Application Note

TPS6594/LP8764 PMIC Application Guide (II)

Lijia Zhu

ABSTRACT

This document was translated from a simplified Chinese source. (ZHCAFE7)

With the rapid development of new energy vehicles, the automotive industry is shifting from electrification to intelligence. Advanced driver assistance systems (ADAS) are being quickly deployed by major OEMs and Tier 1 suppliers. As a result, the TDA4 is being widely used across various terminal applications such as ADAS domain controllers, body domain controllers, and LiDAR. TI's new-generation PMIC family, represented by the TPS6594 and LP8764, is not only an ideal power solution for the TDA4 SOC but also a good choice for other SOCs. The TPS6594 and LP8764 offer high integration, high scalability, and support for high-level functional safety. With numerous and complex features, they are challenging to implement. This series of articles will cover topics about the TPS6594 and LP8764 PMICs, such as key mechanisms, system design considerations, common issue-locating strategies, and custom PMIC firmware (NVM). This is the second article in the series, continuing the discussion from the previous one on the main mechanisms within the PMIC chip.

Table of Contents

List of Figures

1 Introduction

The TPS6594 and LP8764 families are TI's next-generation PMICs, offering high integration, scalability, and functional safety. The TPS6594 integrates five BUCK converters and four LDOs internally, along with a rich set of GPIOs that can be configured via the NVM to achieve a variety of functions, including external wake-up input, watchdog feed input, enable input, external voltage monitoring input, and enable control for external high-side switches/LDOs/BUCKs, further enhancing the scalability of the device. The LP8764 chip shares a similar architecture with the TPS6594 and is often used in systems as a secondary PMIC for the TPS6594. The TPS6594 and LP8764 use SPMI interfaces for state synchronization among multiple PMICs, and the state machines of multiple PMICs operating in parallel can be considered as a whole. In terms of functional safety, they are developed to satisfy ISO26262 requirements during the design phase and incorporate a number of functional safety detection mechanisms to support ASIL D functional safety.

2 Overview of Main Modules

The TPS6594 and LP8764 have similar functional modules, and the main internal modules of the chip are described here using the TPS6594 as an example. We can roughly divide the TPS6594 into the following modules:

1. VSYS input control module
2. Power distribution module
3. Voltage monitoring module
4. BUCK module
5. LDO module
6. IO module
7. Digital logic module

Modules 1-6 were described previously; this article will focus on the digital logic module. The digital logic module is the control core of the entire PMIC. Its simplified architecture is shown in Figure 2-1, consisting of several main modules, such as the NVM (EEPROM) that stores the PMIC operating logic, the register map for recording states, the FSM module that controls the operating logic, the watchdog and ESM mechanisms that ensure proper SOC operation, and the SMPI mechanism for multi-chip collaboration. All of these modules will be covered in this article.

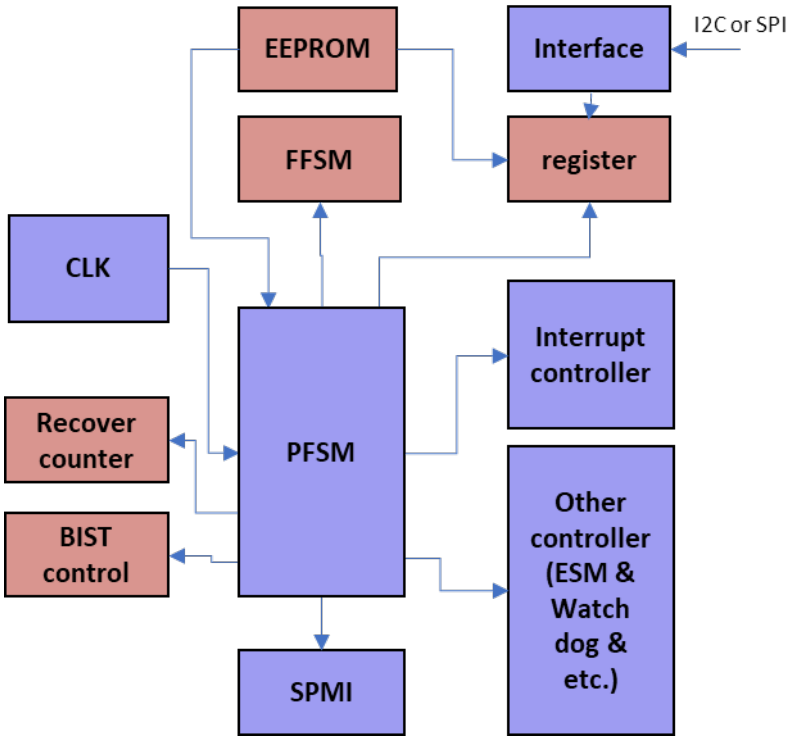

Figure 2-1. TPS6594 Digital Logic Module Block Diagram

2.1 NVM and Registers

2.1.1 Register Paging

The registers of the TPS6594 are divided into several pages, and registers in each page have specific functions:

Page 0: User Registers, including the registers that are open to the customer, covering the configuration and status of various modules (BUCK, LDO, IO, and so forth)

Page 1: NVM Control, Configuration, and Test Registers, including registers for NVM configuration and registers for test modes

Page 2: Trim Registers, which cannot be overwritten via the communication interfaces

Page 3: SRAM for PFSM Registers, containing the operating code for the PFSM

Page 4: Watchdog Registers. These are watchdog-related registers open to customer access

The external MCU has access to the TPS6594 registers using SPI or I2C; the switching between these two modes is determined by the SERIAL_IF_CONFIG in the NVM.

In I2C mode, the chip has two I2C interfaces and allows access to registers in Pages 0-3 using I2C1. Page 0 is accessed using the PMIC's preset I2C1 address, which is configured by the I2C1_ID_REG register in the NVM. Use the I2C1 address + 1 as the address to access registers in Page 1, and so on for subsequent pages. Page 4 is quite special; it has an independent I2C2 address configured by the I2C2_ID_REG register. When the I2C2 watchdog feed mode is not enabled, I2C1 occupies five addresses (I2C1_ID, I2C1_ID+1, I2C1_ID+2, I2C1_ID+3, and I2C2_ID), allowing access to registers in all five pages. However, when the I2C2 watchdog feed mode is enabled, I2C1 occupies four addresses (I2C1_ID, I2C1_ID+1, I2C1_ID+2, and I2C1_ID+3) for access to registers in Pages 0-3, and I2C2 occupies one address (I2C2_ID) for access to registers in Page 4.

In SPI interface mode, page information is distinguished by specific page fields in the SPI packet. See also the relevant manuals for details.

2.1.2 NVM and Register Map

There are two relatively independent storage spaces within a PMIC: the NVM and the register map. The NVM is non-volatile and stores the device's default power-up operating states, while the data in the register map is lost when VCCA powers down. These two spaces interact with each other. During device power-up, the NVM content is loaded into the register map, and the device subsequently operates according to the content in the register map. Conversely, the data in the register map can be written to the NVM; this is the method for updating the NVM via an external interface. It is important to note that accessing register data requires VIO to be powered. In some power distribution network (PDN) designs, VIO is derived from the PMIC's output. For boards using such PDNs, VIO loses power during fault conditions, making it impossible to access registers. In this case, an external VIO supply is required.

