Application Report
SLUA815B–March 2017–Revised April 2019

UCD90xxx Family Frequently Asked Questions and Answers

Zhiyuan Hu, Yihe Hu, Brad Higgins

ABSTRACT

The UCD90xxx family devices are flexible and powerful to meet your sequencing and monitoring needs. This application report addresses frequently asked questions to give you a jump start.

Contents

List of Figures

List of Tables

Trademarks

SMBus is a trademark of Intel.
PMBus is a trademark of SMIF, Inc.
All other trademarks are the property of their respective owners.

1 Introduction

The UCD90xxx family of digital power supply sequencers, also known as system health monitors, are flexible and powerful to meet your sequencing, monitoring, margining, and other needs. The entire family is designed to have the same behavior with a different number of rails. That way, you only need to learn how to use the device once, and then you can seamlessly switch to other devices within the family that best fit future designs. This application report addresses some frequently asked questions to give you a jump start, and all answers apply to all UCD90xxx devices (other than the UCD9080 and UCD9081).

2 PMBus Communication FAQs

2.1 What are SMBus and PMBus?

The system management bus (SMBus™) is a two-wire interface through which various system component chips can communicate with each other and with the rest of the system. It is based on the principles of operating in I^2C. SMBus provides a control bus for system and power management related task. The protocol specification can be downloaded from http://smbus.org/specs/.

The power management bus (PMBus™) is an open standard protocol that defines a means of communication with power conversion and other devices over a digital communication bus (I^2C) and is a variant of the SMBus. PMBus defines a substantial number of domain-specific commands to communicate. The protocol specification can be downloaded from http://pmbus.org/Specifications/OlderSpecifications.

2.2 What hardware interface is required to communicate with UCD90xxx devices?

UCD90xxx devices use PMBus communication. Both configuration and monitoring use the same PMBus interface. UCD90xxx devices are PMBus slaves, so you need to have a PMBus host to communicate with them. This PMBus host can be an MCU or FPGA implementation that complies with PMBus specification Rev. 1.1. However, it is strongly recommended to use TI's PC-based software, Fusion Digital Power Designer GUI, to make development easy.

Fusion Digital Power Designer GUI requires a USB-to-GPIO adapter to connect between PC and the device. The USB-to-GPIO adapter information can be found at http://www.ti.com/tool/USB-TO-GPIO.

2.3 What is the PMBus CONTROL signal?

The CONTROL signal is an active-driven input signal on a PMBus device. The signal is used to turn on and off the device in conjunction with commands received through the serials bus. It can be configured as an active high or active low signal through PMBus Command. See the PMBus specification for details (http://pmbus.org/Specifications/OlderSpecifications).

This signal is optional and does not have a communication function, but it is recommended so that the Fusion GUI can toggle on and off the pin to the sequencer power rails purposely. Note that if you have routed the CONTROL pin to another active-driven source, such as the MCU or FPGA, you need to disconnect the trace before connecting the CONTROL pin to the USB-to-GPIO adapter in order to avoid conflict.

2.4 What is an ALERT signal?

An ALERT signal is an interrupt line for devices that want to trade their ability to master for a pin. ALERT is a wired-AND signal as SCL/SDA. A slave-only device can signal the host through the ALERT signal that it wants to talk. Routing it to the PMBus connector allows the Fusion GUI to monitor and display the ALERT pin state. This signal is optional and does not have a communication function, but it is recommended. Some users may use the ALERT pin as a fault source to shut down the system, but it is not recommended. Many other conditions can assert the ALERT pin, such as communication error and log full. The UCD90240, UCD90320 and UCD90320U devices support the SMBALERT_MASK command, which may be used to prevent a warning or fault condition from asserting the $\overline{\text{SMBALERT}}$ signal.

2.5 Is TI USB-to-GPIO configurable?

Yes.

You can configure the USB-to-GPIO adapter to select different bus speed (100 kHz or 400 kHz), packet error checking (enabled or disabled), and pullup settings (open-drain, 668 Ω, 1 kΩ, and 2.2 kΩ).

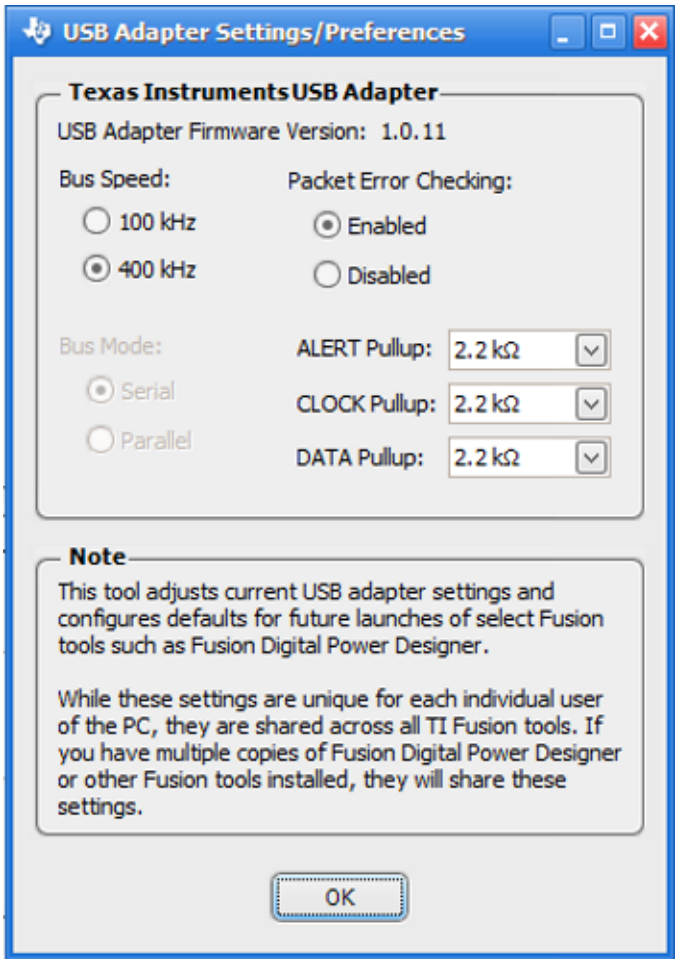

Figure 1. USB Adapter Settings and Preferences

When the adapter is used on TI UCD90xxx EVMs, all pullup settings other than open-drain can be used. The open-drain setting shall be used when the USB-to-GPIO dongle is connected to the customized board, which has the pullup resistor already.

The tool can be launched from the Fusion Digital Power Designer (Tools → SMBus & SAA Tool → SAA Settings).

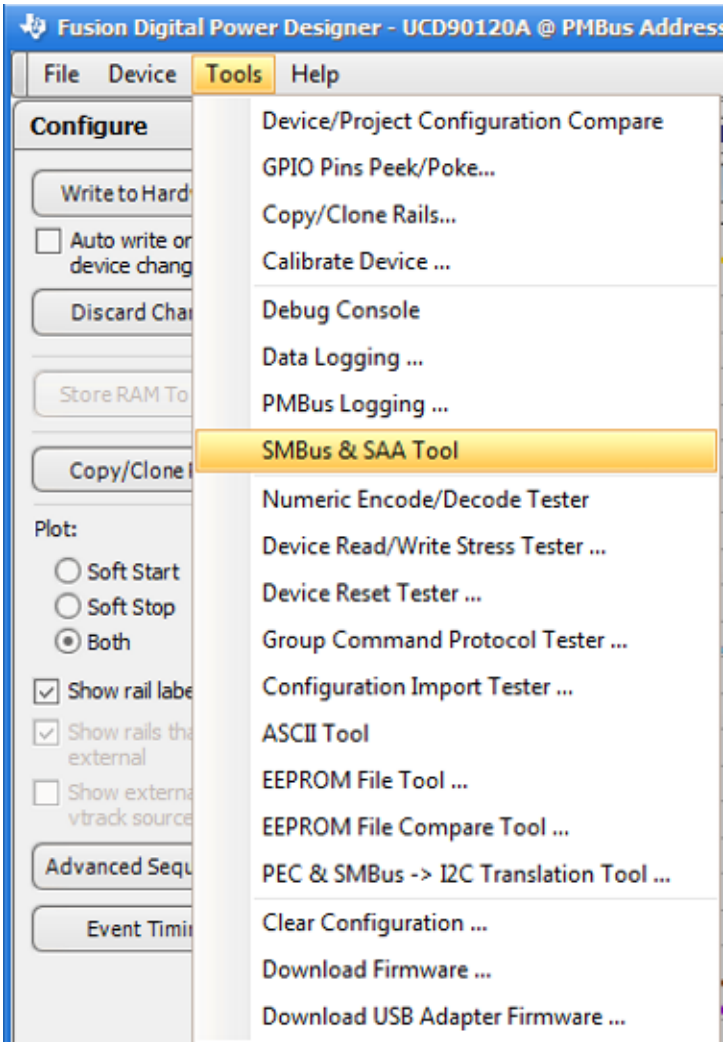

Figure 2. Launching From Fusion Digital Power Designer

The tool can also be launched from the UCD3xxx and UCD9xxx Device GUI.

