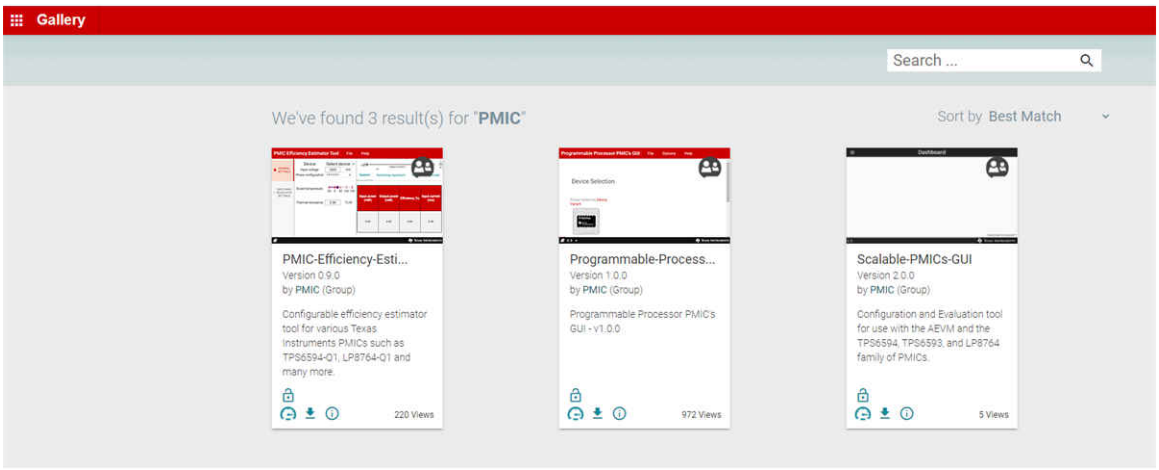

Figure 5-2. Locating the PMIC GUI in the Gallery

5.2 Downloading the Required Software

Both the standalone GUI and the GUI Composer Runtime are available from the PMIC panel. Again, the GUI Composer Runtime enables the GUI to be run through a web browser. This is a small download but still requires an internet connection to be able to run the GUI. By contrast the standalone GUI is much larger but does not require an internet connection.

The download options are found in the pop-up window, as shown in Figure 5-3, when the cursor is placed on the download icon. The upper three options are for a standalone download for the appropriate operating system, while the lower three are for the GUI Composer Runtime.

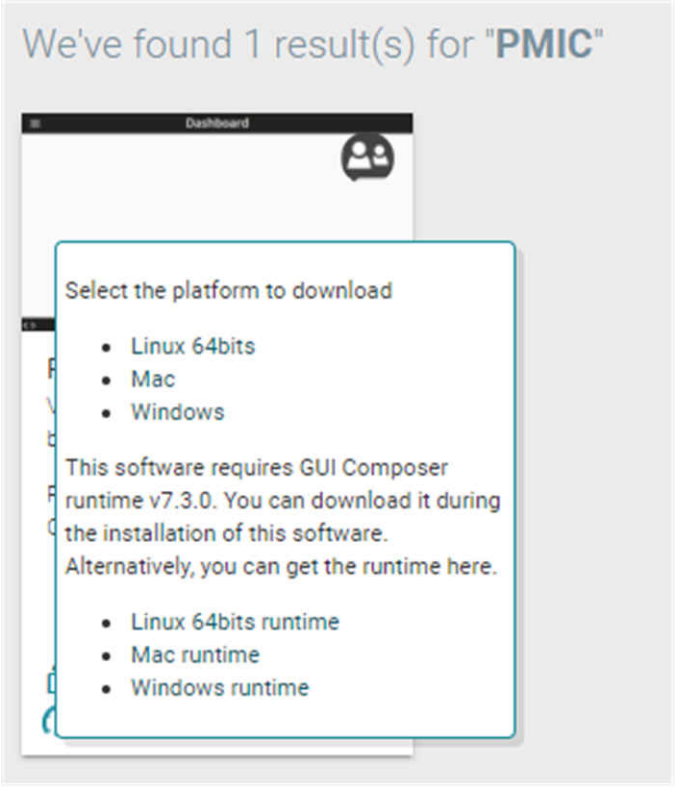

Figure 5-3. GUI Software Download Options

5.3 Launching the GUI

After the appropriate software has been downloaded, the GUI can be launched locally from the PC application or from the TI Cloud using the Gallery. To use the TI Cloud version of the GUI, simply click anywhere in the panel, shown in Figure 5-4, that is not associated with the download or information icons.

Figure 5-4. GUI Panel Within the Gallery

Figure 5-5 shows an example of the PC application.

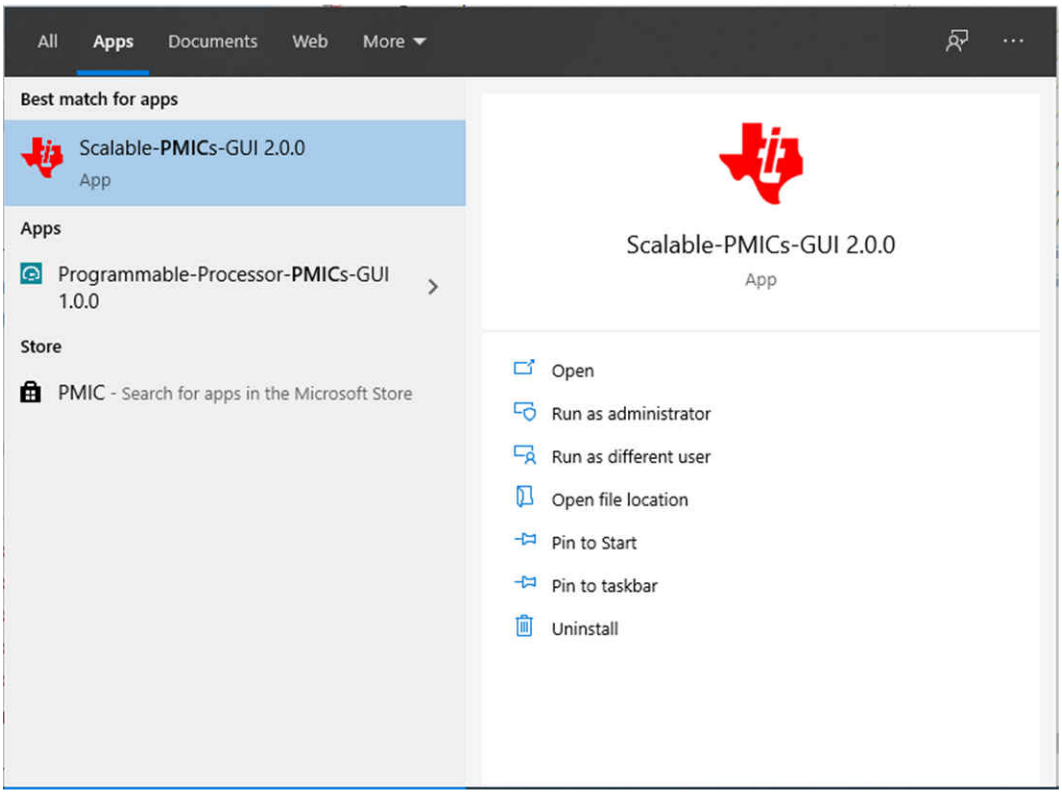

Figure 5-5. PMIC GUI Desktop Application

Launching the GUI automatically loads the Device Home page, shown in Figure 5-6.

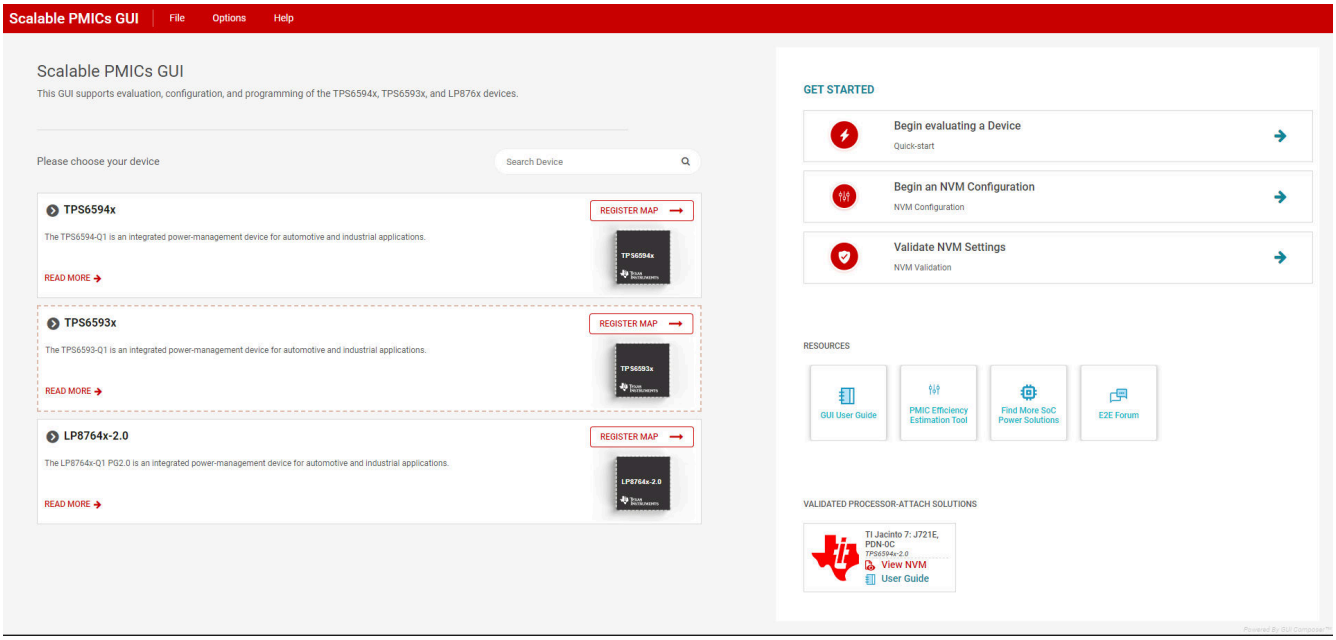

Figure 5-6. Device Home Page

Note

The GUI will not attempt to connect to the AEVM controller unless the register map or Quick-start page is entered. The GUI will attempt to automatically detect and connect to the correct Serial Port. Once this connection is made then the GUI will attempt to connect to the default I^2C address of 0x48. If the address is not acknowledged, then the GUI will provide the Device Settings window to update the interface and address. If the PMIC uses the SPI interface, then the initial attempt to connect to I^2C may impact the AEVM and require rebooting the AEVM with the SPI interface selected from the GUI. Refer to the Section 12 for additional tips on resolving connection issues.

5.4 Connecting to a PMIC

The GUI and AEVM support both I^2C and SPI with and without CRC. As mentioned previously, when entering the Register Map and Quick-start pages, the GUI will attempt to connect to any device with a non-CRC I^2C interface at address 0x48. If an alternate address or configuration is needed, then the *Device Settings* window is provided to change the settings, as shown in Figure 5-7. This window can also be accessed in the drop-down menu below *Options*.

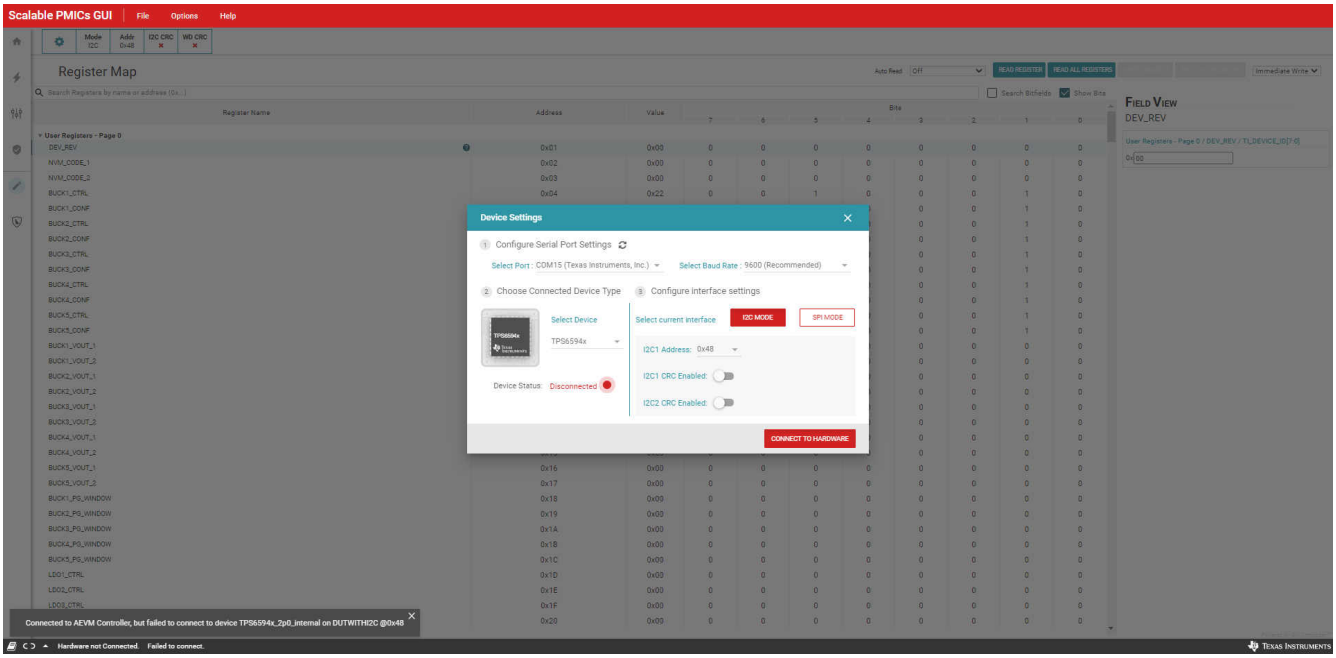

Figure 5-7. Device Settings From Options Tab

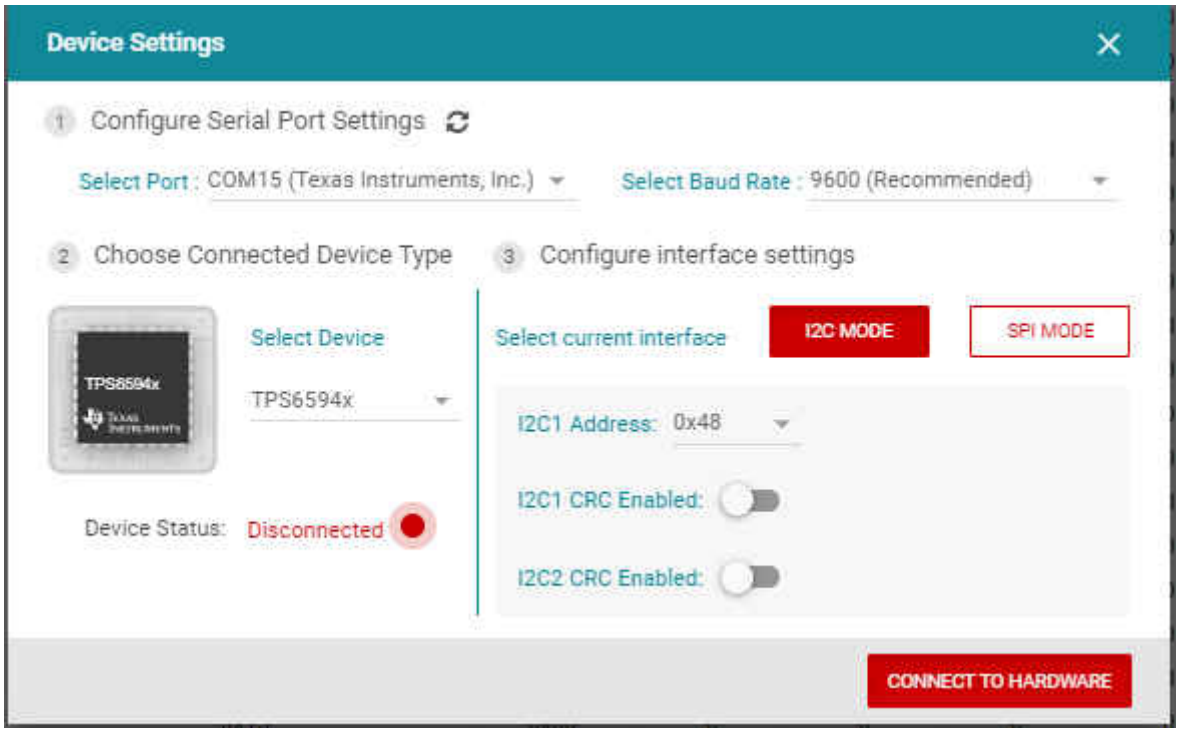

Figure 5-8. Device Interface Settings

As shown in Figure 5-8, options are provided to select the device as well as the interface.

Select Device

The device selection will determine the register interpretation of data written to and read from the PMIC. Failure to select the correct device can result in erroneous data being written to the PMIC or data read from the PMIC to be misinterpreted.

I²C

The I²C1 address selection is limited to valid page 0 addresses of the PMIC. Once the I²C1 address is specified, the GUI will automatically determine the addresses for pages 1-4 as well as determine if the physical I²C2 interface for page 4 is to be used.

SPI

Similar to the I²C implementation, the GUI will automatically update the page information during communication.

The latest AEVM controller firmware will support multi-PMIC operation by providing individual chip select (CS) control. As shown in Figure 5-9, up to 6 PMICs can be cascaded in a primary/secondary configuration and communication with each device is controlled by the chip selection. Table 5-1 shows the relationship with the AEVM GPIO.

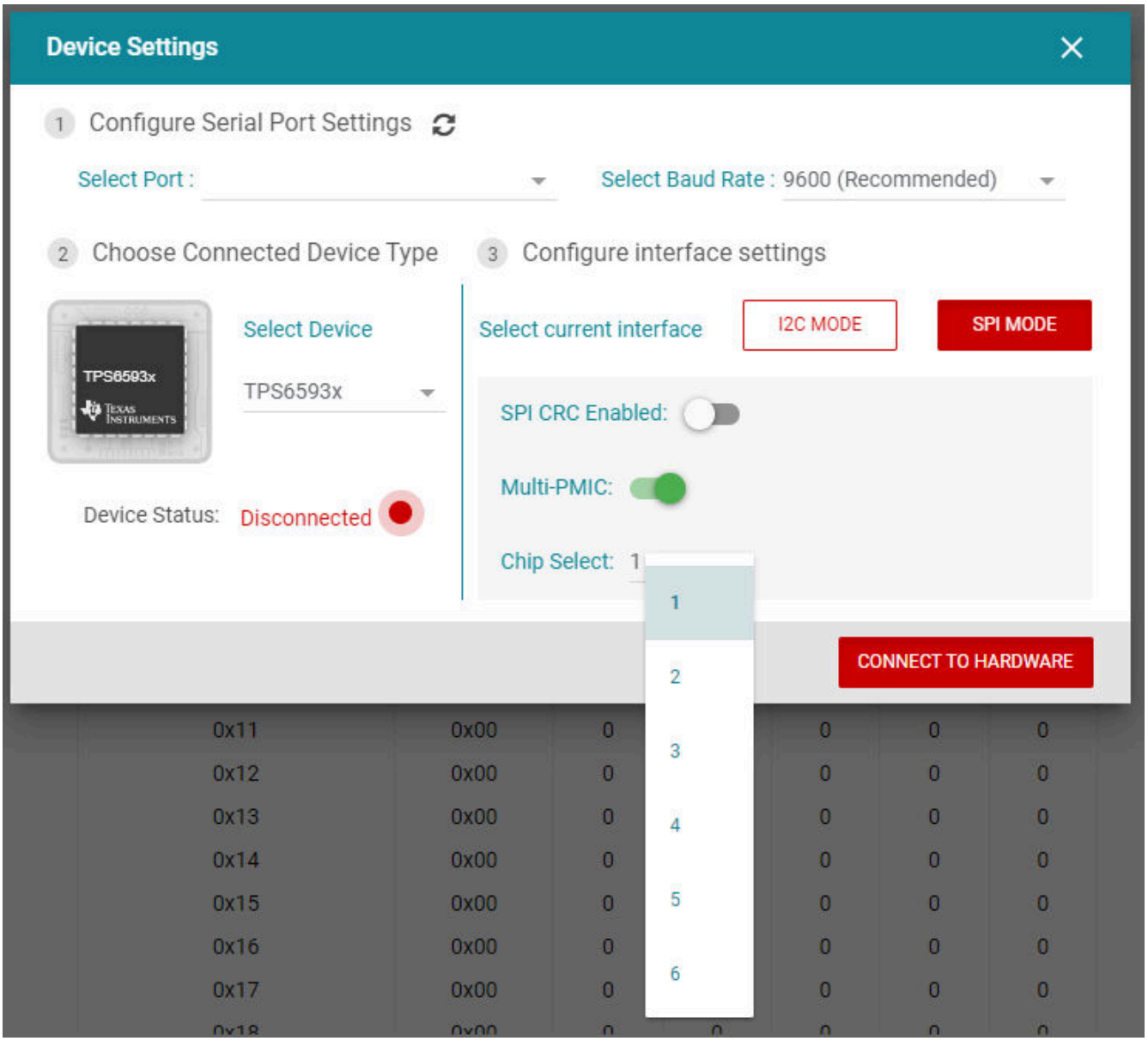

Figure 5-9. SPI Multi-PMIC Chip Select

Table 5-1. Chip Select Representation on AEVM

Chip Select	AEVM Port
Multi-PMIC is not selected	PD2, QSSI module CS is used

Table 5-1. Chip Select Representation on AEVM (continued)

Chip Select	AEVM Port
1	PD2, QSSI module CS is used
2	PD7
3	PC7
4	PC6
5	PC5
6	PC4

Updating the Interface

In addition to the *Option* tab at the top of the GUI, in the Quick-start, Register Map, and Watchdog pages, there is a device settings bar that shows the current interface selection. Clicking the gear icon within the bar also opens the Device Settings window. In the NVM Configuration and NVM Validation pages, the interface selection is provided within the page. Please refer to those sections for more details.

Figure 5-10. Device Settings Bar

Note

Connecting to a device is not required to access the GUI pages. Specifically, connecting to a device is not required to create an NVM configuration as described in section Section 8

6 Quick-start Page

From the initial Quick-start page in Figure 6-1, the *Detect Devices* box is provided to automatically detect all PMIC devices connected to the microcontroller's I^2C or SPI[2] bus. The GUI will attempt to connect through I^2C to the default address associated with the device. The *Hardware Connected* located at the bottom left of the GUI indicates that a device was found at that address. If no device can be found at the default address, then the device address (or configuration in the context of SPI) must be provided for at least one device connected to the GUI to use the *DETECT DEVICES* button. The interface connection can be changed by selecting the *Device Settings* from the *Options* drop-down menu at the top of the page.

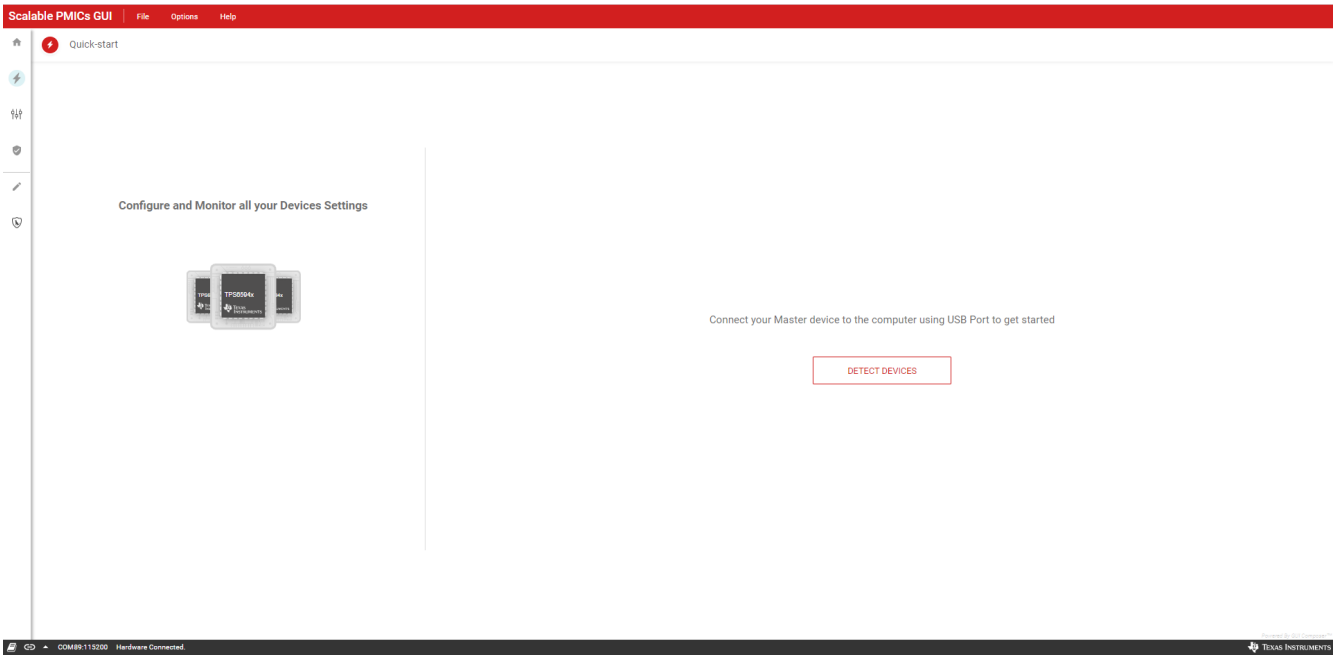

Figure 6-1. Quick-start Scan Page

[2] The Quick-start page does not support or display multiple devices on the same SPI bus. Each device must be selected individually. This can be done manually or with the chip select feature found in the devices settings.

6.1 Device Scan Results

Figure 6-2 shows, for this example, two PMICs are detected. One PMIC is configured as the primary, denoted with the *M* in the blue circle, and the other as the secondary. Additional information regarding the BUCK phase configuration and the *I^2C ID* of each device is indicated. The device label is an editable field and can be updated to provide a custom naming convention. By clicking the *Proceed* button, the quick-start page will advance to the interface window where select device registers can be written or read.

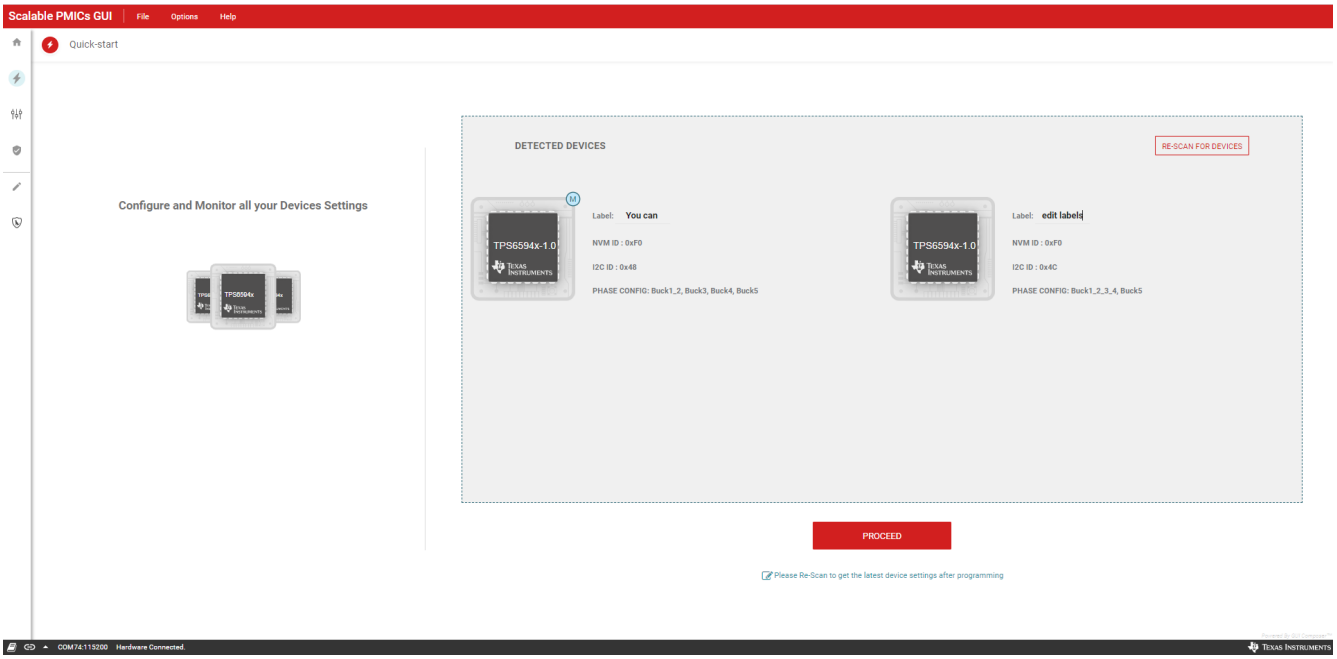

Figure 6-2. Quick-start Scan Page Results

6.2 Configuration and Monitoring

Within the Quick-start page there are six tabs aligned horizontally for editing and four tabs aligned vertical for monitoring and accessing advanced features. These are highlighted in Figure 6-3. These tabs in the Quick-start page are described in the following sections. It is important to note that the GUI is continuously polling the PMIC to update the Interrupts, Resource Status, and GPIO Pin status. Additionally, each update or change in value results in communication to the PMIC to update the appropriate register. The device can be reset with a power cycle to restore the register settings to the NVM values.

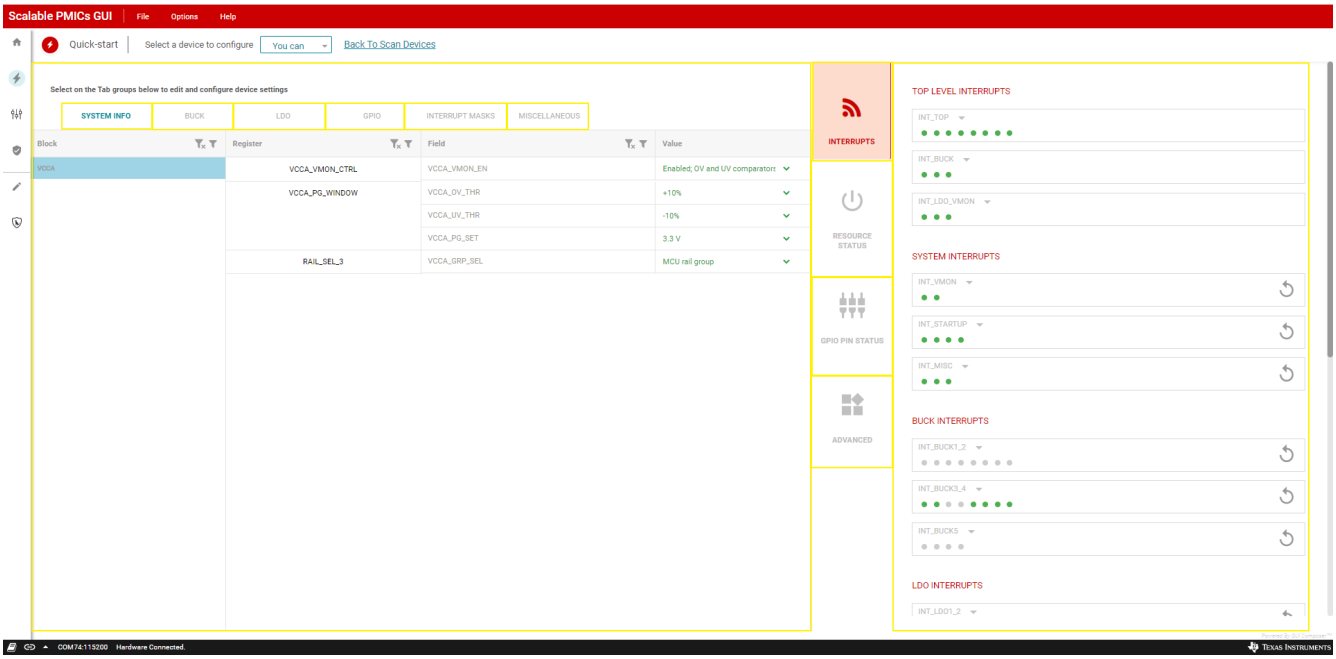

Figure 6-3. Quick-start Page Highlights

Note

At the top of the Quick-start page is a drop-down menu to select the device when multiple PMICs are connected. This label is defined in Figure 6-2.

Note

All register fields are direct references to the device specification. When values within a field are repeated or reserved, the GUI will not show these possible values in the drop-down menu options.

6.2.1 System Info

As shown in Figure 6-4, the system info tab is related to the VCCA input voltage and, when applicable, additional voltage monitors. Drop-down menus are provided to show the possible values for each field.