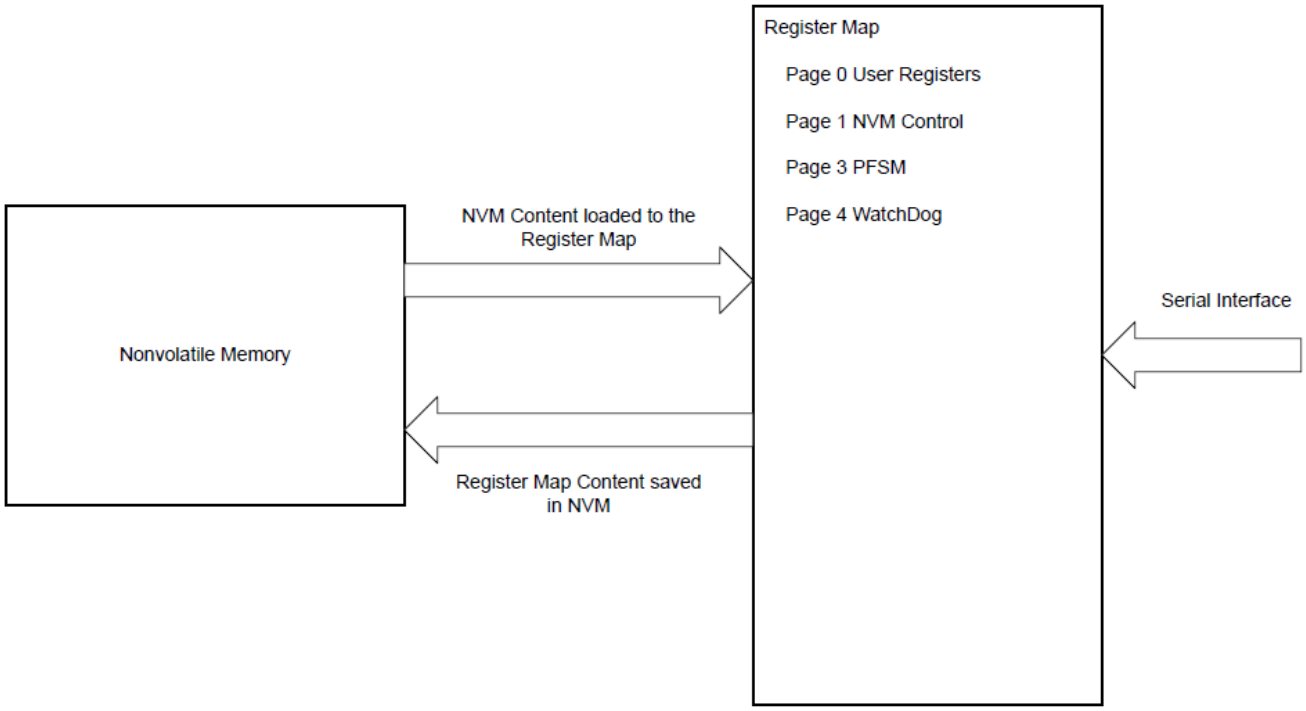

Figure 2-2. Relationship Between NVM and Registers in TPS6594

2.2 Finite State Machine (FSM)

The TPS6594 integrates a Finite State Machine (FSM), which controls the PMIC's behavioral logic by switching between different states. It consists of three sections:

The hardware-fixed FSM section, or the Fixed Finite State Machine (FFSM). It is hardware-fixed and controls logic that includes fixed functions common to all applications.

The programmable FSM section configured by NVM, or the Pre-configurable Finite State Machine (PFSM). Its logic is determined by the content of NVM Page 3. The PFSM works in coordination with the FFSM to leverage the strengths of both, significantly reducing the high complexity of a fully configurable FSM while providing sufficient flexibility to change state machine logic for different applications.

The interrupt handling mechanism, which controls the handling and reporting of interrupts, as well as their interaction with the FSM. It handles different interrupts through the MASK mechanism, determining whether they are reported to the MCU (pulling the interrupt pin low) and whether they affect the FSM.

2.2.1 FFSM

The FFSM controls the device operating logic before the start-up of PMIC rails and after their failure. It is important to note that ENABLE_DRV is always low when the PMIC is in any of the FFSM states.

As shown in Figure 2-3, the FFSM may experience the following states:

1. NO SUPPLY: The device is off. The system enters the INIT state when VCCA > UV
2. BACKUP: Also called the RTC backup battery. Only the RTC clock is active in this state. The system enters this state from any other state when VCCA < UVLO. In this state, if VRTC is also undervoltage, the system enters the NO SUPPLY state. This state can be disabled by directly disconnecting VCCA (shelf mode), causing the system to enter the NO SUPPLY state directly.
3. LP_STANDBY: Low-power standby mode. Only the RTC domain is active in this mode, and the device can be woken up by the RTC timer or the LPWKUP pin. In the mission state (of the PFSM section), the system enters this state when an off-request interrupt signal is received, and LP_STANDBY_SEL is set to 1. If LP_STANDBY_SEL is set to 0, the PFSM only enters the STANDBY state even if an off request is received.

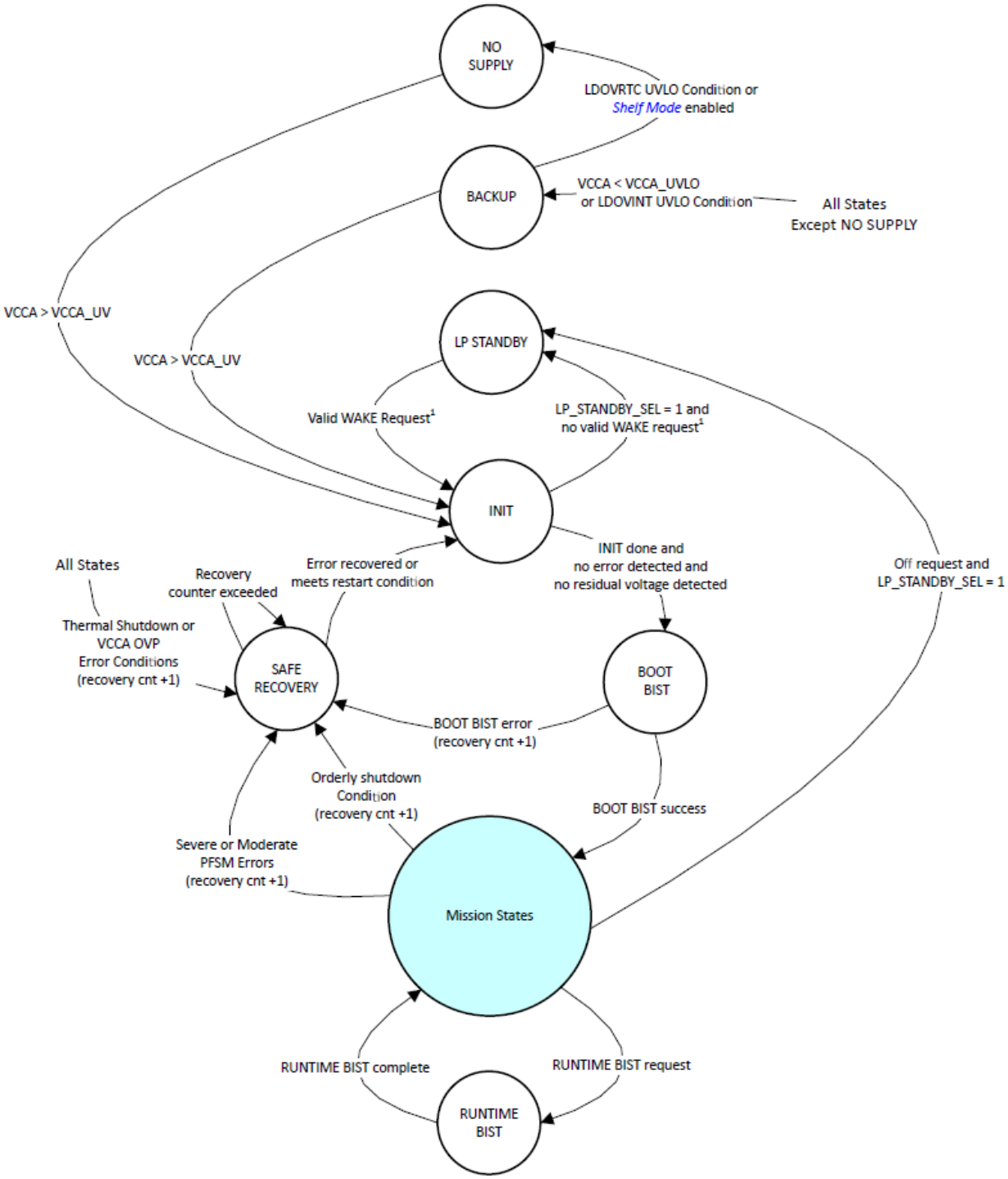

Figure 2-3. TPS6594 FFSM Architecture

4. INIT: Initialization state. The system enters this state when VCCA ≥ VCCA_UV, or the device is woken up from the LP_STANDBY state. In this state, the NVM content is loaded into the register map.
5. BOOT BIST: The self-test phase during boot. The system automatically enters this state after the INIT phase. It includes two parts: LBIST and ABIST. The LBIST includes the self-tests of internal digital modules (for example, PFSM, WD, clock, and so forth), and the ABIST includes power-on self-tests of VMON OV/UV for each power rail and the self-test of the over-temperature monitoring module. The LBIST and ABIST run synchronously with no specific order. Therefore, it is possible for both BIST PASS and BIST FAIL interrupts to be reported simultaneously. When the BIST fails, the system enters the SAFE RECOVERY state and

restarts the BOOT BIST. Only the ABIST is performed during the transition from SAFE RECOVERY back to BOOT BIST. Some registers can be used to disable some BOOT BIST functions, as described below:

a. Set FAST_BOOT_BIST to 1 to skip the LBIST during the transition from NO SUPPLY to BOOT BIST, thereby speeding up the boot process.
b. Set FAST_BIST to 1 to skip the LBIST during the transition from LP_STANDBY to BOOT BIST, thereby speeding up the wake-up process.
c. Set REG_CRC_EN to 0 to disable the CRC check function for registers.
d. Set VMON_ABIST_EN to 0 to disable the BOOT BIST for VMON OV/UV.