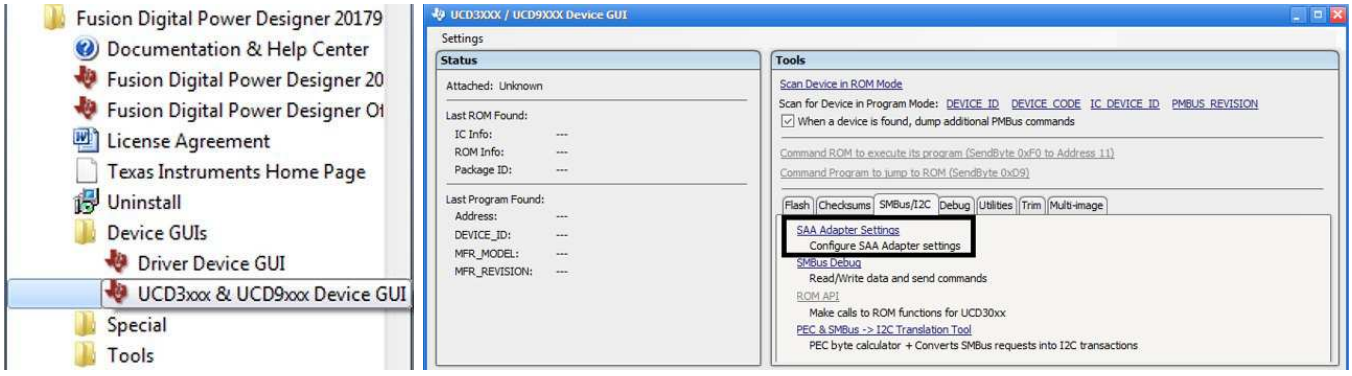

Figure 3. Launching From UCD3xxx and UCD9xxx Device GUI

2.6 Should the Fusion Digital Power Designer GUI be used?

Definitely.

The Fusion Digital Power Designer GUI serves three major functions:

1. Assist you to configure the device with a graphic interface so that you do not have to learn the PMBus commands
2. Generate configuration files and scripts for manufacturing
3. Assist you to monitor and debug prototype boards

In the Fusion GUI, by hovering the mouse on a configuration field, the Tips & Hints window explains the associated PMBus command and device behavior.

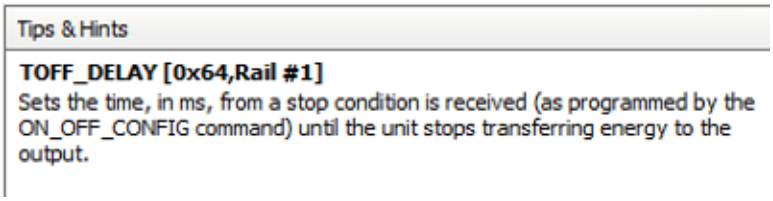

Figure 4. Tips & Hints Window

Even if you decide to develop their own PMBus host, the Fusion GUI is a valuable tool. This GUI can show you how a particular task is done through PMBus commands so that you do not have to dig into the documents.

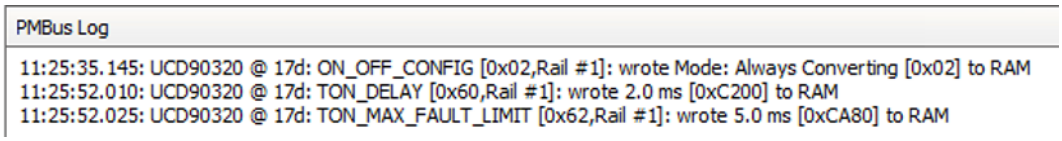

Figure 5. PMBus Log

Also, it has a PMBus Logging feature, which generates a text file to record all background communication activities between the Fusion GUI and the device. You can use the log to learn how to develop own PMBus host.

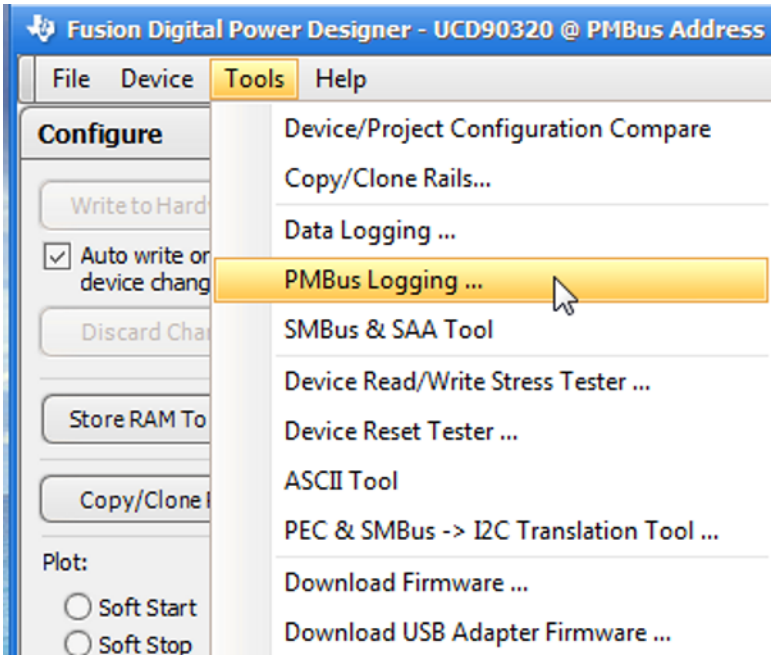

Figure 6. PMBus Logging Feature

The Fusion GUI also comes with the SMBus and SAA Tool (low-level PMBus communication GUI) to help you debug communication problem, as well as the Device/Project Configuration Compare tool to help you see the differences among several configuration files.

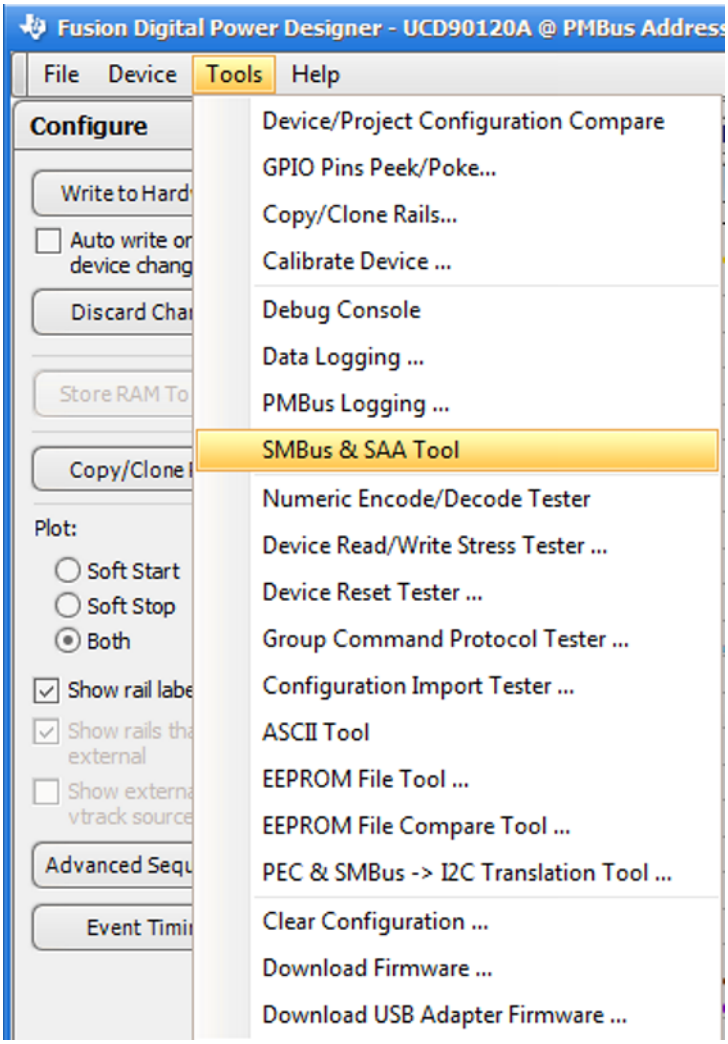

Figure 7. SMBus & SAA Tool Feature

2.7 Is a UCD90xxx PMBus command reference required?

Not necessarily, and it is up to what you plan to do. However, do use it as an ultimate reference. Each function in the Fusion GUI has a corresponding PMBus command. The UCD90xxx PMBus Command Reference contains not only protocol information but also a description of the device behavior of each function. If a customized software driver is required to develop to communicate with the UCD90xxx device, this UCD90xxx PMBus Command Reference is necessary.

2.8 Why are some PMBus commands not described in the UCD90xxx PMBus command reference document?

The UCD90xxx PMBus Command Reference contains mostly manufacturer specific commands (see *UCD90xxx Sequencer and System Health Controller PMBus Command Reference*).

There are also many standard PMBus commands described in PMBus Specification Rev 1.1 (http://pmbus.org/Specifications/OlderSpecifications).

PMBus Spec Rev. 1.1 is a protocol layer on top of SMBus Spec Ver 2.0. The protocol format such as Read Byte, Write Byte, Read Word, Write Word, and so forth, are defined in the SMBus Spec (http://smbus.org/specs/).