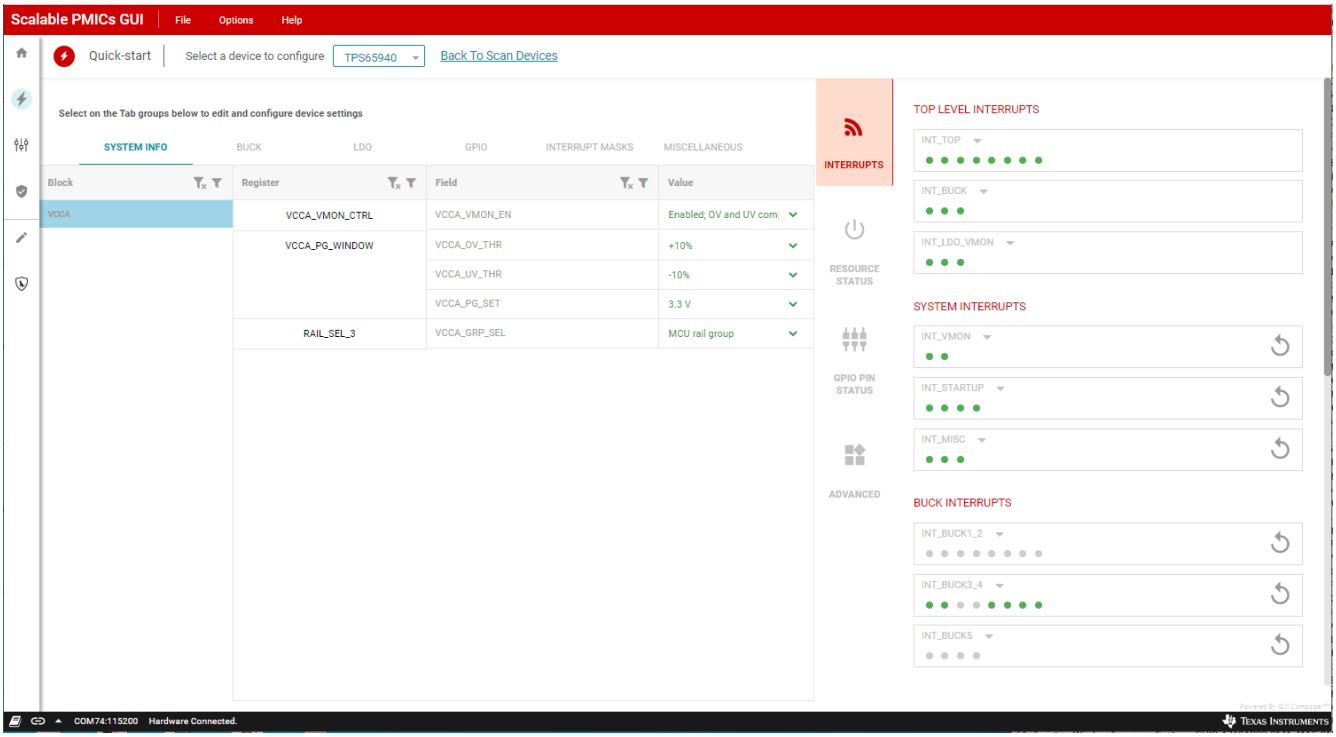

Figure 6-4. System Information

Note

The phase configuration cannot be changed with either the Quick-start or the Register map pages. To change the phase configuration the device NVM must be reconfigured, please see Section 8.

6.2.2 BUCK

The phase configuration will determine which BUCKs are available for editing through the quick-start. In Figure 6-5, the BUCK2 information is not available because BUCK2 is multiphased with BUCK1. The BUCK1 is treated as the primary and BUCK2 as the secondary, with all of the properties of BUCK1 being applied to BUCK2. Please refer to device data sheet for a more detailed description of the fields and the associated operation.

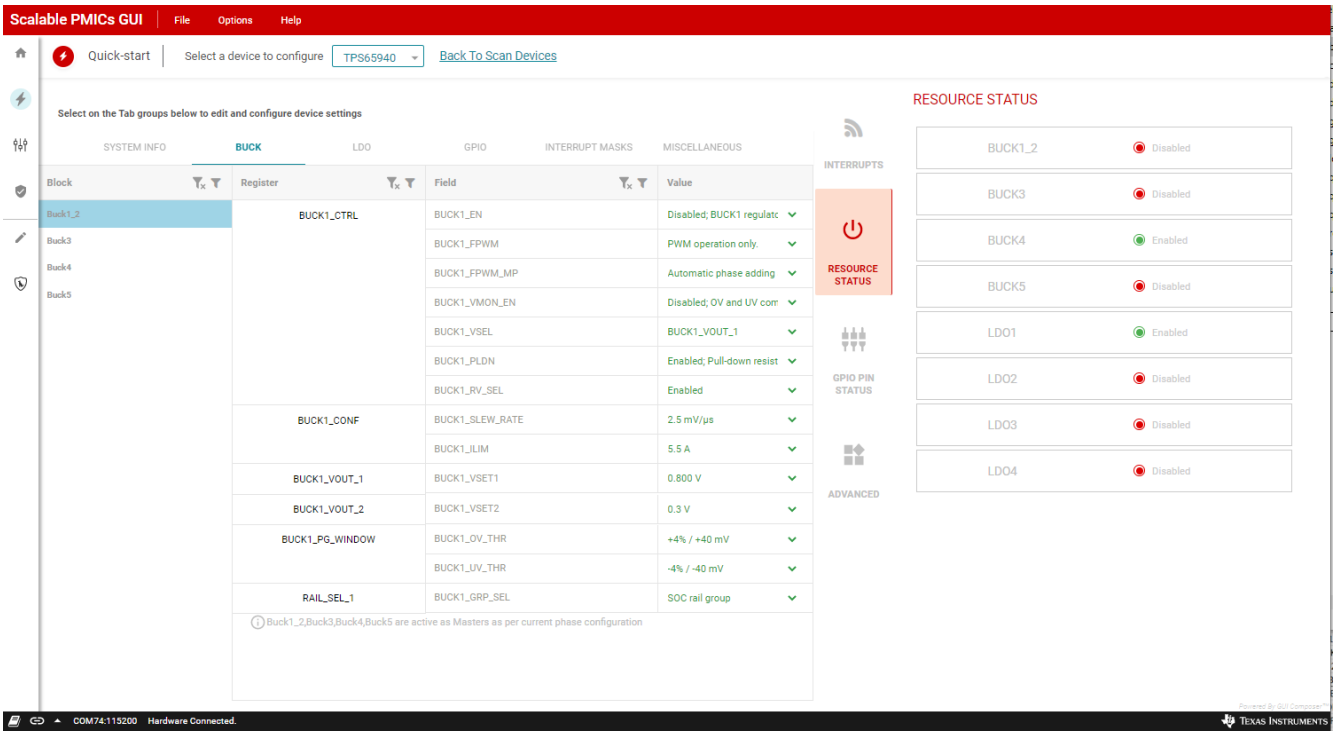

Figure 6-5. BUCK Configuration

Note

The frequency selection register FREQ_SEL is protected and cannot be changed from the quick-start page. To change this field, similar to the phase configuration, the NVM must be updated accordingly.

6.2.3 LDO

From the LDO tab, configuration of the LDOs is available. Please refer to the device data sheet for a more detailed description of the fields and the associated operation. If a PMIC does not have LDOs, then the LDO tab will be omitted.

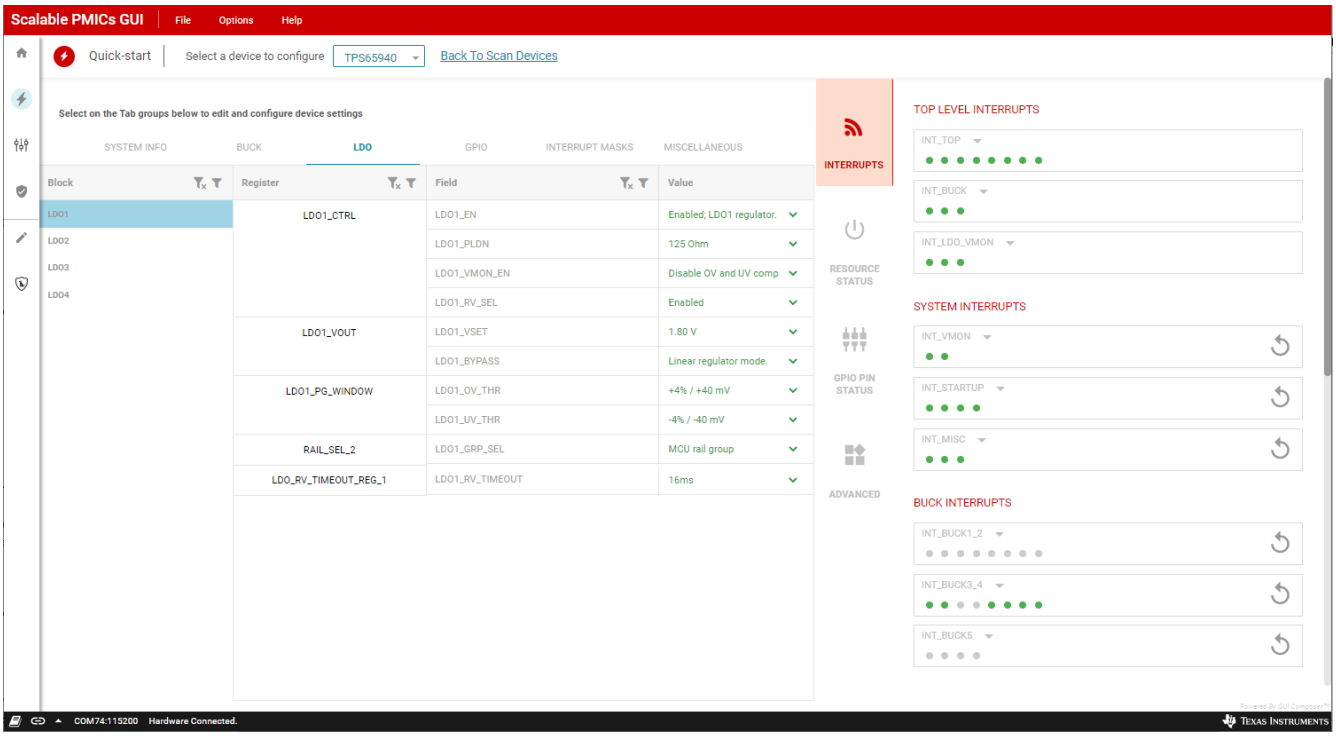

Figure 6-6. LDO Configuration

6.2.4 GPIO

The GUI provides the interface to configure the PMIC GPIOs. When the GPIO is configured as an input, the GUI provides an additional mechanism to drive the GPIO from an associated AEVM pin and view the state change through the GPIO Status vertical tab on the right side of the page.

The PMIC GPIO can be configured as input or output or be mapped to internal functions within the PMIC. For this example, the PMIC is using GPIO1 and GPIO2 for the second I^2C instance. The GPIO PIN STATUS window pane can be used to confirm the function and the level of each pin when configured as GPIO. The CONFIGURE GPIO LEVEL box appears, see Figure 6-7, when the GPIO direction is set to input to initially set the AEVM output level. Please ensure that the hardware platform is configured to support the intended GPIO operation.

Note

When evaluating a multi-PMIC solution, the control of the AEVM outputs are only for the AEVM which is connected to the GUI (connected to the PC through the USB port).

The GPIO pin status, see Figure 6-7, indicates the function of the GPIO as well as the current state (high or low) of GPIO that are not configured for special functionality. The colors are gray, red, and green. In addition to the color, a text description to the right of each indicator is also provided.

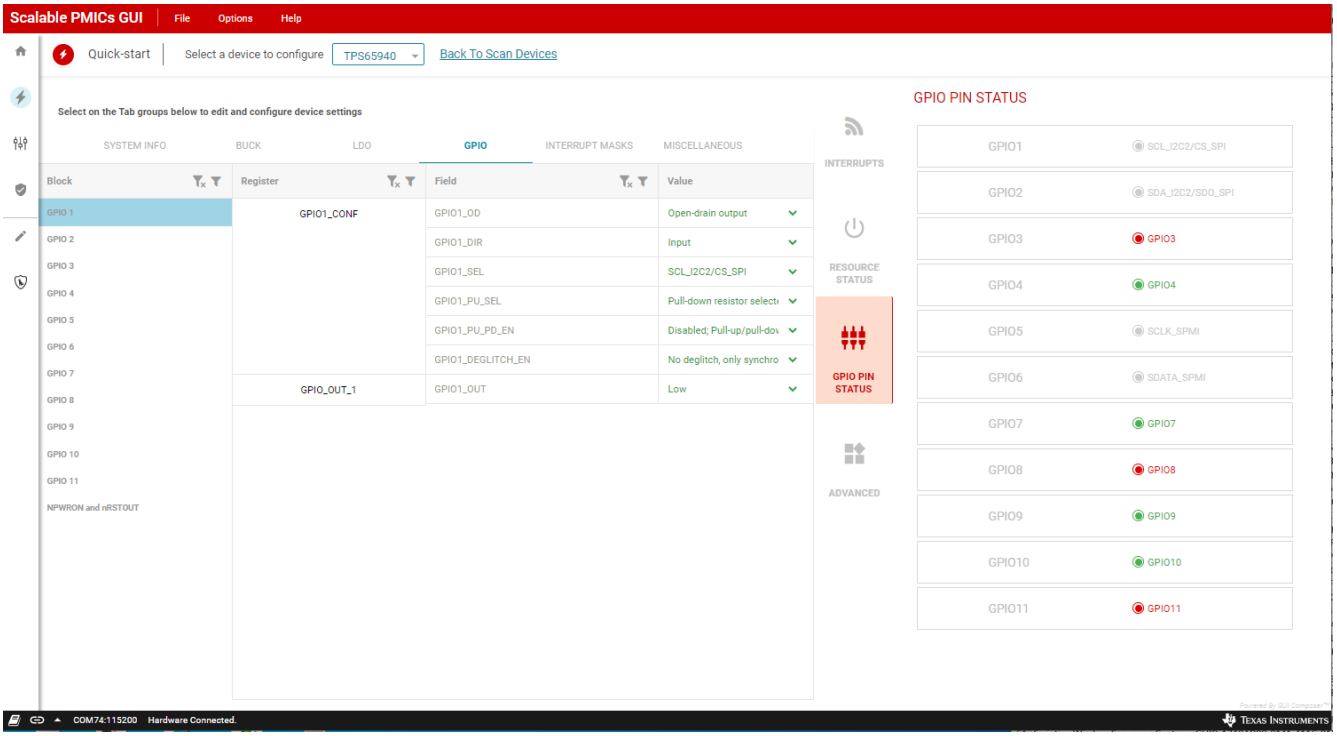

Figure 6-7. GPIO Configuration

6.2.5 Interrupts

From the interrupts tab the user can decide to mask or monitor various interrupt sources: State Machine, GPIOs, BUCKs, and so on.

The interrupt status, shown on the right side of Figure 6-8, can be used to monitor the interrupt events. The interrupts are grouped according to function and can be expanded to see each individual interrupt source. *TOP LEVEL INTERRUPTS* are read only and cannot be cleared. Other interrupts can be cleared at a register or bit level, as indicated by the reset symbol. An *ERROR* status with a red dot indicates that an interrupt has occurred while a *NORMAL* status with a green dot indicates that no interrupt has occurred or that the interrupt has been cleared. Typically, gray represents interrupts which are masked. If an interrupt has a *NORMAL* status with a gray dot, then this indicates that the interrupt is not applicable for the specified phase configuration. The GUI will ignore any attempt to unmask or generate an interrupt that is not applicable to the device phase configuration.

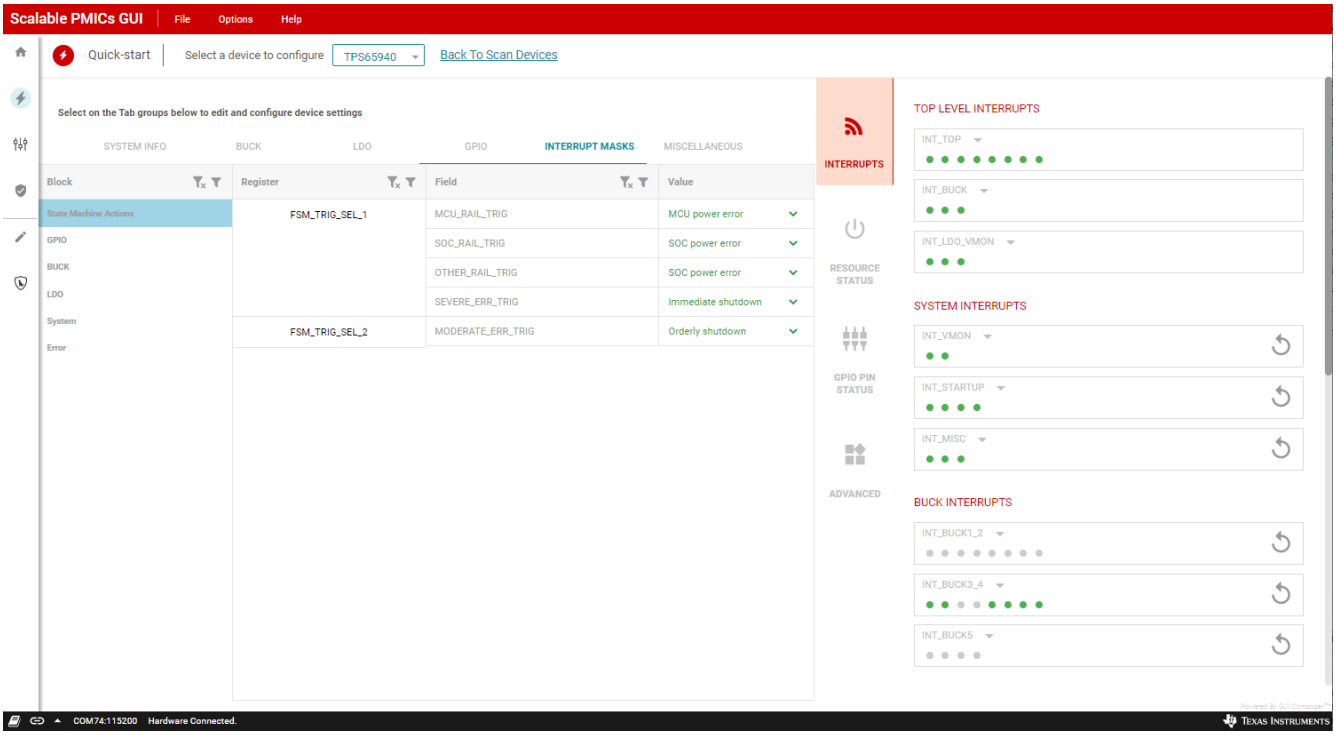

Figure 6-8. Interrupt Mask and Status

6.2.6 Miscellaneous Settings

The MISCELLANEOUS tab includes settings for the power good (PGOOD), spread spectrum configuration, and additional configurations of the PMIC.

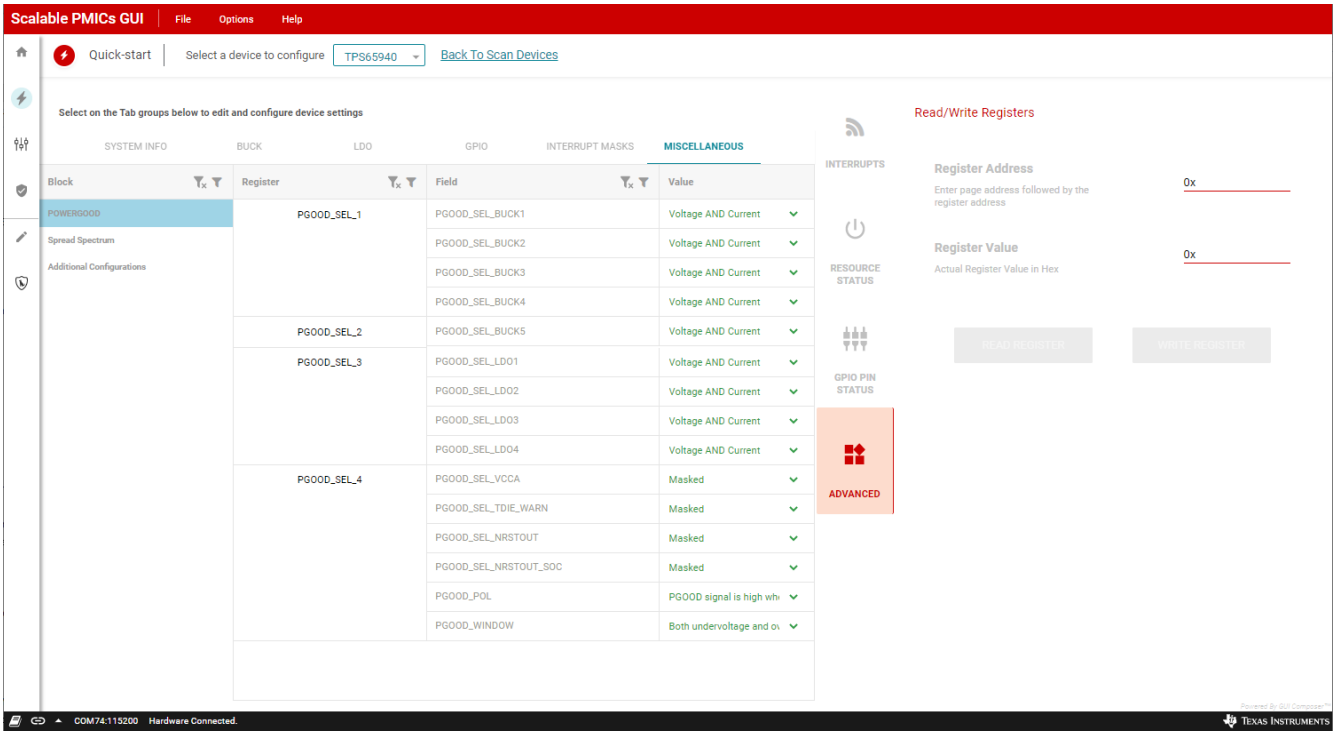

Figure 6-9. Miscellaneous Settings

6.2.7 Advanced

The Advanced tab provides direct write and read access to and from the registers. The access format is 0xabc, where *a* is the page number and *bc* is the register address. Refer to the device data sheet for the page number and register address for a given device register.

7 Register Map Page

The Register Map page lists the different registers available for configuration. Unlike the Quick-start page, there is no device label to select if multiple PMICs are present. In the case of I^2C, the device address is selected in the *Device Settings* below the *Options* tab at the top of the GUI. In the case of SPI, the HW connection to the chip select pin will determine which PMIC the GUI communicates with and whose contents are displayed in the Register Map page.

The Register Map page is intended for direct read and writes to the PMIC registers. The read can be done individually or all at once. Similarly, writing to registers can also be all at once or individually. In the *Immediate Write* mode (option located at the top right of the page), only individual registers are written to immediately with each change in the Field View, change in bits, or change in hexadecimal value. In *Deferred Write* mode, the writing of a single register or all registers is deferred until the **WRITE REGISTER** or **WRITE ALL REGISTERS** button is selected.

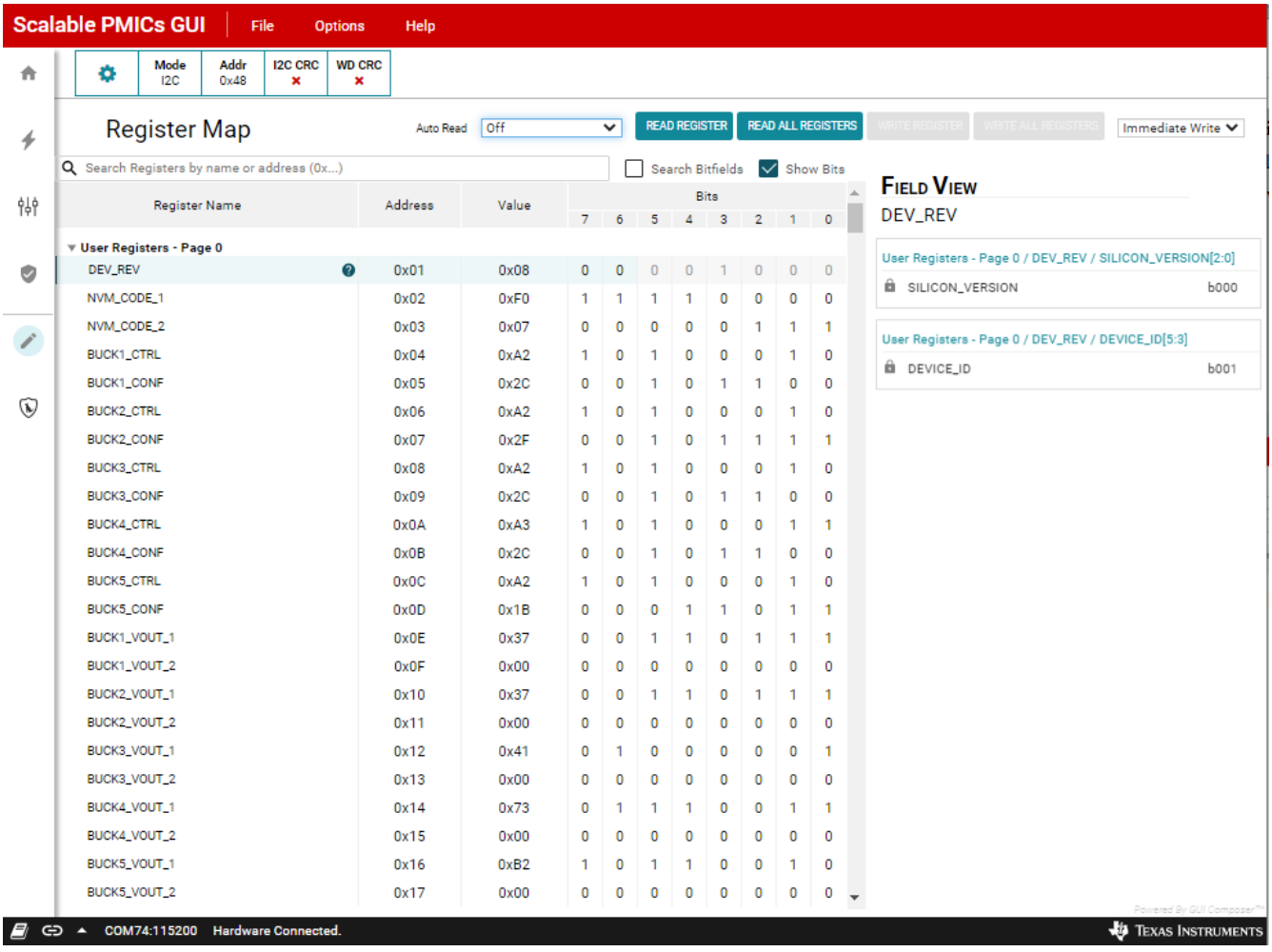

Figure 7-1. Register Map

Note

Although visible from the Register Map Page, not all registers can be edited from this page. Specifically, the interface configuration cannot be changed, and similar with the Quick-start page the Buck frequencies cannot be changed. No error is reported, however, with each write is an associated read. The read will update the display so that writes to protected fields will accurately reflect that the write was unsuccessful.

8 NVM Configuration Page

The NVM configuration page is the main feature of the GUI and highlights the configurability of the PMIC. The configuration is comprised of the static and dynamic (PFSM) settings which are customizable for a broad range of applications. The NVM configuration page also provides the interface to download the configuration into the NVM of a target device. The download can be done after completion of creating a custom configuration, or this can be done with an existing NVM configuration.

8.1 Creating a Custom Configuration

The NVM Configuration page does not require hardware to develop an NVM configuration. Connection with an actual device is needed only when attempting to upload to a target device.

There are three mechanisms available to start development. The first is to use the *Open Configuration* feature,below the File tab at the top of the screen, to open a configuration previously saved with the *Save Configuration* feature below the same File tab. Second, standard configurations are also provided as templates and can be selected below the *Select a template to start with*. Figure 8-1 shows the initial Select view with the upload and template mechanisms selected.

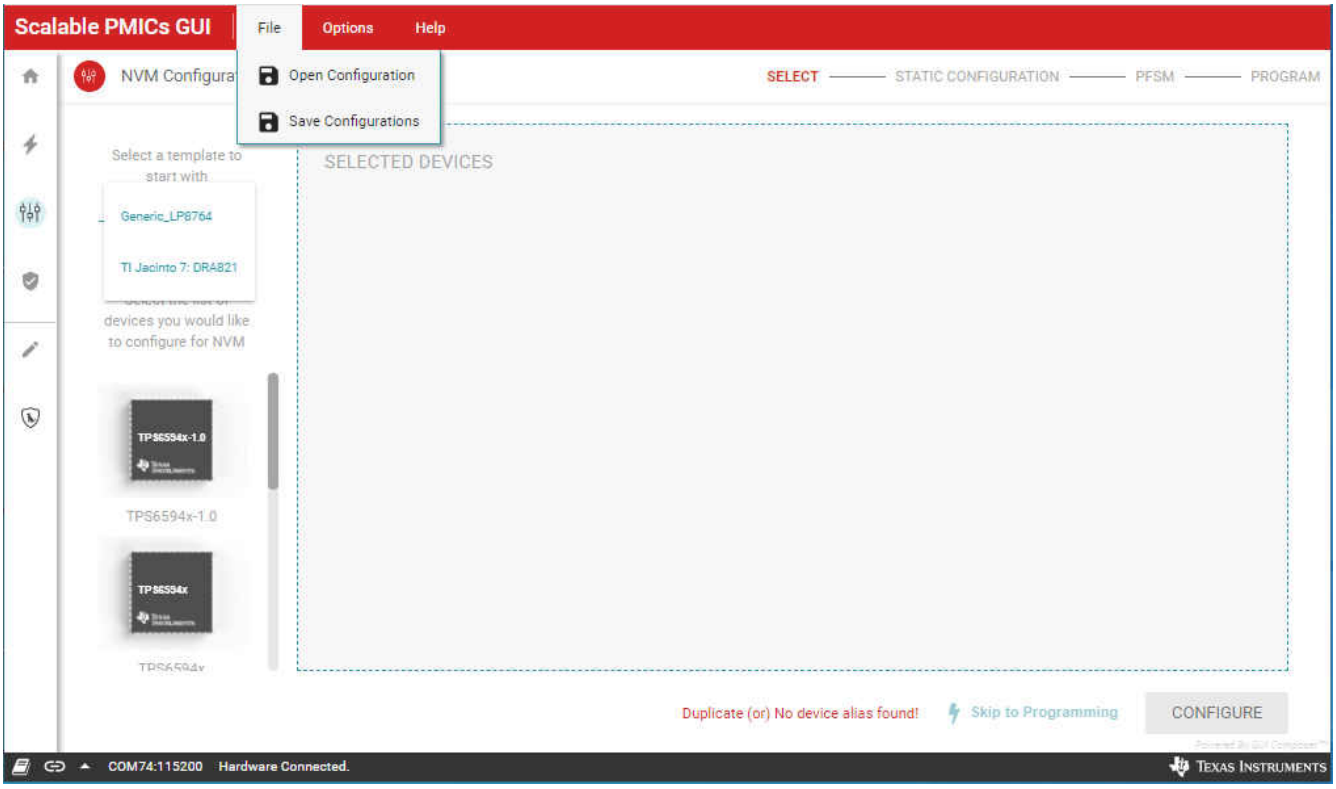

Figure 8-1. Open an Existing Configuration

Finally, device icons are provided on the left-hand-side which can be selected to create a single or multi-PMIC application, as shown in Figure 8-2. As devices are added the Device Name can be edited and the primary/ secondary selection can be made. The GUI requires unique device names and only one primary.

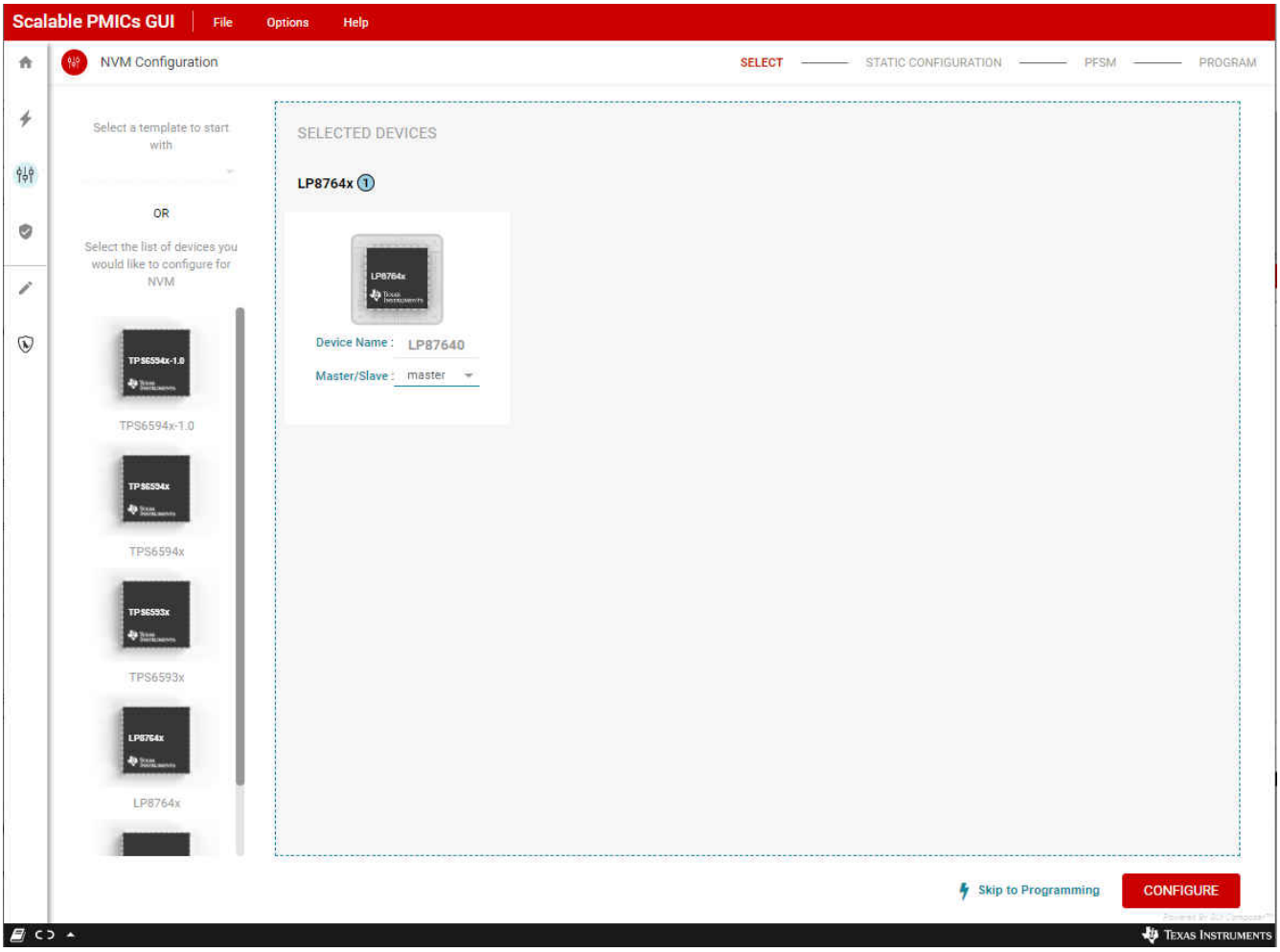

Figure 8-2. Starting from a blank template

Once the configuration is uploaded or the devices for the application selected, the development flow will move through the Static Configuration and PFSM perspectives and then finally to the Program perspective as highlighted in Figure 8-3. Section Section 8.2.1 will describe how to program an existing NVM Configuration using the *Skip to Programming* option found at the bottom of Figure 8-3.