An additional note on the BIST for VMON OV/UV: Figure 2-4 shows the diagrams for the VMON OV/UV BIST of the BUCK and LDO. After each rail starts up, the BIST for the corresponding VMON OV/UV is enabled after a defined delay (startup delay + ramp-up time + setting time).

During the 50us window shown in the blue dashed box, the OV/UV results are masked from the state machine. Meanwhile, the thresholds for OV and UV are swapped during the first 25us, during which the BIST logic expects both OV and UV to be triggered. In the subsequent 25us, the thresholds are restored to their normal operating settings, and the BIST logic expects neither OV nor UV to be triggered. It is clear that within this 50us window, the rail must reach the expected voltage value, or the BIST fails, causing the interrupt register to report BIST FAIL. This condition typically occurs when the rail ramps up too slowly or the output voltage deviates from the expected voltage during power-up. Note that in these cases, the PMIC does not report UV or OV alerts.

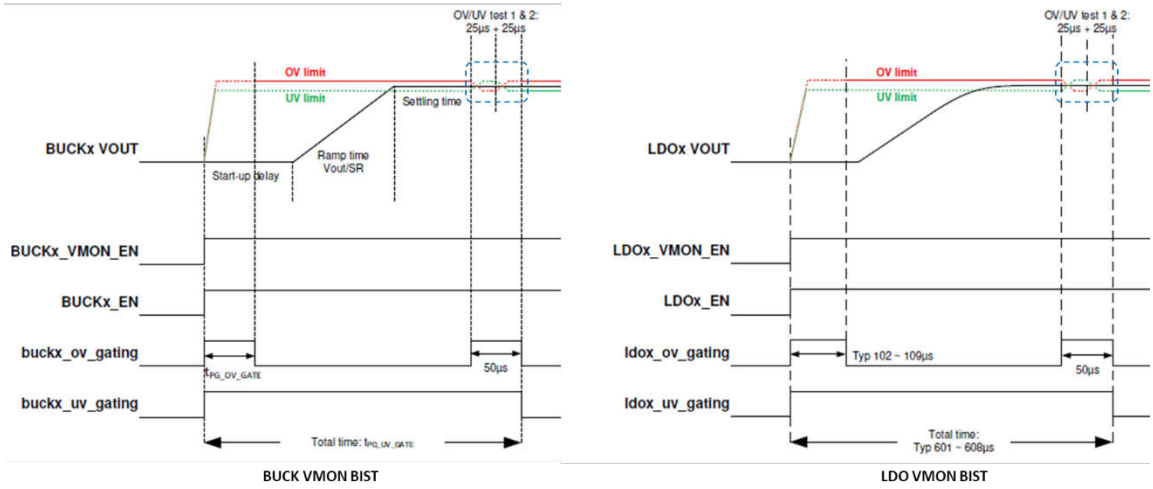

Figure 2-4. TPS6594 VMON OV/UV BIST Diagram

6. RUNTIME BIST: The BIST operation actively initiated by the MCU while the device is running. If the BIST passes, the system continues to run, and the BIST PASS interrupt is asserted. If the BIST fails, the system enters the SAFE RECOVERY state, and the BIST FAIL interrupt is asserted.
7. SAFE RECOVERY: The system enters this state when a fault occurs. All outputs are turned off. When the fault disappears within the recovery time (this delay varies depending on the fault) and the recovery counter does not exceed its threshold, the state machine enters the INIT state to attempt restarting, and the recovery counter increments by 1. After the recovery counter reaches its upper threshold, the chip must be restarted (VCCA < UVLO) for the PMIC to resume output. The recovery counter result is stored in bits 0-3 of the 0x83 register, and the recovery counter upper threshold (RECOV_CNT_THR) is set via bits 0-3 of the 0x84 register.

   During normal system operation, if the system occasionally enters the SAFE RECOVERY state or a SOFT REBOOT command is written to manually restart the PMIC, the recovery counter accumulates. The MCU clears the recovery counter before it reaches the upper threshold to prevent the count from accumulating and reaching the threshold. Once the recovery counter reaches the upper threshold, the PMIC will not reboot even if the counter is cleared; the chip must be restarted.

   Some special systems (such as debug) require the PMIC to restart indefinitely in fault scenarios rather than staying in the SAFE RECOVERY state after the counter reaches the upper threshold. This can be

achieved by configuring RECOV_CNT_THR to 0x0 in the NVM. Note that modifying the RECOV_CNT_THR register to zero during operation does not achieve the same effect, because RECOV_CNT_THR in the NVM overrides the 0 value in the register when the system recovers from SAFE RECOVERY through the INIT state.

8. Mission State: To be precise, it is not a state, but a PFSM nested within the FFSM. The PMIC goes through the BIST before entering the PFSM section.

2.2.2 Interrupt Handling Mechanism

The interrupt handling mechanism must be understood before we move on to the PFSM section. The TPS6594 has a hierarchically maskable interrupt mechanism. Each event (interrupt source) has two effects: being reported to the external MCU (pulling the interrupt pin low) and triggering a state transition of the FSM. Figure 2-5 shows a diagram from the manual. The top layer contains all faults and events. After passing through the first layer of interrupt masking logic, unmasked interrupts are output to the interrupt pin and simultaneously fed to the next layer of logic where the PFSM resides. Based on the power group configuration, each power rail sends its faults to the corresponding power group. The power group then aggregates these faults and reports the result to the FSM.

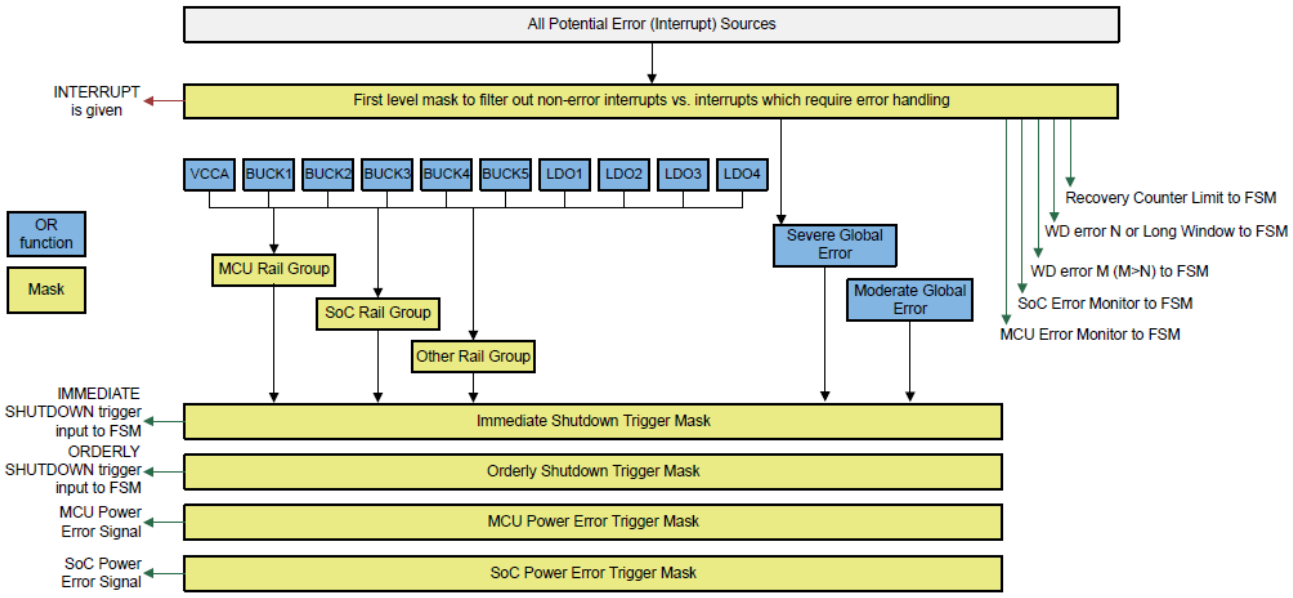

Figure 2-5. TPS6594 Interrupt Control Architecture

Figure 2-5 is relatively hard to understand, especially when the MASK mechanism herein conflates the MASK for reporting to the MCU with the MASK for acting as an FSM trigger. To understand the interrupt mechanism in more depth, see also the Summary of Interrupt Signals table in the manual. Figure 2-6 and Figure 2-7 are excerpted as examples. The table can be divided into three parts:

1. On the far left, we have a list of possible events, which can be external signals or internal faults, such as enable pull-ups and SPMI communication failures. They can also be direct signals or aggregated secondary interrupts, such as BUCK undervoltage and MCU power group error.
2. In the middle is the effect of each event on the FSM. The first column shows the FSM trigger name and whether it is maskable (providing the masking register if it is maskable). The second column lists the effect of each event on the FSM. The third column contains the condition for returning to the original state after the FSM state transition due to an event.
3. The rightmost section describes how the event affects the interrupt pin. The first column lists the interrupt register for each event (which remains latched after the event clears). The second column provides the register used to mask the interrupt from being reported. The third column shows the real-time status register for the event. The fourth column describes how to clear the interrupt register.

From the example of the interrupt-FSM interaction shown in Figure 2-6, it can be found that the effect of the BUCK and LDO on the FSM goes through the power group stage, which corresponds to the lower left section in Figure 2-5.

As shown in Figure 2-7, there are a large number of events that directly act as triggers for the FSM, in addition to the BUCK and LDO events. All FSM triggers are comprehensively described in the PFSM Trigger Selections table in the manual. A snippet is shown in Figure 2-8.

EVENT	TRIGGER FOR FSM	RESULT (1)	RECOVERY	INTERRUPT BIT	MASK FOR INTERRUPT	LIVE STATUS BIT	INTERRUPT CLEAR
BUCK regulator forward current limit triggered	**EN_ILIM_FSM_CTRL=1:** According to BUCKn_GRP_SEL and x_RAIL_TRIG bits **EN_ILIM_FSM_CTRL=0:** N/A	**EN_ILIM_FSM_CTRL=1:** Transition according to FSM trigger and interrupt **EN_ILIM_FSM_CTRL=0:** Interrupt only	Depends on PFSM configuration, see PFSM transition diagram	BUCKn_ILIM_INT = 1	BUCKn_ILIM_MASK	BUCKn_ILIM_STAT	Write 1 to BUCKn_ILIM_INT bit Interrupt is not cleared if current limit violation is active
LDO regulator current limit triggered	**EN_ILIM_FSM_CTRL=1:** According to LDOn_GRP_SEL and x_RAIL_TRIG bits **EN_ILIM_FSM_CTRL=0:** N/A	**EN_ILIM_FSM_CTRL=1:** Transition according to FSM trigger and interrupt **EN_ILIM_FSM_CTRL=0:** Interrupt only	Depends on PFSM configuration, see PFSM transition diagram	LDOn_ILIM_INT = 1	LDOn_ILIM_MASK	LDOn_ILIM_STAT	Write 1 to LDOn_ILIM_INT bit Interrupt is not cleared if current limit violation is active
BUCK output or switch short circuit detected	According to BUCKn_GRP_SEL and x_RAIL_TRIG bits	Regulator disable and transition according to FSM trigger and interrupt	Depends on PFSM configuration, see PFSM transition diagram	BUCKn_SC_INT = 1	N/A	N/A	Write 1 to BUCKn_SC_INT bit
LDO output short circuit detected	According to LDOn_GRP_SEL and x_RAIL_TRIG bits	Regulator disable and transition according to FSM trigger and interrupt	Depends on PFSM configuration, see PFSM transition diagram	LDOn_SC_INT = 1	N/A	N/A	Write 1 to LDOn_SC_INT bit
BUCK output residual voltage violation	**BUCKn_RV_SEL = 1** According to BUCKn_GRP_SEL and x_RAIL_TRIG bits **BUCKn_RV_SEL = 0** N/A	**BUCKn_RV_SEL = 1** Regulator disable and transition according to FSM trigger and interrupt **BUCKn_RV_SEL = 0** N/A	Depends on PFSM configuration, see PFSM transition diagram	BUCKn_SC_INT = 1	N/A	N/A	Write 1 to BUCKn_SC_INT bit
LDO output residual voltage violation	**LDOn_RV_SEL = 1** According to LDOn_GRP_SEL and x_RAIL_TRIG bits **LDOn_RV_SEL = 0** N/A	**LDOn_RV_SEL = 1** Regulator disable and transition according to FSM trigger and interrupt **LDOn_RV_SEL = 0** N/A	Depends on PFSM configuration, see PFSM transition diagram	LDOn_SC_INT = 1	N/A	N/A	Write 1 to LDOn_SC_INT bit
BUCK regulator overvoltage	According to BUCKn_GRP_SEL and x_RAIL_TRIG bits	Transition according to FSM trigger and interrupt	Depends on PFSM configuration, see PFSM transition diagram	BUCKn_OV_INT = 1	BUCKn_OV_MASK	BUCKn_OV_STAT	Write 1 to BUCKn_OV_INT bit Interrupt is not cleared if it is active
BUCK regulator undervoltage	According to BUCKn_GRP_SEL and x_RAIL_TRIG bits	Transition according to FSM trigger and interrupt	Depends on PFSM configuration, see PFSM transition diagram	BUCKn_UV_INT = 1	BUCKn_UV_MASK	BUCKn_UV_STAT	Write 1 to BUCKn_UV_INT bit Interrupt is not cleared if it is active
LDO regulator overvoltage	According to LDOn_GRP_SEL and x_RAIL_TRIG bits	Transition according to FSM trigger and interrupt	Depends on PFSM configuration, see PFSM transition diagram	LDOn_OV_INT = 1	LDOn_OV_MASK	LDOn_OV_STAT	Write 1 to LDOn_OV_INT bit Interrupt is not cleared if it is active
LDO regulator undervoltage	According to LDOn_GRP_SEL and x_RAIL_TRIG bits	Transition according to FSM trigger and interrupt	Depends on PFSM configuration, see PFSM transition diagram	LDOn_UV_INT = 1	LDOn_UV_MASK	LDOn_UV_STAT	Write 1 to LDOn_UV_INT bit Interrupt is not cleared if it is active