2.9 Does the TI Fusion GUI provide a general purpose PMBus or I²C tool?

Yes, the SMBus & SAA Tool mentioned in Section 2.6 allows you to communicate to any I²C or PMBus device. The tool can be launched from either:

- Fusion Digital Power Designer GUI (Tools → SMBus & SAA Tool)
- Fusion Digital Power Designer → Tools → SMBus & I2C & SAA Debug Tool

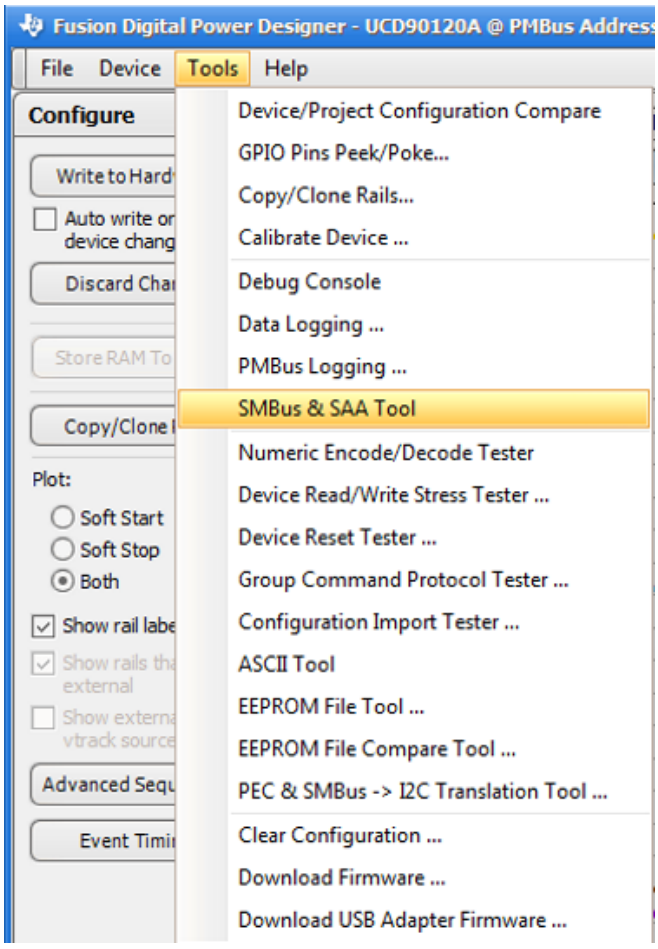

Figure 8. Launching From Digital Power Designer GUI

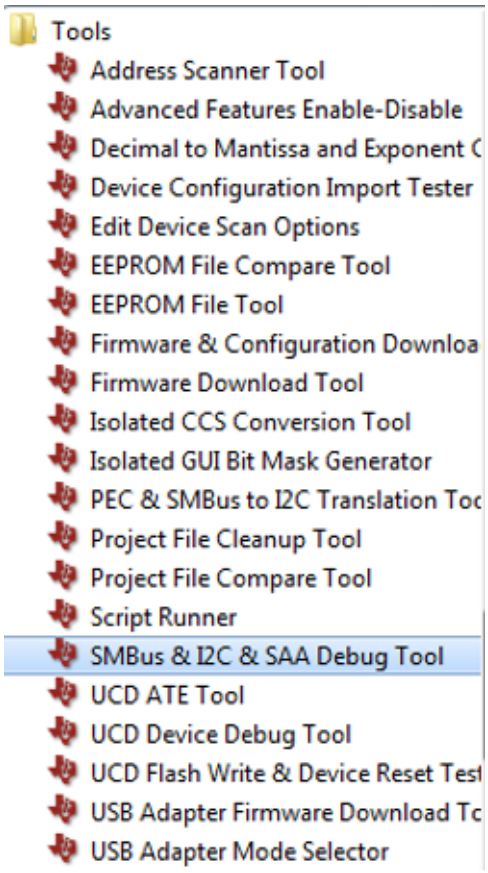

Figure 9. Launching From Digital Power Designer Directory

2.10 How many PMBus masters can be used on the same bus?

Only one host at a time. Communication is corrupted when more than one PMBus hosts try to send information at the same time. The USB-to-GPIO adapter is a PMBus host.

2.11 Is clock stretching required by the PMBus master?

Yes. UCD90xxx devices may hold clock low while processing the PMBus inquiry. According to PMBus specifications, a host needs to support clock stretching for up to 35 ms. It is recommended leaving a 1-ms interval between two PMBus inquiries; otherwise, the UCD90xxx may occasionally hold clock low for 35 ms and then timeout (hardware release clock line).

2.12 Is packet error code required by the PMBus master?

It is optional. UCD90xxx devices send and receive a packet error code (PEC) if the PMBus host supports it. No special configuration is needed.

A PEC is a CRC-8 checksum appended after each transaction. It enables devices to detect communication errors and discard corrupted messages. It is always recommended to implement PECs in the host.

2.13 Can the configuration be protected?

Yes. The UCD90xxx has a security feature to protect against unintended or unauthorized flash write. You can turn on security by using the Fusion GUI (the security option is only available when the Fusion GUI connects to an online device), or by sending a corresponding PMBus command.

For safety reasons, the passcode is not stored in project file (.xml).

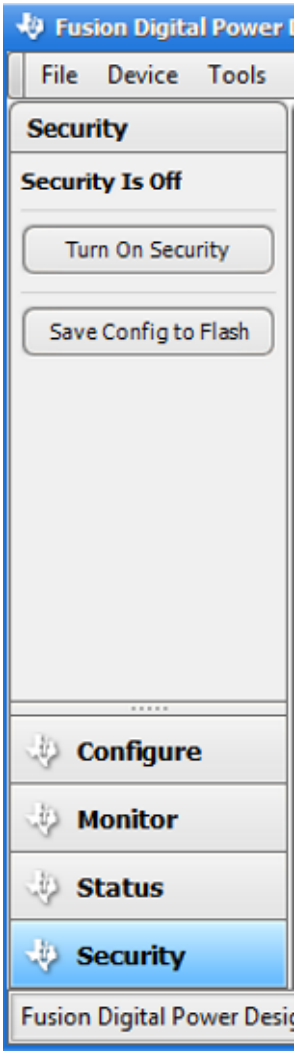

Figure 10. Security Feature

3 UCD90xxx Programming FAQs

3.1 What are the options to program UCD90xxx devices?

See *Configuration Programming of UCD Devices* for details on the UCD90240, UCD90320 and UCD90320U devices (http://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/212/1007.Configuration-Programming-of-UCD-Devices.pdf).

On a physical level, all UCD90xxx devices support programming through a PMBus port, and some support a JTAG port (JTAG not supported by the UCD90320U, UCD90320, and UCD90240 devices).

PMBus programming includes two general approaches:

1. Sending parameters individually using the project file (.xml)
2. Sending data flash image using the data flash file (.hex)

The project file contains all the configurations. UCD90xxx devices receive parameters individually through corresponding PMBus commands and validate the value ranges before storing into the SRAM and execute the new settings. This is safe, but takes a relatively long time to communicate. The settings take effect immediately, which may generate some glitches on the IO output.

The data flash file is an image of the device flash. The PMBus host transfers data into the UCD90xxx by chuck, and thus the communication is relatively fast. There is a checksum included in the data flash file. If the data flash content does not match the checksum, the UCD90xxx does not execute the configuration and behaves as a blank device. In this way, the UCD90xxx is also protected against communication error. The settings transferred by the image file do not take effect until a reset or power cycle is triggered.

The Fusion GUI can only export a data flash file from a connected online device. For both approaches, the Fusion GUI can generate a script to download a project file or data flash file into the device. You only need to execute the script according PMBus protocol, without the need to learn each PMBus command. It is recommended to implement a PEC in the PMBus host, which provides an additional layer of protection.

JTAG programming does not have the layers of protection as PMBus because JTAG has direct access to flash. If not executed correctly, it may "brick" the device. Also, due to the many versions of JTAG, you must verify that their JTAG tool is compatible with the device. The best way is to try the tool on the EVM.

There are three ways to program the devices in production:

1. Download the configuration to the device at the first board power-up.
2. Use a programmer to download configuration before assembly.
3. Use a third-party service.

TI does not directly support a programmer. Some programmer venders include System General, BP Micro, Elenc, and so forth. Some distributors provide a programming service such as Arrow and Avnet.

3.2 How do users validate the flash content after programming?

To verify data flash content, use one of the following methods. By using a data flash script generated by the Fusion GUI, you have the option to include flash read back and check value.

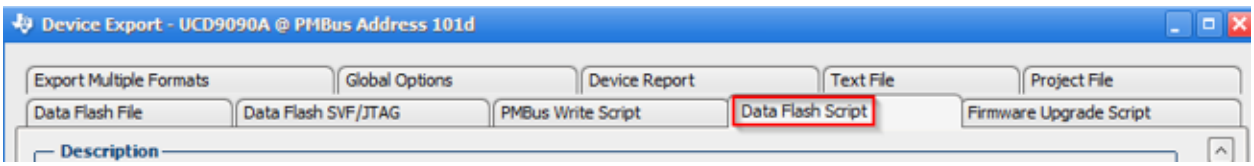

Figure 11. Using Data Flash Script

1. After a device reset, if flash content does not match the checksum value stored in the flash, the device sets the Hardcoded Parms bit in the MFR_STATUS. You can read this bit to detect unsuccessful programming.