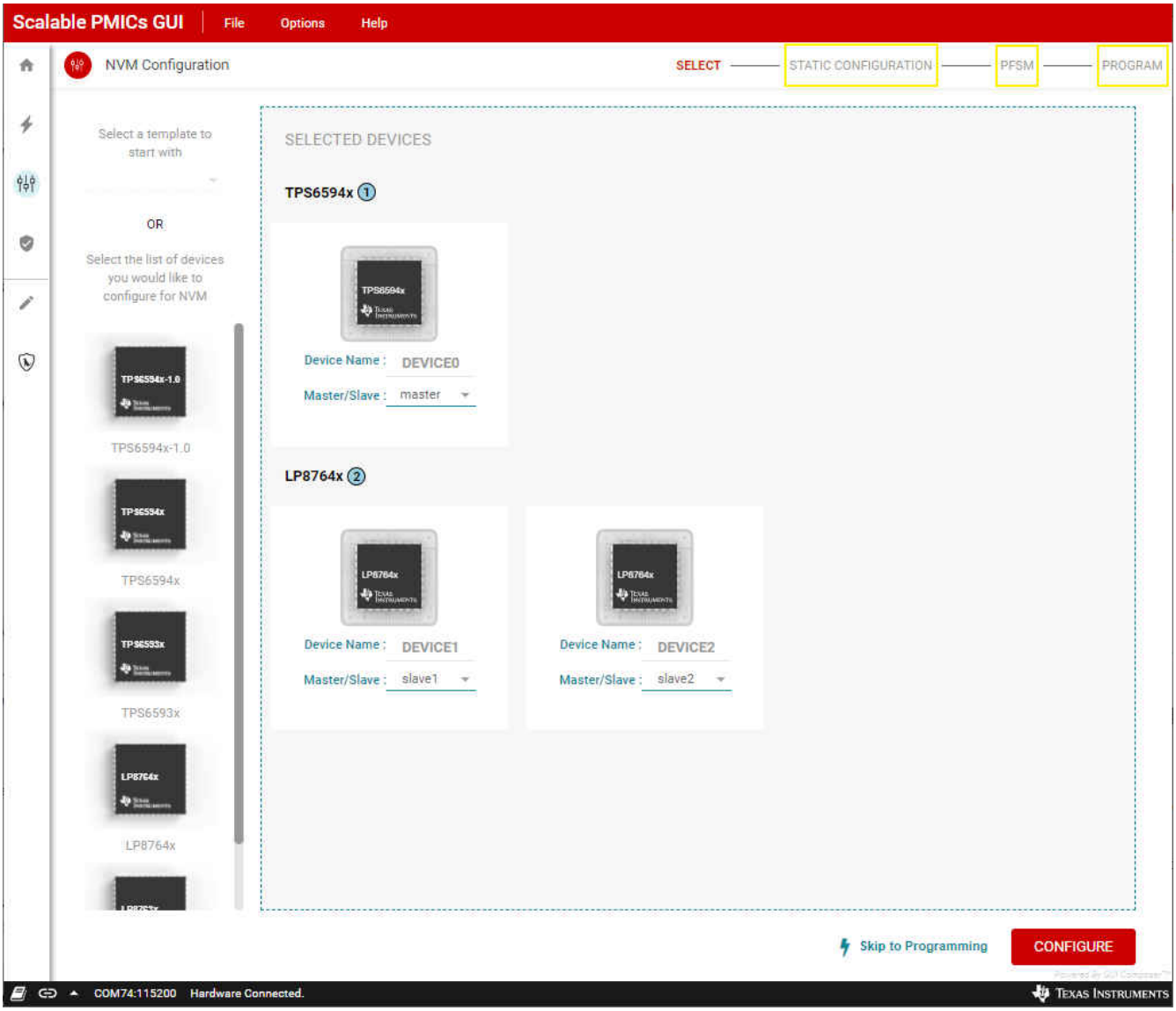

Figure 8-3. Configuration Development Flow

Note

It is not recommended to change or edit the selected devices of an existing configuration, with the exception of the device name. Doing so can break dependencies found in the PFSM. If during development it is discovered that the number or type of devices was defined in error, then a new configuration, restarted from a blank or existing template, is required.

8.1.1 Static Configuration

The Static Configuration perspective provides a similar interface as to what is found in the Quick-start page. The recommended flow is to start with the System Info tab on the far left, updating each block within the System Info tab and then proceeding to the next configuration tab. In a multi-device system, simply select the device to be configured from the Select a device to configure drop-down menu at the top left of the page.

Within each block are a list of registers and fields which directly match the device register and field names. It is recommended to use the data sheet specification to understand and properly set the field values for a given application. Within the BUCK blocks there is an additional graphical selection tool, see Figure 8-4, to abstract the register settings into the use cases also described in the device data sheet. The graphical selection tool will appear in the far right column of the display, providing the Configuration Use case name as well as the recommended inductor value.

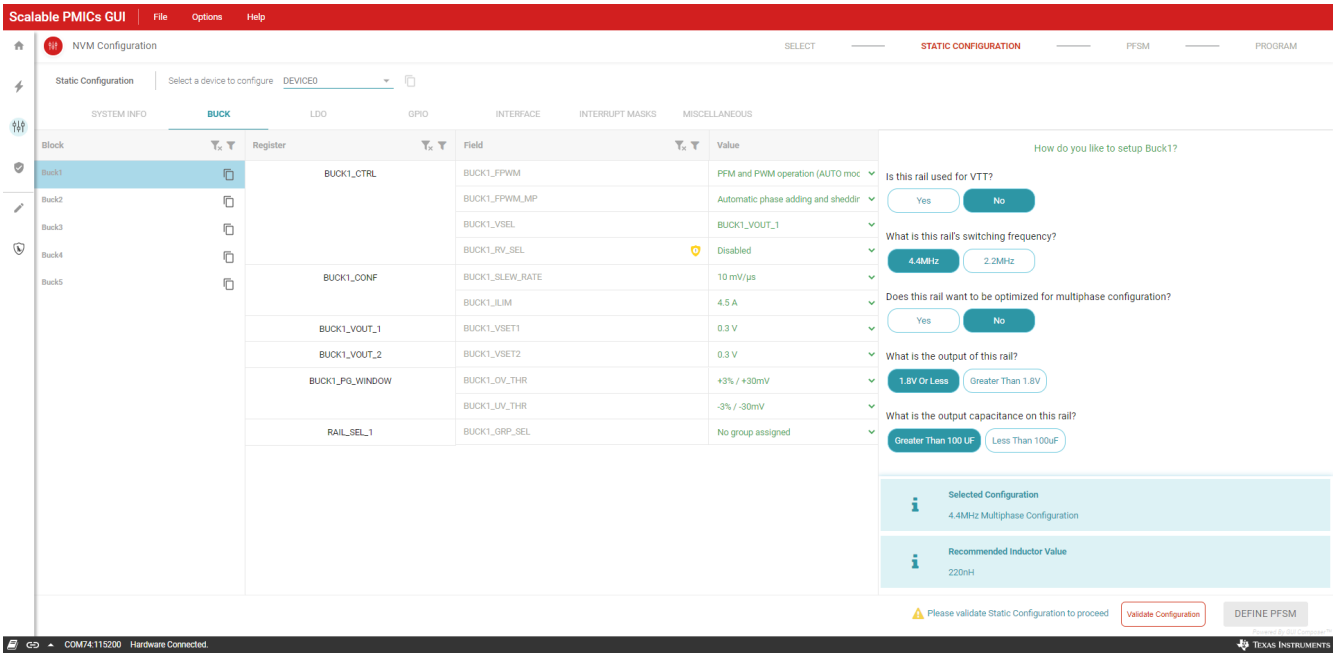

Figure 8-4. BUCK Static Configuration

Note

Only one device is visible at a time. Be sure that all devices in the application are defined.

Within the Static Configuration perspective, the GUI is monitoring the validity of the configuration. Specifically, for multi-device solutions the GUI is making sure that both the power (VCCA and nPWRON) and interface selections match. By clicking on the *Validation Failed* text at the bottom of the perspective a pop-up window, as shown in Figure 8-5, will describe all of the issues which are invalid and preventing the next stage of the development.

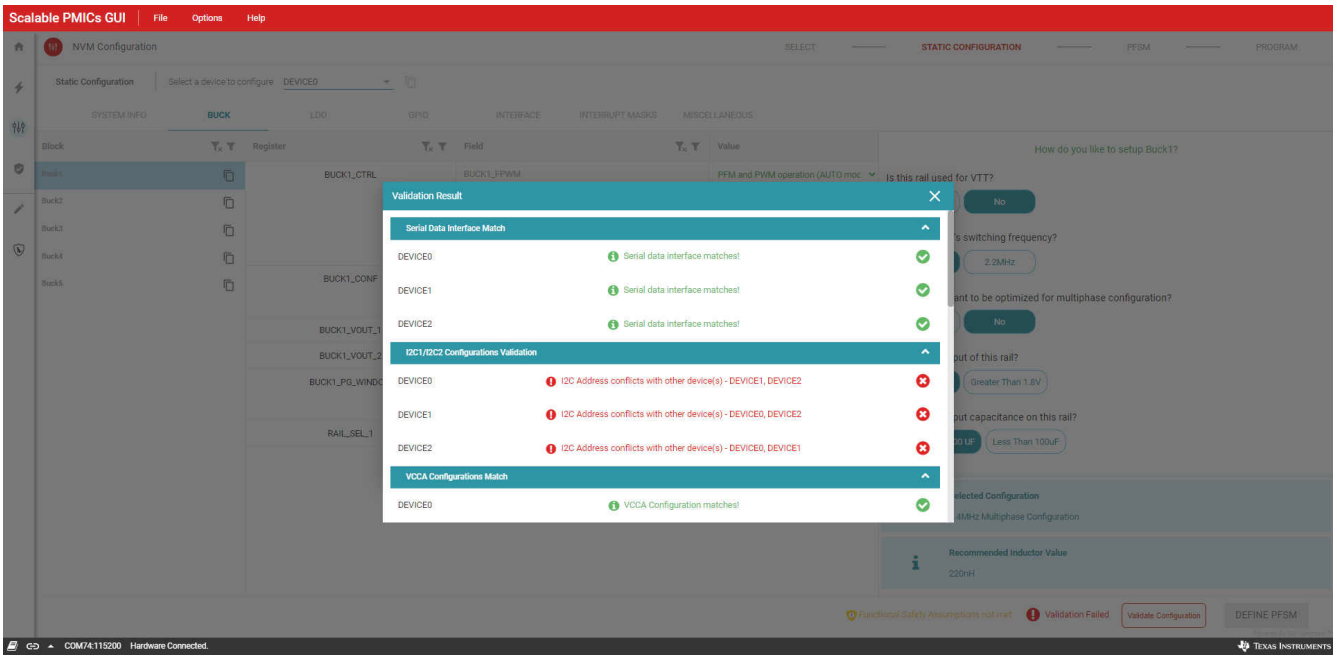

Figure 8-5. Static Configuration, Failed Verification

In this example, all three devices have the same I^2C address(es). After updating each device to a unique address, the GUI now shows Validation Success and the Define PFSM button is now active.

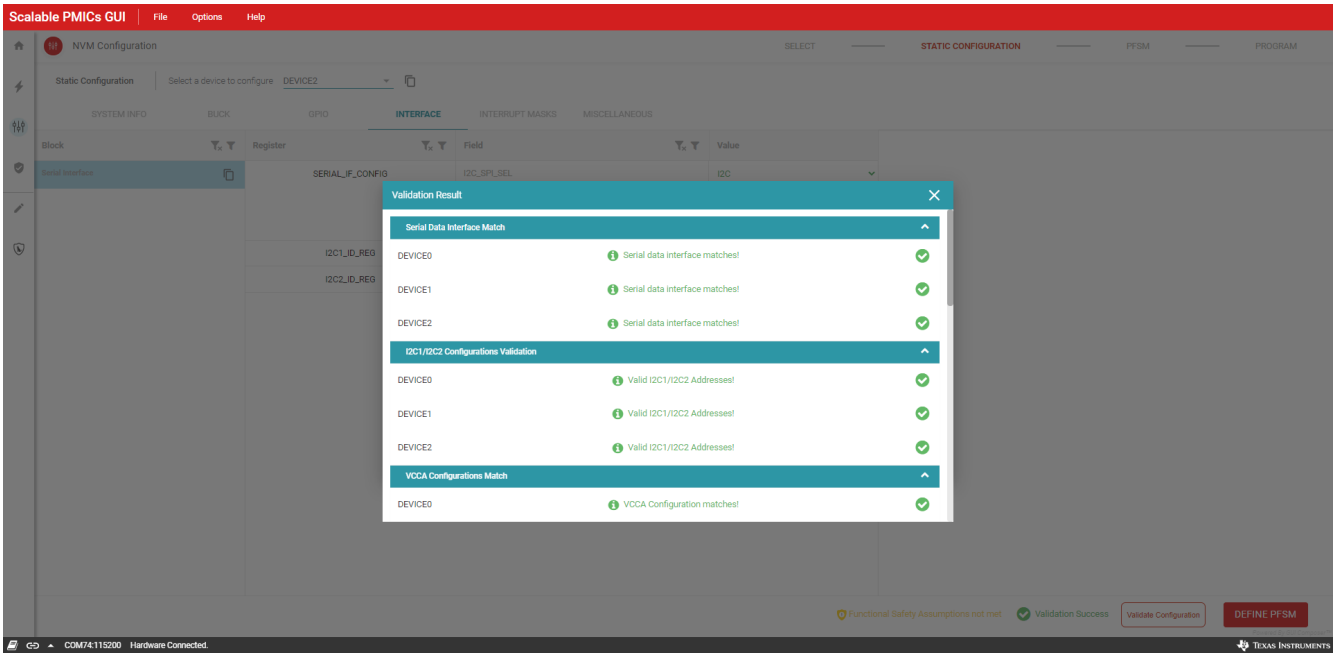

Figure 8-6. Static Configuration, Passing Verification

In addition to the check of the static settings, in devices which support functional safety, there is an additional check of the safety related features. By clicking on the *Functional Safety Assumptions not met* text, a pop-up window, as shown in Figure 8-7, displays. The window lists the parameters related to the device's functional safety assumptions. Refer to the device's functional safety manual and the application's functional safety goals, to confirm that the selected settings meet their functional safety target.

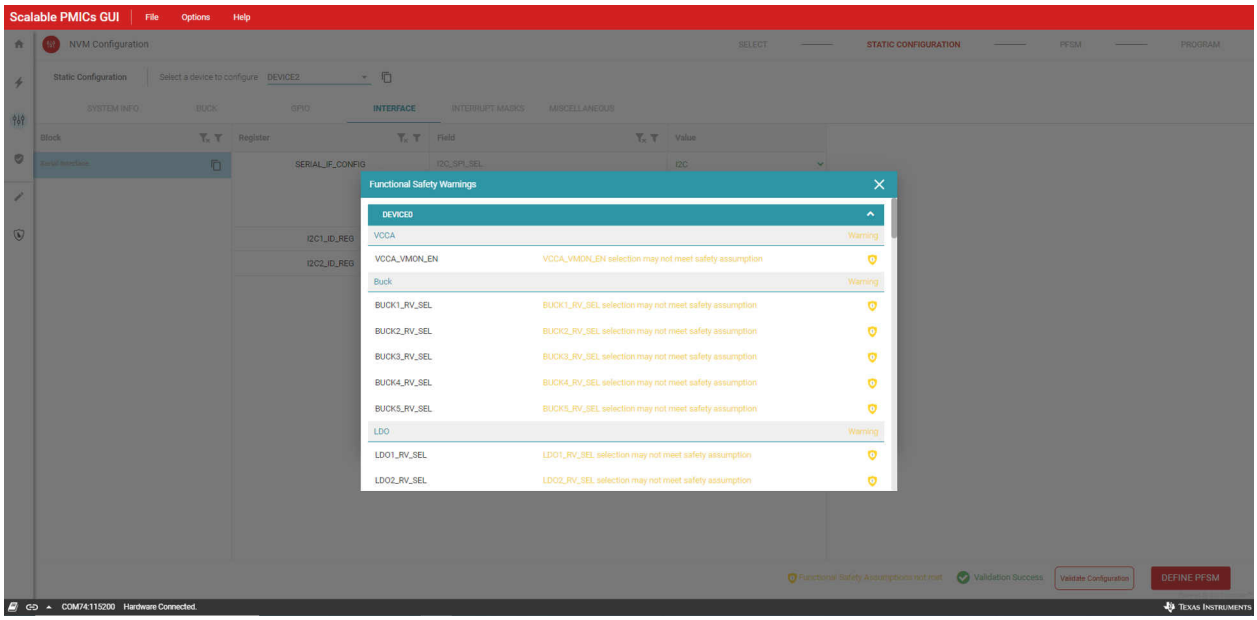

Figure 8-7. Function Safety Assumptions Check

When transitioning to the PFSM perspective from the static settings if any values were not updated, then a warning message will appear. Values that are not updated will appear with a yellow highlight around the value. In this example, only the I^2C addresses have been updated. Texas Instruments recommends reviewing and confirming all settings that are highlighted as unchanged.

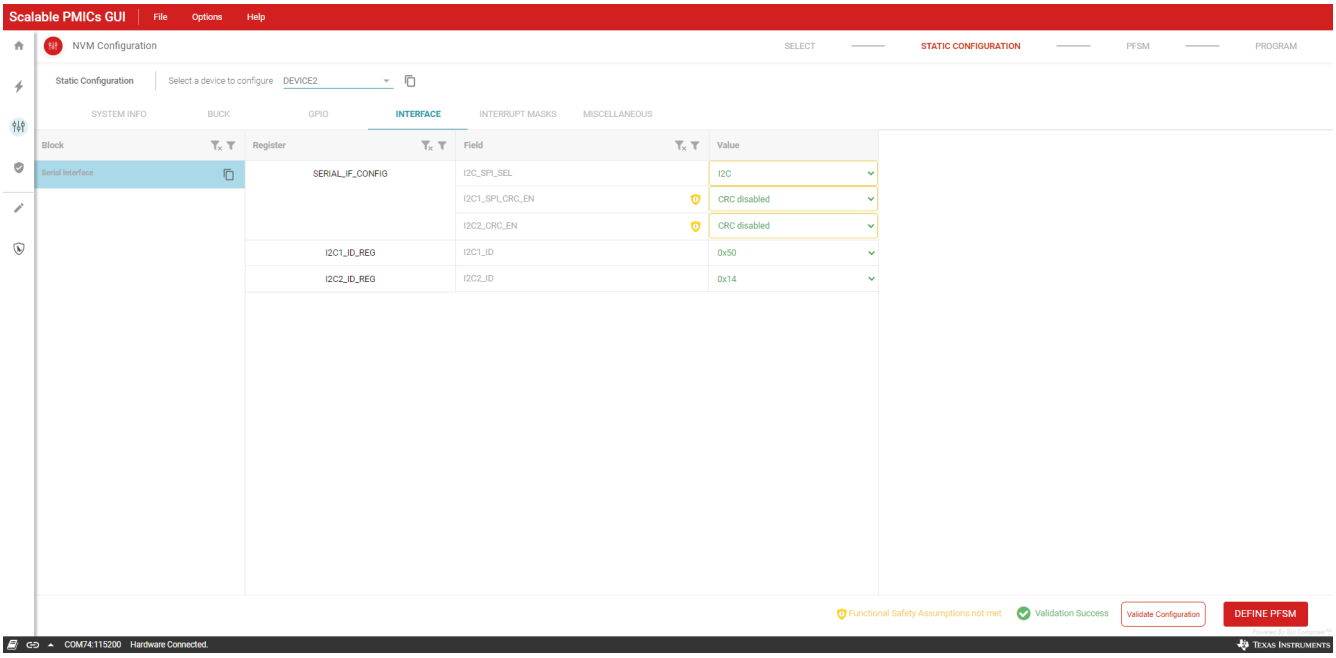

Figure 8-8. Example of non-updated Static Configuration Values

Note

The GUI does not provide an auto-save feature. Save often using the Save Configurations below the File tab shown in Figure 8-9.

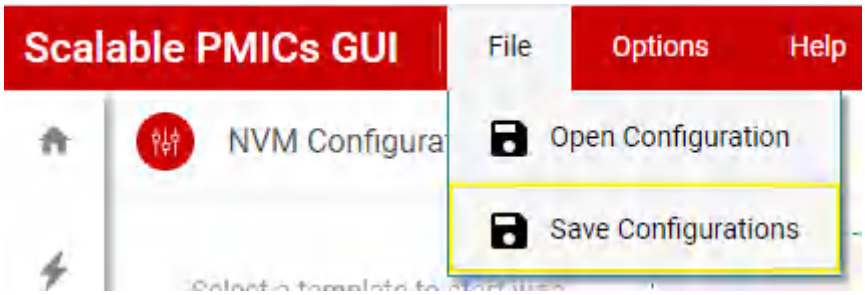

Figure 8-9. Save Often

8.1.2 Pre-Configurable Mission States (PFSM)

While the static configuration was done on a per device basis, the perspective of the PFSM is that of all devices working together. Individual commands are created for each device, but they are grouped in the context of states and transitions of the system solution.

When entering the *PFSM* perspective and no template was chosen from the *SELECT* perspective, a starter template is presented to aid in the development of the PFSM. Figure 8-10 shows the template.

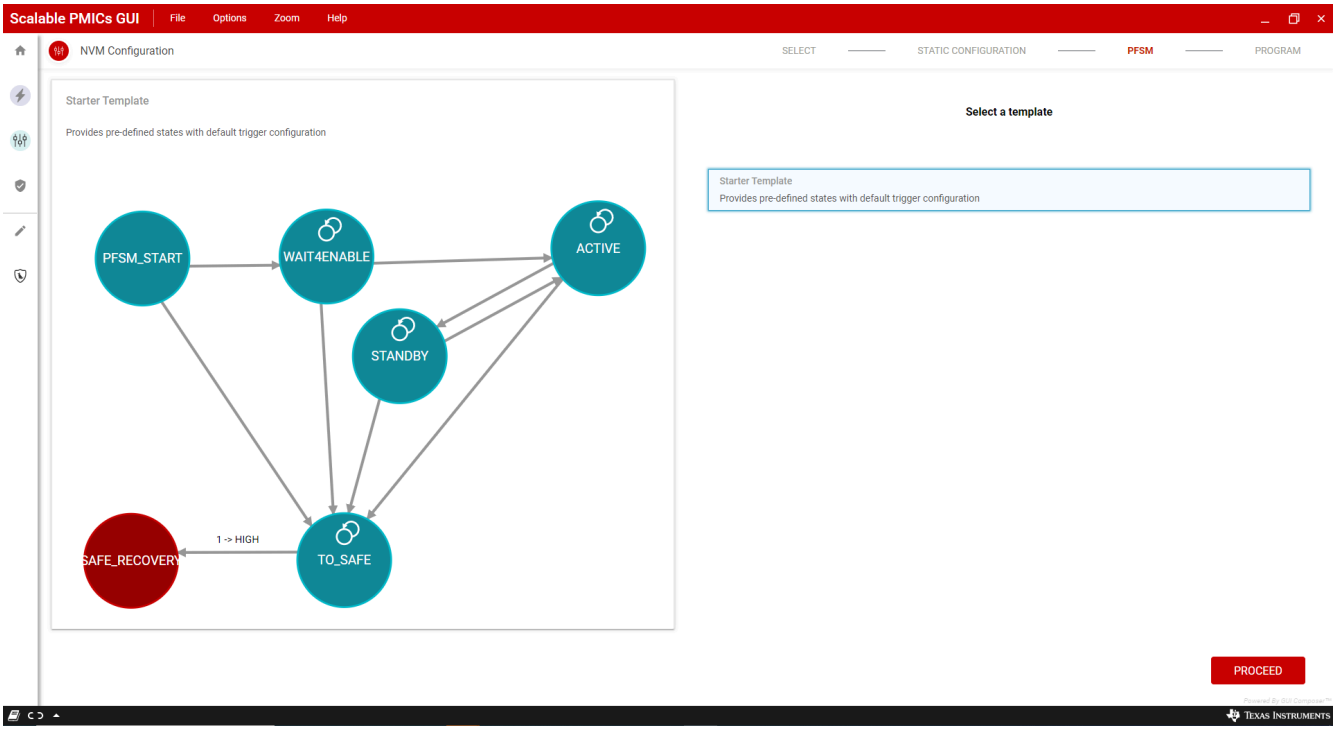

Figure 8-10. PFSM Starting Template

Once the template is selected, then the GUI transitions into the PFSM perspective.

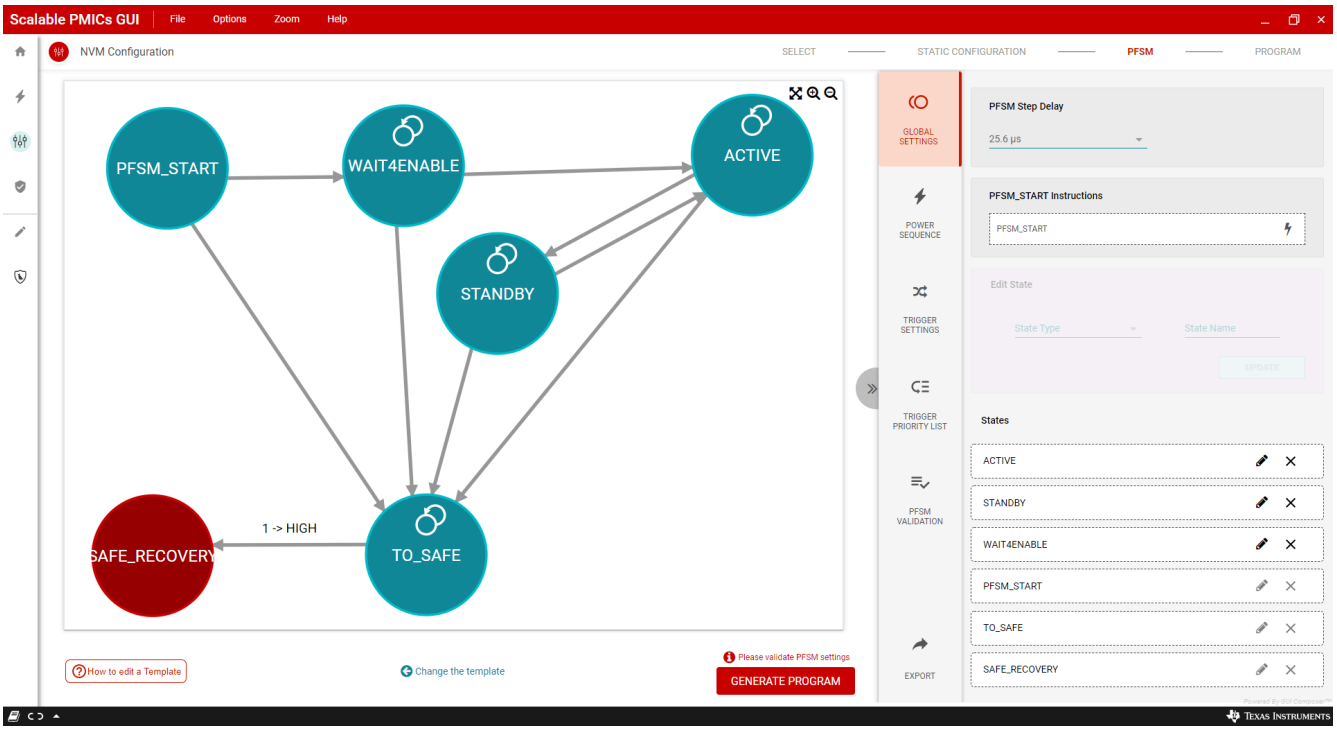

Figure 8-11. PFSM Configuration

The PFSM perspective provides a work space on the left-hand-side to draw the PFSM. How to add and remove states and transitions is described in Table 8-1 as well as within the *How to edit a Template* pop-up window, located in the bottom left corner.

Table 8-1. Actions to Edit the State Machine

Action	Interface
Create a new state	Shift+left mouse click
Create a transition	Shift+left mouse click+drag from within the source state to the destination state
Create a loop	Left mouse click on the loop icon within the state
Move a state	Left mouse click within the state and drag
Delete	Left mouse click on the state, transition, or loop and then press the backspace(delete)
Zoom	Place mouse within the drawing space and scroll
Pan	Left mouse click + drag from any blank space within the work space

Note

The GUI does not provide an auto-save feature. Save often using the *Save Configurations* below the File tab shown in Figure 8-9.

Developing the PFSM can be an iterative and non-linear process. The linear process listed here is only intended to show functionality and a basic work flow. It is possible to return to earlier steps (steps 1-5) at any time and make changes.

1. Create a state diagram
   a. Adding States
   b. Adding Transitions (Triggers)
2. Global Settings
3. Power Sequences
4. Trigger Settings
5. Trigger Priority List
6. PFSM Validation

8.1.2.1 Creating a State Diagram

The state diagram which is developed in the PFSM panel includes the user defined mission states and three hardware states, **SAFE_RECOVERY**, **LP_STANDBY**, and **RUNTIME_BIST**. The user defined mission state **PFSM_START** is required to bridge between the hardware states and the missions states. The **TO_SAFE** state is required as the transition between mission states and the hardware state **SAFE_RECOVERY**. As shown in Figure 8-12, the panel focuses on the mission states and the **SAFE_RECOVERY**, **LP_STANDBY**, and **RUNTIME_BIST** hardware states which can be accessed from the mission state via the PFSM.

WARNING

PFSM_START, TO_SAFE, and SAFE_RECOVERY states are required.

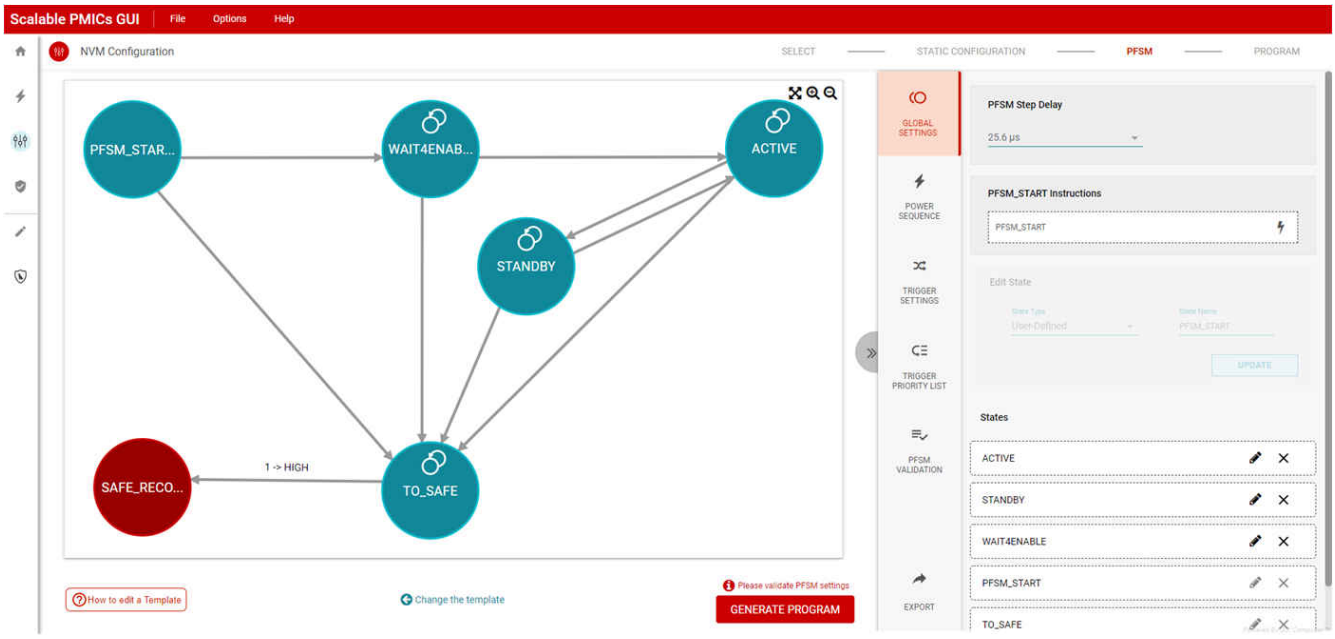

Figure 8-12. PFSM State Diagram Panel

Figure 8-13 shows the complete state machine with the additional transitions associated between the hardware and mission states. The complete state machine can be viewed by selecting the export icon (next to GENERATE PROGRAM) in the lower right corner. Please refer to the device specification for a more detailed description of the states and the transitions.