Figure 2-6. Interrupt Summary 1

EVENT	TRIGGER FOR FSM	RESULT (1)	RECOVERY	INTERRUPT BIT	MASK FOR INTERRUPT	LIVE STATUS BIT	INTERRUPT CLEAR
Thermal shutdown, orderly sequenced	ORDERLY_SHUTDOWN (MODERATE_ERR_INT)	All regulators disabled and Output GPIOx set to low in a sequence and interrupt(1)	Automatic start-up to STARTUP_DEST[1:0] state after temperature is below TWARN level	TSD_ORD_INT = 1	N/A	TSD_ORD_STAT	Write 1 to TSD_ORD_INT bit Interrupt is not cleared if temperature is above thermal shutdown level
Thermal shutdown, immediate	IMMEDIATE_SHUTDOWN (SEVERE_ERR_INT)	All regulators disabled with pull-down resistors and Output GPIOx set to low immediately and interrupt(1)	Automatic start-up to STARTUP_DEST[1:0] state after temperature is below TWARN level	TSD_IMM_INT = 1	N/A	TSD_IMM_STAT	Write 1 to TSD_IMM_INT bit Interrupt is not cleared if temperature is above thermal shutdown level
BIST error	ORDERLY_SHUTDOWN (MODERATE_ERR_INT)	All regulators disabled and Output GPIOx set to low immediately and interrupt(1)	Automatic start-up to STARTUP_DEST[1:0] state	BIST_FAIL_INT = 1	BIST_FAIL_MASK	N/A	Write 1 to BIST_FAIL_INT bit
Register CRC error	ORDERLY_SHUTDOWN (MODERATE_ERR_INT)	All regulators disabled and Output GPIOx set to low immediately and interrupt(1)	Automatic start-up to STARTUP_DEST[1:0] state	REG_CRC_ERR_INT = 1	REG_CRC_ERR_MASK	N/A	Write 1 to REG_CRC_ERR_INT bit
SPMI communication error	ORDERLY_SHUTDOWN (MODERATE_ERR_INT)	All regulators disabled and Output GPIOx set to low immediately and interrupt(1)	Automatic start-up to STARTUP_DEST[1:0] state	SPMI_ERR_INT = 1	SPMI_ERR_MASK	N/A	Write 1 to SPMI_ERR_INT bit
SPI frame error	N/A	Interrupt only	Not valid	COMM_FRM_ERR_INT = 1(4)	COMM_FRM_ERR_MASK	N/A	Write 1 to COMM_FRM_ERR_INT bit
I2C1 or SPI CRC error	N/A	Interrupt only	Not valid	COMM_CRC_ERR_INT = 1	COMM_CRC_ERR_MASK	N/A	Write 1 to COMM_CRC_ERR_INT bit
I2C1 or SPI address error(5)	N/A	Interrupt only	Not valid	COMM_ADR_ERR_INT = 1	COMM_ADR_ERR_MASK	N/A	Write 1 to COMM_ADR_ERR_INT bit
I2C2 CRC error	N/A	Interrupt only	Not valid	I2C2_CRC_ERR_INT = 1	I2C2_CRC_ERR_MASK	N/A	Write 1 to I2C2_CRC_ERR_INT bit
I2C2 address error(5)	N/A	Interrupt only	Not valid	I2C2_ADR_ERR_INT = 1	I2C2_ADR_ERR_MASK	N/A	Write 1 to I2C2_ADR_ERR_INT bit
PFSM error	IMMEDIATE_SHUTDOWN (SEVERE_ERR_INT)	All regulators disabled with pull-down resistors and Output GPIOx set to low immediately and interrupt(1)	Automatic start-up to STARTUP_DEST[1:0] state. If previous PFSM_ERR_INT is pending, VCCA power cycle needed for recovery.	PFSM_ERR_INT = 1		N/A	Write 1 to PFSM_ERR_INT bit

Figure 2-7. Interrupt Summary 2

Trigger Name	Trigger Source
IMMEDIATE_SHUTDOWN	An error event causes one of the triggers defined in the FSM_TRIG_SEL_1/2 register to activate, and the intended action for the activated trigger is to *immediate shutdown* the device
MCU_POWER_ERROR	Output failure detection from a regulator which is assigned to the MCU rail group (x_GRP_SEL = '01')
ORDERLY_SHUTDOWN	An event which causes MODERATE_ERR_INT = '1'
FORCE_STANDBY	nPWRON long-press event when NPOWRON_SEL = '01', or ENABLE = '0' when NPOWERON_SEL = '00'
SPMI_WD_BIST_DONE	Completion of SPMI WatchDog BIST
ESM_MCU_ERROR	An event which causes ESM_MCU_FAIL_INT
WD_ERROR	An event which causes WD_INT
SOC_POWER_ERROR	Output failure detection from a regulator which is assigned to the SOC rail group (x_GRP_SEL = '10')
ESM_SOC_ERROR	An event which causes ESM_SOC_FAIL_INT
A	NSLEEP2 and NSLEEP1 = '11'. More information regarding the NSLEEP1 and NSLEEP2 functions can be found under Section 8.4.1.2.4.3
WKUP1	A rising or falling edge detection on a GPIO pin which is configured as WKUP1 or LP_WKUP1
SU_ACTIVE	A valid On-Request detection when STARTUP_DEST = '11'
B	NSLEEP2 and NSLEEP1 = '10'. More information regarding the NSLEEP1 and NSLEEP2 functions can be found under Section 8.4.1.2.4.3
WKUP2	A rising or falling edge detection on a GPIO pin which is configured as WKUP2 or LP_WKUP2
SU_MCU_ONLY	A valid On-Request detection when STARTUP_DEST = '10'
C	NSLEEP2 and NSLEEP1 = '01', More information regarding the NSLEEP1 and NSLEEP2 functions can be found under Section 8.4.1.2.4.3
D	NSLEEP2 and NSLEEP1 = '00'. More information regarding the NSLEEP1 and NSLEEP2 functions can be found under Section 8.4.1.2.4.3
SU_STANDBY	A valid On-Request detection when STARTUP_DEST = '00'
SU_X	A valid On-Request detection when STARTUP_DEST = '01'

Figure 2-8. FSM Trigger Summary

The interrupt alert registers of the TPS6594 also feature a hierarchical mechanism. As shown in Figure 2-9, the 5A register aggregates all interrupt alerts, and each bit set indicates that the corresponding lower-layer register has an interrupt. This nesting can go up to three layers.

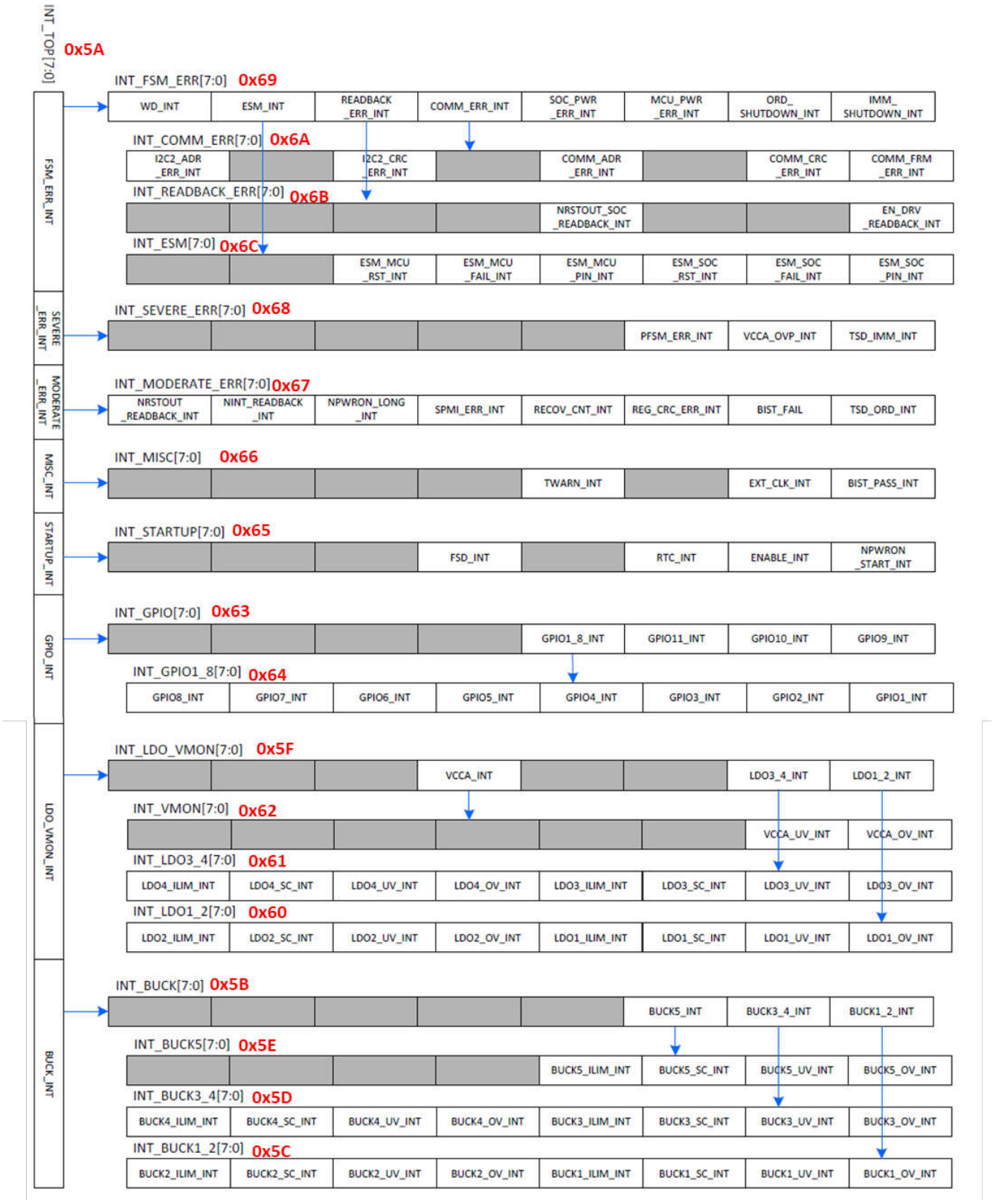

Figure 2-9. TPS6594 Interrupt Register Summary

2.2.3 PFSM

The PFSM controls specific power-up and power-down sequences of the device. Unlike the registers in other pages, the PFSM cannot be modified during operation. It is important to note that the PFSM architecture diagram in this section, referenced from the device manual, is just one example. Because of the editable nature of the PFSM, each PMIC has a different PFSM setup. Here we take the PDN 1A (TPS65941213 + TPS65941111) as an example. Figure 2-10 shows the entire FSM architecture, and documentation for this PDN is available on the TI website by searching for the keyword "PDN 1A". Documentation for other PDNs is also available by searching for their names.