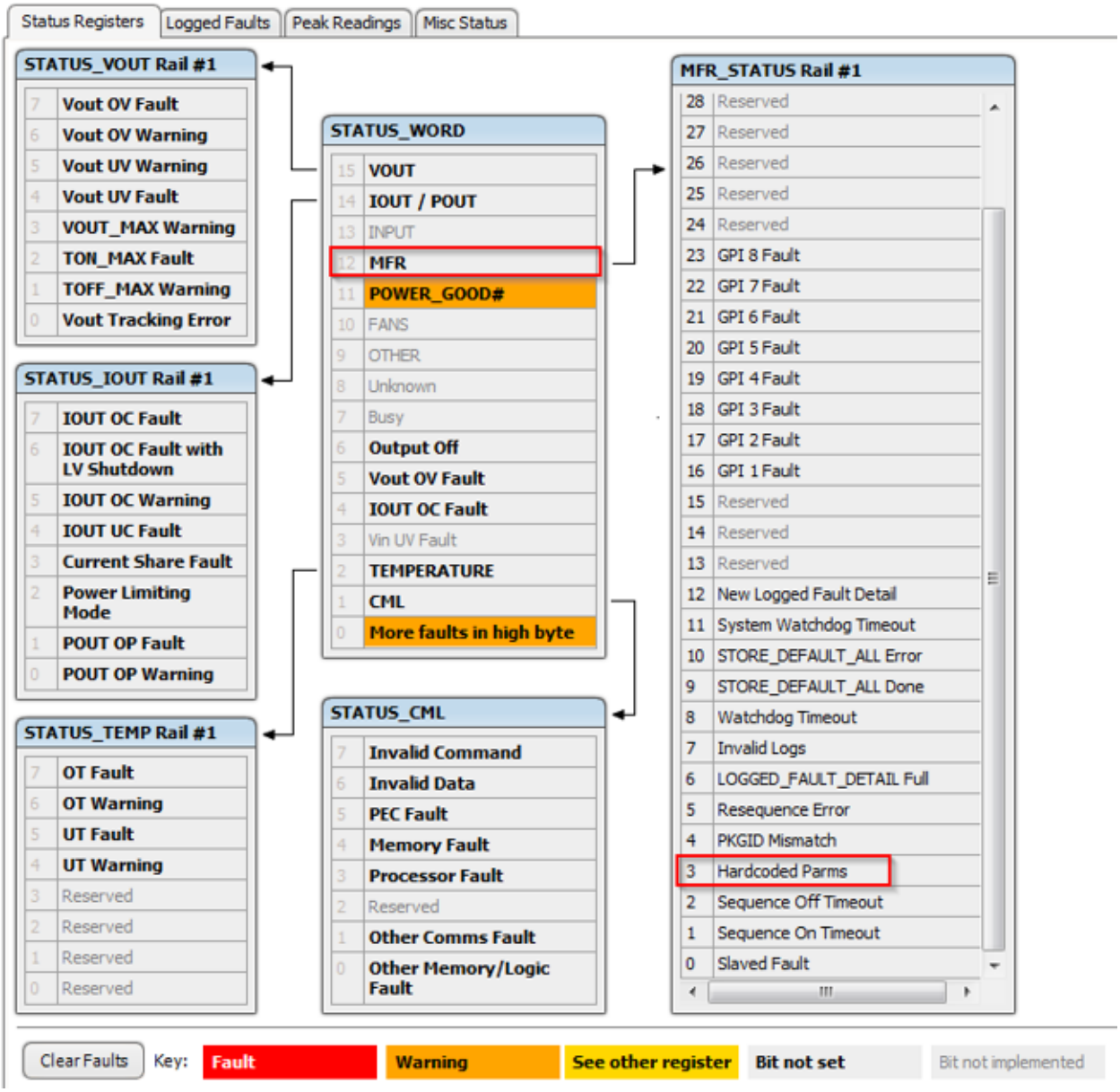

Figure 12. Status Registers

2. After a device resets, if flash content does not match the checksum value stored in the flash, the device ignores the flash content and behaves as a blank device. Then you can query number of pages configured through the NUM_PAGES command. If the return is zero, it means the checksum validation has failed.
3. You can query other known user configured strings such as MFR_ID, MFR_MODEL, MFR_REVISION, MFR_SERIAL, and so forth. If the returned value is correct, it means the whole data flash content is accepted and validated by the firmware.

3.3 Why does the read back of the entire data flash not match the programming image file?

The data flash (0x18800 through 0x19000) contains two areas: user configuration data and log. Once the device is up and running, the log area is updated runtime. Therefore, the flash content in the log area is different from the original image. The user configuration area locations and sizes are different among the device family members as shown in Table 1; the remaining area is for log.

Table 1. User Configuration Area Locations and Sizes (1)

DEVICES	UCD90120A	UCD9090	UCD90124A	UCD90160	UCD90910
VALID ADDRESS	0x18800 — 0x18D9F	0x18800 — 0x18CFF	0x18800 — 0x18E1F	0x18800 — 0x18DDF	0x18800 — 0x18DFF
DEVICES	UCD90120	UCD90124	UCD90124N	UCD9090A	UCD90160A
VALID ADDRESS	0x18800 — 0x18DDF	0x18800 — 0x18E5F	0x18800 — 0x18E1F	0x18800 — 0x18D1F	0x18800 — 0x18DFF

(1) Logs of the UCD90240, UCD90320 and UCD90320U devices are stored in a separate EEPROM.

3.4 Is data flash checksum useful?

Data flash checksum is a static value stored in the data flash. During device initialization, the device calculates the actual checksum value based on data flash content and compares it with the static checksum value. If the actual checksum does not match the static checksum, the data flash content is compromised. Therefore, for the UCD9090, UCD90120/UCD90120A, UCD90124/UCD90124A, and UCD90160/UCD90160A devices, reading the static checksum value after programming does not reflect the actual data flash content.

The UCD90240, UCD90320, and UCD90320U devices validate checksum before writing it to flash. If the checksum does not match the received configuration data, the device does not write the checksum into the flash. Instead, the device erases flash for next configuration programming. Therefore, for UCD90240, UCD90320, and UCD90320U devices, reading back checksum validates that the programming was successful.

3.5 When importing a project file into a device, why do some GPIO pins unexpectedly turn on and off?

This is expected behavior. When importing a project file into the device, the configurations are imported by corresponding PMBus commands in a sequence. The imported parameters take effect immediately. In the middle of the process, the half-configured device is running with partial configuration, and thus the GPIO pins may have unexpected behavior. After importing, a reset is needed for the device to operate correctly.

If this behavior is a problem, you can use data flash file (.hex file) programming. With data flash file programming, all the changes do not take effect until the device is reset. This action prevents unexpected GPIO pin behavior during configuration programming.

3.6 What is a system file?

The Fusion GUI can save the configurations of multiple devices on the same bus into a system file. With a system file, you can download configurations to multiple devices at once.

3.7 What is a data flash script file?

The Fusion GUI can read the data flash from the device, parse the content, package the content with particular PMBus command sets to meet the device requirement, and store it into a "script" file (.csv) that can be used by a microcontroller or other I^2C master to download data flash content to the device. The .csv file is a very good reference for those who want to implement their own I^2C download tool to download the configuration (.hex).

3.8 What is a PMBus write script file?

The Fusion GUI can detail the writes necessary to write current configuration to a device into a script file. The PMBus Write Script file is composed of standard SMBus or PMBus commands.

The PMBus write script file is different than a data flash script file. The former accesses the RAM of the device directly while the latter accesses the data flash directly. This format is not support by the UCD90320U, UCD90320, and UCD90240 devices.

3.9 Can the GUI generate a data flash image file (.hex) offline?

No. A data flash image file can only be generated from a live device or a data flash script file. When offline, the GUI can generate a project file (.xml), which can be imported to the live device to generate a .hex file and PMBus write script file.

4 UCD90xxx Sequencing FAQs

4.1 How many rails can each UCD90xxx device support?

The number of supported rails vary depending on the device. Be aware that some UCD90xxx devices have more MON pins than the number of rails it can support. The additional MON pins can be assigned as current sense or temperature sense pins. See the UCD90xxx datasheet for details.

4.2 How do users know that the sequencing dependencies are met?

UCD90xxx devices support three different events as dependencies: Rail, GPI, and LGPO (UCD9090A, UCD90160A, UCD90320, and UCD90320U only).

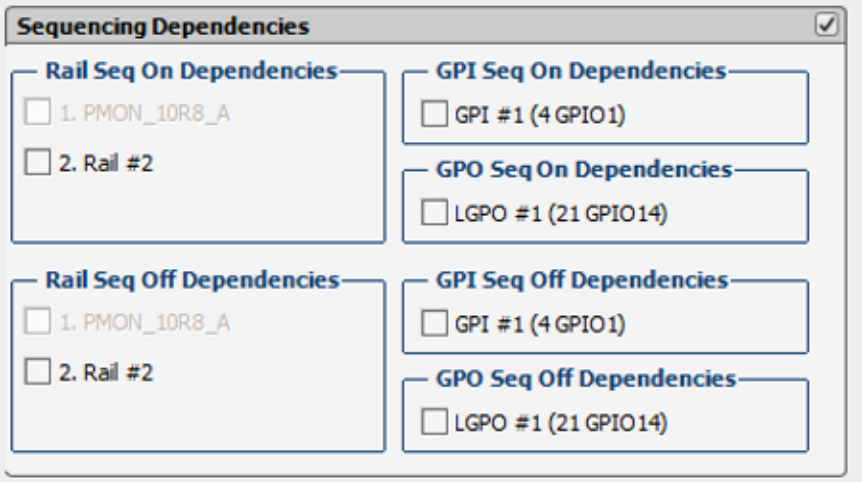

Figure 13. Sequencing Dependencies

Table 2. Sequencing Dependencies Events

EVENT	RAIL		GPI	LGPO
Sequence ON condition Met	Voltage monitoring: Above POWER_GOOD threshold	Other monitoring: EN signal is asserted	ASSERTED(1)	The logic output is TRUE
Sequence OFF condition Met	Voltage monitoring: Below POWER_GOOD_OFF threshold	Other monitoring: EN signal is de-asserted	DE-ASSERTED(1)	The logic output is FALSE

(1) The input signal is ASSERTED if it matches the defined active polarity; otherwise, it is DE-ASSERTED.

4.3 Why do the rails not start or stop?

First, check if the EN pin output from the UCD90xxx is at the correct state. If not, it means the On/Off Config or dependency conditions are not met for the rail to start or stop. If the state is correct but the rail does not start or stop, check the POL controller.

4.4 Why are there no undervoltage faults? For some reason, the rails do not start.

The UCD90xxx only starts to monitor undervoltage (UV) faults after the rail voltage reaches the Power Good On threshold for the first time. Before that, the rail stays in a "Ramp up" state, and thus UV fault is not monitored.

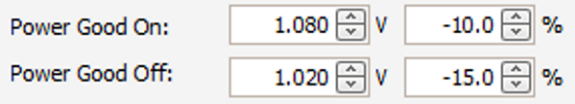

Figure 14. Power Good Monitoring

To detect a rail failure to reach the Power Good On threshold within a specified time period after EN pin assertion, set the Max Turn On threshold to a finite value.