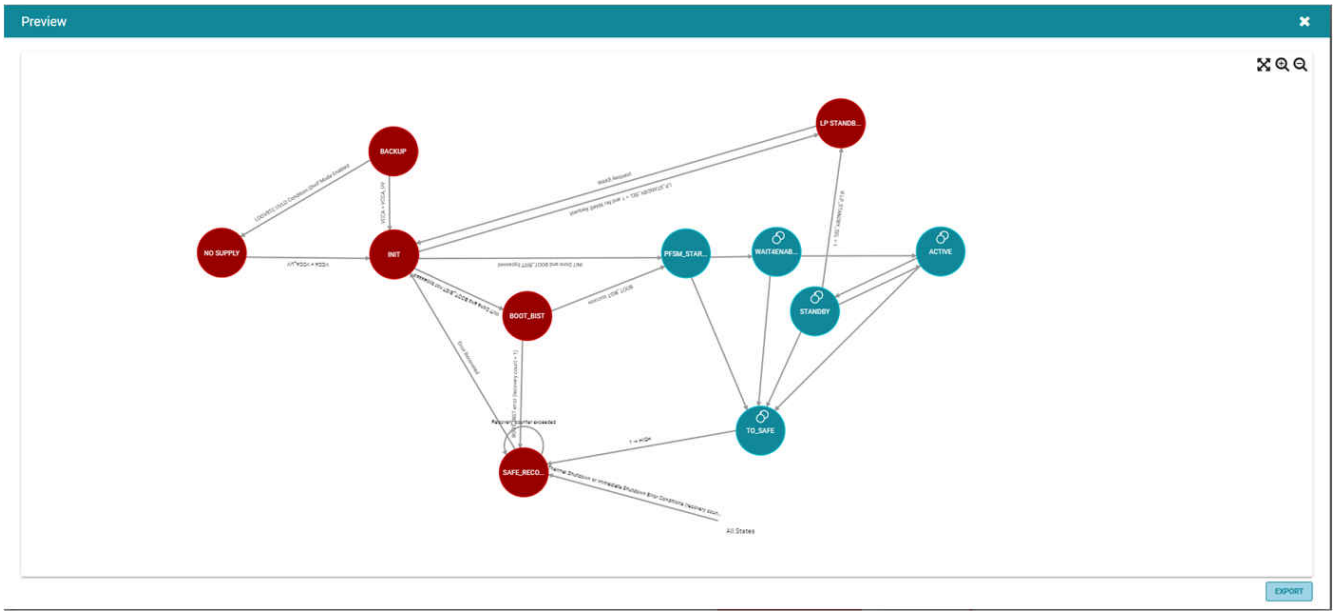

Figure 8-13. Complete State Diagram

8.1.2.2 Global Settings

As states are added they will appear in the *GLOBAL SETTINGS* panel, as shown in Figure 8-14. The names of the states are configurable but the type of state is limited to either a user definition or a Hardware State. Hardware states are already defined within the finite state machine within the PMIC and by definition there is no power sequence associated with transitions to hardware states, with the exception of LP_STANDBY, and no transitions can be defined from Hardware states.

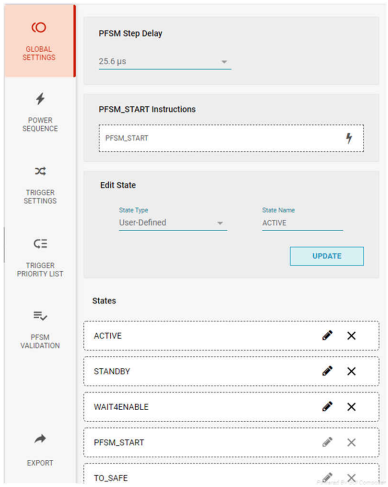

Figure 8-14. Global Settings

The PFSM will always start from the PFSM_START state. This state includes all of the TRIG_SET definitions as well as the initial TRIG_MASK. By default the TRIG_MASK found in the PFSM_START is defined by the arrows leaving the PFSM_START state in the GUI. No arrows can be defined to the PFSM_START state. From the *GLOBAL SETTINGS* the user can edit the TRIG_MASK in the PFSM_START state and also add instructions which will be appended to the PFSM_START sequence after the last TRIG_SET instruction.

The *PFSM Step Delay* setting is also part of the *GLOBAL SETTINGS*. The PFSM Step Delay setting will determine which time interval the GUI will use to attempt to meet the required delays found throughout the power sequences. The actual delays are a function of the desired delay, instruction being used, as well as the PFSM Step Delay. Instruction delays are limited to either 6 or 8-bit multiples of the step delay. In the event that the GUI cannot reach the desired delay time with the existing step delay, or if the step size is actually larger than the desired delay, then the GUI will generate an error during the PFSM validation. Table 8-2 is provided to exemplify the actual delay versus requested delay times as a function of the PFSM Step Delay.

Note

Choosing a PFSM Step Delay which is a common factor of the majority of the delays needed in power sequencing will optimize the memory usage in the device.

Table 8-2. Examples of Requested and Actual Delay Times

PFSM Step Delay (us)	Delay Requested(us)	Delay Instruction	Actual Delay[3] (us)
25.6	2500	DELAY_IMM (8-bit)	2483.2
		REG_WRITE_VCTRL_IMM(6-bit)	2457.6
204.8	40000	DELAY_IMM (8-bit)	39936
		REG_WRITE_VCTRL_IMM(6-bit)	39321
409.6	300000	DELAY_IMM (8-bit)	299827
		REG_WRITE_VCTRL_IMM(6-bit)	294912

8.1.2.3 Power Sequence

Power sequences are dependent upon the target state definition, and therefore it is required to define the state in the global settings before creating a power sequence. Most sequences apply to the transition to a specific state. Some sequences can be made target state agnostic, so that the same sequence can be reused in multiple states. These sequences require that the target state selected be 'ANY' and mapped only to transitions which are self terminating as shown in Figure 8-15 and Figure 8-16.

3

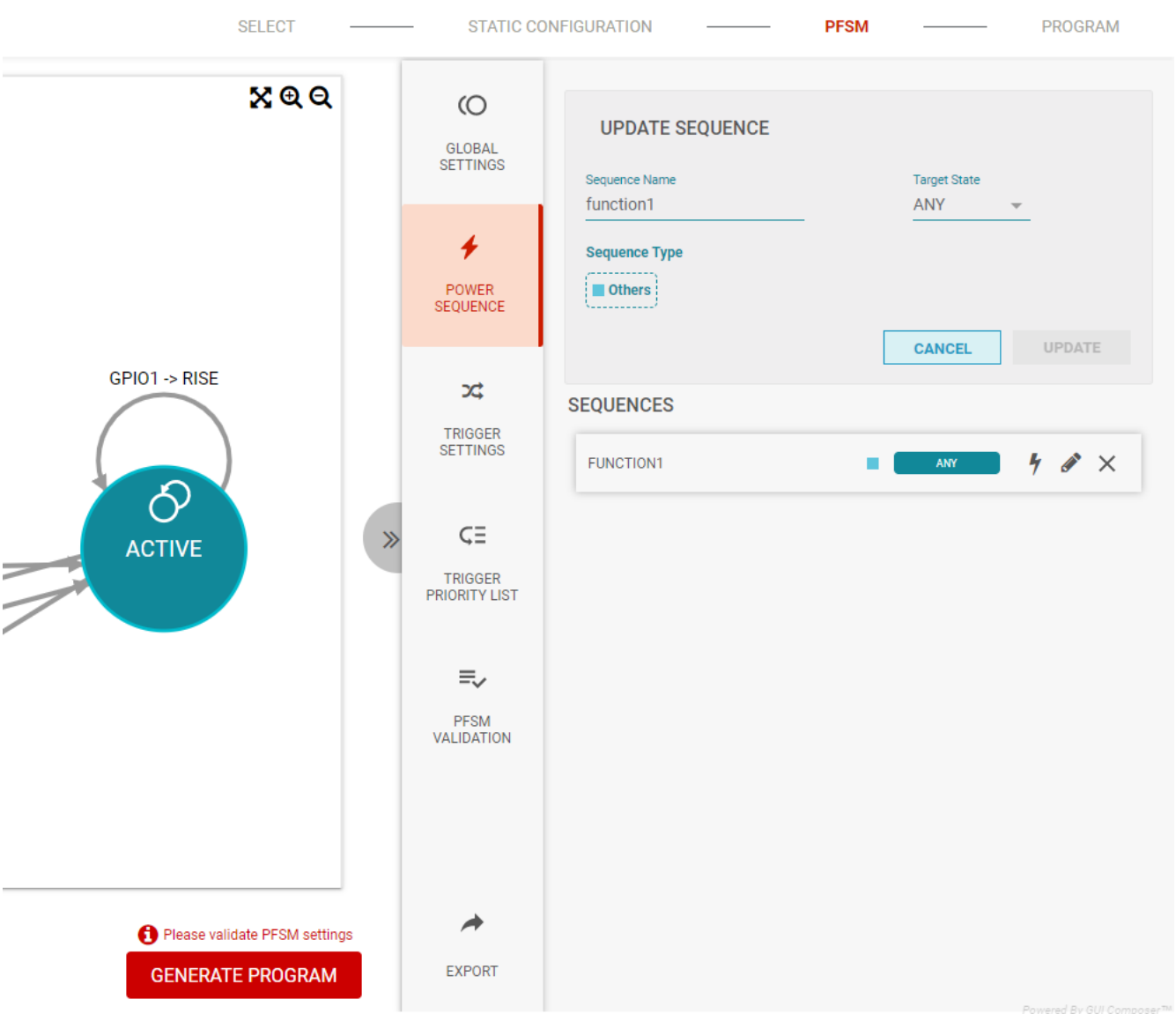

Figure 8-15. Target State set to ANY

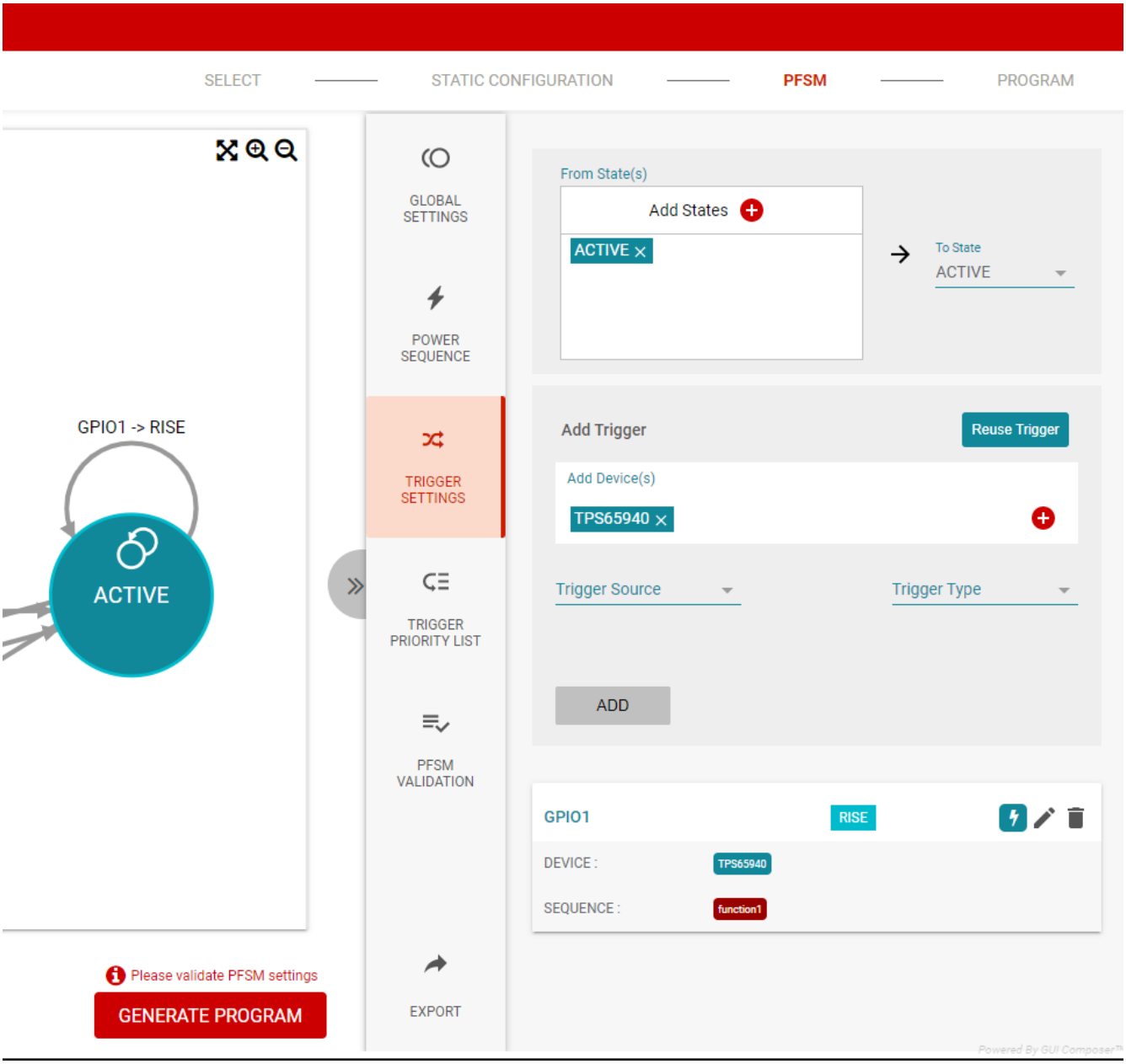

Figure 8-16. Self-Terminating Transition and Trigger Association

In addition to the target state, there are options to select the sequence type. The sequence types *Power Up*, *Power Down*, and *Reset* come pre-populated with ACTIVATE and or DEACTIVATE commands based upon the sequence type. The *Power Down* and *Reset* types are not recommended for use in multi-PMIC applications. Please see the DEACTIVATE descirption, 16, for a discussion on how to properly power down multiple PMICs. All sequence types have the TRIG_MASK pre-populated. These commands are described in Section 8.1.2.3.1.

The *To Safe* sequence type is a new means to identify which sequences lead to the SAFE_RECOVERY state. Functionally this selection has no impact, but it does alert the PFSM validation routine to inspect the sequence for required instructions before entering SAFE_RECOVERY.

Once a new sequence has been created, then the sequence name is added to the list of sequences (see Figure 8-17), and the sequence can be edited (lightning bolt), updated (pencil), or deleted ('X'). The update simply allows the name of the sequence to be updated as well as the target state. Using the edit icon will open a new window as shown in Figure 8-18.

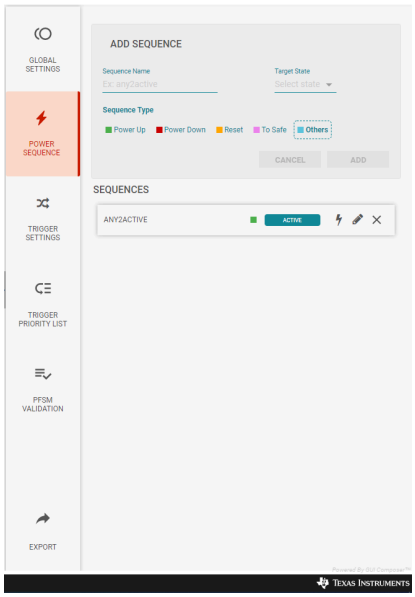

Figure 8-17. How to Add a Sequence

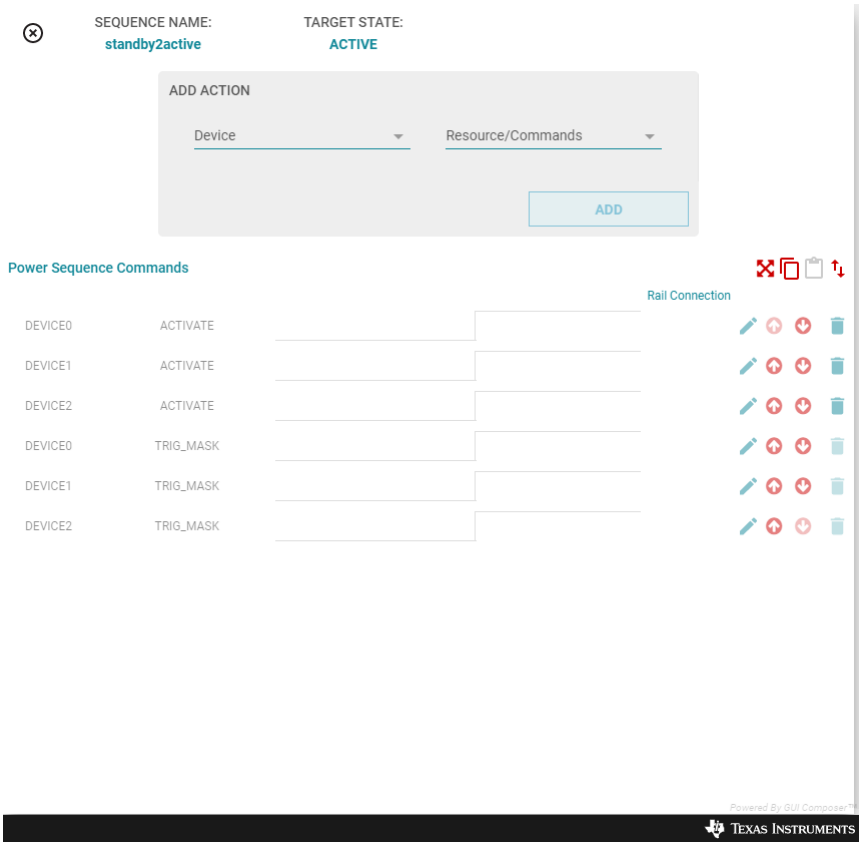

Figure 8-18. Power Sequence Command Window

From this window, commands for each PMIC can be added and the timing relationship between each command is represented. Arrows are provided to move commands within the sequence.

Note

Delays in the *ADD ACTION* or *UPDATE ACTION* windows are relative to the previous action. Delays shown in the Power Sequence Commands list are relative to the start of the sequence.

There is a distinction between commands which have Rail Connections and commands which do not. The commands with Rail Connections are typically representative of physical outputs from the device and voltage monitors. The Rail Connections are what will appear in the power sequence diagram as described in section Section 8.1.2.3.3. The names of the Rail Connections are editable and can be updated to more meaningful names related to the application.

Initially, the trigger masks for each device are provided. As additional commands are added, they will always be placed above the trigger masks. The trigger masks can be moved in the sequence order, but typically these are the last commands in the sequence. The trigger mask at the end of the power sequence is automatically generated based upon the available triggers of the destination state. This mask can be edited to mask triggers from triggering a power sequence and a state change. Before editing the mask. it is recommended to define the triggers as described in the Section 8.1.2.4.

8.1.2.3.1 Power Sequence Resources and Commands

This section lists the different commands and resources available. The description is provided with reference to the PMIC instruction set which is described in the device data sheet.

Resources and Commands

Not all commands are available on all devices. Refer to the device data sheet.

1. BUCKx

   The BUCK resource is an abstraction of the assembly instructions REG_WRITE_VCTRL_IMM and REG_WRITE_VOUT_IMM. Selecting either the VCTRL or the VOUT tab in the update action window will determine which instruction is applied. Within each tab are the various parameters of each instruction.

   BUCKx commands are available for each available BUCK. If BUCKs are multi-phased then the grouping is reflected in the resource name, for example BUCK1_2_3_4, but only the primary BUCK information is shown in the parameter window. In the resulting program, the primary buck will be the only one reflected in the instruction.
2. BUCKx Monitor

   The BUCK monitors are a special subset of the REG_WRITE_VCTRL_IMM and REG_WRITE_VOUT_IMM instructions. The parameters which are not used for this resource are not selectable. The VIN tab is the same but represents the monitor voltage setting.
3. VMONx

   Similar to the BUCKx Monitor, the VMONx is a dedicated voltage monitor found on the LP876x-Q1 family of devices. This resource is only available when the GPIO function is configured to the VMONx function. The VMONx is an abstracted REG_WRITE_MASK_IMM to register address 0×2B (VCCA_VMON_CTRL).
4. LDOx

   Similar to the BUCK and BUCK Monitor Resources, this is an abstraction of the assembly instructions REG_WRITE_VCTRL_IMM and REG_WRITE_VOUT_IMM. Selecting either the VCTRL or the VOUT tab in the update action window will determine which instruction is applied. Within each tab are the parameters of each instruction.
5. nRSTOUT

   The nRSTOUT command writes or clears the nRSTOUT bit found in the MISC_CTRL register. The command is an abstraction of the REG_WRITE_MASK_IMM assembly instruction to the address 0x81, register MISC_CTRL. The data and mask are determined by the selection of unchanged, high, or low. The use of the *unchanged* selection has no impact and only serves as a delay of one instruction cycle.
6. nRSTOUT_SOC

   The nRSTOUT_SOC command writes or clears the nRSTOUT_SOC bit found in the MISC_CTRL register. The command is an abstraction of the REG_WRITE_MASK_IMM assembly instruction to the address 0x81, register MISC_CTRL. The data and mask are determined by the selection of unchanged, high, or low. The use of the *unchanged* selection has no impact and only serves as a delay of one instruction cycle.

Note

Both nRSTOUT and nRSTOUT_SOC are found in the MISC_CTRL register. Instead of using two separate commands; nRSTOUT and nRSTOUT_SOC, a single REG_WRITE_MASK_IMM command can be used more efficiently to set and clear both bits.

7. WAIT

   The wait command provides a conditional branch in the instruction set, similar to an *if* or *while* statement. When the timeout is provided, the wait condition is effectively an *if statement* continuing to the next instruction if the condition is true and jumping to the destination if the condition is false. If the timeout is non-zero, then the PMIC will wait until the condition is true and then execute the next instruction or until the timeout is reached and then jump to the destination. The destination must always be after the wait command because the skip count of the wait instruction is always positive. In version 3.0.0, the timeout value must be achievable with the current PFSM_DELAY_STEP. If the timeout is not achievable the GUI will return an error in the PFSM Validation. Use the PFSM_DELAY_STEP command to update the PFSM_DELAY_STEP.

   Note

   Using a timeout is not recommended with multiple devices as the other devices have no information of how long the wait took if the condition was met before the timeout.

8. JUMP

   The jump command is special implementation of a wait command which has a timeout of *0* and a condition which is always false. The destination must be after the jump command.

9. RESET_BUCKs

   The RESET_BUCKs command is a direct write to the BUCK_RESET_REG register at address 0×87. In this command the resets are per BUCK and each BUCK must be configured even when the BUCKs are multi-phased. The RESET_BUCKs command is translated into a REG_WRITE_MASK_IMM command to address 0×87 to either clear or set bits 0 through 4, representing BUCKS 1-5.

   CAUTION

   The RESET_BUCK command will stop the BUCK switching. This command should only be used in power down sequences.

10. GO_TO_LP_STANDBY

    The GO_TO_LP_STANDBY command is a direct write of *1* to the LDOINT disable bit found in the LDOINT_CTRL register, address 0×21. LDOINT is a self-clearing bit. The GO_TO_LP_STANDBY command is a translated into a REG_WRITE_MASK_IMM command to address 0×21, with a data value of 0×01 and a mask of 0xFE.

11. SET_WD_LONGWINDOW

    The SET_WD_LONGWINDOW command is a direct write to the WD_LONGWIN field found in the WD_LONGWIN_CFG register, address 0×405. This field is for programming the duration of the Watchdog Long Window. The SET_WD_LONGWINDOW command is a translated into a REG_WRITE_MASK_IMM command to address 0×405, with a data value range from 0×00 to 0×FF, and a mask of 0×00. A value of 0×00 is approximately 100,ms while a value of 0×FF is approximately 12 minutes.

12. GO_TO_LONGWIN

    The GO_TO_LONGWIN command is a direct write of *1* to the WD_RETURN_LONGWIN bit found in the WD_MODE_REG register, address 0×406. This command will cause the watchdog to return to the Long-Window after completion of the current watchdog-sequence. The GO_TO_LONGWIN command is a translated into a REG_WRITE_MASK_IMM command to address 0x406, with a data value of 0×01 and a mask of 0xFE.

13. FIRST_STARTUP_DONE

    The FIRST_STARTUP_DONE command is a direct write of *1* to the FIRST_STARTUP_DONE bit found in the RTC_CTRL_2 register, address 0×C3. The FIRST_STARTUP_DONE command is a translated into a REG_WRITE_MASK_IMM command to address 0×C3, with a data value of 0×80 and a mask of 0×7F.

14. INCREASE_RECOVERY_COUNT

The INCREASE_RECOVERY_COUNT command is a direct write of *1* to the INCREASE_RECOVERY_COUNT bit found in the RECOV_CNT_PFSM_INCR, address 0×A5. This bit is self-clearing, so each command increments the recovery counter. The INCREASE_RECOVERY_COUNT command is a translated into a REG_WRITE_MASK_IMM instruction to address 0×A5, with a data value of 0×01 and a mask of 0×FE.

15. ACTIVATE

    The ACTIVATE command is a composite of several commands associated with a power up sequence. These commands include the following:

    a. REG_WRITE_MASK_IMM to ENABLE_DRV_STAT to clear SPMI_LPM_EN.
    b. REG_WRITE_MASK_IMM to VCCA_VMON_CTRL to either clear or set VCCA_VMON_EN, depending upon the value selected from the ACTIVATE command window.
    c. REG_WRITE_MASK_IMM to MISC_CTRL to either clear or set AMUXOUT_EN/REFOUT_EN and CLKMON_EN, depending upon the value selected from the ACTIVATE command window. Included in this instruction is the clearing of LPM_EN.

16. DEACTIVATE

    The DEACTIVATE command is a composite of several commands associated with a power down sequence. This command is not recommended for use in multi-PMIC applications. These commands include the following:

    a. REG_WRITE_MASK_IMM to ENABLE_DRV_STAT to set SPMI_LPM_EN.
    b. REG_WRITE_MASK_IMM to VCCA_VMON_CTRL to either clear or set VCCA_VMON_EN, depending upon the value selected from the DEACTIVATE command window.
    c. REG_WRITE_MASK_IMM to MISC_CTRL to either clear or set AMUXOUT_EN/REFOUT_EN, depending upon the value selected from the ACTIVATE command window. Included in this instruction is the setting of LPM_EN.

The Deactivate command is not recommended for multi-PMIC applications. The control of parameters SPMI_LPM_EN, VCCA_VMON_EN, AMUXOUT_EN/REFOUT_EN, CLKMON_EN, and LPM_EN are abstracted by the ACTIVATE and DEACTIVATE commands. SPMI_LPM_EN sets SPMI in a low power mode which stops SPMI WD (Bus heartbeat). In multi-PMIC applications, the SPMI_LPM_EN must be handled at similar times to prevent SPMI WD failures. Therefore, to mitigate clock variations, setting and clearing of SPMI_LPM_EN must be done early in the sequence of each PMIC.

The LPM_EN parameter puts the PMIC in a low power mode. The intended use case is for the PFSM to set LPM_EN upon entering a low power consumption state. The end objective is to disable the digital oscillator to reduce power consumption. Refer to the data sheet for functions disabled when LPM_EN is set.

Table 8-3. SPMI_LPM_EN, FORCE_EN_DRV_LOW, and LPM_EN example assembly instructions

TO_SAFE	TO_STANDBY	TO_RETENTION	TO_ACTIVE
SPMI_LPM_EN=1, FORCE_EN_DRV_LOW =1	SPMI_LPM_EN=1, FORCE_EN_DRV_LOW =1	nRSTOUT = 0, nRSTOUT_SOC=0	LPM_EN=0, AMUXOUT_EN = 1, CLKMON_EN = 1
nRSTOUT = 0, nRSTOUT_SOC=0	nRSTOUT = 0, nRSTOUT_SOC=0	SPMI_LPM_EN=1, FORCE_EN_DRV_LOW =1	SPMI_LPM_EN = 0
...	...	...	...
LPM_EN=1, AMUXOUT_EN = 0, CLKMON_EN = 0	LPM_EN=1, AMUXOUT_EN = 0, CLKMON_EN = 0	LPM_EN=1, AMUXOUT_EN = 0, CLKMON_EN = 0	FORCE_EN_DRV_LOW =0
			nRSTOUT = 1, nRSTOUT_SOC=1

Note

The EN_DRV, nRSTOUT, and nRSTOUT_SOC pins may not be used in the application but are shown here for completeness. In these examples, VCCA_VMON_EN remains at '1.'

The key requirement for multi-pmic solutions is that the SPMI_LPM_EN should occur first while the LPM_EN should occur last in the sequence. Because the DEACTIVATE command encapsulates both the LPM_EN

and the SPMI_LPM_EN in a set of executions placing the DEACTIVATE at the beginning or the end will violate one of the requirements.

17. ENDRV

    The ENDRV command is a direct write to the FORCE_EN_DRV_LOW bit found in the ENABLE_DRV_STAT register, address 0×82. The ENDRV command is a translated into a REG_WRITE_MASK_IMM command to address 0×82, with a data value of 0×08 or 0×00, and a mask of 0×F7.

18. DELAY_IMM

    The DELAY_IMM command is a direct representation of the DELAY_IMM instruction. The delay specified in this command is applied to all devices. In version 3.0.0, the GUI will no longer automatically adjust the PFSM_STEP_SIZE if the size of the DELAY_IMM cannot be achieved with the existing PFSM_STEP_SIZE. The user is responsible for managing the PFSM_STEP_SIZE for the desired delays. This also applies to all instructions which have a timing component. In the event that the delay cannot be achieved an error is provided in the PFSM Validation.

    Note

    The assembler will optimize DELAY_IMM instructions and combine with the following instruction if that instruction includes a timing parameter.

19. REG_WRITE_MASK_IMM

    The REG_WRITE_MASK_IMM command is a direct representation of the REG_WRITE_MASK_IMM instruction. The REG_WRITE_MASK_IMM command includes a mask to write or clear specific bits without impacting other bits within the register.

20. TRIG_MASK

    The TRIG_MASK command is a direct representation of the TRIG_MASK instruction. The trigger mask will determine which interrupts are enabled and disabled. Using the Automatic trigger will set the trigger based upon the trigger settings from the TARGET STATE.

21. REG_WRITE_IMM

    The REG_WRITE_IMM command is a direct representation of the REG_WRITE_IMM instruction. The REG_WRITE_IMM command overwrites all bits within the register specified.

22. REG_WRITE_MASK_SREG

    The REG_WRITE_MASK_SREG command is a direct representation of the REG_WRITE_ MASK_SREG instruction. The REG_WRITE_ MASK_SREG command includes a mask to write or clear specific bits without impacting other bits within the register. The data is sourced from the specified scratch register.

23. SREG_READ_REG

    The SREG_READ_REG command is a direct representation of the SREG_READ_REG instruction. This command copies the contents of the register to the specified scratch register.

24. SREG_WRITE_IMM

    The SREG_WRITE_IMM command is a direct representation of the SREG_WRITE_IMM instruction.

25. DELAY_SREG

    The DELAY_SREG command is a direct representation of the DELAY_SREG instruction. This delay is different than the other delays found in the resources and commands. The delay is only applied to the device specified.

26. PFSM_DELAY_STEP

    The PFSM_DELAY_STEP is a direct representation of the SET_DELAY instruction which writes to the PFSM_DELAY_STEP register.

27. END

    The END command is direct representation of the END instruction. This can be used to terminate branches created in the PFSM sequence using the JUMP and WAIT commands.

8.1.2.3.2 Sub-sequences

Sub-sequences are groups of commands within the power sequence and are associated with the JUMP or WAIT instructions. Functionally, the sub-sequence is simply a destination label for the JUMP or the WAIT statements

to *jump* to. Graphically, the sub-sequence can be used to group commands and then the entire group can be moved within the power sequence. In Figure 8-19, the WAIT instruction is used to test GPIO1, if GPIO1 is high, then the timeout occurs and the execution jumps to the SKIPBUCK5 label and continues execution. Specifically, if GPIO1 is low then BUCK5 is enabled, if GPIO1 is high, then the regulator is not enabled. Figure 8-20 shows that there are no instructions within the sub-sequence. Instructions placed within or after the sub-sequence will be chronologically equivalent. Again, the main benefit of placing instructions within the sub-sequence is to logically group the commands and then to move them as a group when needed.