Based on this PFSM, we can explain a common issue: If the ENABLE INIT interrupt is cleared after the PMIC starts up normally, the PMIC directly loses power:

1. As shown in Figure 2-11, there are two triggers for the state transition from standby to active: SU ACTIVE and WKUP1. We focus on the SU ACTIVE trigger. According to Figure 2-8, this trigger indicates that the enable pin is high; therefore, the external enable pin is pulled high, causing the PMIC to go active, and each rail outputs externally.
2. After start-up, the interrupt pin is low because of the BIST PASS and ENABLE INIT interrupts. All interrupt registers must be cleared to release the interrupt pin. The interrupt pin will be pulled low again when another fault occurs later, so that the MCU can handle the fault in a timely manner based on this signal.
3. If the ENABLE INIT register is cleared at this point, the condition for FORCE STANDBY, ENABLE = 0, is met. As shown in Figure 2-12, the PFSM enters the STANDBY state, and each rail is turned off.
4. As shown in Figure 2-13, a few more triggers may hold the FSM in the ACTIVE state. Here, we focus on trigger A. According to Figure 2-8, trigger A indicates that the NSLEEP bit is set to 11. Figure 2-14 shows the priority of each trigger. The larger the number, the higher the priority. Therefore, trigger A that holds the PFSM in the ACTIVE state has a higher priority than FORCE STANDBY, which triggers the PFSM to enter the STANDBY state.
5. As a result, by setting both NSLEEP0 and NSLEEP1 to 1 via I2C registers, a trigger A is activated, holding the PFSM in the ACTIVE state. In this case, regardless of clearing ENABLE INIT or pulling the ENABLE pin low, the PFSM stays in the ACTIVE state. All outputs remain unchanged, and the system does not power down.