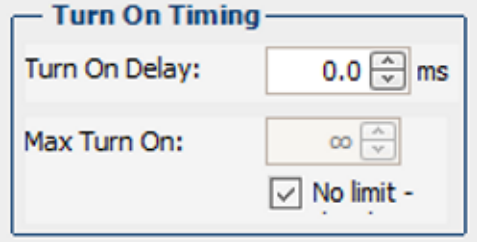

Figure 15. Turn On Timing

4.5 What are the sequencing states?

Figure 16 shows how the sequencing works.

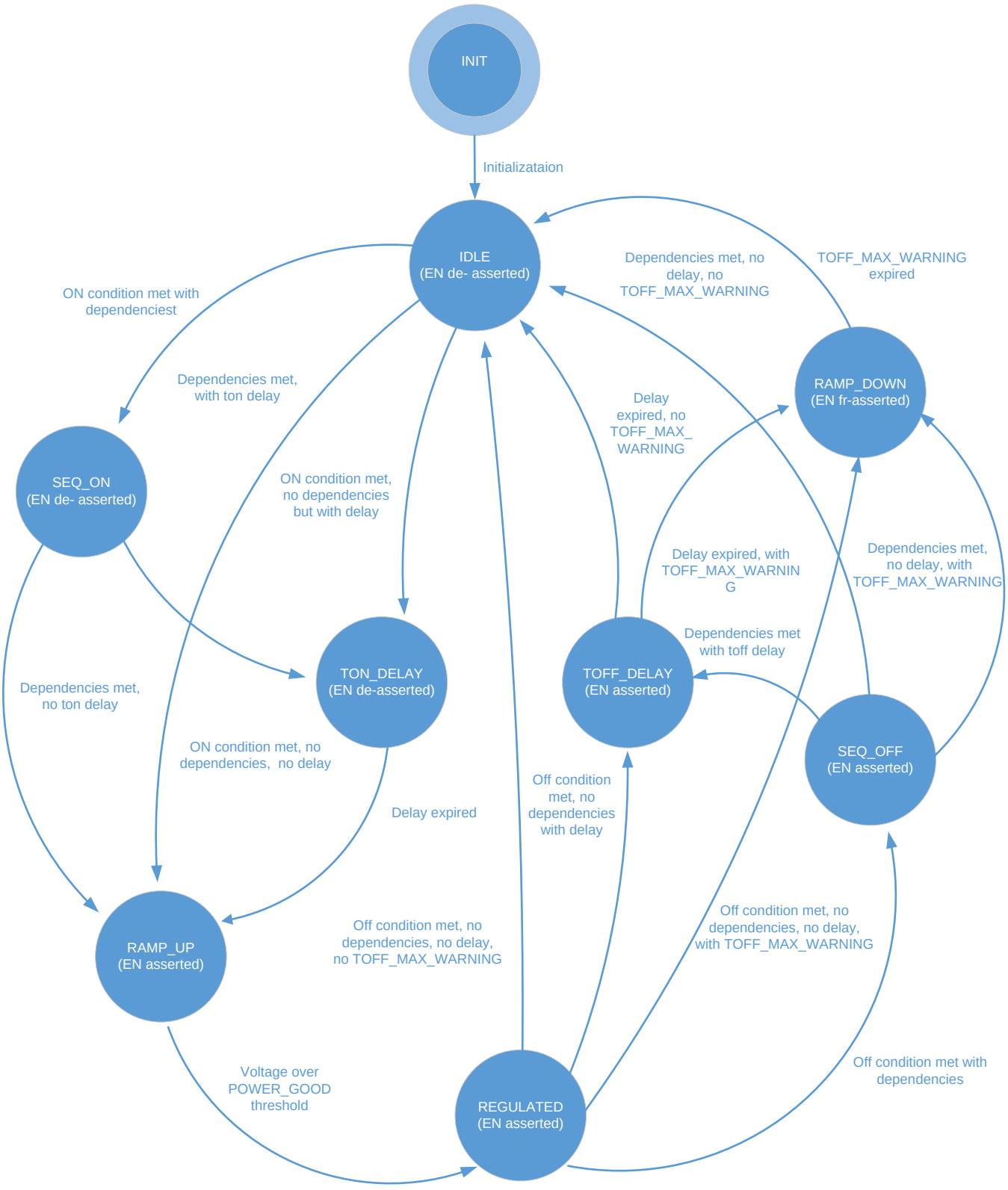

Figure 16. State Machine Diagram

4.6 Why does the GPI not turn off the rails when it is de-asserted if the GPI is configured as a rail's sequence-off dependency?

Each rail has its own ON/OFF Config setting. In the setting, there are three sources that can control rail on and off:

1. OPERATION: the rail attempts to turn on and off when it receives an OPERATION command.
2. CONTROL pin: the rail attempts to turn on and off when the CONTROL pin state is toggled.
3. NONE (Auto Enable): the rail automatically attempts to turn on once the UCD90xxx device is out of reset.

Dependencies are only examined after the rail attempts to change the on and off state. Therefore, toggling the GPI sequence-off dependency alone does not turn off the rail. A rail must first receive instructions to change state and then checks dependency conditions to determine the actual timing to change its EN pin state.

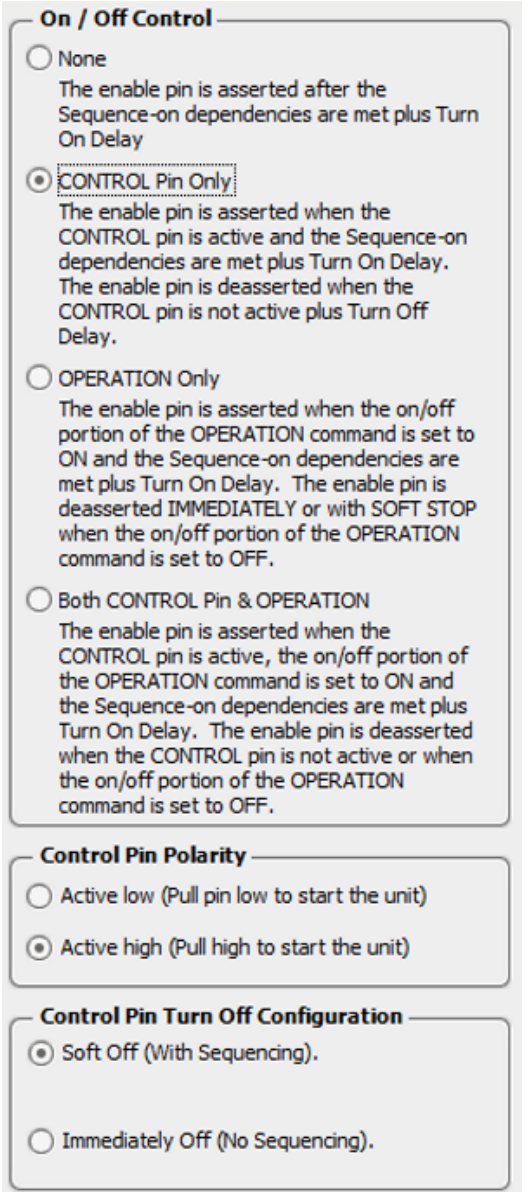

Figure 17. On and Off Control

Another mechanism to sequence down rails is fault response (overvoltage (OV), undervoltage (UV), overcurrent (OC), undercurrent (UC), over-temperature (OT), time on max, GPI fault, and so forth). A rail can turn off immediately according to its fault response setting. Each rail can include several other rails as fault shutdown slaves. The slave rails turn off according to their respective dependency and delay settings. There is one condition: a rail must shut down itself in fault response configuration in order to shut down its slaves. If a rail ignores the fault in its fault response configuration, it does not shut down its slaves.

The UCD9090A, UCD90160A, UCD90240, UCD90320, and UCD90320U devices support GPI faults to turn off rails. The UCD9090, UCD90160, UCD90120, UCD90120A, UCD90124, and UCD90160 devices do not support GPI faults to turn off rails.

Figure 18. Fault Shutdown Slaves

4.7 How should users implement a power-down sequence when the rails ON_OFF_CONFIG are set to AUTO Mode?

Auto Enable means the UCD90xxx device automatically sequences on rails once the device is powered up. It is not controlled by the CONTROL pin or OPERATION command, so you cannot use the CONTROL pin or OPERATION command to sequence off rails. When ON_OFF_CONFIG is set to AUTO mode, the only source that can initiate a power-down sequence is by fault response.

To initiate a "normal" power down sequence (as opposed to a fault shutdown, which can trigger fault logging and the ALERT line), one way is to select the CONTROL Pin or OPERATION command in the On/Off Config setting instead of using Auto Enable (or None). Then you need to assert the CONTROL pin or send an OPERATION command to sequence on the rails, and likewise, de-assert the CONTROL pin or send an OPERATION command to sequence off the rails. The CONTROL Pin is a special logic IO pin as part of PMBus port. The OPERATION command is a PMBus command. In the Fusion Digital Power Designer GUI, you can toggle a CONTROL pin state or send an OPERATION command inside the Monitor page.

Another way to sequence off rails is through the Pin Selected States feature. You can define up to three GPI pins to toggle among up to eight rail states. In each state, you can program certain rails to be on and certain rails to be off. For the simplest case, you can have two states, toggled by one GPI pin. In State 0, all rails are off; in State 1, all rails are on. During the transition of two states, the rails that are going to change turnon and turnoff states follow their respective delay and dependency settings in sequenced manner. The Pin Selected states can be configured through the Fusion GUI. Note that to use the Pin Selected States feature, the On/Off Config setting must be in OPERATION Only or Both CONTROL Pin and OPERATION.