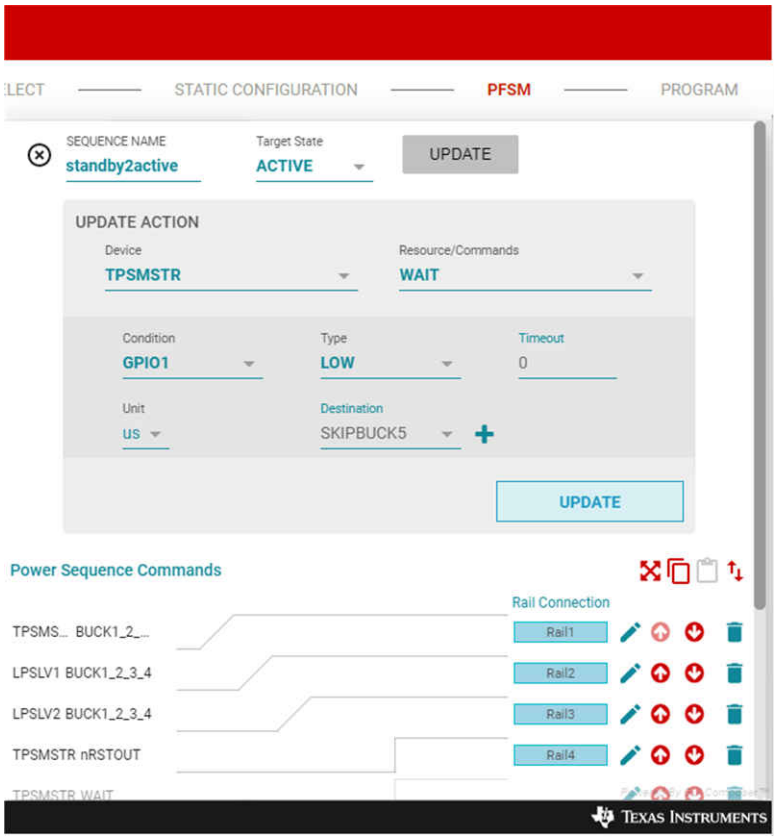

Figure 8-19. WAIT Command Action

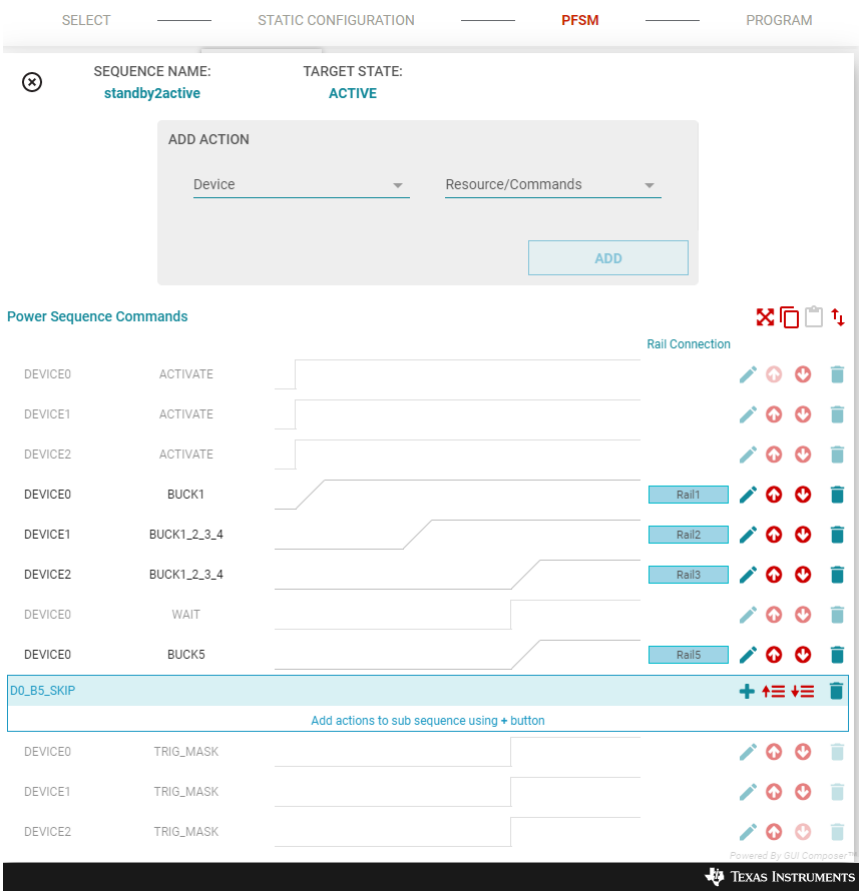

Figure 8-20. Empty Sub-Sequence

Note

WAIT and JUMP statements can only jump forwards in the sequence of commands. The destination can never be placed before the JUMP or WAIT statement.

8.1.2.3.3 Power Sequence Editing Tools

In addition to the four editing tools for each individual command (update, move up, move down, delete), there are four tools available for editing the power sequence in its entirety. From left-to-right, the tools are export power sequence, copy sequence, paste sequence, and reverse sequence (see Figure 8-21). The copy, paste, and reverse sequence tools create a convenient way to copy a power sequence, for example, a power-on sequence, and to paste in a new sequence, then reverse to create a power-off sequence.

Note

The reversal is not applied to the TRIG_MASKS as these are always intended to be at the end of the sequence. The reversal is only of the order of the commands and not the timing relationship or polarity; enable does not become disable.

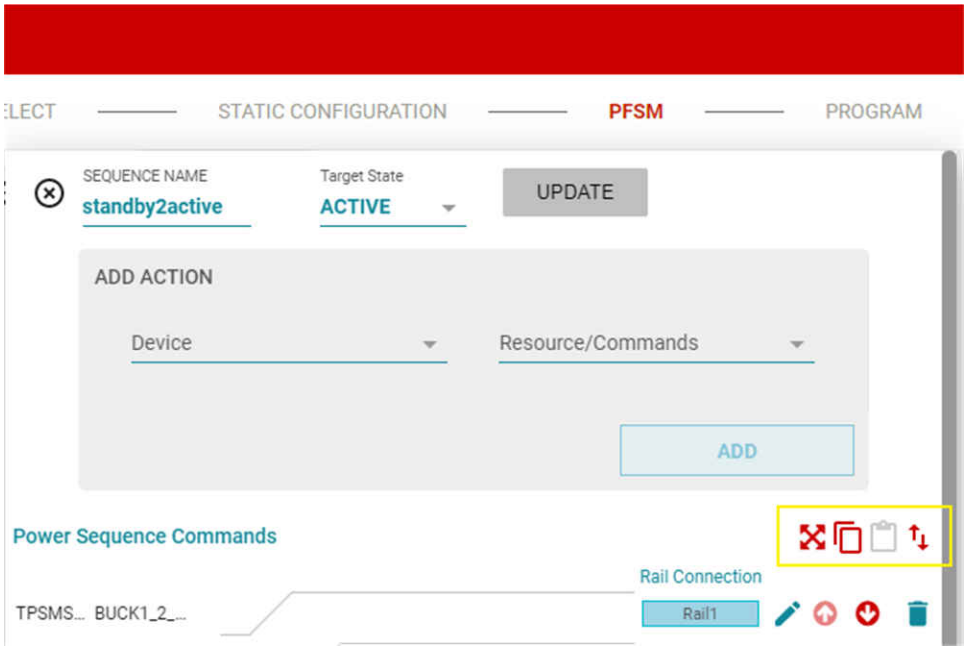

Figure 8-21. Power Sequence Tools

The export power sequence provides a more complete graphical view which can be exported. From this view the user has the ability to provide inputs for the conditional WAIT command and generate timing diagrams for different conditions. As shown in Figure 8-22, either the true condition of the timeout can be selected and the timing diagram is updated appropriately.

Note

Texas Instruments recommends exporting power sequences to confirm that the timing aligns with system requirements.

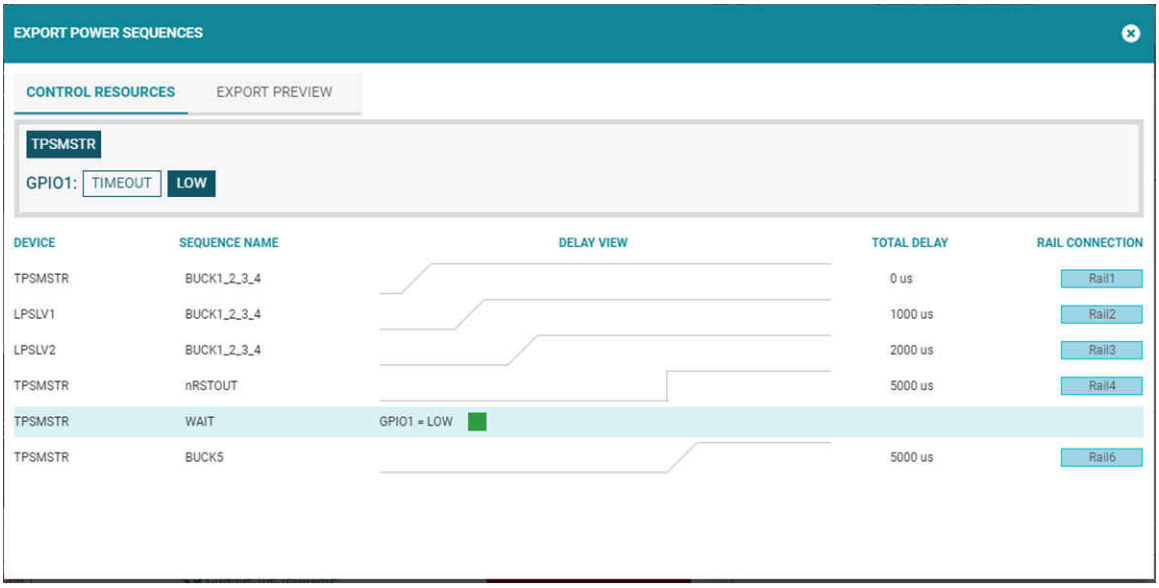

Figure 8-22. Exported Sequence with Variable Conditions

8.1.2.4 Trigger Settings

The maximum number of triggers available is 28. Please refer to the device specification for the usage of reserved triggers. The GUI will ensure that the maximum number of triggers is not exceeded and manage the usage of triggers across devices in a multiple PMIC application.

The trigger settings identify what triggers will move the PMIC operation from one state to another. In the context of the GUI, the arrows between states (or arrows looping back to the same state) must have at least one trigger definition. An example is used to outline the steps configuring the trigger settings. For this example, the *IMMEDIATE_SHUTDOWN* going high on any device will trigger all devices to execute a power down sequence and then transition to the **SAFE** state, which in turn will automatically transition all devices, without power sequence, to the hardware state **SAFE_RECOVERY**, to reset the devices. The following steps are how to setup this example.

1. Select the transition between the **STANDBY** and **SAFE** states, by clicking on the transition in the diagram.

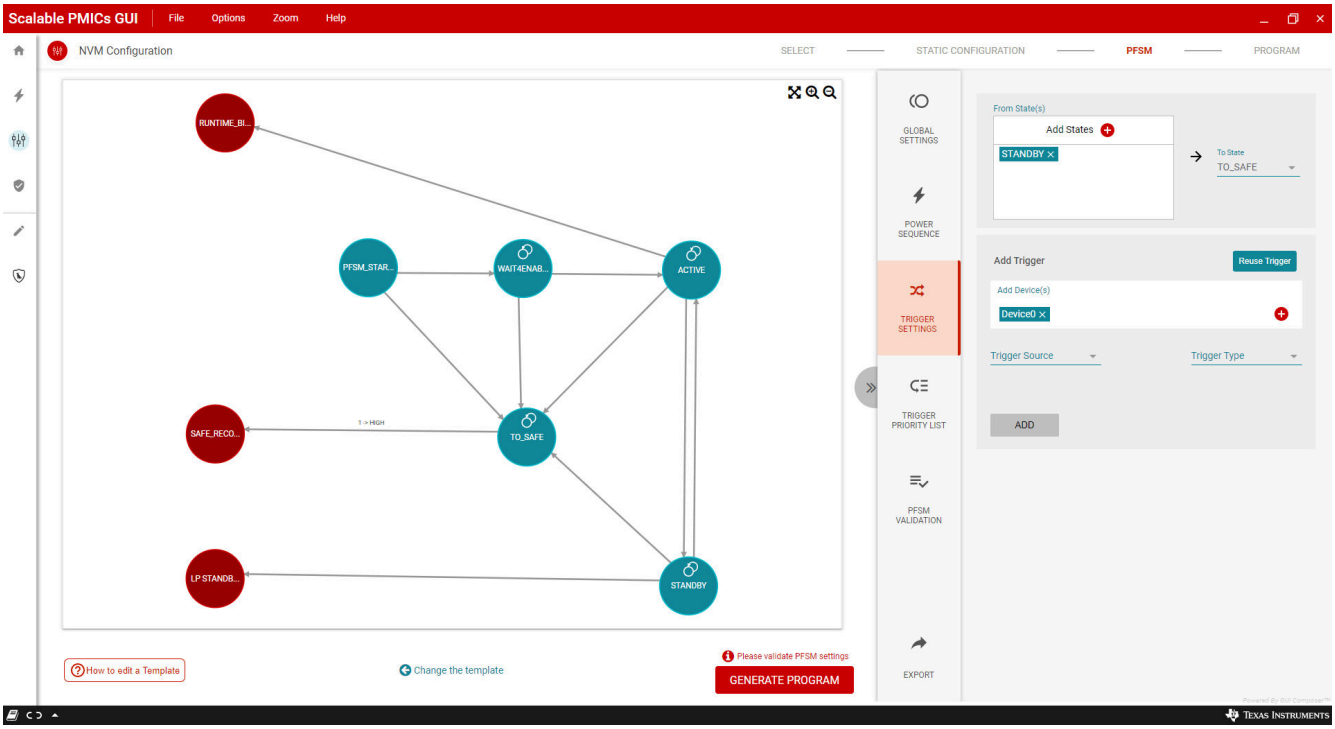

Figure 8-23. Trigger: STANDBY to SAFE

2. In the *From States* add **ACTIVE**, **PFSM_START**, and **WAIT4ENABLE**. [4]
3. In the 'Add Devices' select *Any*, so that all devices appear in the window[5].
4. From the *Trigger Source* drop-down menu, select *IMMEDIATE_SHUTDOWN*.
5. From the *Trigger Type* drop-down menu, select *HIGH*.
6. Select the *Immediate* check box. This means that the trigger can happen immediately and does not wait for the current sequence to finish.

[4] The Reuse Trigger button is an alternative option. Step 2 is omitted. After step 7, the other transitions to the **SAFE** state are selected. At this point the 'Reuse Trigger' button is selected and then the same trigger used for the **STANDBY** to **SAFE** is selected.

[5] While an example, Texas Instruments recommends that the IMMEDIATE_SHUTDOWN trigger be present in all devices. For other triggers, it is acceptable that the trigger is present in only one device.

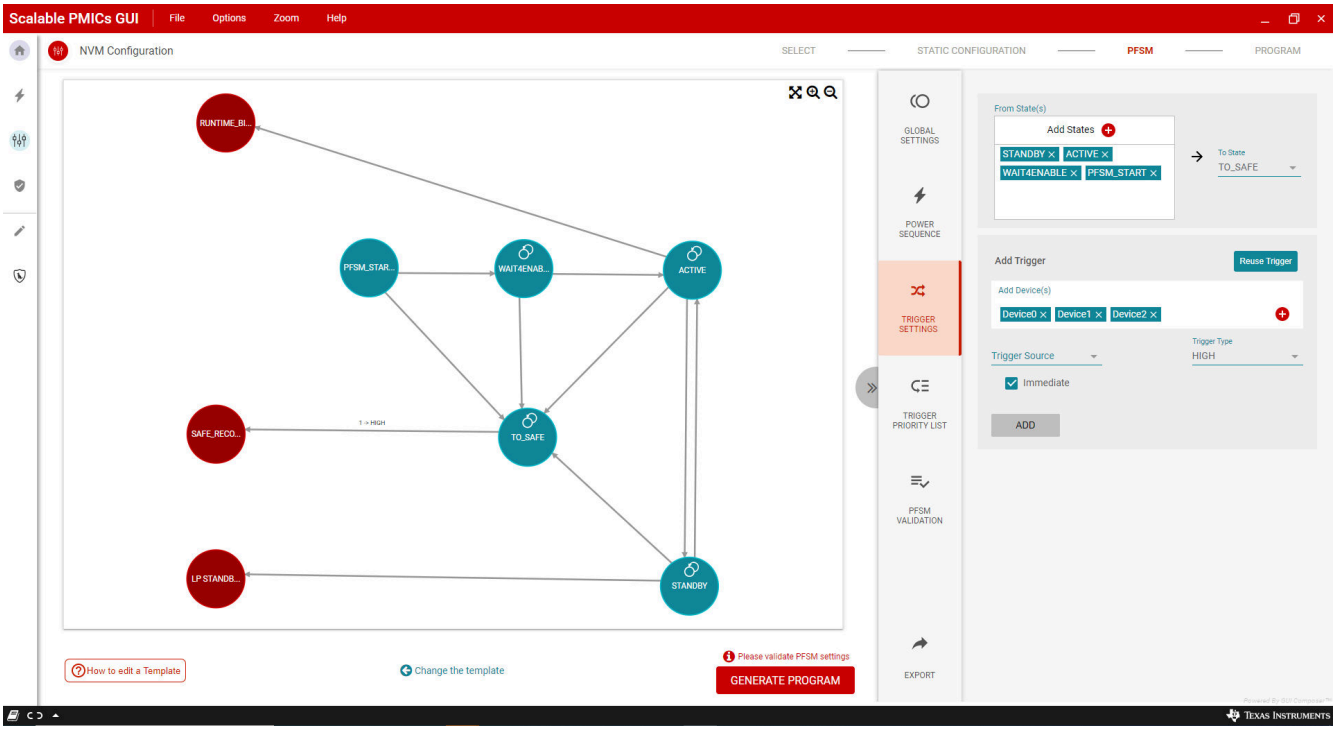

Figure 8-24. Trigger: STANDBY to SAFE (continued)

7. Click *ADD*
8. Now click on the transition between **SAFE** and **SAFE_RECOVERY**. No action is required as this trigger is populated automatically.

Note

Since **SAFE_RECOVERY** is a hardware state the trigger will not have an associated power sequence. This is indicated by the EXT attribute in the trigger description. Similarly, the **RUNTIME_BIST** will also have the EXT attribute.

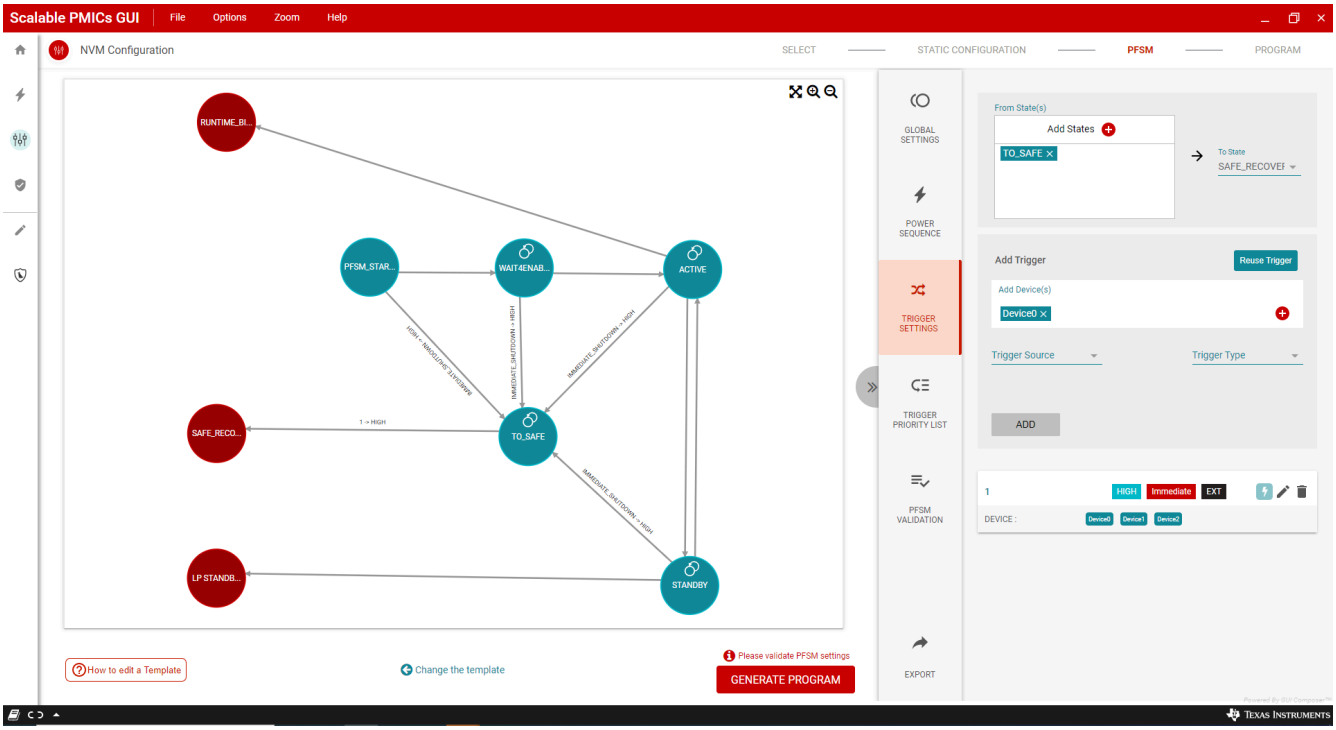

Figure 8-25. Trigger: SAFE to SAFE_RECOVERY

At the bottom of the TRIGGER SETTINGS, highlighted in Figure 8-25, is a summary of the trigger(s) associated with a transition. A scroll bar is provided to see the bottom of the pane. The last step is to associate a power sequence with the transition. Since SAFE_RECOVERY is a hardware state, the EXTERNAL flag is set and no sequence is needed in the transition to SAFE_RECOVERY. The transitions to SAFE, however, do require a sequence. Making the association between the trigger and the sequence is listed in the following instructions.

1. Click on the transition between **STANDBY** and **SAFE** (or **ACTIVE** and **SAFE**).
2. Click the lightning bolt icon. A window displays showing all of the sequences which have the same destination or target state. In this case there is only one, *any2safe*.

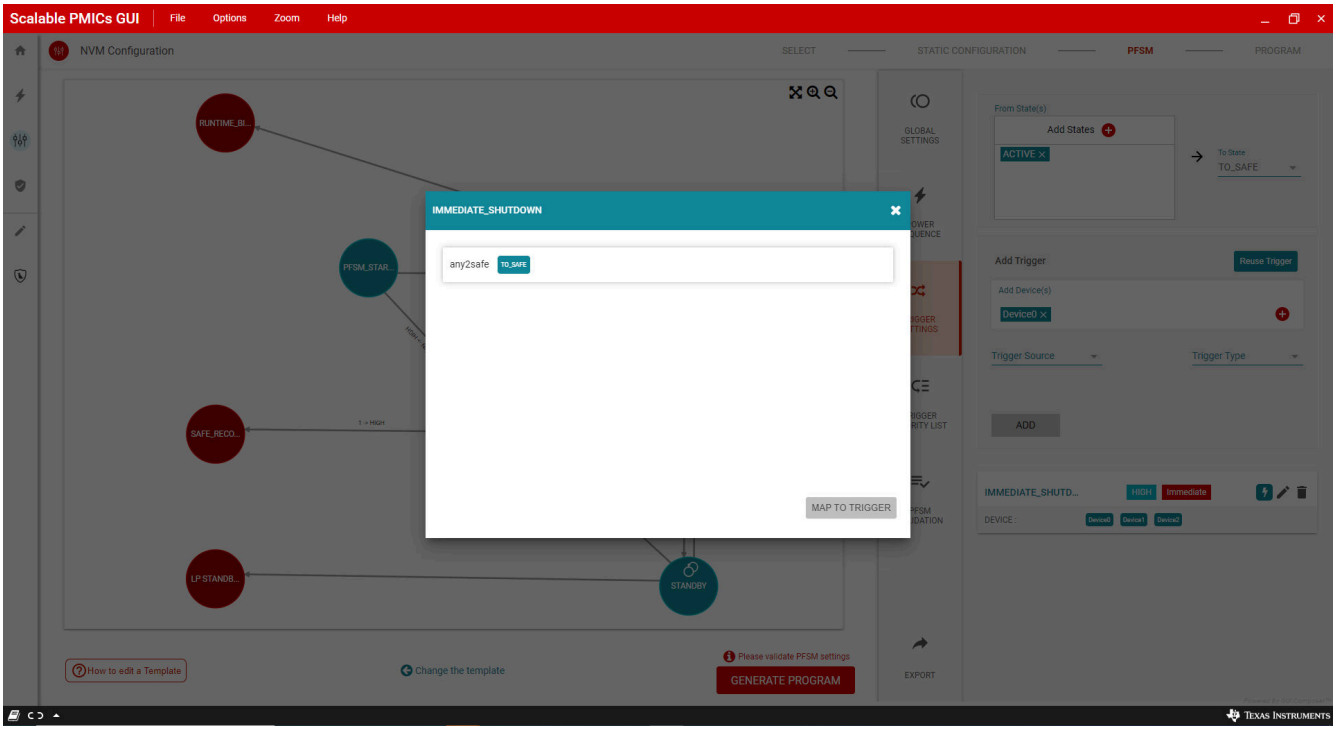

Figure 8-26. Mapping a Sequence to a Trigger

3. Select the sequence and then click *MAP TO TRIGGER*. The Trigger Setting for the transition is now complete.

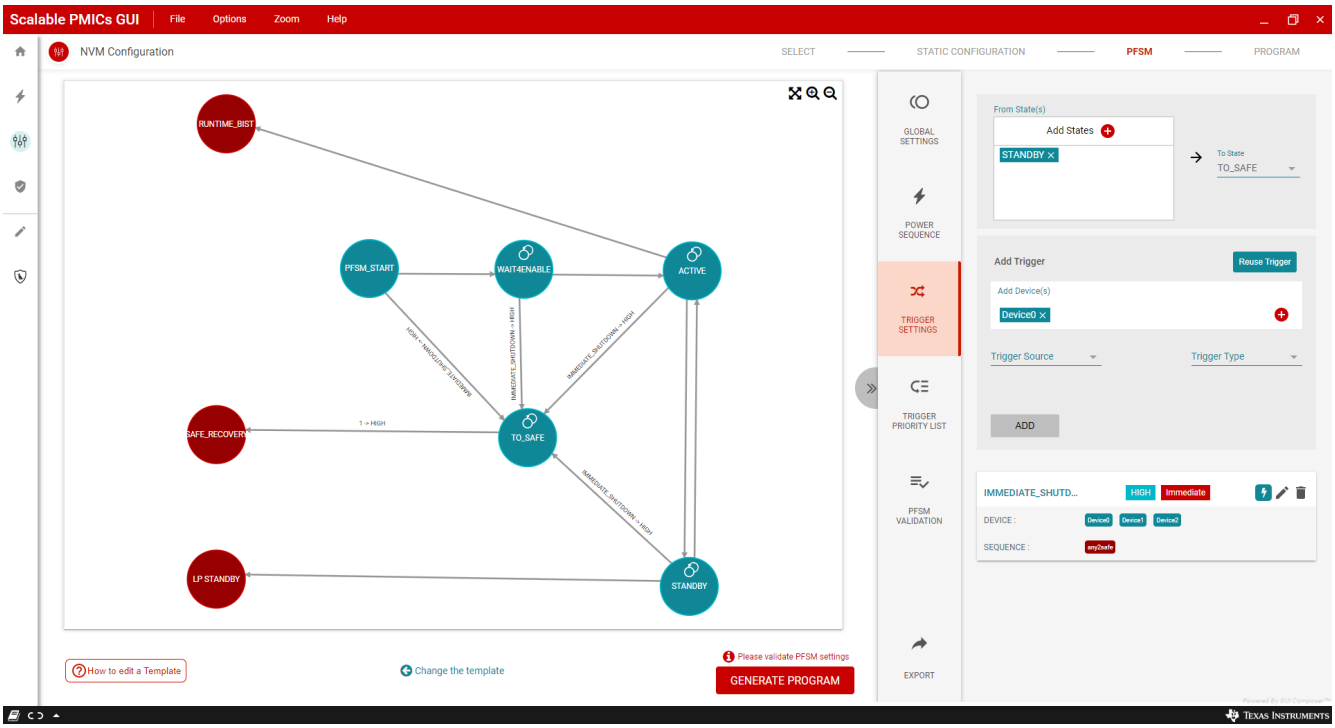

Figure 8-27. Completed Trigger Settings

Once all of the transitions have been assigned triggers and all of the triggers have been associated with power sequences, then the default TRIGGER MASKS within the power sequences will be updated. Please note the various components and relationships discussed to this point:

1. States

   The states are either hardware or mission states.

2. Transitions

   Transitions have a source and target state. Transitions can have multiple triggers but require at least one. Transitions to the same target state can share the same trigger.
3. Triggers

   Available triggers are defined in the device specification. Each trigger can have only one power sequence associated with it. Multiple triggers can share the same power sequence. In the special case that the transition target state is a hardware state, the trigger type is EXT and there is no power sequence.
4. Power Sequences

   Power sequences are defined in terms of the target state and therefor can potentially be associated with multiple triggers or transitions. Within the power sequence is the trigger mask. The automatic trigger mask is defined by all transitions from the target state and the triggers defined in those transitions. Manual trigger masks can be used to create custom trigger masks. In the TRIG_MASK command there is an option to select either an automatic or a manual trigger mask.

8.1.2.5 Trigger Priority List

Triggers are initially prioritized based upon the TRIG_SEL value for each trigger as defined in the data sheet. Lower values have higher priority. It is important to confirm that the priority within the TRIGGER PRIORITY LIST matches the desired priority of the application. Figure 8-28 shows the priority list from the simple example in the previous section. Click the arrows within the list to move triggers up and down in priority.

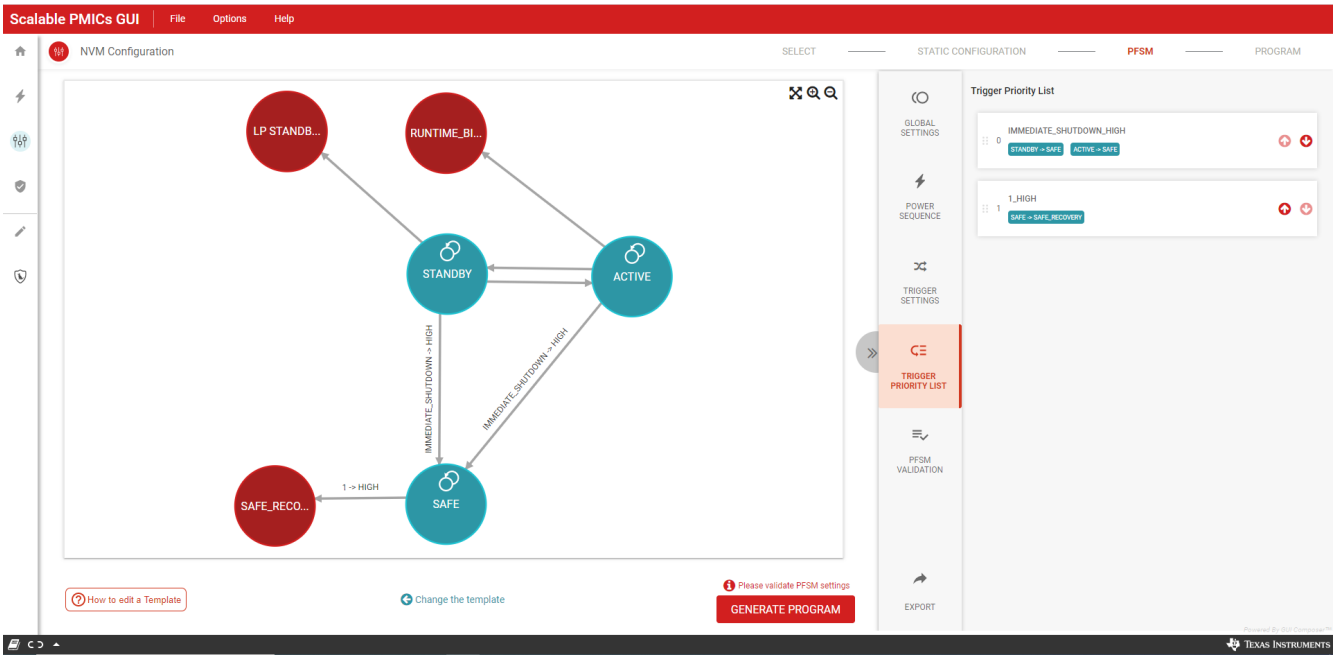

Figure 8-28. Trigger Priority List

8.1.2.6 PFSM Validation

The TRIGGER PRIORITY LIST is the last component to building a PFSM configuration. When the configuration is complete, the PFSM validation is made available to check and validate the PFSM content. Within the PFSM Validation view, click the *Validate PFSM* button.

Within the PFSM view, a list of results will be displayed. Any errors or warning will be accompanied by instructions and recommendations. Errors and warnings will not prevent program generation but indicate a potential risk in device performance within the application. It is recommended to address all errors as this can prevent proper compilation or file generation. Once all errors have been addressed, click the *GENERATE PROGRAM* button to move to the PROGRAM perspective.