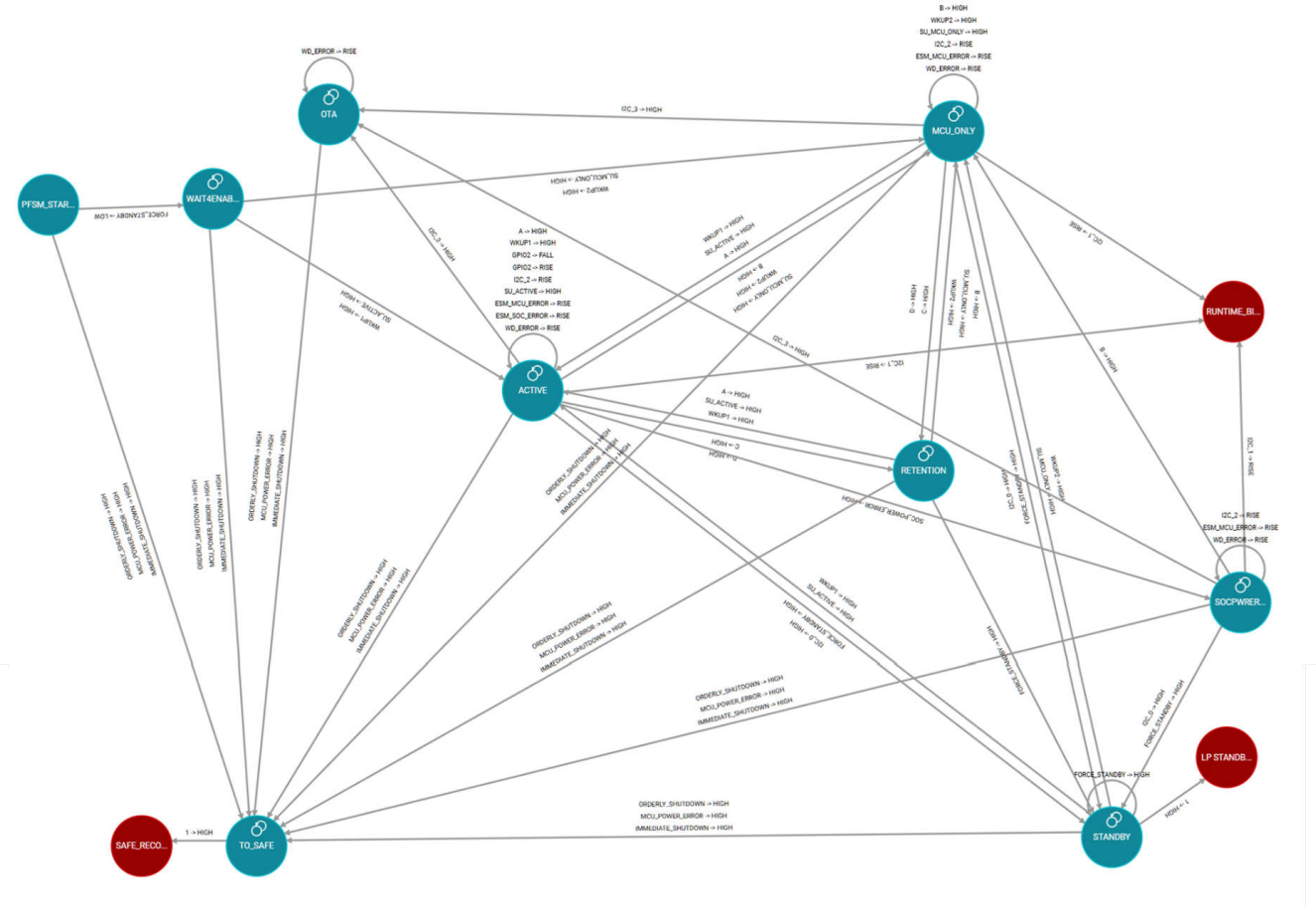

Figure 2-10. PDN 1A FSM Architecture

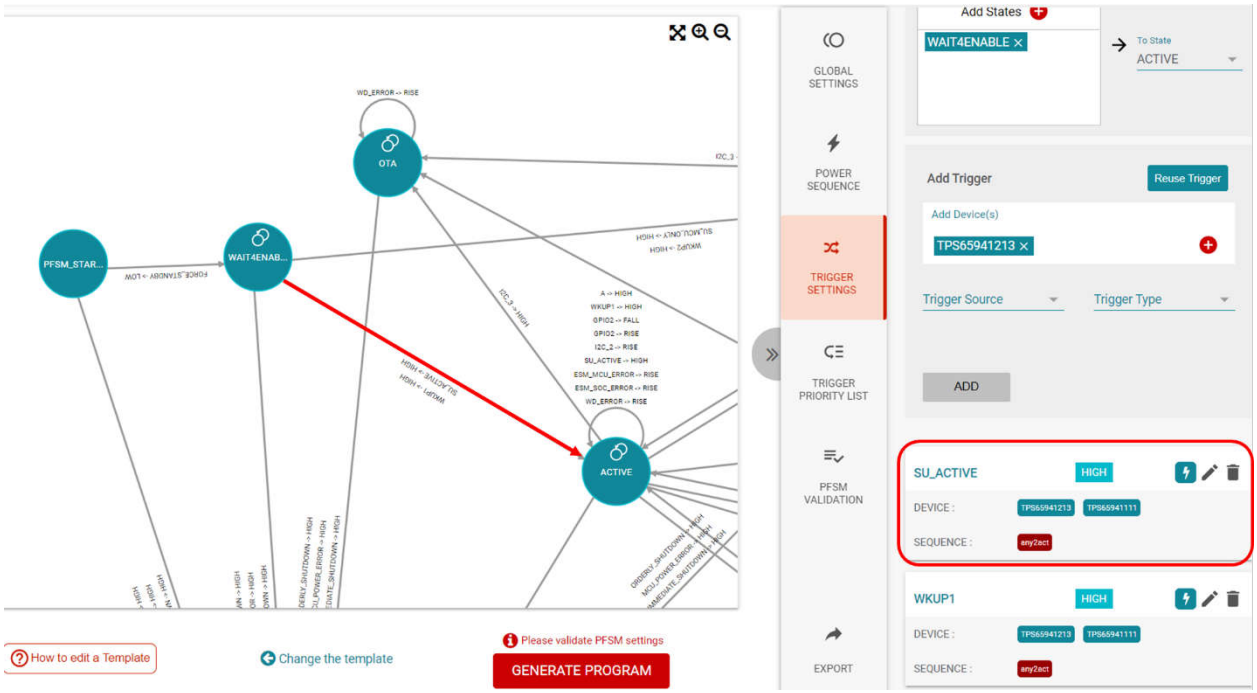

Figure 2-11. FSM Partial Architecture 1

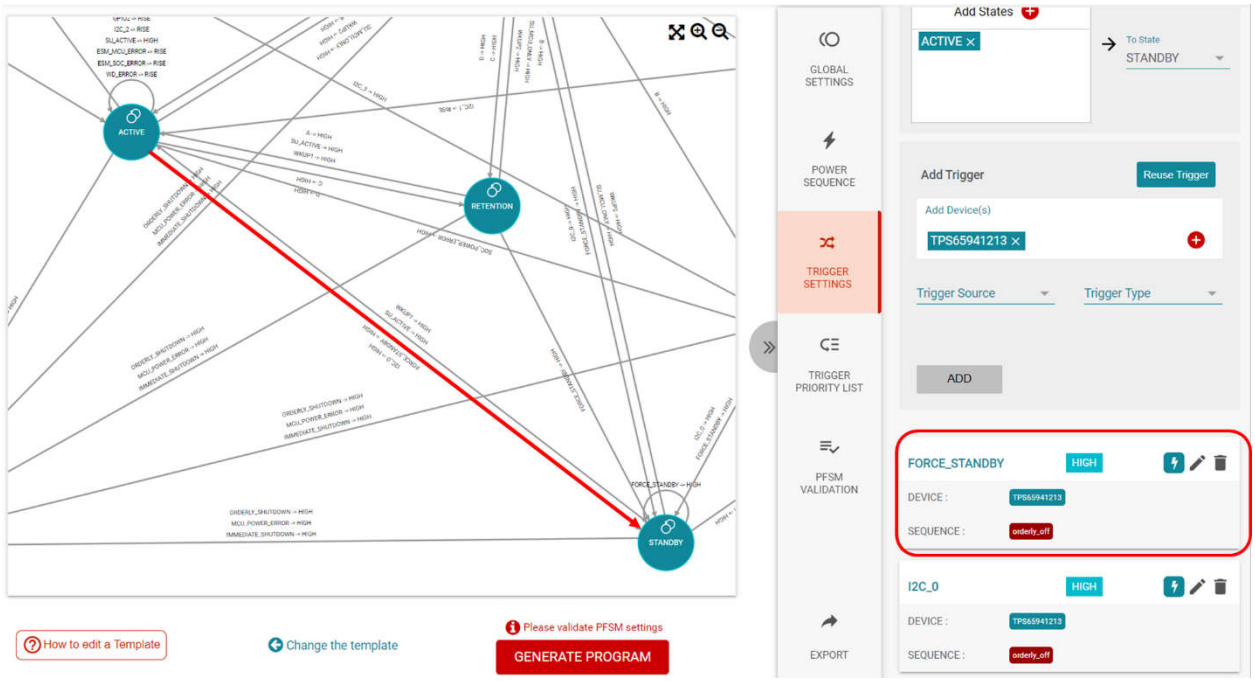

Figure 2-12. FSM Partial Architecture 2

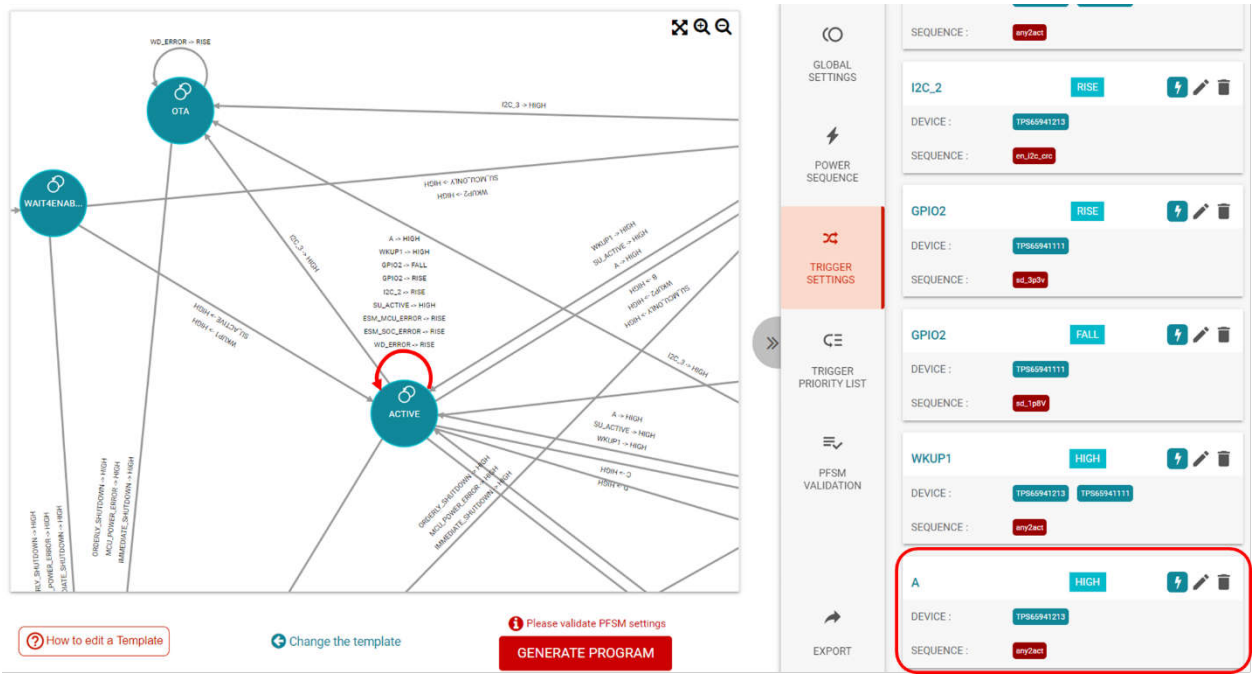

Figure 2-13. FSM Partial Architecture 3

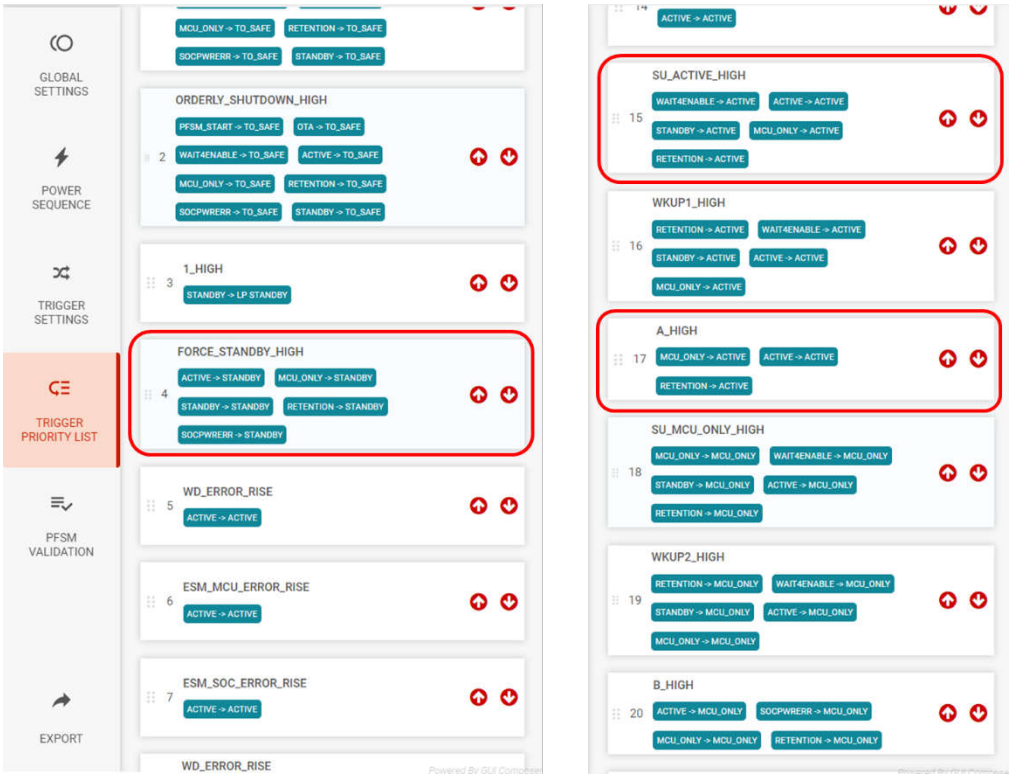

Figure 2-14. FSM Trigger Priority

2.3 SPMI Communication Mechanism

The System Power Management Interface (SPMI) is the communication protocol used on PMICs, as specified by the MIPI Alliance. Similar to I2C, SPMI uses two lines for communication: a clock line and a data line. It also allows multiple PMIC devices to be connected to a single bus. The SPMI module of the TPS6594/LP8764 is simplified as shown in Figure 2-15 and includes two parts: Controller and Target. When a PMIC is the primary PMIC, its Controller and Target modules are both enabled. There is only one primary PMIC in a system. The Controller module serves as the controller on the bus to initiate SPMI communication, and the secondary PMIC only enables its Target module to respond to the commands from the Controller and report its own status.