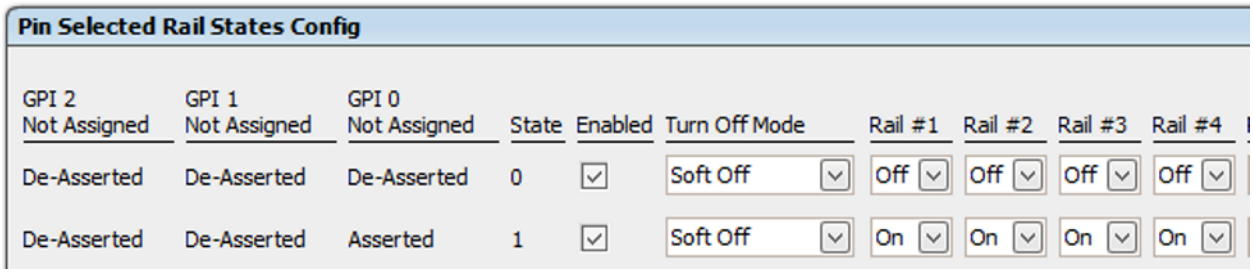

Figure 19. Pin Selected Rail States Configuration

4.8 What are the fault shutdown slaves?

Fault shutdown slaves are the salve rails of the faulted rail. Each rail can have its own fault slaves rails (as shown in Figure 18). When a fault occurs on the master rail, if its response is to shut down, all slave rails are also shut down. If retries are specified for the master rail, the slave rails remain running until all retries are exhausted. For the UCD90120 and UCD90124 devices, the slave pages are shut down in the same way as the master page is (immediately or soft stop). For devices other than the UCD90120 and UCD90124, the slave pages are shut down using sequence off dependencies and TOFF_DELAY. The slave pages do not perform any retries during the fault slave shutdown. After being shut down, slave rails are latched off as if they had experienced the fault. A status bit is set in their MFR_STATUS word indicating the reason they are latched off. If a re-sequence is enable at fault response of master rail, the fault shutdown slaves will be re-sequenced along with master rail.

If the fault response of master rail is set to ignore and continue operation, the fault shutdown slaves are meaningless.

4.9 What are restart (retry) and re-sequence?

Restart only disable-then-enable the faulted rail to see whether it can recover from the previous fault. Restart is only applied on the faulted rails. The device repeats this until the restart is exhausted.

Re-sequence is applied on the fault rails plus its fault shutdown slaves. All rails designated as fault shutdown slaves of the faulted rail are disabled based on its own shutdown sequence first. Then the device tries to sequence on all rails to see whether they can recover from the previous fault.

After the number of restart is exhausted, re-sequence takes place. The maximum number of re-sequence is configured under the Global Configuration tab → Misc Config.

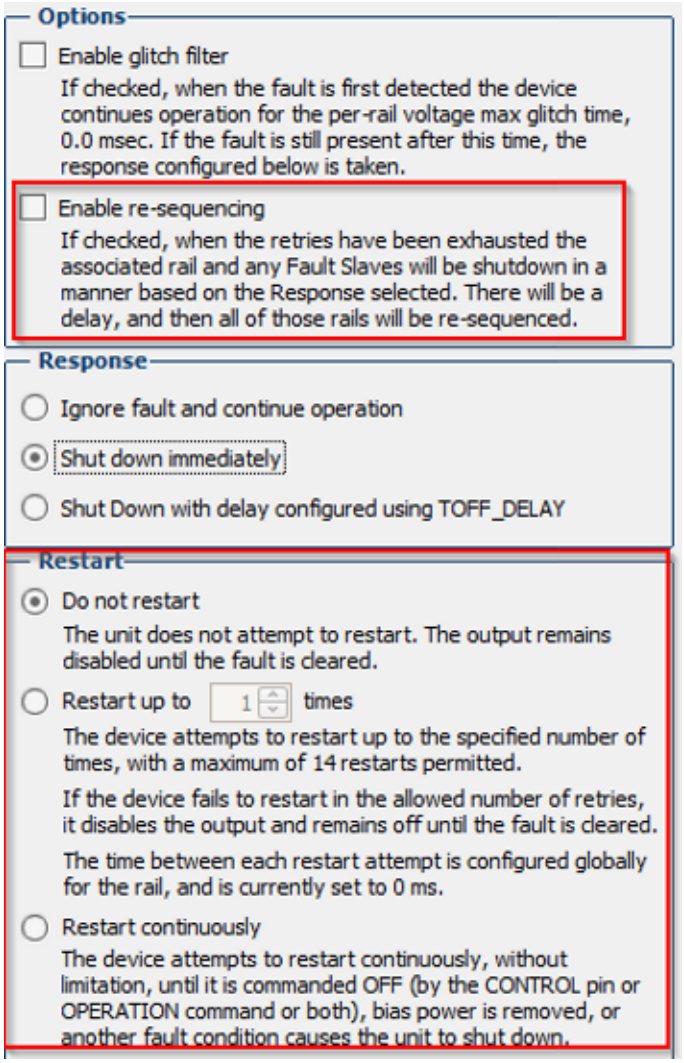

Figure 20. Restart and Re-Sequencing Options

Figure 21 shows a flow chart of restart and re-sequence.

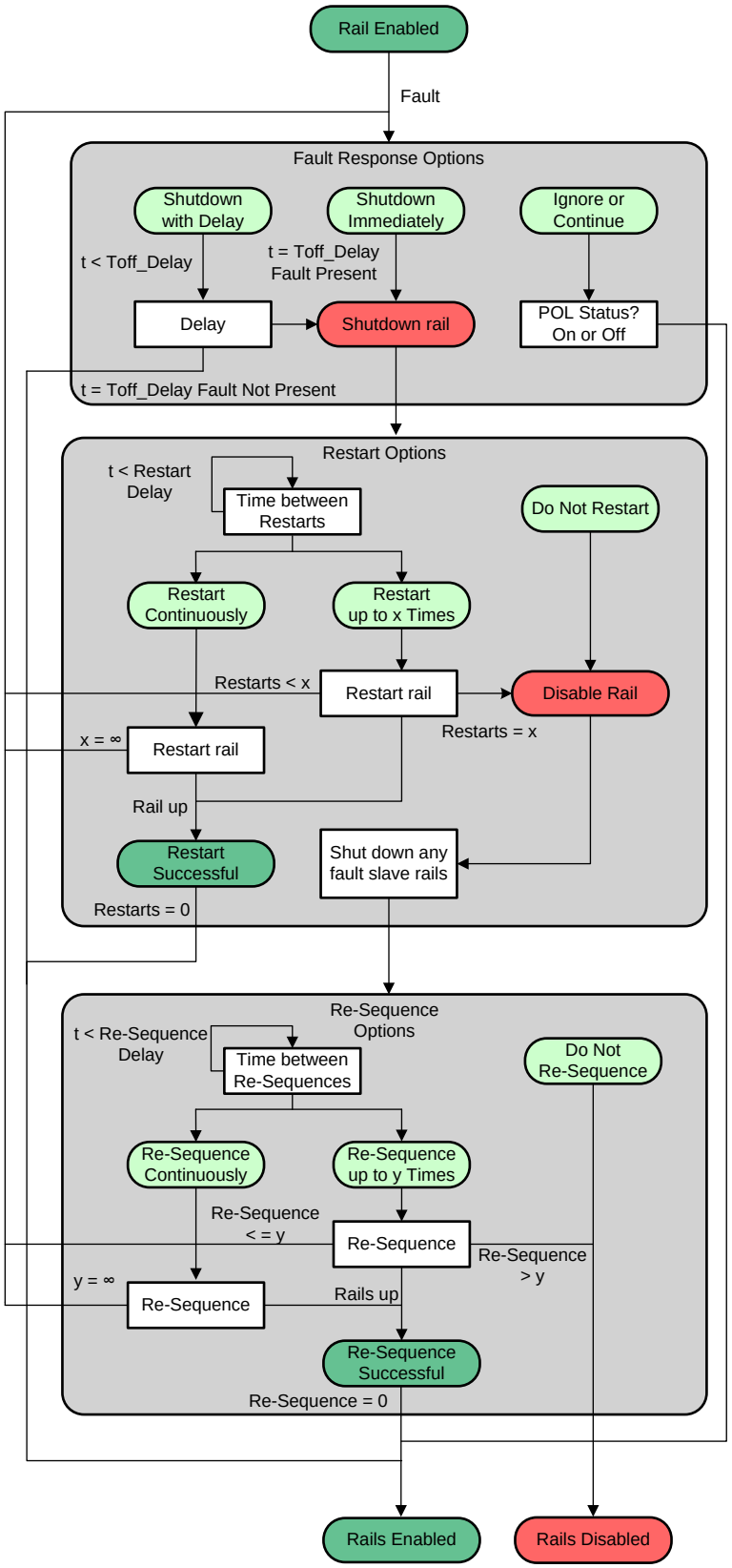

Figure 21. Restart and Re-Sequence Flow Chart

A UV fault can be only set to restart once because the UV fault cannot be detected until the unit has reached regulation; otherwise, the rail would never turn on initially. This means every UV condition is a unique event as the rail would have been regulating before it occurs.

To handle the condition where the rail fails to start after a UV condition, TON_MAX_FAULT_LIMIT should be used (it is listed as Max Turn On beneath the Turn On Delay setting in the GUI). This sets the maximum amount of time allowed before the rail is expected to be within regulation. If this is exceeded, then a fault is declared and a separate fault response handles this condition (with a different name, Time On Max). Time On Max can be set to trigger multiple times.

4.10 How can users re-sequence all rail after they are off due to fault response when the rails' ON_OFF_CONFIG are set to AUTO mode?

For rails that are off due to fault response and ON_OFF_CONFIG set to AUTO, the only way to power them on again is to enable the re-sequence option at its fault response. With the re-sequence option, the faulted rail plus its fault shutdown slaves is re-sequenced when the retry account of the faulted rail is exhausted. The re-sequencing times and intervals are set through Global Configuration → Misc_config as shown in Figure 22.