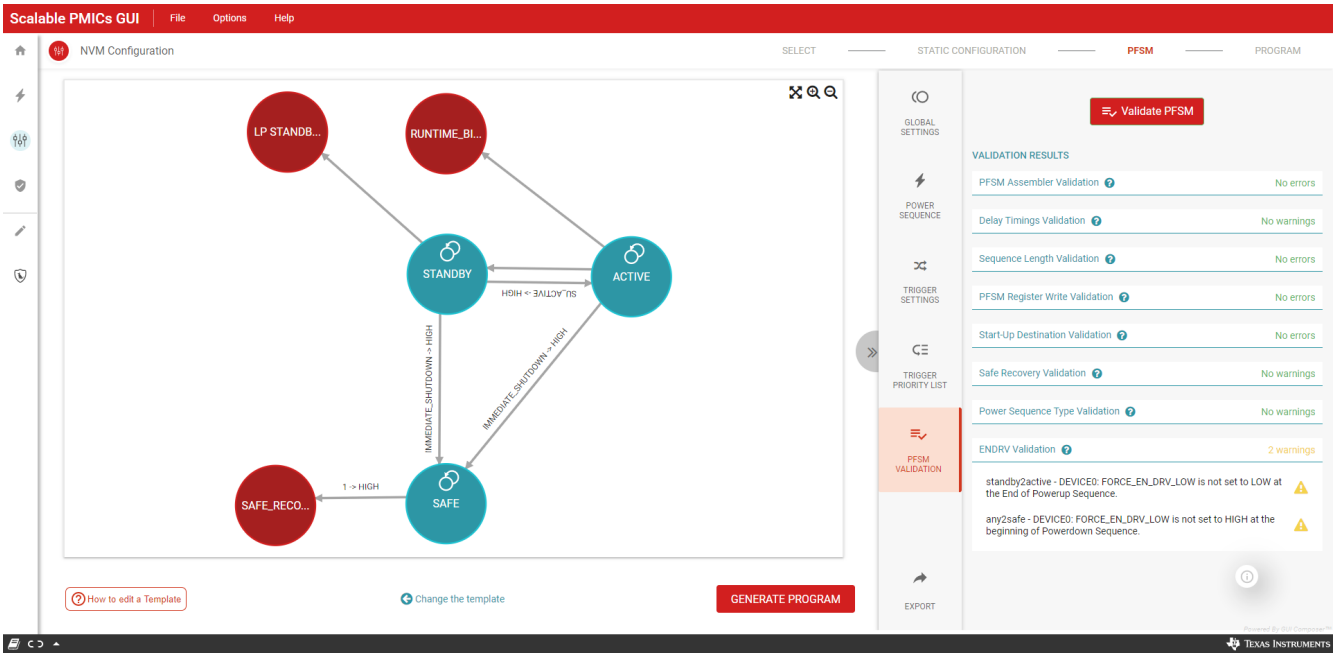

Figure 8-29. PFSM Validation Results

8.2 Program

Note

Before programming is a good time to save the configuration, see Figure 8-9

The program page shows the results of the generated program in the *Generated Program* tab, as shown on the left side of Figure 8-30. This is a text format file and a scroll bar is available to view the entire content. On the right side are the control mechanisms for programming.

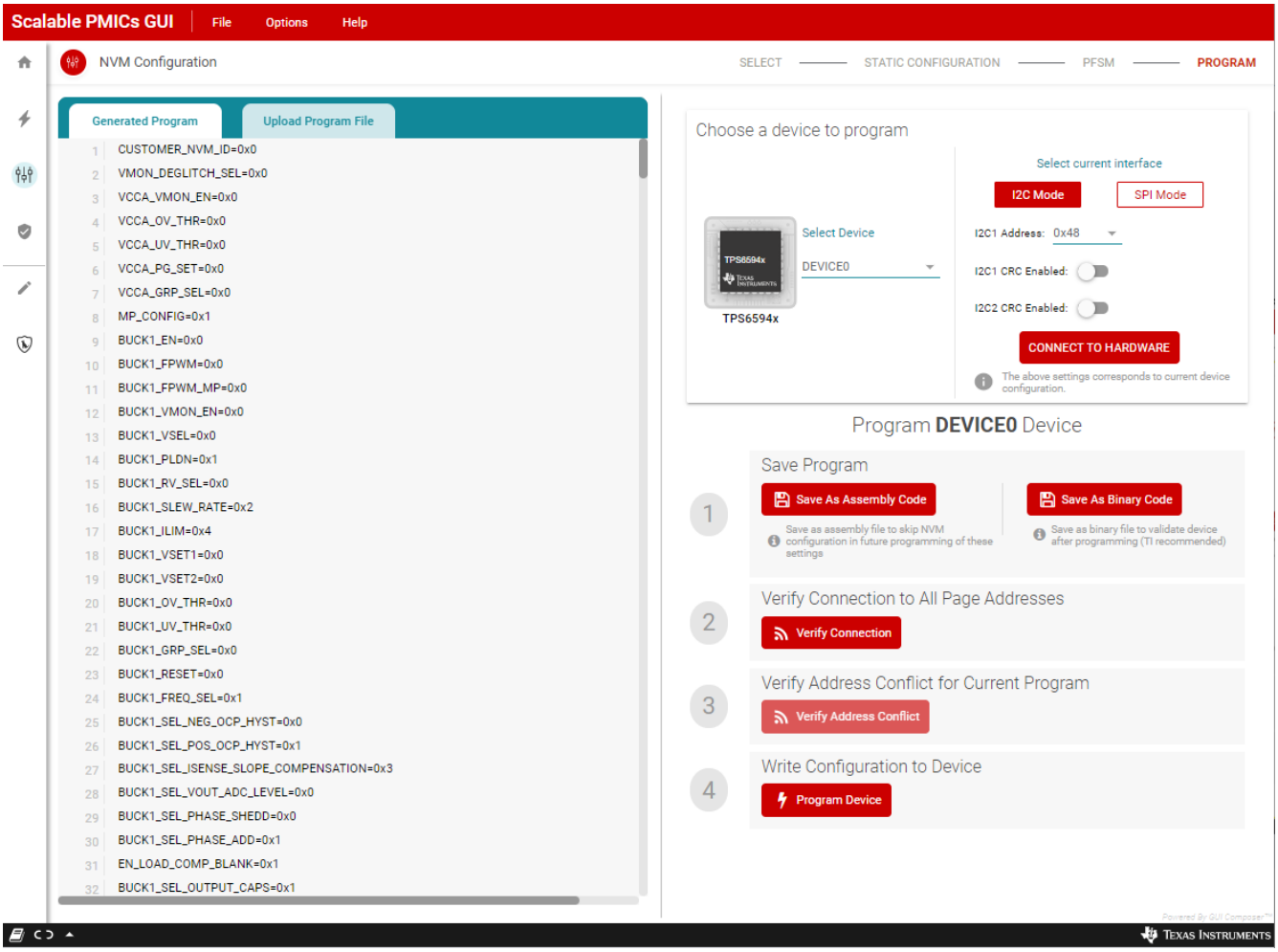

Figure 8-30. NVM Programming

The *Select Device* drop-down menu is provided when multiple PMICs have been configured. The associated program appears in the Generated Program tab when a device is selected. This will also determine which program will be saved and programmed.

The *Select current interface* will determine which physical address is used to verify the connection and program.

Two options are provided for saving the program, *Save as Assembly Code* and *Save as Binary Code*. The *Save as Assembly Code* is the same format as what is shown in the *Generate Program* tab. The *Save as Binary Code* format is of the register addresses and the hexadecimal values at those addresses. Both of these formats can be uploaded to the program page and programmed into the selected device without the use of the configuration steps, as discussed in Section 8.2.1. The *binary* format can also be used in the NVM Validation page to validate the NVM contents of a device. Once the physical connection is verified, then the device can be programmed.

Note

Texas Instruments recommends saving all three file types: Configuration, Assembly, and Binary.

WARNING

Texas Instruments will not support manually edited assembly or binary files.

8.2.1 Program an Existing NVM Configuration

The NVM Configuration page can also be used to program a device with an existing NVM Configuration in the assembly or binary format. The process is simply to start at the beginning of the page and choose which type of device to program by clicking the device icon, as shown in Figure 8-31. Once a device is chosen, click the *Skip to Programming* button to go directly to the programming page.

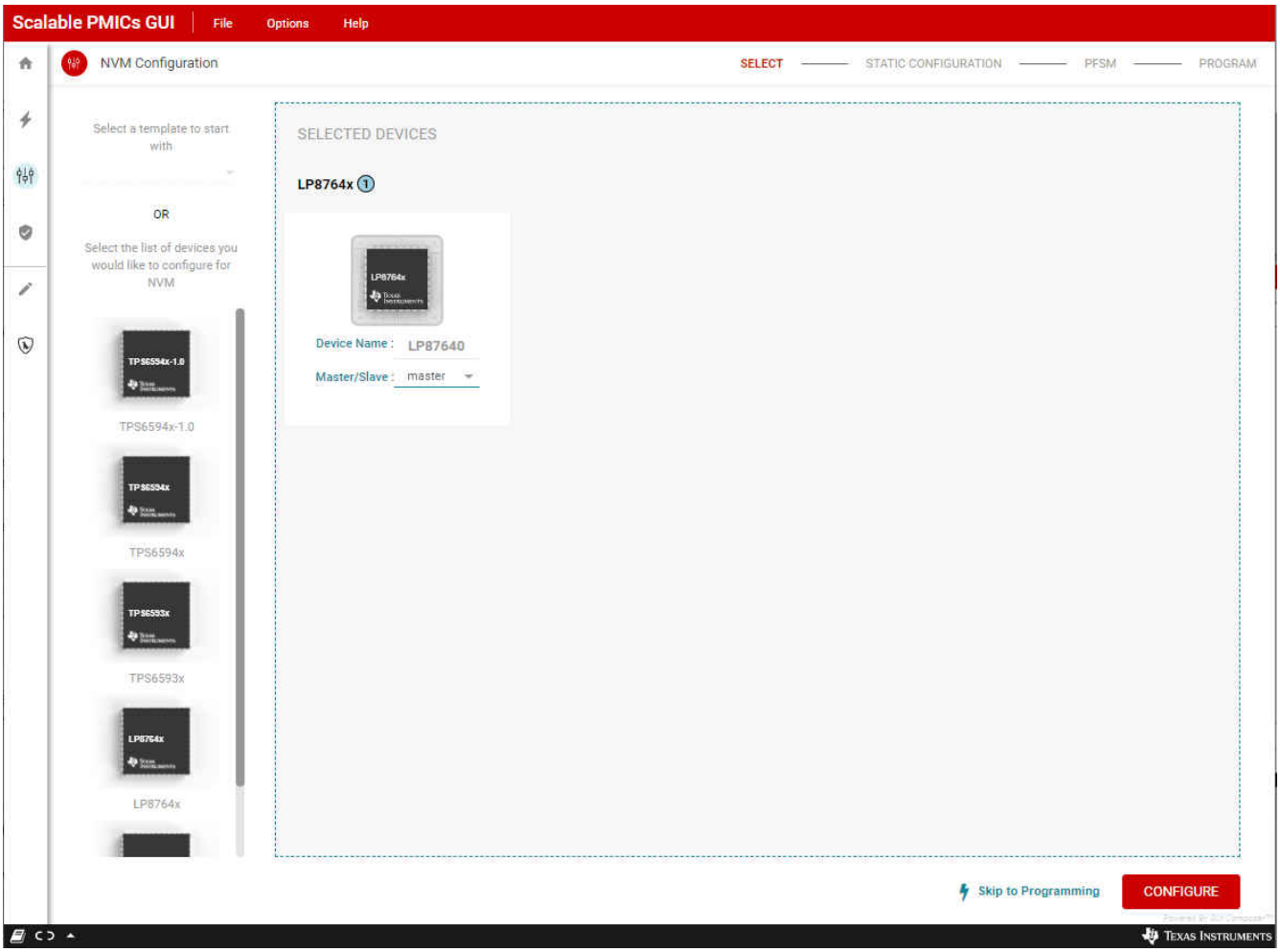

Figure 8-31. Skip to Programming with an existing NVM Configuration

The programming perspective has changed slightly from what was described in Section 8.2. As shown in Figure 8-32, only the *Uploaded Program File* tab is available, the save options are disabled, and the *Select Device* reflects the device selection at the beginning. The interface selection should match the device being programmed. If the interface is not setup correctly the connection indicator at the bottom of the screen will reflect that the *Hardware not Connected. Failed to connect* and any attempt to *Verify Connection* will fail. Once the correct[6] device is connected and the file to program selected, click *Verify Connection* and then *Program Device*.

[6] When multiple devices are connected to the AEVM, ensure that for I^2C that the address is that of the device to be programmed. In the case of SPI, make sure that the chip select is connected to the device to be programmed.

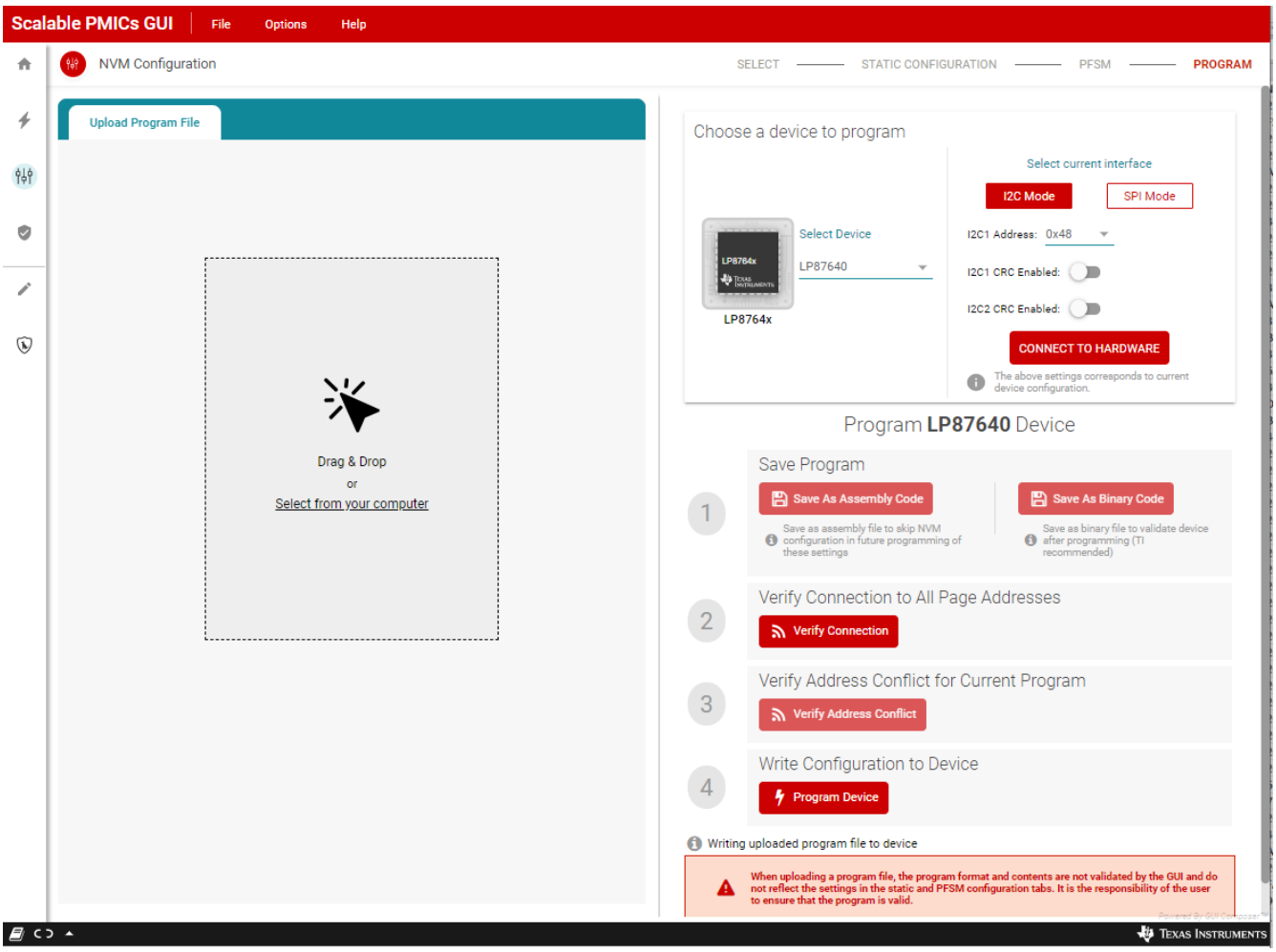

Figure 8-32. Program with an Existing Configuration

8.2.2 NVM Configuration Special Use Case: Changing the Communication Interface

The GUI tool writes the register settings using the identified interface and then uses the bulk programming method to transfer the register settings to the NVM. In the event that the NVM image uses a different interface, then it is important that the device being programmed supports both interfaces. When the GUI writes to the register settings to change the interface (SPI to I^2C or I^2C to SPI), the GUI will pause and display a prompt, see Figure 8-33, to change the hardware configuration to enable the new interface. This message specifically refers to the EVM, but can be generalized to any hardware configuration where changes are made to transition from I^2C to SPI or SPI to I^2C.

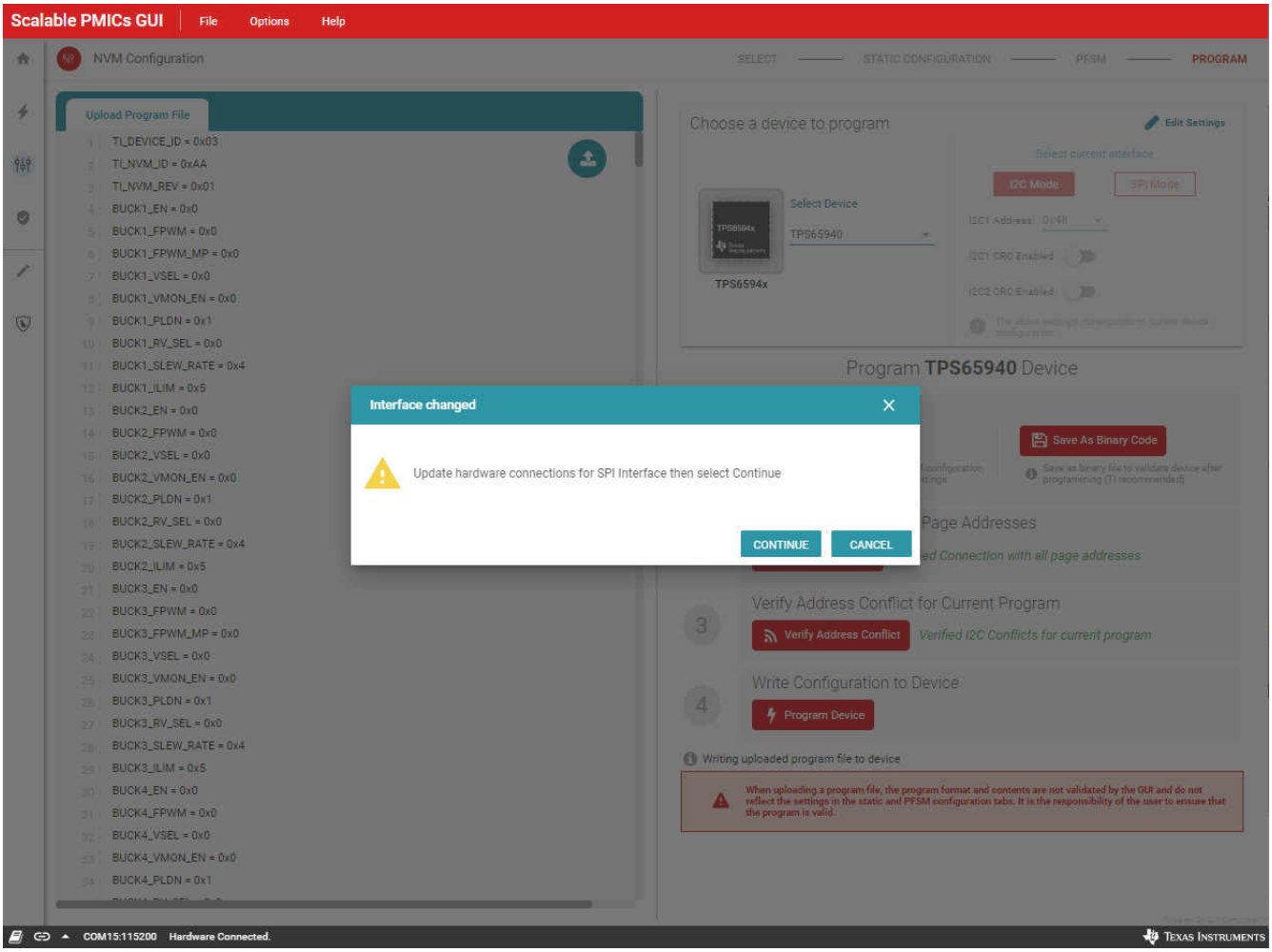

Figure 8-33. Interface Change during Programming

8.2.3 Lock Option During NVM Programming

The PMIC NVM can be permanently locked, preventing any changes to the NVM configuration. While this is not recommended during development, the feature is supported by the GUI. During the programming, a dialog window appears, see Figure 8-34, with the option to lock the NVM. Once this is done, then the PMIC NVM cannot be changed.

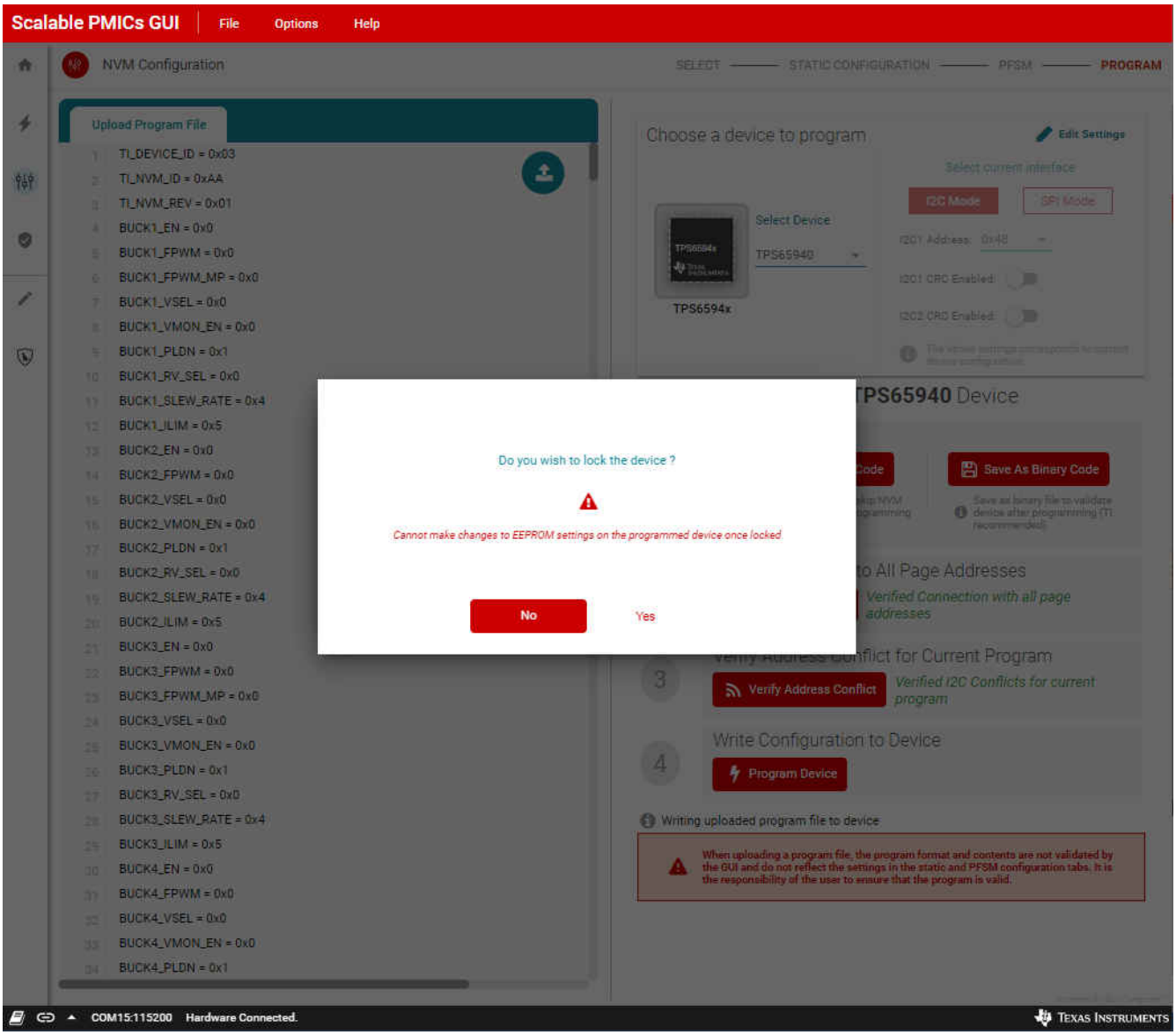

Figure 8-34. NVM Lock Option

CAUTION
Locking a device is permanent.

9 NVM Validation Page

Note

Texas Instruments recommends as best practice to always use the NVM validation to confirm the correct configuration of the device following programming from the NVM Configuration page.

The NVM Validation page is used to download the current register or NVM settings to the host PC or compare the settings to a file from the host PC. One of the important aspects of this page is that the NVM settings are accessed by overwriting the current register settings. If the *NVM Settings* button is selected, as shown in Figure 9-1, when the *DOWNLOAD CONFIGURATION* button is selected, then the GUI immediately issues a set of commands to the PMIC to overwrite all existing register settings with the contents from NVM. This will overwrite any settings or configuration to the device made through the Quick-start or Register Map pages. After the overwrite is complete, then the register contents are read out through the communication interface.

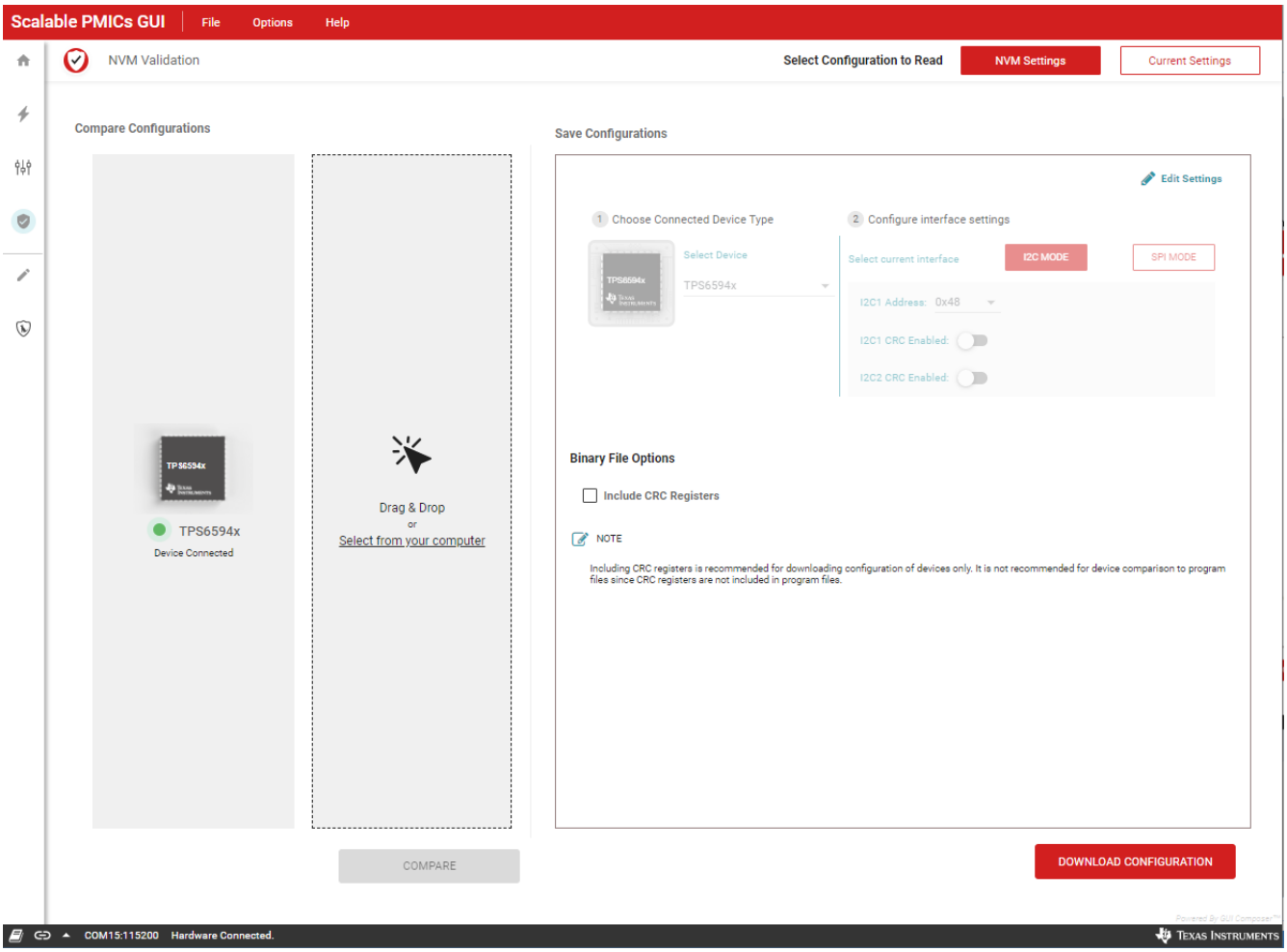

Figure 9-1. NVM Validation

The top right of the page indicates which of the *Current Settings* or the *NVM Settings*, are being read. On the left side of the page is an interface to select a known file to compare the read contents to. This provides a quick visual pass/fail response to evaluate the content read. The *DOWNLOAD CONFIGURATION* option is found on the right side of the page.

Compare the Current or NVM Settings with an existing binary file

1. Confirm that the correct device is connected.
   a. Select the correct Device Type.
   b. Select the correct Communication Interface.
2. Select the binary file from the host PC to compare the device NVM with.
   a. Use the Drag and Drop feature or the file navigator.
   b. When a valid file is selected, the *COMPARE* button will become active.
3. Press the *COMPARE* button.

Download the Current or NVM Settings to a binary file.

1. Confirm that the correct device is connected.
   a. Select the correct Device Type.
   b. Select the correct Communication Interface.
2. Select the *Current Settings* or *NVM Settings* button
3. Press the *DOWNLOAD CONFIGURATION* button.

10 Watchdog Page

The Watchdog page is an interactive evaluation tool for exercising the watchdog functionality in both TRIGGER and Q&A modes. This tool uses the MSP432E microcontroller on the AEVM to create the watchdog stimulus for both correct and incorrect use cases. Status monitoring is provided to show the PMIC response.

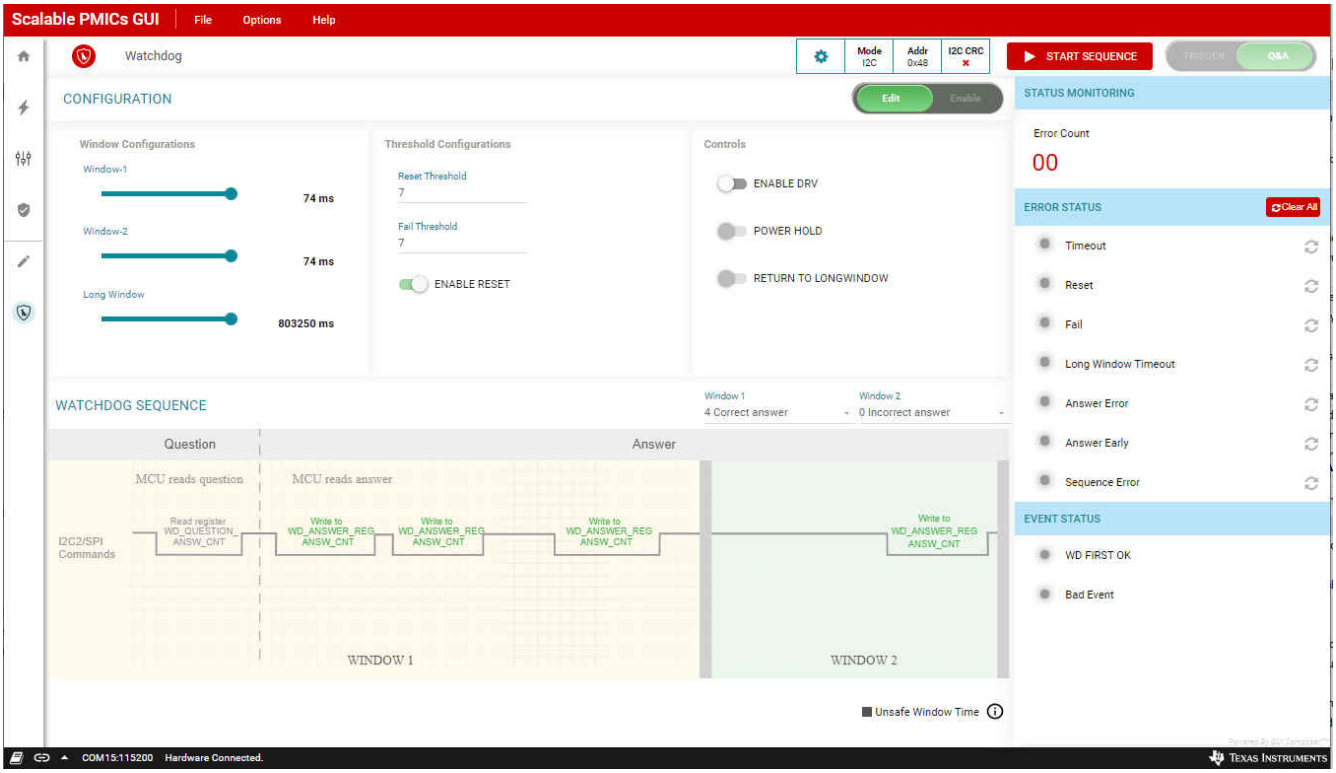

Figure 10-1. Watchdog Page

To exercise the watchdog module please make sure the following conditions are met in the PMIC:

1. The device supports the watchdog feature
2. The PMIC is in a mission state. A hardware state, like **SAFE RECOVERY**, will prevent the watchdog from operating.
3. NRSTOUT (found in the register MISC_CTRL) must be *1*
4. WD_EN (register WD_THR_CFG) must be *1*
5. If applicable to the device, any GPIO configured as DISABLE_WDOG should be logic *low*
6. WD_PWRHOLD must be *0*
7. For evaluation purposes it is recommended to set WD_RST_EN to *0* so that the device does not enter *WARM RESET* during the evaluation.