Similar to I2C, SPMI also has an addressing mechanism, divided into Controller ID (CID), Target ID (TID), and Group ID (GTID). In a system, the Controller has a unique CID of 1, and the Target module uses this CID to respond to the Controller's packets. The TPS6594 supports up to six TIDs (the Primary PMIC TID is 0101, and TIDs for secondary PMICs 1-5 are 0011, 1100, 1001, 0110, and 1010, respectively). A system can have at most six PMICs. When the Controller communicates with a specific Target, the packet uses the corresponding TID. All Target modules listen for messages on GTID 1111, and packets use this GTID when the Controller needs to broadcast information.

The TIDs of the secondary PMICs used in the system are sequentially recorded by five registers in the NVM of the primary PMIC. If the total number of PMICs in the system is less than six, the spare registers must be written to 0. Therefore, the combination of PMICs is not arbitrary. Incorrect, duplicate, and missing TIDs all cause SPMI communication failure, which requires attention in the application.

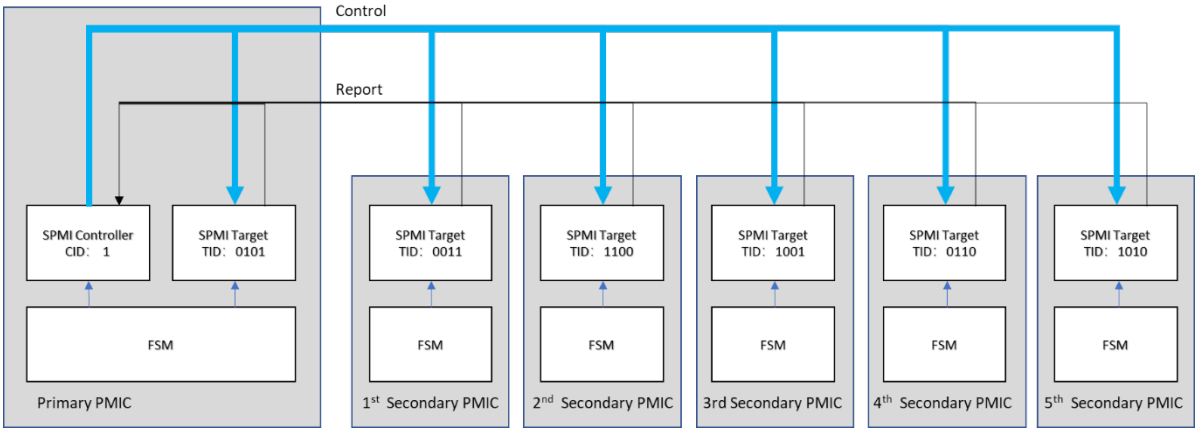

Figure 2-15. SPMI Block Diagram

There are two types of messages that are passed on the SPMI bus:

1. The status of the respective PFSM synchronized between the SPMI Controller and the SPMI Target.
2. The NVM ID that the SPMI Target passes to the SPMI Controller.

A SPMI Target communicates with the SPMI Controller or other SPMI Targets only in the event of an internal error that is not related to SPMI. When multiple Targets initiate communication, the priority is determined by the arbitration mechanism defined in the SPMI specification. The Target winning the arbitration uses GTID 1111 to communicate the status of the PFSM to all PMICs.

The SPMI Controller initiates the SPMI BUS BIST to check the validity of SPMI communication and verify whether the correct NVM is being used by the individual devices. It also initiates a broadcast command at intervals determined by the registers SPMI_WD_BOOT_INTERVAL and SPMI_WD_RUNTIME_INTERVAL. Each SPMI Target must respond with its own NVM ID. If any Target does not respond, the Controller keeps trying to resend until the SPMI module's internal watchdog times out (the timeout is configured by the SPMI_WD_RESPONSE_TIMEOUT register, with a default value of approximately 820ms). Figure 2-16 shows a normal SPMI communication waveform, where PDN 0A (TPS65941120 + TPS65941421 + LP876411B5) is used. Four data segments are marked with orange lines. The first segment is the broadcast initiated by the Controller, and the next three segments are the responses from three Targets. All four segments

are indispensable. It is possible to check whether the communication is abnormal by measuring the SPMI signal with a logic analyzer or oscilloscope. A common issue is that a soldering fault causes abnormal SPMI communication, and certain data segments are missing on the waveform in this case.

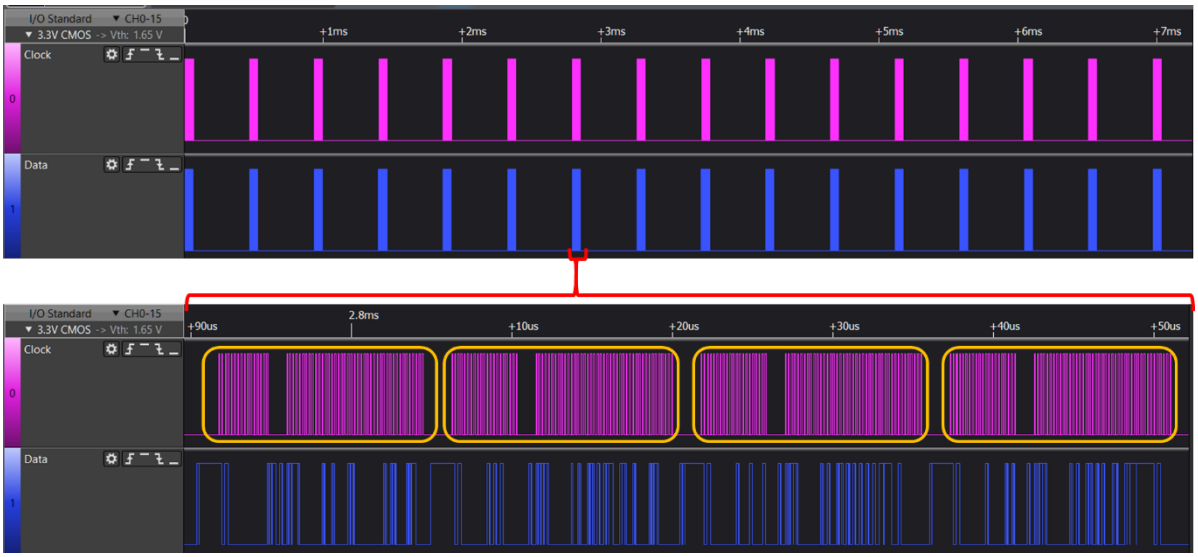

Figure 2-16. SPMI BIST Waveform

3 References

1. Datasheet "TPS6594-Q1 Power Management IC (PMIC) with 5 BUCKs and 4 LDOs for Safety-Relevant Automotive Applications"
2. Datasheet "LP8764-Q1 Four-Phase, 20-A Buck Converter With Integrated Switches"
3. Application Note "Scalable PMIC NVM Update Guide"
4. Application Note "TPS65941120-Q1, TPS65941421-Q1 and LP876411B5-Q1 PMIC User Guide for J721S2, PDN-0A"
5. Application Note "Optimized TPS65941213-Q1 and TPS65941111-Q1 PMI User Guide for Jacinto™ 7 J721E, PDN-0C"

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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

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

Last updated 10/2025