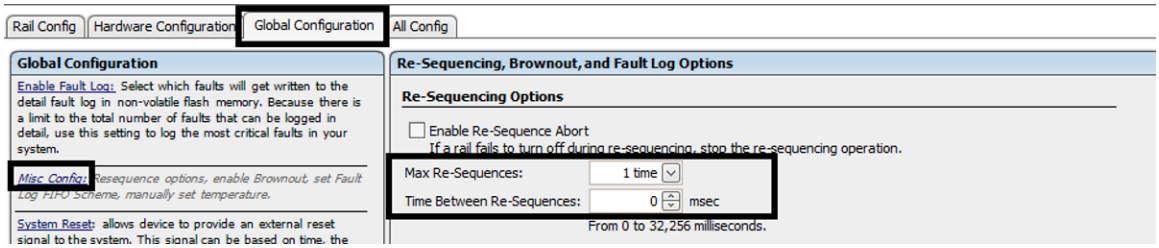

Figure 22. Setting Re-Sequencing Times and Intervals

4.11 When is the retry count reset?

The retry count is reset whenever the rail stays above the POWER_GOOD threshold for a TON_MAX_FAULT_LIMIT amount of without having a glitch. If TON_MAX_FAULT_LIMIT is set to 0, 4 seconds are used for the time.

4.12 When is the re-sequencing count reset?

The re-sequencing count is reset when all re-sequencing rails stay above the POWER_GOOD threshold for more than 1 second. Once the re-sequencing count is exhausted, the device does not perform any re-sequencing until a reset or power cycle is triggered.

4.13 Why do rails not re-sequence after restart is exhausted?

Re-sequence cannot start until all re-sequencing rails are below the POWER_GOOD_OFF threshold. Make sure that all re-sequencing rails are off properly.

4.14 Why does the time interval between two re-sequences longer than the value set in "time between re-sequences"?

The "time between re-sequences" only starts to count when all rails are below their POWER_GOOD_OFF thresholds. Because the monitored rail voltage may take a relatively long time to decay, it is expected to see the time interval between the previous shutdown and the following re-sequence longer than the value set in the time between re-sequences.

4.15 Why does the turnon and turnoff sequence or re-sequence hang or take very long to complete?

If the sequence hangs, check at which step the sequence is stuck, then find the cause. For example, if Rail 3 fails to start, first check its EN pin status. If the EN pin is asserted, check the POL controller. If the EN pin is not asserted, check the dependency of Rail 3, such as its GPI dependency, and whether the prior rail has reached Power Good state, and so forth.

If the sequence takes very long to complete, check if there is gating condition in the dependencies. For example, the whole sequence may be waiting for a GPI dependency for a long time, or waiting for a rail to ramp up or discharge output voltage to its Power Good On/Off thresholds for a long time.

You can set Sequencing Timeout parameters to decide the maximum wait time for dependencies to be met. The timer starts when the rail is instructed to change state (from On/Off Config or Fault Shutdown Slave). When timeout period is expired, the rail takes an action regardless.

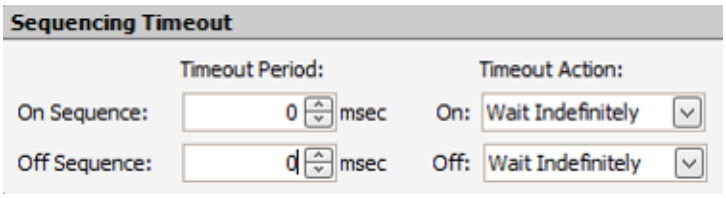

Figure 23. Sequencing Timeout

4.16 Can the logical GPO (LGPO) output be used to turn on and off the power rail?

Yes, but it is not recommended to do so. A rail EN pin has many features that logic GPO pins do not have, such as enable and disable with different dependencies, delays, fault responses, re-sequences, fault logging, and so forth. Fault shutdown with a rail is also faster than using an LGPO.

4.17 Why are the rails off immediately instead of following the sequencing settings (dependencies, TOFF delay) when toggling the CONTROL?

There is a CONTROL signal turnoff configuration: Soft Off (with Sequencing) or Immediately Off. You can use this configuration to choose a desirable behavior.

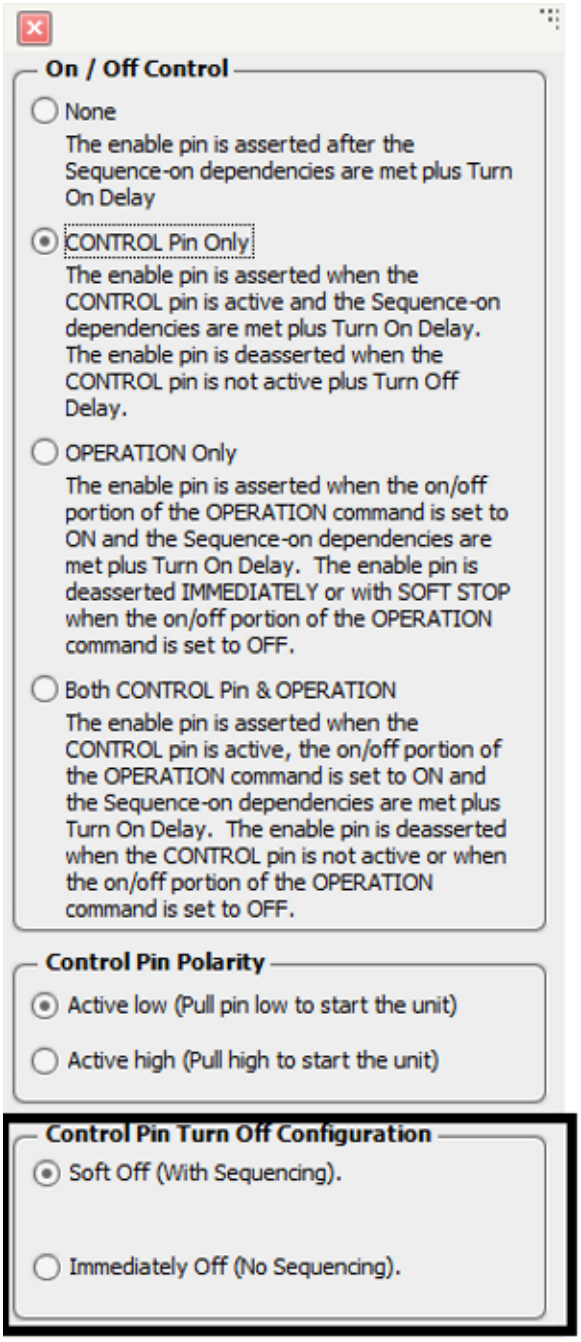

Figure 24. Control Pin Turnoff Configuration

4.18 Why are the fault shutdown slave rails not off immediately?

The faulted rail is configurable to either shutdown immediate or soft off. This is set at the fault response as shown in Figure 25.

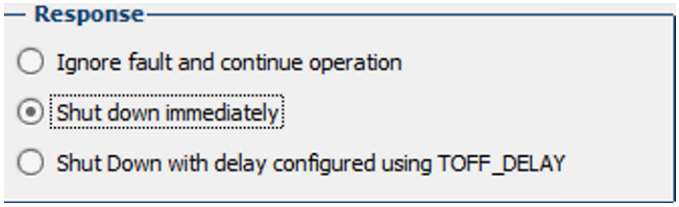

Figure 25. Configuring Shutdown Settings

For all fault shutdown slave rails, they are sequenced off based on their own sequencing configurations such as dependencies and delays.

5 UCD90xxx Fusion GUI FAQs

5.1 Where is the latest version of the TI Fusion GUI?

Make sure to use the latest version of the TI Fusion Designer GUI (Fusion Digital Power Designer). TI also provides an "Alert Me" feature to notify the registered users when a new GUI version is available to download.

Figure 26. Alert Me Feature

5.2 Why are the settings not saved after device reset?

You must click **Write to Hardware** first followed by **Store RAM to Flash** to store all settings into non-volatile memory. Otherwise, the settings are not saved.

5.3 Why is $\overline{\text{POWER\_GOOD}}$ flag set?

It means that at least one monitored rail does not reach its POWER_GOOD_ON threshold. Check the EN signals and POLs to find the root cause.

5.4 Why is the invalid logs flag set?

An invalid log bit is set when the device detects corrupted log in the flash. The log is stored in an area in the flash. It has a checksum to protect its integrity (do not confuse it with data flash checksum, which protects the integrity of configuration data). Every time a new log is recorded, the checksum needs to be updated. In the event power is disconnected, the UCD90xxx device may not be able to finish writing new logs before the 3.3-V supply vanishes; therefore, the next time UCD90xxx powers up, the log content does not match its checksum, and the invalid log flag is set to warn you.

To avoid an invalid log, there are two approaches:

1. Properly sequence down all the rails before disconnecting power.
2. Implement a brownout circuit as described in the device datasheet.

The brownout circuit includes a Scotty diode and several capacitors to sustain a 3.3-V supply voltage for the UCD90xxx to finish fault logging in the event of power disconnection.

The UCD90240, UCD90320, and UCD90320U devices have a dedicated checksum for each log entry. As a result, power disconnection does not generate this flag.

5.5 Why is the invalid logs flag not cleared?

The Invalid Logs flag is cleared only on the next boot up if the device does not see any corrupted logs. The Hardcoded Parms flag has the same behavior. This flag is set when the device detects that your configuration data is corrupted and is only cleared when the device detects a complete your configuration on the new boot up.

5.6 Why is a fault logged even if the fault is ignored by a glitch filter?

A glitch filter only applies to a fault response. The fault itself is still logged.

5.7 What happens when the fault log is full?

There are three configurable behaviors when the fault log is full:

1. Stop logging new faults.
2. Use FIFO mode, the oldest log is overwritten by the newest log.
3. Keep the first half number of logs unchanged; only the second half logs are in FIFO mode (not available on the UCD90320U, UCD90320 and UCD90240 devices).