Trigger Mode

The watchdog page will reflect the settings in the controls section. To change from *QA* to *TRIGGER* mode, the *Edit/Enable* indicator, as highlighted in yellow in Figure 10-1, should be switched to *Enable*. Once in the trigger mode, switch back to *Edit*, and adjust the default *Window-1* and *Window-2* values to a time frame that does not exceed the limits of the AEVM.

With valid timing windows, the watchdog can be re-enabled and the watchdog sequence started.

Start Sequence:

1. Enable the watchdog, select *Enable*
2. Select the mode, *TRIGGER* or *Q&A*
3. Edit the AEVM trigger waveform, select *Edit*
4. Adjust *Window-1* and *Window-2*

5. Select the *Sequence Configuration*
6. Enable the watchdog, select *Enable*
7. Select *START SEQUENCE*

In the EVENT STATUS view, the WD FIRST OK is illuminated and green, see Figure 10-2, indicating that the first watchdog trigger sequence was received correctly. The PMIC is continuously polled during operation and the ERROR STATUS view is updated.

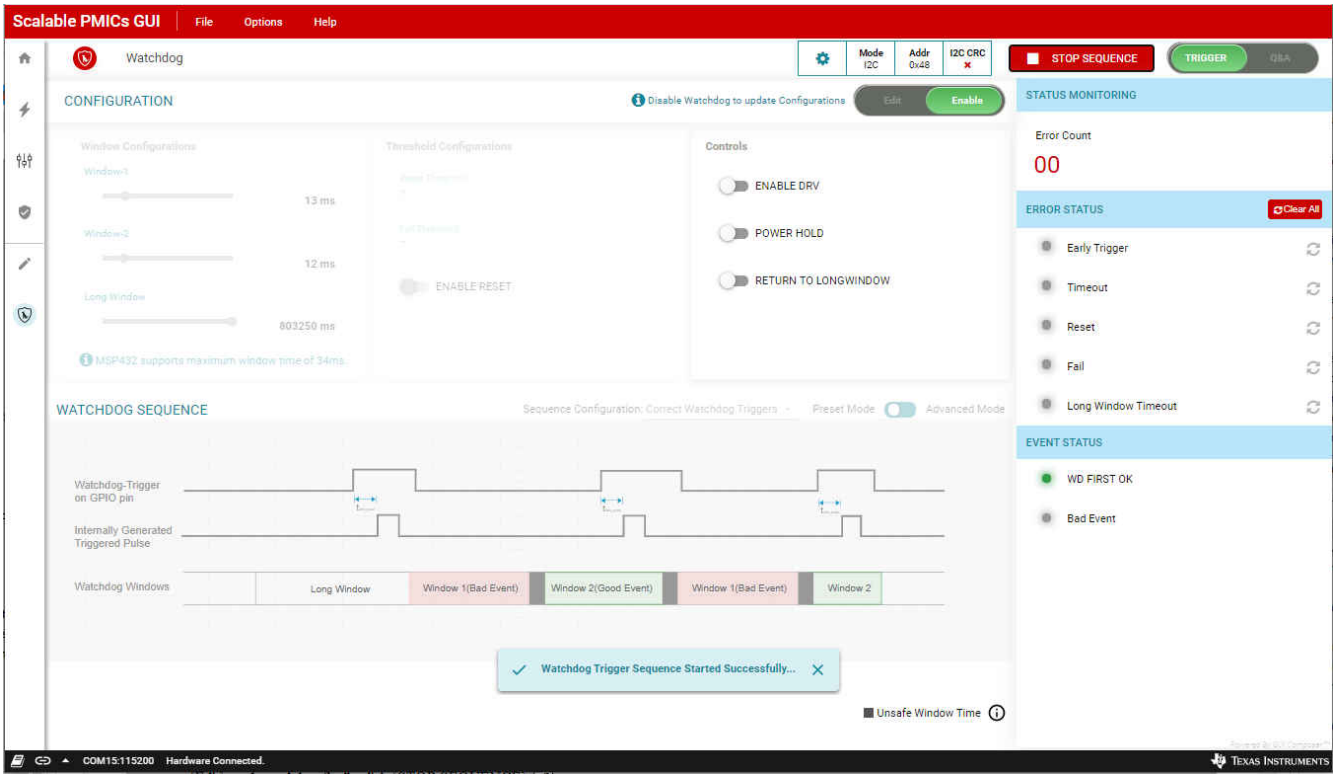

Figure 10-2. Watchdog Trigger Mode

To stop the evaluation, first use the control to set the RETURN TO LONGWINDOW, second switch from *Enable* to *Edit* mode, and then click *STOP SEQUENCE*.

Stop Sequence:

1. Select *RETURN TO LONGWINDOW*
2. Select *Edit*
3. Select *STOP SEQUENCE*

The sequence configuration can be modified to intentionally provide an incorrect sequence, using the drop-down menu *Sequence Configuration*. The incorrect timing and the consequent bad event is shown in Figure 10-3. To evaluate this configuration, first, switch from *Edit* to *Enable* mode, second, disable the *RETURN TO LONGWINDOW*, and then click *START SEQUENCE*.

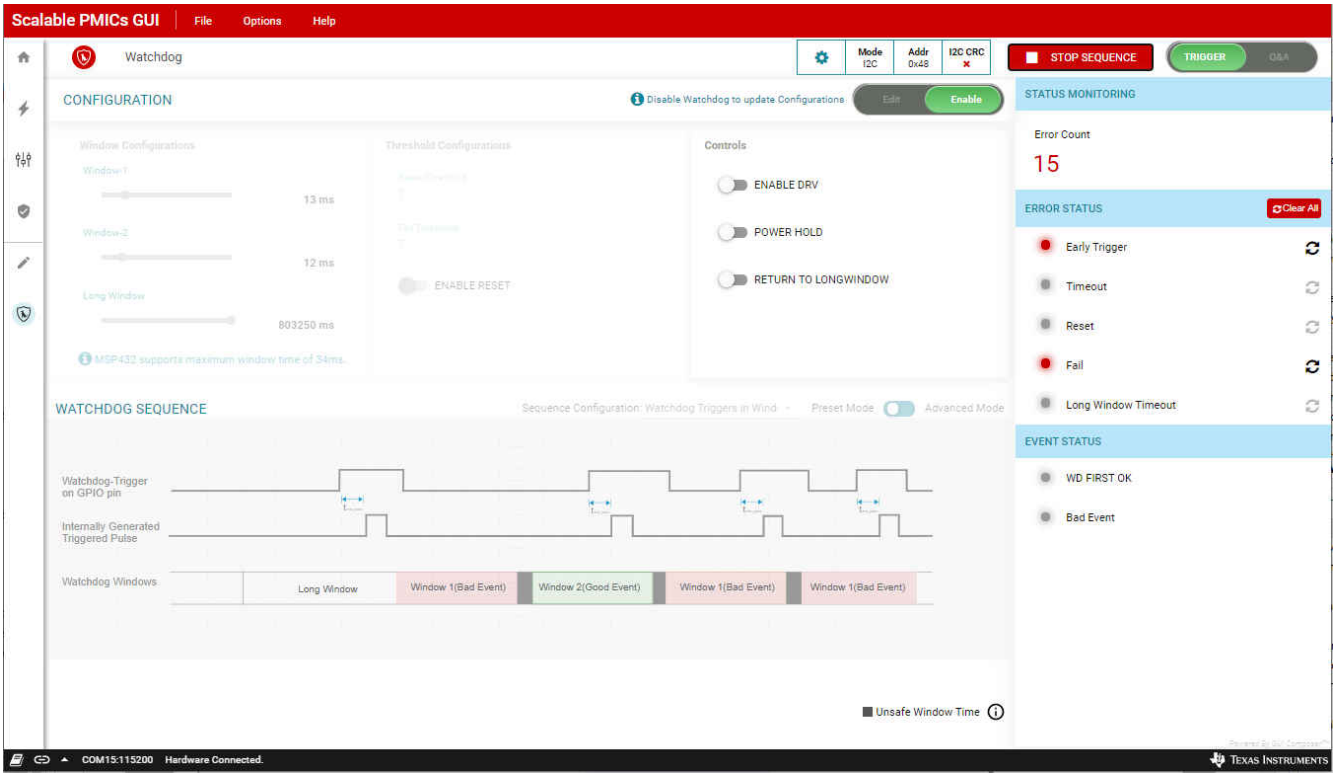

Figure 10-3. Invalid Trigger Watchdog Input

The errors can be cleared using the icons to the right of each error, or by clicking *Clear All*.

Q&A Mode

As previously stated, the Q&A mode can be selected when the *Edit/Enable* indicator is moved to *Enable*.

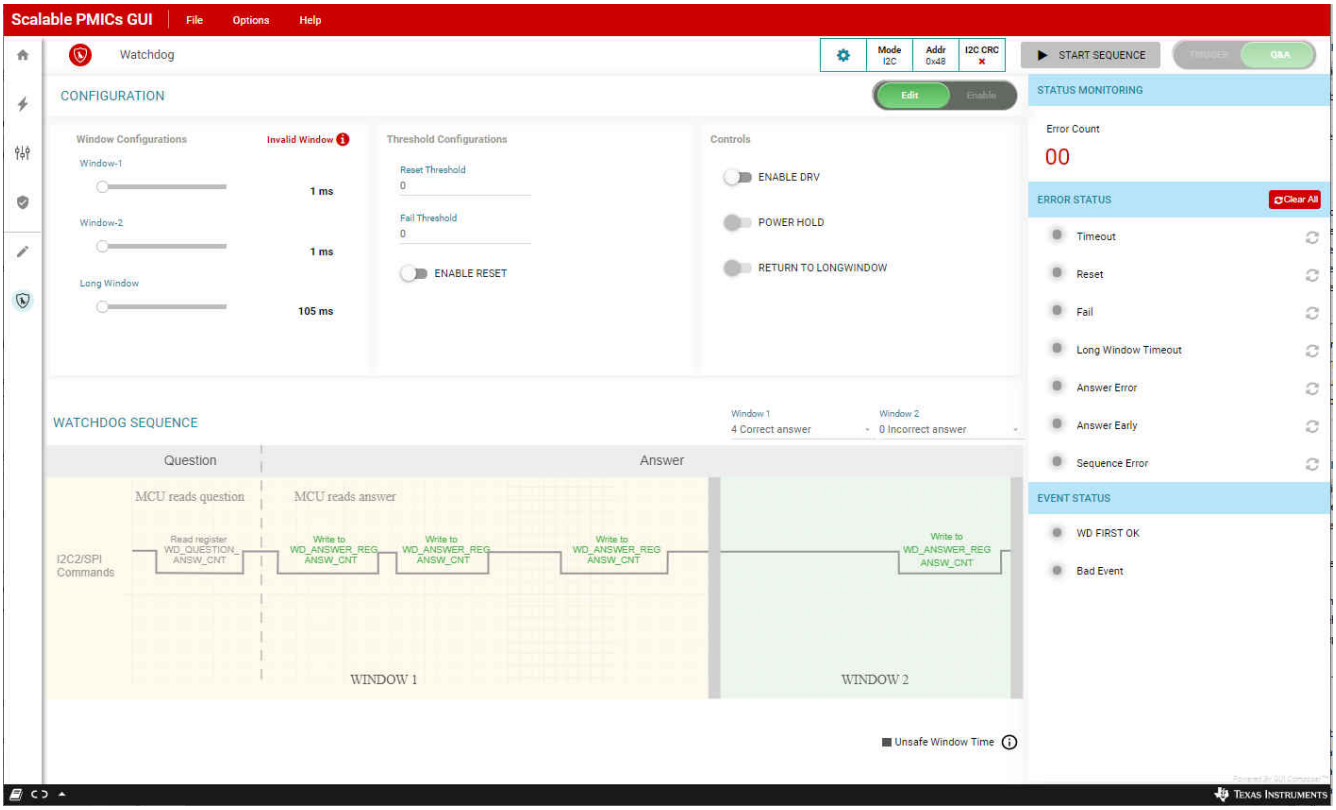

Figure 10-4. Watchdog Q&A

Once the Window configuration is updated for the Q&A a similar process can be employed to evaluate the Q&A mode. Figure 10-5 shows a correct sequence of answers to the questions while Figure 10-6 shows an incorrect answer.

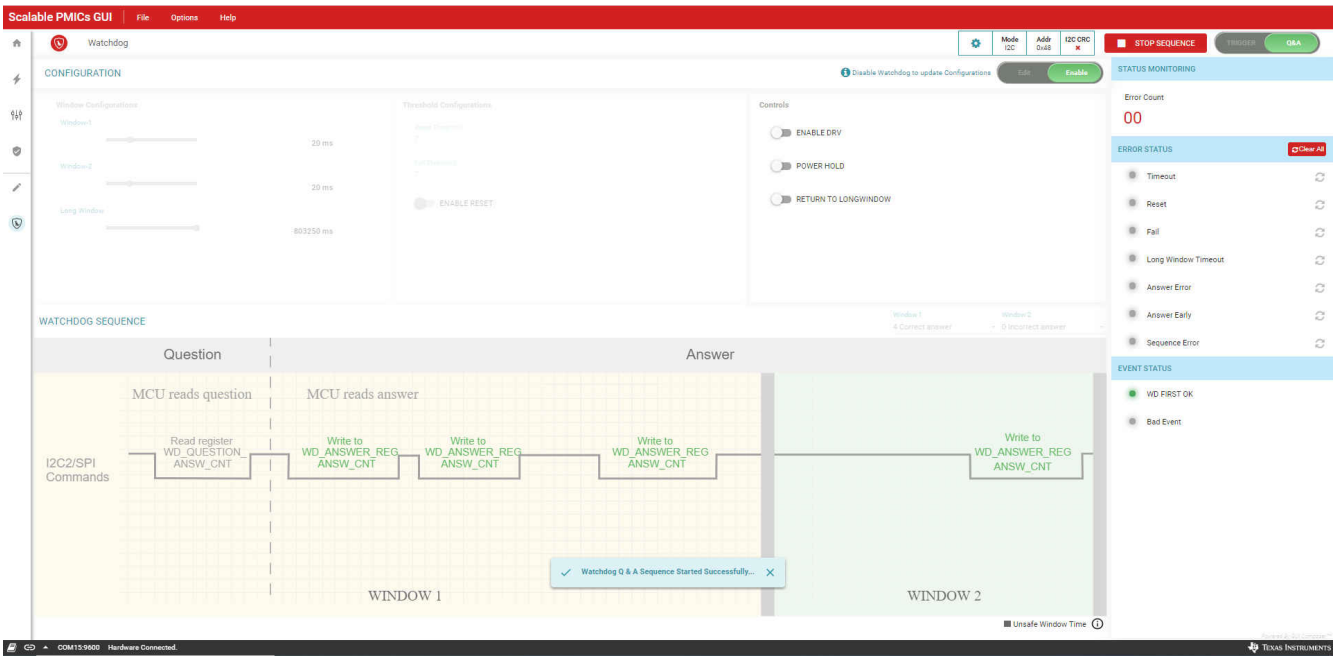

Figure 10-5. Watchdog Q&A Valid Response

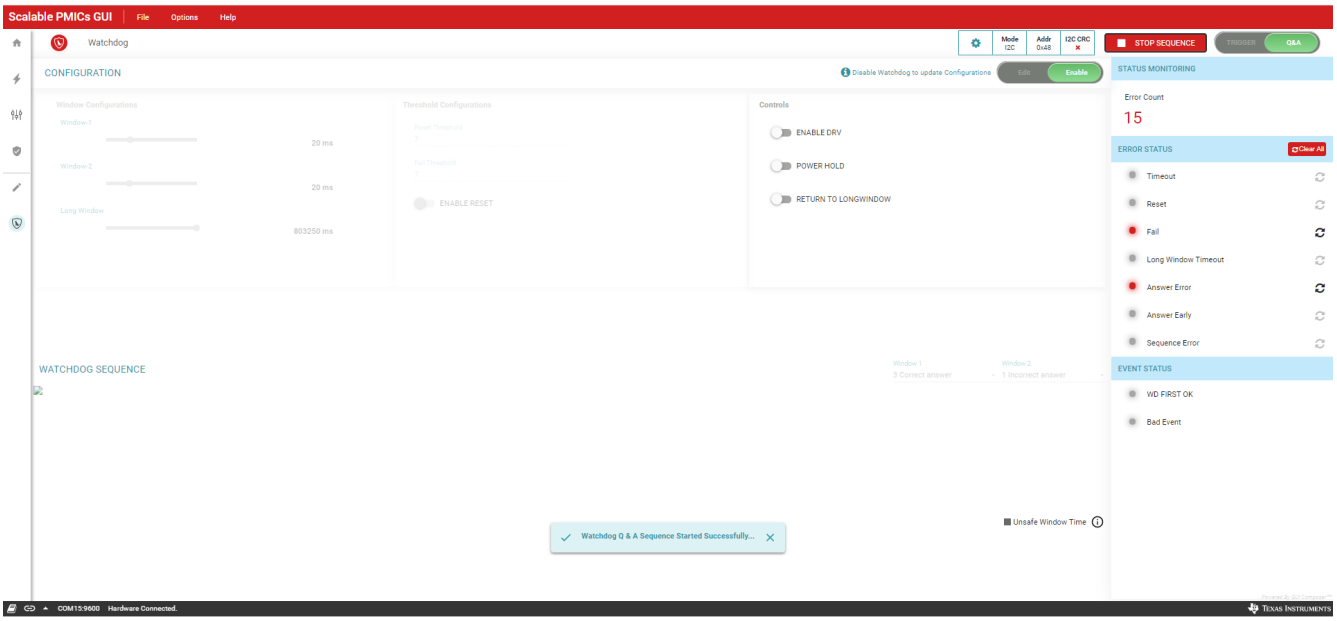

Figure 10-6. Watchdog Q&A Invalid Response

11 Additional Resources

1. SimpleLink™ Ethernet MSP432E401Y MCU Launchpad™ Development Kit
2. GUI Composer User's Guide.
3. E2E Support Forum.

12 Appendix A: Troubleshooting

12.1 Hardware Platform Not Recognized

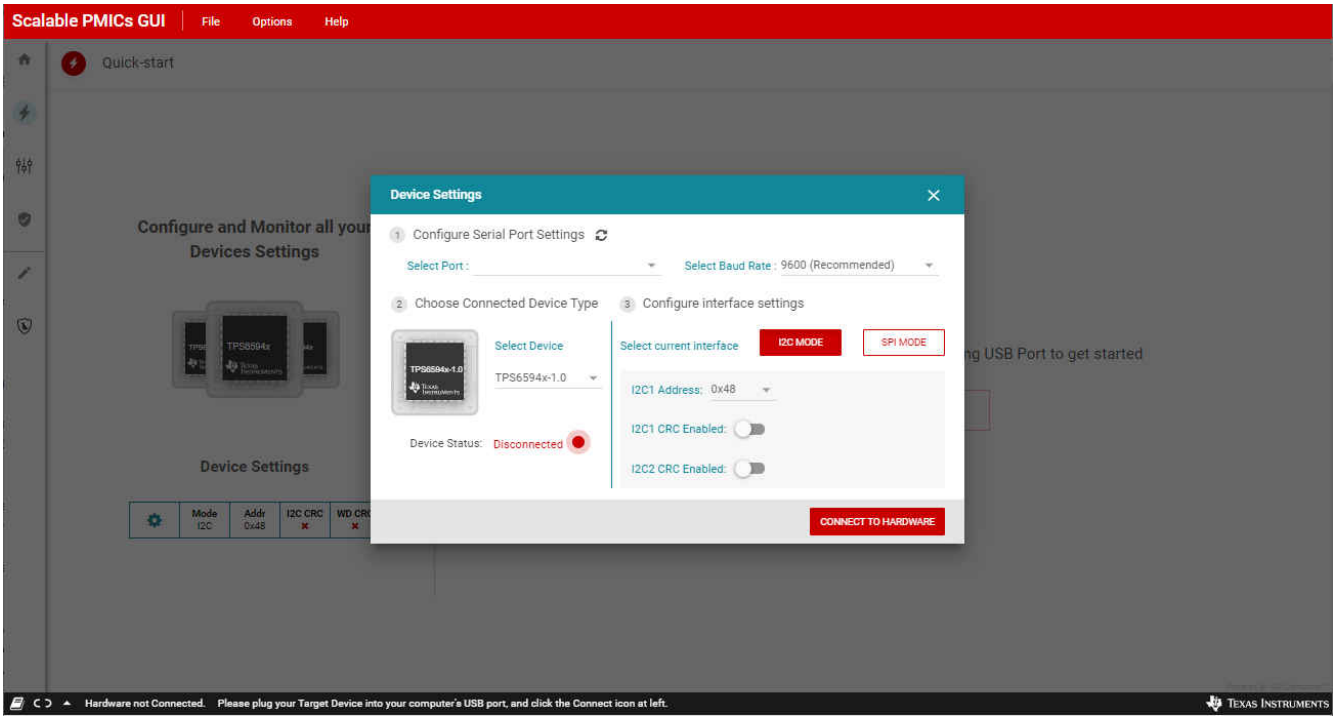

Figure 12-1. Hardware Platform Error

The GUI will automatically connect to the AEVM (micro controller with the analog evm controller firmware) and then to the PMIC. Typically, if the GUI cannot find the hardware platform this is due to either a faulty USB connection or the GUI is attempting to connect to the wrong communication (COM) port. The hardware platform will enumerate as three devices; two CDC classes and one DFU class. From another program, like the device manager in windows, the user can verify which COM port is the ACCtrl and which is the ACCtrl Console. From the GUI option, the user should select the COM port number of the ACCtrl and ***not*** the ACCtrl Console. The AEVM supports a baud rate of 115200.

Other devices connected to the PC, may also enumerate at CDC class devices and the GUI may attempt to communicate with these devices. If no response is made from the device, then it is possible that the GUI will not attempt to connect to other devices until the current device responds.

Only one AEVM should be connected to a host (MAC of PC) at one time. The GUI does not support the ability to handle and respond to multiple AEVM devices connected to the host.

12.2 PMIC Device Not Found

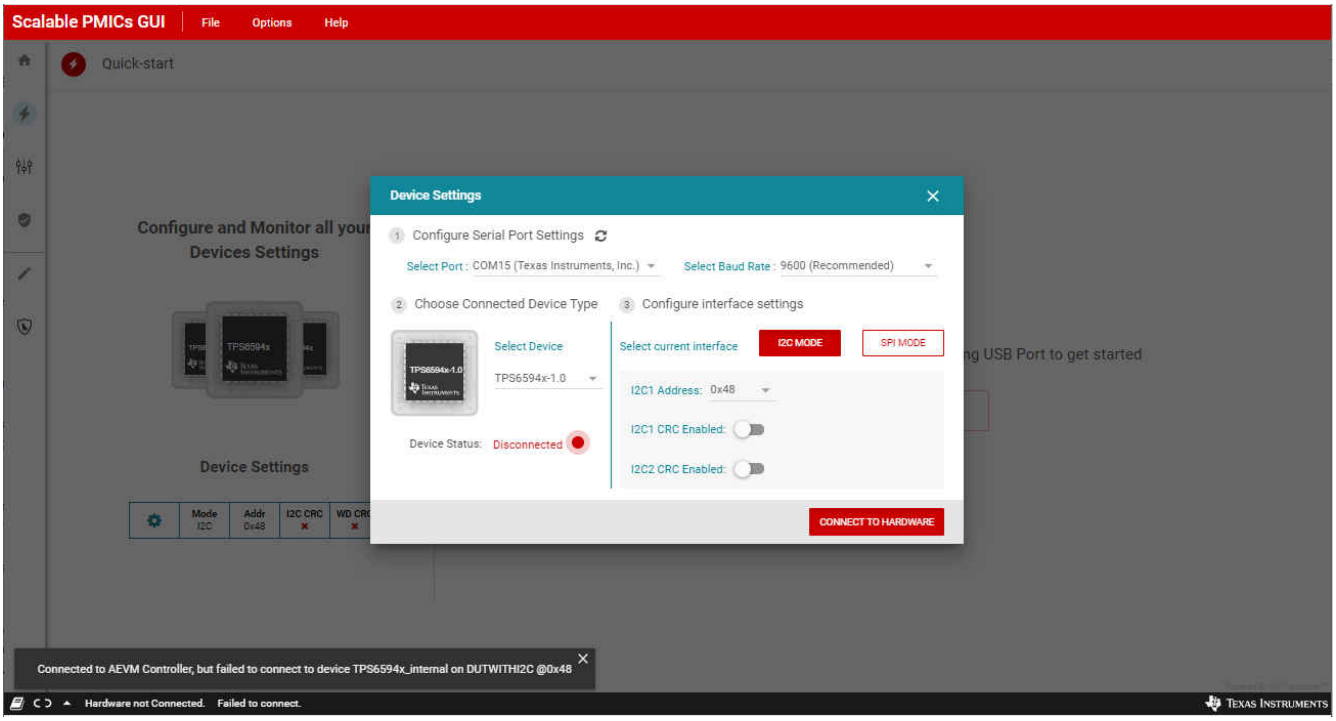

Figure 12-2. PMIC Communication Error

If the hardware platform is connected but the PMIC device is not found, then the GUI will report an error regarding the address: *Connected to AEVM Controller, but failed to connect to device TPS6594x_external on I2C @xx*. In the case of I^2C this means that the address was not acknowledged (NACKed). Perform appropriate I^2C bus checks: appropriate pull-up resistors, verify that no device is holding the clock low (clock-stretching), and so forth. Verify that the correct I^2C address is being sent. The default address which the GUI uses may not be correct and it is necessary to update the address using the *Device Settings* below the *Options* tab.

Similarly with SPI, ensure that the hardware connection is correct and that the chip select is connected to the appropriate PMIC when multiple PMICs are in use.

12.3 I2C2 is configured but not connected

The PMIC supports a dedicated I^2C interface, I2C2, for access to the page 4 register space. This interface is only needed for watchdog configuration and only enabled with the GPIO is configured appropriately. The GUI will automatically interrogate the PMIC and use the I2C2 through the AEVM. If the I2C2 is not physically connected but the GPIO is configured for I2C2 operation the GUI will report an error when attempting to access the page 4 register space. In GUI version 2.0.0 this error is seen when the user attempts to read registers from page 4; *READ REGISTER* button. In GUI version 3.0.0 this error is seen immediately when the PMIC updates all registers upon connection and when using the *READ ALL REGISTERS* button. In both the GUI version 2.0.0 and 3.0.0 the error is reported as *I2C read error. Status = 4294966526*

This error can be ignored if the page 4 information is not needed. Also, establishing the I2C2 connection will remove this error.

13 Appendix B: Advanced Topics

13.1 Scripting Window

Scripting is a convenient way to send a sequence of commands (reads or writes) to the PMIC device registers as opposed to the individual commands associated with an update to a parameter in either the quick-start or Register page views. *Scripting* is found below the Options tab located at the top of the GUI. Opening the scripting window will open a new window while the GUI window will still be active as shown in the following paragraph.

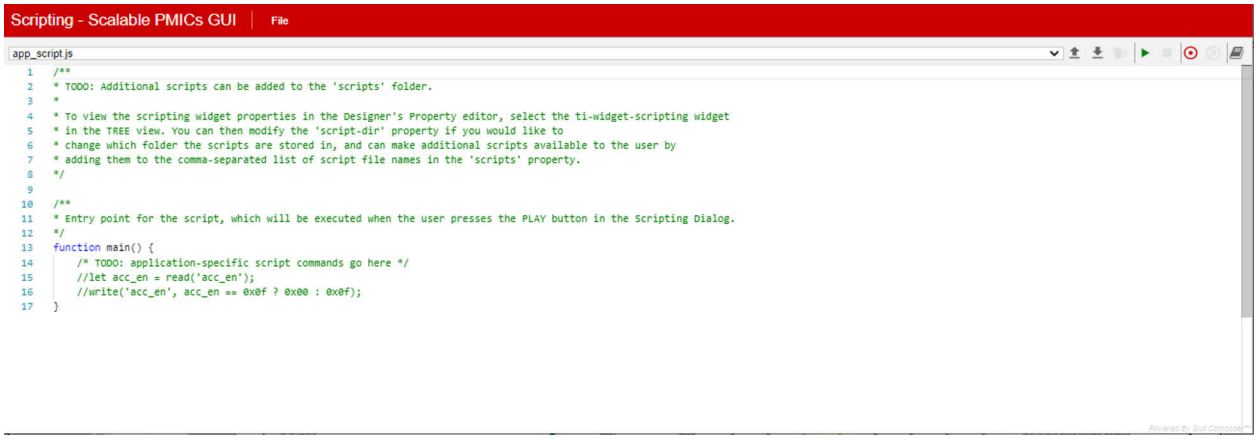

Figure 13-1. Default Scripting Window

Figure 13-1 shows the initial scripting window and the default text provided. This file can be edited to provide a desired sequence of commands. One quick way to start using the scripting window is to use the record feature to capture a sequence of commands. In the upper right hand corner is the record icon. Hovering the cursor over the icon will reveal a *Start Recording* help box. In the example shown in Figure 13-2 and Figure 13-3, the recording is started and then when returning to the GUI window the Register Page is used to read DEV_REV and NVM_CODE_2 and then write to the BUCK1_CTRL register. Once these sequence of actions are completed, then returning to the scripting window will reveal the recorded commands. At this point, the recording can be stopped and these commands can be expanded and repeated for the various registers. Once the sequence is complete, then click the *Run* icon to execute the sequence.

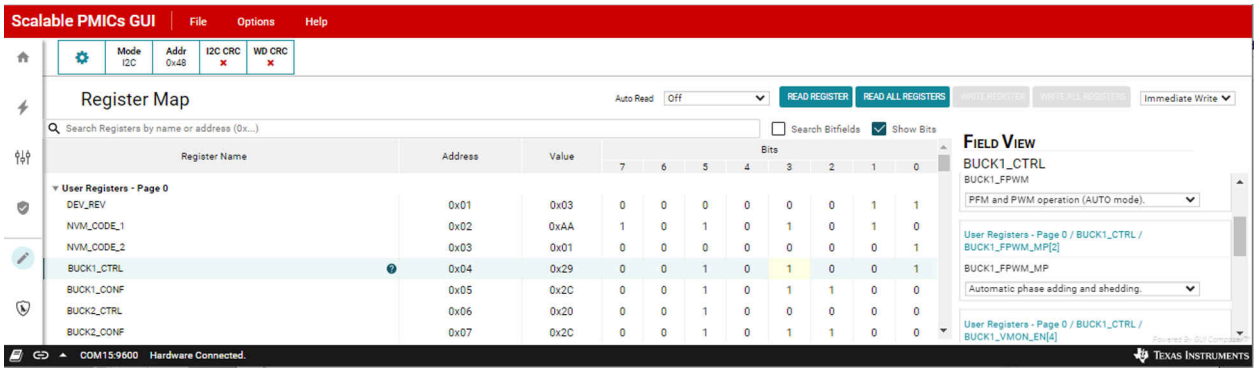

Figure 13-2. Scripting, Recording Register Read and Writes

Figure 13-3. Scripting, Running a recorded sequence

14 Appendix C: Known Limitations

This section contains known limitations of the GUI and which versions the limitations apply to. Please use the support forums to report any issues or limitations found that are not on the following list.

Table 14-1. GUI Limitation and Relevant Version

Number	1.0.0	2.0.0	3.0.0
1	X	X	X
2	X	X	X
3	X	X	X
4	X	X	
5	X	X	X
6	X	X	X
7	X	X	
8	X	X	X
9	X	X	X
10			X
11			X
12		X	X

Table 14-2. GUI Limitations Description

Number	Description	Workaround
1	NVM validation when including the register CRC can report differences associated with addresses 0x0F4 (CRC_5), 0x0F5 (CRC_6), 0x0FE(CRC_15), and 0x0FF (CRC_16).	None. The GUI will report that the NVM files do not match. Inspect the registers and disregard differences associated with these registers. Each piece of silicon can have unique values in CRC related to production information.
2	Modifying Device Type and its configuration (for example, Primary/Secondary, Phase Configuration.. etc) is not recommended after creating PFSM as it may not produce expected results	In the case of modifying the phase configuration, remove all references in the PFSM before attempting to change the phase configuration. Changing the device type will require starting from a new or blank template.
3	SPI Hardware connection status displayed in the status bar can be detected incorrectly.	Use the register map page to confirm that values other than 0x00 and 0xFF can be read from the device. A dialog message appears to address the same.
4	The GUI does not provide control for multiple SPI Chip Select outputs.	None. In multi-device SPI configurations the Chip Select signal from the AEVM needs to be manually moved to each PMIC individually.
5	USB Connections issues with COM ports associated with Bluetooth devices.	Disable or remove devices which enumerate as COM ports on the host pc.
6	Enumeration (connecting the USB cable to the AEVM) will attempt to connect to the device with the previous or default settings and may fail to connect.	Use the Device Settings and click the *Connect to Hardware* to update the GUI to the correct connection settings.
7	Programming with register CRC enabled can result in a wrong CRC value. Some user registers are not backed by NVM but are part of the CRC calculation.	Program the NVM twice.
8	Targets with SPI interfaces should be connected to the PC after the GUI has been configured to SPI mode.	Recycle power to the AEVM after selecting SPI in the GUI.
9	Programming an LP876x-Q1 family device and changing GPIO functionality between nRSTOUT and some other function will result in a temporary loss of the serial interface. Typically, this results in a NACK in I2C and a frame error in SPI. The NACK will be seen by the GUI as an error and the programming sequence will stop.	Use the register map page to change the GPIO setting to match the setting in the NVM to be programmed. Reprogramming after the first attempt fails (without power cycling the PMIC) will have the same affect, since the user registers for the GPIO settings were updated by the initial attempt to program.
10	PFSM validation will report a warning in the 'Delay Timings Validation' if the PFSM delay step is not restored at the end of a sequence even if the PMIC will transition to the SAFE_RECOVERY state after the sequence is executed.	None. When the PMIC transitions to the SAFE_RECOVERY state the PFSM delay step is restored automatically to the default value. The user can disregard this warning, provided the PMIC transitions to SAFE_RECOVERY. This can be seen in the J721E_TPS6594 template.