5.8 Why does the GUI not show the correct name of the rails, GPIs, or GPOs when the board is connected to a new computer?

The names of rails, GPIs, and logic GPOs are stored in the local PC instead of the respective devices. Therefore, when you connect the boards to a new PC, the names are not transferred automatically. You have to import the project with name option enable to get transfer done as shown in Figure 27.

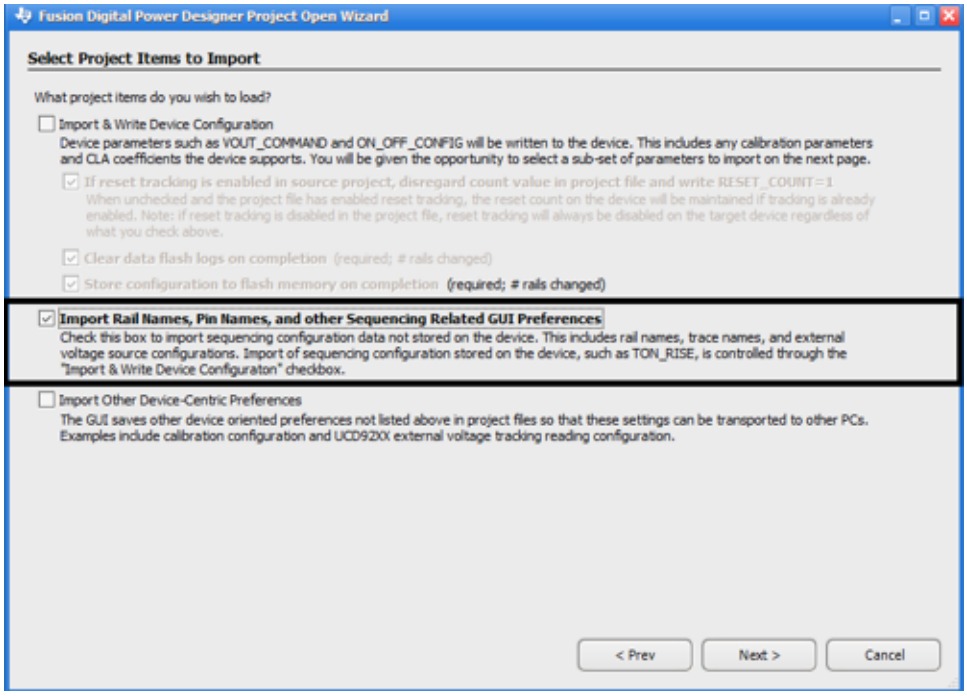

Figure 27. Importing Rail Names, Pin Names, and Other Sequencing Related GUI Preferences

5.9 What are the fast GPIs and how to enable them?

The UCD90xxx family has eight interruptible GPIs (GPIO1 through 4, 12, 16, 17, and 18) that can achieve a faster response. These GPIO pins must be defined as shown in Table 3 to have the interrupt feature enabled.

Table 3. Interruptible GPIO Pins(1)

GPIO PIN	GPIO1	GPIO2	GPIO3	GPIO4	GPIO13	GPIO16	GPIO17	GPIO18
GPI	GPI1	GPI2	GPI3	GPI4	GPI5	GPI6	GPI7	GPI8

(1) All GPIs in the UCD90240, UCD90320 and UCD90320U devices are fast GPIs.

5.10 Can less than three GPIs be used for PIN SELECTED RAIL STATE (PSRS)?

Yes, PSRS supports up to three GPIs. When the number of GPIs used for PSRS is less that the configured GPIs, enable all supported states based on the number of configured GPIs and configure the state based on the number of GPIs for PSRS. For example, three GPIs are configured, but only the first GPI is used for PIN PSRS. You have to enable all eight states, configure all even states the same as state 0, and configure all odd states the same as state 1. Therefore, the changes on the second and third GPIs do not affect desired function.

5.11 Why does GUI show 0 V while the real measuring result is 2.1 V?

Each monitor rail has its Vout Exponent, which limits the maximum voltage the device shall return. Make sure that the Vout Exponent is set to a proper value for the application. When the "Set for Me" box is checked, the GUI sets the Vout Exponent automatically based on the your inputs.

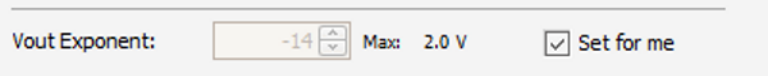

Figure 28. Vout Exponent

5.12 Why does GUI show 0.2 V while the real measuring result is 0 V?

Some monitor pins of the UCD90xxx have a minor limitation on the lower range of monitoring—they cannot detect voltage below 0.2 V. Anything below that is reported as 0.2 V. Take this into consideration when the pins are used.

Table 4. Limited Monitor Pin

PARAMETER	UCD9012x or UCD90910	UD90160x	UCD9090x
Pin number	50	50	37
	52	52	N/A
	54	54	N/A
	56	N/A	N/A

6 UCD90xxx Margining FAQs

6.1 Does the UCD90xxx family support voltage margining?

Yes. The UCD90xxx family employs a PWM method to do voltage margining. Compared to the DAC method, the PWM method provides better resolution and current driving capacity. When not used by a margining function, PWM pins can be use as GPIO pins (except UCD90240), which provides additional flexibility for you to allocate pin resources.

6.2 When does the UCD90xxx start rail margining?

The UCD90xxx only starts the margining when the desired rails reach its POWER_GOOD threshold and the margining command is enabled.

6.3 Are there any application notes for margining function?

Yes. TI has provided the following application notes to design voltage margining:

- *Design Voltage Margining Circuit for UCD90xxx Power Sequencer and System Manager* (SLVA845)
- *UCD90xxx Voltage Margining Circuit Design Tool* (SLVC676)

7 UCD90xxx Hardware FAQs

7.1 Is the PMBus connector for the Fusion GUI still required if a PMBus host is already available on the board?

Although it is optional, it is always recommended to have a PMBus connector to connect the Fusion GUI. You may need this connector to program the device. It also comes in handy when you need to debug the board and adjust configurations. Unlike analog devices, the UCD90xxx provides full visibility to the internal status of the device. It helps you analyze complex sequencing logics and identify problems quickly.

7.2 Can the RESET pin be used to sequence off rails?

No. Unlike analog devices, RESET pin of the UCD90xxx resets its digital core abruptly. All GPIO pins immediately enter a high-impedance state, and all FPWM pins immediately enter a drive-low state. There is no sequencing. Also, upon releasing the RESET pin, the device is subject to at least a 20-ms initialization time before it starts to function.

The correct method to sequence off rails is to use the CONTROL pin. A delay and dependency configuration of each rail is observed, and there is no initialization time delay at the following sequence on.

The datasheet recommends pulling the RESET pin to V33D with a local decoupling capacitor. This way, the RESET pin voltage ramps up with V33D. Also, though unnecessary, you can opt to keep the RESET pin low by an external source until after the V33D supply has completed ramping up. However, avoid pulling RESET pin low once the device is in operation. Glitch is not allowed on the RESET pin.

7.3 Are the decoupling capacitors needed on MON pins?

Decoupling capacitors on MON pins are recommended. MON pins are ADC channels. There is no internal filtering to average the ADC samples. Therefore, decoupling capacitors are used to remove rail ripple voltage. Also, if resistor dividers are used to scale down the voltages, adding decoupling capacitors can reduce input impedance and improve ADC sample accuracy. 100 nF to 10 nF are good numbers for decoupling capacitor value.

7.4 What are the states of IO pins when the UCD90xxx is under reset?

All IO pins of the UCD90320U, UCD90320, and UCD90240 devices are Hi-Z state. For the rest of the UCD90xxx family, FWPM pins are in a driven-low state and the rest IOs are at Hi-Z state.

7.5 Are pulldown resistors needed on EN pins?

Pulldown resistors on EN pins are needed in some situations. When the UCD90xxx device is in reset or in initialization, GPIO pins are in high-impedance state (FPWM pins are in drive-low state). If the EN pin of the POL controller has internal pullup, the EN pins of the UCD90xxx (GPIO pins in Hi-Z state) are unable to keep the POL converters off. If this is the case, pulldown resistors are needed to keep EN pins low when UCD90xxx is in reset or initialization.

7.6 Is an RC filter needed for a V33A rail?

V33A supplies power to voltage reference and ADC blocks inside the device. A large ripple on V33A can affect the voltage reference. It is recommended using an RC filer to separate V33D and V33A in order improve V33A quality. The recommended values are 5 Ω and 4.7 μF. See application information in the device datasheet for details.

7.7 Why is there leakage current on pins when the UCD90xxx is not powered up?

The UCD90xxx device has an internal ESD diode to protect on all IO pins and MON pins. When the UCD90xxx device is not powered by a 3.3-V supply, if a positive voltage is applied to a pin, the ESD diode is forward biased and a leakage current may be observed. If such a scenario can be anticipated, resistors should be used to limit the current flow below 1 mA.

Revision History

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

Changes from A Revision (September 2018) to B Revision **Page**

- Edited application report for clarity. 2
- Changed Valid Address rows in Table 1. 13

Changes from Original (March 2017) to A Revision **Page**

- Added reference to UCD90320U device in Section 2.4 3
- Added reference to UCD90320U device in Section 3 11
- Added reference to UCD90320U device in Section 3.4 13
- Added reference to UCD90320U device in Section 3.8 14
- Added reference to UCD90320U device in Section 4.6 18
- Added reference to UCD90320U device in Section 5.4 25
- Added reference to UCD90320U device in Table 3 27
- Added reference to UCD90320U device in Section 7.4 29

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, 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 (www.ti.com/legal/termsofsale.html) 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.

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