Table 14-2. GUI Limitations Description (continued)

Number	Description	Workaround
11	PFSM validation will report an error in the 'Sequence Length Validation' if the SREG_DELAY instruction is used to create the required timing relationship between the primary and secondary PMICs.	None. The SREG_DELAY instruction is a unique delay instruction which can change in value during runtime, depending upon what value is loaded into the SREG. The user can disregard this error provided that the usage of the delay is understood and the timing relationship preserved. This only applies to multi-pmic applications. This can be seen in the J721E_TPS6594 template.
12	BUCK Voltage selection, BUCKx_VOUT1 and BUCKx_VOUT2, is not limited by the BUCK use case.	Users must ensure that output voltages selected are within the ranges specified in the datasheet for the given configuration use case.

15 Appendix D: Migration Topics

The changes associated with version 3.0.0 will make the json files created with version 2.0.0 incompatible, requiring the following updates.

1. Migrating from LP8764-Q1 silicon version PG1.0 to silicon version PG2.0.
2. Update the PFSM to include pfsm_start.
3. Update all timing delays for manual implementation.
4. Update trigger priority and settings.

15.1 Migrating from LP8764-Q1 PG1.0 to PG2.0

The differences between the LP8764-Q1 PG1.0 and LP8764-Q1 PG2.0 in the GUI are strictly related to the format and will result in the error shown in Figure 15-1. Requests can be made through the e2e forum, 3, to update the format. Alternatively, users can elect to modify the json file. The modifications are minor and only relate to the device names found in the json file. This example uses the *Generic_LP8764* template as an example for migrating from PG1.0 to PG2.0.

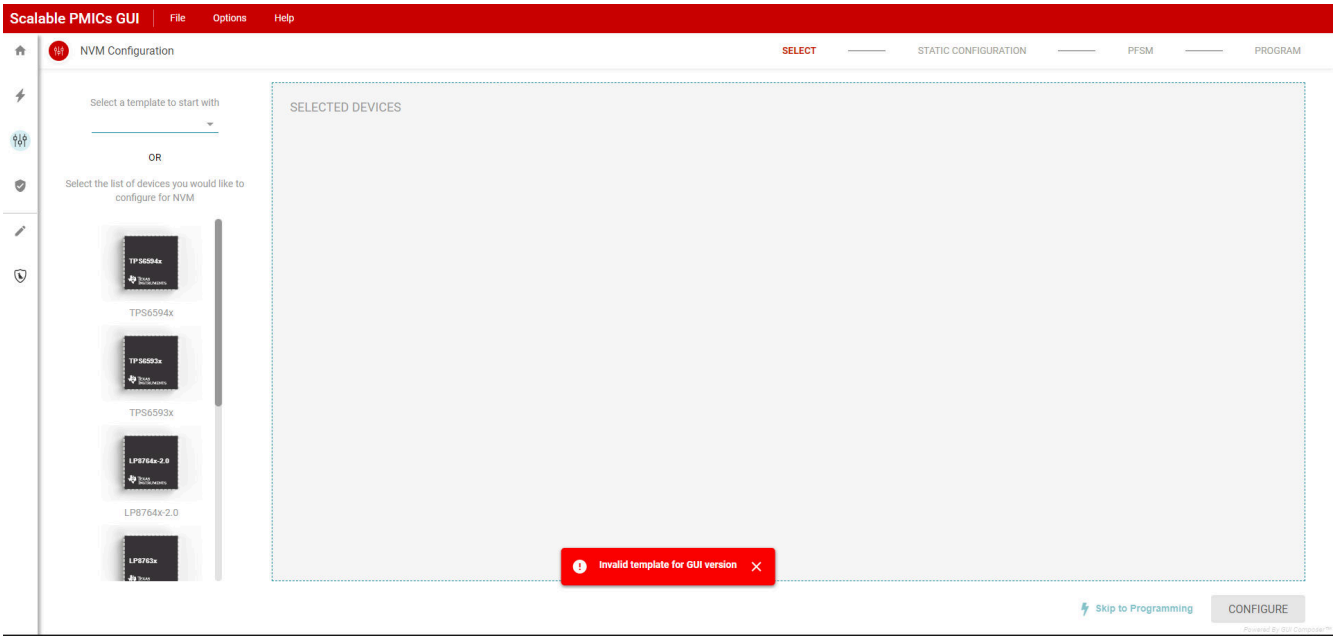

Figure 15-1. Error Message when Attempting to Import LP8764-Q1 PG1.0 into GUI Version 3.0.0

As shown in Figure 15-2, the TPS6594-Q1 PG1.0 must be removed and the LP8764-Q1 must be updated to PG2.0. On the left side of the original json file and on the right-side are the required changes to make the json file compatible with the GUI version 3.0.0.

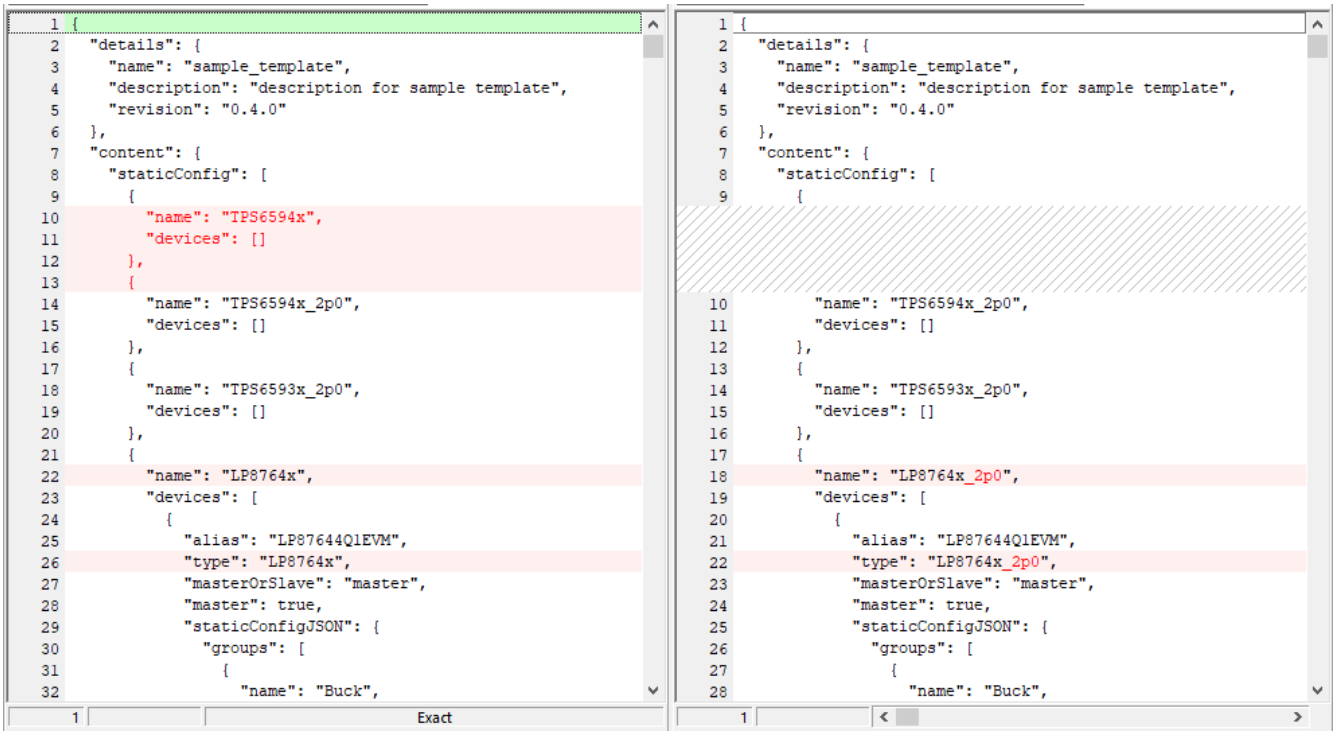

Figure 15-2. NEED TITLE *

Note

Migration from pre-release silicon TPS6594-Q1 PG1.0 to post release silicon TPS6594-Q1 PG2.0 is not supported. GUI versions 2.0.0 and 3.0.0 both support TPS6594-Q1 PG2.0.

15.2 Update the PFSM to Include the PFSM_START State

This migration example continues to use the *Generic_LP8764* with the required changes to the json file as described in Section 15.1.

As described in section Section 8.1.2.1, the PFSM_START state is required and must be added. In the *Generic_LP8764* example, the STANDBY state can be entered from either the ACTIVE or MCU_ONLY states and therefore a separate state for PFSM_START must be added as shown in Figure 15-3. If the STANBY state has no inputs from other states and is designated as the start state, then it is possible to simply rename the STANDBY state to PFSM_START.

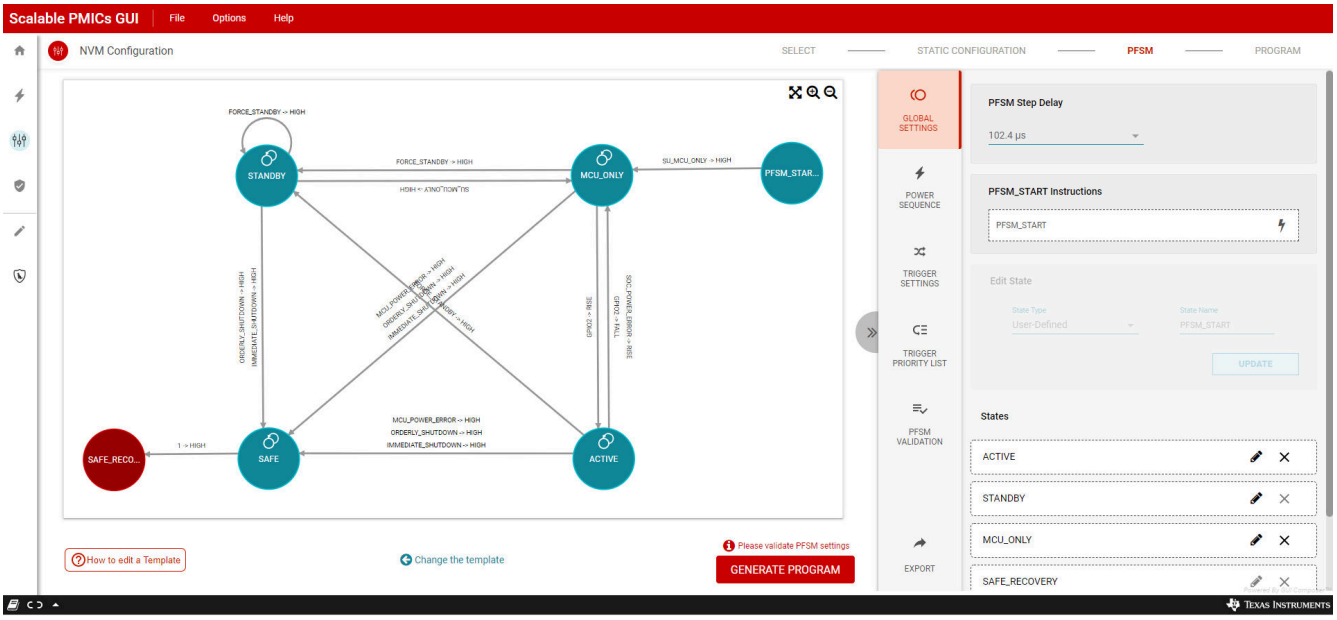

Figure 15-3. Adding PFSM_START State

15.3 Update Timing Delays

The removal of the automated time management in GUI version 3.0.0 requires the timing to be updated manually. The places which require updates are shown in the PFSM Validation. As shown in Figure 15-4, the DELAY_IMM 500 ms command, requires that the PFSM_STEP_DELAY be updated before that command is performed. The DELAY_IMM 500 ms and the associated fix are in the *immediate_pd_shutdown* power sequence, as shown in Figure 15-5.

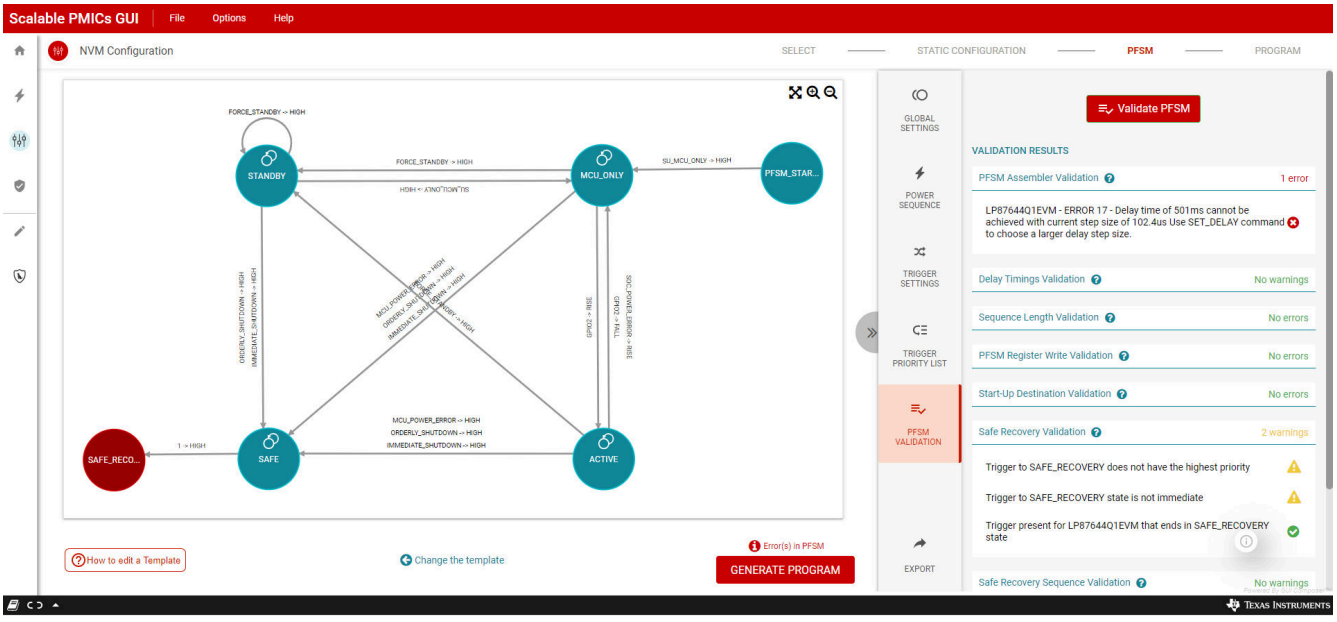

Figure 15-4. PFSM Assembler Validation Error with Delay Timing

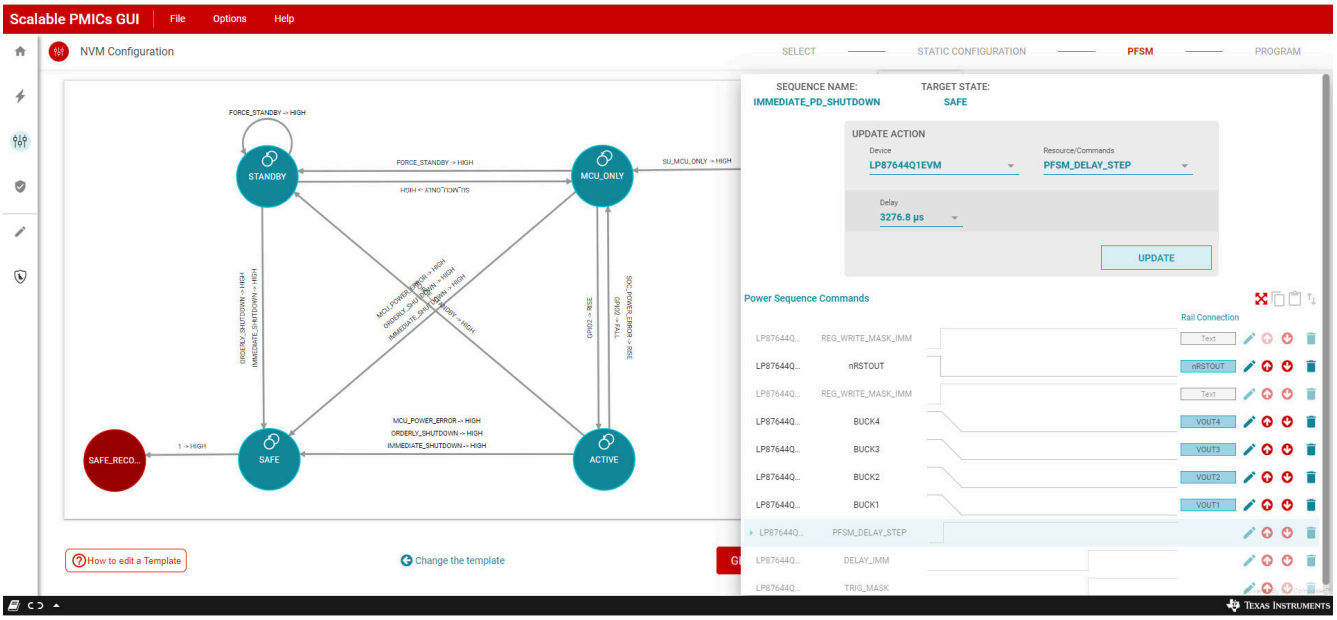

Figure 15-5. Delay Time Fix for Migration

Once the sequence has been updated the error is resolved as shown in Figure 15-6. The warning is an intentional result of this example. In most cases, the PFSM_DELAY_STEP must be restored to the global setting. In this special case, the sequence is executed before transitioning to the *SAFE* state and then automatically to the hardware SAFE_RECOVERY state. In this case, the PFSM_DELAY_STEP is automatically restored to the global setting when the PMIC returns to the mission states from the SAFE_RECOVERY state and the warning message can be ignored.

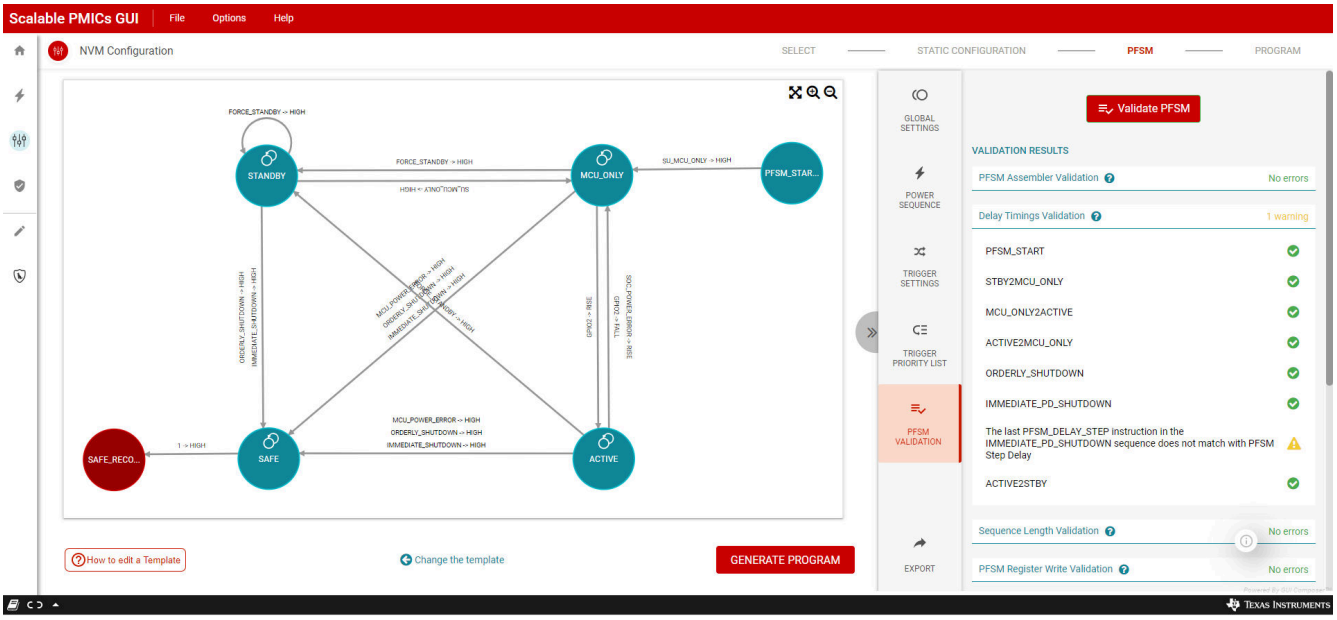

Figure 15-6. PFSM Validation Results with Updated Timing Delay

15.4 Update Trigger Priority and Settings

Continuing to use the example from Section 15.3, there are additional warnings associated with the triggers. Figure 15-7shows the warnings relative to the trigger to SAFE_RECOVERY.

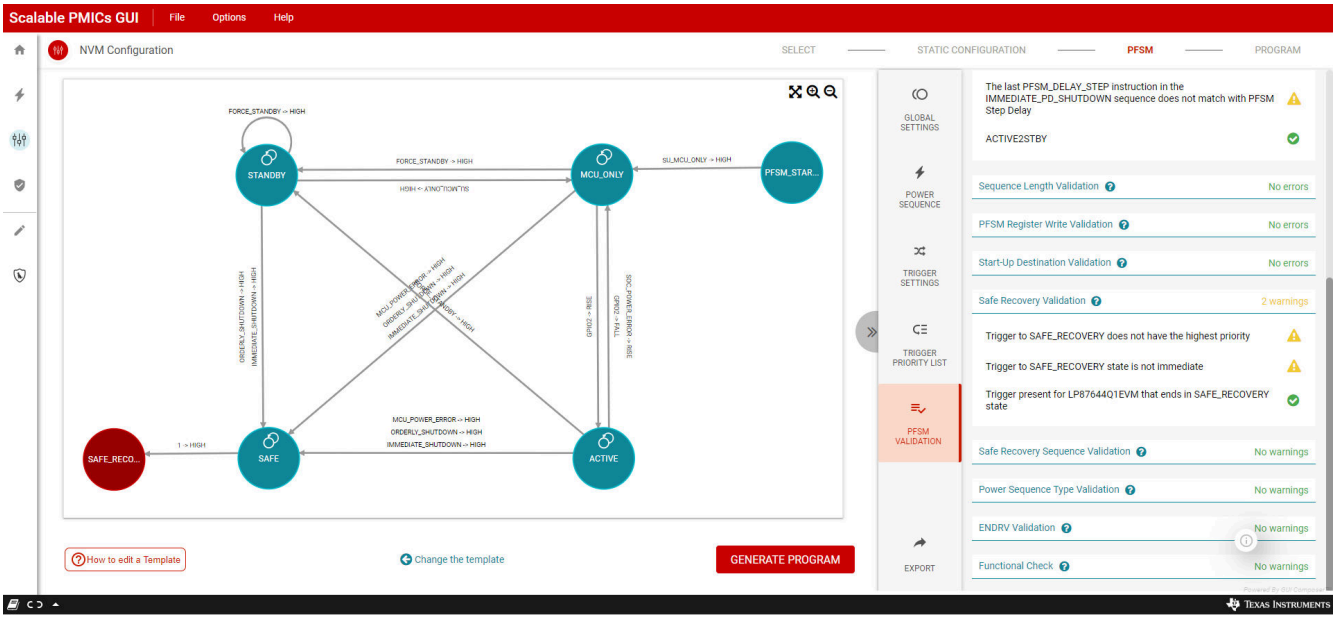

Figure 15-7. Warnings for SAFE_RECOVERY Trigger

The trigger priority can be updated in the *Trigger Priority List* as shown in Figure 15-8. Figure 15-9 shows the immediate trigger feature indicated in the *Update Trigger* found in the *TRIGGER SETTINGS*.

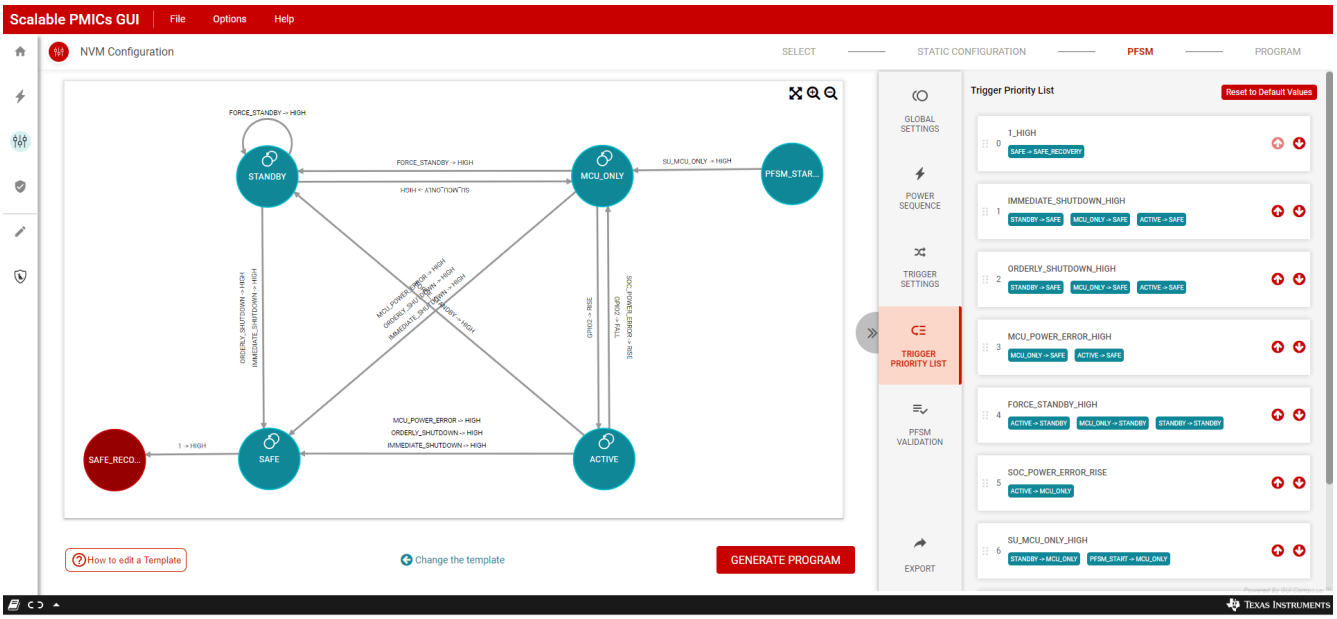

Figure 15-8. Priority List Update

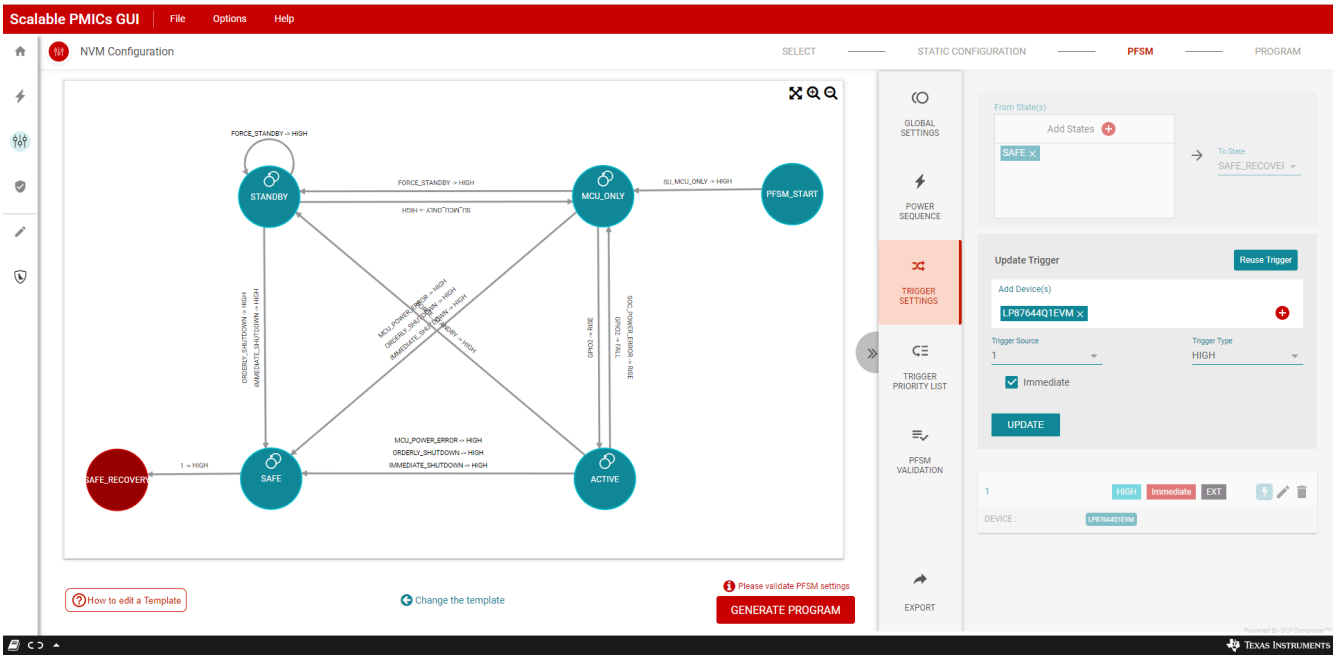

Figure 15-9. Immediate Trigger Feature

Figure 15-10 shows that with these fixes, the PFSM Validation has only one warning, that was discussed in Section 15.3.

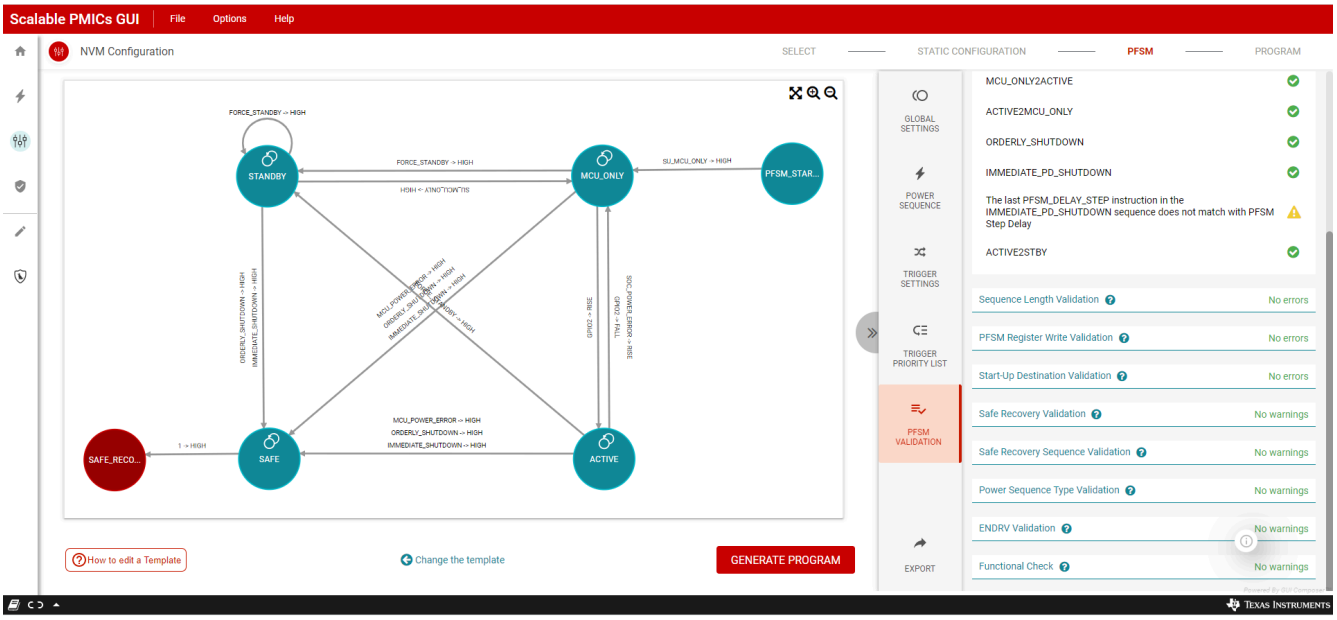

Figure 15-10. PFSM Validation Results after Addressing Trigger Warnings

16 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision A (November 2020) to Revision B (June 2022)	Page
• Added addition families of devices	1
• Updated with inclusive language	3
• Added summary for version 3.0.0	4
• Updated image for version 3.0.0	5
• Updated version to 3.0.0	6
• Added note about initially connecting to a PMIC configured with a SPI serial interface	8
• Added information for multi-PMIC SPI option	9
• Updated SPI information	13
• Deleted note	23
• Update images for version 3.0.0	28
• Added PFSM_START state	30
• Updated Global settings for version 3.0.0	31
• Updated Power sequence window for version 3.0.0	32
• Added instructions: nRSTOUT_SOC, VMONx, PFSM_DELAY_STEP, and END with descriptions. Updated DEACTIVATE description: not recommended for multi-pmic solutions	36
• Updated images for version 3.0.0	42
• Added table to cross issues with version, and add new limitations	66
• Added Appendix D	68

Changes from Revision * (December 2019) to Revision A (November 2020)	Page
• Updated for release v2.0.0 throughout entire document	3

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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 will 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 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.

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265